

Packet-Mode Emulation of Output-Queued Switches

(Extended Abstract)

Hagit Attiya[†] David Hay^{†*} Isaac Keslassy^{**}

[†]Department of Computer Science, Technion, {hagit,hdavid}@cs.technion.ac.il

^{**}Department of Electrical Engineering, Technion, isaac@ee.technion.ac.il

March 7, 2006

Abstract

Most common network protocols (e.g., the Internet Protocol) work with 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 studies *packet-mode scheduling* for the *combined input output queued (CIOQ)* switch architecture and investigates its cost.

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 are 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 2 suffices to mimic an ideal output-queued switch. This implies that a packet-mode scheduler does not require a fundamentally 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.

Finally, we show that a speedup arbitrarily close to 1 suffices to mimic an output-queued switch with a bounded buffer size.

Submitted to the regular track.

*Contact Author, hdavid@cs.technion.ac.il, Tel: +972-4-8293942, Fax: +972-4-8293900.

1 Introduction

In many network protocols, from very large Wide Area Networks (WANs) to small Networks on Chips (NoCs), traffic is comprised of variable size *packets*. A prime example is provided by IP datagrams whose sizes typically vary from 40 to 1500 bytes [22]. Real-life switches, however, operate with fixed-size *cells*, which are easier to buffer and schedule synchronously in an electronic domain.

Transmitting packets over cell-based switches requires the use of packet segmentation and reassembly modules, resulting in a significant computation and communication overhead. *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 spectrum and might be unable to afford their segmentation and reassembly. Cell-based schedulers, unaware of packet boundaries, may cause performance degradation; indeed, packet-aware switches typically have better drop-rate, since they may reduce the number of retransmissions by ensuring that only complete packets are sent over the switch fabric (cf. [29, Page 44]).

Packet mode schedulers [10, 20] bridge this gap 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.

It is imperative to explore whether such packet mode schedulers can provide similar performance guarantees as cell-based schedulers. We address this question by focusing on *Combined Input-Output Queued* (CIOQ) switches and investigating whether a packet-mode CIOQ switch can mimic an ideal Output-Queued (OQ) switch with bounded *relative queuing delay*.

CIOQ switches with crossbar fabric (Figure 1) are widely used today as the core of contemporary switches [8, 11], since they operate at the external line rate and can scale with an increasing number of ports. Another reason for their popularity is the fact that *cell-based CIOQ* switches with small speedup¹ $S = 2$ emulate an OQ switch [5, 6, 17, 24–26], and therefore, provide perfect performance guarantees.

Output-queued 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 should hold for arbitrary traffic patterns, including adversarial traffic. Therefore, this performance measure is entirely deterministic and is not burdened by probabilistic assumptions on the incoming traffic that can be unsubstantiated. In addition, the evaluation is worst-case in nature, implying that QoS demands are *guaranteed*, unlike probabilistic or empirical evaluation based on simulations.

Our results: This paper presents packet-mode schedulers for CIOQ switches that mimic an OQ switch. Since such mimicking requires CIOQ switches with a certain speedup, we further investigate the trade-off between the speedup of the switch and its relative queuing delay.

We devise pipelined frame-based schedulers, in which scheduling decisions are done at the frame boundaries. Our schedulers and their analysis rely on matrix decomposition techniques. 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. The major challenge in these decompositions is ensuring contiguous packet delivery while decomposing the demand matrix to as few as possible permutations.

Specifically, we show (Theorem 4.6) that a *packet-mode scheduler* for CIOQ switch does not require a fundamentally higher speedup than a cell-based CIOQ switch, if high relative queuing delay can be tolerated. 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 $\mathcal{R} = O(N \cdot \text{LCM}(L_{\max}))$ time-slots; here, L_{\max} is the maximum packet size,

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

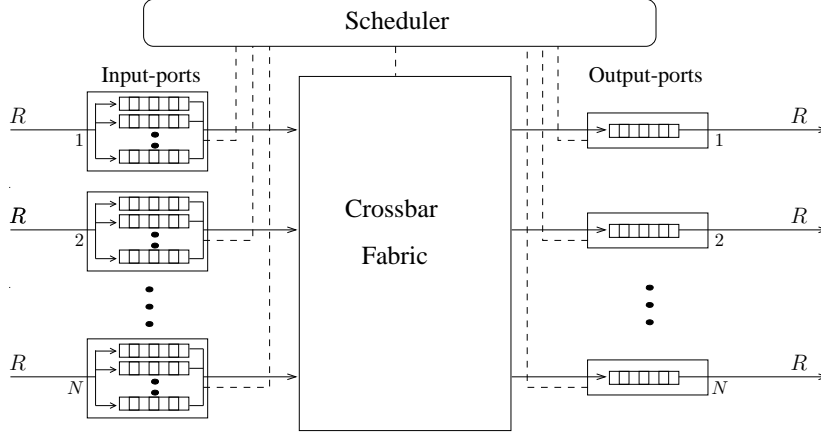


Figure 1: Combined Input-Output Queued Switch with Virtual Output-Queuing. The switch fabric operate at rate $S \cdot R$, where S is the switch's speedup.

$\text{LCM}(L_{\max})$ is the least common multiple of $1, \dots, L_{\max}$, and N is the number of input (output) ports. This result also holds in the common case where only few packet sizes are legal, and the resulting relative queuing delay is $O(N \cdot \text{LCM}(\mathcal{L}))$ time-slots, where \mathcal{L} is the restricted set of legal packet sizes.

The relative queuing delay can be significantly reduced with just a doubling of the speedup. We show (Theorem 4.5) that a speedup of $4 + O(\frac{1}{\mathcal{R}})$ suffices to ensure that a packet-mode CIOQ switch mimics an OQ switch with a more reasonable relative queuing delay of $\mathcal{R} = O(NL_{\max})$ time-slots.

The relative queuing delay can be further reduced to be only $L_{\max} - 1$ time-slots, if the speedup is increased to $2L_{\max}$ (Theorem 3.2). In addition, we show (Theorem 3.1) that it is impossible to achieve relative queuing delay of less than $L_{\max}/2 - 3$, regardless of the speedup used.

Finally, we consider mimicking an output-queued switch with a bounded buffer size B at each output-port. Extending the matrix decomposition techniques, we show (Corollary 5.3) that with a smaller speedup of $1 + O(\frac{1}{\mathcal{R}})$ and relative queuing delay $\mathcal{R} = O(B + N \cdot \text{LCM}(L_{\max}))$, a packet-mode CIOQ mimics OQ switch with buffer size B .

Figure 2 summarizes our results and demonstrates the trade-off between the speedup required for OQ switch mimicking and the resulting relative queuing delay.

Related work: Previous work [10, 20] considers packet-mode scheduling in an *input-queued* (IQ) switch with crossbar fabric speedup $S = 1$ 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 [21]. Marsan et al. [20] 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 [12].

The ability of a *cell-based* CIOQ switch to emulate an output-queued switch has been extensively investigated (e.g., [5, 6, 17, 24–26]). For example, Chuang et al. [6] introduce the *critical cells first* (CCF) algorithm, which allows a CIOQ switch with speedup 2 to emulate (exactly) an output-queued switch. In addition, they show that a CIOQ switch needs speedup of at least $2 - \frac{1}{N}$ in order to emulate an output-queued switch. To the best of our knowledge, the ability of packet-mode CIOQ switches to emulate OQ switches

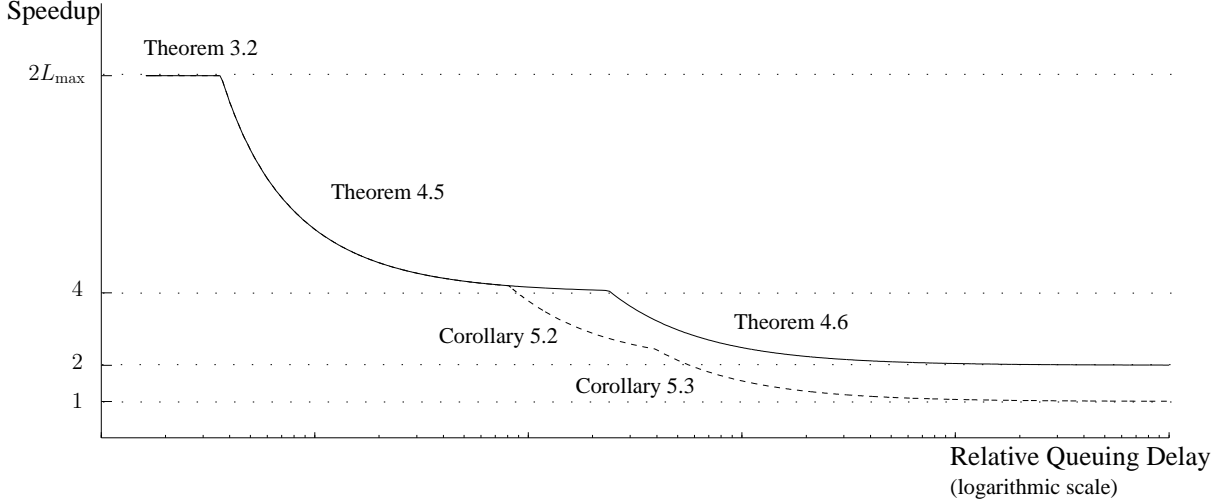


Figure 2: Summary of our results. The solid line represents emulation of OQ switch with unbounded buffer size, while the dashed line represents emulation of OQ switch with buffer size B . Relative queuing delay scale is logarithmic.

has not been investigated before.

Recently, Turner [28] investigated 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 [7] and packet-mode schedulers [28]. The algorithms proposed in [28] do not apply to the unbuffered switch fabric we study.

Birkhoff von-Neumann matrix decomposition is used for scheduling when the arrival rate can be estimated [1, 4, 13]. Other matrix decomposition heuristics are employed in frame-based schedulers for optical switching and in satellite-switched time-division multiple access (SS/TDMA) schedulers [15, 18, 27, 31]. These decompositions are not packet-aware and may violate the contiguous delivery of cells corresponding to the same packet.

Weller and Hajek [31] 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 the algorithm described in Theorem 4.5.

Organization of the paper: Section 2 defines CIOQ switches and packet-mode OQ switch mimicking. Section 3 shows that CIOQ switches cannot provide precise packet-mode emulation of OQ switches; it further shows how to mimic an OQ switch with small relative queuing delay when the speedup is very large. In Section 4, we give the necessary background about matrix decomposition, and introduce our main contributions—scheduling algorithms that provide OQ switch mimicking for packet-mode CIOQ switches with small speedup. This section also describes the trade-off between the speedup of the switch and its relative queuing delay. Mimicking a packet-mode OQ switch with bounded buffer size is discussed in Section 5. We conclude in Section 6 with a discussion of our results.

2 Preliminaries

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 , as illustrated in Figure 1. Packets of variable size arrive at the input-ports and leave the output-ports in a time-slotted manner (that is, all the switch external lines are synchronized). 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 our description.

The packet size is measured by 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 in 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 and $first(p)$, $last(p)$ are the first and last cells of packet p .

In this paper, we consider a *combined input-output queued (CIOQ)* switch with speedup S . In such a switch, 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 (recall Figure 1). 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, which is called *virtual output-queue* and denoted by VOQ_{ij} . 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.² 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, \dots, c_ℓ) 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_ℓ 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 under-utilize its speedup. For example, suppose that the first cell c_1 of a packet $p = (c_1, c_2, \dots, c_\ell)$ 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).

²For non-integral 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 [11].

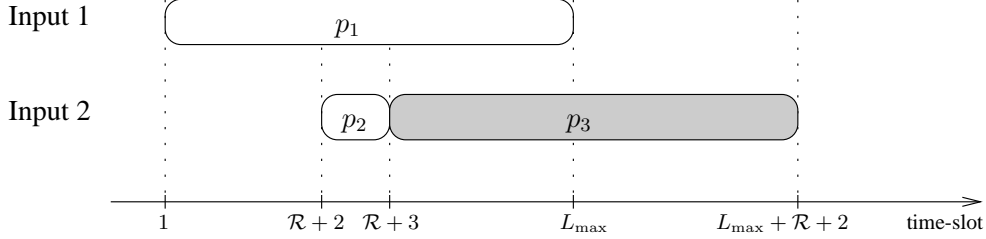


Figure 3: Illustration of the proof of Theorem 3.1; white packets are destined for output-port 1 and the gray packet is destined for output-port 2.

Packet-mode output-queued switch mimicking: 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 $p = (c_1, \dots, c_\ell)$, $t_{OQ}(c_i) = t_{OQ}(c_j) + (i - j)$ for $1 \leq j \leq i \leq \ell$.

In addition, a packet-mode OQ switch should 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 $p \prec p'$ (i.e., packet p should be handled before packet p').

We now define the stability criterion used in this paper:

Definition 1 (Output-queued switch mimicking [14]) 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-slot after it would have left the output-queued switch, where \mathcal{R} is a constant independent of the elapsed time.

An OQ switch mimicking without relative queuing delay (i.e., $\mathcal{R} = 0$) is sometimes referred to as *output-queued switch emulation* [6]. Since an output-queued switch is a work-conserving switch³, it implies that any switch that emulates an output-queued switch is also work-conserving.

3 Bounds on the Relative Queuing Delay

We show that a packet-mode CIOQ switch cannot emulate an OQ switch, regardless of its speedup, by showing that even a mimicking with a small relative queuing delay is impossible.

Figure 3 depicts the traffic used to establish this result. In case the CIOQ switch provides relative queuing delay \mathcal{R} , output port 1 must handle packet p_1 between time-slots $\mathcal{R} + 1$ and L_{\max} . Similarly, input port 2 must handle packet p_3 between time-slots $2\mathcal{R} + 3$ and $L_{\max} + \mathcal{R} + 2$. This implies that if $\mathcal{R} < L_{\max}/2 - 3$, then packet p_2 cannot be scheduled before time-slot $L_{\max} + \mathcal{R} + 2$, and therefore attains relative queuing delay higher than \mathcal{R} . (Detailed proof appears in Appendix A.1.)

Theorem 3.1 A packet-mode CIOQ switch cannot mimic an OQ switch with a relative queuing delay $\mathcal{R} < L_{\max}/2 - 3$ time-slots.

³A *work-conserving* switch guarantees that 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 . [6, 16, 17]

Note that this result holds since the CIOQ switch waits for the cells of the different packets to arrive, and therefore under the situation described in the proof of Theorem 3.1, the switch in fact degrades to work at the external line rate (i.e., with $S = 1$), as an IQ switch. The result is therefore consistent with the known result that IQ switches (at speedup 1) cannot emulate output-queued switches [6].

We now show that a CIOQ switch can mimic an OQ switch with a small relative queuing delay of $L_{\max} - 1$ time-slots provided it has a sufficiently large speedup of $2L_{\max}$. The algorithm closely follows the CCF algorithm, which emulates (precisely) a cell-based OQ switch with speedup $S = 2$ [6].

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 detailed algorithm and its proof appear in Appendix A.2.)

Theorem 3.2 *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.*

This result only demonstrates the possibility of OQ mimicking with bounded delay, since a speedup $S = 2L_{\max}$ is unreasonable in practical switches. In the rest of the paper, we show how to provide OQ mimicking with small speedup.

4 The Speedup Required for Mimicking an Output-Queued Switch

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 some *demand matrices*, and then decompose these matrices into some permutations (or sub-permutations). 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 next.

4.1 Matrix Decomposition

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

In the rest of the paper, we refer to sub-permutations as permutations.

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 4.1 (Birkhoff von-Neumann decomposition [3, 9, 30]) *If a matrix $A \in \mathbb{N}^{N \times N}$ is C -bounded, 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. For reasons that will become clear shortly, we are interested in the following class of permutations:

Definition 4 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 of 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 .

The next theorem shows that if a matrix is decomposed by any maximal matchings then the number of permutations needed is at most twice the number needed in Theorem 4.1. 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 $A(m) = 0$. We next show that 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 (proof omitted).

Theorem 4.2 ([31, Theorem 2.2]) For every C -bounded matrix $A \in \mathbb{N}^{N \times N}$, the decomposition procedure described above stops after less than $2C - 1$ iterations.

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

Our schedulers operate by constructing a demand matrix at each frame boundary, and then using its decomposition results for scheduling decisions in 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.

The demand matrix in each frame is built according to the times in which an underlying CCF algorithm forwards cells over the switch fabric. A key insight is that, using the CCF algorithm, a cell-based CIOQ switch with speedup $S = 2$ can emulate any *push-in-first-out* (PIFO) cell-based output-queued switch [6].

In turn, packet-mode OQ switches can be implemented by a PIFO cell-based OQ switch with the following buffer management policy: 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' . Since the order of cells in the buffer does not change, and cells are served from the head of the buffer, this policy indeed complies with a PIFO queuing discipline for cells.

Let $\text{CCF}(c)$ be the time-slot in which a cell c is forwarded over the switch fabric by the CCF algorithm that emulates an OQ switch with this PIFO discipline [6]. Clearly, $\text{CCF}(c) \leq t_{OQ}(c)$. We have the next lemma, whose proof appears in Appendix A.3.

Lemma 4.3 If a scheduling algorithm ALG schedules the cell $\text{last}(p)$ of every packet p by time-slot $\text{CCF}(\text{last}(p)) + \delta$ then the maximum relative queuing delay of ALG is at most $\delta + L_{\max} - 1$, where L_{\max} is the maximum packet size.

We now explore the trade-off between the speedup S in which the CIOQ switch operates and its relative queuing delay. We devise a frame-based scheduler in which packets that did not fully arrive until the frame boundary are queued in the input-side of the switch until the next frame, as captured by the next definition:

Definition 5 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 $a_{ij}(T, k)$, includes all cells $c \notin a_{ij}(T, k - 1)$ such that all cells $c' \in \text{packet}(c)$ have $\text{CCF}(c') \leq kT$. Let $a_{ij}(T, 0)$ be \emptyset .

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

The next lemma, whose proof appears in Appendix A.4, bounds the number of cells, sharing an input-port or an output-port, that should be scheduled within the same frame:

Lemma 4.4 For every input-port i , output-port j , frame size T and frame number $k > 0$,

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

Lemma 4.4 and Theorem 4.1 imply that the eligible cells of each frame can be scheduled within $2T + N(L_{\max} - 1)$ scheduling opportunities. Unfortunately, the decomposition described in Theorem 4.1 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 2. The following decomposition of A into six permutations

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

violates the packet-mode constraints: Contiguous transmission of packet $a_{1,1}$ requires that the first three permutations are 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 yields that at least one permutation must be adjacent to three permutations.

To circumvent this problem, we introduce a different decomposition algorithm, which guarantees contiguous packet delivery but requires twice as much scheduling opportunities:

At each frame boundary, the algorithm constructs a $2T + N(L_{\max} - 1)$ -bounded matrix $B(k) = [b_{ij}]$ such that $b_{ij} = |a_{ij}(T, k)|$. Then, the algorithm repeatedly builds *maximal* matchings of matrix $B(k)$ and keeps contiguous packet delivery by ensuring the following property: 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, then the algorithm keeps the matching between input-port i and output-port j for the next iteration. (This procedure is sometimes called *exhaustive service matching* [19].) By Theorem 4.2, the algorithm using this procedure mimics an output-queued switch with a speedup arbitrarily close to 4 and attains a relative queuing delay of $O(NL_{\max})$. (The detailed algorithm and its proof appear in Appendix A.5.)

Theorem 4.5 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.

Notice that if the switch speedup is $S > 4$ then the relative queuing delay induced by this algorithm is $\frac{2N(L_{\max}-1)-1}{S-4} + L_{\max} - 2$ time-slots.

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

We now show that a speedup arbitrarily close to 2 suffices for OQ mimicking. This is done in the context of the common situation where packet size are restricted to be from the set \mathcal{L} . Notice that this case generalizes the unrestricted packet size case, where $\mathcal{L} = \{1, \dots, L_{\max}\}$. Let $\text{LCM}(\mathcal{L})$ be the least common multiple of all elements in \mathcal{L} . For each frame, the scheduler described in Theorem 4.5 schedules all eligible cells with the same origin and the same destination contiguously, implying that in fact it considers them as a single packet. Using a more fine-grained scheduler and the Birkhoff von-Neumann decomposition, we show that a smaller speedup suffices, albeit with larger relative queuing delay:

Theorem 4.6 *A packet-mode CIOQ switch with speedup $S = 2 + \frac{N(L_{\max}\text{LCM}(\mathcal{L})-1)}{T}$ can mimic an OQ switch with a relative queuing delay of $2T + L_{\max} - 2$ time-slots.*

Proof: Fix a frame size T . For each possible packet size $\ell \in \mathcal{L}$, let $a_{ij}^\ell(T, k) \subseteq a_{ij}(T, k)$ be the set of eligible cells that correspond to packets of size ℓ . Let $B(\ell, k) = [b(\ell, k)_{ij}]$ be the matrix with values $b(\ell, k)_{ij} = \frac{|a_{ij}^\ell(T, k)|}{\ell}$, that is, the number of eligible *packets* of size ℓ in frame k .

For every packet size ℓ , the algorithm first tries to concatenate $\text{LCM}(\mathcal{L})/\ell$ packets one after the other in order to get one *mega-packet* of size $\text{LCM}(\mathcal{L})$, each such mega-packet consists of packets of the same size. The matrix $B(\text{LCM}(\mathcal{L}), k) = [b((\text{LCM}(\mathcal{L}), k)_{ij})]$ counts the number of such mega-packets:

$$b((\text{LCM}(\mathcal{L}), k)_{ij}) = \sum_{\ell \in \mathcal{L}} \left\lfloor \frac{\ell \cdot b(\ell, k)_{ij}}{\text{LCM}(\mathcal{L})} \right\rfloor$$

We first bound the sum of each row and each column of the matrix $B(\text{LCM}(\mathcal{L}), k)$. Consider some row i of the matrix (the proof for a column j follows analogously):

$$\begin{aligned} \sum_{j=1}^N b((\text{LCM}(\mathcal{L}), k)_{ij}) &= \sum_{j=1}^N \sum_{\ell \in \mathcal{L}} \left\lfloor \frac{\ell \cdot b(\ell, k)_{ij}}{\text{LCM}(\mathcal{L})} \right\rfloor \leq \sum_{j=1}^N \sum_{\ell \in \mathcal{L}} \frac{\ell \cdot b(\ell, k)_{ij}}{\text{LCM}(\mathcal{L})} = \frac{1}{\text{LCM}(\mathcal{L})} \sum_{j=1}^N \sum_{\ell \in \mathcal{L}} |a_{ij}^\ell(T, k)| \\ &= \frac{1}{\text{LCM}(\mathcal{L})} \sum_{j=1}^N |a_{ij}(T, k)| \leq \frac{1}{\text{LCM}(\mathcal{L})} \cdot (2T + N(L_{\max} - 1)) \quad \text{By Lemma 4.4} \end{aligned}$$

By Theorem 4.1, the matrix $B(\text{LCM}(\mathcal{L}), k)$ can be decomposed into $\frac{2T + N(L_{\max} - 1)}{\text{LCM}(\mathcal{L})}$ permutations. Denote by $\mathcal{P}(\text{LCM}(\mathcal{L}), k)$ the set of these permutations.

We now turn to deal with leftover packets. The matrices $B'(\ell, k) = [b'(\ell, k)_{ij}]$ count the number of packets that are not concatenated into mega-packets. Note that $b'(\ell, k)_{ij} \leq (\text{LCM}(\mathcal{L}) - 1)/\ell$, since it is the remainder of dividing $b(\ell, k)_{ij}$ by $\text{LCM}(\mathcal{L})/\ell$. This implies that the sum of each row and each column of matrix $B'(\ell, k)$ is bounded by $N(\text{LCM}(\mathcal{L}) - 1)/\ell$, and therefore by Theorem 4.1 it can be decomposed into $\frac{N(\text{LCM}(\mathcal{L}) - 1)}{\ell}$ permutations. Let $\mathcal{P}(\ell, k)$ be the set of the permutations used to decompose the matrix $B'(\ell, k)$.

After obtaining these sets of permutations, the algorithm forwards contiguously all the mega-packets by holding each permutation $P \in \mathcal{P}(\text{LCM}(\mathcal{L}), k)$ for $\text{LCM}(\mathcal{L})$ consecutive iterations. Then for every $\ell \in \mathcal{L}$, the algorithm holds each permutation $P \in \mathcal{P}(\ell, k)$ for ℓ consecutive iterations. Clearly, all the cells of a specific packet are forwarded contiguously, and the algorithm satisfies the packet-mode scheduling constraints.

The number of iterations needed for the algorithm to complete is bounded by:

$$\text{LCM}(\mathcal{L}) \frac{2T + N(L_{\max} - 1)}{\text{LCM}(\mathcal{L})} + \sum_{\ell \in \mathcal{L}} \ell \frac{N(\text{LCM}(\mathcal{L}) - 1)}{\ell} \leq 2T + N(L_{\max} - 1) + NL_{\max}(\text{LCM}(\mathcal{L}) - 1)$$

Thus, with a speedup of $2 + \frac{N(L_{\max} \text{LCM}(\mathcal{L}) - 1)}{T}$ the algorithm schedules all cells correspond to frame k within the next frame. This implies that for each packet p the maximum relative queuing delay of cell $\text{last}(p)$ is less than two frame sizes, namely at most $2T - 1$ time-slots. Hence, Lemma 4.3 implies that the maximum relative queuing delay is at most $2T + L_{\max} - 2$. ■

Notice that if the switch speedup is $S > 2$ then the relative queuing delay induced by this algorithm is $\frac{N(L_{\max} \text{LCM}(\mathcal{L}) - 1)}{S - 2} + L_{\max} - 2$ time-slots.

5 Mimicking an OQ Switch with Bounded Buffer Size

In many practical applications, CIOQ switches are required to emulate OQ switches with *bounded* buffer size. We show that smaller speedup suffices for mimicking an OQ switch with output buffer B . 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 5 and Lemma 4.4:

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 $a_{ij}(T, k)$, is the set of cells c that are delivered successfully by the output-queued switch, $c \notin a_{ij}(T, k - 1)$, and all cells $c' \in \text{packet}(c)$ arrive at the switch before time-slot kT . Let $a_{ij}(T, 0)$ be \emptyset .

As in Definition 5, all the cells of each packet p are in the same set of eligible cells. The next lemma bounds the size of these sets. (Proof appears in Appendix A.6.)

Lemma 5.1 For every input-port i , output-port j , frame size T and frame number $k > 0$,

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

In order to mimic an output-queued switch, the CIOQ switch drops all cells that are dropped by the OQ switch. By employing Lemma 5.1 in the proofs of Theorems 4.5 and 4.6, we get the following results:

Corollary 5.2 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 5.3 A packet-mode CIOQ switch with speedup $S = 1 + \frac{B + N(L_{\max} \text{LCM}(\mathcal{L}) - 1)}{T}$ can mimic an OQ switch with buffer size B with a relative queuing delay of $2T + L_{\max} - 2$ time-slots.

6 Discussion

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

Packet-mode scheduling imposes very confining restrictions on scheduling algorithms. Unlike cell-based CIOQ switches with speedup N that are in fact output-queued switches, packet-mode CIOQ switches cannot exactly emulate output-queued switches, regardless of their speedup. On the other hand, if relative queuing delay can be tolerated, a speedup arbitrarily close to 2 suffices for such mimicking. This result matches the same result regarding cell-based scheduling, implying that, somewhat surprisingly, 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 switch's speedup, and prove that a reasonable relative queuing delay can be achieved by CIOQ switch with speedup close to 4. Based on preliminary simulations, we expect this algorithm to incur even smaller relative queuing delay with speedup smaller than 2, under real Internet traffic traces. Note also that for fixed-size packets (i.e., $L_{\max} = 1$) our schedulers are identical to the CCF algorithm.

Scheduling algorithm complexity is often seen as the main performance limitation of CIOQ switches [2], since scheduling decisions are typically done every time-slot, requiring the scheduling algorithm to be as fast as the external line rate. *Frame-based* algorithms [2, 23], 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 3.1, $L_{\max}/2 - 3$ is a lower bound on the relative queuing delay, regardless of the switch speedup.

Acknowledgments: We would like to thank Keren Censor for helpful comments.

References

- [1] E. Altman, Z. Liu, and R. Righter. Scheduling of an input-queued switch to achieve maximal throughput. *Probability in the Engineering and Informational Sciences*, 14:327–334, 2000.
- [2] A. Bianco, P. Giaccone, E. Leonardi, and F. Neri. A framework for differential frame-based matching algorithms in input-queued switches. In *IEEE INFOCOM*, 2004.
- [3] G. Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucuman Rev. Ser. A*, 5:147–151, 1946.
- [4] C.S. Chang, D.S. Lee, and Y.S. Jou. Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering. *Computer Communications*, 25:611–622, 2002.
- [5] A. Charny, P. Krishna, N. S. Patel, and R.J. Simcoe. Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup. In *Sixth IEEE/IFIP International Workshop on Quality of Service*, pages 235–244, 1998.
- [6] S. Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching output queueing with a combined input output queued switch. In *IEEE INFOCOM*, pages 1169–1178, 1999.
- [7] S. Chuang, S. Iyer, and N. McKeown. Practical algorithms for performance guarantees in buffered crossbars. In *IEEE INFOCOM*, 2005.
- [8] Cisco Systems, Inc. *Cisco 12000 Series Gigabit Switch Routers*. Available online at: <http://www.cisco.com/warp/public/cc/pd/rt/12000/prodlit/gsr.ov.pdf>.
- [9] L. Dulmage and I. Halperin. On a theorem of Frobenius-Konig and J. von Neumann's game of hide and seek. *Trans. Roy. Soc. Canada III*, 49:23–29, 1955.
- [10] Y. Ganjali, A. Keshavarzian, and D. Shah. Input queued switches: cell switching vs. packet switching. In *INFOCOM 2003*, volume 3, pages 1651–1658, March 2003.

- [11] P. Giaccone, E. Leonardi, B. Prabhakar, and D. Shah. Delay bounds for combined input-output switches with low speedup. *Performance Evaluation*, 55(1-2):113–128, 2004.
- [12] D. Guez, A. Kesselman, and A. Rosén. Packet-mode policies for input-queued switches. In *the 16th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 93–102, New York, NY, USA, 2004. ACM Press.
- [13] A. Hung, G. Kesidis, and N. McKeown. ATM input-buffered switches with guaranteed-rate property. In *IEEE ISCC*, pages 331–335, 1998.
- [14] S. Iyer and N. McKeown. Making parallel packet switches practical. In *IEEE INFOCOM*, pages 1680–1687, 2001.
- [15] I. Keslassy, M. Kodialam, T.V. Lakshman, and D. Stiliadis. On guaranteed smooth scheduling for input-queued switches. In *IEEE INFOCOM*, 2003.
- [16] L. Kleinrock. *Queueing Systems, Volume II*. John Wiley&Sons, 1975.
- [17] P. Krishna, N. S. Patel, A. Charny, and R.J. Simcoe. On the speedup required for work-conserving crossbar switches. *IEEE Journal on Selected Areas in Communications*, 17(6):1057–1066, June 1999.
- [18] X. Li and M. Hamdi. On scheduling optical packet switches with reconfiguration delay. *IEEE Journal on Selected Areas in Communications*, 21(7):1156–1164, 2003.
- [19] Y. Li, S. Panwar, and H.J. Chao. Exhaustive service matching algorithms for input queued switches. In *IEEE Workshop on High Performance Switching and Routing*, pages 253–258, 2004.
- [20] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri. Packet-mode scheduling in input-queued cell-based switches. *IEEE/ACM Transactions on Networking*, 10(5):666–678, October 2002.
- [21] N. McKeown, A. Mekkitikul, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. *IEEE Transactions on Communications*, 47(8):1260–1267, August 1999.
- [22] The National Laboratory for Applied Network Research. *WAN packet size distribution*. Available online at: <http://www.nlanr.net/NA/Learn/packetsizes.html>.
- [23] P. Pappu, J. Turner, and K. Wong. Work-conserving distributed schedulers for terabit routers. In *ACM SIGCOMM*, pages 257–268. ACM Press, 2004.
- [24] B. Prabhakar and N. McKowen. On the speedup required for combined input and output queued switching. *Automatica*, 35(12):1909–1920, December 1999.
- [25] D.C. Stephens and H. Zhang. Implementing distributed packet fair queueing in a scalable architecture. In *IEEE INFOCOM*, pages 282–290, 1998.
- [26] I. Stoica and H. Zhang. Exact emulation of an output queueing switch by a combined input output queueing switch. In *Sixth IEEE/IFIP International Workshop on Quality of Service*, pages 218–224, 1998.
- [27] B. Towles and W.J. Dally. Guaranteed scheduling for switches with configuration overhead. *IEEE/ACM Transactions on Networking*, 11(5):835–847, 2003.
- [28] J. Turner. Strong performance guarantees for asynchronous crossbar schedulers. In *IEEE INFOCOM*, 2006. To appear.
- [29] Unisphere Solutions, Inc. *ERX-700/1400, Edge Routing Switch*. Available online at: <http://www.juniper.net/techpubs/software/erx/erx130/product.overview.pdf>.
- [30] J. von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games*, 2:5–12, 1953.
- [31] T. Weller and B. Hajek. Scheduling nonuniform traffic in a packet-switching system with small propagation delay. *IEEE/ACM Transactions on Networking*, 5(6):813–823, 1997.

A Omitted Proofs

A.1 Proof of Theorem 3.1

Assume towards a contradiction that the CIOQ switch mimics an OQ switch with relative queuing delay of $\mathcal{R} < L_{\max}/2 - 3$, and consider the following traffic, comprising of only three packets (see Figure 3): At time-slot 1 a packet p_1 of size L_{\max} arrives at input-port 1, destined for output-port 1. At time-slot $\mathcal{R} + 2$, another packet, denoted p_2 , of size 1 arrives at input-port 2, destined for output-port 1. At time-slot $\mathcal{R} + 3$, a 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 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} , then output-port 1 is busy handling p_1 at least until time-slot L_{\max} .

Using the same arguments, the first cell of packet p_3 must be delivered to output-port 2 at time-slot $2\mathcal{R} + 3$, and input-port 2 is busy handling p_3 at least until time-slot $L_{\max} + \mathcal{R} + 2$.

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

A.2 Proof of Theorem 3.2

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 comprising only of cells in $\mathcal{T}(t)$ that are the first cells of their corresponding packets. Denote by $\text{CCF}'(c)$ 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}' .

We propose a *packet-mode CCF algorithm* (PM-CCF) that 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 $t_{PM-CCF} = \text{CCF}'(\text{first}(p)) + L_{\max} - 1$.

Note that since the cell-based CCF works with speedup $S = 2$, for each time-slot t there are at most two cells which 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 $\text{CCF}'(\text{first}(p)) + L_{\max} - 1$.

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 $t_{OQ}(c) \geq t_{OQ}(\text{first}(\text{packet}(c))) \geq t'_{OQ}(\text{first}(\text{packet}(c)))$, 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} :

$$t_{OQ}(c) \geq t'_{OQ}(\text{first}(\text{packet}(c))) \geq \text{CCF}'(\text{first}(\text{packet}(c))) = t_{PM-CCF}(c) - (L_{\max} - 1)$$

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.

A.3 Proof of Lemma 4.3

Consider a cell c , let k be its place in $packet(c)$, and let ℓ 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 algorithms ALG and CCF forward the cells of $packet(c)$ in order,

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

This implies that 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 deliver it from the CIOQ switch at this time-slot. Notice that ALG does not transmit two cells c, c' from the same output-port in the same time-slot, 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.

A.4 Proof of Lemma 4.4

Note that the CCF algorithm works with CIOQ switch with speedup 2. Thus, the number of cells c that share the same input-port (output-port) and have $(k-1)T < CCF(c) \leq kT$ is at most by $2T$. 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 $(k-1)T$. Since the maximum packet size is L_{\max} , the last cell of each packet was forwarded by the CCF after time-slot $(k-1)T$, and the number of input-ports (output-ports) is N , the number of such cells is bounded by $N(L_{\max} - 1)$ and the sum is bounded by $2T + N(L_{\max} - 1)$.

A.5 Proof of Theorem 4.5

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 4.4 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 of 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. Hence, Definition 5 implies that Algorithm 1 forwards all the cells corresponding to a specific packet contiguously: this clearly satisfies the packet-mode scheduling constraints.

All matchings used by Algorithm 1 are maximal and the sum of each column and each row in $B(k)$ is at most $2T + N(L_{\max} - 1)$. Theorem 4.2 implies that 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 the last cell $last(p)$ of some packet p . Definition 5 implies that if $last(p) \in a_{ij}(T, k)$ then $CCF(last(p)) > (k-1)T$. Since Algorithm 1 schedules $last(p)$ by time-slot $(k+1)T$, it follows that $last(p)$ maximum relative queuing delay is $2T - 1$. By Lemma 4.3, the maximum relative queuing delay is at most $2T + L_{\max} - 2$.

Algorithm 1 Coarse-Grained Maximal Matching Algorithm

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:        $P := \text{MAX-MATCH}(B, P)$  ▷ returns a maximal matching of  $B$  that dominates  $P$ 
7:     for all  $P[i][j]$  do
8:       if  $P[i][j] = 1$  and  $B[i][j] > 0$  then
9:         forward a cell from input-port  $i$  to output-port  $j$ 
10:     $B := B - P$ 
11:    for all  $B[i][j]$  do ▷ avoid negative values in matrix  $B$ 
12:       $B[i][j] := \max\{B[i][j], 0\}$ 

13: matrix procedure MAX-MATCH(matrix  $B$ , matrix  $P$ )
14:   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
15:      $P[i][j] = 1$ 
16:   return  $P$ 
```

A.6 Proof of Lemma 5.1

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

Assume, by way of 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 $\ell_2 \geq 0$ be the number of cells stored in the buffer of output-port j in time-slot $(k-1)T$. By the definition of an output-queued switch, 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} , the last cell of each packet arrives after time-slot $(k-1)T$, and the number of input-ports (output-ports) is N , the number of such cells is bounded by $N(L_{\max}-1)$, and the sum is therefore bounded by $T+B+N(L_{\max}-1)$.