CHICAGO JOURNAL OF THEORETICAL COMPUTER SCIENCE
http://cjtcs.cs.uchicago.edu/

Best Effort and Priority Queuing Policies for Buffered Crossbar Switches

Alex Kesselman Kirill Kogan Michael Segal*

August 14, 2012

Abstract: The buffered crossbar switch architecture has recently gained considerable research attention. In such a switch, besides normal input and output queues, a small buffer is associated with each crosspoint. Due to the introduction of crossbar buffers, output and input contention is eliminated, and the scheduling process is greatly simplified. We analyze the performance of switch policies by means of competitive analysis, where a worst-case throughput guarantee is provided for all traffic patterns. The goal of the switch policy is to maximize the total value of packets sent out of the switch. For the case of unit length and value packets (Best Effort), we present a simple greedy switch policy that is at most 4-competitive and at least $3/2$-competitive. Moreover, we demonstrate that any online policy for this case is at least $3/2$-competitive for a special case of unit size buffers.

For the case of variable value packets, we consider the Priority Queueing (PQ) mechanism, which provides better Quality of Service (QoS) guarantees by decreasing the delay of real-time traffic. We propose a preemptive greedy switch policy with a preemption factor of β whose competitve ratio is at most $(\beta+2)^2+2/(\beta-1)$ (16.24 for $\beta=1.53$) and at least $(2\beta-1)/(\beta-1)$ (3.87 for $\beta=1.53$). The results for upper bounds hold for any value of the switch fabric *speedup*. Moreover, the presented policies incur low overhead and are amenable to efficient hardware implementation at wire speed. To the best of our knowledge, this is the first work on competitive analysis for the buffered crossbar switch architecture.

1 Introduction

The main task of a *router* is 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 output links. The switching fabric in a router is responsible for transferring packets from the input ports to

*The work on this paper has been partially supported by US Air Force European Office of Aerospace Research and Development, grant FA8655-09-1-3016, Deutsche Telecom, European project FLAVIA and Israeli Ministry of Industry, Trade and Labor (consortium CORNET).

Key words and phrases: Buffered Crossbar Switches, Control Policies, Competitive Analysis.

the output ports. 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 queueing (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 an 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 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 queueing (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. For example, for random traffic, uniformly distributed over all outputs, the throughput (i.e. the average number of packets sent in a time unit) of an IQ switch has been shown to be limited to approximately 58% of the throughput achieved by an OQ switch [18]. 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 queueing (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 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. In order to combine the advantages of both OQ and IQ switches, Combined Input-Output Queued (CIOQ) switches balance between the crossbar speedup and the complexity of scheduling algorithms. They usually have a fixed small speedup of 2, and thus need buffer space at both input and output side. This architecture has been extensively studied in the literature, see e.g. [9, 12, 13].

Most CIOQ switches use a crossbar switching fabric with a centralized scheduler. While it is theoretically possible to build crossbar schedulers that give 100% throughput [27] or rate and delay guarantees [9, 16] they are considered too complex to be practical. No commercial backbone router today can make hard guarantees on throughput, rate or delay. In practice, commercial systems use heuristics such as iSLIP [26] with insufficient speedup to give guarantees. Perhaps the most promising way of obtaining guaranteed performance has been to use maximal matching with a speedup of two in the switch fabric [12]. The parallel matching process can be characterized by three phases: request, grant, and accept. Therefore, the resolution time would be the time spent in each of the phases plus the transmission delays for the exchange of request, grant, and accept information. Unfortunately, the performance of schedulers that are based on matching computations does not scale well with the increase of the switch speedup.

A possible solution to minimize the scheduling overhead is to use buffers in the crosspoints of the crossbar fabric, or buffered crossbar. The adoption of internal buffers drastically improves the overall performance of the switch. The main benefit of the buffered crossbar switch architecture is that each input and output port can make efficient scheduling decisions independently and in parallel, eliminating the need for a centralized scheduler. As a result, the scheduler for a buffered crossbar is much simpler than that for a traditional unbuffered crossbar [10]. Note that the number of buffers is proportional to the number of crosspoints, that is $O(N^2)$. However, the crosspoint buffers are typically very small.

Buffered crossbar switches have received significant research attention. Javidi et al. [17] demonstrated that a buffered crossbar switch with no speedup can achieve 100% throughput under a uniform traffic. Nabeshima [28] introduced buffered crossbar switches with VOQs and proposed a scheme based on the Oldest Cell First (OCF) arbitration at the input as well as the crosspoint buffers. Chuang et al. [10] described

a set of scheduling algorithms to provide throughput, rate and delay guarantees with a moderate speedup of 2 and 3.

In the previous research, the scheduling policies for the buffered crossbar switch architecture were analyzed by means of simulations that assumed particular traffic distributions. However, Internet traffic is difficult to model and it does not seem to follow the traditional Poisson arrival model [30, 31]. In this work we do not assume any specific traffic model and rather analyze our policies against arbitrary traffic using competitive analysis [29, 7], which provides a uniform worst-case throughput guarantee for all traffic patterns. In competitive analysis, the online policy A is compared to the optimal clairvoyant offline policy OPT that 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 the total value of packets sent by OPT out of σ, and that of A.

1.1 Our Results

We consider a buffered crossbar switch with three levels of buffering: input, crosspoint, and output of arbitrary capacity. The switch policy controlling the switch consists 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 crosspoint buffers and from crosspoint buffers to output buffers. The goal of the switch policy is to maximize the total value of transmitted packets. When all packets have a unit value, this corresponds to the number of packets sent out of switch. When packets have variable values, this corresponds to the total value of the sent packets.

First we study the case of unit value packets, which abstracts the *Best Effort* model [11]. We introduce a simple greedy policy that is at most 4-competitive. Moreover, we show that this policy also is at least $3/2$-competitive. In addition we prove that any online policy for this problem is at least $3/2$-competitive for a special case of unit size buffers. Then we study *Priority Queueing* (PQ) buffers, where packets of the highest priority must be forwarded first. We assume that each packet has an intrinsic value designating its priority, which abstracts the *Differentiated Services* (DiffServ) model [8]. We propose a preemptive greedy switch policy with a preemption factor of β whose competitive ratio is at most $(\beta+2)^2+2/(\beta-1)$ (16.23 for $\beta = 1.53$) and at least $(2\beta-1)/(\beta-1)$ (3.87 for $\beta = 1.53$). Our results for upper bounds hold for any speedup. Moreover, the proposed policies have low implementation overhead and can operate at high speeds. We are not aware of any previous work on the competitive analysis of buffered crossbar switches.

1.2 Related Work

Kesselman et al. [19] studied preemptive policies for FIFO buffers in output-queued (OQ) switches and introduced a new bounded-delay model. Competitive analysis of preemptive and non-preemptive scheduling policies for shared memory OQ switches was given by Hahne et al. [15] and Kesselman and Mansour [23], respectively. Kesselman et al. [22] studied the throughput of local buffer management policies in a system of merge buffers.

Azar and Richter [5] presented a 4-competitive algorithm for a valued uniformly sized packets in input-queued (IQ) switch with FIFO buffers. An improved 3-competitive algorithm was given by Azar and Richter [4]. Albers and Schmidt [2] proposed a deterministic 1.89-competitive algorithm for the case of unit-value packets. Azar and Litichevskey [3] derived a 1.58-competitive algorithm for the same special case with large buffers. Recently, Albers and Jacobs [1] gave an experimental study of new and known online packet buffering algorithms.

Kesselman and Rosén [24] studied combined-input-output-queued (CIOQ) switches with FIFO buffers. 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 $8\min(k, 2\log\beta)$, where S is the speedup of the switch, k is the number of distinct packet values and β is the ratio between the largest and the smallest values. Azar and Richter [6] proposed the β-PG algorithm (Preemptive Greedy with a preemption factor of β) that is 8-competitive for an arbitrary speedup value when $\beta = 3$. Kesselman et al. [21] improved upon their result by showing that this algorithm achieves a competitive ratio of 7.5 for $\beta = 3$ and an arbitrary value of speedup. Kesselman and Rosén [25] considered the case of CIOQ switches with PQ buffers and proposed a policy that is 6-competitive for any value of speedup.

Kesselman et al. [20] considered combined-input-crossbar-output (CICOQ) switches with FIFO buffers and proposed the β-preemptive greedy algorithm that is 19.95-competitive for $\beta = 1.67$. [14] surveyed the most of the recent results in online buffer-management policies for different switch architectures and queueing models.

1.3 Paper Organization

The rest of the paper is organized as follows. The model description appears in Section 2. The cases of unit and variable value packets are analyzed in Section 3 and Section 4, respectively. We conclude with Section 5.

2 Model Description

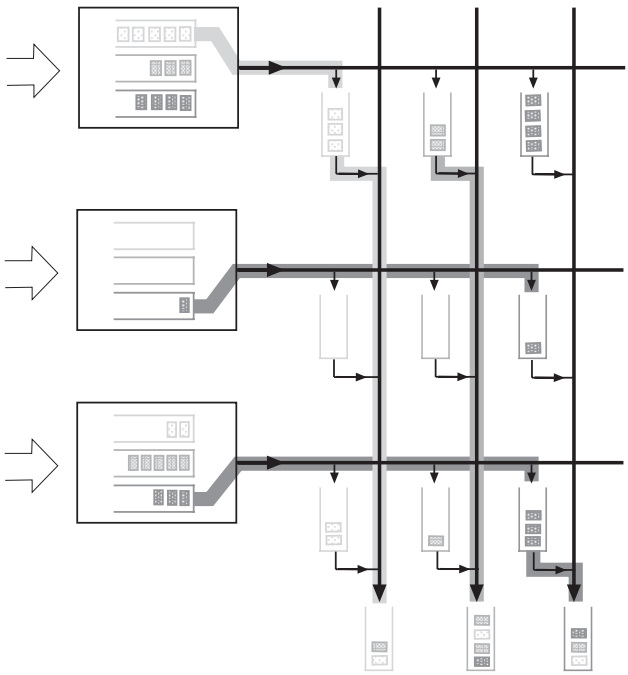

Figure 1: An example of a buffered crossbar switch.

We consider an crossbar switch with N input ports and output ports (see Fig 1). Packets, of equal length, 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. The switch has three levels of buffering: each input i maintains for each output j a separate virtual output queue $VOQ_{i,j}$ of capacity $BI_{i,j}$; each crosspoint corresponding to input i and output j maintains a queue $CQ_{i,j}$ of capacity $BC_{i,j}$; each output j maintains a queue OQ_j of capacity BO_j. We denote the length of queue q by $|q|$. Sometimes we use "*" to refer to all queue indices in range $[1,N]$.

The *buffering model* defines in which order packets should be fetched out of the buffer. We consider the First-In-First-Out (FIFO) model under which packets must leave the buffer in the order of their arrivals and the Priority Queuing (PQ) model under which packets of the highest value (priority) must be forwarded first.

We divide time into discrete steps, where a step is defined as the time between the departure of two consecutive packets. That is, during each time step more than one packet can arrive at each input port and only one packet can be sent out of 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. The second phase is the *arrival phase*, during which zero or at least one packet arrives at each input port. The third phase is the *scheduling phase*, which consists of so called *input* and *output* subphases. During the input scheduling subphase each input port may transfer one packet from a virtual output queue to the corresponding crosspoint queue. During the output scheduling subphase each output port can fetch one packet from a crosspoint queue. Notice that a packet arriving at the input port i and destined to the output port j passes through three buffers before it leaves the switch, namely $VOQ_{i,j}$, $CQ_{i,j}$ and OQ_j.

In a switch with a speedup of S, up to S packets can be removed from any input port and up to S packets can be added to each output port during the scheduling phase. This is done in (up to) S consecutive *scheduling cycles*, where each cycle comprises input and output scheduling subphases.

Suppose that the switch is managed by a policy A. We estimate the effectiveness of a switch policy by means of *competitive analysis* [7]. In competitive analysis, the online policy is compared to the optimal offline policy OPT, which knows the entire input sequence in advance. The aim of a switch policy is to maximize the total value of the packets sent out of the switch. Let σ be a sequence of packets arriving at the input ports of the switch. We denote by $V^A(\sigma)$ the total value of packets transmitted by A under the input sequence σ. The competitive ratio is defined as follows [7].

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

3 Unit Value Packets

In this section we consider the case of unit value packets. First we introduce a general lower bound on the competitive ratio of any online policy for this case. Later we define a simple *Greedy Unit Switch Policy* (see Figure 2) and analyze its lower and upper bounds. Note that *GU* never drops accepted packets and therefore implements back pressure at all buffering levels inside the switch.

Theorem 3.1. *The competitive ratio of any online algorithm A is at least* $3/2$ *for a switch with all buffers of uniform size* $B = 1$ *and a speedup* $S = 1$.

Proof. Consider the following scenario. At time slot $t = 0$, each input i receives two packets: one packet is destined to output i and the other one is destined to output $a_i \neq i$. During the next time slot $t = 1$ the online algorithm A sends one packet destined to output l_i (either i or a_i) from each input port i while OPT sends the packet to the alternative output $o_i \neq l_i$. During the next time slot $t = 2$ a packet destined to output o_i arrives at each input i. OPT accepts all these packets while A has to drop them due to lack of space. As result the competitive ratio of A is at least $3N/2N = 3/2$. □

Next we demonstrate that a lower bound of *GU* is at least $3/2$ for the buffers of uniform size B.

Theorem 3.2. *The competitive ratio of GU is at least* $3/2$ *for a switch with all buffers of uniform size* $B > 1$, *and a speedup* $S = 1$.

Greedy Unit Switch Policy (GU)

Transmission Phase : Transmit the first packet from each non-empty output queue.

Arrival Phase : Accept the arriving packet p if there is free space in the buffer. Drop p in case the buffer is full.

Scheduling Phase :

Input Subphase: for each input i choose an arbitrary head-of-line packet p if any in $VOQ_{i,j}^{GU}$ such that $CQ_{i,j}^{GU}$ is not full and transfer it to $CQ_{i,j}^{GU}$.

Output Subphase: for each output j if OQ_j^{GU} is not full choose an arbitrary head-of-line packet p if any in $CQ_{i,j}^{GU}$ and transfer it to OQ_j^{GU}.

Figure 2: GU Switch Policy for Unit Length and Value Packets.

Proof. At time slot $t = 0$ each input i receives a burst of $2B$ packets: B packets destined to output i and B packets destined to output $a_i \neq i$. During the following B time slots GU sends B packets destined to output i from each input i while OPT sends B packets destined to output a_i. At time $t = B$ a burst of B packets destined to output a_i arrives at each input i. OPT accepts all these packets while GU has to drop them due to lack of space. Therefore, the competitive ratio of GU is at least $3NB/2NB = 3/2$. □

Now we show that the GU policy is 4-competitive for any value of speedup. 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.

Definition 3.3. For a given switch policy A, a packet sent by OPT from output port j at time t is said to be *extra* if A does not transmit a packet from output port j at this time.

Next we define a wider class of so called *potential extra* packets that encompass extra packets.

Definition 3.4. For a given policy A, a packet p located at queue Q of OPT is called *potential extra* if the number of packets in Q preceding p with respect to the FIFO order is greater than the length of the corresponding queue of A.

Clearly, each extra packet should eventually become potential extra prior to transmission. We will map every potential extra packet to a packet sent by GU, in such a way that at most three potential extra packets are mapped to each GU packet. This mapping technique was first introduced in [15]. We need some auxiliary claims.

Claim 3.5. *No new potential extra packets appear during a transmission phase.*

Proof. Consider an OPT output queue OQ_j^{OPT}. If OQ_j^{OPT} is empty at the beginning of the transmission phase, then we are done. Otherwise, OPT transmits a packet out of OQ_j^{OPT} and thus the difference between $|OQ_j^{OPT}|$ and $|OQ_j^{GU}|$ cannot increase. □

Claim 3.6. *The number of potential extra packets does not increase during an arrival phase.*

Proof. Consider a virtual output queue $VOQ_{i,j}^{OPT}$. We argue that the difference between $|VOQ_{i,j}^{OPT}|$ and $|VOQ_{i,j}^{GU}|$ cannot increase unless $VOQ_{i,j}^{GU}$ is full. It follows from the fact that GU greedily accepts all arriving packets if the input buffer is not full. Obviously, $VOQ_{i,j}^{OPT}$ may not contain any potential extra packets if $VOQ_{i,j}^{GU}$ is full. □

In the following claim we bound the number of new potential extra packets that may appear during an input scheduling subphase.

Claim 3.7. *Consider an input scheduling subphase. For any input port i, the number of new potential extra packets in the virtual output queues $VOQ^{OPT}_{i,*}$ and crosspoint queues $CQ^{OPT}_{i,*}$ that appear at the end of this subphase is at most two.*

Proof. New potential extra packets may appear only in $VOQ^{OPT}_{i,j}$ if GU transfers a packet from $VOQ^{GU}_{i,j}$ and in $CQ^{OPT}_{i,k}$ if OPT transfers a packet to this queue provided that $j \neq k$. Thus, at most two new potential extra packets may occur. □

The next claim limits the number of new potential extra packets that may occur during an output scheduling subphase.

Claim 3.8. *Consider an output scheduling subphase. For any output port j, the number of new potential extra packets in the crosspoint queues $CQ^{OPT}_{*,j}$ and output queue OQ^{OPT}_{j} that appear at the end of this subphase is at most one.*

Proof. Consider an output scheduling subphase t_{os}. We have that one new potential extra packet may appear in $CQ^{OPT}_{i,j}$ if GU transfers a packet from $CQ^{GU}_{i,j}$. Notice that if the number of potential extra packets in OQ^{OPT}_{j} increases, then all crossbar queues $CQ^{GU}_{*,j}$ must have been empty at the beginning of t_{os}. In this case, the potential extra packet appearing in OQ^{OPT}_{j} must have already been a potential extra packet in a queue $CQ^{OPT}_{i,j}$ at the beginning of t_{os}. That establishes the claim. □

The mapping routine presented in Figure 3 maps all potential extra packets to the packets sent by GU (we will show in the sequel that the routine is feasible). The routine runs at each (sub)phase, and adds some mappings according to the actions of GU and OPT.

Mapping Routine:

- *Step 1: Arrival Phase*. For each $VOQ^{OPT}_{i,j}$, if OPT accepts a packet that becomes potential extra, this packet replaces in the mapping the preceding packet in $VOQ^{OPT}_{i,j}$ that ceases to be potential extra.
- *Step 2: Scheduling Phase*. (The next sub-steps are repeated S times for each scheduling cycle.)
  For each input port i, map new potential extra packet(s) in the virtual output queues $VOQ^{OPT}_{i,*}$ and crosspoint queues $CQ^{OPT}_{i,*}$ to the packet transferred by GU from input port i.
  - *Sub-Step 2.2: Output Scheduling Subphase*. For each output port j, map new potential extra packet in the crosspoint queues $CQ^{OPT}_{*,j}$ and output queue OQ^{OPT}_{j} to the packet transferred by GU to output port j.

Figure 3: Mapping Routine for the GU policy.

We make the following observation concerning potential extra packets.

Observation 3.9. All potential extra packets are mapped by the mapping routine.

The observation is due to the fact that the routine runs during all but the transmission phase. Notice that by Claim 3.5, no new potential extra packets appear during a transmission phase. The next lemma shows that the mapping routine is feasible and at most three potential extra packets are mapped to a packet transmitted out of the switch by GU.

Lemma 3.10. *The mapping routine is feasible and no GU packet is mapped more than three times by the mapping routine prior to transmission out of the switch for any value of the speedup S.*

Proof. According to Claim 3.6, the number of potential extra packets in the virtual output queues does not increase during the arrival phase. Therefore, if a new potential extra packet appears in $VOQ_{i,j}^{OPT}$, it must be the case that another packet in $VOQ_{i,j}^{OPT}$ ceases to be potential extra at the end of the arrival phase. Hence, Step 1 of the mapping routine is feasible and no new mappings are added to *GU* packets.

Claim 3.7 implies that the number of new potential extra packets in the virtual output queues and crosspoint queues corresponding to an input port that appear at the end of the input scheduling subphase is at most two. We argue that no new potential extra packet may appear if *GU* does not transmit a packet from this input port. In this case, the potential extra packet appearing in a crosspoint queue of *OPT* must have already been potential extra at the beginning of the input scheduling subphase under consideration. Thus, Step 2.1 of the mapping routine is feasible and no *GU* packet is mapped more than twice.

By Claim 3.8, the number of new potential extra packets in the crosspoint queues and output queue corresponding to an output port that appear at the end of the output scheduling subphase is at most one. We claim that no new potential extra packet may appear if *GU* does not transfer a packet to this output port. In this case, the potential extra packet appearing in the output queue of *OPT* must have already been potential extra at the beginning of the output scheduling subphase under consideration. Therefore, Step 2.2 of the mapping routine is feasible and no *GU* packet is mapped more than once.

Note the *GU* does not drop packets that have been admitted to the switch. Therefore, the mapping is persistent and all mapped *GU* packets are eventually sent out of the switch. Furthermore, no *GU* packet is mapped more than three times in total. □

Now we will show that *GU* achieves a competitive ratio of 4.

Theorem 3.11. *The competitive ratio of GU is at most* 4 *for any speedup value.*

Proof. Fix an input sequence σ. Evidently, the number of packets sent by *OPT* is bounded by the number of packets sent by *GU* plus the number of extra packets. Observe that every extra packet at first becomes a potential extra packet prior to transmission. By Lemma 3.10, the number of extra packets is bounded by three times the number of packets transmitted by *GU*. In this way we obtain, $V^{OPT}(\sigma) \leq 4V^{GU}(\sigma)$. □

4 Variable Value Packets

In this section we study the case of variable value packets under the Priority Queueing buffering model. Remember that packets of the highest value have a strict priority over packets with lower values and are always forwarded first. The goal of the switch policy is to maximize the total value of packets that cross the switch.

We define the β-*Preemptive Greedy Variable Switch Policy* (see Figure 4). The rationale behind β-*PGV* is that it preempts packets inside the switch only to serve significantly more valuable packets (β times the value of the evicted packet). Although the Priority Queueing mechanism may violate the global FIFO order, it still maintains the FIFO order within each individual flow consisting of packets with the same value.

First we demonstrate two different lower bounds on the performance of the β-*PGV* policy.

Theorem 4.1. *The competitive ratio of* β-*PGV is at least* $(2\beta+1)/(\beta+2)$ *for* $B = N = S$.

Proof. During the time $t = 0,1$ the input port 1 receives a burst of B packets of value 1 destined to output port 1. At time $t = 2,3$ the input port 1 receives a burst of B packets of value $\beta - \varepsilon$ destined to output port 1. The policy β-*PGV* drops all but three packets from the last burst received at time $t = 3$. The total value obtained by β-*PGV* is $2N * 1 + (N+3) * (\beta - \varepsilon)$. On the other hand, *OPT* drops all but three packets from the second burst received at time $t = 1$ gaining the value of $(N+3) * 1 + 2N * (\beta - \varepsilon)$. Hence, the competitive ratio of β-*PGV* is at least $(2\beta+1)/(\beta+2)$ for sufficiently large N. □

β-Preemptive Greedy Variable Switch Policy (β-PGV)

Transmission Phase : For each non-empty output queue, transmit the first packet in the FIFO order with the *largest value*.

Arrival Phase : Accept an arriving packet p if there is a free space in the corresponding virtual output queue $VOQ_{i,j}^{PGV}$. Drop p if $VOQ_{i,j}^{PGV}$ is full and $V(p)$ is less than the minimal value among the packets currently in $VOQ_{i,j}^{PGV}$. Otherwise, drop from $VOQ_{i,j}^{PGV}$ a packet p' with the minimal value and accept p. We say that p preempts p'.

Scheduling Phase :

Input Subphase: For each input port i do the following. For each virtual output queue $VOQ_{i,j}^{PGV}$, choose the packet p that is the first packet in the FIFO order among the packets with the largest value if any. If $CQ_{i,j}^{PGV}$ is not full, mark p as eligible. Otherwise, consider a packet p' with the smallest value in $CQ_{i,j}^{PGV}$. If $V(p) \geq \beta * V(p')$, then mark p as eligible (p will preempt p' if selected for transmission). Among the eligible packets with the largest value in $VOQ_{i,*}^{PGV}$, select an arbitrary packet p'' and transfer p'' to the corresponding crosspoint queue while preempting a packet with the smallest value from that queue if necessary.

Output Subphase: For each output port j do the following. For each crosspoint queue $CQ_{i,j}^{PGV}$, choose the packet that is the first packet in the FIFO order among the packets with the largest value if any. Among all chosen packets in $CQ_{*,j}^{PGV}$, select an arbitrary packet p with the largest value. If OQ_j^{PGV} is not full, then transfer p to OQ_j^{PGV}. Otherwise, consider a packet p' with the smallest value in OQ_j^{PGV}. If $V(p) \geq \beta * V(p')$, then preempt p' and transfer p to OQ_j^{PGV}.

Figure 4: β-PGV Switch Policy for Priority Queuing Model.

Theorem 4.2. *The competitive ratio of β-PGV is at least $(2\beta-1)/(\beta-1)$ for $B = N = S$ and $\beta > 1$.*

Proof. During the time slot $t = i$ the input i receives a burst of B packets of value β^i for $0 \leq i < N$ destined to output port 1. At time slot $0 < t < B$, β-*PGV* preempts $B-1$ packets of value β^{i-1} from output buffer BO_1 and transmits one such packet. The total value that is obtained by β-*PGV* is $N * \beta^{(N-1)} + (\beta^N - 1)/(\beta - 1)$. *OPT* buffers all packets at the crosspoint buffers and transmits all of them gaining the value of $N * \beta^{(N-1)} + N * (\beta^N - 1)/(\beta - 1)$. Hence, the competitive ratio of β-*PGV* is at least $(2\beta-1)/(\beta-1)$ for sufficiently large N. □

Now we show that β-*PGV* achieves a competitive ratio of $(\beta+2)^2 + 2/(\beta-1)$ for any speedup. In order to show the competitive ratio of β-*PGV* we will assign values to the packets sent by β-*PGV* so that no packet is assigned more than $(\beta+2)^2 + 2/(\beta-1)$ times its value, and then show that the value assigned is indeed at least $V^{OPT}(\sigma)$. Our analysis is done along the lines of the work in [25], which studies Priority Queuing (PQ) buffers for CIOQ switches.

For the analysis, we assume that *OPT* maintains *FIFO* order and never preempts packets. Notice that any optimal schedule can be transformed into a non-preemptive *FIFO* schedule of the same value.

Lemma 4.3. *There is OPT algorithm that maintains FIFO order and never preempts packets.*

Proof. The proof is similar to that for CIOQ switches [24]. We argue that any feasible schedule of optimal algorithm 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 the optimal algorithm in the non-FIFO model never preempts packets at the inputs or drops packets at the crosspoints and 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 none-FIFO schedule of the optimal algorithm

- **Step 1** Assign to each packet scheduled by β-*PGV* during the input scheduling subphases of t_s *once* its own value; assign to each packet scheduled by β-*PGV* during the output scheduling subphases of t_s β times own its value.

For each input port i, let p' be the packet scheduled by *OPT* from $VOQ_{i,j}^{OPT}$ if any during the input scheduling subphase of t_s. Let p be the first packet with the largest value in $VOQ_{i,j}^{PGV}$ if any or a dummy packet with zero value otherwise.

- **Step 2** If $V(p') \leq V(p)$ and p is not eligible for transmission, then proceed as follows. Consider the beginning of the output scheduling subphase that takes place during a scheduling cycle t_s' when *OPT* schedules p' from $CQ_{i,j}^{OPT}$ and let p'' be the first packet with the largest value in $CQ_{i,j}^{PGV}$ if any or a dummy packet with zero value otherwise.
  - **Sub-Step 2.1** If $V(p'') \geq V(p)/\beta$ and p'' is not eligible for transmission at the beginning of the output scheduling subphase of t_s', let $\hat{p}$ be the packet that will be sent out of OQ_j^{PGV} at the same time at which *OPT* will send p' from OQ_j^{OPT} (we will later show that $\hat{p}$ exists and its value is at least $V(p')/\beta^2$)). Assign the value of p' to $\hat{p}$.
  - **Sub-Step 2.2** If $V(p'') < V(p)/\beta$, consider the set of packets with value at least $V(p')/\beta$ that are scheduled by *PGV* from $CQ_{i,j}^{PGV}$ prior to t_s'. Assign the value of $V(p')$ to a packet in this set that has not previously been assigned any value by Sub-Step 2.2 (we will later show that such a packet exists).
- **Step 3** If $V(p') > V(p)$ then proceed as follows:
  - **Sub-Step 3.1** If p' was already scheduled by *PGV*, then assign the value of $V(p')$ to p'.
  - **Sub-Step 3.2** Otherwise, consider the set of packets with value at least $V(p')$ that are scheduled by *PGV* from $VOQ_{i,j}^{PGV}$ prior to the scheduling cycle 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 this subphase, and has not previously been assigned any value by either Sub-Step 3.1 or Sub-Step 3.2 (we will later show that such a packet exists).
- **Step 4** If a packet q preempts a packet q' at a crosspoint or output queue of *PGV*, re-assign to q the value that was or will be assigned to q'.

Figure 5: Assignment Routine for β-*PGV* policy - executed for every scheduling cycle t_s.

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. The lemma follows. □

The assignment routine presented on Figure 5 specifies how to assign values to the packets sent by β-*PGV*. Observe that the routine assigns some value only to packets that are scheduled out of the virtual output queues and crosspoint queues. Furthermore, if a packet is preempted, then the total value assigned to it is re-assigned to the packet that preempts it.

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

Lemma 4.4. *The assignment routine is feasible and the value of each packet scheduled by OPT is assigned to a PGV packet. The result holds for any value of the speedup.*

Proof. First we show that the assignment routine as defined is feasible. Step 1, Sub-Step 3.1 and Step 4 are trivially feasible. Consider Sub-Steps 2.1, 2.2 and 3.2.

Sub-Step 2.1. Let p'' be the first packet with the largest value in $CQ_{i,j}^{PGV}$ at the beginning of the output scheduling subphase of t_s' and suppose that p'' is not eligible for transmission. If $V(p'') \geq V(p)/\beta$ then,

by the definition of PGV, the minimal value among the packets in OQ_j^{PGV} is at least $V(p'')/\beta \geq V(p)/\beta^2$ and OQ_j^{PGV} is full. Thus, during the following BO_j time steps PGV will send packets with value of at least $V(p')/\beta^2$ out of OQ_j^{PGV}. The packet p' scheduled by OPT from $VOQ_{i,j}^{OPT}$ will be sent from OQ_j^{OPT} in one of these time steps (recall that by our assumption OPT maintains FIFO order). Since $V(p') \leq V(p)$, we have that the packet $\hat{p}$ of PGV as specified in Step 2.1 indeed exists, and its value is at least $V(p')/\beta^2$.

Sub-Step 2.2. If $V(p'') < V(p)/\beta$, then evidently PGV scheduled at least $BC_{i,j}$ packets with value at least $V(p')/\beta$ out of $CQ_{i,j}^{PGV}$ during $[t_s, t_s')$. By the construction, at most $BC_{i,j}-1$ of these packets have been assigned some value by Sub-Step 2.2. That is due to the fact that p' is still present in $CQ_{i,j}^{OPT}$ at the beginning of the output scheduling subphase of t_s' and by our assumption OPT maintains FIFO order. Henceforth, one of these packets must be *available* for assignment, i.e., it has not been assigned any value by Sub-Step 2.2 prior to t_s'.

Sub-Step 3.2. First note that if this case applies, then the packet p' (scheduled by OPT from $VOQ_{i,j}^{OPT}$ during the input scheduling subphase of t_s) is dropped by PGV from $VOQ_{i,j}^{PGV}$ during the arrival phase $t_a < t_s$. Let $t_a' \geq t_a$ be the last arrival phase before t_s at which a packet of value at least $V(p')$ is dropped from $VOQ_{i,j}^{PGV}$. Since the greedy buffer management policy is applied to $VOQ_{i,j}^{PGV}$, it contains $BI_{i,j}$ packets with value of at least $V(p')$ at the end of t_a'. Let P be the set of these packets. Note that $p' \notin P$ because it has been already dropped by PGV by this time. We have that in $[t_a', t_s)$, PGV has actually scheduled all packets from P, since in $[t_a', t_s)$ no packet of value at least $V(p')$ has been dropped, and at time t_s all packets in $VOQ_{i,j}^{PGV}$ have values less than $V(p')$. We show that at least one packet from P is *available* for assignment, i.e., it has not been assigned any value by Step 3 prior to t_s and is not currently present in $VOQ_{i,j}^{OPT}$. Let x be the number of packets from P that are present in $VOQ_{i,j}^{OPT}$ at the end of the scheduling cycle t_s. By the construction, these x packets are unavailable for assignment. From the rest of the packets in P, a packet is considered available for assignment 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 $[t_a', t_s)$ (when it is scheduled). We now argue that OPT has scheduled at most $BI_{i,j}-1-x$ packets out of $VOQ_{i,j}^{OPT}$ in $[t_a', t_s)$, 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 the beginning of the scheduling cycle t_s, were already present in $VOQ_{i,j}^{OPT}$ at the end of the arrival phase t_a'. The same applies to the 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 $[t_a', t_s)$ were also present in $VOQ_{i,j}^{OPT}$ at the end of the arrival phase t_a'. 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 for assignment 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 some value by Step 3.

We show that the value of each packet that is scheduled by OPT is assigned to a PGV packet. Note that the assignment routine handles all packets scheduled by OPT out of the virtual output queues. The only two cases left uncovered by Step 2 and Step 3 of the assignment routine are (i) $V(p') \leq V(p)$ and p is eligible for transmission and (ii) $V(p') \leq V(p)$, p is not eligible for transmission, $V(p'') \geq V(p)/\beta$ and p'' is eligible for transmission. We show that these cases are covered by Step 1: for the case (i), the value of p' is assigned during the input scheduling subphase when p is scheduled since $V(p) \geq V(p')$; for the case (ii), the value of p' is assigned during the output scheduling subphase when p'' is scheduled since $V(p'') \geq V(p)/\beta$. If a PGV packet is preempted, the value assigned to it is re-assigned to the preempting packet by Step 4. □

Lemma 4.5. *No PGV packet is assigned more than* $(\beta+2)^2 + 2/(\beta-1)$, $\beta > 1$ *times its value. The result holds for any value of the speedup.*

Proof. Consider a packet p sent by PGV. Observe that p can be assigned at most $1+\beta$ times own its value by Step 1 and at most $\beta+\beta^2$ times its own value by Step 2. By the specification of Sub-Step 3.2, it 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 now show that Sub-Step 3.1 does not assign any value to p if it is assigned a value by 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 this sub-step during t_s, then p is not present 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 once its own value by Step 3. Hence, a packet that does not perform preemptions can be assigned at most $2+2\beta+\beta^2$ times its value.

Next we analyze Step 4. We say that p *transitively* preempts a packet q if either p directly preempts q or p preempts another packet that transitively preempts q. Firstly, p can preempt another packet q' in the crosspoint queue such that $V(q') \leq V(p)/\beta$. Observe that any preempted packet in a crosspoint queue is assigned at most once its own value by Step 1, once its own value by Step 3 and no value by Step 2. Hence, the total value that can be assigned to p by Step 4 due to transitively preempted packets when p preempts q' is bounded by twice its own value. Secondly, p can preempt another packet q'' in the output queue such that $V(q'') \leq V(p)/\beta$. Observe that any preempted packet in an output queue is assigned at most $1+\beta$ times its own value by Step 1, β times own its value by Step 2, once its own value by Step 3, and $2/(\beta-1)$ times own its value by Step 4. Thus, the total value that can be assigned to p by Step 4 due to transitively preempted packets when p preempts q'' is bounded by $2+2\beta+2/(\beta-1)$ times its own value. Therefore, in total no PGV packet is assigned more than $(\beta+2)^2+2/(\beta-1)$ times its own value. □

The main theorem follows directly from Lemma 4.4 and Lemma 4.5.

Theorem 4.6. *The competitive ratio of the PGV policy is at most* $(\beta+2)^2+2/(\beta-1)$, $\beta > 1$ *for any speedup.*

5 Conclusions

As switch speeds constantly grow, centralized switch scheduling algorithms become the main performance bottleneck. In this paper we consider competitive switch policies for buffered crossbars switches with PQ buffers. The major advantage of the buffered crossbar switch architecture is that the need for centralized arbitration is eliminated and scheduling decisions can be made independently by the input and output ports.

Our main result is the preemptive greedy switch policy with a preemption factor β. We show that its competitive ratio is at most $(\beta+2)^2+2/(\beta-1)$ (16.24 for $\beta = 1.53$) and at least $(2\beta-1)/(\beta-1)$ (3.87 for $\beta = 1.53$) for the general case of unit length and variable value packets.

We also propose a simple greedy switch policy that achieves a competitive ratio of at most 4 and at least $3/2$ in the case of unit length and value packets. The results for upper bounds hold for any value of switch fabric speedup. As far as we know, these are the first results on competitive analysis for the buffered crossbar switch architecture. We believe that this work advances the design of practical switch policies with provable worst-case performance guarantees for state-of-the-art switch architectures.

References

[1] S. Albers and T. Jacobs, "An Experimental Study of New and Known Online Packet Buffering Algorithms", *Algorithmica*, Vol 57, Issue 4, pp. 754-765, 2010. 3

[2] S. Albers and M. Schmidt, "On the Performance of Greedy Algorithms in Packet Buffering", *SIAM Journal on Computing*, Vol. 35, No. 2, pp. 278-304, 2005. 3

[3] Y. Azar and M. Litichevskey, "Maximizing Throughput in Multi-queue Switches", *Algorithmica*, Vol. 45, No. 1, pp. 69-90, 2006. 3

[4] Y. Azar and Y. Richter, "The Zero-one Principle for Switching Networks", *Proc. STOC*, 2004, pp. 64-71. 3

[5] Y. Azar and Y. Richter, "Management of Multi-Queue Switches in QoS Networks", *Algorithmica*, Vol. 43, No. 1-2, pp. 81-96, 2005. 3

[6] Y. Azar and Y. Richter, "An Improved Algorithm for CIOQ switches", *ACM Transactions on Algorithms*, Vol. 2, No. 2, pp. 282-295, 2006. 4

[7] A. Borodin and R. El-Yaniv, "Online Computation and Competitive Analysis", *Cambridge University Press*, 1998. 3, 5

[8] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services", *Internet RFC 2475*, December 1998. 3

[9] 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, December 1999. 2

[10] S.T. Chuang, S. Iyer, N. McKeown, "Practical Algorithms for Performance Guarantees in Buffered Crossbars", *Proc. INFOCOM 2005*, Vol. 2, pp. 981-991. 2

[11] D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service", *IEEE/ACM Trans. on Networking*, Vol. 6, No. 4, pp. 362-373, August 1998. 3

[12] J. Dai, and B. Prabhakar, "The Throughput of Data Switches with and without Speedup", *Proc. IEEE INFOCOM 2000*, Vol. 2, pp. 556-564, March 2000. 2

[13] P. Giaccone, E. Leonardi, B. Prabhakar and D. Shah, "Delay Performance of High-speed Packet Switches with Low Speedup", *Proc. IEEE GLOBECOM 2002*, Vol. 3, pp. 2629-2633, November 2002. 2

[14] Michael H. Goldwasser, "A Survey of Buffer Management Policies for Packet Switches", *SIGACT News*, Vol. 41, pp. 100-128, 2010. 4

[15] E. L. Hahne, A. Kesselman and Y. Mansour, "Competitive Buffer Management for Shared-Memory Switches", *Proc. SPAA*, pp. 53-58, July 2001. 3, 6

[16] S. Iyer, R. Zhang, and N. McKeown, "Routers with a Single Stage of Buffering", *ACM SIGCOMM*, Vol. 3, No. 4, pp. 251-264, September 2002. 2

[17] T. Javidi, R Magill, and T. Hrabik, "A High Throughput Scheduling Algorithm for a Buffered Crossbar Switch Fabric", *Proc. IEEE International Conference on Communications*, Vol. 5, pp. 1586-1591, 2001. 2

[18] M. Karol, M. Hluchyj and S. Morgan, "Input versus Output Queuing an a Space Division Switch", *IEEE Trans. Communications*, Vol. 35, Issue 12, pp. 1347-1356, 1987. 2

[19] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber and M. Sviridenko, "Buffer Overflow Management in QoS Switches", *SIAM Journal on Computing*, Vol. 33, Issue 3, pp. 563-583, 2004. 3

[20] A. Kesselman, K. Kogan and M. Segal, "Packet mode and QoS algorithms for buffered crossbar switches with FIFO queuing", *Distributed Computing*, Vol. 23, Issue 3, pp. 163-175, 2010. 4

[21] A. Kesselman, K. Kogan and M. Segal, "Improved Competitive Performance Bounds for CIOQ Switches", *Algorithmica*, Vol. 63, Issue 1-2, pp. 411-424, 2012. 4

[22] A. Kesselman, Z. Lotker, Y. Mansour and B. Patt-Shamir, "Buffer Overflows of Merging Streams," *Proc. ESA*, pp. 244-245, 2003 3

[23] 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, Issue 2-3, pp. 161-182, 2004. 3

[24] A. Kesselman and A. Rosén, "Scheduling Policies for CIOQ Switches," *Journal of Algorithms*, Vol. 60, No. 1, pp. 60-83, 2006. 3, 9

[25] A. Kesselman and A. Rosén, "Controlling CIOQ Switches with Priority Queuing and in Multistage Interconnection Networks," *Journal of Interconnection Networks*, 9(1/2), pp. 53-72, 2008. 4, 9

[26] N. McKeown, "iSLIP: A Scheduling Algorithm for Input-Queued Switches", *IEEE Transactions on Networking*, Vol. 7, No. 2, pp. 188-201, April 1999 2

[27] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch", *IEEE Transactions on Communications*, Vol. 47, No. 8, pp. 1260-1267, August 1999. 2

[28] M. Nabeshima, "Performance Evaluation of Combined Input-and-crosspoint- queued Switch", *IEICE Trans. Commun.*, Vol. E83-B, No. 3, pp. 737-741, March 2000. 2

[29] D. Sleator and R. Tarjan, "Amortized Efficiency of List Update and Paging Rules", *Communications of the ACM*, Vol. 28, No. 2, pp. 202-208, Feb. 1985. 3

[30] V. Paxson, and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling", *IEEE/ACM Transactions on Networking*, Vol. 3, No. 3, pp. 226-244, June 1995. 3

[31] A. Veres and M. Boda, "The Chaotic Nature of TCP Congestion Control", *Proc. INFOCOM*, Vol. 3, pp. 1715-1723, March 2000. 3

AUTHORS

Alex Kesselman
Google Inc., USA
alx@google.com

Kirill Kogan
Ben-Gurion University of the Negev, Israel
kirill.kogan@gmail.com

Michael Segal
Professor
Communication Systems Engineering Department
Ben Gurion University of the Negev, Beer-Sheva, Israel
segal@cse.bgu.ac.il