On Tracking the Behavior of an Output-Queued Switch Using an Input-Queued Switch

Amir Gourgy, *Member, IEEE*, Ted H. Szymanski, *Member, IEEE*, and Douglas G. Down, *Senior Member, IEEE*

***Abstract*—We address the problem of fair scheduling of packets in Internet routers with input-queued (IQ) switches and unity speedup. Scheduling in IQ switches is formulated as *tracking* the behavior of an output-queued (OQ) switch that provides optimal performance. We present the notion of "lag" as a performance metric that measures the difference between a packet's departure time in an IQ switch over that provided by an OQ switch. We prove that per-packet mean lag is bounded for a maximum weight-matching scheduling policy that uses lag values for its weights and derive a bound on the mean lag value using a Lyapunov function technique. Furthermore, we propose a simple heuristic tracking scheduling policy and evaluate its performance by simulation.**

***Index Terms*—Input-queueing, Lyapunov functions, packet switch, scheduling, switching.**

I. Introduction

THERE is a tremendous demand for Internet core nodes to provide quality-of-service (QoS) guarantees for multimedia services and to provide high switching capacity that makes use of the virtually unlimited bandwidth of optical fibers. The Internet's success depends on the deployment of high-speed switches and routers that meet these two demands.

On the one hand, the demand for QoS guarantees can be met using output-queued (OQ) switches, which can provide optimal throughput. In addition, much research effort has been devoted to packet scheduling at output ports to support fair bandwidth sharing that provides delay bounds for regulated traffic, considering algorithms such as the weighted fair queueing (WFQ) family (e.g., [1]). However, output queueing for an $N \times N$ switch requires the switching fabric and memory to run up to N times faster than the line rate; unfortunately, for large or high-speed data lines, memories with sufficient bandwidth are not available. On the other hand, the fabric and the memory of an input-queued (IQ) switch need only to run as fast as the line rate. This property makes input queueing very appealing for switches with fast line rates or with a large number of ports. However, IQ switching can suffer from head-of-line (HOL) blocking, which limits the throughput to just 58.6%, if each input maintains a single FIFO queue [2]. One method that has been proposed to reduce HOL blocking is to increase the *speedup* of a switch. A switch with a speedup of S can remove up to S packets from each input and deliver up to S packets to each output within a *time slot*, where a time slot is the amount of time needed to transfer a packet from the input side to the output side of a switch.

A theoretical result [3] established that an $N \times N$ combined input and output-queued (CIOQ) switch with a speedup of two could exactly *emulate* an $N \times N$ OQ switch for any traffic pattern of input cells. Emulation occurs at every time instance if, under identical inputs, both systems produce identical departures. Unfortunately, the complexity of the scheduling algorithm presented in [3] renders OQ switch emulation infeasible (see [4] and [5] for a discussion of the complexity). The tradeoff between the delay and speedup in a CIOQ switch has been analyzed in [6]. Furthermore, Minkenberg [7] has shown that exact emulation of an OQ switch using a CIOQ switch is possible only if the CIOQ switch has infinite output buffers.

Many commercial high-performance switches and routers (e.g., CISCO 12000 [8], BBN [9], Lucent Cajun [10] family, or Avici TSR45000 [11]) use IQ switches. Most of these high-speed switches are built around a crossbar switch that is configured using a centralized scheduler designed to provide high throughput and use a fixed-length cell as a transfer unit. Fixed-length switching technology is widely accepted for achieving high switching efficiency such that variable-length packets are segmented into fixed-length cells at the inputs and are reassembled at outputs. We assume fixed-length cell scheduling for the remainder of this paper.[1]

We consider scheduling policies in an IQ-crossbar switch with a unity speedup. Given that an IQ switch requires at least a speedup of two to exactly emulate an OQ switch [3], an IQ scheduling policy with a unity speedup cannot exactly emulate the behavior of an OQ switch, under all possible traffic patterns. Consequently, we formulate scheduling in an IQ switch as the problem of tracking an OQ switch. We propose the "lag" as a performance metric that measures the difference between a packet's departure time in an IQ switch over that provided by an OQ switch. We present an IQ scheduling policy with unity speedup for which the lag is bounded and derive a bound

Manuscript received May 09, 2007; revised July 02, 2008; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Fumagalli. First published September 25, 2009; current version published December 16, 2009. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Center of Excellence—Communication and Information Technology Ontario (OCE-CITO).

A. Gourgy was with McMaster University, Hamilton, ON L8S 4K1, Canada. He is now with the Research in Motion, Inc., Waterloo, ON N2L 3W8, Canada (e-mail: agourgy@ieee.org).

T. H. Szymanski and D. G. Down are with McMaster University, Hamilton, ON L8S 4K1, Canada (e-mail: teds@mail.ece.mcmaster.ca; downd@mcmaster.ca).

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

Digital Object Identifier 10.1109/TNET.2009.2014948

[1]The words *packet* and *cell* are used interchangeably for the remainder of this paper.

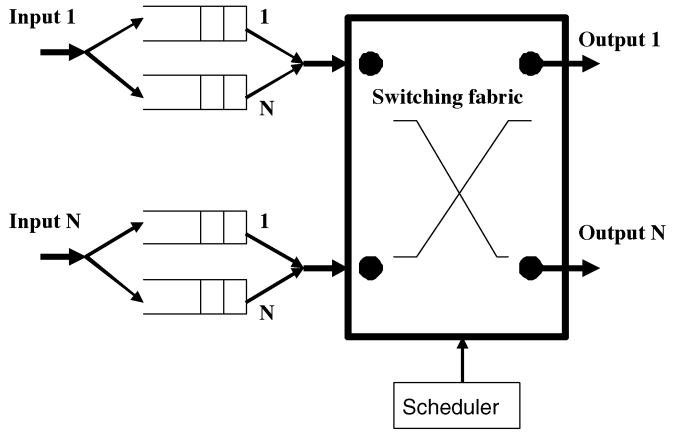

Fig. 1. Logical structure of an IQ switch.

on the mean lag value per packet. Furthermore, we propose a simple heuristic tracking scheduling policy and evaluate its performance by simulation. In this paper, we describe the case of tracking an OQ switch implementing a FIFO scheduling policy.

This paper is organized as follows. Section II formulates scheduling in an IQ switch with unity speedup as tracking the behavior of an OQ switch. Section III provides motivation for tracking the behavior of an OQ switch and discusses related work. In Section IV, we present two scheduling policies for tracking the behavior of an OQ switch. First, we present a scheduling policy called *maximum weighted lag* (MWL). We prove that the mean lag value is bounded for MWL and derive an upper bound on its value using a Lyapunov function technique. The MWL scheduling policy has a high implementation cost, but serves as a solid base for developing other practical scheduling policies that approximate its performance. Consequently, we present a simpler heuristic tracking policy that can be readily implemented in hardware. The performance of the proposed scheduling policies is evaluated by simulation in Section VI. Section VII provides our conclusions.

II. Problem Formulation

We consider an $N \times N$ OQ switch that uses scheduling policy Π_{OQ} and an IQ switch with unity speedup that uses scheduling policy Π_{IQ}. For an $N \times N$ switch, we use the following notational conventions: i an input, $1 \leq i \leq N$; j an output, $1 \leq j \leq N$; $Q_{i,j}$ is the virtual output queue (VOQ) at input i and buffers cells destined for output j; $\mathrm{HOL}_{i,j}$ is the head-of-line cell at $Q_{i,j}$.

In this paper, we consider only the case of $\Pi_{OQ} = \mathrm{FIFO}$. The architecture of our IQ switch is shown in Fig. 1. We use VOQs at each input port of the switch and a crossbar as the switching fabric.

Let the average cell arrival rate at input i for output j be λ_{ij}. We assume that incoming traffic is admissible; that is, $\sum_{i=1}^{N} \lambda_{ij} < 1$, and $\sum_{j=1}^{N} \lambda_{ij} < 1$. The arrival process is identical to both switches. The goal is to find a scheduling policy Π_{IQ} that tracks the behavior of the OQ switch as close as possible, where we define what tracking means more precisely after introducing some definitions. Given that an IQ switch requires at least a speedup of two to exactly emulate an OQ switch [3], a scheduling policy for an IQ switch with a unity speedup cannot exactly emulate the behavior of an OQ switch, under all possible traffic patterns. In general, cells arriving to the IQ switch implementing Π_{IQ} will depart at some later time than the OQ switch implementing Π_{OQ}. Consequently, we say that an IQ switch implementing Π_{IQ} lags the behavior of the OQ switch implementing Π_{OQ}.

A. Definition of Terms

Here we make precise some of the terminology used throughout this paper.

Definition 1: Arrival Rate Matrix (λ): $\lambda \equiv [\lambda_{ij}]$, where the arrival process is assumed to be admissible and stationary; that is, $\sum_{i=1}^{N} \lambda_{ij} < 1$, $\sum_{j=1}^{N} \lambda_{ij} < 1$, $\lambda_{ij} \geq 0$. The associated arrival rate vector $\underline{\lambda} \equiv (\lambda_{1,1}, \ldots, \lambda_{1,N}, \ldots, \lambda_{N,1}, \ldots, \lambda_{N,N})^T$. The arrival process at each input port i is assumed to be an i.i.d. process of fixed-size cells. At the beginning of each slot, either zero or one cell arrives at each input port. Virtual output queueing is used such that when a cell arrives at time slot n for output j at input i, it is placed in queue $Q_{i,j}$.

Definition 2: Ideal Departure Time (IDT): The ideal departure time for a cell c, $\mathrm{IDT}(c)$, is the time slot at which c will depart from an OQ switch using Π_{OQ}.

Definition 3: Actual Departure Time (ADT): The actual departure time for a cell c, $\mathrm{ADT}(c)$, is the time slot at which c departs from the switch under consideration (i.e., IQ implementing Π_{IQ}).

Definition 4: Cell Lag (CL): The cell lag for a cell c, $\mathrm{CL}(c)$, is the difference between the IDT and the ADT. Precisely

$$\mathrm{CL}(c) \equiv \begin{cases} \mathrm{ADT}(c) - \mathrm{IDT}(c), & \mathrm{ADT}(c) > \mathrm{IDT}(c) \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

In addition, we define the cell lag for a cell c given the current time slot n, $\mathrm{CL}(c|n)$, as the difference between the IDT and the current time slot. Precisely

$$\mathrm{CL}(c|n) \equiv \begin{cases} n - \mathrm{IDT}(c), & n > \mathrm{IDT}(c) \\ 0, & \text{otherwise.} \end{cases}$$

The goal of a scheduling policy can be characterized by any statistical metric that attempts to minimize the cell lag. For example, in Section IV-B, we present a scheduling policy that minimizes the mean lag value per packet.

Note that according to (1), the lag is nonnegative and generally a cell's ADT is greater than its IDT. However, a cell may occasionally depart from an IQ switch earlier than an OQ switch. For example, consider a 2×2 switch at a specific time slot such that the two most lagging cells for its outputs (e.g., outputs 1 and 2) reside at the same input port (e.g., input 1). Because the scheduling policy can transfer at most one cell from each input port (e.g., input 1), another cell with an IDT in the future can be selected from the other input port (e.g., input 2) to improve the throughput.

III. Motivation and Related Work

In an OQ switch, arriving packets are immediately available at the outgoing link. Consequently, the only shared resource in an OQ switch is the outgoing link for which packets contend for access (output contention). In an IQ switch, packets are queued

at the input port of the switch, and they must first contend for access to the switch fabric (input contention) before contending for the outgoing link. That is, in an IQ switch, there are two shared resources: the switch fabric and the outgoing link.

All present IQ scheduling policies resolve input and output contention using heuristics such as using a round-robin scheme at both the input and output to solve the contention fairly [12] or using the packet's age (i.e., time in the switch) to resolve contention [13]. All these schemes can be seen as an approximation to the ideal case of an OQ switch, where all of the outgoing links are independent and packets are served fairly in each outgoing link. That is, by tracking the behavior of an OQ switch and minimizing the lag, we automatically resolve input and output contention in a fair manner and eliminate any starvation problem of inputs that other scheduling policies have to carefully handle.

Tabatabaee *et al.* [14] consider the related problem of packetizing arbitrary fluid policies in an $N \times N$ crossbar switch using FIFO virtual output queues. They define trackable fluid policies such that for each pair of input and output ports, at each time-step, the cumulative number of packets sent between these ports differs from the cumulative fluid scheduled between these ports by less than 1. They prove that a tracking policy always exists for the special case of a 2×2 switch, provide an example for a 3×3 switch where a nonanticipative tracking policy does not exist, and propose several heuristics for packetizing fluid policies on general $N \times N$ switches. Rosenblum *et al.* [15] further extend the results in [14] by relaxing the tracking constraint such that the cumulative difference in the number of packets sent using the fluid and packetized policies can be more than one packet. This paper's results are compatible with the seminal results of [14]. Our work differs from [14] and [15] in that we track the precise packet departure sequence in an OQ switch rather than the aggregate rate provided by a fluid scheduling policy in an IQ switch, which does not necessarily track an OQ switch. For two scheduling policies to provide the same service rate, they need to serve only the same number of packets per link, rather than tracking the precise packet departure order, which can be different between the two scheduling policies. This issue is discussed in more detail in Section IV-B.

IV. Tracking Scheduling Policies

A. Computing the Ideal Departure Time

For $\Pi_{\mathrm{OQ}} = \mathrm{FIFO}$, arriving cells at the IQ switch can be immediately assigned an IDT using a simple parallel prefix circuit [16] (i.e., a ranker circuit). Let $N_j(n)$ be the number of cells in the OQ switch destined to output j at time slot n. The IQ switch uses N rankers such that each ranker calculates the number of cells present in the OQ switch being tracked. At the beginning of each time slot, n, the number of packets in the OQ switch is computed as follows:

$$N_j(n) \equiv \begin{cases} N_j(n-1) - 1, & N_j(n-1) > 0 \\ 0, & N_j(n-1) = 0. \end{cases}$$

Note that the subtraction of one in the previous equation accounts for one (cell/time slot) departure in the OQ switch. For every new cell c arriving at time slot n destined to output j, ranker j assigns a numeric rank (from 1 to N) in a linear[2] order to packets arriving for output port j. The IDT of each cell is equal to its numeric rank plus $N_j(n-1)$, and $N_j(n-1)$ is updated accordingly. The complexity of computing the $\mathrm{IDT}(c)$ in hardware using a parallel prefix computation is $\Theta(\log N)$ depth and $\Theta(N)$ circuit size, expressed in terms of binary operators [16].

B. Maximum Weighted Lag Scheduling Policy

MWL is based on the implementation of a maximum bipartite weight-matching algorithm (MWM) [17]. A maximum weight matching on a bipartite graph with weighted edges is defined as a set of edges between input and output nodes with the maximum total weight among all possible sets satisfying the constraint that any input node is matched to at most one output node. At every time slot n, we associate a weight $W_{i,j}$ to every $Q_{i,j}$ such that $W_{i,j} = \mathrm{CL}(\mathrm{HOL}_{i,j}|n)$; that is, $W_{i,j}$ is the lag of an HOL packet in $Q_{i,j}$. The maximum weighted lag scheduling policy finds a matching M that maximizes $\sum_{(i,j)\in M} W_{i,j}$ and can be found by solving an equivalent network flow problem [17]. The sequential run time complexity of MWM is $\Theta(N^3)$ [17].

Previous work on MWM considered only the weight to be either some function of the occupancy of the VOQs (i.e., the number of packets in each VOQ) or the waiting time of the cell at the HOL of each VOQ (e.g., [13] and [18]–[21]). Consequently, these algorithms do not necessarily track the behavior of an OQ switch, and a cell's departure time may deviate from the ideal case under nonuniform traffic. In addition, using the occupancy of the VOQs as the edge weight can lead to starvation of certain inputs [13].

Because MWL computes the matching with the maximum possible total weight during every time slot, it aims at minimizing the mean lag. Although this algorithm is too complex to implement in practice, it serves as a reference model for which other approximation algorithms are developed.

The stability of maximum weighted matching scheduling policies is a well-studied problem in the literature. McKeown *et al.* [13] proved the stability of longest queue first (LQF) and oldest cell first (OCF) maximum weight matching for all admissible independent identically distributed (i.i.d.) arrival processes using a Lyapunov function technique; Dai and Prabhakar [19] extended the results to prove the stability of a maximum weight matching algorithm under any admissible arrival processes using fluid model techniques.

Although the results for the fluid model technique established in [19] could easily be used to prove the stability of MWL, they cannot be further extended to derive a bound on the expected lag value. Consequently, we use a Lyapunov function technique that allows us to derive a bound on the expected lag value as described in Appendix II.

Theorem 1: A FIFO tracking policy that uses the MWL as the scheduling policy is stable (achieves 100% throughput) for all admissible i.i.d. arrival processes.

[2]We investigated diverse ordering schemes (e.g., round-robin, linear, etc.) for assigning IDT to simultaneous cell arrivals destined to the same output and found it to have an insignificant effect on the results.

Proof: The proof is given in Appendix I. The proof of Theorem 1 for the stability of MWL uses similar techniques to the proof for stability of OCF scheduling presented in [13].

Having established the stability of the MWL scheme in Theorem 1, we now proceed to bound the maximum lag experienced by an individual cell. A methodology for deriving bounds on the cell delay and queue size is described in [21]. In [13], it was shown that LQF could potentially lead to starvation. Longest Port First (LPF) was proposed in [22] and was shown by simulation to provide better performance than LQF and OCF, but it is possible to construct a traffic pattern that leads to starvation for LPF [23]. All the previous results are applicable to stability in a single node (switch). The problem of scheduling a network of input-queued switches is considered in [24], and it is shown that both the LQF and LPF scheduling policies can be unstable for a fixed traffic pattern in a simple network of eight IQ switches.

Before we proceed, we need the following definitions in addition to the definitions used for the proof of Theorem 1 in Appendix I.

Definition 5: L_1 Norm: Given a vector $\underline{Z} \in \mathbb{R}^{N^2}$, the norm $||\underline{Z}||_1$ is defined as

$$||\underline{Z}||_1 = \sum_{k=1}^{N^2} |z_k|.$$

Definition 6: Input-Output Norm: Given a vector $\underline{Z} \in \mathbb{R}^{N^2}$, $\underline{Z} = \{z_k, k = (N-1)i + j, i, j = 1, \ldots, N\}$, the norm $||\underline{Z}||_{\mathrm{IO}}$ is defined as

$$||\underline{Z}||_{\mathrm{IO}} = \max_{j=1,\ldots,N} \left\{ \sum_{k=1}^{N} |z_{N(k-1)+1}|, \sum_{l=1}^{N} |z_{N(j-1)+l}| \right\}.$$

$||\underline{Z}||_{\mathrm{IO}}$ takes the maximum of the sum of quantities related to all the queues referring either to the same input or to the same output. For example, the traffic arrival vector is admissible if and only if $||\underline{\lambda}||_{\mathrm{IO}} < 1$.

Definition 7: Let $\underline{L}(n)$ be the lag vector at time slot n such that

$$\underline{L}(n) \equiv (L_{1,1}(n), \ldots, L_{1,N}(n), \ldots, L_{N,1}(n), \ldots, L_{N,N}(n))^T$$

where $L_{i,j}(n)$ is the lag of $\mathrm{HOL}_{i,j}$ at time slot n.

Theorem 2: A bound on the mean lag, $E[||\underline{L}(n)||_1]$, using a MWL scheduling policy under any admissible i.i.d. arrival process is given by

$$E\left[||\underline{L}(n)||_1\right] \leq \frac{N^3 + 3N^2||\underline{\lambda}||_1}{2\left(1 - ||\underline{\lambda}||_{\mathrm{IO}}\right)}.$$

Proof: The proof is given in Appendix II.

We emphasize that the bound in Theorem 2 is a much stronger property than bounding the average packet delay in an IQ switch over that in an OQ switch. Not only does Theorem 2 provide a bound on the additional mean delay for all packets departing an IQ switch using MWL over an OQ switch, it also applies to any individual packet departing the IQ switch. Specifically, Theorem 2 provides a bound on the difference between the precise packet departure sequence from an IQ switch using MWL over that provided by an OQ switch. For example, consider an IQ scheduling policy that periodically serves the same number of packets per output port as an OQ switch over a time interval larger than the corresponding time interval in an OQ switch. For all admissible traffic, this behavior would imply a bounded per-packet average delay compared to an OQ switch, but it does not imply the property of Theorem 2. This behavior occurs because each packet's departure order could be different from the IQ scheduling policy compared to the OQ scheduling policy. The key difference lies in the lag definition such that a packet departing ahead of its time would have a zero lag. Observe that if a negative lag was allowed, then the mean lag value becomes the additional mean delay in an IQ switch over that in an OQ switch as packets departing ahead of their IDT (negative lag) would offset packets departing after their IDT (positive lag). Furthermore, bounding the mean delay in an IQ switch over that in an OQ switch requires only knowledge about the average service rate per output port in both switches rather than the precise packet departure sequence from each switch.

V. Iterative Lag Scheduling Policy

Iterative lag (iLag) is a simple heuristic based on *maximal matching*. A maximal matching algorithm is one that adds connections incrementally, without removing connections made earlier. iLag can be implemented using an arbiter at each input and output port using a request–grant–accept paradigm. Initially, all input and output arbiters are unmatched, then in each iteration:

1) Request: Each unmatched input sends a request to every unmatched output for which it has a queued cell.
2) Grant: If an unmatched output receives any requests, it chooses the request with the most lagging cell and sends a grant to this input.
3) Accept: If an unmatched input receives any grants, it chooses the grant for its most lagging cell and sends an accept signal to this output.

The input and output arbiter are considered matched. The algorithm executes until either no more matches can be made or a fixed number of iterations are performed.

VI. Simulation Results

The average cell delay and $E[||\underline{L}||_1]$ of MWL and iLag are evaluated by simulation for a 16×16 switch and compared to LPF [22], LQF [13], islip [12], and PIM [25]. All simulations were performed with 99% confidence and 1% accuracy. iLag, islip, and PIM were executed with four iterations. Bernoulli traffic distribution is used for performance evaluation.

A. Bernoulli Traffic Distribution

For Bernoulli i.i.d. distribution, we use three traffic models: uniform, log diagonal, and diagonal arrival patterns. Let $|k| = (\mathrm{k \bmod N})$.

1) Uniform: $\lambda_{i,j} = \rho/N \ \forall i, j$, where $N = 16$ is the size of the switch.
2) LogDiagonal: $\lambda_{i,j} = 2\lambda_{i,|j+1|}$, and $\sum_i \lambda_{i,j} = \rho$. For example, the distribution of the load at input 1 across all outputs is $\lambda_{1,j} = 2^{N-j}\rho/(2^N - 1)$. This arrival pattern is more skewed than uniform loading.

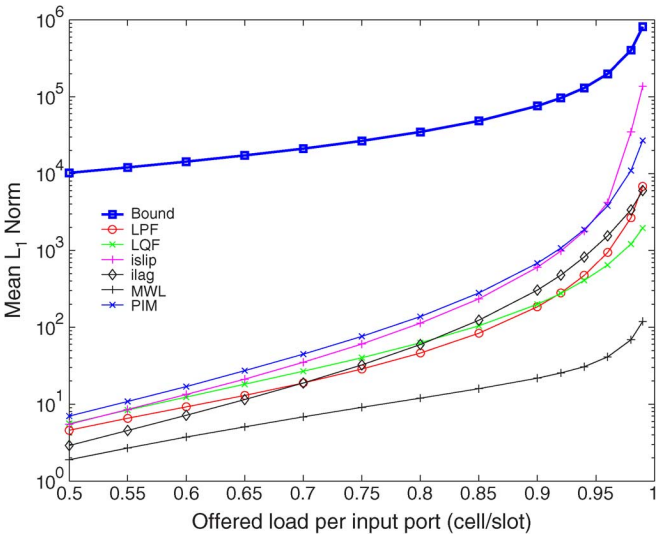

Fig. 2. $E[||\underline{L}||_1]$ versus offered load for uniform Bernoulli i.i.d traffic.

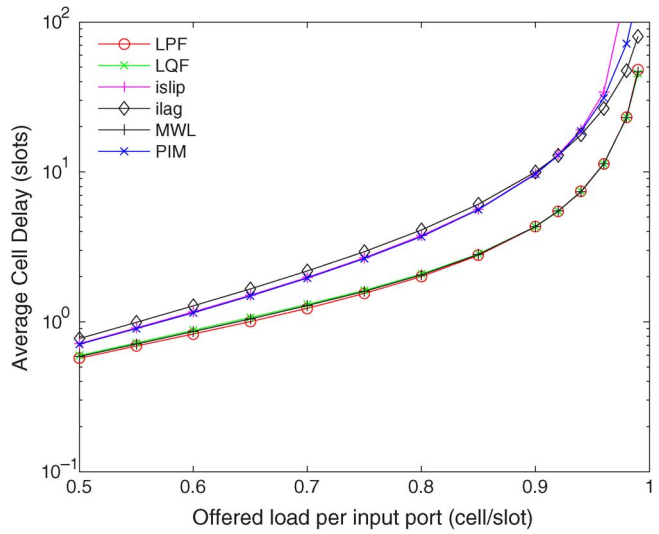

Fig. 3. Average cell delay versus offered load for uniform Bernoulli i.i.d. traffic.

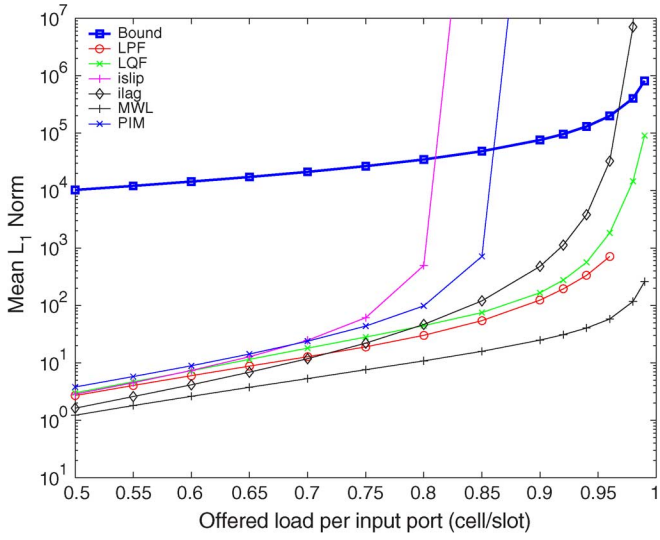

Fig. 4. $E[||\underline{L}||_1]$ versus offered load for log diagonal traffic.

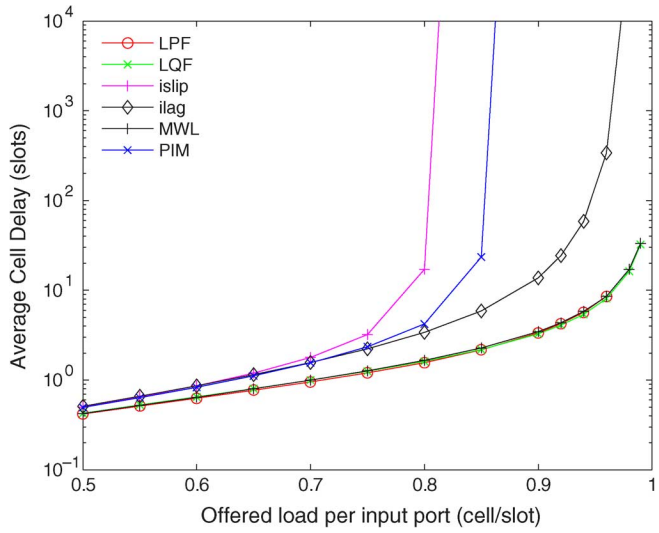

Fig. 5. Average cell delay versus offered load for log diagonal traffic.

3) Diagonal: $\lambda_{i,j} = 2\rho/3$, $\lambda_{i,|i+1|} = \rho/3\ \forall i$, and $\lambda_{i,j} = 0$ for all other i and j. This is a very skewed loading and is more difficult to schedule than uniform loading.

As shown in Fig. 2, MWL provides the lowest $E[||\underline{L}||_1]$ compared to other maximum weight-matching schemes under uniform Bernoulli arrivals, although all maximum weight-matching schemes have almost the same average cell delay as shown in Fig. 3. The same trend occurs for iLag compared to islip and PIM. In particular, among the maximum weight-matching schemes considered, at a load of 0.9, the MWL scheme has $E[||\underline{L}||_1] = 10$ time slots, whereas the LQF and LPF have $E[||\underline{L}||_1] = 100$ time slots. Among the iterative schemes, iLag has $E[||\underline{L}||_1] = 300$ time slots, whereas the iSLIP has $E[||\underline{L}||_1] = 900$ time slots. In both cases, the use of the lag metric results in significant reductions in the mean lag.

Similarly, under log diagonal traffic, MWL provides the lowest $E[||\underline{L}||_1]$, as shown in Fig. 4, whereas the delay of all maximum weighted-matching scheduling policies is almost identical as shown in Fig. 5.

The same trend occurs for diagonal traffic as shown in Figs. 6 and 7.

VII. Conclusion

IQ switches are commercially used in most Internet routers due to their capability of operating at high line speeds with a lower memory bandwidth requirement than OQ switches. In this paper, we addressed the issue of fair scheduling in Internet routers with IQ switches. We formulated switch scheduling in an IQ switch with unity speedup as tracking the behavior of an OQ switch. By tracking the behavior of an OQ switch, an IQ switch resolves input and output contention fairly, eliminates any starvation of inputs, and approximates the behavior of an OQ switch as close as possible. We introduced the lag as a performance metric that measures the difference between

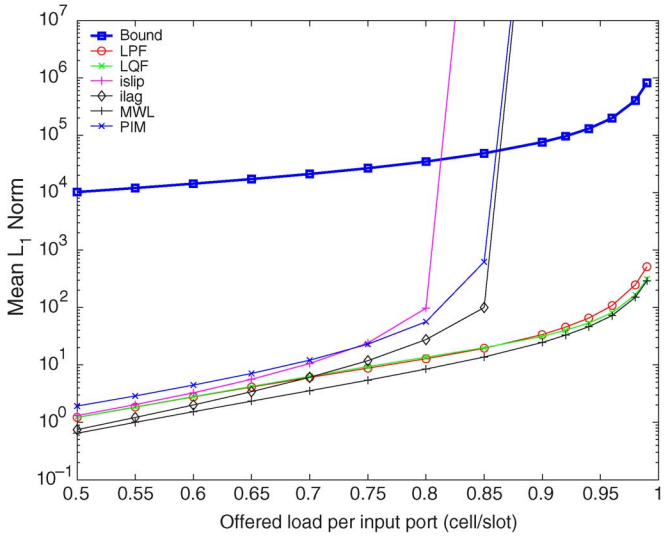

Fig. 6. $E[\|\underline{L}\|_1]$ versus offered load for diagonal traffic.

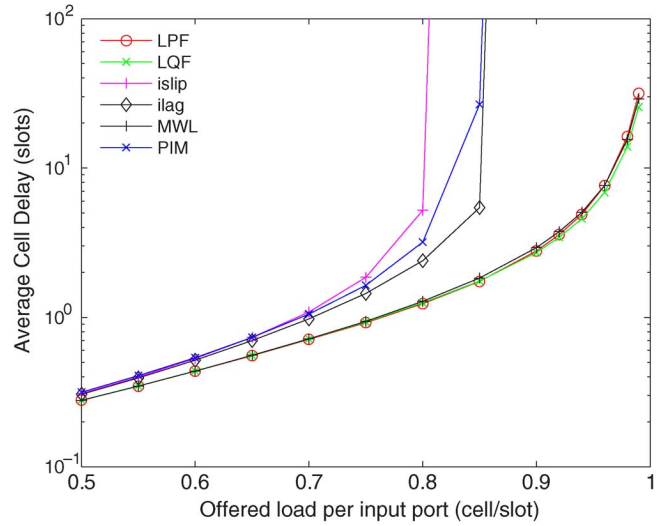

Fig. 7. Average cell delay versus offered load for diagonal traffic.

a packet's departure time in an IQ switch compared to an OQ switch. We proved that per-packet lag is bounded for a maximum weighted-matching scheduling policy that uses lag values for its weights and derived a bound on the mean lag value using a Lyapunov function technique. Finally, we proposed a heuristic tracking scheduling policy and evaluated its performance by simulation. The proposed iterative lag-based scheduler offers significant reductions in the mean lag compared to other iterative schemes such as iSlip.

Appendix I
MWL Stability Proof

Definition 8: Let $\underline{Q}(n)$ be the occupancy vector at time slot n such that

$$\underline{Q}(n) \equiv (Q_{1,1}(n), \ldots, Q_{1,N}(n), \ldots, Q_{N,1}(n), \ldots, Q_{N,N}(n))^T.$$

Definition 9: Let $\lambda_{\min} \equiv \min(\lambda_{i,j}, 1 \le i,j \le N)$. Without loss of generality, $\lambda_{\min} > 0$.

Definition 10: Let $C_{i,j}(n)$ denote the HOL cell of $Q_{i,j}$ at time slot n.

Definition 11: Let $A(n)$ be the arrival matrix representing the arrivals into each queue at time slot n, $A(n) \equiv [A_{i,j}(n)]$ where

$$A_{i,j}(n) \equiv \begin{cases} 1, & \text{if an arrival occurs at } Q_{i,j} \text{ at time slot } n \\ 0, & \text{otherwise.} \end{cases}$$

The associated arrival vector is

$$\underline{A}(n) \equiv (A_{1,1}(n), \ldots, A_{1,N}(n), \ldots, A_{N,1}(n), \ldots, A_{N,N}(n))^T.$$

Definition 12: Let $S(n)$ be the service matrix indicating which queues are served during time slot n, $S(n) \equiv [S_{i,j}(n)]$ where

$$S_{i,j}(n) \equiv \begin{cases} 1, & \text{if } Q_{i,j} \text{ is served at time slot } n \\ 0, & \text{otherwise} \end{cases}$$

and $S(n) \in S$, the set of service matrices. This definition of the "service" matrix is a permutation matrix, which includes the case where an empty queue is served. Note that $S(n)$ is a permutation matrix; that is, $\sum_{i=1}^{N} S_{ij} = \sum_{j=1}^{N} S_{ij} = 1$. We define the associated service vector $\underline{S}(n) \equiv (S_{1,1}(n), \ldots, S_{1,N}(n), \ldots, S_{N,N}(n))^T$.

Definition 13: Given a vector $\underline{X} \in \mathbb{R}^{N^2}$, the second order norm $\|\underline{X}\|$ is defined as

$$\|\underline{X}\| = \sqrt{\sum_{k=1}^{N^2} (X_k)^2}.$$

Definition 14: Let $\underline{L}(n)$ be the lag vector at time slot n such that

$$\underline{L}(n) \equiv (L_{1,1}(n), \ldots, L_{1,N}(n), \ldots, L_{N,1}(n), \ldots, L_{N,N}(n))^T$$

where $L_{i,j}(n)$ is the lag of $\mathrm{HOL}_{i,j}$ at time slot n.

Definition 15: Let $L_{\max}(n) \equiv \max(L_{i,j}(n), 1 \le i,j \le N)$.

Definition 16: Let T be a positive-definite diagonal matrix whose diagonal elements are

$$\lambda_{1,1}, \ldots, \lambda_{1,N}, \ldots, \lambda_{N,1}, \ldots, \lambda_{N,N}.$$

Definition 17: $[\underline{a} \odot \underline{b} \odot \underline{c}]$ denotes a vector in which each element is a Hadamard product of the corresponding elements of the vectors: $\underline{a}$, $\underline{b}$, and $\underline{c}$, i.e., $a_{i,j} b_{i,j} c_{i,j}$.

Definition 18: Let $\underline{1}$ denote a column vector of dimension N^2 whose elements are all ones.

Definition 19: Let $\underline{0}$ denote a column vector of dimension N^2 whose elements are all zeros.

Definition 20: Let $E_{i,j}(n)$ denote the arrival time of the HOL cell in $Q_{i,j}$ at time n. Let $\tau_{i,j}(n)$ denote the interarrival time between the HOL cell and the cell behind it in $Q_{i,j}$. Therefore, $E_{i,j}(n+1) = E_{i,j}(n) + \tau_{i,j}(n)$. Let $\underline{\tau}(n)$ be a vector of interarrival times such that

$$\underline{\tau}(n) \equiv (\tau_{1,1}(n), \ldots, \tau_{1,N}(n), \ldots, \tau_{N,1}(n), \ldots, \tau_{N,N}(n))^T.$$

Let $\underline{\Delta\tau}(n)$ be a vector of time intervals, where $\Delta\tau_{i,j}(n) = [E_{i,j}(n), E_{i,j}(n)+\tau_{i,j}(n)]$ such that

$$\begin{aligned}\underline{\Delta\tau}(n) \equiv (&[E_{1,1}(n), E_{1,1}(n)+\tau_{1,1}(n)], \ldots,\\ &[E_{1,N}(n), E_{1,N}(n)+\tau_{1,N}(n)], \ldots,\\ &[E_{N,1}(n), E_{N,1}(n)+\tau_{N,1}(n)], \ldots,\\ &[E_{N,N}(n), E_{N,N}(n)+\tau_{N,N}(n)])^T.\end{aligned}$$

Definition 21: Let $\underline{Z}(\underline{\Delta\tau}(n))$ be the aggregate arrival vector for each output port during the time interval $\underline{\Delta\tau}(n)$ (interpreted component-wise)

$$\underline{Z}\left(\underline{\Delta\tau}(n)\right) = \left(Z_{1,1}\left(\Delta\tau_{1,1}(n)\right), \ldots, Z_{N,N}\left(\Delta\tau_{N,N}(n)\right)\right)^T$$

where $Z_{i,j}(\Delta\tau_{i,j}(n))$ represents the aggregate number of cells that arrived to the switch during the time interval $\Delta\tau_{i,j}(n)$ destined to output j.

Definition 22: The *approximate* Lag next-state vector, $\widetilde{\underline{L}}$, which does not consider the case of an empty queue, is given by (the max function is interpreted component-wise)

$$\widetilde{\underline{L}}(n+1) \equiv \max\left(\underline{L}(n) + \underline{1} - \underline{S}(n) \odot [\underline{\tau}(n) + \underline{Z}\left(\underline{\Delta\tau}(n)\right)], \underline{0}\right). \tag{2}$$

The approximate next-state vector, $\widetilde{\underline{L}}$, assumes that each VOQ always has a packet. The exact next-state vector, $\underline{L}$, takes the empty queue case into account.

$$L_{i,j}(n+1) = \begin{cases} \widetilde{L}_{i,j}(n+1), & Q_{i,j}(n+1) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Explanation: Equation (2) describes the evolution of the lag vector. In the above equation, if $Q_{i,j}$ is not serviced at slot n, then its corresponding $S_{i,j}$ element in $\underline{S}(n)$ is zero, and the corresponding term in $\underline{S}(n) \odot [\underline{\tau}(n) + \underline{Z}(\underline{\Delta\tau}(n))]$ cancels out. In this case, the lag increases by 1. Alternatively, if the HOL cell at $Q_{i,j}$ is serviced at time slot n, then we need to calculate the lag of the cell following it in the queue. We consider two subcases:

CASE A) There were no packet arrivals to the switch destined to output j during the interarrival period between the HOL cell at $Q_{i,j}$ and the cell following it (i.e., $Z_{i,j}(\Delta\tau_{ij})$ is zero). In this case, the corresponding element for $Q_{i,j}$ in $\underline{S}(n)$ is 1 and $Z_{i,j}(\Delta\tau_{i,j})$ is zero. Therefore,

$$L_{i,j}(n+1) = L_{i,j}(n) + 1 - \tau_{i,j}(n)$$

i.e., the new lag is the old lag minus the interarrival time between the two cells.

CASE B) There were arrivals during the interarrival period between the HOL cell in $Q_{i,j}$ and the cell following it. In this case, all cells that arrived during this interarrival period should depart from the switch (or be selected to be transferred across the switch by the scheduler) before the new HOL cell at $Q_{i,j}$, so the new lag is given by

$$L_{i,j}(n+1) = L_{i,j}(n) + 1 - \tau_{i,j}(n) - Z_{i,j}\left(\Delta\tau_{i,j}(n)\right).$$

The following facts are used in the proof of the stability of the lag vector.

Fact 1: For all i, j, n, an interarrival time $\tau_{i,j}(n)$ is independent of the lag $L_{i,j}(n)$. This is true because we are assuming an i.i.d. traffic model.

Fact 2: $\tau_{i,j}(n) \geq 1$ because there is at most one arrival per time slot, so the arrival times of any two consecutive cells must be at least one slot apart.

Proof of Theorem 1

We prove the stability of the lag vector, which implies the stability of the queue occupancy. Recall that the lag is defined in terms of the total occupancy of packets in the switch destined to an output port.

The following Lemma is adapted from [13, Lemma 7].

Lemma 1: $\underline{L}^T(n)\underline{\lambda} - \underline{L}^T(n)\underline{S}^*(n) \leq 0\ \forall \underline{L}(n), \underline{\lambda}$ where $\underline{S}^*(n)$ is such that $\underline{L}^T(n)\underline{S}^*(n) = \max(\underline{L}^T(n)\underline{S}(n))$. Note that $\underline{S}^*(n)$ is the service vector selected by the MWL scheduling policy at time slot n.

Proof: Identical to the proof of [13], Lemma 2. ∎

The following Lemma is adapted from [13, Lemma 8] and is simplified for an $N \times N$ switch rather than an $N \times M$ switch.

Lemma 2: For all $\underline{\lambda} \leq (1-\beta)\underline{\lambda}_m$ (the inequality is interpreted component-wise), $0 < \beta < 1$, where $\underline{\lambda}_m$ is any rate vector such that $\|\underline{\lambda}_m\|^2 = N$, there exist $\varepsilon > 0$ and $K < \infty$ such that

$$E\left[\widetilde{\underline{L}}^T(n+1)T\widetilde{\underline{L}}(n+1) - \underline{L}^T(n)T\underline{L}(n) \middle| \underline{L}(n)\right] \leq -\varepsilon\|\underline{L}(n)\| + K.$$

Proof: By expansion

$$\begin{aligned}\widetilde{\underline{L}}^T(n+1)T\widetilde{\underline{L}}(n+1) = &\underline{L}^T(n)T\underline{L}(n) + 2\underline{L}^T(n)\underline{\lambda}\\ &- 2\underline{L}^T(n)\left[\underline{S}^*(n) \odot \underline{\tau}(n) \odot \underline{\lambda}\right]\\ &- 2\underline{L}^T(n)\left[\underline{S}^*(n) \odot \underline{Z}\left(\underline{\Delta\tau}(n)\right) \odot \underline{\lambda}\right]\\ &+ \sum_{i,j}\lambda_{i,j} - 2\sum_{i,j}S^*_{i,j}(n)\tau_{i,j}(n)\lambda_{i,j}\\ &- 2\sum_{i,j}S^*_{i,j}(n)Z_{i,j}\left(\Delta\tau_{i,j}(n)\right)\lambda_{i,j}\\ &+ \sum_{i,j}S^*_{i,j}(n)\left(\tau_{i,j}(n)\right)^2\lambda_{i,j}\\ &+ 2\sum_{i,j}S^*_{i,j}(n)Z_{i,j}\left(\Delta\tau_{i,j}(n)\right)\\ &+ \sum_{i,j}S^*_{i,j}(n)Z^2_{i,j}\left(\Delta\tau_{i,j}(n)\right)\lambda_{i,j}.\end{aligned}$$

Subtracting $\underline{L}^T(n)T\underline{L}(n)$ from both sides and taking the expected value and observing that the expected value of $\tau_{i,j}(n)$ is $1/\lambda_{i,j}$, then

$$\begin{aligned}&E\left[\widetilde{\underline{L}}^T(n+1)T\widetilde{\underline{L}}(n+1) - \underline{L}^T(n)T\underline{L}(n) \middle| \underline{L}(n)\right]\\ &= 2\underline{L}^T(n)\underline{\lambda} - 2\underline{L}^T(n)\underline{S}^*(n)\\ &\quad - 2\underline{L}^T(n)\left(\underline{S}^*(n) \odot \underline{Z}\left(\underline{\Delta\tau}(n)\right) \odot \underline{\lambda}\right) + \sum_{i,j}\lambda_{i,j}\\ &\quad - 2\sum_{i,j}S^*_{i,j}(n) - 2\sum_{i,j}S^*_{i,j}(n)E\left[Z_{i,j}\left(\Delta\tau_{i,j}(n)\right)\lambda_{i,j}\right]\\ &\quad + \sum_{i,j}\frac{S^*_{i,j}(n)}{\lambda_{i,j}} + 2\sum_{i,j}S^*_{i,j}(n)E\left[Z_{i,j}\left(\Delta\tau_{i,j}(n)\right)\right]\\ &\quad + \sum_{i,j}S^*_{i,j}(n)E\left[Z^2_{i,j}\left(\Delta\tau_{i,j}(n)\right)\lambda_{i,j}\right].\end{aligned} \tag{4}$$

We make use of the following properties to simplify (4) and establish Lemma 2:

(a) $\sum_{i,j} \lambda_{i,j} < N$; (from the admissibility constraints)

(b) $\sum_{i,j} S^*_{i,j}(n) \geq 0$; the term in (4) that contains this sum with a negative coefficient can be ignored to yield an upper bound for the expectation on the left hand side.

(c) $\underline{L}^T(n)(\underline{S}^*(n) \odot \underline{Z}(\underline{\Delta\tau}(n)) \odot \underline{\lambda}) \geq 0$; (because each element in this term is nonnegative; observe that this term has a negative sign in (4), so it can be ignored)

(d) $\sum_{i,j} S^*_{i,j}(n) E[Z_{i,j}(\Delta\tau_{i,j}(n))\lambda_{i,j}] \geq 0$; (because each element in this term is nonnegative, observe that this term has a negative sign in (4), so it can be ignored). Recall that although all the summation terms have N^2 elements, there can be at most N elements with $S^*_{i,j}(n) = 1$. Also, note that the following positive terms in (4) are bounded:

$$\begin{aligned}
&\sum_{i,j} S^*_{i,j}(n)/\lambda_{i,j} \leq N/\lambda_{\min} < \infty \\
&\sum_{i,j} S^*_{i,j}(n) E\left[Z^2_{i,j}\left(\Delta\tau_{i,j}(n)\right)\lambda_{i,j}\right] \\
&\quad \leq \sum_{i,j} S^*_{i,j}(n) E\left[Z^2_{i,j}(\Delta\tau_{i,j})(n)\right] \leq N < \infty \\
&\sum_{i,j} S^*_{i,j}(n) E\left[Z_{i,j}\left(\Delta\tau_{i,j}(n)\right)\right] \leq N < \infty. \qquad (5)
\end{aligned}$$

From (4), properties (a)–(d), and (5), we obtain

$$\begin{aligned}
E\left[\widetilde{\underline{L}}^T(n+1)T\widetilde{\underline{L}}(n+1) - \underline{L}^T(n)T\underline{L}(n)|\underline{L}(n)\right] \\
\leq 2\underline{L}^T(n)\underline{\lambda} - 2\underline{L}^T(n)\underline{S}^*(n) + 4N + N/\lambda_{\min}. \qquad (6)
\end{aligned}$$

Using the assumptions in Lemma 2, we obtain

$$\underline{L}^T(n)\underline{\lambda} - \underline{L}^T(n)\underline{S}^*(n) \leq \underline{L}^T(n)(1-\beta)\underline{\lambda}_m - \underline{L}^T(n)\underline{S}^*(n).$$

Applying Lemma 1 to the $\underline{L}^T(n)\underline{\lambda}_m - \underline{L}^T(n)\underline{S}^*(n)$ term on the right-hand side (RHS) (i.e., the term is eliminated because it is nonpositive), we get

$$\begin{aligned}
\underline{L}^T(n)\underline{\lambda} - \underline{L}^T(n)\underline{S}^*(n) &\leq -\beta\underline{L}^T(n)\underline{\lambda}_m \\
\underline{L}^T(n)\underline{\lambda} - \underline{L}^T(n)\underline{S}^*(n) &\leq -\beta\left\|\underline{L}^T(n)\right\|\|\underline{\lambda}_m\|\cos(\theta) \quad (7)
\end{aligned}$$

where θ is the angle between $\underline{L}^T(n)$ and $\underline{\lambda}_m$. It can be easily shown that $\cos(\theta) > \delta$ for some $\delta > 0$ whenever $\underline{L}^T(n) \neq 0$ using the same approach as in [13, Equations (16)–(18)] to obtain a lower bound on $\cos(\theta)$

$$\cos(\theta) \geq \frac{\lambda_{\min}}{N^{\frac{3}{2}}}. \qquad (8)$$

Substituting (8) in (7) and using $\|\underline{\lambda}_m\|^2 = N$, we get

$$\underline{L}^T(n)\underline{\lambda} - \underline{L}^T(n)\underline{S}^*(n) \leq -\beta\left\|\underline{L}^T(n)\right\|\frac{\lambda_{\min}}{N} \qquad (9)$$

Substituting (9) in (6), we get

$$E\left[\widetilde{\underline{L}}^T(n+1)T\widetilde{\underline{L}}(n+1) - \underline{L}^T(n)T\underline{L}(n)|\underline{L}(n)\right] \leq -\varepsilon\|\underline{L}(n)\| + K \qquad (10)$$

where $\varepsilon = 2\beta(\lambda_{\min}/N)$ and $K = 4N + N/\lambda_{\min}$. ■

Lemma 3: For all $\underline{\lambda} \leq (1-\beta)\underline{\lambda}_m$ (the inequality is interpreted component-wise), $0 < \beta < 1$, where $\underline{\lambda}_m$ is any rate vector such that $\|\underline{\lambda}_m\|^2 = N$, there exist $\varepsilon > 0$ and $K < \infty$ such that

$$E\left[\underline{L}^T(n+1)T\underline{L}(n+1) - \underline{L}^T(n)T\underline{L}(n)|\underline{L}(n)\right] \leq -\varepsilon\|\underline{L}(n)\| + K.$$

Observe that the difference between Lemmas 2 and 3 is that Lemma 2 uses the approximate next-state vector, whereas Lemma 3 uses the exact next-state vector. The approximate next-state vector assumes that each VOQ always has a packet. The exact next-state vector takes the empty queue case into account.

The proof of this lemma is similar to the proof of [13, Lemma 9] and is included here for completeness.

Proof: The fact that T is a positive-definite matrix together with (3) implies that for all n

$$\underline{L}^T(n+1)T\underline{L}(n+1) \leq \widetilde{\underline{L}}^T(n+1)T\widetilde{\underline{L}}(n+1).$$

Therefore

$$\begin{aligned}
E\left[\underline{L}^T(n+1)T\underline{L}(n+1) - \underline{L}^T(n)T\underline{L}(n)|\underline{L}(n)\right] \\
\leq E\left[\widetilde{\underline{L}}^T(n+1)T\widetilde{\underline{L}}(n+1)|\underline{L}(n)\right].
\end{aligned}$$

This proves the lemma. ■

Lemma 4: There exists a quadratic Lyapunov function $V(\underline{L}(n))$ such that

$$E\left[V\left(\underline{L}(n+1)\right) - V\left(\underline{L}(n)\right)|\underline{L}(n)\right] \leq -\varepsilon\left\|\underline{L}(n)\right\| + K \quad (11)$$

where $K, \varepsilon > 0$.

Proof: From Lemma 3, $V(\underline{L}(n)) = \underline{L}^T(n)T\underline{L}(n)$, $\varepsilon = 2\beta(\lambda_{\min}/N)$, and $K = 4N + N/\lambda_{\min}$. ■

Theorem 3: Under MWL, the expectations of the lag values are bounded for all n under all admissible and independent arrival processes, i.e., $\forall n$, $E[\|\underline{L}(n)\|] < \infty$.

Proof: While $\{\underline{L}(n)\}$ is not a Markov chain, by appending the N-dimensional vector $\underline{Q}_{\mathrm{OQ}}(n)$, whose ith entry is the queue occupancy of an OQ switch, we see that $\{(\underline{L}(n), \underline{Q}_{\mathrm{OQ}}(n))\}$ is a Markov chain. The addition of $\underline{Q}_{\mathrm{OQ}}(n)$ to the state of the process allows one to compute the ideal departure times and the lag vector (see Section IV-A and Definition 4) from the current state of the process. Hence, the augmented process is Markov. Furthermore, each entry in $\underline{Q}_{\mathrm{OQ}}(n)$ is the queue length for a stable Geom/D/1 queue given a finite size N or the queue length for a stable M/D/1 queue as N approaches infinity. Therefore, it is straightforward to show that for some $\varepsilon_{\mathrm{OQ}} > 0$ and $B_{\mathrm{OQ}} < \infty$

$$\begin{aligned}
E\left[\underline{Q}^T_{\mathrm{OQ}}(n+1)\underline{Q}_{\mathrm{OQ}}(n+1)|\underline{Q}_{\mathrm{OQ}}(n)\right] \\
\leq -\varepsilon_{OQ}\|\underline{Q}_{\mathrm{OQ}}(n)\| + B_{\mathrm{OQ}}. \qquad (12)
\end{aligned}$$

Combining (11) and (12), we see that $V((\underline{L}(n), \underline{Q}_{\mathrm{OQ}}(n))) = \underline{L}^T(n)T\underline{L}(n) + \underline{Q}^T_{\mathrm{OQ}}(n)\underline{Q}_{\mathrm{OQ}}(n)$ is a quadratic Lyapunov function, and according to the arguments in [26], it follows that the expectations of the lag values are bounded for all n under the MWL scheduling policy. ■

Theorem 4: Under the MWL scheduling policy, the expectations of the queue occupancy are bounded for all n under all admissible and independent arrival processes, i.e., $\forall n$, $E[||\underline{Q}(n)||] < \infty$.

Proof: That stability of the lag values implies the stability of the per packet additional waiting in the IQ switch. Given the traffic admissibility constraints, each packet's delay in the OQ switch being tracked is finite. Consequently, the total delay provided by the IQ switch using MWL is bounded. Therefore, all the queue occupancies in the IQ switch under MWL are bounded for all n. ■

APPENDIX II
LAG BOUND FOR MWL SCHEDULING POLICY

In addition to the definitions given in Appendix I, the following definitions are necessary in this part.

Definition 23: The unit vector parallel to $\underline{X}$ is denoted by $\hat{\underline{X}}$, and is defined as

$$\hat{\underline{X}} = \frac{\underline{X}}{||\underline{X}||_1}.$$

To proceed, we need the following trivial generalization of a theorem due to Leonardi *et al.* [21, Theorem 3.6], which is presented here in a form appropriate for the problem under consideration. We do not provide the proof, as it is unchanged from that in [21].

Theorem 5: Consider a system of queues whose evolution is described by a discrete time Markov chain (DTMC) with state vector $Y_n = (\underline{L}(n), X(n))$, where $\{X(n)\}$ is itself a DTMC with all states positive recurrent. If all of the polynomial moments of lag distributions are finite and if a lower-bounded polynomial function $V(\underline{L}(n))$,$V : \mathbb{N}^N \to R$, can be found, such that $E[V(\underline{L}(n))|Y_n] < \infty$ and there exist two positive real numbers $\epsilon \in \mathbb{R}^+$ and $B \in \mathbb{R}^+$, such that

$$E\left[V\left(\underline{L}(n+1)\right) - V\left(\underline{L}(n)\right)|Y_n\right] \leq -\epsilon f\left(||\underline{L}(n)||\right) \quad (13)$$

for all Y_n such that $||\underline{L}(n)|| > B$ and where $f(x)$ is a continuous function in $\mathbb{R}^+$, then

$$\lim_{n\to\infty} E\left[f\left(||\underline{L}(n)||\right)\right] \leq \lim_{n\to\infty} E\left[f\left(||\underline{L}(n)||\right) + \frac{V\left(\underline{L}(n+1)\right) - V\left(\underline{L}(n)\right)}{\epsilon}|Y_n \in H_B\right] \times P[Y_n \in H_B]. \quad (14)$$

Note that for MWL, we can simply choose $X(n) = \underline{Q}_{\mathrm{OQ}}(n)$ in Theorem 5. In addition, all of the polynomial moments of the lag distribution can be shown to be finite by a corresponding trivial generalization of [21, Theorem 3.5].

The proof of Theorem 2 consists of two steps. First, we find a lower bound on ϵ in (13). The second step is to use (14) to derive the bound on $E[||\underline{L}(n)||_1]$.

Using (14) with $f(||\underline{L}(n)||) = ||\underline{L}(n)||_1$ and $V(\underline{L}(n)) = \underline{L}^T(n)T\underline{L}(n)$

$$-\frac{E\left[\underline{L}^T(n+1)T\underline{L}(n+1) - \underline{L}^T(n)T\underline{L}(n)|\underline{L}(n)\right]}{||\underline{L}(n)||_1} \geq \epsilon \quad (15)$$

for all $\underline{L}(n)$ such that $||\underline{L}(n)||_1 > B$ and for some $B > 0$. The function on the left-hand side (LHS) of (15) admits a limit for $||\underline{L}(n)||_1 \to \infty$, which depends on the direction of the vector $\underline{L}(n)$. Let $\epsilon_{\min}$ be smallest value for this limit, i.e. (5) holds

$$\epsilon_{\min} = \liminf_{||\underline{L}(n)||_1\to\infty} \left(-\frac{E\left[\underline{L}^T(n+1)T\underline{L}(n+1)}{||\underline{L}(n)||_1} + \frac{\underline{L}^T(n)T\underline{L}(n)|\underline{L}(n)\right]}{||\underline{L}(n)||_1}\right).$$

Substituting (4) in the previous equation and observing that all of the terms in the numerator that do not contain $\underline{L}(n)$ will go to zero upon dividing by $||\underline{L}(n)||_1 \to \infty$, we get

$$\epsilon_{\min} = \liminf_{||\underline{L}(n)||_1\to\infty} \left(-\frac{2\underline{L}^T(n)\underline{\lambda} - 2\underline{L}^T(n)\underline{S}^*(n)}{||\underline{L}(n)||_1} + \frac{2\underline{L}^T(n)\left(\underline{S}^*(n) \odot \underline{Z}\left(\underline{\Delta\tau}(n)\right) \odot \underline{\lambda}\right)}{||\underline{L}(n)||_1}\right).$$

Taking 2 as a common factor and rearranging the terms, we get

$$\epsilon_{\min} = 2\liminf_{||\underline{L}(n)||_1\to\infty} \left(\frac{\underline{L}^T(n)\underline{S}^*(n) - \underline{L}^T(n)\underline{\lambda}}{||\underline{L}(n)||_1} + \frac{\underline{L}^T(n)\left(\underline{S}^*(n) \odot \underline{Z}\left(\underline{\Delta\tau}(n)\right) \odot \underline{\lambda}\right)}{||\underline{L}(n)||_1}\right). \quad (16)$$

We make use of the following proposition, which was proved in [21, Proposition A.1] and is included here for completeness.

Proposition 1: For any nonnull normalized vector $\hat{\underline{Z}}(n) \in \mathbb{R}^{+N^2}$

$$\underline{S}^*(n)\hat{\underline{Z}}^T(n) \geq \frac{1}{N}.$$

Applying Proposition 1 to the second term in (16), we get

$$\frac{\underline{L}^T(n)\left(\underline{S}^*(n) \odot \underline{Z}\left(\underline{\Delta\tau}(n)\right) \odot \underline{\lambda}\right)}{||\underline{L}(n)||_1} \geq \frac{\underline{Z}\left(\underline{\Delta\tau}(n)\right) \cdot \underline{\lambda}}{N} \geq 0. \quad (17)$$

Now, we use a technique from [21, pp. 542–543] to bound the following term:

$$\frac{\underline{L}^T(n)\underline{S}^*(n) - \underline{L}^T(n)\underline{\lambda}}{||\underline{L}(n)||_1}.$$

Consider the vector $\underline{U}(n) = E[\underline{A}(n)] + (1 - ||\underline{\lambda}||_{\mathrm{IO}})\underline{S}^*(n)$. It is straightforward to prove that $||\underline{U}(n)||_{\mathrm{IO}} \leq 1$. Also, the fact that the system is stable implies $E[\underline{A}(n)] = E[\underline{S}^*(n)] = \underline{\lambda}$. Thus

$$\frac{\underline{S}^*(n)\underline{L}^T(n) - \underline{U}(n)\underline{L}^T(n)}{||\underline{L}(n)||_1} = \frac{\underline{S}^*(n)\underline{L}^T(n) - \underline{\lambda} \odot \underline{L}^T(n)}{||\underline{L}(n)||_1} - \frac{\left(1 - ||\underline{\lambda}||_{\mathrm{IO}}\right)\underline{S}^*(n)\underline{L}^T(n)}{||\underline{L}(n)||_1} \geq 0$$

and from Lemma 1 we have

$$\frac{\underline{S}^*(n)\underline{L}^T(n) - \underline{\lambda}\underline{L}^T(n)}{||\underline{L}(n)||_1} \geq \frac{\left(1 - ||\underline{\lambda}||_{IO}\right)\underline{S}^*(n)\underline{L}^T(n)}{||\underline{L}(n)||_1}.$$

Applying Proposition 1, we get

$$\frac{\underline{S}^*(n)\underline{L}^T(n) - \underline{\lambda}\underline{L}^T(n)}{||\underline{L}(n)||_1} \geq \frac{(1 - ||\underline{\lambda}||_{\mathrm{IO}})}{N}. \tag{18}$$

Using (17) and (18) to evaluate (16), we get

$$\epsilon_{\min} \geq \frac{2}{N}(1 - ||\underline{\lambda}||_{\mathrm{IO}}). \tag{19}$$

The next step is to evaluate (14)

$$\lim_{n\to\infty} E[f(||\underline{L}(n)||)] \leq \lim_{n\to\infty} E\Bigg[f(||\underline{L}(n)||) + \frac{V(\underline{L}(n+1)) - V(\underline{L}(n))}{\epsilon} | Y_n \in H_B\Bigg] \times P[Y_n \in H_B].$$

Evaluating the term $E[V(\underline{L}(n+1)) - V(\underline{L}(n))|Y_n \in H_B]$ appearing in (13) and using (4)

$$\begin{aligned}
&E[V(\underline{L}(n+1)) - V(\underline{L}(n))|Y_n \in H_B] \\
&= E\left[\underline{L}^T(n+1)T\underline{L}(n+1) - \underline{L}^T(n)T\underline{L}(n)|\underline{L}(n)\right] \\
&= 2\underline{L}^T(n)\underline{\lambda} - 2\underline{L}^T(n)\underline{S}^*(n) \\
&\quad - 2\underline{L}^T(n)(\underline{S}^*(n) \odot \underline{Z}(\underline{\Delta\tau}(n)) \odot \underline{\lambda}) \\
&\quad + \sum_{i,j}\lambda_{i,j} - 2E\left[\sum_{i,j} S^*_{i,j}(n)\right] \\
&\quad - 2E\left[\sum_{i,j} S^*_{i,j}(n) Z_{i,j}(\Delta\tau_{i,j}(n))\lambda_{i,j}\right] \\
&\quad + E\left[\sum_{i,j}\frac{S^*_{i,j}(n)}{\lambda_{i,j}}\right] + 2E\left[\sum_{i,j} S^*_{i,j}(n) Z_{i,j}(\Delta\tau_{i,j}(n))\right] \\
&\quad + E\left[\sum_{i,j} S^*_{i,j}(n) Z^2_{i,j}(\Delta\tau_{i,j}(n))\lambda_{i,j}\right].
\end{aligned}$$

Using (17) and (18), we get

$$\begin{aligned}
&E\left[\underline{L}^T(n+1)T\underline{L}(n+1) - \underline{L}^T(n)T\underline{L}(n)|\underline{L}(n)\right] \\
&\leq \epsilon_{min}||\underline{L}(n)||_1 + \sum_{i,j}\lambda_{i,j} - 2E\left[\sum_{i,j} S^*_{i,j}(n)\right] \\
&\quad - 2E\left[\sum_{i,j} S^*_{i,j}(n) Z_{i,j}(\Delta\tau_{i,j}(n))\lambda_{i,j}\right] \\
&\quad + E\left[\sum_{i,j}\frac{S^*_{i,j}(n)}{\lambda_{i,j}}\right] + 2E\left[\sum_{i,j} S^*_{i,j}(n) Z_{i,j}(\Delta\tau_{i,j}(n))\right] \\
&\quad + E\left[\sum_{i,j} S^*_{i,j}(n) Z^2_{i,j}(\Delta\tau_{i,j}(n))\lambda_{i,j}\right].
\end{aligned}$$

From stability, we have $E[\underline{S}(n)] = \underline{\lambda}$, $E[\underline{S}^T(n)\underline{S}(n)] = ||\underline{\lambda}||_1$, and $E[\underline{\lambda}^T\underline{S}(n)] = \underline{\lambda}^T E[\underline{S}(n)] = ||\underline{\lambda}||_2^2$. Also, $E[S_{i,j}] = \lambda_{i,j}$, so $E[\sum_{i,j}(S^*_{i,j}(n)/\lambda_{i,j})] = N^2$ because we are summing over N^2 elements and each element is 1. Substituting these equalities in the previous equation and using the inequality $E[\sum_{i,j} S^*_{i,j}(n) Z_{i,j}(\Delta\tau_{i,j}(n))] \leq N||\underline{\lambda}||_1$, we get

$$\begin{aligned}
&E\left[\underline{L}^T(n+1)T\underline{L}(n+1) - \underline{L}^T(n)T\underline{L}(n)|\underline{L}(n)\right] \\
&\leq \epsilon_{\min}||\underline{L}(n)||_1 - ||\underline{\lambda}||_1 \\
&\quad - 2E\left[\sum_{i,j} S^*_{i,j}(n) Z_{i,j}(\Delta\tau_{i,j}(n))\lambda_{i,j}\right] + N^2 + 2N||\underline{\lambda}||_1 \\
&\quad + E\left[\sum_{i,j} S^*_{i,j}(n) Z^2_{i,j}(\Delta\tau_{i,j}(n))\lambda_{i,j}\right].
\end{aligned} \tag{20}$$

Substituting (20) in (14), we get

$$\begin{aligned}
&E[||\underline{L}(n)||_1] \\
&\leq E\left[||\underline{L}(n)||_1 + \frac{V(\underline{L}(n+1)) - V(\underline{L}(n))}{\epsilon}|\underline{L}(n)\right] \\
&E[||\underline{L}(n)||_1] \\
&\leq E\left[||\underline{L}(n)||_1\left(1 - \frac{\epsilon_{\min}}{\epsilon}\right)\right] \\
&\quad + \frac{N^2 + 2N||\underline{\lambda}||_1 + E\left[\sum_{i,j} Z^2_{i,j}(\Delta\tau_{i,j}(n))\lambda^2_{i,j}\right]}{\epsilon} \\
&\quad - \frac{||\underline{\lambda}||_1 + 2E\left[\sum_{i,j} Z_{i,j}(\Delta\tau_{i,j}(n))\lambda^2_{i,j}\right]}{\epsilon}.
\end{aligned} \tag{21}$$

If we set $\epsilon = \epsilon_{\min}$, we get

$$\begin{aligned}
E[||\underline{L}(n)||_1] &\leq \frac{N^2 + 2N||\underline{\lambda}||_1 + E\left[\sum_{i,j} Z^2_{i,j}(\Delta\tau_{i,j}(n))\lambda^2_{i,j}\right]}{\epsilon_{\min}} \\
&\quad - \frac{||\underline{\lambda}||_1}{\epsilon_{\min}} - \frac{2E\left[\sum_{i,j} Z_{i,j}(\Delta\tau_{i,j}(n))\lambda^2_{i,j}\right]}{\epsilon_{\min}} \\
&\leq \frac{N^2 + 2N||\underline{\lambda}||_1 + E\left[\sum_{i,j} Z^2_{i,j}(\Delta\tau_{i,j}(n))\lambda^2_{i,j}\right]}{\epsilon_{\min}} \\
&\leq \frac{N^2 + 2N||\underline{\lambda}||_1 + N||\underline{\lambda}||_1}{\frac{2}{N}(1 - ||\underline{\lambda}||_{IO})} \\
&\leq \frac{N^2 + 3N||\underline{\lambda}||_1}{\frac{2}{N}(1 - ||\underline{\lambda}||_{IO})} \\
&\leq \frac{N^3 + 3N^2||\underline{\lambda}||_1}{2(1 - ||\underline{\lambda}||_{IO})}.
\end{aligned}$$

REFERENCES

[1] A. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, Jun. 1993.

[2] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division switch," *IEEE Trans. Commun.*, vol. COM-35, no. 12, pp. 1347–1356, Dec. 1987.

[3] S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 6, pp. 1030–1039, Jun. 1999.

[4] P. Krishna, N. Patel, A. Charny, and R. Simcoe, "On the speedup required for work-conserving crossbar switches," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 6, pp. 1057–1066, Jun. 1999.

[5] B. Magill, C. Rohrs, and R. Stevenson, "Output-queued switch emulation by fabrics with limited memory," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 606–615, May 2003.

[6] P. Giaccone, E. Leonardi, B. Prabhakar, and D. Shah, "Delay bounds for combined input-output switches with low speedup," *Perf. Eval.*, vol. 55, no. 1–2, pp. 113–128, Jan. 2004.
[7] C. Minkenberg, "Work-conservingness of CIOQ packet switches with limited output buffers," *IEEE Commun. Lett.*, vol. 6, pp. 452–454, Oct. 2002.
[8] Cisco 12000 Series-Internet Routers. 2004 [Online]. Available: http://www.cisco.com
[9] C. Partridge, P. P. Carvey, E. Burgess, I. Castineyra, T. Clarke, L. Graham, M. Hathaway, P. Herman, A. King, S. Kohalmi, T. Ma, J. Mcallen, T. Mendez, W. C. Milliken, R. Pettyjohn, J. Rokosz, J. Seeger, M. Sollins, S. Storch, B. Tober, G. D. Troxel, D. Waitzman, and S. Winterble, "A 50-Gb/s IP router," *IEEE/ACM Trans. Netw.*, vol. 6, no. 3, pp. 237–248, Jun. 1998.
[10] Lucent Technologies, Inc., Holmdel, NJ, 2004 [Online]. Available: http://www.lucent.com
[11] Avici Systems, Inc., Billercia, MA, 2004 [Online]. Available: http://www.avici.com
[12] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, pp. 188–201, Apr. 1999.
[13] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260–1267, Aug. 1999.
[14] V. Tabatabaee, L. Georgiadis, and L. Tassiulas, "QoS provisioning and tracking fluid policies in input queueing switches," *IEEE/ACM Trans. Netw.*, vol. 9, no. 5, pp. 605–617, Oct. 2001.
[15] M. Rosenblum, M. X. Goemans, and V. Tarokh, "Universal bounds on buffer size for packetizing fluid policies in input queued, crossbar switches," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 1126–1134.
[16] T. H. Szymanski, "Design principles for practical self-routing nonblocking connection networks with $n \log(n)$ bit-complexity," *IEEE Trans. Comput.*, vol. 46, no. 10, pp. 1057–1069, Oct. 1997.
[17] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[18] I. Keslassy and N. McKeown, "Analysis of scheduling algorithms that provide 100% throughput in input-queued switches," presented at the 39th Annu. Allerton Conf. Commun., Control, Comput., Oct. 2001.
[19] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. IEEE INFOCOM*, Tel Aviv, Israel, Mar. 2000, pp. 556–564.
[20] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "Bounds on average delays and queue size averages and variances in input-queued cell-based switches," in *Proc. IEEE INFOCOM*, Alaska, Apr. 2001, pp. 1095–1103.
[21] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "Bounds on delays and queue lengths in input-queued cell switches," *JACM*, vol. 50, pp. 520–550, Jul. 2003.
[22] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *Prco. IEEE INFOCOM*, 1998, pp. 792–799.
[23] A. Mekkittikul, "Scheduling non-uniform traffic in high speed packet switches and routers," Ph.D. dissertation, Stanford University, Stanford, CA, Nov. 1998.
[24] M. Andrews and L. Zhang, "Achieving stability in networks of input-queued switches," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 848–857, Oct. 2003.
[25] T. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High speed switch scheduling for local area networks," *Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319–352, Nov. 1993.
[26] P. R. Kumar and S. P. Meyn, "Stability of queueing networks and scheduling policies," *IEEE Trans. Autom. Control*, vol. 40, no. 2, pp. 251–260, Feb. 1995.

Amir Gourgy (M'06) received the B.Eng. degree in computer engineering from McMaster University, Hamilton, ON, Canada, in 1998; the M.A.Sc. degree in computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2001; and the Ph.D. degree in electrical engineering from McMaster University in 2006.

He is currently with Research in Motion, Inc., Waterloo, ON, Canada. He has held several positions at Synopsys, Inc., AMCC, Soma Networks Canada, and Bell Canada Software Reliability Laboratory.

Ted H. Szymanski (M'87) received the B.A.Sc. degree in engineering science and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 1980, 1982, and 1987, respectively.

He holds the Bell Canada Chair in Data Communications at McMaster University, Hamilton, ON, Canada. He has held faculty positions at Columbia University, New York, NY, where he was affiliated with the Center for Telecommunications Research (CTR), and McGill University, Montreal, QC, Canada, where he was affiliated with the Canadian Institute for Telecommunications Research (CITR). From 1993 to 2003, he was a principal architect in a national research program on Photonic Systems funded by the Canadian Networks of Centers of Excellence (NCE) program. The program brought together significant industrial and academic collaborators, including Nortel Networks, Newbridge Networks (now Alcatel), Lockheed-Martin/Sanders, Lucent Technologies, and McGill, McMaster, Toronto, and Heriot-Watt Universities. The program demonstrated a free-space "intelligent optical backplane" exploiting emerging optoelectronic technologies, with 1024 microoptic laser channels per square centimeter of bisection area, for which he holds two patents. He has consulted for several companies and has served as the Associate Chair (undergraduate) in the ECE Department, McMaster University. His current interests include switching, scheduling, and network QoS for emerging telemedicine and telerobotic control systems.

Douglas G. Down (SM'05) received the B.A.Sc. and M.A.Sc. degrees in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 1982 and 1988, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 1994.

He is an Associate Professor in the Department of Computing and Software at McMaster University, Hamilton, ON, Canada. From 1996 to 2000, he was an Assistant Professor in the School of Industrial and Systems Engineering at Georgia Institute of Technology, Atlanta.