Scheduling Policies for CIOQ Switches [⋆]

Alex Kesselman

Max Planck Institut fur Informatik, Saarbrucken, Germany.

Adi Rosén

Dept. of Computer Science, Technion, Haifa, Israel.

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. The switch policy that controls such switches must consist of two components. A buffer management policy that controls admission to buffers, and a scheduling policy that is responsible for the transfer of packets from input buffers to output buffers. The goal of the switch policy is to maximize the throughput of the switch. When all packets have a uniform value (or importance), this corresponds to the number of packets sent from the switch. When packets have variable values, this corresponds to the total value of the sent packets.

We derive a number of scheduling policies for CIOQ switches and analyze their throughput using competitive analysis. We thus give for these policies a uniform throughput guarantee, regardless of specific traffic patterns. For the case of packets with uniform values we present a switch policy that is 3-competitive for any speedup. For the case of packets with variable values we propose two switch policies. One achieves a competitive ratio of $4S$, and the other achieves a competitive ratio of $8\min(k, 2\lceil\log\alpha\rceil)$, where k is the number of distinct packet values and α is the ratio between the largest and the smallest value.

Key words: CIOQ Switches, Scheduling Policies, Buffer Management, Competitive Analysis

[⋆] A preliminary version of this paper appeared in the Proceedings of SPAA 2003.
Email addresses: `akessel@mpi-sb.mpg.de` (Alex Kesselman), `adiro@cs.technion.ac.il` (Adi Rosén).

1 Introduction

Switch architectures based on a non-blocking fabric are widely used in today's packet networks. A critical aspect of such an architecture is the placement of switch buffers. In the output queuing (OQ) architecture, packets arriving from 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 at least the sum of all the input line rates. The recent developments in networking technology produced a dramatic growth of line rates and have made the 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 input lines are queued at input ports. The packets are then extracted from input queues to cross the switching fabric and to be forwarded to the output ports.

However, the IQ architecture can lead to low throughput, and it does not allow the control of latency through the switch. For random traffic, uniformly distributed over all outputs, the throughput (i.e. the average number of packets sent in a time step) of an IQ switch has been shown to be limited to approximately 58% of the throughput achieved by an OQ switch [15]. A major problem of the IQ architecture is head-of-line (HOL) blocking, which occurs when packets at the head of the various input queues contend on the same 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). A large number of scheduling algorithms, based on different kinds of matchings between input and outputs ports, have been proposed in the literature for the IQ architecture: these are PIM [4], IRRM [24], iSLIP [22], iOCF [23] RPA [20] and Batch [12], 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 the traffic is non-uniform [19].

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 switch fabric. A switch is said to have a speedup S, if the switch fabric runs S times faster than each of the input or output lines. Hence, an OQ switch has a speedup of N (where N is the number of lines), while an IQ switch has a speedup of one. 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 a combined input and output queued (CIOQ) architecture. CIOQ switches with a moderate speedup S have received considerable attention in the literature

[9,11,13,30]. Prabhakar and McKeown [26] consider the question whether a CIOQ switch can be designed to behave identically to an OQ switch. It is proved that a CIOQ switch with VOQ at the inputs and a speedup of just 4 can be designed to exactly mimic the behavior of an OQ switch, regardless of the nature of the arriving traffic. This result has been later improved by Chuang et al. [10] who show that a speedup of $2 - 1/N$ is necessary and sufficient to exactly emulate an OQ switch.

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, packet drop due to insufficient buffer space never occurs, and all packets arriving to the switch eventually cross the switch. However, contrary to this setting, it is observed empirically that in the Internet 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 aim of the policy is that of maximizing the throughput of the switch, i.e., maximizing the number of packets that cross the switch rather than being dropped due to insufficient buffer space. We provide robust control policies for CIOQ switches. Since Internet traffic is difficult to model and it does not seem to follow the more traditional Poisson arrival model [25,28], we do not assume any specific traffic model. Rather, we analyze our policies against arbitrary traffic and provide a uniform throughput guarantee for all traffic patterns using competitive analysis.

The switch policy controlling a CIOQ switch consists of a buffer management policy and a scheduling policy. The buffer management policy controls the usage of the buffers while the scheduling policy selects packets to be transferred from the inputs to the outputs. We consider the cases of uniform (unit) value packets, as well as variable value packets. The unit value case corresponds to the Best Effort model. In the case of variable value packets, each packet has an intrinsic value, and this corresponds to the DiffServ model [7]. The actual value of a packet may be proportional to the amount of money charged by the Internet Service Provider (ISP) for the corresponding service, or may represent the relative priority of the various packets. The goal of the switch policy is that of maximizing the total value of packets sent.

Our results. First we consider the case of unit value packets. We present a switch policy that is 3-competitive for any speedup and is 2-competitive for a speedup of 1. We note that implementing "back pressure" (i.e. packets are not transferred to output ports whose buffers are full) helps to achieve a constant competitive ratio in this case.

For the case of variable value packets, we give two scheduling policies, which can be combined with an arbitrary buffer management policy for the input buffers. If the buffer management policy is c-competitive for a single buffer, then the first policy is $(2 \cdot c \cdot S)$-competitive while the second policy is $(4 \cdot c \cdot \min(k, 2 \cdot \lceil \log \alpha \rceil))$-competitive for any speedup, where k is the number of distinct packet values and α is the ratio between the highest and the lowest packet value.

To conclude the paper we briefly consider the question of comparing the throughput of a CIOQ switch to the throughput of an OQ switch with FIFO buffers and having a “similar” amount of memory.

Related work. The control of OQ switches with limited buffer space is essentially reduced to the control of a single output buffer. Thus, work on the control of a single finite buffer, in the face of arbitrary traffic, can be regarded as a work on the control of OQ switches (clearly, the question is of interest when packets have variable values). A number of such works considering a single First-In-First-Out (FIFO) buffer appeared in the literature in recent years. If the buffer policy is allowed to drop packets that have been already accepted, it is said to be *preemptive*, otherwise it is said to be *non-preemptive.* Aiello et al. [2] analyze various non-preemptive policies for the special case of two different packet values. Andelman et al. [3] generalize these results to multiple packet values. The question of video smoothing is studied by Mansour et al. [21], where they establish an upper bound of 4 on the competitive ratio of the preemptive greedy policy. This result has been later improved to 2 by Kesselman et al. [16], where they also introduced a new bounded-delay packet model. The work of Kesselman and Mansour [17] studies the case in which all packet values are powers of some constant and analyzes the loss rather than the throughput of a policy. The analysis of a single buffer has been later extended to shared memory OQ switches. Competitive analysis of preemptive and non-preemptive scheduling policies was given by Hahne et al. [14] and Kesselman and Mansour[18], respectively. Aiello et al. [1] study the throughput of various protocols in a network of OQ switches with limited buffer space.

The work mostly related to the present paper is the work of Azar and Richter [6], where they consider a system of multiple FIFO buffers. The main contribution of that work is to show that one can control such a system by a specific scheduling policy, defined in that work, and a separate arbitrary local buffer management policy for any of the buffers. Azar and Richter show that the competitive ratio of the resulting policy is twice that of the local buffer management policy. Using previous results on the management of a single buffer they thus provide a 4-competitive algorithm for this model. The present paper extends the work of Azar and Richter to CIOQ switches, by using their technique of decoupling the buffer management policy from the scheduling policy.

The rest of the paper is organized as follows. The model description appears in

Section 2. Our switch policies are defined in Section 3. The analysis of switch policies for CIOQ switches appears in Section 4. In Section 5 we compare the throughput of a CIOQ switch to that of an OQ switch. We mention some conclusions in Section 6.

2 Model Description

We consider an $N \times N$ CIOQ switch (see Figure 1). Packets, of equal size, arrive at input ports, and each packet is labeled with the output port on which it has to leave the switch. For a packet p, we denote by $V(p)$ its value. We assume that packets can have k distinct values, all in the range $[1..\alpha]$. For simplicity of presentation, we also assume that the sizes of the buffers are divisible by $\min(k, \lceil \log \alpha \rceil)$.

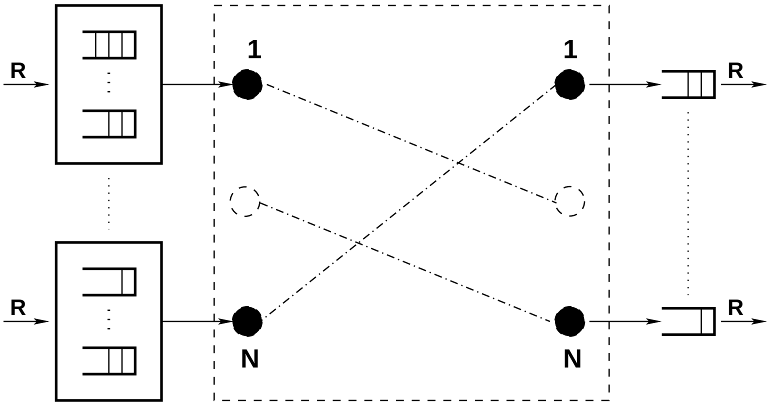

Fig. 1. An example of a CIOQ switch

Unless otherwise stated, we assume that VOQ (Virtual Output Queuing) is implemented at the input ports, and each input i maintains for each output j a separate queue $VOQ_{i,j}$ of capacity $BI_{i,j}$. Each output j maintains a queue OQ_j of capacity BO_j.

We divide time into discrete steps, where a step is the arrival time between two packets at an input. That is, during each time step one packet can arrive at each input port, and one packet can be forwarded on each output port.

We divide each time step into three phases. The first phase is the *transmission* phase during which the first packet from each non-empty output queue is sent on the output link. We denote by $P_j(t)$ the packet that is sent from output j in time step t if any, or a dummy packet with zero value otherwise. We denote by t_T the transmission phase of time step t. The second phase is the *arrival* phase. In the arrival phase at most one packet arrives at each input port. We denote by t_A the arrival phase of time step t. The third phase is the *scheduling* *phase* when packets are transferred from input buffers to 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 at most one packet is removed from each input and at most one packet is added to each output. Thus, during the scheduling phase we compute (up to) S matchings between input and outputs. 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. We introduce the following notation. For any time τ (τ may be a time step t, or a phase t_A or t_T or a scheduling cycle t_s), we denote by $L^A_{i,j}(\tau)$ the length of $VOQ_{i,j}$, by $O^A_j(\tau)$ the length of OQ_j and by $L^A(p, \tau)$ the position of packet p in the queue in which it resides, just before time τ. By $X^A_{i,j}(t_s)$ we denote the variable indicating whether a packet has been scheduled from input i to output j in scheduling cycle t_s (i.e., $X^A_{i,j}(t_s) = 1$ if some packet has been scheduled from input i to output j and $X^A_{i,j}(t_s) = 0$ otherwise).

The *state* of the switch just before a scheduling cycle begins is described by an $N \times N$ bipartite multi-graph. The set of nodes V_{N_I,N_O} represents the input and the output ports and each packet p in $VOQ_{i,j}$ induces an edge (i, j) whose weight equals the value of p, $V(p)$. We denote by $E^A(t_s)$ the set of packets in the input buffer of A at the very beginning of scheduling cycle t_s. We also denote by $G^A(t_s) = (V_{N_I,N_O}, E^A(t_s))$ the corresponding bipartite graph.

We usually assume that FIFO order is maintained, i.e., packets must leave a virtual output buffer, or an output buffer, in the order of their arrivals. So, only the first packet (in the FIFO order) from each queue can participate in the matching. We also consider for our constructions some switch policies in a relaxed, non-FIFO, model in which packets may leave a buffer not necessarily according to FIFO order. However, these policies will be used only as a tool for the analysis and as building blocks for actual policies.

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

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 *drop* it. An accepted packet can be later *preempted* if the preemption is allowed. Separate buffer management policies may control different buffers. However, in all our constructions we use the same policy for all input buffer and the same policy for all output buffers.

Scheduling Policy. The scheduling policy at every scheduling cycle first

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

decides which packets are eligible for being transferred from the inputs to the outputs. Then it specifies which of those packets are actually transferred. This is done by computing a matching in a bipartite graph between the inputs and the outputs. Then the packets are transferred according to this matching. A scheduling policy may compute a *constrained matching* where no packet is destined to an output if its buffer is full. This mechanism is called "back pressure".

When a policy in the relaxed, non-FIFO, model is defined, one has also to specify how packets are sent from the output buffers. This is done by specifying the *transmission policy*. For policies in the FIFO model such specification is not needed (since packets are always sent out of the output buffers in FIFO order).

Competitive Analysis. The aim of a switch policy is that of maximizing the total value of the packets sent from the output buffers. 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 is defined as follows.

Definition 1 *An online switch policy A is said to be c-competitive if for every input sequence of packets σ, $V^{OPT}(\sigma) \leq c \cdot V^A(\sigma) + a$, where a is a constant independent of σ.*

The competitive ratio of a buffer management policy for a single FIFO buffer is defined in a similar way.

3 Switch Policies

In this section we describe the switch policies that we consider in this paper. First we define a simple tail-drop buffer management policy.

Tail Drop Buffer Management Policy (TD).
Accept the arriving packet p if there is free space in the buffer. Drop p in case the buffer is full.

Next we present a natural preemptive greedy buffer management policy.

Greedy Buffer Management Policy (GRD).
Accept the 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 the packet with the minimal value and accept p.

Now we describe a greedy switch policy for unit value packets.

Greedy Unit Switch Policy (GU).
Input/Output Buffer Management: Apply the TD policy.
Scheduling: The set of eligible packets is defined with respect to the FIFO order, with the restriction that no packet is destined to an output if its buffer is full (i.e., back pressure is enforced). Compute a maximum size matching.

Now we turn to the case of variable value packets. Following [6], we define a switch policy that is based on a simulation of another switch policy that may break the FIFO order, i.e., in this simulation packets from a queue may be sent in an arbitrary order. The scheduling decisions of the simulated policy will be used to determine the actual scheduling of our switch policy, which will extract (possibly different) packets from the same input queues at the same scheduling cycles, but in the FIFO order. First we define a greedy switch policy in the relaxed non-FIFO model.

Greedy Variable Relaxed Switch Policy (GVR).
Input/Output Buffer Management: Apply the GRD policy.
Scheduling: The set of eligible packets includes one packet from each $VOQ_{i,j}$. The eligible packet from a queue $VOQ_{i,j}$ is a packet with the maximal value among the packets in $VOQ_{i,j}$. Compute a maximum weight matching.
Transmission: Send the packet at the head of each output queue (i.e., in FIFO order).

Next we define a greedy switch policy in the FIFO model that uses the schedule of the GVR policy, and an arbitrary buffer management policy P for the input buffers.

Greedy Variable FIFO Switch Policy (GVF^P).
Input Buffer Management: Apply the policy P.
Output Buffer Management: Apply the GRD policy.
Scheduling: Simulate the GVR switch policy and follow its schedule.

Similarly, we define another switch policy in the non-FIFO model, to be later used in the construction of a switch policy in the FIFO model. This policy partitions the resources of the switch (buffer space and internal and output bandwidth) equally between different classes of packets. We divide the packets into classes according to their values. If $k \leq 2\lceil \log \alpha \rceil$, we divide the packets into k classes so that each class contains packets with the same value. Otherwise we define $\lceil \log \alpha \rceil$ classes where the packets in class $1 \leq i \leq \lceil \log \alpha \rceil$ have values in the range $[2^{i-1}, 2^i)$. Let M be the number of classes, that is $M = k$ if $k \leq 2\lceil \log \alpha \rceil$ and $M = \lceil \log \alpha \rceil$ otherwise.

Partition Variable Relaxed Switch Policy (PVR).
Input/Output Buffer Management: For a buffer of size B, allocate B/M buffer

space for each class (i.e., simulate a complete partition policy of every buffer). Apply the TD policy within each class using the space allocated for that class. Namely, a packet of class i is accepted iff the number of packets of this class in the buffer does not exceed B/M.
Scheduling (scheduling cycle t_s): We define $j = ((t-1) \cdot S + s)\%M + 1$ to be the current packet class in a Round-Robin order. The set of eligible packets is the set of packets in class j, with the restriction that no packet is destined to an output if its buffer is full (i.e., back pressure is enforced). A buffer is considered full if all the space allocated to the relevant class is occupied. Compute a maximum size matching.
Transmission (time step t): We define $j = (t-1)\%M + 1$ to be the current packet class in a Round-Robin order. Send the packet of the j-th class closest to the head of the buffer (i.e., we transmit packets out of each output buffer in a Round-Robin order between the classes).

The corresponding partition switch policy in the FIFO model is as follows.

Partition Variable FIFO Switch Policy (PVF^P).
Input Buffer Management: Apply the policy P.
Output Buffer Management: Apply the TD policy.
Scheduling: Simulate the PVR switch policy and follow its schedule.

4 Analysis of the Switch Policies

In this section we analyze the performance of our switch policies.

4.1 Unit Value Packets

In this section we consider the case of packets with unit values. We show that the GU policy is 3-competitive for any speedup and 2-competitive for a speedup of 1. We note that a result in [6] implies that no online deterministic switch policy can have a competitive ratio better that $2 - 1/N$.

In what follows we assume a given input packet sequence σ. To analyze the throughput of the GU policy we introduce some helpful definitions. The next definition concerns packets that OPT may deliver during a time step while GU does not (recall that the value of each packet is exactly 1).

Definition 2 *For a given switch policy A, a packet sent by OPT from output j at time t is said to be* extra *if $V(P_j^{OPT}(t)) = 1$ and $V(P_j^A(t)) = 0$.*

In the following definition we consider the difference between the queue length of an online policy A and OPT, which will be later related to extra packets.

Definition 3 *For a switch policy A and a scheduling cycle t_s, we denote* $\max(L_{i,j}^{OPT}(t_s) - L_{i,j}^{A}(t_s), 0)$ *by* $DL_{i,j}^{A,OPT}(t_s)$ *and* $\max(O_j^{OPT}(t_s) - O_j^{A}(t_s), 0)$ *by* $DO_j^{A,OPT}(t_s)$.

We will map each extra packet of OPT to a packet sent by GU, in a such way that each GU packet is mapped to at most twice. First we need some auxiliary claims.

Observation 1 *Consider an extra packet p, and let t_s be the scheduling cycle in which p was transferred by OPT to its output buffer. Then, at the beginning of scheduling cycle t_{s+1}, p's position in the output queue of OPT is larger than the length of the corresponding output queue maintained by GU.*

The observation follows from the fact that all extra packets sent by OPT should eventually appear in an output queue of OPT when the corresponding GU queue is empty, and from the fact that both OPT and GU send packets from output buffers greedily.

Definition 4 *We call a packet p scheduled during scheduling cycle t_s to OQ_j of OPT a* potentially extra packet, *if* $L^{OPT}(p, t_{s+1}) > O_j^{GU}(t_{s+1})$.

The following claim bounds from above the number of new potentially extra packets that can be created in OPT during a scheduling cycle.

Claim 1 *The number of new potentially extra packets created during scheduling cycle t_s, that is $\sum_{j=1}^{N} DO_j^{GU,OPT}(t_{s+1}) - \sum_{j=1}^{N} DO_j^{GU,OPT}(t_s)$, is bounded from above by the size of a maximum matching in the graph $G' = (V_{N_I,N_O}, E^{OPT}(t_s) \setminus E^{GU}(t_s))$ plus the size of a maximum constrained matching in the graph $G^{GU}(t_s)$.*

Proof: Obviously, the number of packets from G' scheduled by OPT during scheduling cycle t_s is bounded by the size of a maximum matching in G'. It remains to consider the packets that OPT schedules from $G'' = (V_{N_I,N_O}, E^{OPT}(t_s) \cap E^{GU}(t_s))$. Assume that OPT and GU schedule matchings M in G'' and MC in $G^{GU}(t_s)$, respectively. If $|M| \leq |MC|$, we are done. So suppose that $|M| > |MC|$. It must be the case that M contains at least $|M| - |MC|$ packets destined to the outputs which have full buffers in GU. Otherwise, there exists another constrained matching MC' obtained from M by removing the packets destined to the full outputs in GU such that $|MC'| > |MC|$, which contradicts to maximality of MC. Obviously, OPT cannot produce new potentially extra packets in the output buffers that are currently full in GU. Therefore, the number of new potentially extra packets from G'' is bounded by the size of a maximum constrained matching in $G^{GU}(t_s)$. ■

The next claim takes care of the situation in which the difference between the length of an input queue of OPT and the corresponding queue of GU grows.

Claim 2 *For a given scheduling cycle t_s, the increase in the difference between the length of an input queue of OPT and the length of the corresponding input queue of GU, that is $DL_{i,j}^{GU,OPT}(t_{s+1}) - DL_{i,j}^{GU,OPT}(t_s)$, is bounded by $X_{i,j}^{GU}(t_s) - X_{i,j}^{OPT}(t_s)$.*

Proof: If $s \neq S$ (i.e., the considered cycle is not the last cycle of a time step) then between scheduling cycle t_s and scheduling cycle t_{s+1} there is no arrival phase and trivially $DL_{i,j}^{GU,OPT}(t_{s+1}) - DL_{i,j}^{GU,OPT}(t_s)$ is a binary indicator of whether GU scheduled some packet at this scheduling cycle while OPT did not schedule any. Otherwise, if $s = S$, then between scheduling cycle t_s and scheduling cycle t_{s+1} a packet p may arrive to $VOQ_{i,j}$. Note that GU admits p unless its buffer is full while OPT may or may not accept it. But this may only decrease the difference. ■

The following mapping routine guarantees that all potentially extra packets are mapped to packets sent by GU (this will be proved in what follows). The routine is executed at each scheduling cycle, and adds some mappings according to the actions of GU and OPT. Note that once a packet of OPT is mapped to some packet of GU, this mapping is never changed.

Mapping Routine (scheduling cycle t_s).
Step 1. For each output j, and each input i, if $L_{i,j}^{OPT}(t_{s+1}) > L_{i,j}^{GU}(t_{s+1})$ and $DL_{i,j}^{GU,OPT}(t_{s+1}) > DL_{i,j}^{GU,OPT}(t_s)$ then map the last packet that is still unmapped in $VOQ_{i,j}$ of OPT to the packet scheduled by GU from input i to output j during scheduling cycle t_s.
Step 2. For each *unmapped* packet p scheduled by OPT to output j during scheduling cycle t_s such that $L^{OPT}(p, t_{s+1}) > O_j^{GU}(t_{s+1})$, map p to a packet scheduled during scheduling cycle t_s by GU that is mapped to at most once.

Note that each GU packet is mapped to at most twice (once at Step 1 and once at Step 2).

Lemma 1 *The mapping routine is feasible. Each packet of OPT that becomes a potentially extra packet is immediately mapped.*

Proof: The proof is by induction on the scheduling cycle. The basis is trivial. Suppose that the mapping is feasible till scheduling cycle t_{s-1} and let us show that it is also feasible at scheduling cycle t_s. By Claim 2, there exists a sufficient number of packets scheduled by GU to be mapped to at Step 1 and each such packet can therefore be used exactly once. Each such packet has not been previously used by the mapping routine since it was not yet scheduled. We now consider Step 2. According to Claim 1, the number of new potentially extra packets is bounded by the size of a maximum matching in the graph

$G' = (V_{N_I,N_O}, E^{OPT}(t_s) \setminus E^{GU}(t_s))$ plus the size of a maximum constrained matching in the graph $G^{GU}(t_s)$. However, all packets from G' are already mapped by Step 1 at scheduling cycle t_s or beforehand. Thus, the number of new potentially extra packets that are still to be mapped is bounded by the number of packets scheduled by GU during t_s. Hence, Step 2 is feasible as well, which proves the lemma. ■

Now we are ready to show that GU has the competitive ratio of 3.

Theorem 1 *The competitive ratio of GU is at most 3 for any speedup.*

Proof: Clearly, the number of packets sent by OPT is bounded by the number of packets sent by GU plus the number of extra packets. By Observation 1, every extra packet must first be a potentially extra packet. Lemma 1 implies that all potentially extra packets are mapped by the mapping routine to GU packets and no GU packet is mapped to more than twice. Therefore, $V^{OPT}(\sigma) \leq 3V^{GU}(\sigma)$, for any input sequence σ. ■

We also show that GU achieves the competitive ratio of 2 for the special case of $S = 1$.

Theorem 2 *The competitive ratio of GU is at most 2 for a speedup $S = 1$.*

Proof: We use a variant of the mapping routine, in which in Step 2 *every* unmapped packet scheduled by OPT is mapped (i.e., we drop the condition that only packets satisfying $L^{OPT}(p, t_{s+1}) > O_j^{GU}(t_{s+1})$ are mapped).

First, observe that the mapping routine remains feasible (for $S = 1$). To see that note that any OPT packet that has to be mapped in Step 2 (i.e., it is not already mapped), is in a buffer $VOQ_{i,j}$ such that $L_{i,j}^{GU}(t_s) > 0$. For $S = 1$, GU schedules a maximum size matching on its nonempty buffers (no back pressure is used since in fact there are no output buffers). Therefore, the number of packets that have to be mapped in Step 2 at scheduling cycle t_s is at most the number of packets scheduled by GU during t_s.

Furthermore, observe that the modified mapping routine maps *every* packet scheduled by OPT out of the input buffers. The theorem follows since the mapping routine maps at most two OPT packets to every packet scheduled by GU, and GU transmits out of the switch every packet scheduled out of the input buffers. ■

Most of the scheduling policies in commercial switches are based on maximal matching, which can be easily computed in a distributed fashion, as opposed to maximum matching. If GU uses maximal matching rather than maximum matching, its competitive ratio is increased by 1, compared to the original policy. To see that we can use a mapping routine in which in Step 2 every

GU packet is used for the mapping of at most two OPT packets (rather than just one OPT packet as in the original routine). Observe that the modified mapping routine remains feasible. The feasibility of Step 1 does not depend on the particular scheduling policy used. The feasibility of Step 2 follows from the arguments for the original GU policy and from the fact that the size of a maximal matching is at least half the size of a maximum matching. We therefore have that GU with maximal matching is 4-competitive in the general case and 3-competitive in the case of $S = 1$.

4.2 Variable Value Packets

In this section we consider the case of packets with variable values. We study two policies that may use an arbitrary local buffer management policy P for the input buffers (it may be preemptive or non-preemptive). Let the competitive ratio of P for a single buffer be C_P and let $M' = \min(k, 2\lceil \log \alpha \rceil)$. We show that the GVF^P policy is $(2 \cdot S \cdot C_P)$-competitive and that the PVF^P policy is $(4 \cdot M' \cdot C_P)$-competitive. This implies that GVF^{GRD} and PVF^{GRD} are $4S$-competitive and $8M'$-competitive, respectively, since GRD is 2-competitive [16].

4.2.1 Simulation Technique

We extend the technique of [6]. Specifically, we show that by combining a schedule of a C_R-competitive switch policy (which does not drop packets at the outputs) that runs in the relaxed non-FIFO model, together with any C_P-competitive local (input) buffer management policy, we obtain a new switch policy that achieves a competitive ratio of $C_R \cdot C_P$ in the FIFO model. First we need some lemmas. The following lemma shows that for any given finite input sequence, the value of an optimal solution in the FIFO model equals that of an optimal solution in the relaxed non-FIFO model.

Lemma 2 *For any finite input sequence σ, the value of OPT in the non-FIFO model equals the value of OPT in the FIFO model.*

Proof: We argue that any feasible schedule in the non-FIFO model can be transformed to a schedule in the FIFO model, in which the same set of packets is sent. First, without loss of generality, assume that OPT in the non-FIFO model never preempts packets at the inputs or drops packets at the outputs. If this is not the case, one can admit to the input buffers only packets that are eventually sent from the output buffers without affecting the value of the solution. Second, we transform the schedule by swapping the order in which packets are sent so that FIFO order is maintained. Such a transformation is always feasible since no packet is scheduled before its arrival time. The value of the resulting solution does not change since the number of packets in any buffer at any given time does not change. Hence, no packet is dropped at the buffers or the output buffers. The lemma follows. ■

Clearly, a similar claim holds when we consider a single buffer rather than a switch with input and output buffers.

In the next lemma we consider the total value of packets *scheduled out of input buffers* by a switch policy operating in the FIFO model, which uses an arbitrary buffer management policy P, and whose schedule is defined by another (simulated) switch policy which operates in the non-FIFO model. We compare the total value of packets scheduled out of input buffers by this policy to the total value of packets scheduled out of input buffers by the optimal policy (i.e., we consider the competitive ratio of this policy, with respect to the measure of packets scheduled out of the input buffers, rather than packets sent out of the switch). The ratio we show is a function of the competitive ratio of the simulated policy, and the competitive ratio of its input buffer management policy for a single buffer, P. A similar claim is implicitly proved in [6]. For an input sequence σ and a schedule H, we denote the total value of packets scheduled out of input buffers by a policy A in model M (FIFO or non-FIFO) by $V_M^A(\sigma, H)$. We also denote the total value of packets scheduled out of $VOQ_{i,j}$ by A in model M by $V_M^A(VOQ_{i,j}, \sigma, H)$.

Lemma 3 *Fix an input sequence σ. Consider a switch policy A_R, running in the non-FIFO model, in which every packet scheduled out of an input buffer is eventually sent out of the switch. Consider now a switch policy A running in the FIFO model, that uses the schedule H of A_R on σ, and a buffer management policy P for the input buffers. Then, the total value of packets scheduled out of the input buffers by A is at least $V_{FIFO}^A(\sigma, H) \geq V_{FIFO}^{OPT}(\sigma)/(C_R \cdot C_P)$, where C_R is the competitive ratio of A_R, and C_P is the competitive ratio of P.*

Proof: Fix a schedule H of A_R on input sequence σ. In a nutshell, H defines the time steps at which the various input queues are allowed to transmit. Since P is C_P-competitive in the FIFO model,

$$V_{FIFO}^A(VOQ_{i,j}, \sigma, H) \geq V_{FIFO}^{OPT}(VOQ_{i,j}, \sigma, H)/C_P.$$

By arguments similar to those in the proof of Lemma 2, applied to a single buffer, we get:

$$\begin{aligned} V_{non-FIFO}^{A_R}(VOQ_{i,j}, \sigma, H) &\leq V_{non-FIFO}^{OPT}(VOQ_{i,j}, \sigma, H) \\ &= V_{FIFO}^{OPT}(VOQ_{i,j}, \sigma, H). \end{aligned}$$

Hence we obtain:

$$V_{FIFO}^A(VOQ_{i,j}, \sigma, H) \geq V_{non-FIFO}^{A_R}(VOQ_{i,j}, \sigma, H)/C_P.$$

Notice that the value of A equals the total value sent out of the input buffers since no packet is dropped at the outputs. Therefore, we obtain:

$$\begin{aligned}
V^A_{FIFO}(\sigma, H) &= \sum_{i=1}^{N}\sum_{j=1}^{N} V^A_{FIFO}(VOQ_{i,j}, \sigma, H) \\
&\geq \sum_{i=1}^{N}\sum_{j=1}^{N} V^{A_R}_{non-FIFO}(VOQ_{i,j}, \sigma, H)/C_P \\
&= V^{A_R}_{non-FIFO}(\sigma)/C_P \\
&\geq V^{OPT}_{non-FIFO}(\sigma)/(C_R \cdot C_P) \\
&= V^{OPT}_{FIFO}(\sigma)/(C_R \cdot C_P),
\end{aligned}$$

where we use the fact that A_R is C_R-competitive and the last equality follows by Lemma 2. ■

4.2.2 Analysis of the GVF^P Policy

First we demonstrate that GVR is 2-competitive for $S = 1$ in the non-FIFO model. We emphasize that the GVR policy is used only as a simulation tool to determine the scheduling decisions of the GVF^P policy.

We follow the line of the proof in [6]. Let us denote by $R_{i,j,k}(\tau)$ the packet with the k-th largest value in $VOQ_{i,j}$ just before an arbitrary time τ (τ may be a time step t, or a phase t_A or t_T or a scheduling cycle t_s). For $k > L_{i,j}(\tau)$, let $R_{i,j,k}(\tau) = 0$. We define the potential of the system just before time τ to be the sum of all *positive* pairwise differences between the sorted values in all input queues of OPT and GVR just before time τ. That is,

$$\Phi(\tau) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{BI_{i,j}} \max((R^{OPT}_{i,j,k}(\tau) - R^{GVR}_{i,j,k}(\tau)), 0).$$

Notice that $\Phi(\tau) \geq 0$ for any time τ.

The next claim follows immediately from the proof of Claim 1 in [6].

Claim 3 ([6]) *Under the GRD buffer management policy, the potential does not increase during the arrival phase.*

Let us denote by $W^A(\sigma, t)$ the total value of packets that a switch policy A *schedules* out of the input buffers by the end of time step t on input sequence σ. We give the following lemma.

Lemma 4 *For any speedup S, for any sequence of packets σ and for any time step t, $W^{OPT}(\sigma, t) + \Phi(t+1) \leq 2W^{GVR}(\sigma, t)$.*

Proof: The proof is by induction on the time step. We assume that initially all buffers of both GVR and OPT are empty. Therefore we have that $\Phi(1) = 0$, and that the lemma trivially holds at time $t = 0$. Now assume that the lemma holds for time step t and let us show that it also holds for time step $t+1$.

We consider the transmission phase, the packet arrival phase and the scheduling phase separately. For a given phase, let us denote by ΔW^{GVR} the total value of packets scheduled by GVR during this phase, by ΔW^{OPT} the total value of packets scheduled by OPT during this phase and by $\Delta\Phi$ the increase in Φ during this phase.

Transmission phase. Notice that for the transmission phase we have $\Delta W^{GVR} = \Delta W^{OPT} = 0$, and that Φ does not change. We therefore have for the transmission phase $\Delta W^{OPT} + \Delta\Phi \leq \Delta W^{GVR}$.

Arrival phase. Next we deal with the packet arrival phase. Clearly, $\Delta W^{GVR} = \Delta W^{OPT} = 0$. By Claim 3, $\Delta\Phi \leq 0$. Therefore, we have that $\Delta W^{OPT} + \Delta\Phi \leq \Delta W^{GVR}$ for the arrival phase.

Scheduling phase. Last, we concentrate on the scheduling phase. We consider the scheduling cycles of the phase in sequence. For a given cycle s, denote by ΔW_s^{GVR} the total value of packets scheduled by GVR during the cycle, by ΔW_s^{OPT} the total value of packets scheduled by OPT during the cycle and by $\Delta\Phi_s$ the increase in Φ during the cycle. The increase in the potential during the scheduling cycle of GVR is as follows.

$$\begin{aligned}
\Delta_1\Phi = & \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{BI_{i,j}} X_{i,j}^{GVR}(t_s)\max((R_{i,j,k}^{OPT}(t_s) - R_{i,j,k+1}^{GVR}(t_s)), 0) \\
& - \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{BI_{i,j}} X_{i,j}^{GVR}(t_s)\max((R_{i,j,k}^{OPT}(t_s) - R_{i,j,k}^{GVR}(t_s)), 0) \\
& \leq \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{BI_{i,j}} X_{i,j}^{GVR}(t_s) \cdot \\
& \cdot (\max((R_{i,j,k}^{OPT}(t_s) - R_{i,j,k}^{GVR}(t_s)), 0) + (R_{i,j,k}^{GVR}(t_s) - R_{i,j,k+1}^{GVR}(t_s))) \\
& - \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{BI_{i,j}} X_{i,j}^{GVR}(t_s)\max((R_{i,j,k}^{OPT}(t_s) - R_{i,j,k}^{GVR}(t_s)), 0) \\
& = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{BI_{i,j}} X_{i,j}^{GVR}(t_s)(R_{i,j,k}^{GVR}(t_s) - R_{i,j,k+1}^{GVR}(t_s))
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{BI_{i,j}} X_{i,j}^{GVR} R_{i,j,1}(t_s) \\
&= \Delta W_s^{GVR}.
\end{aligned}
$$

Now we consider the scheduling cycle of OPT. Suppose that OPT schedules the $r_{i,j}$-th largest packet from $VOQ_{i,j}$, if any. Otherwise (if OPT does not schedule a packet from $VOQ_{i,j}$), let $r_{i,j} = BI_{i,j} + 1$. We have that the increase in the potential during the scheduling cycle of OPT is as follows.

$$
\begin{aligned}
\Delta_2 \Phi &= \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=r_{i,j}}^{BI_{i,j}} X_{i,j}^{OPT}(t_s) \max((R_{i,j,k+1}^{OPT}(t_s) - R_{i,j,k}^{GVR}(t_s)), 0) \\
&- \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=r_{i,j}}^{BI_{i,j}} X_{i,j}^{OPT}(t_s) \max((R_{i,j,k}^{OPT}(t_s) - R_{i,j,k}^{GVR}(t_s)), 0) \\
&\leq \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=r_{i,j}+1}^{BI_{i,j}} X_{i,j}^{OPT}(t_s) \max((R_{i,j,k}^{OPT}(t_s) - R_{i,j,k}^{GVR}(t_s)), 0) \\
&- (\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=r_{i,j}+1}^{BI_{i,j}} X_{i,j}^{OPT}(t_s) \max((R_{i,j,k}^{OPT}(t_s) - R_{i,j,k}^{GVR}(t_s)), 0) \\
&+ \sum_{i=1}^{N} \sum_{j=1}^{N} X_{i,j}^{OPT}(t_s) \max((R_{i,j,r_{i,j}}^{OPT}(t_s) - R_{i,j,r_{i,j}}^{GVR}(t_s)), 0)) \\
&\leq \sum_{i=1}^{N} \sum_{j=1}^{N} X_{i,j}^{OPT}(t_s) (R_{i,j,r_{i,j}}^{GVR}(t_s) - R_{i,j,r_{i,j}}^{OPT}(t_s)) \\
&\leq \sum_{i=1}^{N} \sum_{j=1}^{N} X_{i,j}^{GVR}(t_s) R_{i,j,1}^{GVR}(t_s) - \Delta W_s^{OPT} \\
&= \Delta W_s^{GVR} - \Delta W_s^{OPT}.
\end{aligned}
$$

The last inequality follows from the fact that GVR computes a maximum weight matching that in particular has weight greater than or equal to the weight of the matching scheduled by OPT with respect to the GVR packets. Putting it altogether, we obtain that for scheduling cycle s

$$
\begin{aligned}
\Delta W_s^{OPT} + \Delta \Phi_s &= \Delta W^{OPT} + \Delta_1 \Phi + \Delta_2 \Phi \\
&\leq \Delta W_s^{OPT} + \Delta W_s^{GVR} + \Delta W_s^{GVR} - \Delta W_s^{OPT} \\
&= 2 \Delta W_s^{GVR}.
\end{aligned}
$$

Summing over all the scheduling cycles we have for the scheduling phase that

$$\Delta W^{OPT} + \Delta\Phi \leq 2\Delta W^{GVR}.$$

The lemma now follows from the inductive hypothesis and the three inequalities for the three phases. ■

Using the above lemma, we can prove the following theorem.

Theorem 3 *The competitive ratio of GVR is at most 2 for a speedup $S = 1$.*

Proof: Suppose that OPT schedules the last packet in σ out of an input buffer at time step t^*. Since Φ is nonnegative at any time we have that by Lemma 4, $W^{OPT}(\sigma, t^*) \leq 2W^{GVR}(\sigma, t^*)$. Note that for a speedup $S = 1$, any packet scheduled by GVR to an output buffer at time step t is transmitted at time step $t+1$. Therefore, all packets scheduled by GVR to output buffers are also transmitted out of the switch. The theorem follows since OPT transmits out of the switch at most the total value of packets scheduled out of the input buffers. ■

Now we can derive the competitive ratio of the GVF^P policy. For the case of a speedup $S = 1$, for which GVR does not drop packets at the outputs, the competitive ratio follows directly from Lemma 3 and Theorem 3.

Theorem 4 *The competitive ratio of the GVF^P policy is at most $2C_P$ for a speedup $S = 1$.*

Next we consider the case of $S > 1$.

Theorem 5 *The competitive ratio of the GVF^P policy is at most $2SC_P$ for any speedup.*

Proof: Intuitively, the theorem follows since for a speedup of $S > 1$, GVF^P loses at the output buffers at most a factor of S with respect to the value of packets scheduled out of the input buffers.

To give a formal proof of the theorem, we proceed as follows. We first consider the GVR policy in a setting where the output buffers are not limited in space. That is, any packet scheduled out of the input buffers is added to an output buffer and is eventually sent out of the switch. There is no shortage of buffer space at the outputs. In this setting, we have that the GVR policy is 2-competitive for any speedup S by arguments similar to those used in the proof of Theorem 3. Considering still this setting we therefore have that GVF^P is $2C_P$ competitive for any speedup using Lemma 3 (as for the case of $S = 1$).

We now compare the total value of packets sent by GVF^P in the above setting to the total value of packets sent by GVF^P in the ("regular") setting of output buffers of limited space. We argue that we loose at most a factor of S with respect to the setting of output buffers of unlimited space. To see this we note that each output buffer is managed separately by GRD. Denote by x the total value of packets accepted by GRD into an output buffer and by y the total value of packets preempted by GRD from the output buffer. Then, the total value of packets sent out of an output buffer equals $x - y$. Now observe that for each time step one packet is first sent out of the output buffer (unless it is empty) and then at most S packets arrive. Therefore, before the S packets arrive there is at least one free slot in the buffer. It follows that for each time step, the increase in $x - y$ is at least an $1/S$ fraction of the total value of the packets that arrive to the output buffer. This concludes the proof of the theorem. ■

4.2.3 Analysis of the PVF^P Policy

Next we demonstrate that PVF^P achieves a competitive ratio of $4M'C_P$ for any speedup (recall that $M' = \min(k, 2\lceil \log \alpha \rceil)$). First we demonstrate that the simulated PVR policy is $4M'$-competitive. For a given input sequence σ, and any class l, we will compare the total value of packets of the l-th class sent by PVR, denoted $V_l^{PVR}(\sigma)$, to the total value of packets of the same class sent by OPT, denoted $V_l^{OPT}(\sigma)$, and show that $V_l^{PVR}(\sigma) \geq V_l^{OPT}(\sigma)/(4M')$.

In what follows we assume a given input sequence σ and restrict our attention to packets from the l-th class. Recall that PVR operates in the non-FIFO model and divides the buffer space and the bandwidth equally between the different classes. Roughly speaking, each class is allocated $1/M$ fraction of the input buffer space, switch fabric bandwidth, output buffer space and the output bandwidth. We will map each packet from the l-th class sent by OPT to a packet sent by PVR, in such a way that each PVR packet is mapped to at most $4M$ times.

Let $VOQ_{i,j}^{PVR}(l)$ and $OQ_j^{PVR}(l)$ be the space of $VOQ_{i,j}$ and OQ_j, respectively, allocated by PVR to packets from the l-th class. Let us denote by $L_{i,j}^{PVR}(t_s, l)$ and $O_j^{PVR}(t_s, l)$ the number of packets in $VOQ_{i,j}^{PVR}(l)$ and $OQ_j^{PVR}(l)$ at the beginning of scheduling cycle t_s.

The following mapping routine guarantees that all packets from the l-th class sent by OPT are mapped to packets from the l-th class sent by PVR (this will be proved in what follows). The routine is executed at each scheduling cycle at which PVR schedules packets from the l-th class, and adds or modifies some mappings according to the actions of PVR and OPT.

Mapping Routine (scheduling cycle t_s at which PVR scheduled the

l-th class).
Let t'_q be the previous scheduling cycle at which PVR scheduled the l-th class.
Step 1. Consider each input queue $VOQ_{i,j}$ separately. If during $(t'_q, t_s]$ PVR accepted at least one packet of the l-th class into $VOQ_{i,j}^{PVR}(l)$, then add a mapping from all the packets of the l-th class accepted by OPT into $VOQ_{i,j}$ during $(t'_q, t_s]$ to that packet.
If during $(t'_q, t_s]$ both PVR and OPT did not accept any packet of the l-th class into $VOQ_{i,j}$ nothing has to be done.
If during $(t'_q, t_s]$ PVR did not accept any packet into $VOQ_{i,j}^{PVR}(l)$ while OPT accepted at least one packet of the l-th class into $VOQ_{i,j}$, $VOQ_{i,j}^{PVR}(l)$ must be full just before t_s. In this case, cancel all mappings involving packets of PVR that are in $VOQ_{i,j}^{PVR}(l)$ just before t_s (if any). Then, add a mapping from all the OPT packets of the l-th class that are in $VOQ_{i,j}$ just before t_s, to the packets of PVR in $VOQ_{i,j}^{PVR}(l)$ just before t_s, in an even way. (We will later show that each packet of PVR is mapped to at most M times in any of the cases.)
Step 2. Consider all packets from the l-th class scheduled during $(t'_q, t_s]$ by OPT from an input queue $VOQ_{i,j}$ such that $L_{i,j}^{PVR}(t_s, l) > 0$ to an output queue OQ_j such that $O_j^{PVR}(t_s, l) < BO_j/M$. First, cancel all mappings involving these packets (if any). Then, add a mapping from all these packets to packets from the l-th class scheduled by PVR during scheduling cycle t_s in an even way. (We will later show that each PVR packet is mapped to at most M times.)
Step 3. Consider all packets from the l-th class scheduled during $(t'_q, t_s]$ by OPT to output queues OQ_j such that $OQ_j^{PVR}(l)$ is full just before t_s, i.e. $O_j^{PVR}(t_s, l) = BO_j/M$. First, cancel all the mappings involving these packets (if any). Then, map each such packet p scheduled by OPT to OQ_j during scheduling cycle t''_r ($t'_q < t''_r \leq t_s$) to the packet in position $\lfloor L^{OPT}(p, t''_r)/M \rfloor$ in $OQ_j^{PVR}(l)$ just before t_s. (We will later show that such a packet must exist, and that any such packet is mapped to at most $2M$ times in Step 3 during the execution of the algorithm.)

Consider an OPT packet p of the l-th class scheduled from $VOQ_{i,j}$ to O_j. The mapping routine is built so that p is mapped no later than scheduling cycle t_s, where t_s is the first scheduling cycle after p is scheduled by OPT, in which PVR schedules the l-th class. If $VOQ_{i,j}^{PVR}(l)$ is not empty just before t_s, then p is mapped by either Step 2 or Step 3 (depending on whether $O_j^{PVR}(l)$ is full or not just before t_s). If $VOQ_{i,j}^{PVR}(l)$ is empty just before t_s, then the mapping of p that is done in Step 1 (possibly already in previous cycles) is used.

First we demonstrate that all OPT packets are mapped.

Claim 4 *Each packet from the l-th class sent by OPT is mapped to a PVR packet from the l-th class.*

Proof: We show that each OPT packet p of the l-th class scheduled from $VOQ_{i,j}$ to O_j receives a mapping which is final (i.e. never changes) no later than t_s, where t_s is the first scheduling cycle after p is scheduled by OPT, in which PVR schedules the l-th class. First observe that each packet accepted by OPT to some input queue at time step t', is mapped by Step 1 no later than t'_s, where t'_s is the first scheduling cycle in which PVR scheduled packets of the l-th class in time step t' or later. Now, to see that p receives its final mapping by the claimed time, observe that by the mapping routine, p is never mapped again *after* t_s. If $VOQ_{i,j}^{PVR}(l)$ is not empty and $O_j^{PVR}(l)$ is not full just before t_s, then p is mapped by Step 2 at t_s. If $VOQ_{i,j}^{PVR}(l)$ is not empty and $O_j^{PVR}(l)$ is full just before t_s, then p is mapped at t_s by Step 3. If $VOQ_{i,j}^{PVR}(l)$ is empty just before t_s, then the mapping of p created in Step 1 either at t_s or beforehand is used (observe that p must be mapped in Step 1 no later than t_s). ■

Next we show that there always exists a PVR packet to map to as required by the mapping routine, and that each PVR packet is mapped to at most $4M$ times.

Lemma 5 *The mapping routine is feasible. Each PVR packet is mapped to at most $4M$ times.*

Proof: We consider each of the steps of the the mapping routine separately.

Step 1. First observe that by the specifications of the step, the required PVR packets always exist. Next observe that if a PVR packet is mapped to in more than one execution of Step 1 (in two different scheduling cycles) then the previous mappings are first canceled. We now argue that each PVR packet is mapped to at most M times in a single execution of Step 1. Note that during $(t'_q, t_s]$ at most M packets (of the l-th class) can arrive (and be accepted by OPT) to $VOQ_{i,j}$. Therefore, if in Step 1 a mapping to a new accepted packet of PVR is created, at most M packets are mapped to it. In case in Step 1 new mappings are created for all the OPT packets of the l-th class in $VOQ_{i,j}$, then observe that $L_{i,j}^{OPT}(t_s) \leq BI_{i,j}$ and $L_{i,j}^{PVR}(t_s, l) = BI_{i,j}/M$.

Step 2. Consider all packets from the l-th class scheduled by OPT during $(t'_q, t_s]$ from an input queue $VOQ_{i,j}$ such that $L_{i,j}^{PVR}(t_s, l) > 0$ to an output queue OQ_j such that $O_j^{PVR}(t_s, l) < BO_j/M$. Since PVR computes a maximum size constrained matching during t_s and in each input queue under consideration there exists a packet (from the l-th class) that is eligible for scheduling, the number of packets from the l-th class scheduled by OPT from these queues during $(t'_q, t_s]$ is bounded by M times the the number of packets from the l-th class scheduled by PVR during t_s. Thus, we can map all such packets scheduled by OPT to packets scheduled by PVR so that each PVR packet is mapped to at most M times.

Step 3. Consider a packet p scheduled by OPT that arrives at t''_r, $t'_q < t''_r \leq t_s$, to an output queue OQ_j of OPT such that $O_j^{PVR}(t_s, l) = BO_j/M$. The specified mapping is well-defined because $O_j^{OPT}(t''_r) \leq BO_j$ and $O_j^{PVR}(t_s, l) = BO_j/M$. Note that for any scheduling cycle t_s in which a mapping is added in Step 3, no packet is sent from $OQ_j^{PVR}(l)$ during $[t'_q, t_s)$, because otherwise $OQ_j^{PVR}(l)$ would not be full just before scheduling cycle t_s. Therefore, the position of a packet p to which a mapping is added in Step 3 at scheduling cycle t_s is the same position it had in any scheduling cycle in $[t'_q, t_s)$ (and in particular such packet was in $OQ_j^{PVR}(l)$ during the whole interval of time).

We now argue that any PVR packet receives at most $2M$ mappings in Step 3 during the course of the algorithm, that is we show that at most $2M$ different packets can be mapped to any single packet that passes through $OQ_j^{PVR}(l)$. Note that for both OPT and PVR no packet is ever preempted from OQ_j and a packet is sent out whenever possible (i.e. for OPT, in each time step when OQ_j^{OPT} is not empty, and for PVR, in each time step when packets of the l-th class are to be sent and $OQ_j^{PVR}(l)$ is not empty). Using Lemma 2, we assume without loss of generality that OPT sends packets out of OQ_j in FIFO order. We therefore observe that the relative order of packets in OQ_j (of OPT) never changes and that the distance between two packets in OQ_j (of OPT) never changes.

Thus, for a given packet p that passes through $OQ_j^{PVR}(l)$ we can identify the *first* packet (according to the order defined in OQ_j of OPT) that is mapped to p. Let this packet be p'. Denote by $\hat{t}_a$ the scheduling cycle in which it arrives to OQ_j and let q' be the position in OQ_j to which it arrives. Let t_s, $\hat{t}_a \leq t_s$ be the scheduling cycle in which the mapping between p' and p is created, and let q be the position of p in $OQ_j^{PVR}(l)$ just before t_s. By the specifications of Step 3 we have that $\lfloor q'/M \rfloor = q$. To show that at most $2M$ packets are mapped to p, we demonstrate that any packet other than p' that is mapped to p must be at distance at most $2M - 1$ from p' in OQ_j.

Assume towards a contradiction that a packet p'' at distance $\delta \geq 2M$ from p' in OQ_j is mapped to p. Let $\hat{t}_b$ be the scheduling cycle in which p'' arrives to OQ_j, and let q'' be the position of p'' in OQ_j after its arrival to OQ_j. Consider time interval $[\hat{t}_b, \hat{t}_a)$, and let x be the number of packets sent out from OQ_j by OPT in this time interval and y be the number of packets sent out from $OQ_j^{PVR}(l)$ by PVR in this time interval. Since p is present in $OQ_j^{PVR}(l)$ at both $\hat{t}_b$ and $\hat{t}_a$, we know that $OQ_j^{PVR}(l)$ is never empty during $[\hat{t}_b, \hat{t}_a)$, and therefore we have that $x \leq M(y+1)$, because (unless $OQ_j^{PVR}(l)$ is empty) OPT cannot send M packets without PVR sending at least one packet in the same time interval. We now have that $q'' = q' - x + \delta$ and therefore $\lfloor (q' - x + \delta)/M \rfloor = q - y$. On the other hand,

$$\begin{aligned}\lfloor (q' - x + \delta)/M \rfloor &\geq \lfloor (q' - M(y+1) + \delta)/M \rfloor \\ &\geq \lfloor (q' - My + M)/M \rfloor \\ &= \lfloor q'/M \rfloor - y + 1 \\ &> q - y\ .\end{aligned}$$

This is a contradiction and hence p'' cannot be mapped to p.

By the above we can conclude that at any time any PVR packet has at most $4M$ mappings: at most M mappings created at Step 1, at most M mappings created at Step 2, and at most $2M$ mappings created at Step 3. ■

The next lemma shows that PVR loses at most a factor of $4M'$ with respect to the optimal throughput of the l-th class.

Lemma 6 *For any input sequence σ and any class l, $V_l^{PVR}(\sigma) \geq V_l^{OPT}(\sigma)/(4M')$ for any speedup.*

Proof: According to Claim 4 and Lemma 5, all packets from the l-th class sent by OPT are mapped by the mapping routine, which maps to any single PVR packet of the l-th class at most $4M$ packets. The lemma follows since the values of packets in the same class differ by at most a factor of 2 if we have $M = \lceil \log \alpha \rceil$ classes, and are identical if we have $M = k$ classes. ■

The following theorem shows that the PVR policy is $(4M')$-competitive.

Theorem 6 *The competitive ratio of PVR in the non-FIFO model is at most $4M'$ for any speedup.*

Proof: By Lemma 6, $V^{PVR}(\sigma) = \sum_{l=1}^{M} V_l^{PVR}(\sigma) \geq \sum_{l=1}^{M} V_l^{OPT}(\sigma)/(4M') = V^{OPT}(\sigma)/(4M')$. ■

Observe that no packets are dropped by PVR at the outputs (because back pressure is used). The following claim states that output buffers never overflow under PVF^P.

Claim 5 *No packet is dropped at the outputs under PVF^P.*

The claim holds since at any time the total number of packets in an output buffer of PVF^P is less than or equal to the number of packets in the corresponding output buffer of PVR, which does not drop packets at the outputs.

Finally, we derive the competitive ratio of the PVF^P policy using Lemma 3, Theorem 6 and Claim 5.

Theorem 7 *The competitive ratio of the PVR^P policy is at most $4 \cdot M' \cdot C_P$ for any speedup.*

5 Comparing to OQ Switches

In this section we briefly consider the question of comparing the throughput of a CIOQ switch to that of an OQ switch with FIFO buffers that has a "similar" amount of memory. We show a memory allocation for the CIOQ switch, and a switch policy which is 4-competitive with respect to the optimal throughput achievable by the OQ switch.

We also formulate a closely related question of how to divide the memory available to a CIOQ switch between the input ports and the output ports. That is, given the total amount of memory and the speedup of the switch, how much memory should be placed at the inputs and how much at the outputs. We assume here, as in [10], that each input port has a single buffer (VOQ is not implemented) and that there is no FIFO constraint imposed on the input buffers (i.e., at any time any packet can be extracted from an input buffer). Suppose that we have a total of M units of memory, each unit capable of storing a single packet. Obviously, for $S = 1$ all M units of memory should be placed at the inputs and for $S = N$ all M units should be placed at the outputs. However, it is unclear what should be done for $1 < S < N$. Xie and Lea [29], study this question using simulations.

In the following lemma we consider a CIOQ switch with a speedup $S = 2$, and the Critical Cell First (CCF) scheduling policy for CIOQ switches [10]. This scheduling policy uses a simulation of a "shadow" OQ switch, and assuming that there are no memory space constraints, allows the CIOQ switch to completely mimic the traffic sent out of the simulated OQ switch [10]. We show a memory allocation for the CIOQ switch, and a feasible memory partition between the input and the output ports, which is sufficient for this setting. That is, the set of packets transmitted from the CIOQ switch will be identical to the set of packets transmitted from the OQ switch.

Lemma 7 *If we want to simulate a FIFO OQ switch that has a total memory of M^{OQ} slots (divided equally between the output ports) using a CIOQ switch that has a speedup $S = 2$ and uses for scheduling CCF [10], then the memory requirement of the CIOQ switch is at most $3M^{OQ}$. A feasible memory partition is to allocate $2M^{OQ}$ slots to the inputs and M^{OQ} slots to the outputs (divided equally between the input and the output ports).*

Proof: Denote by B^{OQ} the size of a buffer in the OQ switch. We denote the size of an input buffer and an output buffer in the CIOQ switch by BI^{CIOQ} and BO^{CIOQ}, respectively. We install in the CIOQ switch buffers of size $BI^{CIOQ} = 2B^{OQ}$ and $BO^{CIOQ} = B^{OQ}$. Now we show that this allocation is sufficient. Notice that the delay of a packet in the OQ switch is at most B^{OQ}. Since CCF completely mimics the OQ switch [10], the delay of any packet in the

CIOQ switch is also bounded by B^{OQ}. Therefore, any input queue would never contain more than $2B^{OQ}$ packets (recall that $S = 2$) and any output queue would never contain more than B^{OQ} packets. Otherwise, the delay constraint would be violated. ∎

We now assume that we have an OQ switch and a CIOQ switch with an arbitrary speedup, and with memory allocation and partition as in the above lemma. We define a switch policy for the CIOQ switch and compare the throughout achieved by this policy to the optimal throughput achievable on the OQ switch. This policy is defined using the CCF scheduling policy, and an arbitrary buffer management policy P.

Simulate OQ Variable Switch Policy (SOV^P).
Input Buffer Management: Simulate a FIFO OQ switch with the buffer management policy P and accept/drop packets that P accepts/drops in the simulated OQ switch.
Output Buffer Management: Apply the TD policy.
Scheduling: Every time step, apply the Critical Cell First (CCF) algorithm [10] and compute two matchings, as if $S = 2$. (CCF makes its decisions based on a simulation of the OQ switch on an input sequence that contains the packets accepted by P.) For $S \geq 2$, schedule both of the matchings. For $S < 2$, select for scheduling the matching with the maximum weight and drop the packets of the second matching.

We show an upper bound on the competitive ratio (with respect to the optimal throughout in the OQ switch) of SOV^P, as a function of C_P.

Theorem 8 *Consider a FIFO OQ switch and a CIOQ switch with memory allocation as in Lemma 7. For any speedup, the competitive ratio of SOV^P is at most $2C_P$ with respect to the optimal throughput achievable in the OQ switch.*

Proof: It is shown in [10] that if there are no memory constraints, a CIOQ switch with speedup $S = 2$ using CCF can completely mimic the OQ switch. First assume that $S \geq 2$. If the switch is provided with memory allocation and partition as in Lemma 7, then there is never a shortage of memory space, and therefore the packets sent from the CIOQ switch are identical to those sent from the OQ switch that uses P. It follows that the competitive ratio of SOV^P (with respect to the optimal throughput in the OQ switch) is at most C_P. For $S < 2$ we lose at most a factor of 2 since during each time step only the heavier matching is scheduled and the other is dropped. ∎

If we use as the buffer management policy (P) the greedy policy GRD which is 2-competitive, we have that SOV^{GRD} is 4-competitive with respect to the optimal throughout in the OQ switch.

6 Concluding Remarks

A major problem addressed today in networking research is the need for fast switch architectures supporting guaranteed QoS. In this paper we consider the CIOQ architecture that gained popularity as a platform for high performance switches. We design robust switch policies that maximize the switch throughput for any traffic pattern and use competitive analysis to analyze their performance.

An intriguing open problem is whether one can obtain a constant-competitive switch policy for an arbitrary speedup in the case of arbitrary packet values or whether a lower bound depending on the speedup can be shown. Another interesting direction is to further study how the available memory should be partitioned between the input ports and the output ports for a given speedup to achieve the best performance, and how the performance of a switch is affected by such a division.

Acknowledgments. We thank Yishay Mansour and Zvi Lotker for useful discussions. We also thank two anonymous referees for useful comments and suggestions.

References

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

[2] W. A. Aiello, Y. Mansour, S. Rajagopolan and A. Rosén, "Competitive Queue Policies for Differentiated Services," *Proceedings of INFOCOM 2000*, pp. 431-440.

[3] N. Andelman, Y. Mansour and An Zhu, "Competitive Queueing Policies for QoS Switches," *The 14th ACM-SIAM SODA*, Jan. 2003.

[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] M. Arpaci and J. A. Copeland, "Buffer Management for Shared-Memory ATM Switches," *IEEE Communications Surveys*, First Quarter 2000.

[6] Y. Azar and Y. Richter, "Management of Multi-Queue Switches in QoS Networks," *Proceedings of STOC 2003.*

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

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

[9] A. Charny, “Providing QoS Guarantees in Input-Buffered Crossbar Switches with Speedup.” *Ph.D. dissertation*, August 1998, MIT.

[10] 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*, pp. 1030-1039, vol. 17, Dec. 1999.

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

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

[13] P. Giaccone, E. Leonardi, B. Prabhakar and D. Shah, “Delay Performance of High-speed Packet Switches with Low Speedup,” *Proceedings of the IEEE GLOBECOM 2002*, Taipei, Taiwan, November 2002.

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

[15] M. Karol, M. Hluchyj and S. Morgan, “Input versus Output Queuing an a Space Division Switch”, *IEEE Trans. Communications*, Vol. 35, pp. 1347-1356, 1987.

[16] A. Kesselmanm, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber and M. Sviridenko, “Buffer Overflow Management in QoS Switches,” *SIAM Journal on Computing*, Vol. 33(3), pp. 563-583, 2004.

[17] A. Kesselman and Y. Mansour, “Loss-Bounded Analysis for Differentiated Services,” *Journal of Algorithms*, Vol. 46(1), pp. 79-95, January 2003.

[18] 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*, to appear.

[19] M. A. Marsan, A. Bianco, E. Filippi, P. Giaccone, E. Leonardi and F. Neri, “A Comparison of Input Queuing Cell Switch Architectures,” *IEEE BSS’99, 3rd International Workshop on Broadband Switching Systems*, Kingston, Canada, June 1999.

[20] M. A. Marsan, A. Bianco, E. Leonardi and L. Milia, “RPA: A Flexible Scheduling Algorithm for Input Buffered Switches,” *IEEE Transactions on Communications*, Vol.47, No.12, pp.1921-1933, December 1999.

[21] Y. Mansour, B. Patt-Shamir and Ofer Lapid, “Optimal Smoothing Schedules for Real-Time Streams,” *Proceedings of PODC 2000*, pp. 21-29.

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

[23] N. Mckeown and A. Mekkittikul, “A Starvation Free Algorithm for Achieving 100*ICCCN 96*, Washington, DC, Oct. 1996.

[24] N. McKeown, P. Varaiya and J. Walrand, “Scheduling Cells in an Input-Queued Switch”, *IEEE Electronics Letters*, pp. 2174-2175, Dec. 9th, 1993.

[25] V. Paxson, and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, June 1995.

[26] B. Prabhakar and N. McKeown, "On the Speedup Required for Combined Input and Output Queued Switching," *Automatica*, Vol. 35, December 1999.

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

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

[29] J. Xie and C. T. Lea, "Speedup and buffer division in input/output queuing ATM switches," *Proc. IEEE Global Telecommunications Conference*, vol. 1a, pp. 49-53, 1999.

[30] M. Yang and S. Q. Zheng, "Efficient Scheduling for CIOQ Switches with Space-division Multiplexing Speedup", *Proceedings of INFOCOM 2003.*