# Packet-Mode Emulation of Output-Queued Switches

Hagit Attiya, David Hay, *Member*, *IEEE*, and Isaac Keslassy, *Member*, *IEEE*

**Abstract**—Most common network protocols transmit variable size *packets*, whereas contemporary switches still operate with fixed-size *cells*, which are easier to transmit and buffer. This necessitates packet segmentation and reassembly modules, resulting in significant computation and communication overhead that might be too costly as switches become faster and bigger. It is, therefore, imperative to investigate an alternative mode of scheduling in which packets are scheduled contiguously over the switch fabric. This paper investigates the cost of *packet-mode scheduling* for the *combined input-output-queued (CIOQ)* switch architecture. We devise frame-based schedulers that allow a packet-mode CIOQ switch with small speedup to *mimic* an ideal output-queued switch, with bounded relative queuing delay. The schedulers are pipelined and based on matrix decomposition. Our schedulers demonstrate a trade-off between the switch speedup and the relative queuing delay incurred while mimicking an output-queued switch. When the switch is allowed to incur high relative queuing delay, a speedup arbitrarily close to two suffices to mimic an ideal output-queued switch. This implies that packet-mode scheduling does not require higher speedup than a cell-based scheduler. The relative queuing delay can be significantly reduced with just a doubling of the speedup. We further show that it is impossible to achieve zero relative queuing delay (that is, a perfect emulation), regardless of the switch speedup. In addition, simpler algorithms can mimic an output-queued switch with a bounded buffer size, using speedup arbitrarily close to one. Simulations confirm that packet-mode emulation with reasonable relative queuing delay can be achieved with moderate speedup. Furthermore, a simple and practical heuristic is shown by simulations to also provide effective packet-mode emulation.

**Index Terms**—Internetworking, packet-switching networks, routers, sequencing and scheduling.

## 1 Introduction

In many network protocols, from very large Wide Area Networks (WANs) to small Networks on Chips (NoCs), traffic consists of *variable-size packets*. A prime example is provided by IP datagrams whose sizes typically vary from 40 to 1,500 bytes [1].

Real-life routers, however, operate with *fixed-size cells*, which are easier to buffer and schedule synchronously. For instance, cell-based *Combined Input-Output-Queued (CIOQ)* architectures with a crossbar switch fabric (Fig. 1) are widely used today as the core of contemporary routers [2], [3]. These switches operate at a rate close to the external line rate, and therefore, can scale with an increasing number of ports. In addition, *cell-based* CIOQ switches are well known for providing strong performance guarantees. In fact, with a small speedup[1] of two, they can exactly emulate an ideal Output-Queued (OQ) switch [4], [5], [6], [7], [8], [9].

1. The speedup of a switch is the ratio between the rate at which the switch fabric operates and the external rate.

- H. Attiya is with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel. E-mail: hagit@cs.technion.ac.il.
- D. Hay is with the Department of Electrical Engineering, Columbia University, 1300 S.W. Mudd, Mail code 4712, 500 West 120th Street, NY 10027. E-mail: hdavid@ee.columbia.edu.
- I. Keslassy is with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel. E-mail: isaac@ee.technion.ac.il.

*Manuscript received 7 July 2008; revised 6 Apr. 2009; accepted 23 Nov. 2009; published online 11 Dec. 2009.*
*Recommended for acceptance by X. Zhang.*
*For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2008-07-0335.*
*Digital Object Identifier no. 10.1109/TC.2009.186.*

Nevertheless, while they are apparently simpler, *cell-based switches* with fixed-size cells can actually cause many significant problems in practice:

Transmitting packets over cell-based switches requires the use of packet segmentation and reassembly modules, resulting in a significant computation and communication overhead [10]. Specifically, in order to provide contiguous packet delivery from the switch, all the cells of a packet should be stored in a designated buffer until such a successful delivery is certain (in most cases, until all the cells arrive at the output port). Moreover, each output port should be able to reassemble simultaneously and separately many packets coming from many input ports.

Cell-based scheduling is expected to turn into an even more crucial problem as the use of optics becomes widespread, since future switches could deal with packets in the optical domain and might be unable to afford their segmentation and reassembly.

An even more intricate problem arises in congestion time, when cells need to be dropped. Cell-based schedulers, unaware of packet boundaries, may drop cells from many different packets, thus causing many packets to be dropped, which results in a significant performance degradation. In contrast, packet-aware switches can ensure that drops are localized at a few packets, thus confining the performance degradation and the number of retransmissions needed (cf. [11, Page 44]).

*Packet-mode schedulers* [12], [13] provide an attractive middle ground by delivering packets contiguously over the switch fabric, implying that until a packet is fully transmitted, neither its originating port nor its destination port can handle different packets.

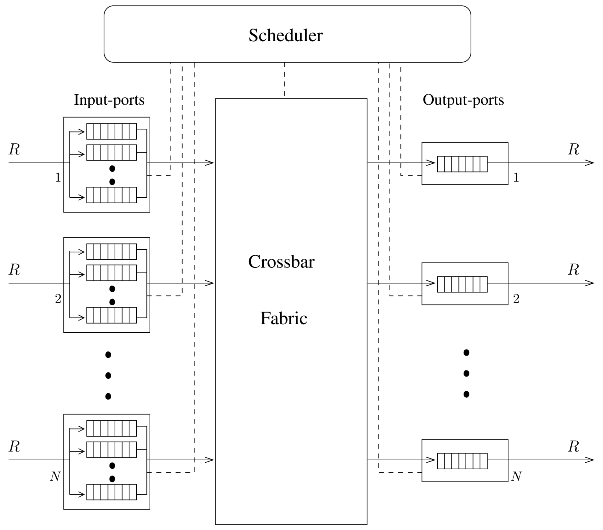

Fig. 1. Combined Input-Output-Queued Switch with Virtual Output Queuing. The switch fabric operates at rate $S \cdot R$, where $S$ is the speedup of the switch.

However, while packet-mode schedulers seem more attractive than cell-based switches, it is unknown whether they can provide the same strong performance guarantees. In particular, the objective of our paper is to determine whether they can emulate or mimic OQ switches with a speedup of two, or, for that matter, with any speedup.

We answer this question in the affirmative by presenting packet-mode schedulers for CIOQ switches that mimic an OQ switch. We further investigate the trade-off between the speedup of the CIOQ switch and its relative queuing delay.

This paper only considers "strong" deterministic performance guarantees. In particular, *OQ switch mimicking* with relative queuing delay $\mathcal{R}$ allows the CIOQ switch to deliver packets at most $\mathcal{R}$ time slots after they would have been delivered by an ideal OQ switch. This holds for arbitrary traffic patterns, including adversarial traffic. Therefore, this performance measure is entirely deterministic and worst case in nature. It is not subject to probabilistic or empirical assumptions on the incoming traffic that could become unsubstantiated.

## 1.1 Background

Previous work [12], [13] considers packet-mode scheduling in an *input-queued* (IQ) switch with crossbar fabric whose speedup is 1 and proves analytically that packet-mode IQ switches can achieve 100% throughput, provided that the input-traffic is well behaved; this matches earlier results on cell-based scheduling [14]. Ajmone Marsan et al. [12] also show that under low load and small packet size variance, packet-mode schedulers may achieve better average packet delay than cell-based schedulers. A different line of research used competitive analysis to evaluate packet-mode scheduling, when each packet has a weight representing its importance and a deadline until which it should be delivered from the switch [15].

The ability of a *cell-based* CIOQ switch to emulate an output-queued switch has been extensively investigated (e.g., [4], [5], [6], [7], [8], [9]). Chuang et al. [5] introduce the *critical cells first* (CCF) algorithm, which allows a CIOQ switch with speedup 2 to emulate (exactly) an output-queued switch. They also show that a CIOQ switch needs speedup of at least $2 - \frac{1}{N}$ in order to emulate an output-queued switch. To our knowledge, the ability of packet-mode CIOQ switches to emulate OQ switches was not investigated before.

Recently, Turner [16] studied packet-mode emulation in *buffered crossbar* switches. Essentially, these are CIOQ switches with additional buffers in the crosspoints: a cell arriving at input port $i$ and destined for output port $j$ is first buffered at input port $i$'s buffer, then it is sent to an $(i, j)$ *crosspoint buffer*, and finally, it is forwarded from the crosspoint buffer to output port $j$'s buffer. Turner shows that a buffered crossbar switch with speedup 2 and crosspoint buffers of size $5L_{\max}$ can mimic an output-queued switch with relative queuing delay of $(7/2)L_{\max}$ time slots. The algorithms rely on the fact that, unlike in CIOQ switches, the buffers at the crosspoints introduce orthogonality between the operations of input ports and output ports. This strong property simplifies the design of both cell-based schedulers [17] and packet-mode schedulers [16]. The algorithms proposed in [16] do not apply to the unbuffered switch fabric we study.

Our schedulers and their analysis rely on matrix decomposition techniques. Birkhoff von-Neumann matrix decomposition is used for scheduling when the arrival rate can be estimated [18], [19], [20]. Other matrix decomposition heuristics are employed in frame-based schedulers for optical switching and in satellite-switched time-division multiple-access (SS/TDMA) schedulers [21], [22], [23], [24]. These decompositions are not packet aware and may violate the contiguous delivery of cells corresponding to the same packet.

Weller and Hajek [24] show that a decomposition using maximal matchings requires at most twice as many scheduling decisions as a Birkhoff von-Neumann decomposition. This property forms the basis for the analysis of one of our algorithms (described in Theorem 8).

## 1.2 Our Results

We devise pipelined frame-based schedulers in which scheduling decisions are done at the frame boundaries. At each frame, a *demand matrix*, representing the total size of packets between each input-output pair, is decomposed into permutations that dictate the scheduling decisions in the next frame.

We show that, unlike a cell-based CIOQ switch, a packet-mode CIOQ switch *cannot* exactly emulate an OQ switch, or even achieve relative queuing delay smaller than $L_{\max}/2 - 3$ (where $L_{\max}$ is the maximum packet size[2]) *whatever the speedup is* (Theorem 1). However, once we allow for a bounded additional relative queuing delay, we find that a packet-mode CIOQ switch does not require a fundamentally higher speedup than a cell-based CIOQ switch. Moreover, we provide tradeoffs between the speedup of the switch and its relative queuing delay.

Specifically, a packet-mode CIOQ switch can mimic an OQ switch with a speedup $2L_{\max}$ and relative queuing delay of only $L_{\max} - 1$ time slots (Theorem 2). However, note that such a speedup is prohibitive in real-life switches.

We further show (Theorem 8) that a smaller speedup of $4 + O(\frac{1}{\mathcal{R}})$ suffices to ensure that a packet-mode CIOQ switch mimics an OQ switch with relative queuing delay

2. The *size* of a packet is measured by the number of time slots it takes to transmit the packet over the switch fabric.

of $\mathcal{R} = O(NL_{\max})$ time slots, where $N$ is the number of input (output) ports.

Finally, a speedup of $2 + O(\frac{1}{\mathcal{R}})$ suffices to ensure that a packet-mode CIOQ switch mimics an OQ switch with relative queuing delay $\mathcal{R} = O(NL_{\max})$ time slots. The relative queuing delay introduced by this algorithm is larger by a constant factor than the relative queuing delay incurred by the algorithm described in Theorem 8.

It is important to note that these algorithms rely on an underlying cell-mode OQ emulation algorithm (e.g., CCF).

We also consider mimicking an output-queued switch with a bounded buffer size $B$ at each output port. Extending the matrix decomposition techniques, we show (Corollary 12) that with a smaller speedup of $1 + O(\frac{1}{\mathcal{R}})$ and relative queuing delay $\mathcal{R} = O(B + NL_{\max})$, a packet-mode CIOQ mimics an OQ switch with buffer size $B$. The algorithm that provides this mimicking does not rely on an underlying cell-mode OQ emulation algorithm, and thus, may be more practical to implement in real-life switches.

We evaluate the performance of our schedulers through simulations, both under real-life traffic traces and under various stochastic traffic models. These simulations verify that, in practice, our schedulers perform significantly better than their theoretical bounds. Furthermore, we offer a simple heuristic that can be directly implemented in practice; our simulations show that this heuristic provides packet-mode emulation with small relative queuing delay and speedup less than 2.

## 2 Preliminaries

### 2.1 Packet-Mode CIOQ Switch

We consider an $N \times N$ *packet switch* that routes packets arriving at $N$ input ports at rate $R$ and destined for $N$ output ports working at the same rate $R$ (see Fig. 1). *Packets* of variable size arrive at the input ports and leave the output ports in a time-slotted manner; namely, all the switch external lines are synchronized (as typically assumed in the literature [5], [14], [25]). For convenience, we refer to each part of a packet that is transmitted during a single slot as a fixed-size *cell*. Note that packets are not really segmented into cells, and the cell abstraction is used solely to simplify the description.

The packet size is measured in cell units, where the minimal packet size is one cell and the maximum packet size is $L_{\max}$ cells. All cells of the same packet arrive at the switch contiguously on the same input port and are destined for the same output port. Therefore, we refer to a packet simply as a sequence of cells and assume that its size is known upon arrival of its first cell (e.g., the total size is written in the header).

For every cell $c$, we denote by $orig(c)$ and $dest(c)$ the input port at which $c$ arrives and the output port for which $c$ is destined. In addition, $packet(c)$ denotes the packet that corresponds to cell $c$; $first(p)$, $last(p)$ are the first and last cells of packet $p$.

In a CIOQ switch with speedup $S$, packets arriving at rate $R$ are first buffered in the input side and then forwarded over the switch fabric to the output side as dictated by a scheduling algorithm. Packets that arrive at input port $i$ and are destined for output port $j$ are stored in the input side of the switch in a separate buffer $VOQ_{ij}$, which is called *virtual output queue*.

The switch fabric operates at rate $S \cdot R$, where $S$ is the *speedup* of the switch, implying that the switch has $S$ scheduling opportunities (or scheduling decisions) every time slot.[3] When $S > 1$, some buffering should be done also in the output side of the switch.

A packet-mode CIOQ switch ensures that if a packet $p$ from input port $i$ to output port $j$ consists of the cells $(c_1, \ldots, c_\ell)$, then after cell $c_1$ is transmitted across the switch fabric, no cells of packets other than $p$ are transmitted from input port $i$ or to output port $j$ until cell $c_\ell$ is transmitted. Naturally, cells of the same packet are transmitted in order.

It is possible that some input port $i$ starts transmitting cells of a packet $p$ before all the cells of packet $p$ arrived at the switch. Since the speedup of the switch is typically greater than 1, this may cause the switch to underutilize its speedup. For example, suppose that the first cell $c_1$ of a packet $p = (c_1, c_2, \ldots, c_\ell)$ arrives at input port $i$ at time slot $t$ and is immediately sent to output port $j$ in the first scheduling opportunity of time slot $t$. Since cell $c_2$ arrives at the switch only at time slot $t+1$, no cells can be sent from input port $i$ or to output port $j$ for the next $S-1$ scheduling opportunities (even if there are cells of other packets in one of the relevant buffers).

### 2.2 Mimicking a Packet-Mode OQ Switch

The first requirement from a packet-mode OQ switch is to ensure that cells of the same packet are contiguous, that is, cells of the same packet should leave the switch one after the other with no interleaving of cells from other packets. We denote by $t_{OQ}(c)$ the time slot at which cell $c$ is delivered by the output-queued switch. Therefore, for any packet $p = (c_1, \ldots c_\ell)$, $t_{OQ}(c_i) = t_{OQ}(c_j) + (i - j)$ for $1 \le j \le i \le \ell$.

As in the previous works on OQ switch mimicking (e.g., [5], [17], [26]), a packet-mode OQ switch should also provide a relaxed notion of a *first-come-first-serve* (FCFS) discipline. If the last cell of packet $p$ arrives at the switch before the first cell of packet $p'$ and both packets share the same output port, then all cells of packet $p$ should leave the switch before the cells of packet $p'$. We denote this partial order of packets by $p \prec p'$.

Finally, (packet-mode) OQ switches should also be *work-conserving*: Namely, if a cell is pending for output port $j$ at time slot $t$, then some cell leaves the switch from output port $j$ at time slot $t$ [5], [6], [27].

We now define the stability criterion used in this paper as follows:

**Definition 1 (Output-Queued Switch Mimicking [26]).** *A switch* mimics *an output-queued switch with relative queuing delay $\mathcal{R}$ if, under identical input traffic, every cell leaves the switch at most $\mathcal{R}$ time slots after it would have left the output-queued switch, where $\mathcal{R}$ is a constant independent of the elapsed time.*

We say that a switch *emulates* an OQ switch if it mimics it without relative queuing delay (i.e., $\mathcal{R} = 0$) [5]. Since an output-queued switch is work-conserving, it implies that any switch that emulates an output-queued switch is also work-conserving.

3. For nonintegral speedup values, the speedup $S$ is the average number of such scheduling decisions per time slot, where at each time slot, the switch makes between $\lfloor S \rfloor$ and $\lceil S \rceil$ scheduling decisions [3].

## 3 Simple Upper and Lower Bounds on the Relative Queuing Delay

### 3.1 The Lower Bound

We show that a packet-mode CIOQ switch cannot mimic with small relative queuing delay an OQ switch, regardless of the CIOQ switch speedup. In particular, this result implies that a packet-mode CIOQ cannot exactly emulate an OQ switch, *whatever the speedup used*. This runs against the conventional wisdom that "speedup $N$ solves every problem."

**Theorem 1.** *For any speedup $S$, a packet-mode CIOQ switch with speedup $S$ cannot mimic an OQ switch with relative queuing delay $\mathcal{R} < L_{\max}/2 - 3$ time slots.*

**Proof.** Assume toward a contradiction that the CIOQ switch mimics an OQ switch with relative queuing delay $\mathcal{R} < L_{\max}/2 - 3$, and consider the following traffic, consisting of only three packets: At time slot 1, packet $p_1$ of size $L_{\max}$ arrives at input port 1, destined for output port 1. At time slot $\mathcal{R} + 2$, packet $p_2$ of size 1 arrives at input port 2, destined for output port 1. At time slot $\mathcal{R} + 3$, packet $p_3$ of size $L_{\max}$ arrives at input port 2, destined for output port 2.

At time slot 1, packet $p_1$ is the only packet destined for output port 1; since the OQ switch is work-conserving, the first cell of $p_1$ is delivered by the OQ switch at time slot 1, implying that it must be delivered by the CIOQ switch by time slot $\mathcal{R} + 1$. Packet scheduling restricts the switch from delivering cells of other packets to output port 1 until the last cell of packet $p_1$ is delivered. Since the last cell of packet $p_1$ arrives at the switch at time slot $L_{\max}$, output port 1 is busy handling $p_1$ at least until time slot $L_{\max}$.

The same argument shows that the first cell of packet $p_3$ is delivered to output port 2 at time slot $\mathcal{R} + 3$, and input port 2 is busy handling $p_3$ at least until time slot $L_{\max} + \mathcal{R} + 2$.

Since $L_{\max} > \mathcal{R} + 3$, packet $p_2$ cannot be delivered to output port 1 until time slot $L_{\max} + \mathcal{R} + 2$. However, packet $p_2$ is delivered by the OQ switch in time slot $L_{\max} + 1$, implying that its relative queuing delay is at least $\mathcal{R} + 1$, contradicting the assumption. □

This result holds since the CIOQ switch waits for the cells of the different packets to arrive, and in this situation, the switch degrades to work at the external line rate (i.e., with $S = 1$), as an IQ switch. Theorem 1 is, therefore, consistent with the known result that IQ switches with speedup 1 cannot emulate output-queued switches [5].

### 3.2 The Upper Bound

We now show that a CIOQ switch can mimic an OQ switch with relative queuing delay of $L_{\max} - 1$ time slots provided it has speedup $2L_{\max}$. The algorithm closely follows the CCF algorithm [5], which emulates (precisely) a cell-based OQ switch with speedup $S = 2$.

Intuitively, multiplying the speedup by the maximum packet size $L_{\max}$ reduces the problem of packet-mode switching to cell-based switching: each cell-based scheduling decision can be mapped to $L_{\max}$ contiguous packet-mode scheduling decisions, implying that a packet can be transmitted contiguously. In addition, a relative queuing delay of $L_{\max} - 1$ time slots allows the scheduler to wait until a packet fully arrives at the switch before it is scheduled. The following theorem captures this simple result:

**Theorem 2.** *A packet-mode CIOQ switch with speedup $S = 2L_{\max}$ can mimic an OQ switch with relative queuing delay of $L_{\max} - 1$ time slots.*

**Proof.** For each time slot $t$, let traffic $\mathcal{T}(t)$ be the collection of cells that arrive at the switch by time slot $t$ and let $\mathcal{T}'(t) \subseteq \mathcal{T}(t)$ be a traffic consisting only of cells in $\mathcal{T}(t)$ that are the first cells of their corresponding packets. Denote by $\mathrm{CCF}'(c)$ the time slot in which the CCF algorithm with speedup $S = 2$ schedules a cell $c$ of traffic $\mathcal{T}'(t)$ over the switch fabric, and let $t'_{OQ}(c)$ be the time slot in which $c$ leaves a cell-based OQ switch that handles traffic $\mathcal{T}'$.

The *packet-mode CCF algorithm* (PM-CCF) simulates the behavior of a cell-based CCF: For each packet $p$ of traffic $\mathcal{T}(t)$, PM-CCF forwards the entire packet $p$ contiguously over the switch fabric in time slot $t_{PM\text{-}CCF} = \mathrm{CCF}'(first(p)) + L_{\max} - 1$.

Since the cell-based CCF works with speedup $S = 2$, for each time slot $t$, there are at most two cells that share the same input or output port and are forwarded over the switch fabric by the cell-based CCF in time slot $t$. PM-CCF works correctly since it has $2L_{\max}$ scheduling opportunities at each time slot, and therefore, can schedule the packets corresponding to these two cells entirely in the same time slot $t$. In addition, the contiguous arrival of packets at the input ports ensures that packet $p$ has fully arrived to the switch by time slot $\mathrm{CCF}'(first(p)) + L_{\max} - 1$.

For each cell $c$ of traffic $\mathcal{T}$, $t_{OQ}(c)$ denotes the time slot in which $c$ leaves the packet-mode OQ switch. Note that $t_{OQ}(c) \geq t_{OQ}(first(packet(c))) \geq t'_{OQ}(first(packet(c)))$ because cells corresponding to the same packet are delivered in order and traffic $\mathcal{T}'$ is a subset of traffic $\mathcal{T}$. Since the cell-based CCF emulates cell-based OQ switch, it follows that for each cell $c$ of traffic $\mathcal{T}$:

$$\begin{aligned} t_{OQ}(c) &\geq t'_{OQ}(first(packet(c))) \\ &\geq \mathrm{CCF}'(first(packet(c))) \\ &= t_{PM\text{-}CCF}(c) - (L_{\max} - 1). \end{aligned}$$

This implies that every cell $c$ can be delivered from a CIOQ switch with packet-mode CCF at time slot $t_{OQ}(c) + L_{\max} - 1$, and the claim follows. □

This result only demonstrates the possibility of OQ mimicking with bounded delay, since speedup $2L_{\max}$ is unreasonable in practical switches. Note that had it been a cell-mode emulation algorithm with better speedup $S < 2$, OQ mimicking could have been obtained with speedup $SL_{\max}$ by following the proof of Theorem 2. The rest of the paper shows how to mimic an OQ switch with significantly smaller speedup.

## 4 Toward Trade-Offs between Speedup and Relative Queuing Delay

Our scheduling algorithms operate in a frame-based pipelined manner, with scheduling decisions done only at the frame boundaries. At each frame boundary, the algorithms first construct *demand matrices*, and then decompose these matrices into permutations (or subpermutations). The algorithms satisfy the demands by scheduling the cells in the next frame according to the resulting permutations.

The algorithms and their analysis rely on some results of matrix theory, which are presented in the following section. Then, we give some general results, which lay the groundwork for our scheduling algorithms described in Sections 5 and 6.

### 4.1 Matrix Decomposition

**Definition 2.** *A* permutation *$P$ is a 0-1 matrix such that the sum of each row and the sum of each column are exactly 1. A* subpermutation *$P$ is a 0-1 matrix such that the sum of each row and the sum of each column are at most 1.*

For simplicity, we refer to subpermutations as permutations in the rest of the paper. The following definition captures the fact that the number of cells that should be scheduled from a single input port or to a single output port is bounded:

**Definition 3.** *A matrix $A \in \mathbb{N}^{N\times N}$ is $C$-*bounded *if the sum of each row and each column in $A$ is at most $C$.*

A classical result says that any $C$-bounded matrix $A$ can be decomposed into $C$ permutations, whose sum dominates $A$:

**Theorem 3 (BIRKHOFF VON-NEUMANN DECOMPOSITION [28], [29], [30]).** *If a matrix $A \in \mathbb{N}^{N\times N}$ is $C$-bounded, for some integer $C$, then there are $C$ permutations $P_1, \dots, P_C$ such that $A \leq \sum_{i=1}^{C} P_i$.*

The Birkhoff von-Neumann decomposition implies that every $C$-bounded demand matrix can be scheduled, cell by cell, in $C$ scheduling opportunities (or equivalently, in $\lceil C/S \rceil$ time slots) when permutation $P_i$ dictates the scheduling in opportunity $i$. However, such a scheduling may violate the packet-mode restrictions, since there is no relation between adjacent permutations in the sequence.

### 4.2 Common Properties of Our Scheduling Algorithms

Our schedulers operate by constructing a demand matrix at each frame boundary and then decomposing this matrix in order to determine the scheduling decisions for the next frame. The relative queuing delay of the schedulers corresponds to the size of the frame, while the speedup of the switch is determined by the ratio between the frame size and the number of permutations obtained in the decomposition.

A key insight is that packet-mode OQ switches can be implemented by a *push-in-first-out (PIFO)* cell-based OQ switch. In such OQ switches, arriving cells are placed in an arbitrary location in their destination's buffer, and the switch always outputs the cells at the head of its buffers [5]. The PIFO policy is an extension of the first-in-first-out (FIFO) policy that can also implement Quality-of-Service–aware (QoS-aware) algorithms, such as WFQ and strict priority:

**Lemma 4.** *A packet-mode OQ switch can be implemented by a PIFO cell-based OQ switch.*

**Proof.** Consider the following PIFO cell-based OQ scheduler: The first cell of a packet $p$ arriving at the switch is placed at the end of the relevant OQ switch buffer. Each consecutive cell $c_i$ of packet $p$ is placed immediately after cell $c_{i-1}$; in each time slot, the cell at the head of the buffer departs from the switch.

Since cells of the same packet are placed one after the other in the buffer, they leave the OQ switch contiguously. In addition, if $p \prec p'$, then the last cell of packet $p$ is placed in the buffer before the first cell of packet $p'$, implying that packet $p$ is served before packet $p'$. Thus, this is a valid packet-mode OQ scheduler and the claim follows. □

Note that, using the CCF algorithm, a cell-based CIOQ switch with speedup $S = 2$ can emulate cell-based OQ switch with *any* PIFO discipline [5], and in particular, the above-mentioned discipline. Our algorithms use CCF in order to construct the demand matrix of each frame. It is important to note that we could use any other algorithm that provides PIFO cell-based emulation.

Let $\mathrm{CCF}(c)$ be the time slot in which a cell $c$ is forwarded over the switch fabric by the CCF algorithm. Clearly, $\mathrm{CCF}(c) \leq t_{OQ}(c)$. We have the next lemma:

**Lemma 5.** *If a scheduling algorithm* ALG *schedules the cell $last(p)$ of every packet $p$ by time slot $\mathrm{CCF}(last(p)) + \delta$, then the maximum relative queuing delay of* ALG *is at most $\delta + L_{\max} - 1$, where $L_{\max}$ is the maximum packet size.*

**Proof.** Consider a cell $c$, let $k$ be its place in $packet(c)$ and $\ell$ be the size of $packet(c)$. The contiguous packet delivery in the output-queued switch dictates that $t_{OQ}(c) = t_{OQ}(last(packet(c)) - (\ell - k)$. Let $t_{ALG}(c)$ be the time slot in which ALG forwards cell $c$ over the switch fabric. Since both ALG and CCF forward the cells of $packet(c)$ in their order within the packet,

$$\begin{aligned} t_{ALG}(c) &\leq t_{ALG}(last(packet(c)) \\ &\leq \mathrm{CCF}(last(packet(c)) + \delta \\ &\leq t_{OQ}(last(packet(c)) + \delta \\ &= t_{OQ}(c) + \delta + \ell - k \\ &\leq t_{OQ}(c) + \delta + \ell - 1 \\ &\leq t_{OQ}(c) + \delta + L_{\max} - 1. \end{aligned}$$

Therefore, every cell $c$ is in the output side of the switch by time slot $t_{OQ}(c) + \delta + L_{\max} - 1$, and therefore, ALG can output cell $c$ from the CIOQ switch at time slot $t_{OQ}(c) + \delta + L_{\max} - 1$. Note that ALG does not transmit two cells $c, c'$ at the same time slot from the same output port, since $t_{OQ}(c) + \delta + L_{\max} - 1 = t_{OQ}(c') + \delta + L_{\max} - 1$ implies that $t_{OQ}(c) = t_{OQ}(c')$, contradicting the definition of an output-queued switch. □

We devise a frame-based scheduler in which the demand matrix in each frame is built according to the times at which the underlying CCF algorithm forwards cells over the switch fabric. Packets that are not fully forwarded by the CCF algorithm until the frame boundary are queued in the input side of the switch until the next frame, as captured by the next definition given below.

**Definition 4.** *For every input port $i$, output port $j$, frame size $T$, and frame number $k > 0$, the* set of eligible cells of frame $k$, *denoted by $a_{ij}(T,k)$, includes all cells $c \notin \bigcup_{k'<k} a_{ij}(T,k')$ such that $\mathrm{CCF}(last(packet(c))) \leq kT$. By convention, $a_{ij}(T,0) = \emptyset$.*

Note that by definition, all the cells of a packet $p$ are in the same set of eligible cells.

The following lemma bounds the number of cells, sharing an input port or an output port, which should be scheduled within the same frame:

**Lemma 6.** *For every input port $i$, output port $j$, frame size $T$, and frame number $k > 0$,*

$$\sum_{j'=1}^{N} \left|a_{ij'}(T,k)\right| \leq 2T + N(L_{\max} - 1) \text{ and}$$
$$\sum_{i'=1}^{N} \left|a_{i'j}(T,k)\right| \leq 2T + N(L_{\max} - 1).$$

**Proof.** Since CCF works with CIOQ switch with speedup 2, the number of cells $c$ that share the same input port (output port) and have been forwarded by CCF within frame $k$ (namely $(k-1)T < \mathrm{CCF}(c) \leq kT$) is at most $2T$.

Since in each virtual output queue $VOQ_{i,j}$, all cells of the same packet $p$ are stored one after the other, there is no cell of a different packet that is forwarded by CCF between cells of packet $p$. Therefore, only cells of one packet are in $a_{ij}(T,k)$ and were forwarded by CCF before time slot $(k-1)T$; we next bound the number of such cells.

Because the maximum packet size is $L_{\max}$ and the last cell of each packet was forwarded by the CCF after time slot $(k-1)T$, at most $L_{\max} - 1$ such cells share the same input port and the same output port. Thus, the number of such cells that share an input port (output port) is at most $N(L_{\max} - 1)$. This implies that both

$$\sum_{j'=1}^{N} \left|a_{ij'}(T,k)\right| \text{ and } \sum_{i'=1}^{N} \left|a_{i'j}(T,k)\right|$$

are bounded by $2T + N(L_{\max} - 1)$. □

Lemma 6 and Theorem 3 imply that the eligible cells of each frame can be scheduled within $2T + N(L_{\max} - 1)$ scheduling opportunities. Unfortunately, the Birkhoff-von Neuman decomposition does not ensure that the packet-mode scheduling constraints are satisfied, and therefore, cannot be used directly. For example, consider the matrix

$$A = [a_{ij}] = \begin{pmatrix} 3 & 1 & 2 & 0 \\ 0 & 2 & 2 & 2 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 0 & 3 \end{pmatrix}$$

in which, for example, element $a_{1,1}$ represents a single packet of size 3 and elements $a_{2,2}$,$a_{2,3}$,$a_{2,4}$ represent packets of size two. The following decomposition of $A$ into six permutations

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} +$$
$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

violates the packet-mode constraints: Contiguous transmission of packet $a_{1,1}$ requires that the first three permutations be scheduled contiguously. On the other hand, each permutation $i$ ($i \in \{1,2,3\}$) must also be adjacent to permutation $i+3$ in order to ensure contiguous transmission of packet $a_{2,i+1}$. These requirements cannot be satisfied simultaneously, since it implies that at least one permutation must be adjacent to three permutations.

## 5 Mimicking an Output-Queued Switch with Speedup $S \approx 4$

Our first algorithm decomposes the matrix of eligible cells differently, guaranteeing contiguous packet delivery but requiring twice as many scheduling opportunities. It uses the following class of permutations:

**Definition 5.** *A* maximal matching *for a matrix $A = [a_{ij}]$ is a permutation matrix $P = [p_{ij}] \leq A$ such that if $p_{ij} = 0$ and $a_{ij} > 0$, then there exist $i'$ such that $p_{i'j} = 1$ or $j'$ such that $p_{ij'} = 1$.*

Intuitively, a permutation $P \leq A$ is a maximal matching for a matrix $A$ if no element can be added to $P$, resulting in a matrix that is still a permutation and is dominated by $A$. Note that using maximal matching does not necessarily guarantee contiguous packet delivery.

The next theorem shows that if a matrix is decomposed by *any* sequence of maximal matchings, then the number of permutations needed is at most twice the number needed in Theorem 3. The decomposition procedure of a $C$-bounded matrix $A$ works iteratively: In each iteration $m$, a maximal matching $P(m)$ for the matrix $A(m-1)$ is found and then subtracted from $A(m-1)$ to form $A(m)$ (negative values are treated as zeros). The procedure stops when $A(m) = 0$. By Theorem 7, this happens after at most $2C - 1$ iterations, regardless of the choice of the maximal matching in each iteration, implying that the matrix $A$ is decomposed into less than $2C$ permutations.

**Theorem 7 ([24, Theorem 2.2]).** *For every $C$-bounded matrix $A \in \mathbb{N}^{N \times N}$, the decomposition procedure described above stops after at most $2C - 1$ iterations.*

Our decomposition algorithm works as follows: At each frame boundary, the algorithm counts the number of cells in each set $a_{ij}(T,k)$ and constructs a matrix $B(k) = [b_{ij}]$ accordingly (namely $b_{ij} = |a_{ij}(T,k)|$). Then, the algorithm repeatedly builds *maximal* matchings for matrix $B(k)$ and keeps contiguous packet delivery by keeping the matching between input port $i$ and output port $j$ for another iteration, if additional cells are pending from $i$ to $j$. (This procedure is sometimes called *exhaustive service matching* [31].) Example 1 demonstrates the behavior of this algorithm under a specific traffic pattern.

Since the algorithm uses only maximal matchings, Theorem 7 implies that the algorithm needs twice as many iterations as Birkhoff von-Newmann decomposition. In particular, for every frame size $T$, the algorithm needs at most $4T + 2N(L_{\max} - 1)$ iterations to complete. This implies that it can mimic an output-queued switch with a speedup arbitrarily close to 4, while attaining a relative queuing delay of $O(NL_{\max})$.

**Theorem 8.** *A packet-mode CIOQ switch with speedup*

$$S = 4 + \frac{2N(L_{\max} - 1) - 1}{T}$$

*can mimic an OQ switch with a relative queuing delay of* $2T + L_{\max} - 2$ *time slots.*

**Proof.** Fix a frame size $T$ and let $B(k) = [b_{ij}]$ be the $N \times N$ matrix such that $b_{ij} = |a_{ij}(T, k)|$. Lemma 6 implies that the sum of each row and each column of $B(k)$ is at most $2T + N(L_{\max} - 1)$.

Algorithm 1 works by repeatedly constructing *maximal* matchings $P$ for matrix $B(k)$. If a cell in the set $a_{ij}(T, k)$ is forwarded in some iteration of the algorithm, and there are more cells in $a_{ij}(T, k)$ to be forwarded, the algorithm keeps the matching between input port $i$ and output port $j$ for the next iteration. Therefore, cells of a specific set are forwarded contiguously. Definition 4 implies that Algorithm 1 forwards all the cells corresponding to a specific packet contiguously.

**Algorithm 1.** Coarse-Grained Maximal Matchings

```
Local Variables:
    $B$: matrix of values in $\mathbb{N}$, initially $B = B(k)$
    $P$: matrix of values in $\{0, 1\}$, initially 0

1:  procedure SCHEDULE(matrix $B$)
2:      while $B \neq 0$ do
3:          for all $P[i][j]$ do
4:              if $P[i][j] = 1$ and $B[i][j] = 0$ then
5:                  $P[i][j] = 0$
6:              end if
7:          end for
8:          $P :=$ MAX-MATCH$(B, P)$
                        ▷ returns a maximal matching
                          of $B$ that dominates $P$
9:          for all $P[i][j]$ do
10:             if $P[i][j] = 1$ and $B[i][j] > 0$ then
11:                 forward a cell from input $i$ to output $j$
12:             end if
13:         end for
14:         $B := B - P$
15:         for all $B[i][j]$ do     ▷ avoid negative values in $B$
16:             $B[i][j] := \max\{B[i][j], 0\}$
17:         end for
18:     end while
19: end procedure

20: matrix procedure MAX-MATCH(matrix $B$, matrix $P$)
21:     while there are $i, j$ such that $B[i][j] = 1$ and
                $\sum_{j'=1}^{N} P[i][j'] = 0$ and $\sum_{i'=1}^{N} P[i'][j] = 0$ do
22:         $P[i][j] = 1$
23:     end while
24:     return $P$
25: end procedure
```

All matchings used by Algorithm 1 are maximal and the sum of each column and each row in $B(k)$ is at most $2T + N(L_{\max} - 1)$. By Theorem 7, Algorithm 1 needs at most

$$2 \cdot (2T + N(L_{\max} - 1)) - 1 = 4T + 2N(L_{\max} - 1) - 1$$

iterations to complete. Thus, with speedup

$$4 + \frac{2N(L_{\max} - 1) - 1}{T},$$

the algorithm schedules all cells corresponding to $B(k)$ within the next frame, that is, by time slot $(k + 1)T$.

Consider some packet $p$. Definition 4 implies that if the last cell of $p$ is in $a_{ij}(T, k)$, then $CCF(last(p)) > (k - 1)T$. Since Algorithm 1 schedules $last(p)$ by time slot $(k + 1)T$, it follows that the relative queuing delay of $last(p)$ is at most $2T - 1$. By Lemma 5, the relative queuing delay is at most $2T + L_{\max} - 2$. □

The following example demonstrates Algorithm 1 under a specific traffic pattern:

**Example 1.** Consider a $5 \times 5$ switch with maximum packet length $L_{\max} = 3$ and frame size $T = 60$, and consider the following arrival traffic pattern (depicted in Fig. 2a):

Input ports 1 and 4 receive continuously packets of size 3 destined for output 1, input port 2 receives alternately one packet of size 1 destined for output 1 and one packet of size 2 destined for output 2, input port 3 receives alternately one packet of size 1 destined for output 2 and one packet of size 2 destined for output 1, and input port 5 receives alternately 10 packets of size 3 destined for output 1 and 10 packets of size 3 destined for output 2.

The total demand matrix of the first frame (that is, time interval $[0, 59]$) is

$$\begin{pmatrix} 60 & 0 & 0 & 0 & 0 \\ 20 & 40 & 0 & 0 & 0 \\ 40 & 20 & 0 & 0 & 0 \\ 60 & 0 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 0 \end{pmatrix}.$$

However, the demand matrix of eligible cells (that is, cells $c$ with $\mathrm{CCF}(last(packet(c))) < 60$) is

$$\begin{pmatrix} 30 & 0 & 0 & 0 & 0 \\ 10 & 40 & 0 & 0 & 0 \\ 20 & 19 & 0 & 0 & 0 \\ 30 & 0 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 0 \end{pmatrix}.$$

Suppose that Algorithm 1 picks the following maximal matching in time slot 60:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

then, this matching is kept for 30 scheduling decisions until there are no more cells to forward from input port 1 to output port 1. Because at this time, there are still 10 cells to forward from input port 2 to output port 2, the next matching must keep this pair of nodes matched for the next 10 scheduling decisions, for example,

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The rest of the decomposition is as follows (see illustration in Fig. 2b):

$$20\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} + 10\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} +$$
$$19\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} + 11\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} +$$
$$20\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Note that Theorem 8 implies that the switch may need a speedup of $S = 4 + \frac{2N(L_{\max}-1)-1}{T} = 4 + \frac{19}{60} = 4.32$ in order to deal with all possible traffic patterns; the specific traffic pattern in this example is scheduled within 120 scheduling decisions, thus, requiring a speedup of only two.

## 6 Mimicking an Output-Queued Switch with Speedup $S \approx 2$

Using a more fine-grained scheduler and the Birkhoff von-Neumann decomposition, we now show that a smaller speedup suffices, albeit with larger relative queuing delay.

The scheduler described in Theorem 8 schedules within each frame *all* eligible cells with the same origin and the same destination contiguously, namely it considers them as a single packet.

Algorithm 2 also works in a frame-based manner and deals only with eligible packets of each frame (see Definition 4). The algorithm starts by greedily *concatenating* packets from the same input port to the same output port into *megapackets* of size $(\alpha+1)L_{\max}$; $\alpha$ is a function of the frame size $T$ that will be determined later. The concatenation ensures that no original packet spans over multiple megapackets. This yields a demand matrix of megapackets of size $(\alpha+1)L_{\max}$, such that for every input-output pair, all except one megapacket have at least $\alpha L_{\max}$ bits (otherwise, an additional packet of size $\ell \le L_{\max}$ would have been added to the matrix).

**Algorithm 2.** Fine-Grained Decomposition

```
Local Variables:
    $B$: matrix of values in $\mathbb{N}$, initially $B = B(k)$
    $P[]$: vector of matrices of values in $\{0,1\}$, initially all 0
    $M$: matrix of values in $\mathbb{N}$, initially 0
    $\alpha$: integer

1: procedure SCHEDULE(matrix $B$, int $T$)
2:     $\alpha = \left\lceil \sqrt{\frac{T}{NL_{\max}}} \right\rceil$
3:     for all $A[i][j]$ do
4:         $M[i][j] = \left\lceil \frac{B[i][j]}{\alpha L_{\max}} \right\rceil$
5:     end for
6:     $P[] :=$ BVN-DECOMPOSITION ($M$)
                 ▷ returns a sequence of permutation
                   which is a Birkhoff von-Neumann
                   decomposition of $M$
7:     for $x = 0; x < P.size; x := x + 1$ do
                 ▷$P.size$ is the size of the vector $P[]$
8:         SCHED-PERMUTATION($P[x], \alpha$)
                 ▷ Each invocation of SCHED-PERMUTATION
                   takes $(\alpha+1)L_{\max}$ time slots
9:     end for
10: end procedure

11: procedure SCHED-PERMUTATION(permutation $Q$,
        int $\alpha$)
12:     for $y = 0; y < (\alpha+1)L_{\max}; y := y + 1$ do
                 ▷ Each iteration corresponds to a single
                   time slot
13:         for all $Q[i][j]$ do
14:             $c :=$ The cell at the head of $VOQ_{ij}$
                         ▷ NULL if $VOQ_{ij}$ is empty
15:             if $c \ne NULL$ and $Q[i][j] = 1$ then
16:                 if $first(packet(c)) = c$ and $packet(c).size >$
                        $(\alpha+1)L_{\max} - y$ then
17:                     $Q[i][j] := 0$
                         ▷ Packet cannot be fully
                           scheduled
18:                 else
19:                     forward cell $c$ from input $i$ to output $j$
20:                 end if
21:             end if
22:         end for
23:     end for
24: end procedure
```

Finally, the demand matrix of megapackets is optimally decomposed, using Birkhoff von-Neumann. Each permutation is held for $(\alpha+1)L_{\max}$ consecutive scheduling decisions, to ensure contiguous packet delivery.

The next theorem describes this algorithm in detail and proves that it works with speedup $S \approx 2$ and relative queuing delay $O(NL_{\max})$, by taking

$$\alpha = \left\lceil \sqrt{\frac{T}{NL_{\max}}} \right\rceil.$$

A detailed example of this algorithm under a specific traffic pattern follows.

**Theorem 9.** *A packet-mode CIOQ switch with speedup*

$$S = 2 + 6\sqrt{\frac{NL_{\max}}{T}},$$

*where $T \geq NL_{\max}$, can mimic an OQ switch with relative queuing delay of $2T + L_{\max} - 2$ time slots.*

**Proof.** Fix a frame size $T \geq NL_{\max}$ and focus on a frame $k$. Let the matrix $B((\alpha+1)L_{\max}, k) = [b((\alpha+1)L_{\max}, k)_{ij}]$ count the number of megapackets. Because the concatenation is greedy,

$$[b(\alpha+1)L_{\max}, k)_{ij}] = \left\lceil \frac{|a(T,k)_{ij}|}{\alpha L_{\max}} \right\rceil.$$

We first bound the sum of each row and each column of the matrix $B((\alpha+1)L_{\max}, k)$. Consider some row $i$ of the matrix (the proof for a column is analogous):

$$\begin{aligned}
\sum_{j=1}^{N} b((\alpha+1)L_{\max}, k)_{ij} &= \sum_{j=1}^{N} \left\lceil \frac{|a(T,k)_{ij}|}{\alpha L_{\max}} \right\rceil \\
&\leq \sum_{j=1}^{N} \left( \frac{|a(T,k)_{ij}|}{\alpha L_{\max}} + 1 \right) \\
&= N + \sum_{j=1}^{N} \frac{|a(T,k)_{ij}|}{\alpha L_{\max}} \\
&= N + \frac{1}{\alpha L_{\max}} \sum_{j=1}^{N} |a(T,k)_{ij}| \\
&\leq N + \frac{2T + N(L_{\max}-1)}{\alpha L_{\max}},
\end{aligned}$$

where the last inequality follows by Lemma 6.

Theorem 3 implies that the matrix $B(((\alpha+1)L_{\max}, k)$ can be decomposed into at most

$$N + \frac{2T + N(L_{\max}-1)}{\alpha L_{\max}}$$

permutations. The algorithm forwards contiguously all the megapackets (and thus, all the packets) by holding each permutation for $(\alpha+1)L_{\max}$ consecutive scheduling decisions. The number of scheduling decisions needed is therefore bounded by

$$\begin{aligned}
&(\alpha+1)L_{\max}\left(N + \frac{2T + N(L_{\max}-1)}{\alpha L_{\max}}\right) \\
&\quad = (\alpha+1)L_{\max}N + (1 + 1/\alpha)(2T + N(L_{\max}-1)) \\
&\quad = 2T + \frac{2T}{\alpha} + (\alpha+1)L_{\max}N + N(L_{\max}-1) + \\
&\qquad + \frac{N(L_{\max}-1)}{\alpha} \leq 2T + \frac{2T}{\alpha} + 2\alpha L_{\max}N,
\end{aligned}$$

where the last inequality holds when $\alpha \geq 1 + \sqrt{2}$.

Thus, with a speedup of $2 + \frac{2}{\alpha} + \frac{2\alpha L_{\max}N}{T}$, the algorithm schedules all cells corresponding to frame $k$ within the next frame. This implies that for each packet $p$, the maximum relative queuing delay of cell $last(p)$ is less than two frame sizes, namely at most $2T - 1$ time slots. By Lemma 5, the maximum relative queuing delay is at most $2T + L_{\max} - 2$.

Taking

$$\alpha(T) = \left\lceil \sqrt{\frac{T}{NL_{\max}}} \right\rceil$$

yields a speedup of at most

$$2 + 4\sqrt{\frac{NL_{\max}}{T}} + 2\frac{NL_{\max}}{T} \leq 2 + 6\sqrt{\frac{NL_{\max}}{T}}$$

with relative queuing delay of at most $2T + L_{\max} - 2$. $\square$

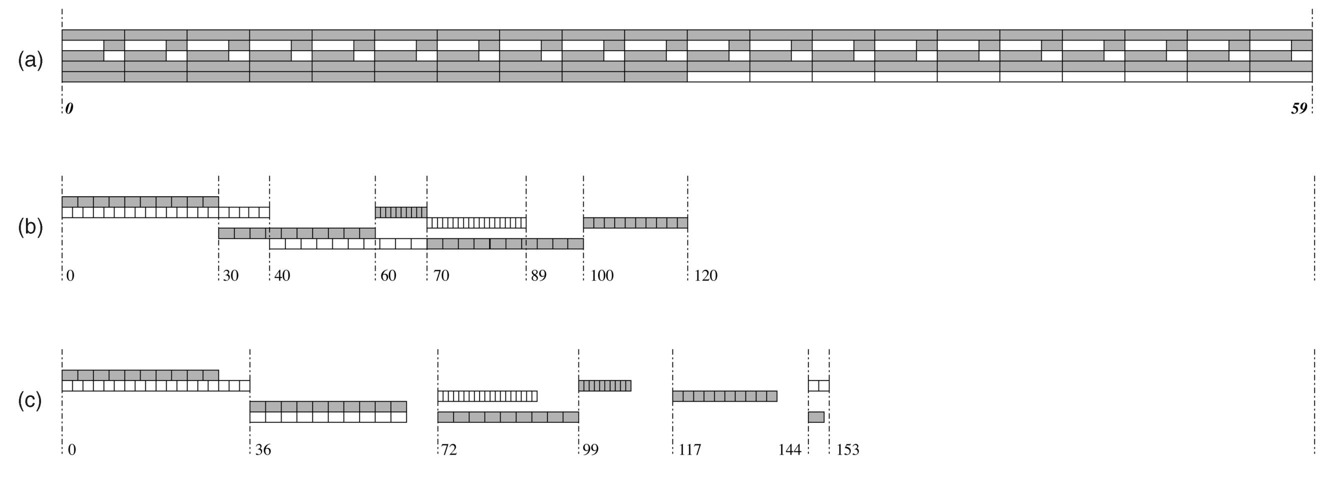

Fig. 2. Outline of Examples 1 and 2. Each row represents an input port. White packets are destined for output port 1, and gray packets are destined for output port 2. (a) The arrival pattern described in Examples 1 and 2; the numbers denote time-slots. (b) The scheduling decisions, within the corresponding frame, of Algorithm 1 as a response to the traffic depicted in (a); the numbers denote scheduling decisions (in this example, there are four scheduling decisions within each time slot). (c) The scheduling decisions, within the corresponding frame, of Algorithm 2 as a response to the traffic depicted in (a); the numbers denote scheduling decisions.

Note that this algorithm works with speedup smaller than Algorithm 1 when the frame size $T$ is larger than $\frac{2}{7-3\sqrt{5}}NL_{\max} = 6.85NL_{\max}$ time slots.

We next demonstrate the behavior of Algorithm 2.

**Example 2.** Consider a $5 \times 5$ switch with maximum packet length $L_{\max} = 3$ and frame size $T = 60$, and the arrival traffic pattern used in Example 1 (depicted in Fig. 2a). Recall that the demand matrix of eligible cells is

$$\begin{pmatrix} 30 & 0 & 0 & 0 & 0 \\ 10 & 40 & 0 & 0 & 0 \\ 20 & 19 & 0 & 0 & 0 \\ 30 & 0 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 0 \end{pmatrix}.$$

For $\alpha = \sqrt{\frac{T}{NL_{\max}}} = \sqrt{\frac{60}{5\cdot 3}} = 2$, the megapacket size is $(\alpha+1)L_{\max} = 9$. Thus, the demand matrix in terms of megapackets is

$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 2 & 5 & 0 & 0 & 0 \\ 3 & 3 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \\ 4 & 4 & 0 & 0 & 0 \end{pmatrix}.$$

This matrix is decomposed to the following permutations yielding the scheduling decisions described in Fig. 2c:

$$4\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + 4\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} +$$

$$3\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} + 2\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} +$$

$$3\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + 1\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

For this specific example, a speedup $S = 153/60 = 2.55$ suffices for the switch to handle the incoming traffic. However, under these settings, in order to handle any traffic pattern, the switch needs a speedup $S = 2 + \frac{2}{\alpha} + \frac{2\alpha L_{\max} N}{T} = 4$; one can reduce the speedup of the switch by taking a larger frame size. Furthermore, it is important to note that although in this specific example Algorithm 1 outperforms this algorithm, this is not the case in a worst-case traffic and larger frame size.

## 7 Mimicking an OQ Switch with Bounded Buffers

We next show that smaller speedup suffices for mimicking an OQ switch with bounded output buffer $B$. Intuitively, the reason for this better performance is that an OQ switch with bounded buffers cannot handle all incoming traffic patterns without dropping cells. Therefore, by using the extra information about the legal incoming traffic patterns, the CIOQ switch can optimize its scheduling decisions, resulting in a simpler and more efficient scheduling algorithm.

Unlike the previous algorithms, algorithms for bounded mimicking do not rely on the CCF algorithm, and use the following definition and lemma, which are adapted from Definition 4 and Lemma 6:

**Definition 6.** *For every input port $i$, output port $j$, frame size $T$, and frame number $k > 0$, the* set of eligible cells of frame $k$, *denoted by $a_{ij}(T,k)$, is the set of cells $c$ that is delivered successfully by the output-queued switch, $c \notin \bigcup_{k'<k} a_{ij}(T,k')$, and all cells $c' \in packet(c)$ arrive at the switch before time slot $kT$. By convention, $a_{ij}(T,0) = \emptyset$.*

As in Definition 4, all the cells of each packet $p$ are in the same set of eligible cells. The next lemma bounds the size of these sets, showing that the incoming traffic does not oversubscribe any output port by more than a bounded number of eligible cells.

**Lemma 10.** *For every input port $i$, output port $j$, frame size $T$, and frame number $k > 0$,*

$$\sum_{j'=1}^{N} |a_{ij'}(T,k)| \le T + B + N(L_{\max} - 1) \text{ and}$$
$$\sum_{i'=1}^{N} |a_{i'j}(T,k)| \le T + B + N(L_{\max} - 1).$$

**Proof.** Clearly, at most $T$ cells arrive at each input port between time slots $(k-1)T$ and $kT$. We next show that at most $T+B$ cells arrive between time slots $(k-1)T$ and $kT$, destined for a single output port $j$, and are successfully delivered by the OQ switch.

Assume toward a contradiction that $\ell_1 > T + B$ cells destined for output port $j$ arrive at the switch within frame $k$ and are not dropped by the OQ switch. Let $\ell_2 \ge 0$ be the number of cells stored in the buffer of output port $j$ in time slot $(k-1)T$. At most $T$ cells are delivered from output port $j$ between time slots $(k-1)T$ and $kT$; hence, the number of cells that are stored in the buffer by the end of frame $k$ is at least $\ell_1 + \ell_2 - T > B$ cells, contradicting the fact that the buffer size is $B$.

Since all cells of the same packet $p$ arrive at the switch contiguously, only cells of one packet are in $a_{ij}(T,k)$ and arrived at the switch before time slot $(k-1)T$. Since the maximum packet size is $L_{\max}$ and the last cell of each packet arrives after time slot $(k-1)T$, the number of such cells that share the same input port and the same output port is bounded by $L_{\max} - 1$. Thus, the number of such cells that share the same input port (output port) is bounded by $N(L_{\max} - 1)$, and the sum is, therefore, bounded by $T + B + N(L_{\max} - 1)$. □

In order to mimic an output-queued switch, the CIOQ switch drops all cells that are dropped by the OQ switch. By employing Lemma 10 in the proofs of Theorems 8 and 9, we get the following results:

**Corollary 11.** *A packet-mode CIOQ switch with speedup*

$$S = 2 + \frac{2B + 2N(L_{\max} - 1) - 1}{T}$$

*can mimic an OQ switch with buffer size B with a relative queuing delay of $2T + L_{\max} - 2$ time slots.*

**Corollary 12.** *A packet-mode CIOQ switch with speedup*

$$S = 1 + 3\sqrt{\frac{NL_{\max}}{T}} + \frac{2B}{T}$$

*can mimic an OQ switch with buffer size B with a relative queuing delay of $2T + L_{\max} - 2$ time slots, where $T \geq NL_{\max}$.*

## 8 Simulation Results

In this section, we show that, in practice, Algorithm 1 outperforms its analytical worst-case bounds and achieves small relative queuing delay with small speedup. We also show that Algorithm 1 outperforms Algorithm 2 for practical frame sizes. Note that analytically Algorithm 2 is superior to Algorithm 1 only when the frame size $T$ is larger than $6.85NL_{\max}$, and therefore, its importance—showing that packet-mode emulation does not need additional speedup to cell-mode emulation—is mainly theoretical.

The results are obtained by conducting simulations under various synthetic and trace-driven traffic patterns.

### 8.1 Simulation Settings

Recall that when our algorithms work with a frame of size $T$, their relative queuing delay is bounded by $2T + L_{\max} - 2$. Thus, in order to demonstrate the trade-off between the speedup and the relative queuing delay, we measure the loss ratio of packets, for fixed incoming traffic characteristics, under various speedup and frame-size values. In this way, for each frame-size value, we find the minimal speedup in which no loss is observed.

Specifically, for each traffic pattern, we fix a certain speedup $S$ and a frame size $T$. For each frame $k$, our algorithms construct a demand matrix $B(k)$ and then decompose the demand matrix $B(k-1)$ of the previous frame into a sequence of scheduling decisions. Under the fixed speedup $S$, the algorithms schedule at most $S \cdot T$ of these scheduling decisions and drop all packets with cells in the remaining scheduling decisions. Finally, for each frame size, we compute the corresponding relative queuing delay upper bound and present the minimal speedup in which no packet drop is observed.

### 8.2 Simulations Based on Stochastic Traffic Patterns

We study the following three stochastic traffic patterns. These patterns were also used by Ajmone Marsan et al. [12] in order to investigate the performance of a packet-mode Input-Queued switch (with no speedup). It is important to note that our results are even stronger than real-life performance, since some of the traffic patterns are especially constructed in order to reflect starvation and unfairness due to the contiguous forwarding of large packets:

1. **Uniform traffic:** In this traffic pattern, packet sizes are chosen uniformly at random in the range $[1, 192]$. For each packet, its destination is chosen uniformly at random among all output ports. This uniform traffic setting is frequently used in simulations and stochastic analysis of switch performance.
2. **Spotted traffic:** Packet sizes are 100 cells with probability 0.5 and 3 cells with probability 0.5; packet destination is chosen according to the following $8 \times 8$ matrix; each input port $i$ chooses a destination uniformly at random among all destinations with entry 1 in row $i$:

$$\begin{pmatrix} 1&1&1&0&1&0&1&0\\ 0&1&0&1&1&1&0&1\\ 1&0&1&0&1&1&1&0\\ 1&1&0&1&0&1&0&1\\ 1&0&1&0&1&0&1&1\\ 0&1&0&1&0&1&1&1\\ 1&0&1&1&1&0&1&0\\ 0&1&1&1&0&1&0&1 \end{pmatrix}.$$

   Since the matrix is doubly stochastic and the sum of each row (column) is five, it implies that each input port sends packets to five output ports, and each output port receives packets from five input ports. This specific traffic matrix aims to highlight starvation and loss of throughput due to the contiguous forwarding of large packets [12].
3. **Diagonal traffic:** Packet destinations are chosen uniformly at random. For every cell $c$, if $orig(c) = dest(c)$, then the packet size is 100; otherwise, the packet size is 1. In this traffic pattern, the flows on the diagonal of the switching matrix consist only of long packets, while the flows that are not on the diagonal of the switching matrix consist only of short packets. Like the spotted traffic setting, this traffic pattern stresses the effects of contiguously delivering packets of variable sizes [12].

These stochastic traffic patterns are modeled as ON-OFF processes: The ON period length is chosen according to a specific packet size distribution (that is, each ON period models an arrival of a single packet), while the OFF period is distributed geometrically with some probability $p$; the parameter $p$ is chosen so that a certain load is achieved.

All simulations were run for 100,000 time slots, and they were performed on a $16 \times 16$ switch (except the *spotted traffic* simulations that were performed on an $8 \times 8$ switch, to compare with [12]).

Figs. 3, 4, and 5 present the minimal speedup in which no packet drop was observed. As expected, the results demonstrate that Algorithm 1 needs smaller speedup to achieve smaller relative queuing delay. Moreover, the results show that as the load of the traffic increases, the speedup required by Algorithm 1 also increases.

These results show that, even in extreme situations, a speedup of less than two suffices to achieve ideal switch mimicking with frame size of only $8L_{\max}$ time slots. This can be explained by carefully investigating the reasons behind the upper bound of Theorem 8: A speedup $S \geq 4$ is required due to frames at which the underlying CCF algorithm forwards $2T$ cells from the same input port or to the same output port; moreover, the additional factor of two is caused by a poor selection of maximal matchings resulting in an inefficient contiguous decomposition as captured by Theorem 7. Under nonadversarial traffic, these two situations rarely occur in practice, certainly not simultaneously. A relative queuing delay of $\frac{2N(L_{\max}-1)-1}{S-4} + L_{\max} - 2$ time slots occurs in even more extreme situation when there is a frame $k$ and an input port $i$ (or an output port $j$) such that from any flow traversing this input port

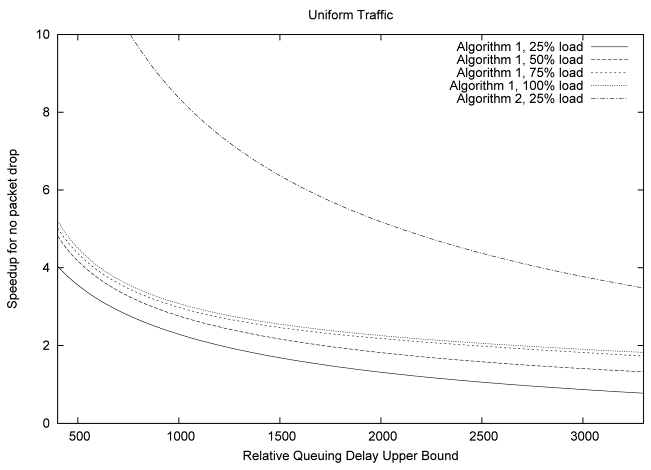

Fig. 3. Simulation results for a $16 \times 16$ switch under uniform traffic pattern and different input loads.

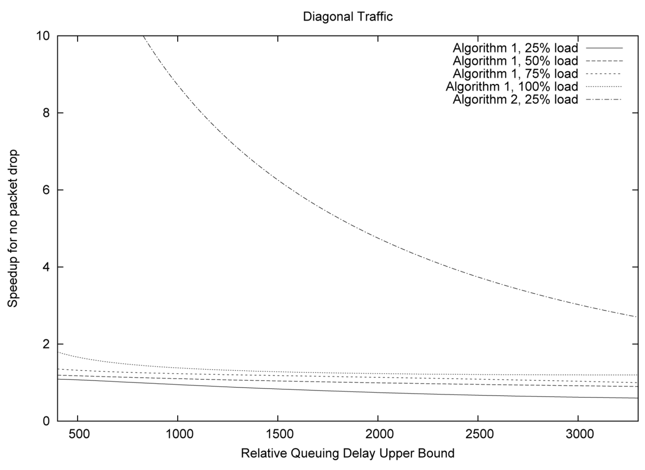

Fig. 5. Simulation results for a $16 \times 16$ switch under diagonal traffic pattern and different input loads.

(output port) there is a packet $p$ whose first cell is sent by the underlying CCF algorithm before time slot $(k-1)T$ and its last cell is sent by the CCF between time slots $(k-1)T$ and $k \cdot T$. Clearly, this situation hardly ever happens.

Finally, our simulations show that the speedup required by Algorithm 2 significantly exceeds the speedup required by Algorithm 1. The prime reason is that the frame sizes used are too small for Algorithm 2 to perform well: Recall that Algorithm 2 theoretically outperforms Algorithm 1 only when the frame size is larger than $6.85NL_{\max}$, which is 21,043 time slots for the uniform traffic setting, 5,480 time slots for the spotted traffic setting, and 10,960 time slots for the diagonal traffic setting. Yet, as evident by our simulations, frame sizes of less than 1,600 suffice for packet emulation with small speedup (using Algorithm 1). The differences between Algorithm 2 and Algorithm 1 are highlighted under the diagonal traffic for which Algorithm 2 scales according to the maximum packet size (100 cells), although packets with this size are very rare (most of the packets are of size one and the average packet size is 7.185 cells).

## 8.3 Trace-Driven Simulations

We also conducted trace-driven simulation using trace data of TCP-dominated Internet traffic over OC-48 links, taken from CAIDA [32]. We show that, also in this nonsynthetic case, Algorithm 1 performs better than its theoretical upper bounds. To the best of our knowledge, these are the first trace-driven simulations of packet-mode CIOQ switches.

Fig. 6 presents the performance of Algorithms 1 and 2 in the trace-driven experiments. We conducted these experiments in granularity of 30 bytes (that is, the cell unit size is 30 bytes) yielding a maximum packet size of 50 cells (i.e., 1,500 bytes). Furthermore, we compressed the traffic, so each input port is fully utilized (that is, 100 percent load). Compressing the traces to 100 percent load intuitively represents the worst-case traffic that should be handled by the switch; this intuition is further confirmed by our previous experiments, which show that as the traffic load increases, the required speedup also increases. As in the previous synthetic traffic patterns, these trace-driven simulations also show that Algorithm 1 performs better than its theoretical bounds and better than Algorithm 2 under reasonable frame sizes.

Finally, we compare the performance of Algorithm 1 to the following more practical heuristic, which we call the *store and forward greedy algorithm*: The algorithm gets a certain relative queuing delay $\mathcal{R}$ as a parameter and ensures that each packet either attains relative queuing delay less than $\mathcal{R}$ or is dropped. The algorithm chooses randomly a

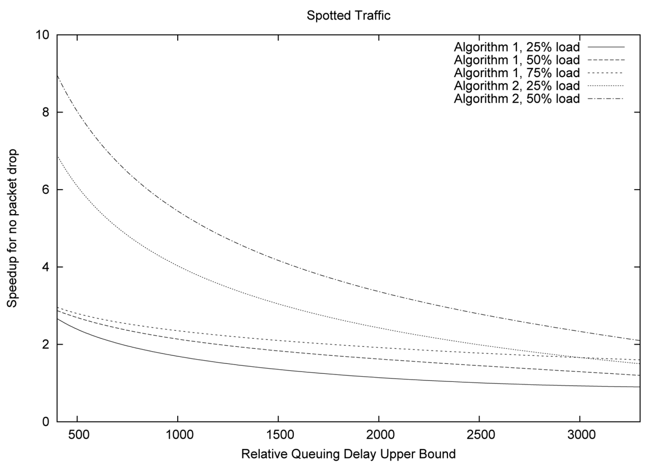

Fig. 4. Simulation results for a $8 \times 8$ switch under spotted traffic pattern and different input loads.

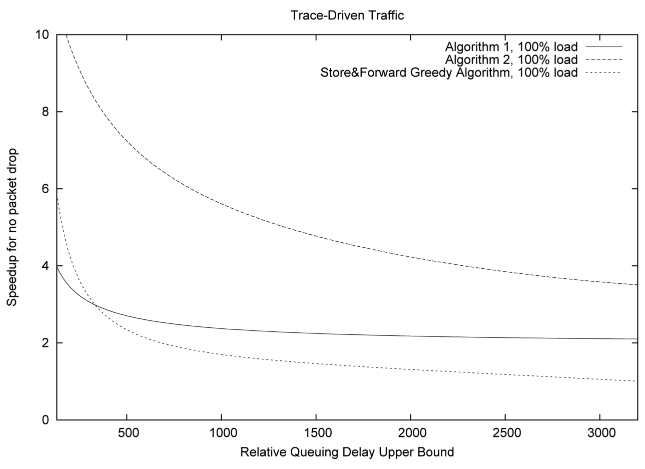

Fig. 6. Trace-driven simulation results for a $16 \times 16$ switch under 100 percent input loads.

TABLE 1
The Scheduling Complexity and Computation Running Time of Our Algorithms, Normalized by the Computation Running Time of the Underlying CCF Algorithm

| Algorithm | Time Complexity | Normalized Computation Running Time |
|---|---|---|
| Underlying CCF | $O\left(N^2\right)$ | 1.000 |
| Algorithm 1 | $O\left(N^2\right)$ | 1.274 - 1.350 |
| Algorithm 2 | $O\left(N^2+\frac{N^{2.5}}{\alpha L_{\max}}\right)$ | 1.018 - 1.027 |
| Store&Forward Greedy | $O\left(N^2\right)$ | 0.108 - 0.129 |

maximal matching over all fully arrived packets at the input side of the switch, and like Algorithm 1, keeps an input-output pair matched until the corresponding packet is fully transmitted. Before a packet is selected for transmission, a packet is dropped if its relative queuing delay is above the threshold $\mathcal{R}$.

Our trace-driven simulations indicate that this heuristic converges quickly until no packet drop is observed, and in fact, it outperforms Algorithm 1. It is important to note that the store and forward greedy algorithm and Algorithm 1 behave in a very similar manner, except that the store and forward greedy algorithm does not run an underlying CCF algorithm and does not operate in a frame-based manner. However, there are no worst-case upper bounds for the store and forward greedy algorithm.

### 8.4 Scheduling Complexity and Running Times

In this section, we compare the complexity of our algorithms and their relative running time.

First, it is important to note that both Algorithm 1 and Algorithm 2 run an underlying CCF algorithm on a shadow cell-mode CIOQ switch: The CCF algorithm is based on solving a stable marriage in bipartite graph and its complexity is $O(N^2)$ per scheduling decision [5].

Algorithm 1 additionally computes (partial) maximal matchings cell by cell, where obtaining each such matching has complexity of $O(N^2)$ as well, thus, its total complexity is $O(N^2)$. On the other hand, Algorithm 2 performs Birkhoff von-Neuman decomposition on matrices of packets with size $(\alpha+1)L_{\max}$, which is equivalent to computing a maximum size matching every $(\alpha+1)L_{\max}$ scheduling decision. Hence, its total complexity is $O(N^2+\frac{N^{2.5}}{\alpha L_{\max}})$.

Finally, although the store and forward greedy algorithm does not run an underlying CCF algorithm, it has also a complexity of $O(N^2)$, required to compute a maximal matching in each scheduling decision.

However, in practice, the most time-consuming task of our algorithms is to run the underlying CCF algorithm. Table 1 shows representative computation running times of the algorithms in our nonoptimized implementation, normalized by the computation running time of CCF. Since CCF is a common component in both Algorithms 1 and 2, this normalization masks external effects on the running time of specific executions, and therefore, allows the comparison between the different algorithms. To achieve the same comparison, we also ran and timed the same underlying CCF simultaneously with the store and forward algorithm.

The running time was measured under different traffic patterns with various frame sizes. Moreover, only computation tasks were accounted for, and not data structure operations, traffic generation, and I/O operations. For example, the matrix decompositions of Algorithm 2 require only between two and three percent of the time it takes to run CCF, while the matrix decompositions of Algorithm 1 require about 30 percent of the CCF total running time. The difference between these algorithms is explained by the granularity of their matrix decompositions and the fact that in the considered traffic patterns $\alpha L_{\max} > N$.

Finally, it is important to note that when implementing scheduling algorithm in real-life switches, their *hardware* implementation should be considered. For such implementations, two main issues should further be addressed: whether parallelism significantly reduces the complexity of the algorithm and what is the information complexity of the algorithm. As CCF is known to be impractical [5], [10], implementing either Algorithm 1 or Algorithm 2 is difficult in practice.

## 9 Discussion

This paper shows that strong performance guarantees can be provided in packet-mode CIOQ switches, regardless of the incoming traffic. These guarantees are provided by mimicking, with bounded relative queuing delay, an ideal OQ switch.

Packet-mode scheduling is an alternative to the traditional cell-mode scheduling, which eliminates the need for the computationally expensive segmentation and reassembly modules. It is expected to become even more useful as the use of optics becomes widespread, since it is prohibitively expensive to fragment packets in the optical domain.

Packet-mode scheduling imposes very confining restrictions on scheduling algorithms. While cell-based CIOQ switches with speedup $N$ are in fact output-queued switches, packet-mode CIOQ switches cannot exactly emulate output-queued switches, regardless of their speedup. Nevertheless, if relative queuing delay can be tolerated, a speedup arbitrarily close to two suffices for such mimicking. This matches the same result regarding cell-based scheduling, implying that no additional speedup is required in order to keep packets contiguous over the switch fabric.

These packet-mode schedulers induce high relative queuing delay, which can be prohibitive for real-life switches. We therefore study the trade-off between the relative queuing delay and the speedup of the switch, and prove that a reasonable relative queuing delay can be achieved by a CIOQ switch with speedup close to 4. Using simulations, we show that our algorithms incur even smaller relative queuing delay with speedup smaller than 2, under real Internet traffic traces and under synthetic stochastic traffics.

Scheduling algorithm complexity is often seen as the main performance limitation of CIOQ switches [33]: Scheduling decisions are typically done every time slot, requiring the scheduling algorithm to be as fast as the external line rate. *Frame-based* algorithms [33], [34], like those presented here, overcome this obstacle because scheduling decisions are done only at frame boundaries.

This paper presents upper bounds on the speedup required to achieve a given relative queuing delay, leaving the question of their optimality for future research. Note that by Theorem 1, $L_{\max}/2-3$ is a lower bound on the relative queuing delay, regardless of the switch speedup.

## ACKNOWLEDGMENTS

The authors would like to thank Keren Censor for helpful comments on a previous version of this paper. They would also like to thank Yishai Mansour and Danny Raz for useful suggestions and insights on Algorithm 2. The third author acknowledges the support of the ATS-WD Career Development Chair, the Alon fellowship, and the ERC Starting Grant No. 210389. A previous version of this paper appeared in the *Proceedings of the 18th ACM Symposium on Parallelism in Algorithms and Architectures* (SPAA '06). This work was done while David Hay was with the Department of Computer Science, Technion, Israel.

## REFERENCES

[1] *WAN Packet Size Distribution,* the Nat'l Laboratory for Applied Network Research.

[2] *Cisco 12000 Series Gigabit Switch Routers,* Cisco Systems, Inc., 2009.

[3] P. Giaccone, E. Leonardi, B. Prabhakar, and D. Shah, "Delay Bounds for Combined Input-Output Switches with Low Speedup," *Performance Evaluation,* vol. 55, nos. 1/2, pp. 113-128, Jan. 2004.

[4] A. Charny, P. Krishna, N. Patel, and R. Simcoe, "Algorithms for Providing Bandwidth and Delay Guarantees in Input-Buffered Crossbars with Speedup," *Proc. IEEE/IFIP Int'l Workshop Quality of Service (IWQoS),* pp. 235-244, 1998.

[5] S.T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching Output Queueing with a Combined Input Output Queued Switch," *Proc. IEEE INFOCOM,* pp. 1169-1178, 1999.

[6] P. Krishna, N. Patel, A. Charny, and R. Simcoe, "On the Speedup Required for Work-Conserving Crossbar Switches," *IEEE J. Selected Areas Comm.,* vol. 17, no. 6, pp. 1057-1066, June 1999.

[7] B. Prabhakar and N. McKeown, "On the Speedup Required for Combined Input and Output Queued Switching," *Automatica,* vol. 35, no. 12, pp. 1909-1920, Dec. 1999.

[8] D. Stephens and H. Zhang, "Implementing Distributed Packet Fair Queueing in a Scalable Architecture," *Proc. IEEE INFOCOM,* pp. 282-290, 1998.

[9] I. Stoica and H. Zhang, "Exact Emulation of an Output Queueing Switch by a Combined Input Output Queueing Switch," *Proc. IEEE/IFIP Int'l Workshop Quality of Service (IWQoS),* pp. 218-224, 1998.

[10] K. Kar, T.V. Lakshman, D. Stiliadis, and L. Tassiulas, "Reduced Complexity Input Buffered Switches," *Proc. Conf. HOT Interconnects,* pp. 13-20, 2000.

[11] *ERX-700/1400, Edge Routing Switch,* Unisphere Solutions, Inc., 2000.

[12] M. Ajmone Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Packet-Mode Scheduling in Input-Queued Cell-Based Switches," *IEEE/ACM Trans. Networking,* vol. 10, no. 5, pp. 666-678, Oct. 2002.

[13] Y. Ganjali, A. Keshavarzian, and D. Shah, "Input Queued Switches: Cell Switching vs. Packet Switching," *Proc. IEEE INFOCOM,* vol. 3, pp. 1651-1658, Mar. 2003.

[14] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100 % Throughput in an Input-Queued Switch," *IEEE Trans. Comm.,* vol. 47, no. 8, pp. 1260-1267, Aug. 1999.

[15] D. Guez, A. Kesselman, and A. Rosén, "Packet-Mode Policies for Input-Queued Switches," *Proc. ACM Symp. Parallelism in Algorithms and Architectures (SPAA),* pp. 93-102, 2004.

[16] J.S. Turner, "Strong Performance Guarantees for Asynchronous Buffered Crossbar Schedulers," *IEEE/ACM Trans. Networking,* vol. 17, no. 4, pp. 1017-1028, Aug. 2009.

[17] S.T. Chuang, S. Iyer, and N. McKeown, "Practical Algorithms for Performance Guarantees in Buffered Crossbars," *Proc. IEEE INFOCOM,* 2005.

[18] E. Altman, Z. Liu, and R. Righter, "Scheduling of an Input-Queued Switch to Achieve Maximal Throughput," *Probability in the Eng. and Informational Sciences,* vol. 14, pp. 327-334, 2000.

[19] C.-S. Chang, D. Lee, and Y. Jou, "Load Balanced Birkhoff-Von Neumann Switches, Part I: One-Stage Buffering," *Computer Comm.,* vol. 25, pp. 611-622, 2002.

[20] A. Hung, G. Kesidis, and N. McKeown, "ATM Input-Buffered Switches with Guaranteed-Rate Property," *Proc. IEEE Symp. Computers and Comm. (ISCC),* pp. 331-335, 1998.

[21] I. Keslassy, M. Kodialam, T.V. Lakshman, and D. Stiliadis, "On Guaranteed Smooth Scheduling for Input-Queued Switches," *IEEE/ACM Trans. Networking,* vol. 13, no. 6, pp. 1364-1375, Dec. 2005.

[22] X. Li and M. Hamdi, "On Scheduling Optical Packet Switches with Reconfiguration Delay," *IEEE J. Selected Areas Comm.,* vol. 21, no. 7, pp. 1156-1164, Sept. 2003.

[23] B. Towles and W. Dally, "Guaranteed Scheduling for Switches with Configuration Overhead," *IEEE/ACM Trans. Networking,* vol. 11, no. 5, pp. 835-847, Oct. 2003.

[24] T. Weller and B. Hajek, "Scheduling Nonuniform Traffic in a Packet-Switching System with Small Propagation Delay," *IEEE/ACM Trans. Networking,* vol. 5, no. 6, pp. 813-823, Dec. 1997.

[25] N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Networking,* vol. 7, no. 2, pp. 188-201, Apr. 1999.

[26] S. Iyer and N. McKeown, "Making Parallel Packet Switches Practical," *Proc. IEEE INFOCOM,* pp. 1680-1687, 2001.

[27] L. Kleinrock, *Queuing Systems,* vol. II. John Wiley & Sons, 1975.

[28] G. Birkhoff, "Tres Observaciones Sobre El Algebra Lineal," *Univ. Nac. Tucuman Rev. Ser. A,* vol. 5, pp. 147-151, 1946.

[29] J. von Neumann, "A Certain Zero-Sum Two-Person Game Equivalent to the Optimal Assignment Problem," *Contributions to the Theory of Games,* vol. 2, pp. 5-12, 1953.

[30] L. Dulmage and I. Halperin, "On a Theorem of Frobenius-Konig and J. von Neumann's Game of Hide and Seek," *Trans. Royal Soc. Canada III,* vol. 49, pp. 23-29, 1955.

[31] Y. Li, S. Panwar, and H. Chao, "Exhaustive Service Matching Algorithms for Input Queued Switches," *Proc. IEEE Workshop High Performance Switching Routing,* pp. 253-258, 2004.

[32] Cooperative Assoc. for Internet Data Analysis (CAIDA), http://www.caida.org/, 2009.

[33] A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "A Framework for Differential Frame-Based Matching Algorithms in Input-Queued Switches," *Proc. IEEE INFOCOM,* 2004.

[34] P. Pappu, J. Turner, and K. Wong, "Work-Conserving Distributed Schedulers for Terabit Routers," *Proc. ACM SIGCOMM,* pp. 257-268, 2004.

**Hagit Attiya** received the BSc degree in mathematics and computer science, and the MSc and PhD degrees in computer science from the Hebrew University of Jerusalem in 1981, 1983, and 1987, respectively. She is currently a professor in the Department of Computer Science at the Technion, Israel Institute of Technology. Her research interests include distributed and parallel computation. She is a fellow of the ACM.

**David Hay** (M'09) received the BA (summa cum laude) and PhD degrees in computer science from the Technion—Israel Institute of Technology in 2001 and 2007, respectively. He is currently a postdoctoral research scientist in the Department of Electrical Engineering, Columbia University, New York. His research interests include algorithmic aspects of high-performance switches and routers, in particular, packet classification, QoS provisioning, and competitive analysis. He is a member of the IEEE.

**Isaac Keslassy** (M'02) received the MS and PhD degrees in electrical engineering from Stanford University, California, in 2000 and 2004, respectively. He is currently a faculty member in the Electrical Engineering Department at the Technion, Haifa, Israel. His recent research interests include the design and analysis of high-performance routers and on-chip networks. He is the recipient of the ERC Starting Grant, the Yigal Alon fellowship, and the ATS-WD Career Development Chair. He is a member of the IEEE.