

# On the speedup required for combined input and output queued switching

*Balaji Prabhakar*

Laboratory for Information and Decision Systems  
Massachusetts Institute of Technology.

*Nick McKeown\**

Departments of Electrical Engineering and Computer Science  
Stanford University.

## Abstract

Architectures based on a non-blocking fabric, such as a crosspoint switch, are attractive for use in high-speed LAN switches, IP routers, and ATM switches. These fabrics, coupled with memory bandwidth limitations, dictate that queues be placed at the input of the switch. But it is well known that input-queueing can lead to low throughput, and does not allow the control of latency through the switch. This is in contrast to output-queueing, which maximizes throughput, and permits the accurate control of packet latency through scheduling. We ask the question: Can a switch with combined input and output queueing be designed to *behave identically* to an output-queued switch? In this paper, we prove that if the switch uses virtual output queueing, and has an internal speedup of just four, it is possible for it to behave identically to an output queued switch, regardless of the nature of the arriving traffic. Our proof is based on a novel scheduling algorithm, known as *Most Urgent Cell First*. This result makes possible switches that perform as if they were output-queued, yet use memories that run more slowly.

## 1 Introduction

Many commercial switches and routers today employ output-queueing.<sup>1</sup> When a packet arrives at an output-queued (OQ) switch, it is immediately placed in a queue that is dedicated to its outgoing line, where it will wait until departing from the switch. This approach is known to maximize the throughput of the switch: so long as no input or

---

\*This research was initiated when Nick McKeown visited Balaji Prabhakar at BRIMS, Hewlett-Packard Labs, Bristol in September '96.

<sup>1</sup>When we refer to output-queueing in this paper, we include designs that employ centralized shared memory.

output is oversubscribed, the switch is able to support the traffic and the occupancies of queues remain bounded.

The use of a separate queue for each output means that flows of packets for different outputs are kept separate, and cannot interfere with each other. By carefully scheduling the time that a packet is placed onto the outgoing line<sup>2</sup>, a switch or router can control the packet's latency, and hence provide quality-of-service (QoS) guarantees. But output queueing is impractical for switches with high line rates, or with a large number of ports: the fabric and memory of an  $N \times N$  switch must run  $N$  times as fast as the line rate. Unfortunately, at the highest line rates, memories with sufficient bandwidth are simply not available. For example, consider a  $32 \times 32$  OQ switch operating at a line rate of 10Gbit/s. If we use a 512-bit memory datapath, we require memory devices that can perform both a write *and* a read operation every 1.6ns.

On the other hand, the fabric and the memory of an input queued (IQ) switch need only run as fast as the line rate. This makes input queueing very appealing for switches with fast line rates, or with a large number of ports. That is, for a given speed of memory it is possible to build a faster switch; or for a given speed switch it is possible to use slower, lower-cost memory devices. For example, consider again the  $32 \times 32$  switch operating at a line rate of 10Gbit/s. If the switch uses input-queueing instead of output-queueing, we can use memory devices that perform a write and a read operation every 51.2ns. This is readily achievable with commercially available memories.

But the main problem of IQ switching is head-of-line (HOL) blocking, which can have a severe effect on throughput. It is well-known that if each input maintains a single FIFO, then HOL blocking can limit the throughput to just 58.6% [7].

One method that has been proposed to reduce HOL blocking is to increase the "speedup" of a switch. A switch with a speedup of  $S$  can remove up to  $S$  packets from each input and deliver up to  $S$  packets to each output within a time slot, where a time slot is the time between packet arrivals at input ports. Hence, an OQ switch has a speedup of  $N$  while an IQ switch has a speedup of 1. For values of  $S$  between 1 and  $N$  packets need to be buffered at the inputs before switching as well as at the outputs after switching. We call this architecture a combined input and output queued (CIOQ) switch.

Both analytical and simulation studies of a CIOQ switch which maintains a single FIFO at each input have been conducted for various values of speedup [1, 2, 5, 6, 11]. A common conclusion of these studies is that with  $S = 4$  or 5 one can achieve about 99% throughput when arrivals are independent and identically distributed at each input, and the distribution of packet destinations is uniform across the outputs.

But it has been shown that a throughput of 100% can be achieved with a speedup of just one, if we arrange the input queues differently. That is, HOL blocking can be eliminated entirely using a scheme known as *virtual output queueing* in which each input

---

<sup>2</sup>By using schemes as proposed in [3, 12, 15], for example.

maintains a separate queue for each output. It has been shown that for independent arrivals, the throughput of an IQ switch can be increased to 100% [9]. We may draw the conclusion: *Speedup is not necessary to eliminate the effect of HOL blocking.*

In practice, we are not only interested in the throughput of a switch, but also in the latency of individual packets. This is particularly important if a switch or router is to offer QoS guarantees. Packets in an IQ switch not only contend for an output, they also contend for entry into the switch fabric with packets that are destined for other outputs. We call this phenomenon *input contention*. Each input can deliver only one packet into the fabric at a time; if it has packets for several free outputs, it must choose just one packet to deliver, holding other packets back. This places a packet at the mercy of other packets destined for other outputs. This is in stark contrast with output-queueing, where a packet is unaffected by packets destined for other outputs. We may draw the conclusion: *To control delay, we need a mechanism to eliminate input contention.*

Previous studies of CIOQ switches make no guarantees about the delay of an individual packet; instead they consider only average delay and throughput. While these results are academically interesting, they do not give us the principal benefit of output queueing: the ability to control the delay of individual packets. We believe that a well-designed network switch should perform predictably in the face of *all* types of arrival process, allowing the delay of individual packets to be controlled. Hence our approach is quite different, and our result subsumes previous work. Rather than find values of speedup that work well on average, or with simplistic and unrealistic traffic models, we find the minimum speedup such that a CIOQ switch behaves *identically* to an OQ switch for *all* types of traffic. Here, “behave identically” means that when the *same* inputs are applied to both the OQ switch and to the CIOQ switch, the corresponding output processes from the two switches are completely indistinguishable. Two processes are indistinguishable if and only if their packet sequences are identical – both in terms of packet-occurrence times and packet identities. Further, we place no restrictions on arrivals. Our results apply for any type of traffic, even if it saturates the switch.

The need for a switch that can deliver a certain grade of service, *irrespective of the applied traffic* is particularly important given the number of recent studies that show how little we understand network traffic processes [8, 14]. Indeed, a sobering conclusion of these studies is that it is not yet possible to accurately model or simulate a trace of actual network traffic. Furthermore, new applications, protocols or data-coding mechanisms may bring new traffic types in future years.

In this respect the formulation presented here is both novel and powerful: It allows us to obtain an algorithm that enables a CIOQ switch to perform exactly the same as an OQ switch, using memory devices operating more slowly, for arbitrary switch sizes, and for arbitrary input traffic patterns. Specifically, we exhibit a packet scheduling algorithm that enables a CIOQ switch with a speedup of four to mimic an OQ switch.

## 2 Exact mimicking of output-queueing

Consider the single stage,  $N \times N$  switch shown in Figure 1. Throughout the paper we assume that packets begin to arrive at the switch from time  $t = 1$ , the switch having been empty before that time. Although packets arriving to the switch or router may have variable length, we will assume that they are treated internally as fixed length “cells”. This is common practice in high performance LAN switches and routers; variable length packets are segmented into cells as they arrive, carried across the switch as cells, and reassembled back into packets again before they depart [13, 16]. We take the arrival time between cells as the basic time unit. The switch is said to have a *speedup of  $S$* , for  $S \in \{1, 2, \dots, N\}$  if it can remove up to  $S$  cells from each input and transfer at most  $S$  cells to each output in a time slot. A speedup of  $S$  requires the fabric of the switch to run  $S$  times as fast as the input or output line rate. As mentioned in the introduction, the extreme values of  $S = 1$  and  $S = N$  give a purely input-queued (IQ) and a purely output-queued (OQ) switch respectively. For  $1 < S < N$  buffering is required both at the inputs and at the outputs, and leads to a combined input and output queued (CIOQ) architecture. The following is the problem we wish to solve.

**The speedup problem:** Determine the smallest value of  $S$ , say  $S_{min}$ , and an appropriate cell scheduling algorithm  $\pi$  that

1. allows a CIOQ switch to exactly mimic the performance of an output-queued switch (in a sense that will be made precise),
2. achieves this for any arbitrary input traffic pattern,
3. is independent of switch size.

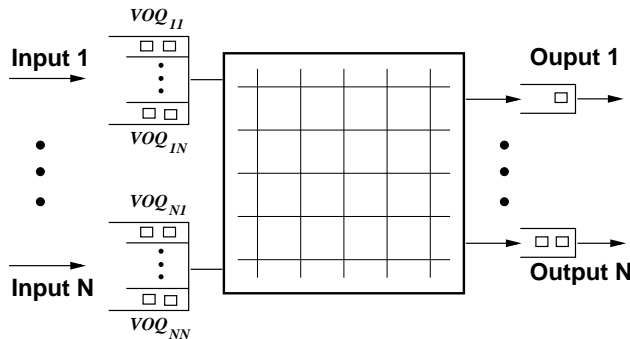


Figure 1: A schematic of a CIOQ switch

In an OQ switch, arriving cells are immediately forwarded to their corresponding outputs. This (a) ensures that outputs never idle so long as there is a cell destined for them in the system, and (b) allows the departure of cells to be scheduled to meet latency constraints. Because of these features an OQ switch has the highest possible throughput and allows a tight control of cell latency which is important for supporting multiple qualities-of-service (QoS). We will require that any solution of the speedup problem possess these

two desirable features; that is, a CIOQ switch must *exactly mimic* the performance of an OQ switch in the following sense.

**Identical Behavior:** Consider an OQ switch whose output buffers are first-in-first-out (FIFO). A CIOQ switch is said to *behave identically* to an OQ switch if, under identical inputs, the departure time of every cell from both switches is identical.

To complete the description of the model, we refer to Figure 1 again. All input and output buffers are assumed to have infinite capacity. Each input maintains a separate FIFO queue for cells destined for each output. Hence, there are  $N$  FIFO queues at each input. Call these queues *Virtual Output Queues (VOQs)* – with the understanding that  $VOQ_{ij}$  buffers cells at input  $i$  destined for output  $j$ . Finally, we wish to make explicit the assumption that the output buffers of the CIOQ switch are not necessarily FIFO, although the OQ switch whose performance it is mimicking has FIFO output buffers.

A scheduling algorithm selects a matching between inputs and outputs in such a way that each non-empty input is matched with at most one output and, conversely, each output is matched with at most one input. The matching is used to configure the switch before cells are transferred from the input side to the output side. A CIOQ switch with a speedup of  $S$  is able to make  $S$  such transfers each time slot.

### 3 MUCFA: A scheduling algorithm that achieves identical behavior

In this section we present a novel scheduling algorithm that allows a CIOQ switch with a small speedup to behave identically to an OQ switch for any input traffic. The algorithm is called the *Most Urgent Cell First Algorithm* (MUCFA).

We begin by introducing the notion of a “phase”.

**Definition** *For a switch with speedup  $S$ , a time slot is said to be divided into  $S$  equal phases. During each phase  $\phi_i$ ,  $1 \leq i \leq S$ , the switch can remove at most one cell from each input and can transfer at most one cell to each output.*

It is assumed that cells arriving at the switch will do so at the beginning of phase  $\phi_1$ , while departures from the switch take place at the end of phase  $\phi_S$ .

A crucial aspect of MUCFA is the concept of the “urgency of a cell”. Recall that the definition of “identical behavior” requires a CIOQ switch to *identically* match cell departures with an OQ switch when they are both subjected to identical inputs. Therefore, our definition of identical behavior requires a CIOQ switch and a reference OQ switch. This is illustrated in Figure 2.

The urgency of a cell is first explained with respect to the reference OQ switch. Every arriving cell to this switch is stamped with a number, which is its “urgency value” at that time. This number indicates the time from the present that it will depart from the switch. At each successive time slot, the urgency value is decremented by one. When the value reaches zero, the cell will depart. Alternatively, since the buffers of the OQ switch are FIFO, the urgency of a cell at any time equals the number of cells ahead of it in the output buffer at that time.

More precisely, if a cell  $c$  arrives at input  $i$  at time  $T$  and departs from output  $j$  at time  $D \geq T$ , its urgency at any time  $R$ ,  $T \leq R \leq D$ , equals  $D - R$ . Suppose there are two cells,  $a$  and  $b$ , in the buffer at output  $j$  at some time, with urgencies  $u_a$  and  $u_b$ , respectively. Cell  $a$  is said to be “more urgent” than  $b$  if  $u_a < u_b$ . Given that the output buffer is FIFO, it is clear that if  $b$  arrived at the switch after  $a$  then necessarily  $u_a < u_b$ . If  $a$  and  $b$  arrive at the same time, then  $u_a < u_b$  iff the number of the input port at which  $a$  arrives is less than the number of the input port at which  $b$  arrives. That is, the OQ switch is assumed to transfer cells from inputs to outputs in a round robin fashion starting with the smallest numbered input first.

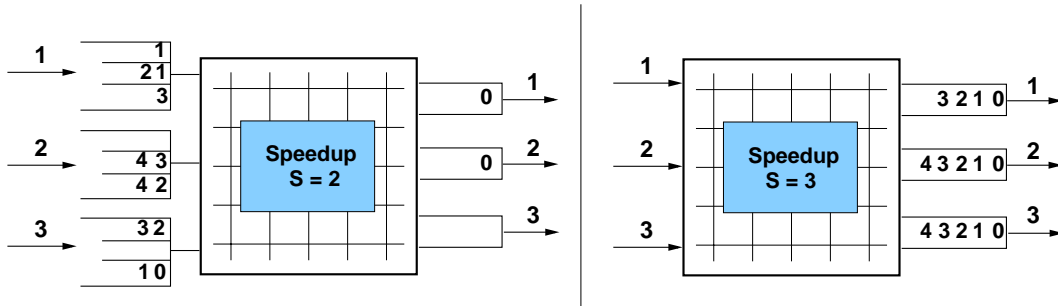


Figure 2: A CIOQ switch (left) and its reference OQ switch (right).

Now consider the CIOQ switch. By assumption, the same input is applied to it and to the OQ switch. Therefore, cell  $c$  arrives at input  $i$  at time  $T$  and is destined for output  $j$ . Since the speedup may now be less than  $N$ ,  $c$  may not be forwarded to the buffer at  $j$  during time slot  $T$ . Note that  $c$  may not be required at output  $j$  for some time, because its clone in the OQ switch is some distance from the HOL. Therefore, the urgency is an indication of how much time there is before  $c$  is needed at its output if the CIOQ switch is to mimic the behavior of the OQ switch. This motivates the following definition.

**Definition** The urgency of a cell in a CIOQ switch at any time is the distance its clone is from the head of the output buffer in the corresponding reference OQ switch.

The cells in any output buffer of the CIOQ switch are arranged in increasing order of urgencies, with the most urgent cell at the head. Once cell  $c$  is forwarded to its output in the CIOQ switch, its position is determined by its urgency.

We are now ready to describe the Most Urgent Cell First Algorithm (MUCFA).

### Phase-by-phase description of MUCFA

1. At the beginning of each phase outputs try to obtain their most urgent cells from the inputs.
2. If more than one output requests an input, then the input will grant to that output whose cell has the smallest urgency number. If there is a tie between two or more outputs, then the output with the smallest port number wins.
3. Outputs that lose contention at an input will try to obtain their next most urgent cell from another input.
4. When no more matching of inputs and outputs is possible, cells are transferred and MUCFA goes to the next phase (Step 1).

The operation of MUCFA over one time slot is illustrated by means of an example in Figure 3. Note that at the beginning of phase 1, both outputs 1 and 2 request input 1 to obtain their most urgent cells. Since there is a tie in the urgency of their requests, by our assumption input 1 grants to output 1. Output 2 proceeds to obtain its next most urgent cell which happens to be at input 2 and has an urgency of 3.

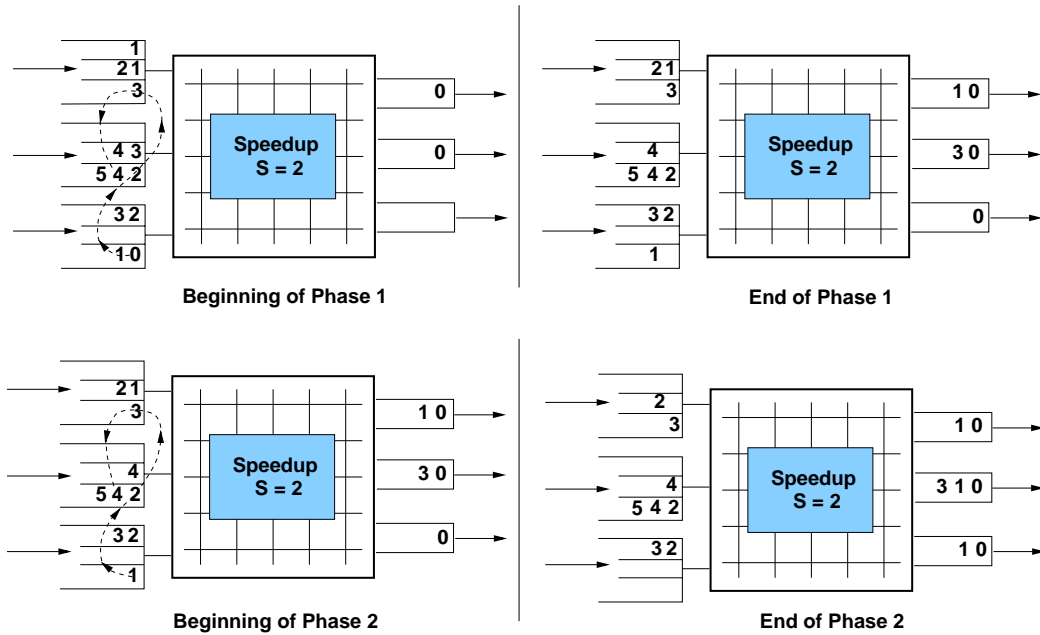


Figure 3: The operation of a  $3 \times 3$  CIOQ switch with  $S = 2$  over one time slot under MUCFA. The dashed lines indicate the “output thread” of the cell with urgency 5 in  $VOQ_{23}$ .

We can make the following key observation about the working of MUCFA: During

any phase, there are exactly two reasons that a cell will not be transferred from its input to its output.

1. **Input contention:** The output is ready to receive the cell, but the input wants to send a more urgent cell. (In the example of Figure 3, output 2 can't receive its most urgent cell in phase 1 because input 1 wants to send to output 1.)
2. **Output contention:** The input wants to send the cell, but the output wants to receive a more urgent cell. (In phase 2 of the example of Figure 3, input 2 can't send its most urgent cell because output 3 wants to receive from input 3.)

### 3.1 MUCFA and the Stable Marriage Problem

The way in which MUCFA matches inputs and outputs is a variation of the *stable marriage problem*, which was first introduced by Gale and Shapley in 1962 [4]. Solutions to the stable marriage problem find a “stable” and complete matching between inputs and outputs. A match is *unstable* if there is an input and an output that are not matched to each other, yet both prefer the other to their partner in the current matching. A *stable* matching is any matching that is not unstable. There exists a well-known algorithm (the Gale-Shapley algorithm, or GSA) that will always find a stable matching in  $N$  iterations.

MUCFA can be implemented using the GSA with preference lists as follows. Output  $j$  first assigns a preference value to each input  $i$ , equal to the urgency of the cell at head-of-line of  $VOQ_{ij}$ . If  $VOQ_{ij}$  is empty then the preference value of input  $i$  for output  $j$  is set to  $+\infty$ . The preference list of the output is the ordered set of its preference values for each input. Likewise, each input assigns a preference value for each output, and creates the preference list accordingly. A matching of inputs and outputs can then be obtained using GSA. The relationship between the stable marriage problem and cell scheduling is explored in more detail in [10].

## 4 The main result

**Theorem 1** *An  $N \times N$  CIOQ switch operating under MUCFA can behave identically to an OQ switch, regardless of input traffic patterns and for arbitrary values of  $N$ , if its speedup  $S \geq 4$ .*

Theorem 2, which is a strengthening of Theorem 1 will be proved in the next section. For now we will explore some of the implications of Theorem 1, assuming that it is true. This will allow us to come to certain conclusions which help in the statement and proof of Theorem 2. In order to proceed, we will need to introduce the concept of “output threads” and “input threads”.

**Definition** *At any time, the output thread of a cell  $c$  queued in  $VOQ_{ij}$  is the ordered set of all cells  $c'$  which are queued in  $VOQ_{i'j}$ ,  $1 \leq i' \leq N$ , and are more urgent than  $c$ . The thread of output  $j$  is the output thread of its least urgent cell.*

For example, the output thread of the cell with urgency five in  $VOQ_{23}$  at the beginning of phase 1 (see Figure 3) has cells with urgencies  $\{0, 1, 2, 3, 4\}$ . The output thread of the same cell at the beginning of phase 2 has cells with urgencies  $\{1, 2, 3, 4\}$ . The dashed lines in Figure 3 indicate the output thread of this cell at the beginning of phases 1 and 2.

**Definition** *The input thread of a cell  $c$  queued in  $VOQ_{ij}$  is the ordered set of all cells  $c'$  which are in  $VOQ_{ij'}$ ,  $1 \leq j' \leq N$ , and are more urgent than  $c$ . If cells  $p$  and  $q$  have the same urgency then  $p$  is placed before  $q$  in an input thread if  $p$ 's output has a smaller number than  $q$ 's output. The thread of input  $i$  is the input thread of its least urgent cell.*

For example, the input thread of the cell with urgency three in  $VOQ_{13}$  at the beginning of phase 1 (see Figure 3) has cells with urgencies  $\{1, 1, 2\}$ . The input thread of the same cell at the beginning of phase 2 has cells with urgencies  $\{1, 2\}$ .

With these definitions, one may draw some inferences about MUCFA. (The following discussion is intended to motivate the statement and proof of Theorem 2 and is therefore presented in an informal manner.) Consider a CIOQ switch with speedup  $S$  operating under MUCFA from time 1, having been empty before that time. It will fail to behave identically to an OQ switch at time  $T$  if an input thread has  $S + 1$  or more cells with urgency 0. If this should happen, then clearly there are not enough phases to transfer all the most urgent cells to their outputs, and MUCFA fails. Therefore, if MUCFA causes a CIOQ switch with speedup  $S$  to behave identically as an OQ switch, it must be the case that *every input thread has  $S$  or fewer cells with urgency 0 at the beginning of every time slot*. Conversely, if there are always  $S$  or fewer cells with urgency 0 at each input, then MUCFA never fails. We record this in the following lemma.

**Lemma 1** *A CIOQ switch with speedup  $S$  operating under MUCFA behaves identically to an OQ switch if, and only if, there are  $S$  or fewer cells with urgency 0 in each input at all times.*

Since cells in an input thread are ordered according to urgency, this is the same as saying that a cell with urgency 0 cannot appear in the  $(S + 1)^{th}$  position in any input thread. Similarly, it is also clear that a cell with urgency 1 cannot appear in the  $(2S + 1)^{th}$  position at any time (assuming that every 0 occupies a position less than or equal to  $S$ ), as this would lead to a failure of MUCFA in the next cell time. In general, Lemma 1 is equivalent to the statement: A CIOQ switch with speedup  $S$  operating under MUCFA behaves identically to an OQ switch if, and only if, a cell with urgency  $l$  cannot occupy position  $(l + 1)S + 1$  in an input thread at any time.

If we assume that MUCFA behaves identically to an OQ switch at all times when the speedup equals  $S$ , it is clear that it will also behave identically at every speedup  $S' > S$ .

Indeed, more ought to be true: Under identical inputs if a tagged cell  $c$  is forwarded to its output  $F$  phases after its arrival when the speedup is  $S$ , then it must be forwarded to its output within  $F' \leq F$  phases when the speedup is  $S'$ . In particular, if  $c$  belongs to the thread of input  $i$  at time  $T$  when the speedup is  $S'$ , then it also belongs to the thread of input  $i$  at time  $T$  when the speedup is  $S$ . This implies the following crucial point.

**Key observation** *If MUCFA behaves identically to an OQ switch at speedup  $S$ , then at any speedup  $S' \geq S$  a cell with urgency  $l$  cannot appear at position  $S(l+1)+1$  in an input thread.*

In Theorem 2 we prove the following stronger statement for  $S \geq 4$ : At the beginning of each time slot  $T$ , a cell with urgency  $l$  does not occupy position  $l+1$  in an input thread; excluding any cell that might have just arrived. If this property were true of all input threads at all times then clearly MUCFA never fails to behave identically to an OQ switch, and Theorem 1 is verified.

## 5 A speedup of 4 suffices

In this section we shall prove Theorem 2 from which Theorem 1 follows as a corollary. But first, we need to develop the following lemma.

**Lemma 2** *Consider a tagged cell  $c$  which, at the beginning of time slot  $T$ , is at an input of a CIOQ switch with speedup  $S$  operating under MUCFA. If  $c$  remains in its input at the end of time slot  $T$  and is not forwarded to its output, then a totality of  $S$  cells either from  $c$ 's input thread or from its output thread must be delivered to their outputs during time slot  $T$ .*

**Proof** This is a consequence of input and output contention. That is,  $c$  is not forwarded to its output during a phase either because a cell in its input (output) thread has kept its input (output) busy. And there are  $S$  such phases in each time slot. ■

**Theorem 2** *Consider an  $N \times N$  CIOQ switch operating under MUCFA with a speedup of  $S$ . Suppose that the switch has been operating from time slot 1, having been empty before that time. Let  $S^i(t)$  be the thread at input  $i$  just at the beginning of time slot  $t$ , before any new cells have arrived. Then for each  $i$  and for each  $t$ , it is never the case that a cell with urgency  $l$  occupies position  $l+1$  in  $S^i(t)$  so long as  $S \geq 4$ .*

**Proof** The proof is by contradiction. Suppose  $T$  is the first time that such a thing happens at some input, say  $I$ . That is,  $S^I(T)$  has a cell of urgency  $l$  occupying position  $l+1$ . Consider the thread  $S_{l+1}^I(T) \subset S^I(T)$  consisting of the first  $l+1$  cells of  $S^I(T)$ . Note that the least urgent cell of  $S_{l+1}^I(T)$  has an urgency of  $l$ .

- (1) Let  $c$  be the cell belonging to  $S_{l+1}^I(T)$  that arrived earliest, and let  $u$  be its urgency at time  $T$ . It follows that  $u \leq l$ . It also follows that  $c$  arrived at least  $l + 1$  cell times ago.
- (2) Suppose  $c$  actually arrived at time  $T - A$ . By (1)  $A \geq l + 1$ , and  $c$ 's urgency upon arrival equals  $u + A$  precisely.
- (3) By Lemma 2, every time slot that  $c$  is in the system on the input side, a totality of  $S$  cells belonging to the input and/or output threads of  $c$  must be sacrificed in order to prevent  $c$  from going to its output.
- (4) Since  $c$  arrives at time  $T - A$  and remains in its input until time  $T - 1$ , the number of "sacrifice cells" required during this time period equals the number of phases in  $[T - A, T - 1]$  which equals  $S \times A$ .
- (5) By assumption of  $T$  being the first time at which things go wrong, the maximum number of cells in  $c$ 's input thread at time  $T - A$  is less than or equal to  $u + A$ . These are possible "sacrifice cells".
- (6) By definition of urgency, the maximum number of cells in  $c$ 's output thread at time  $T - A$  is less than or equal to  $u + A$ . These are also possible "sacrifice cells".
- (7) Putting (5) and (6) together, when  $c$  arrives, the maximum number of sacrifice cells in its input and output threads is no more than  $2(u + A)$ .
- (8) Between  $T - A + 1$  and  $T - 1$ , the maximum number of cells that can arrive at input  $I$  is less than or equal to  $A - 1$ . Of these arrivals  $l$  will belong to  $S_{l+1}^I(T)$  and hence cannot be "sacrifice cells". This implies that the maximum number of sacrifice cells that can arrive at input  $I$  after  $c$  is no more than  $A - 1 - l$ .
- (9) A grand total on the maximum possible "sacrifice cells" is (putting (7) and (8) together):  $2(u + A) + A - 1 - l = 3A + u + (u - l) - 1$ . But,

$$\begin{aligned}
3A + u + (u - l) - 1 &\leq 3A + u && \text{(since } u \leq l\text{)} \\
&\leq 4A - 1 && \text{(since } u \leq l \leq A - 1\text{)}.
\end{aligned}$$

- (10) The number in (9) falls short of the requirement in (4) if  $S \geq 4$ . This contradiction proves the theorem. ■

## 6 Conclusion

With the continued demand for faster and faster switches, it is increasingly difficult to implement switches that use output queueing or centralized shared memory. Before long, it may become impractical to build the highest performance switches and routers using these techniques.

It has been argued for some time that most of the advantages of output-queuing (OQ) can be achieved using combined input and output queueing (CIOQ). While this has been argued for very specific, benign traffic patterns there has always been a suspicion that the advantages would diminish in a more realistic operating environment.

Our result proves that a CIOQ switch can *behave identically* to an OQ switch, or one using centralized shared memory. Perhaps more importantly, we show this is true for any sized switch, or for any traffic arrival pattern. The three sufficient conditions for this result to hold are: (i) virtual output queues are maintained at each input, (ii) at the end of each cell time, a novel scheduling algorithm, which we call *Most Urgent Cell First* be used to configure the non-blocking switch fabric, and (iii) the switch fabric and memory run four times as fast as the external line rate; i.e. at a speedup of four.

**Acknowledgement:** The authors thank Shang-tse Chuang and Mingyan Zhu of Stanford University for various discussions of the speedup problem. Nick McKeown also thanks Jeremy Gunawardena, Scientific Director of Hewlett-Packard's Basic Research Institute in the Mathematical Sciences (BRIMS), for inviting him to BRIMS where much of this work was initiated.

## References

- [1] C-Y. Chang, A.J. Paulraj, T. Kailath, "A Broadband Packet Switch Architecture with Input and Output Queueing," *Proc. Globecom '94*, pp.448-452.
- [2] J.S.-C. Chen, T.E. Stern, "Throughput analysis, optimal buffer allocation, and traffic imbalance study of a generic nonblocking packet switch," *IEEE J. Select. Areas Commun.*, Apr. 1991, vol. 9, no. 3, pp. 439-49.
- [3] A. Demers, S. Keshav, S. Shenker, "Analysis and simulation of a fair queueing algorithm", *Internetworking Research and Experience*, vol. 1, 1990.
- [4] D. Gale, L.S. Shapley, "College Admissions and the stability of marriage", *American Mathematical Monthly*, vol.69, pp.9-15, 1962.
- [5] A.L. Gupta, N.D. Georganas, "Analysis of a packet switch with input and output buffers and speed constraints," *Proc. InfoCom '91*, Bal Harbour, FL, April 1991, pp.694-700.
- [6] I. Iliadis, W.E. Denzel, "Performance of packet switches with input and output queueing," *Proc. ICC '90*, Atlanta, GA, April 1990. pp.747-53.
- [7] M. Karol, M. Hluchyj, S. Morgan, "Input Versus Output Queueing on a Space Division Switch", *IEEE Trans. Comm.*, vol.35, no.12, pp.1347-1356.
- [8] W.E. Leland, W. Willinger, M. Taqqu, D. Wilson, "On the self-similar nature of Ethernet traffic", *Proc. of Sigcomm*, San Francisco, pp.183-193. Sept 1993.

- [9] N. McKeown, V. Anantharam, J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch", *INFOCOM '96*, pp.296-302.
- [10] N. McKeown, "Scheduling Algorithms for Input-Queued Cell Switches", *PhD Thesis, University of California at Berkeley*, May 1995.
- [11] Y. Oie, M. Murata, K. Kubota, H. Miyahara, "Effect of speedup in nonblocking packet switch," *Proc. ICC '89*, Boston, MA, June 1989, pp. 410-14.
- [12] A.K. Parekh, R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case", *IEEE/ACM Transactions on Networking*, vol. 1, 1993.
- [13] C. Partridge, et al. "A fifty gigabit per second IP router," To appear in *IEEE/ACM Transactions on Networking*.
- [14] V. Paxson, S. Floyd, "Wide area traffic: the failure of Poisson modeling." *IEEE/ACM Transactions on Networking* Vol.3, no.3, pp.226-44. June 1995.
- [15] L. Zhang, "A New Architecture for Packet Switching Network Protocols", PhD Thesis, *Dept. of Electrical Engineering and Computer Science, MIT*, 1989.
- [16] Cisco Systems GSR 12000 Technical Product Description, <http://www.cisco.com/warp/public/733/12000/index.shtml>.