Improved Competitive Performance Bounds for CIOQ Switches [*]

Alex Kesselman
Google, Inc.
Email: alx@google.com

Kirill Kogan
Cisco Systems, South Netanya, Israel
and
Communication Systems Engineering Dept., Ben Gurion University, Beer-Sheva, Israel
Email: kkogan@cisco.com

Michael Segal
Communication Systems Engineering Dept., Ben Gurion University, Beer-Sheva, Israel
Email: segal@cse.bgu.ac.il

December 1, 2010

Abstract

Combined Input and Output Queued (CIOQ) architectures with a moderate fabric *speedup* $S > 1$ have come to play a major role in the design of high performance switches. In this paper we study CIOQ switches with First-In-First-Out (FIFO) buffers providing Quality of Service (QoS) guarantees. The goal of the switch policy is to maximize the total value of packets sent out of the switch. We analyze the performance of a switch policy by means of competitive analysis, where a uniform worst-case performance guarantee is provided for all traffic patterns. Azar and Richter [8] proposed the β-PG algorithm (Preemptive Greedy with a preemption factor of β) that is 8-competitive for an arbitrary speedup value when $\beta = 3$. We improve upon their result by showing that this algorithm achieves a competitive ratio of 7.5 and 7.47 for $\beta = 3$ and $\beta = 2.8$, respectively. Basically, we demonstrate that β-PG is at most $\frac{\beta^2+2\beta}{\beta-1}$ and at least $\frac{\beta^2}{\beta-1}$-competitive.

[*]A preliminary version of this work appeared in Proceedings of ESA 2008

1 Introduction

The main tasks of a router are to receive packets from the input ports, to find their destination ports using a routing table, to transfer the packets to their corresponding output ports, and finally to transmit them on the input links.

If a burst of packets destined to the same output port arrives, it is impossible to transmit all the packets immediately, and some of them must be buffered inside the switch (or dropped).

A critical aspect of the switch architecture is the placement of buffers. In the output queuing (OQ) architecture, packets arriving from the input lines immediately cross the switching fabric, and join a queue at the switch output port. Thus, the OQ architecture allows one to maximize the throughput, and permits the accurate control of packet latency. However, in order to avoid contention, the internal speed of an OQ switch must be equal to the sum of all the input line rates. The recent developments in networking technology has produced a dramatic growth in line rates, and have made the internal speedup requirements of OQ switches difficult to meet. This has in turn generated great interest in the input queuing (IQ) switch architecture, where packets arriving from the input lines are queued at the input ports. The packets are then extracted from the input queues to cross the switching fabric and to be forwarded to the output ports.

It is well-known that the IQ architecture can lead to low throughput, and it does not allow the control of latency through the switch. The main problem of the IQ architecture is head-of-line (HOL) blocking, which occurs when packets at the head of various input queues contend on a specific output port of the switch. To alleviate the problem of HOL blocking, one can maintain at each input a separate queue for each output. This technique is known as virtual output queuing (VOQ).

Another method to get the delay guarantees of an IQ switch closer to that of an OQ switch is to increase the *speedup* S of the switching fabric. A switch is said to have a speedup S, if the switching fabric runs S times faster than each of the input or the output lines. Hence, an OQ switch has a speedup of N (where N is the number of input/output lines), 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. This architecture is called Combined Input and Output Queued (CIOQ) switch. CIOQ switches with a moderate speedup S have received increasing attention in the literature over the last decade, see e.g. [11, 12].

Given a CIOQ switch, the switch policy consists of a buffer management policy controlling the usage of the buffers, a scheduling policy controlling the switch fabric, and a transmission policy controlling the output buffers. The buffer management policy decides for any packet that arrives to a buffer, whether to accept or reject it (in the latter case the packet is lost). If preemption is allowed, the buffer management policy can drop from the buffer a packet that was previously accepted to make room for a new packet. The scheduling policy is responsible for selecting the packets to be transferred from the input queues to the output queues. This has to be done in a way that prevents contention, i.e., at any given time at most one packet can be removed from any CIOQ input port, and at most one packet can be added to any CIOQ output port. The transmission policy selects the packet to be sent on the output link.

In the present paper we consider CIOQ switches with First-In-First-Out (FIFO) buffers. We study the case of traffic with packets of variable values where the value of a packet represents its priority. This corresponds to the DiffServ (Differentiated Services) model [9]. The goal of the switch policy is to maximize the total value of the packets sent out of the switch.

Since Internet traffic is difficult to model and it does not seem to follow the more traditional Poisson arrival model [24, 26], we do not assume any specific traffic model. We rather analyze our policies against arbitrary traffic and provide a uniform worst-case throughput guarantee for all traffic patterns, using competitive analysis [25, 10]. In competitive analysis, the online policy is compared to the optimal offline policy OPT, which knows the entire input sequence in advance. The competitive ratio of a policy A is the maximum, over all sequences of packet arrivals σ, of the ratio between the value of packets sent by OPT out of σ, and the value of packets sent by A out of σ.

Our results. We consider a CIOQ switch with FIFO buffers of limited capacity. We assume that each packet has an intrinsic value designating its priority. We analyze the β-Preemtive Greedy policy (β-PG) that was shown to be 8-competitive by Azar and Richter [8] for $\beta = 3$. We improve upon their result by establishing that β-PG is 7.47-competitve for $\beta = 2.8$. Basically, we demonstrate that the β-PG policy achieves a competitive ratio of $\frac{\beta^2+2\beta}{\beta-1}$ (for $\beta > 1$). In particular, our result implies that for the value $\beta = 3$ used by Azar and Richter [8] the competitive ratio of β-PG is at most 7.5. Our proof technique is completely different from the one presented in [8] and does not make use of dummy packets. In addition, we show a first lower bound of $\frac{\beta^2}{\beta-1}$ on the performance of β-PG for sufficiently large S. Thus, β-PG is at least 4.5 and 4.36-competitive for $\beta = 3$ and $\beta = 2.8$, respectively.

Related work. A large number of scheduling algorithms have been proposed in the literature for the IQ switch architecture: these include PIM [4], iSLIP [23], Batch [14] to name a few. These algorithms achieve high throughput when the traffic pattern is admissible (uniform), i.e. the aggregate arrival rate to an input or output port is less than 1. However, their performance typically degrades when traffic is non-uniform [22]. Most of the above works on the control of IQ and CIOQ switches assume that there is always enough buffer space to store the packets when and where needed. Thus, all packets arriving to the switch eventually cross it. However, contrary to this setting it is observed empirically in the Internet that packets are routinely dropped in switches. In the present work we address the question of the design of control policies for switches, when buffer space is limited, and thus packet drop may occur.

The problem of throughput maximization in the context of a single buffer has been explored extensively in recent years (see [16] for a good survey). Englert and Westermann [15] presented almost matching lower and upper bounds on the competitive ratio of β-preemptive greedy policy (β is the preemption factor) in the context of a single FIFO buffer (lower bound is 1.707 and upper bound is 1.732).

Competitive analysis of preemptive and non-preemptive scheduling policies for shared memory OQ switches was given by Hahne et al. [17] and Kesselman and Mansour [19], respectively. Aiello et al. [1] considered the throughput of various protocols in a network of OQ switches with limited buffer space. Kesselman et al. [18] studied the throughput of local buffer management policies in a system of merge buffers.

Azar and Richter [7] presented a 4-competitive algorithm for a weighted multi-queue switch problem with FIFO buffers. An improved 3-competitive algorithm was later given by Azar and Richter [6]. Albers and Schmidt [3] proposed a deterministic 1.89-competitive algorithm for the case of unit-value packets. Azar and Litichevskey [5] derived a 1.58-competitive algorithm for this special case with large buffers. Albers and Jacobs [2] gave an experimental study of new and known online packet buffering algorithms.

Kesselman and Rosén [20] studied CIOQ switches with FIFO buffers (a generalization of the multi-queue switch problem). For the case of packets with unit values, they presented a switch policy that is 3-competitive for any speedup. For the case of packets with variable values, they proposed two switch policies achieving competitive ratios of $4S$ and $8\min(n, 2\log\alpha)$, where n is the number of distinct packet values and α is the ratio between the largest and the smallest values. Azar and Richter [8] obtained an 8-competitive algorithm for CIOQ switches with FIFO buffers, which is the first algorithm that achieves a constant competitive ratio for the general case of arbitrary speedup and packet values. Kesselman and Rosén [21] considered the case of CIOQ switches with Priority Queuing (PQ) buffers and proposed a policy that is 6-competitive for any speedup.

Organization. The rest of the paper is organized as follows. The model description appears in Section 2. The switch policy is presented and analyzed in Section 3 and Section 4, respectively. We mention some conclusions in Section 5.

2 Model Description

In this section we describe our model. We consider an $N \times N$ CIOQ switch with N input ports, N output ports, and a speedup S (see Figure 1). Packets, of equal size, arrive at input ports. Each packet is labeled with the output port on which it has to leave the switch and is placed in the input queue corresponding to its output port. When a packet crosses the switch fabric, it is placed in the output queue and resides there until it is sent on the output link. For a packet p, we denote by $V(p)$ its value.

Each input i maintains for each output j a separate queue $VOQ_{i,j}$ of capacity $BI_{i,j}$ (Virtual Output Queuing) and each output j maintains a queue OQ_j of capacity BO_j. All queues in the switch are FIFO, namely, packets leave the queues in the order of their arrivals.

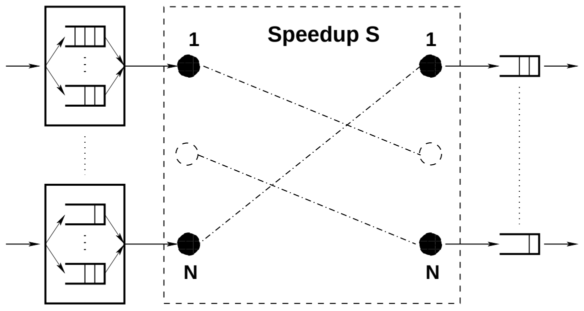


Figure 1: An example of a CIOQ switch.

We divide time into discrete steps. During each time step one or more packets can arrive on each input port, and one packet can be forwarded from each output port. We divide each time step into three phases. The first phase is the *transmission* phase during which a packet from each non-empty output queue can be sent on the output link. The second phase is the *arrival* phase. In the arrival phase one or more packets arrive at each input port. The third phase is the *scheduling* phase when packets are transferred from the input buffers to the output buffers. In a switch with a speedup of S, up to S packets can be removed from any input and up to S packets can be added to each output. This is done in (up to) S cycles, where in each cycle we compute a matching between the inputs and the outputs and transfer the packets accordingly. Specifically, an edge (i, j) in the matching between input i and output j corresponds to transmitting a packet from virtual output queue $VOQ_{i,j}$ to output queue OQ_j. We denote the s-th scheduling cycle ($1 \leq s \leq S$) at time step t by t_s.[1] Suppose that the switch is managed by a policy A. By $VOQ^A_{i,j}$ we denote $VOQ_{i,j}$ as managed by A, and by OQ^A_j we denote OQ_j as managed by A. We represent the state of a switch as an $N \times N$ bipartite multi-graph with the set of nodes $V_{N_I} \cup V_{N_O}$ representing the input and the output ports. Each packet p in $VOQ_{i,j}$ creates an edge (i, j) whose weight equals $V(p)$.

The switch policy is composed of three main components, namely, a transmission policy, a buffer management policy and a scheduling policy.

Transmission Policy. The transmission policy at each time step decides which packet is transmitted out of each output buffer.

Buffer Management Policy. The buffer management policy controls the admission of packets into the buffers. More specifically, when a packet arrives to a buffer, the buffer management policy decides whether to *accept* or *reject* it. An accepted packet can be later *preempted* (dropped).

Scheduling Policy. At every scheduling cycle, the scheduling policy first decides which packets can be scheduled. Then it specifies, according to computed matching, which packets are transferred from the inputs

[1]With slight abuse of notation we say that $t_0 = (t-1)_S$, $t_{S+1} = (t+1)_1$ and $t = t_1$.

to the outputs. The aim of the switch policy is that of maximizing the total value of packets sent from the output ports. Let σ be a sequence of packets arriving at the inputs of the switch. Let $V^A(\sigma)$ and $V^{OPT}(\sigma)$ be the total value of packets transmitted out of the sequence σ, by an online switch policy A and an optimal offline policy OPT, respectively. The competitive ratio of a switch policy is defined as follows.

Definition 2.1 *An online switch policy A is said to be c-competitive if for every input sequence of packets σ, $V^{OPT}(\sigma) \leq c \cdot V^A(\sigma) + d$, where d is a constant independent of σ.*

3 β-Preemptive Greedy Switch Policy

In this section we describe the switch policy that was first introduced by Azar and Richter [8]. We treat each virtual input or output queue as a separate buffer with *independent* buffer management policy. The β-preemptive greedy (β-PG) policy appearing in Figure 2 uses a natural preemptive greedy buffer management policy and a scheduling policy based on maximum weight matching. The value of the parameter β will be determined later. Observe that a packet p is not scheduled to an output buffer if it will be dropped or if it will preempt another packet p' such that $V(p') > V(p)/\beta$. In what follows when we say "first packet", or "last packet", we mean the first or last packet according to FIFO order in the relevant set.

- **Transmission:** Transmit the first packet from each non-empty output queue.
- **Buffer Management of Input and Output Buffers (greedy):** Accept an arriving packet p if there is free space in the buffer. Drop p if the buffer is full and $V(p)$ is less than the minimal value among the packets currently in the buffer. Otherwise, drop from the buffer a packet p' with the minimal value and accept p (we say that p *preempts* p').
- **Scheduling:** For each buffer $VOQ_{i,j}$, consider the first packet in $VOQ_{i,j}$ and denote its value by w. Mark this packet as *eligible*, if OQ_j is not full or if the minimal value among the packets in OQ_j is at most w/β.
  Compute a *maximum weight* matching.

Figure 2: The β-Preemptive Greedy Switch Policy (β-PG).

4 Analysis

We will show that β-PG achieves a competitive ratio of $\frac{\beta^2+2\beta}{\beta-1}$ for any speedup S assuming that $\beta > 1$. We also derive a lower bound of $\frac{\beta^2-\beta+1}{\beta-1}$ on the performance of β-PG for sufficiently large S. Our analysis proceeds along the lines of the work in [21], which studies Priority Queuing (PQ) buffers. However, extension from PQ to FIFO buffers is technically challenging.

In what follows we fix an input sequence σ. To prove the competitive ratio of β-PG we will assign value to the packets sent by β-PG so that no packet is assigned more than $\frac{\beta^2+2\beta}{\beta-1}$ times its value and then show that the value assigned is indeed at least $V^{OPT}(\sigma)$.

For the analysis, we consider OPT that never preempts packets. Obviously, such OPT exists since it knows the whole input a priori and can accept only the packets that will be transmitted.

The assignment routine presented in Figure 3 specifies how to assign value to the packets sent by β-PG (we will show that it is feasible).

- **Step** 1**:** Assign to each packet scheduled by β-PG at time t_s its own value.
- Let p' be the packet scheduled by OPT at time t_s from $VOQ_{i,j}^{OPT}$, if any. Let p be the first packet in $VOQ_{i,j}^{PG}$ at time t_s if any or a dummy packet with zero value otherwise.
- **Step** 2**:** If p is *not eligible* for transmission and either (i) $V(p') \leq V(p)$ or (ii) $V(p') > V(p)$, p' is present in $VOQ_{i,j}^{PG}$ and p' has been previously assigned some value by Step 4, then proceed as follows: Let p'' be the packet that will be sent from OQ_j^{PG} at the same time at which OPT will send p' from OQ_j^{OPT} (we will later show that p'' exists and its value is at least $V(p)/\beta$). If (i), assign the value of p' to p''. If (ii), re-assign to p'' the value that was previously assigned to p' by Step 4.
- **Step** 3**:** If $V(p') > V(p)$ then proceed as follows:
  - **Sub-Step** 3.1**:** If p' was scheduled by β-PG prior to time t_s, then assign the value of $V(p')$ to p'.
  - **Sub-Step** 3.2**:** Else if p' is not present in $VOQ_{i,j}^{PG}$, consider the set of packets with value at least $V(p')$ that are scheduled by β-PG from $VOQ_{i,j}^{PG}$ prior to time t_s. Assign the value of $V(p')$ to a packet in this set that is not in $VOQ_{i,j}^{OPT}$ at the beginning of t_s, and has not previously been assigned a value by either Sub-Step 3.1 or Sub-Step 3.2 (we will later show that such a packet exists).
  - **Sub-Step** 3.3**:** Else (p' is present in $VOQ_{i,j}^{PG}$), remove the value assigned to p' by Step 4 and assign the value of $V(p')$ to p' (we will later show that the removed value is re-assigned by Step 1).
- **Step** 4**:** If a packet q preempts a packet q' at an *input* or *output* queue of β-PG, re-assign to q the value that has been previously assigned to q'.

Figure 3: Assignment Routine – executed at the end of scheduling cycle t_s.

Observe that the assignment routine assigns some value only to packets that are scheduled out of the input queues. Furthermore, if a packet is preempted at an output queue then the total value assigned to it is re-assigned to the packet that preempts it. The following observation follows from the finiteness of the input sequence.

Observation 4.1 *When the assignment routine finishes, only packets that are eventually sent by β-PG are assigned some value.*

The following claim bounds the total value that can be assigned to a β-PG packet before it leaves a virtual output queue.

Claim 4.2 *The weight assigned to a β-PG packet before it leaves a virtual output queue is at most its own value.*

Proof. Initially, a β-PG packet q' in a virtual output queue can be assigned its own value by Sub-Step 3.3. If q' is later preempted by a packet q, then q is re-assigned the value that was assigned to q' by Step 4. Obviously, q is assigned at most its own value as $V(q) > V(q')$. Note that if q will be assigned its own value by Sub-Step 3.3, then the value assigned to q by Step 4 is either re-assigned by the case (ii) of Step 2 or removed by Step 3.3 and re-assigned by Step 1. The claim follows. □

In the next claim we show that when the case (ii) of Step 2 re-assigns the value assigned to a β-PG packet located at a virtual output queue, the value of the first packet in this queue is at least the value that needs to be re-assigned.

Claim 4.3 *If the case (ii) of Step 2 or Step 3.3 apply and we re-assign the value assigned to the packet p' in $VOQ_{i,j}^{PG}$ by Step 4, then we have that $V(p)$ is at least the value to be re-assigned, where p is the first packet in $VOQ_{i,j}^{PG}$.*

Proof. Consider the time step at which p' has arrived and was accepted by both β-PG and OPT. If $p' \in VOQ_{i,j}^{PG}$ has been assigned some value, p' should have preempted another packet q' in $VOQ_{i,j}^{PG}$ and was re-assigned the value that had been previously assigned to q' by Step 4. Since β-PG always preempts the least valuable packet from a queue, all packets in $VOQ_{i,j}^{PG}$ preceding p', and p in particular, must have a value of at least $V(q')$. Moreover, according to Claim 4.2, q' had been assigned at most its own value. That establishes the claim. □

We demostrate that the value assigned by Step 1 of the assighnment routine covers the value of scheduled OPT packets that are not dealt with by Step 2 or Step 3 as well as the value re-assigned by Sub-Step 3.3.

Claim 4.4 *The value assigned by Step 1 of the assignment routine is at least as large as the total value of packets scheduled by OPT whose value is not assigned by Step 2 or Step 3 and the value re-assigned by Sub-Step 3.3.*

Proof. Firstly note that OPT packets packets whose value is not assigned by Step 2 or Step 3 are scheduled from queues where β-PG had an eligible packet with value larger than that of OPT's. Secondly observe that Sub-Step 3.3 of the assignment routine applies only if the packet residing in the corresponding β-PG queue, $VOQ_{i,j}^{PG}$, is eligible for transmission, for otherwise its value would have been re-assigned by Step 2 (case ii). According to Claim 4.3, the value that needs to be re-assigned by Sub-Step 3.3 is at most the value of the packet at the head of $VOQ_{i,j}^{PG}$. The claim follows by the maximality of matching computed by β-PG. □

Now we show that the assignment routine is feasible and establish an upper bound on the value assigned to a single packet.

Lemma 4.5 *The assignment routine is feasible.*

Proof. First we show that the assignment as defined is feasible. Step 1, Sub-Step 3.1, Sub-Step 3.3 and Step 4 are clearly feasible. We therefore consider Steps 2 and 3.2.

First we consider Step 2. Let p be the first packet in $VOQ_{i,j}^{PG}$. Assume that p is not eligible for transmission. Then, by the definition of β-PG, the minimal value among the packets in OQ_j is at least $V(p)/\beta$ and OQ_j is full. Thus, during the following BO_j time steps, β-PG will send packets with value of at least $V(p)/\beta$ out of OQ_j. The packet p' scheduled by OPT from $VOQ_{i,j}^{OPT}$ at time t_s will be sent from OQ_j^{OPT} in one of these time steps. Since p was not eligible for transmission we have that the packet as specified in Step 2 indeed exists, and its value is at least $V(p)/\beta$.

Next we consider Sub-Step 3.2. First note that if this case applies, then the packet p' (scheduled by OPT from $VOQ_{i,j}^{OPT}$ at time t_s) is dropped by β-PG from $VOQ_{i,j}^{PG}$ at some time $t_q < t_s$.

Let $t_r \geq t_q$ be the last time before t_s at which a packet of value at least $V(p')$ is dropped from $VOQ_{i,j}^{PG}$. Since the greedy buffer management policy is applied to $VOQ_{i,j}^{PG}$, $VOQ_{i,j}^{PG}$ contains $BI_{i,j}$ packets with value of at least $V(p')$ at this time. Let P be the set of these packets. Note that $p' \notin P$ because it has already been dropped by β-PG at this time. We have that in $[t_r, t_s)$, β-PG has actually scheduled all packets from

P, since in $[t_r, t_s)$ no packet of value at least $V(p')$ has been dropped, and at time t_s the packet at the head of $VOQ_{i,j}^{PG}$ has a value less than $V(p')$.We show that at least one packet from P is *available* for assignment at time t_s, i.e., it has not been assigned any value by Step 3 and is not currently present in $VOQ_{i,j}^{OPT}$. Let x be the number of packets from P that are currently present in $VOQ_{i,j}^{OPT}$. By the construction, these x packets are unavailable. From the rest of the packets in P, a packet is considered available unless it has been already assigned a value by Step 3. Observe that a packet from P can be assigned a value by Step 3 only during $[t_r, t_s)$ (when it is scheduled).

We now argue that OPT has scheduled at most $BI_{i,j} - 1 - x$ packets out of $VOQ_{i,j}$ in $[t_r, t_s)$, and thus P contains at least one available packet. To see this observe that the x packets from P that are present in $VOQ_{i,j}^{OPT}$ at time t_s, were already present in $VOQ_{i,j}^{OPT}$ at time t_r. The same applies to packet p' (recall that $p' \notin P$). Since OPT maintains FIFO order, all the packets that OPT scheduled out of $VOQ_{i,j}^{OPT}$ in $[t_r, t_s)$ were also present in $VOQ_{i,j}^{OPT}$ at time t_r. Therefore, the number of such packets is at most $BI_{i,j} - 1 - x$ (recall that the capacity of $VOQ_{i,j}$ is $BI_{i,j}$). We obtain that at least one packet from P is available for assignment at Sub-Step 3.2 since $|P| = BI_{i,j}$, x packets are unavailable because they are present in $VOQ_{i,j}^{OPT}$ and at most $BI_{i,j} - 1 - x$ packets are unavailable because they have been already assigned a value by Step 3. □

Lemma 4.6 *No packet is assigned more than $\frac{\beta^2+2\beta}{\beta-1}$ times its own value.*

Proof. Consider a packet p sent by β-PG. Claim 4.2 implies that p can be assigned at most once its own value before it leaves the virtual output queue. In addition, p is assigned its own value by Step 1.

By the specification of Sub-Step 3.2, this step does not assign any value to p if it is assigned a value by either Sub-Step 3.1 or Sub-Step 3.2. We also show that Sub-Step 3.1 does not assign any value to p if it is assigned a value by either Sub-Step 3.1 or Sub-Step 3.2. That is due to the fact that by the specification of Sub-Step 3.2, if p is assigned a value by Sub-Step 3.2 at time t_s, then p is not in the input buffer of OPT at this time. Therefore, Sub-Step 3.1 cannot be later applied to it. We obtain that p can be assigned at most its own value by Sub-Step 3.1 and Sub-Step 3.2 after it leaves the virtual output queue.

Now let us consider Step 2. Observe that cases (i) and (ii) are mutually exclusive. Furthermore, if case (ii) apples, then by Claim 4.3 the value of the first packet in the β-PG queue is at least the value that needs to be re-assigned. We obtain that p can be assigned at most β times its own value by Step 2 of the assignment routine.

Finally, we bound the value assigned to a packet by Step 4 in the output queue. Note that this assignment is done only to packets that are actually transmitted out of the switch (i.e. they are not preempted). In addition, p can preempt another packet p' such that $V(p') \le V(p)/\beta$. We say that p *transitively* preempts a packet p'' if either p directly preempts p'' or p preempts a packet p' that transitively preempts p''. Observe that any preempted packet in an output queue can be assigned at most three times its own value by Step 1, Step 3 and Step 4 due to preemption in the virtual output queue. Hence, the total value that can be assigned to p by Step 4 due to transitively preempted packets in the output queue is bounded by $\frac{3}{\beta-1}$ times its own value.

We have that in total no packet is assigned more than $3 + \beta + \frac{3}{\beta-1} = \frac{\beta^2+2\beta}{\beta-1}$ times its own value. □

Let $W^{OPT}(\sigma, t_s)$ be the total value of packets *scheduled* out of the virtual output queues of OPT by time t_s and let $M^{PG}(\sigma, t_s)$ be the total value assigned to packets in β-PG by time t_s, on input sequence σ. We show that the value gained by OPT is bounded by the value assigned by the assignment routine.

Lemma 4.7 *For any time t_s the following holds: $W^{OPT}(\sigma, t_s) \le M^{PG}(\sigma, t_s)$.*

Proof. By $X_{i,j}^{A}(t_s)$ we denote the binary variable that indicates whether an algorithm A has scheduled a packet from input i to output j in scheduling cycle t_s ($X_{i,j}^{A}(t_s) = 1$ if some packet has been scheduled from input i to output j and $X^A_{i,j}(t_s) = 0$ otherwise). By $P^A_{i,j}(t_s)$ we denote the packet itself in case $X^A_{i,j}(t_s) = 1$, or a dummy packet with zero value otherwise. Firstly, note that according to Claim 4.4, the value assigned by Step 1 covers OPT packets whose value is not assigned by Step 2 or Step 3 and the value re-assigned by Sub-Step 3.3. The proof proceeds by induction on time. The lemma trivially holds for time zero. Now assume that the lemma holds at time t_{s-1} and let us show that it also holds at time t_s. First we define two indicator variables.

$$G_{i,j}(t_s) = \begin{cases} 1 : & \text{If the value of the first packet in } VOQ^{PG}_{i,j} \text{ at time } t_s \text{ is at least } V(P^{OPT}_{i,j}(t_s)), \\ 0 : & \text{Otherwise;} \end{cases}$$

$$E_{i,j}(t_s) = \begin{cases} 1 : & \text{If the first packet from } VOQ^{PG}_{i,j} \text{ is eligible at time } t_s, \\ 0 : & \text{Otherwise.} \end{cases}$$

We aim to show that $\Delta W^{OPT} = W^{OPT}(\sigma, t_s) - W^{OPT}(\sigma, t_{s-1})$ is bounded by $\Delta M^{PG} = M^{PG}(\sigma, t_s) - M^{PG}(\sigma, t_{s-1})$. We have that,

$$\begin{aligned} \Delta W^{OPT} &= \sum_{i=1}^{N}\sum_{j=1}^{N} X^{OPT}_{i,j}(t_s)V(P^{OPT}_{i,j}(t_s)) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} G_{i,j}(t_s)E_{i,j}(t_s)X^{OPT}_{i,j}(t_s)V(P^{OPT}_{i,j}(t_s)) \\ &+ \sum_{i=1}^{N}\sum_{j=1}^{N} G_{i,j}(t_s)(1 - E_{i,j}(t_s))X^{OPT}_{i,j}(t_s)V(P^{OPT}_{i,j}(t_s)) \\ &+ \sum_{i=1}^{N}\sum_{j=1}^{N}(1 - G_{i,j}(t_s))X^{OPT}_{i,j}(t_s)V(P^{OPT}_{i,j}(t_s)). \end{aligned}$$

We examine each of these terms separately. If $G_{i,j}(t_s)E_{i,j}(t_s)X^{OPT}_{i,j}(t_s) = 1$, then $VOQ^{PG}_{i,j}$ contains an eligible packet with value greater than or equal to that of the packet scheduled by OPT from $VOQ_{i,j}$ at time t_s. Note that β-PG computes a maximum weight matching considering eligible packets and the total value of this matching is at least as large as the total value of the packets scheduled by OPT out of the corresponding input buffers. Thus, we obtain that

$$\begin{aligned} &\sum_{i=1}^{N}\sum_{j=1}^{N} G_{i,j}(t_s)E_{i,j}(t_s)X^{OPT}_{i,j}(t_s)V(P^{OPT}_{i,j}(t_s)) \\ &\qquad \leq \sum_{i=1}^{N}\sum_{j=1}^{N} X^{PG}_{i,j}(t_s)V(P^{PG}_{i,j}(t_s)). \end{aligned}$$

Note that this value is assigned by Step 1 of the assignment routine.

Now consider the second and the third terms. By the specification of the assignment routine, the value of

$$\sum_{i=1}^{N}\sum_{j=1}^{N} G_{i,j}(t_s)(1 - E_{i,j}(t_s))(t_s)X^{OPT}_{i,j}(t_s)V(P^{OPT}_{i,j}(t_s))$$

and

$$\sum_{i=1}^{N}\sum_{j=1}^{N}(1 - G_{i,j}(t_s))X^{OPT}_{i,j}(t_s)V(P^{OPT}_{i,j}(t_s))$$

is assigned by Step $2(i)$ and Step 3, respectively. Hence, we obtain that $\Delta W^{OPT} \leq \Delta M^{PG}$. The lemma now follows by the inductive hypothesis. $\square$

At this point we are ready to prove the main theorem.

Theorem 4.8 *The competitive ratio of the β-PG policy is at most $\frac{\beta^2+2\beta}{\beta-1}$ for any speedup.*

Proof. Suppose that OPT sends the last packet in σ out of an output buffer at time t^*. By Lemma 4.7,

$$W^{OPT}(\sigma, t^*) \leq M^{PG}(\sigma, t^*).$$

Lemma 4.6 and Observation 4.1 imply that

$$M^{PG}(\sigma, t^*) \leq \frac{\beta^2 + 2\beta}{\beta - 1} V^{PG}(\sigma).$$

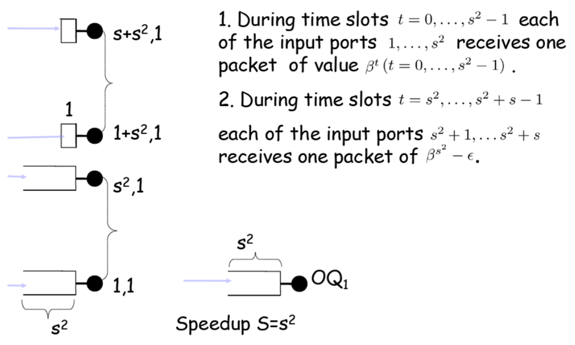
It follows that

$$V^{OPT}(\sigma) \leq \frac{\beta^2 + 2\beta}{\beta - 1} V^{PG}(\sigma),$$

since $W^{OPT}(\sigma, t^*) = V^{OPT}(\sigma)$ (recall that by our assumption OPT does not preempt packets). $\square$

Corollary 4.9 *The competitive ratio of the 2.8-PG policy is at most 7.47 for any speedup.*

Finally, we establish a lower bound on the performance of β-PG.



1. During time slots $t = 0, \ldots, s^2 - 1$ each of the input ports $1, \ldots, s^2$ receives one packet of value β^t $(t = 0, \ldots, s^2 - 1)$.

2. During time slots $t = s^2, \ldots, s^2 + s - 1$ each of the input ports $s^2 + 1, \ldots s^2 + s$ receives one packet of $\beta^{s^2} - \epsilon$.

Figure 4: Scenario for Lower Bound.

Theorem 4.10 *The β-PG algorithm is at least $\frac{\beta^2}{\beta-1}$-competitive for sufficiently large value of speedup S and $\beta > 1$.*

Proof. Consider the following scenario (see Figure 4). All packet arrivals are destined to output port 1 with queue OQ_1 of capacity s^2. The capacity of virtual output queues $VOQ_{i,1}$ for $1 \leq i \leq s^2$ is s^2 and the capacity of virtual output queues $VOQ_{j,1}$ for $s^2 + 1 \leq j \leq s^2 + s$ is one. The value of speedup S is s^2.

During the first phase of arrivals at time slots $t = 0, \ldots, s^2 - 1$ each of the input ports $1, \ldots, s^2$ receives one packet of value β^t $(t = 0, ..., s^2 - 1)$. Later, during the second phase of arrivals at the next time slots $t = s^2, \ldots, s^2+s-1$ each of the input ports $s^2+1, \ldots, s^2+s$ receives one packet of value $\beta^{s^2} - \epsilon$, where $\epsilon > 0$.

During the first phase of arrivals, by the definition of β-PG it will always preempt old packets from OQ_1 and accept there the newly arrived packets since they are more valuable by a factor of β than the previously arrived packets. Moreover, during the first phase of arrivals β-PG sends packets with the total value of $1 + \beta + \ldots + \beta^{s^2-2} = (\beta^{s^2-1} - 1)/(\beta - 1)$. In addition, OQ_1 contains s^2 packets of value β^{s^2-1}.

During the second phase of arrivals the β-PG algorithm will drop all but $2s$ of packets (whose weight is $\beta^{s^2} - \epsilon$) since no packets in OQ_1 will be preempted and by time $t = s^2+s-1$: s of these packets will be buffered in virtual output queues $VOQ_{j,1}$ for $s^2+1 \leq j \leq s^2+s$ and s of these packets will be buffered OQ_1. In addition, β-PG will transmit s^2 packets of weight β^{s^2-1}. So the overall value obtained by β-PG is $V_{PG} = (\beta^{s^2-1} - 1)/(\beta - 1) + s^2\beta^{s^2-1} + 2s(\beta^{s^2} - \epsilon)$.

On the other hand, OPT will first buffer all packets that arrived at input ports $1, \ldots, s^2$ during the first phase (time slots $t = 0, \ldots, s^2-1$) without transferring them to OQ_1. Then OPT will transfer all packets that arrived at input ports $s^2+1, \ldots, s^2+s$ to OQ_1 during the second phase and send them on the output link. Having done with these packets, OPT will deliver all packets buffered at input ports $1, \ldots, s^2$. In this way, the value obtained by OPT is $V_{OPT} = s^2(\beta^{s^2} - 1)/(\beta - 1) + s^2(\beta^{s^2} - \epsilon)$.

For sufficiently large s, which is a function of N, and a constant value of β, V_{PG} is dominated by $s^2\beta^{s^2-1}$. Therefore, V_{OPT}/V_{PG} tends to $\beta/(\beta - 1) + \beta$. □

5 Conclusions

A major problem addressed today in networking research is the need for a fast switch architecture supporting guaranteed QoS. In this paper we study CIOQ switches with FIFO queues. We consider switch policies that maximize the switch throughput for any traffic pattern and use competitive analysis to evaluate their performance. Our main results are an improved upper bound and the first lower bound on the competitive ratio of the switch policy proposed by Azar and Richter [8]. An interesting future research direction is to close the gap between the upper and lower bounds, which still remains rather substantial.

References

[1] W. Aiello, E. Kushilevitz, R. Ostrovsky and A. Rosén, “Dynamic Routing on Networks with Fixed-Size Buffers,” *Proceedings of SODA 2003*, pp. 771-780.

[2] S. Albers and T. Jacobs, ”An experimental study of new and known online packet buffering algorithms”, *Proc. 15th Annual European Symposium on Algorithms*, Springer LNCS 4698, pp. 63-74, 2007.

[3] S. Albers and M. Schmidt, ”On the Performance of Greedy Algorithms in Packet Buffering”, *SIAM Journal on Computing*, Vol. 35, Num. 2, pp. 278-304, 2005.

[4] T. Anderson, S. Owicki, J. Saxe and C. Thacker, “High speed switch scheduling for local area networks”, *ACM Trans. on Computer Systems*, pp. 319-352, Nov. 1993.

[5] Y. Azar and M. Litichevskey, ”Maximizing throughput in multi-queue switches”, *Algorithmica*, Vol. 45, Num. 1, pp. 69-90, 2006.

[6] Y. Azar and Y. Richter, ”The zero-one principle for switching networks”, *Proceedings of STOC 2004*, pp. 64-71, 2004.

[7] Y. Azar and Y. Richter, ”Management of Multi-Queue Switches in QoS Networks”, *Algorithmica*, Vol. 43, Num. 1-2, pp. 81-96, 2005.

[8] Y. Azar and Y. Richter, ”An improved algorithm for CIOQ switches”, *ACM Transactions on Algorithms*, Vol. 2, Num. 2, pp. 282-295, 2006.

[9] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang and W. Weiss, “An Architecture for Differentiated Services,” *Internet RFC 2475*, December 1998.

[10] A. Borodin and R. El-Yaniv, “Online Computation and Competitive Analysis,” *Cambridge University Press*, 1998.

[11] S. T. Chuang, A. Goel, N. McKeown and B. Prabhakar, ”Matching Output Queueing with a Combined Input Output Queued Switch,” *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1030-1039, Dec. 1999.

[12] J. G. Dai and B. Prabhakar, “The Throughput of Data Switches with and without Speedup,” *Proceedings of INFOCOM 2000*, pp. 556-564.

[13] S. Datta and R. K. Sitaraman, “The Performance of Simple Routing Algorithms That Drop Packets,” *Proceedings of SPAA 1997*, pp. 159-169.

[14] S. Dolev and A. Kesselman, ”Bounded Latency Scheduling Scheme for ATM Cells,” *Journal of Computer Networks*, vol. 32(3), pp 325-331, Mar. 2000.

[15] Matthias Englert, Matthias Westermann, ”Lower and Upper Bounds on FIFO Buffer Management in QoS Switches,” *Proceedings of ESA 2006*, pp. 352-363.

[16] L. Epstein and R. Van Stee, “SIGACT news online algorithms,” editor: M. Chrobak, vol. 35, no. 3, pp. 58-66, Sep. 2004.

[17] E. L. Hahne, A. Kesselman and Y. Mansour, “Competitive Buffer Management for Shared-Memory Switches,” *Proceedings of SPAA 2001*, pp. 53-58.

[18] A. Kesselmanm, Z. Lotker, Y. Mansour and B. Patt-Shamir, “Buffer Overflows of Merging Streams,” *Proceedings of ESA 2003*, pp. 349-360.

[19] A. Kesselman and Y. Mansour, “Harmonic Buffer Management Policy for Shared Memory Switches,” *Theoretical Computer Science, Special Issue on Online Algorithms, In Memoriam: Steve Seiden*, vol. 324, pp 161-182, 2004.

[20] A. Kesselman and A. Rosén, “Scheduling Policies for CIOQ Switches,” *Journal of Algorithms*, vol. 60(1), pp. 60-83, 2006.

[21] A. Kesselman and A. Rosén, ”Controlling CIOQ Switches with Priority Queuing and in Multistage Interconnection Networks,” *Journal of Interconnection Networks*, to appear.

[22] M. A. Marsan, A. Bianco, E. Filippi, P. Giaccone, E. Leonardi and F. Neri, “A Comparison of Input Queuing Cell Switch Architectures,” *Proceedings of 3rd International Workshop on Broadband Switching Systems*, Kingston, Canada, June 1999.

[23] N. McKeown, “Scheduling Algorithms for Input-Queued Cell Switches,” *Ph. D. Thesis*, University of California at Berkeley, 1995.

[24] V. Paxson, and S. Floyd, ”Wide Area Traffic: The Failure of Poisson Modeling,” *IEEE/ACM Transactions on Networking*, vol. 3, pp. 226-244, June 1995.

[25] D. Sleator and R. Tarjan, “Amortized Efficiency of List Update and Paging Rules,” *CACM 28*, pp. 202-208, 1985.

[26] A. Veres and M. Boda, “The Chaotic Nature of TCP Congestion Control,” *Proceedings of INFOCOM 2000*, pp. 1715-1723, March 2000.