Output Queued Switch Emulation By A One-Cell-Internally Buffered Crossbar Switch*

Lotfi Mhamdi and Mounir Hamdi
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
Email: lotfi@cs.ust.hk

Abstract — **Output-Queued (OQ) switching architecture is known to be of optimal performance amongst all queuing approaches. However, OQ switches were always known to lack scalability due to the high memory bandwidth constraints. Extensive research work showed that an OQ switch can be exactly emulated by a more scalable crossbar switch (i.e., Input-Queued (IQ) switch) and a small speedup[9]. Unfortunately, this important result was of no practical use due to the high complexity of the proposed scheduling scheme. A similar result was shown in [11] and was based on the Internally Buffered Crossbar (IBC) switching architecture. While the latter result seems to overcome the complexity issue, the scheduling scheme presented, especially the time stamping mechanism performed by the OCF output scheduling scheme, is costly.**

In this paper, we extend our previous work in [6] and prove the same result as in [11] but with less hardware requirements. In particular, we propose a simple scheduling scheme, named *Modified Current Arrival First– Lowest TTL First* (MCAF-LTF), that doesn't require costly time stamping mechanism. Based on the MCAF-LTF, we prove that, with a speedup of just 2, a one-cell-internally buffered crossbar switch can exactly emulate an OQ switch. The reduced complexity of our proposed scheme makes it of high practical value and allows it to be readily implemented in such ultra-high capacity network.

Index Terms—**Buffered crossbar fabric, OQ emulation.**

I. INTRODUCTION

Switches and routers have most often been designed with output queuing strategy. Further to achieving high throughput (100%), a switch or router built around an OQ architecture can control packet's latency and therefore provide guaranteed quality-of-service (QoS) [1][4]. In fact, most of the QoS algorithms that have been proposed assume that the underlying architecture is an "ideal" OQ switch. As a result, in order to take advantage of these QoS algorithms, we either use an OQ switch or we use a switch architecture that can emulate, or mimic, an OQ switch so that all its properties can be inherited.

An OQ switch, however, has no queues at the ingress ports. All arriving cells must be immediately delivered to their outputs. A major disadvantage is that simultaneous delivery of all arriving cells to the outputs requires high internal interconnection bandwidth and memory bandwidth. For an *N*x*N* OQ switch, the memory has to support up to *N* write accesses (to write *N* cells into the output buffer) and one read access (to send one cell to the outgoing link) in one-cell time. This means that the OQ switch must operate *N*+1 times faster than the line rate. This requirement is known as internal speedup of a switch [9]. To ensure that there are no packets queued at the input ports, it is widely believed that an OQ switch has to have an internal speedup of *N*. Unfortunately, the increase in line rate and/or switch size makes it extremely difficult and impractical to build memories with adequate capacity for such high-bandwidth.

IQ switches, on the other hand, are desirable because of their scalability and low hardware requirements. The IQ switch has an internal speed up of 1 because the crossbar fabric has the same speed as that of the external line. It is well known that, if FIFO input queues are used to hold arriving packets, head-of-line (HoL) blocking problem limits the throughput to only 58,6%[7]. In addition to the Virtual Output Queuing (VOQ) architecture [10], one of the solutions that have been proposed to overcome the HoL problem is the use of speed up. In [3] and [5], it has been shown that a crossbar switch with a single FIFO at the input can achieve up to 99% throughput under certain assumptions on the input traffic statistics for speed up range between 4 and 5.

When the internal speedup is between *1* and *N*, buffering is required at both the inputs and outputs. Hence, a combination of an input buffered and an output buffered switch is required, i.e., Combined Input and Output Buffered switch (CIOQ). In [2], it was proven that a speedup of 4 is sufficient for a CIOQ to exactly emulate an OQ switch when the scheduling policy used is MUSF (Most Urgent Cell First). An improved result was found in [9], with a speedup of just two a CIOQ behaves identically like an OQ and a speedup of 2-1/*N* is sufficient to mimic a FIFO-OQ switch. This important result was found at the expense of a very complex scheduling policy called CCF

* This work was supported in part by the Hong Kong Research Grant Council (Grant Number: RGC HKUST6181-01E).

(Critical Cell First). An attempt to reduce the complexity of this algorithm was based on a strategy called DTC (Delay Till Critical), to reduce the number of iterations from N^2 to N, along with an algorithm called GBVOQ (Group-By-Virtual-Output Queue), to reduce the information complexity. Unfortunately, these two solutions cannot be combined since they are mutually exclusive.

In this paper, we adopt a one-cell Internally Buffered Crossbar fabric (IBC) switch with input VOQs. Throughout this paper, we will refer to this architecture as VOQ/IBC architecture. The VOQ/IBC, which was first introduced by [8], was shown to have great potential in eradicating the centralized scheduler's bottleneck and solving the scalability challenges faced by the buffer less crossbar switch architecture. In fact we propose a modified version of the CAF input scheduling scheme presented in [6]. The new scheme is named Modified CAF (MCAF). We also propose an output scheduling scheme named Lowest Time to leave First (LTF). We prove that a VOQ/IBC switch employing the MCAF_LTF and running twice as fast as the external line rate can exactly emulate a FIFO OQ switch irrespective of the switch size. While previous results [9][11] show the same result, the complexity of the schemes presented was a bottleneck. Our MCAF_LTF scheme, however, is simple in hardware and can be implemented at low cost.

The rest of the paper will be organized as follows: Section two introduces the background knowledge, describes the architecture and illustrates the definitions used throughout this article. Section three presents the MCAF_LTF scheme, illustrates the theorems and provides sufficient proof for OQ emulation by VOQ/IBC switch with a speed up of two. Finally section four concludes the paper.

II. BACKGROUND AND DEFINITIONS

1. *Background knowledge*

As shown in Figure1, the VOQ/IBC switch architecture consists of N input cards. Each card maintains N VOQs, one per output. The fabric part is the main characteristic of the VOQ/BCF and this differentiates it from the IQ buffer less architecture.

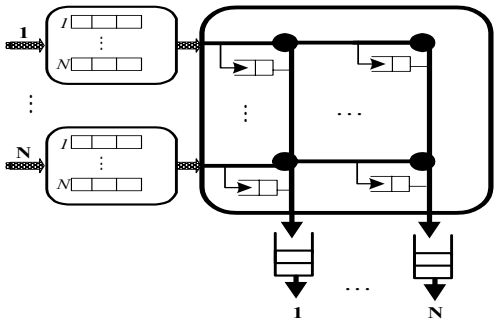

Figure 1: The VOQ/IBC Architecture.

When a packet, destined to output j, $1 \le j \le N$, arrives to the input card i, $1 \le i \le N$, it is held in VOQi,j. A VOQi,j is said to be eligible for being scheduled in the input scheduling process if it is not empty and the internal buffer XPi,j is empty (or free).

The internal fabric consists of N^2 buffered crosspoints (XP). Each crosspoint has a one-cell buffer. A crosspoint XPi,j, holds cells coming from input i and going to output j. Since the internally buffered crossbar fabric contains speedup, queuing is required at the outputs as well and each output maintains an output queue.

A scheduling cycle consists of the following three steps: input scheduling, output scheduling and delivery notifying. During the input scheduling, each input, i, selects, in an independent and parallel way, an eligible VOQ and sends its HoL cell to the internal buffer. Likewise, each output, j, selects, independently and in parallel, a non empty crosspoint buffer, XPi,j, and sends its cell to the output queue. Then, the delivery notifying is performed to carry the flow control between the internal buffers and the input queues. For each delivered cell, the flow control mechanism "informs" the corresponding input of the internal buffer status. That is, change the status of the corresponding VOQ to be eligible.

Fixed size packets, or cells, are considered. Upon arrival to the switch, variable length packets are segmented into cells for internal processing and re-assembled before they leave the switch. A processing cycle has fixed length, called cell or time slot. Since the VOQ/IBC has a speedup of two, each time slot is divided into the following four phases:

- Arrival phase: All arrivals occur during this phase. This phase ends just prior to the first scheduling phase.
- First scheduling phase: a scheduling cycle is performed during this phase.
- Second scheduling phase: a second scheduling cycle is performed during this phase.
- Departure phase: All cells departures occur during this phase. The end of this phase coincides with the end of a time slot.

Now, that we present the architecture and scheduling, we will, in the following section give some definitions that will be often used through the rest of this paper.

2. *Definitions*

Through the rest of this article, we will often refer to the following definitions, which are similar to those presented in [11].

1. *Input Priority List* (*IPL*): Each input scheduler, *i*, maintains an input priority list, *IPL*, of all cells queued at input *i*. The *IPL* of an input scheduler determines the departure order of cells from that input to the internal buffers.
2. *Time-to-Leave* (*TTL*): equals the timeslot during which cell *c* departs as specified by the shadow OQ switch. Note

that all cells, destined for the same output, must have distinct *TTL*s.

3. *Push-In First-Out Queues (PIFO):* As defined in [6], cells are inserted into a PIFO queue based on their TTL field. A cell *c*, destined to a PIFO queue, is inserted ahead of all cells with a greater TTL and behind all cells with smaller TTL.

4. *Output Priority List* (*OPL*): Each output scheduler, *j*, maintains an output priority list, *OPL*, of all cells queued at the internal buffers and destined for output *j*. The *OPL* of an output scheduler is composed of two queues: a FIFO and a PIFO. The cell ordering in the FIFO and the PIFO queues compose the ordering of the OPL and determines the departure order of cells from the internal buffers to the output queue.

5. *Shadow OQ switch*: A theoretical OQ switch that determines the departure order and time of each cell from the VOQ/IBC to exactly emulate an OQ.

6. *Input Thread* (*IT*): The input thread of a cell *c*, *IT*(*c*), is equal to the number of cells ahead of *c* in its input priority list, *IPL*. *IT*(*c*) is defined for each cell queued at an input port. It becomes zero as soon as cell *c* is selected for input scheduling. *IT*(*c*) is influenced by the arrival and input scheduling phases, respectively. A newly arriving cell may cause *IT*(*c*) to increment. However, an input scheduling phase may cause *IT*(*c*) to decrement.

7. *Output Cushion* (*OC*): The output cushion of a cell *c* is equal to the number of cells at *c*'s output queue with lower *TTL* than *c*. Unlike *IT*(*c*), *OC*(*c*) is influenced by the output scheduling and the departure phases, respectively. An output scheduling phase may cause *OC*(*c*) to increment. Conversely, a departure phase may cause *OC*(*c*) to decrement. *OC*(*c*) doesn't change during an input scheduling phase.

8. *Slackness* (*L*): Every time slot, the slackness of cell *c*, *L*(*c*), equals the output cushion of cell *c* minus its input thread. That is, $L(c) = OC(c) - IT(c)$. The slackness is defined for cells queued either at an input port or at a crosspoint buffer.

The slackness of a cell *c* determines the urgency of *c*'s transfer from its incoming port to its outgoing port [9]. Recall that emulating OQ means that every cell must reach its output queue on or before its time to leave as specified by the shadow OQ. The OQ emulation process is highly influenced by the slackness of every cell, *c*, inside the system. Any increase in *L*(*c*) is translated by either an increase in *OC*(*c*) or a decrease in *IT*(*c*). In both cases, *L*(*c*) increases and there is no fear for *c* of reaching its output on time. Any decrease in *L*(*c*), however, is translated by either a decrease in *OC*(*c*) or an increase in *IT*(*c*). In both cases, *L*(*c*) decreases and *c* should be urgently transferred to its output queue before it misses its time to leave. As a result, in order for the OQ emulation to occur, we have to ensure that the slackness of every cell inside the switch is always positive and non-decreasing.

For a VOQ/IBC switch running at a speedup of 2, each time slot consists of four phases as mentioned earlier. At each time slot every cell, *c*, can have one of the following status: just arrived, selected for input scheduling, not selected for input scheduling (blocked by a flow control), selected for output scheduling, not selected for output scheduling(blocked by a more urgent cell) or departs the switch. Note that we no longer get concerned about any cell, *c*, that either reaches its output queue (i.e., the status: selected for output scheduling) or departs the switch. The OQ emulation takes place if, irrespective of its status, any cell, *c*, has a non negative slackness. In the following section, we propose a scheduling scheme along with its complete proof that a VOQ/IBC switch running at a speedup of 2 can exactly emulate an OQ switch.

III. FIFO-OQ EMULATION

This section provides the specification of our proposed scheduling scheme along with the sufficient conditions that prove that, with a speedup of two, a VOQ/IBC switch can exactly emulate an OQ switch. The specification of each of scheduling phase is as follows:

- ***Input Scheduling (MCAF)***

  Each input, *i,* maintains its *IPL* as follows:
  If there is a currently arriving cell, *c*, to a $VOQ_{i,j}$
    Then insert *c*, just after the last entry[1] of $VOQ_{i,j}$ in *IPL*
  If $VOQ_{i,j}$ is eligible
    Then: If $VOQ_{i,j}$ contains other cells than *c*
      Then move the HoL cell of $VOQ_{i,j}$ to the front of *IPL* and assign *c* a priority flag '*P*'.
      Else move the HoL cell of $VOQ_{i,j}$ to the front of *IPL* and assign *c* a priority flag '*F*'.
  Serve cells based on *IPL* order.

- ***Output Scheduling (LTF)***

  Each output, *j*, maintains its *OPL* as follows:
    If *c*'s priority flag is '*P*'
      Then insert *c* into the $PIFO_j$.
      Else insert *c* into the tail of the $FIFO_j$.
  Each output compares the two HoL cells of $PIFO_j$ and $FIFO_j$ and move the one with Lowest *TTL* to the front of the *OPL*.
  Serve cells based on *OPL* order.

With the above specification of the *MCAF_OCF* scheme, we will show that, upon its arrival, every cell, *c*, is inserted with a non negative slackness. Then, as time goes on and so long as the cell *c* is inside the switch (either at input VOQ or inside the internal buffer), its slackness never decreases.

[1] Note that the last entry of an empty VOQ equals to its first entry which equals also to the front of the *IPL*.

Lemma 1: The *MCAF_LTF* scheduling scheme satisfies the non-negative slackness (*NNS*) insertion property for a VOQ/IBC switch running at a speedup of 2.

Proof: (by induction)

Suppose that lemma 1 held up until time slot t. We show that lemma 1 holds at time slot $t+1$.

At time slot $t+1$, a new arriving cell, c, can arrive to either an empty or a non empty *VOQ*, as follows:

- *Case 1*: c arrives to an empty *VOQ*

  Based on MCAF scheme, an empty VOQ to which arrival occurs become highest priority and has the following:

  $IT(c) = 0,\ OC(c) \geq 0,$ thus $L(c) = OC(c) - IT(c) \geq 0$ (1)

- *Case 2*: c arrives to a non empty *VOQ*

  We know that the *NNS* property held up until the end of time slot t. Suppose that cell, c', behind which c is inserted had a positive slackness of $L_t(c')$ at time t. For instance, suppose that the slackness of every cell inside the switch never decreases from time slot to the next (we are going to prove this later).This means that:

  $L_{t+1}(c') = OC_{t+1}(c') - IT_{t+1}(c') \geq L_t(c')$ (2)

  Cell c is behind c', that is:

  $IT_{t+1}(c) = IT_{t+1}(c') + 1$ (3)

  $OC_{t+1}(c) \geq OC_{t+1}(c') + 1$ (both cell are destined to the same output and $TTL(c') < TTL(c)$) (4)

  (2), (3) and (4) imply:

  $$L_{t+1}(c) = OC_{t+1}(c) - IT_{t+1}(c) \geq (OC_{t+1}(c') + 1) - (IT_{t+1}(c') + 1) \geq L_{t+1}(c') \geq L_t(c') \quad (5)$$

Combining (1) and (5) results in a non-negative slackness insertion policy. This completes the proof of the lemma. □

Lemma 2: The *LTF* output scheduling scheme ensures *Lowest Time To Live* (LTTL) scheduling property.

Proof:

Every cell, c, sent from an input $VOQ_{i,j}$ to a crosspoint buffer, $XP_{i,j}$, is inserted either at the tail of $FIFO_j$ or in the $PIFO_j$.

- *Case 1*: cell, c, is inserted at the tail of $FIFO_j$
  - (i). Cell, c, enters the switch at the current time slot.
  - (ii). As, with FIFO OQ, if simultaneous arrivals occur to the same FIFO, tie-breaking[2] is used.
  - (iii). For cell, c', ahead of c in $FIFO_j$, $TTL(c') < TTL(c)$.

Combining (i), (ii) and (iii) implies: cells in $FIFO_j$ are ordered by their TTL and the HoL cell of $FIFO_j$ has the lowest TTL. (6)

- *Case 2*: cell, c, is inserted into $PIFO_j$

  By definition 3, inserted cells in the $PIFO_j$ are ordered by their LTTL. (7)

The LTF scheme compares the $FIFO_j$ HoL cell with $PIFO_j$ HoL cell and moves the cell with Lowest TTL to the front of the OPL. (8)

The LTF serves cells based on the OPL order. (9)

Combining (6), (7), (8) and (9) proves the lemma. □

Having proved the *NNS* insertion policy and the LTTL output scheduling property, we are now going to show, through the next two theorems that the slackness of every cell inside the switch never decreases from time slot to the next.

Theorem1:

The slackness of any cell, c, that does not yet reach its output queue in a VOQ/BCF switch, which employs the *MCAF_OCF* scheduling scheme, increases by at least 1 each scheduling phase.

Proof:

We know that any cell, c, that does not yet reach its output queue can only be either at internal buffer or at an input queue. Therefore we have the two followings cases:

- *Case 1*: c is queued at an internal buffer, $XP_{i,j}$
  - (i). $L(c) = OC(c) - IT(c)$ (def. 8)
  - (ii). Since cell c is queued at the internal buffer, then $IT(c) = 0$ (def. 6)
  - (iii). If c ends the scheduling phase at the internal buffer, we know that cell, c', such that $TTL(c') < TTL(c)$ has been selected for output scheduling. Hence, $OC(c)$ increases by 1 (lemma 2)

Combining (i), (ii), and (iii) yields: $L(c)$ increases by 1. (10)

- *Case 2*: c is queued at an input queue, $VOQ_{i,j}$

  In this case, there are two possibilities: either $VOQ_{i,j}$ is eligible or it is blocked.

  - *Case a*: $VOQ_{i,j}$ is eligible
    - (i). During the input scheduling phase either c is chosen or a cell c' ahead of c in the *IPL* is chosen to be transferred to the internal buffer.
    - (ii). If c is chosen, $IT(c)$ becomes zero, and therefore decreases by at least 1.
    - (iii). If c' is chosen, $IT(c)$ decreases by 1.
    - (iv). $OC(c)$ remains unchanged during input scheduling (def. 7)

  Combining (i), (ii), (iii) and (iv): $L(c)$ increases by 1. (11)

  - *Case b*: $VOQ_{i,j}$ is blocked by an internally queued cell c'
    - (i). Both c and c' belong to the same FIFO $VOQ_{i,j}$. Hence, $TTL(c') < TTL(c)$.
    - (ii). During an output scheduling phase, either c' or c'' (such that $TTL(c'') < TTL(c') < TTL(c)$) is sent to the output. In either cases, $OC(c)$ increases by 1. (lemma 2)
    - (iii). During an input scheduling phase, $IT(c)$ either decreases or remains unchanged.

[2] Tie-breaking method, such as lowest port number first, can be used.

Combining (i), (ii), and (iii) yields: $L(c)$ increases by 1. (12)

The combination of (10), (11) and (12) yields: the slackness of any cell, c, that does not yet reach its output queue, increases at least by 1 during each scheduling phase. Hence, the proof of theorem 1 is complete.

□

Theorem 2:

Consider a VOQ/IBC switch with a speedup of 2 that employs *MCAF_LTF* scheduling policy. For every time slot and for every cell, c, that does not yet reach its output queue, the slackness never decreases.

Proof:

For the VOQ/IBC switch operating at speedup of 2, the time slot is divided into an arrival phase, two scheduling phases (each one consists of input scheduling and output scheduling), and a departure phase.

(i). During an arrival phase: $IT(c)$ can increase by *at most* 1 (in case the new cell is more urgent than c). The possibility that $IT(c)$ increases by *at most* 1 causes $L(c)$ to decrease by 1 (13)

(ii). During a departure phase: $OC(c)$ decreases by exactly 1, since a cell in its output queue left the switch. The decrease of $OC(c)$ by 1 causes $L(c)$ to decrease by *exactly* 1 (14)

Combining (13) and (14) yields: $L(c)$ decreases by *at most* 2. (15)

(iii). From theorem 1, we know that the slackness of any cell, c, that does not yet reach its output queue, increases at least by 1 each scheduling phase. Since we have a speedup of two, every time slot contains two scheduling phases. Hence, every time slot, $L(c)$ increases by *at least* 2. (16)

Summing (15) + (16) yields the slackness, $L(c)$, never decreases. Hence, the proof of theorem 2 is done.

□

Now that we have proved that a VOQ/IBC using *MCAF_LTF* scheduling scheme and a speedup of 2 satisfies the non-negative slackness insertion policy and a non-decreasing slackness from time slot to the next, we are ready to prove our main theorem.

Theorem 3:

A VOQ/IBC switch employing the *MCAF_LTF* scheduling policy with two times speed up can exactly emulate a FIFO_OQ switch.

Proof: (By induction)

Suppose that the VOQ/IBC has emulated a FIFO_OQ switch up until the departure phase of time slot t. We show that any cell, c, such that $TTL(c) = t+1$ reaches its output queue on or before the second scheduling phase of time slot $t+1$ as follows:

- *Case 1*: c is queued at the internal buffers

(i). There are no cells left inside the switch with $TTL < t+1$ (By hypothesis)

(ii). $TTL(c) = t+1$ and the LTF scheme ensures LTTL scheduling (lemma 2)

(i) and (ii) result in c being scheduled during the first output scheduling phase of time slot $t+1$. (17)

- *Case 2*: c is queued at an input $VOQ_{i,j}$

(i). Cell c has the lowest TTL, hence $OC_{t+1}(c) = 0$.

(ii). Cell c was inserted with *NNS* (Lemma 1)

(iii). Since c has the lowest TTL, all cells with lower TTL than c are gone from the system. Thus the internal fabric $XP_{i,j}$ must be available (18)

(i) and (ii) imply $IT_{t+1}(c) = 0$, and thus c must be in the front of the *IPL*. (19)

(18) and (19) result in $VOQ_{i,j}$ being eligible and cell c must be transferred to the internal buffer during the first input scheduling phase. (20)

(18) and (20) result in cell c being chosen by the output scheduler during the first output scheduling phase (21)

- *Case 3*: c already reached its output queue

In this case, we are no longer concerned about c since it reached its output queue before time slot $t+1$ (22)

The combination of (17), (21) and (22) results in cell c reaching its output queue on or before the second scheduling phase of time slot $t+1$. Therefore, theorem 3 is proven and the FIFO OQ emulation exists.

□

The input scheduling scheme MCAF is simple in hardware implementation. No state information (i.e., HoL Cell age, TTL) is needed in order for MCAF to make its decision. The implementation of MCAF can be exactly the same as the input scheduling scheme GBVOQ [11]. The only difference lies in the cell overhead bit that decides whether the cell will be inserted in the tail of the FIFO or in the PIFO queue maintained by the corresponding destination internal buffer.

The output scheme LTF, is much simpler than the OCF scheme used in [11]. The OCF scheme requires a time stamping mechanism to decide which cell has the lowest TTL. This is costly in implementation, and can be even a bottleneck if we consider the very short time during which OCF must decide which cell to send to the output. The LTF, however, consists of maintaining a FIFO queue and a PIFO queue for each output. No processing is required for cells that are inside the FIFO queue. The only cost might be in maintaining the PIFO queue. However, given the nature of the whole scheme MCAF_LTF and the doubled speed at which the fabric runs with respect to the external line speed, we can foresee that most cells will pass through the FIFO queue (currently arriving cells to eligible VOQs) before getting switched to their output. Then the PIFO queue at each output will be

holding a minor number of cells, which means that the costly time stamping mechanism can be reduced to a minimum.

IV. CONCLUSION

This paper sufficiently proves that a one-cell-internally buffered crossbar, VOQ/IBC, switch running twice as fast as the external line rate can exactly emulate a FIFO_OQ switch independent of the switch size. In particular, we proposed a distributed scheduling scheme called MCAF_LTF that met all the requirements for the exact FIFO_OQ emulation with a two times speedup. The complexity of our scheme is much reduced when compared to all previously proposed schemes. Hence, making it of high practical value and easy to implement in real-time for high-speed input traffic.

REFERENCES

[1] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," *Inernetworking: Research and Experience*, Sept. 1990, vol.1, no.1, pp.3-26.

[2] B. Prabhakar, N. McKeown, "On the speedup required for combined input and output queued switching", *Automatica*, 35(12), pp.1909-1920, 1999.

[3] C.-Y. Chang, A. J. Paulraj, and T. Kailath, "A Broadband Packet Switch Architecture with Input and Output Queuing," *Proc. Globecom'94*.

[4] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-switching Networks," *Proceedings of the IEEE*, Oct. 1995, vol.83, no.10, pp.1374-96.

[5] I. Iliadis, and W. Denzel, "Performance of Packet Switches with Input and Output Queueing," *Proc. ICC'90*, 1990.

[6] L. Mhamdi and M. Hamdi, "Practical Scheduling Algorithms for High-Performance Packet Switches", *IEEE ICC'03*, Vol. 3, pp. 1659-1663. Alaska, May 2003.

[7] M. Karol, M. Hluchyj, "Queuing in High-performance Packet-switching," *IEEE J. Selected Area Communications*, Vol. 6, pp. 1587-1597, December 1988.

[8] M. Nabeshima, "Performance Evaluation of Combined Input-and Crosspoint-Queued Switch," *IEICE Trans. Commun.*, Vol. E83-B, No.3 March 2000.

[9] S. chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching Output Queuing with a Combined Input Output Queued Switch," *Computer Systems Technical Report CSL-TR_98-758*, Mar. 1998.

[10] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High-speed Switch Scheduling for Local Area Networks," *ACM Trans. Comput. Syst*., pp. 319-352, Nov. 1993.

[11] B. Magill, C. Rohrs, R. Stevenson, "Output Queued Switch Emulation by a Buffered Crossbar Fabric", *Conference on Communication, Control, and Computing*, Allerton, Oct. 2002.