Noname manuscript No.
(will be inserted by the editor)

Packet Mode and QoS Algorithms for Buffered Crossbar Switches with FIFO Queuing

Alex Kesselman Kirill Kogan Michael Segal

Received: date / Accepted: date

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 dependency is eliminated, and the scheduling process is greatly simplified. We analyze the performance of switch policies by means of competitive analysis, where a uniform guarantee is provided for all traffic patterns. We assume that each packet has an intrinsic value designating its priority and the goal of the switch policy is to maximize the weighted throughput of the switch. We consider FIFO queueing buffering policies, which are deployed by the majority of today's Internet routers. In *packet-mode* scheduling, a packet is divided into a number of unit length cells and the scheduling policy is constrained to schedule all the cells *contiguously*, which removes reassembly overhead and improves Quality-of-Service (QoS). For the case of variable length packets with uniform value density (Best Effort model), where the packet value is proportional to its size, we present a packet-mode greedy switch policy that is 7-competitive. For the case of unit size packets with variable values (Differentiated Services model), we propose a β-preemptive (β is a preemption factor) greedy switch policy that achieves a competitive ratio of $6+4\beta+\beta^2+3/(\beta-1)$. In particular, its competitive ratio is at most 19.95 for the preemption factor of $\beta = 1.67$. As far as we know, this is the first constant-competitive FIFO policy for this architecture in the case of variable value packets. In addition, we evaluate performance of β-preemptive

The preliminary version of this paper has been appeared in ACM Symposium on Principles of Distributed Computing, PODC 2008 where a preemption factor of 2 is considered. Here, we express the competitive ratio as function of the preemption factor β, thus improving upon the *PODC* result from 21 to 19.95 when beta is optimized.

A. Kesselman
Google Inc. E-mail: alx@google.com

K. Kogan
Cisco Systems and the Department of Communication Systems Engineering, Ben-Gurion University, Israel E-mail: kkogan@cisco.com

M. Segal
Department of Communication Systems Engineering, Ben-Gurion University, Israel E-mail: segal@cse.bgu.ac.il

greedy switch policy by simulations and show that it outperforms other natural switch policies. The presented policies are simple and thus can be efficiently implemented at high speeds. Moreover, our results hold for any value of the internal switch fabric *speedup*.

Keywords Buffer Management · Competitive Analysis · Switch

1 Introduction

The main tasks of a router are to receive a packet from the input port, to find its destination port using a routing table, to transfer the packet to that output port, and finally to transmit it on the output link. The switching fabric in a router is responsible for transferring packets from the input ports to the output ports. If a burst of packets destined to the same output port arrives, it is impossible to transmit all the packets immediately in case of contention, and some of them must be buffered inside the switch (or dropped).

In this paper we focus on buffered crossbar switch architecture. Essentially, this is well-studied *Combined-Input-Output-Queued* (CIOQ) switch [8,12,14,15,27] with additional buffers at the crosspoints. In a nutshell, a packet arriving at input-port i and destined for output-port j is first buffered at input-port i's buffer, then it is sent to an (i, j) crosspoint buffer and finally, it is forwarded from the crosspoint buffer to output-port j's buffer. An important characteristic of the switch fabric is its *speedup*. A switch has a speedup S, if the switching fabric runs S times faster than each of the input or the output lines. The switch policy obtains the value of the packet if it is transmitted out of the output port and gains no value otherwise (if the packet is dropped).

Buffered crossbar switches recently received significant research attention [11,18,30, 32,34]. The overall performance of the switch can be significantly improved by using of internal buffers at crosspoints. It allows each input and output port to schedule packets independently and in parallel fashion. As a result, the scheduler for a buffered crossbar is much simpler than that for a traditional unbuffered crossbar [11]. Note that the number of buffers is proportional to the number of crosspoints, that is $O(N^2)$.

In many network protocols traffic is comprised of variable length packets. A prime example is provided by IP datagrams whose sizes typically vary from 40 to 1500 bytes [1]. Real-life switches, however, operate with fixed-size cells, which are easier to buffer and schedule synchronously in an electronic domain. Transmitting packets over cell-based switches requires the use of packet segmentation and reassembly modules, resulting in a significant computation and communication overhead [20]. Cell-based scheduling has a number of drawbacks. First, one must keep a packet-reassembly buffer at each output port. Second, traditional cell-based schedulers are typically unaware of the existence of packets, and thus different cells of the same packet may experience different delays. This may badly affect *Quality-of-Service* (QoS) because the actual delay of a packet is the delay of the last cell.

Katevenis et al. [21] proposed the use of buffered crossbars for variable length packets. Kanizo et al. [19] discussed the design of buffered crossbar switch, where each crosspoint buffer can hold multiple cells. The crosspoint buffer size is expected to keep growing with SRAM density, roughly doubling every 2.5 years [36]. Turner [34] studied the performance of asynchronous work-conserving schedulers that preserve the packet arrival order (while forwarding). Turner showed that for a speedup of two and the

crosspoint buffer size of at least twice the maximum packet size a buffered crossbar scheduler can provide work conservation guarantees. These schedulers can also emulate an OQ switch within a class of restricted queueing disciplines.

The first problem that we investigate in this work is related to *packet-mode* scheduling, where the whole packet rather than a single cell becomes the switching unit [29]. Packet mode scheduling has received a lot of research attention in recent time [5,34]. A packet is divided into a number of unit length cells and the scheduling policy is constrained to schedule all cells of a given packet *contiguously*. The value of a packet equals the number of cells it is comprised of. This is so called *uniform value density* case corresponding to the Best Effort model [13]. The main advantage of this architecture is that the reassembly overhead is reduced. In addition, a packet-based scheduler, which is aware of the packet entity, may use this information to provide better performance through scheduling.

The second problem that we explore in this work is related to QoS guarantees. We assume that all packets have equal size and each packet has an intrinsic value designating its priority, which corresponds to the *DiffServ model* [9]. We study switch policies that maintain the *First-In-First-Out* (FIFO) order between the packets. FIFO buffering policies are deployed by most of the state-of-the-art Internet routers. Furthermore, such policies are beneficial for the dominant TCP transport protocol, whose performance degrades if packet re-ordering occurs.

Internet traffic is difficult to model and it does not seem to follow the traditional Poisson arrival model [31,35]. In this work we do not assume any specific traffic model and rather analyze our switch policies against arbitrary traffic using competitive analysis [33,10], which provides a uniform throughput guarantee for all traffic patterns. In competitive analysis, the online switch policy A is compared to the optimal offline policy OPT, which knows the entire input sequence in advance. The *competitive ratio* of a policy A is the maximum, over all sequences of packet arrivals σ, of the ratio between the the value of packets sent by OPT out of σ, and the value of packets sent by A out of σ.

We consider a buffered crossbar switch with crosspoint buffers 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 to crosspoint buffers and from crosspoint buffers to output buffers. The goal of the switch policy is to maximize the weighted throughput of the switch.

1.1 Our Results

First, we consider packet-mode policies for the case of variable length packets with uniform value density (Best Effort model) assuming that each internal buffer (virtual output, crosspoint or output) has capacity of at least two packets of maximal size. We present a simple greedy policy that is 7-competitive. Then we study the case of unit size packets with variable values (Differentiated Services model). Our main result is a preemptive greedy switch policy β-PGV with preemption factor β that achieves a competitive ratio of $6 + 4\beta + \beta^2 + 3/(\beta - 1)$. In particular, the competitive ratio of the β-PGV policy is at most 19.95 for the preemption factor of $\beta = 1.67$ Remarkably, our results hold for any value of the *fabric speedup*. To the best of our knowledge, this is the first constant-competitive FIFO policy for buffered crossbar

switches in the general case of variable value packets and arbitrary speedup. In addition, we estimate the throughput of β-PGV by means of simulations. On average β-PGV outperforms some other natural switch policies and performs much better than its worst-case performance guarantee. The proposed policies are simple and thus amenable to efficient implementation at high speeds.

1.2 Related Work

This paper is mostly related to the work of Kesselman et al. [23], which studies buffered crossbar. For the case of unit size and unit value packets the authors provide 4-competitive policy. Kesselman et al. [23] also propose a 18-competitive policy for the case of variable value packets with Priority Queuing (PQ) buffers, where packets of the highest value are sent first. Guez et al. [16] consider packet-mode scheduling for CIOQ switches.

Kesselman et al. [26] studied preemptive policies for FIFO buffers in OQ switches and introduce 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. [17] and Kesselman and Mansour[24], respectively. Aiello et al. [2] considered the throughput of various protocols in a setting of a network of OQ switches with limited buffer space. Kesselman et al. [25] studied the throughput of local buffer management policies in a system of merge buffers.

Azar and Richter [7] presented a 4-competitive algorithm for a weighted multi-queue switch problem with FIFO buffers. An improved 3-competitive algorithm was given by Azar and Richter [8]. Albers and Schmidt [4] proposed a deterministic 1.89-competitive algorithm for the case of unit-value packets. In a recent paper, Azar and Litichevskey [6] derived a 1.58-competitive algorithm for switches with large buffers. Albers and Jacobs [3] gave an experimental study of new and known online packet buffering algorithms.

Kesselman and Rosén [27] studied 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 a competitive ratio 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 value. Azar and Richter [8] obtained a 8-competitive policy for CIOQ switches with FIFO buffers for the latter case. Kesselman and Rosén [28] considered the case of CIOQ switches with PQ buffers and proposed a policy that is 6-competitive for any speedup.

1.3 Paper Organization

The rest of the paper is organized as follows. The model description appears in Section 2. Section 3 describes the switch policies, which are analyzed in Section 4 and simulated in Section 5. We conclude with Section 6.

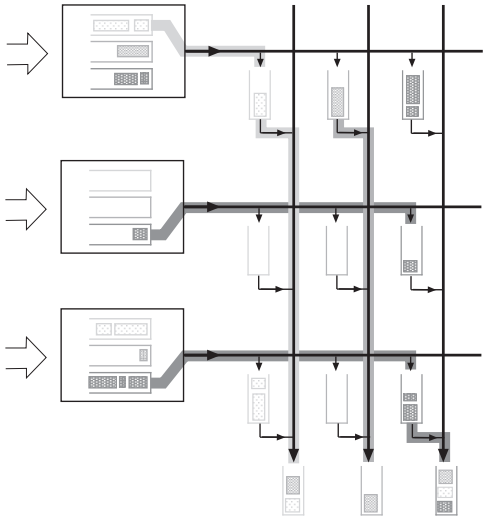

Fig. 1 An example of a buffered crossbar switch.

2 Model Description

We consider an $N \times N$ buffered crossbar switch (see Figure 1). Packets arrive at input ports, and each packet is labeled with the output port on which it has to leave the switch. A packet is divided into a number of unit length cells that must be transmitted contiguously out of the switch. For a packet p, we denote by $l(p)$ the length of p measured in cells and by $V(p)$ the value of p. We denote by l_{max} the maximum packet length. We study the case of *variable length packets*, where the value of a packet equals its length and the case of *variable value packets*, where all packets have a unit length. Packets must be transmitted out of each queue in the First In First Out (FIFO) order.

The switch has three levels of buffering: each input i maintains for each output j a separate 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. The capacity of a queue is measured in cells. In this paper we consider a case when any virtual output, crosspoint or output queue has capacity at least two longest packets. We denote the length of queue q by $|q|$. We also define the *position* of a cell d in a queue q as the number of cells preceding d in q with respect to the FIFO order. Sometimes we use "*" to refer to all queue indices in range $[1, N]$.

We divide time into discrete steps, where a step is the arrival time between two cells at an input port. We further divide each time step into three phases. The first phase is the *transmission phase* during which the first cell from each non-empty output queue is sent on the output link. The second phase is the *arrival phase*. During arrival phase at most one cell 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 cell from virtual output queue to the corresponding crosspoint queue. During the output scheduling subphase each output port can fetch one cell 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 $OQ_{i,j}$. In the packet-mode scheduling, a packet p must be

contiguously transmitted out of each queue in the switch as a whole block. W.l.o.g., we assume that an optimal offline algorithm OPT never preempts packets.

In a switch with a speedup of S, up to S cells can be removed from any input and up to S cells can be added to each output port during the scheduling phase. This is done in (up to) S *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*. The aim of a switch policy is to maximize the total value of the packets sent out of the switch. The system accrues the value of a packet if the packet is successfully transmitted and gains no value otherwise. 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.

Definition 1 An online switch policy A is said to be c-competitive if for every input sequence of packets σ,

$$V^{OPT}(\sigma) = cV^A(\sigma) + a$$

where c and a are constants independent of σ.

3 Switch Policies

In this section we describe the switch policies that we consider in this paper.

In Figure 2 we define a simple *Gready Unit* switch policy for the case of variable length packets. Intuitively, *GU* accepts at the input-port each packet as long as there is room for it in the buffer. As for the scheduling phase, *GU* forwards contiguously an arbitrary-chosen head-of-line packet, if there is a space to store it entirely in the buffer on the receiving side. Moreover, *GU* assumes that a size of queue at any buffering level is at least twice maximal packet size, $2l_{max}$. Such property is crucial to demonstrate a constant competitive ratio. Note that *GU* never drops accepted packets implementing back pressure at all buffering levels inside the switch (i.e. packets are not transferred to crosspoint and output queues whose buffers are full). Moreover, *GU* does not assume that a full packet is received in VOQ prior to its scheduling.

Since *GU* does not define explicitly how to fetch the next packet during input and output scheduling sub-phases we define *RR-GU* switch policy that runs Round-Robin on input and output ports after processing of a full packet according to *GU*'s constraints. In addition, we define cell-based CRR policy that runs round robin on inputs and outputs. During arrival phase CRR mimics *GU*'s behaviour and later during input scheduling sub-phase a packet is fragmented to cells. During output scheduling sub-phase CRR is not packet aware already. We use it as a reference system for comparison with *RR-GU* in Section 5 for simulations.

In Figure 3 we define the *Preemptive Greedy Variable Switch Policy* for the case of variable value packets. The input ports always greedily accept more valuable packets in time of overflow since all these packets share the same buffer and we can only gain more value by preempting packets of lower value. However, inside the switch we have a preemption factor of beta since packets reside in different buffers and preemption takes place when a packet is transferred to a buffer at the next stage. That allows us to avoid pathological scenarios in which we have a sequence of packets preempting each other, where the next packet in the sequence has the value that is only by epsilon larger

Greedy Unit Switch Policy (GU)

Transmission Phase:
Transmit the head-of-line cell from each non-empty output queue.

Arrival Phase:
Accept the first cell of a packet p if the free space in the corresponding virtual output queue $VOQ_{i,j}$ is at least $l(p)$, i.e., $VOQ_{i,j}$ has enough capacity to hold p.
If the first cell of p is accepted, accept each subsequent arriving cell of p.
Otherwise, reject all cells comprising p.

Scheduling Phase:

Input Sub-Phase:
For each *idle* input i (no packet in transfer) choose an arbitrary head-of-line packet p in $VOQ^{GU}_{i,j}$ $(1 \le j \le N)$ such that the free space in $CQ^{GU}_{i,j}$ is at least $l(p)$ and transfer the first cell of p.
For each *busy* input i (there is a packet in transfer in $VOQ^{GU}_{i,j}$), transfer the first remaining cell of p.

Output Sub-Phase:
For each *idle* output j (no packet in transfer) choose an arbitrary head-of-line packet p in $CQ^{GU}_{i,j}$ $(1 \le i \le N)$ such that the free space in OQ^{GU}_{j} is at least $l(p)$ and transfer the first cell of p.
For each *busy* output j (there is a packet in transfer in $CQ^{GU}_{i,j}$), transfer the first remaining cell of p.

Fig. 2 GU switch policy for Best Effort model.

than the previous one. An optimal policy could have transmitted all these packets by delaying them in the buffers they resided at before preemption. Obviously, no policy can achieve a constant competitive ratio without having a reasonable preemption factor.

Next, we define two natural switch polices HVF-HVF and LQF-RR that will be used in Section 5 for simulations. Both of them mimic β-PGV during arrival phase. The HVF-HVF switch policy transmits *highest value first* during input and output scheduling sub-phases.

The LQF-RR switch policy treats *longest queue first* during input scheduling subphase and runs *round robin* during output scheduling subphase.

4 Analysis of the Switch Policies

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

4.1 Variable Length Packets

In this section we consider the case of variable length packets. We assume that the length of each queue is at least $2l_{max}$, which is a reasonable requirement as far as the overall switch memory is concerned. Remember that the value of a packet equals its length, i.e., each cell has a unit value (uniform value density). We show that the GU policy is 7-competitive for any speedup. To analyze the throughput of the GU policy we introduce some helpful definitions. The next definition deals with packets that OPT may deliver during a time step while GU does not.

Definition 2 For a given switch policy A, a cell sent by OPT from output port j at time t is said to be *extra* if A does not transmit a cell from output port j at this time.

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

transmission Phase:
For each non-empty output queue, transmit the head-of-line packet.

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, consider virtual output queues $VOQ_{i,*}^{PGV}$ and for each $VOQ_{i,j}^{PGV}$ choose the head-of-line packet p. If $CQ_{i,j}^{PGV}$ is not full, mark the packet 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 it is selected for transmission).
Among all the eligible packets in $VOQ_{i,*}^{PGV}$, select an arbitrary packet with the largest value and transfer it to the corresponding crosspoint queue preempting a packet with the smallest value from that queue if necessary.

Output Subphase:
For each output port j, consider crosspoint queues $CQ_{*,j}^{PGV}$. Let p be the packet with the largest value among all head-of-line packets in $CQ_{*,j}^{PGV}$. 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}.

Fig. 3 β-PGV switch policy for Differentiated Services model.

Next, we define a wider class of so called *potentially extra* cells that will encompass extra cells, similarly to [17].

Definition 3 For a given policy A, a cell d located at buffer B of OPT is called *potentially extra* if the position of d in B is greater than the length of the corresponding queue of A (see Figure 4).

Clearly, each extra cell should eventually become potentially extra prior to transmission. We will map each potentially extra cell to a cell sent by GU, in a such way that each GU cell is mapped to at most three potentially extra cells. We need some auxiliary claims. First, we will show that no potentially extra cell is formed during the transmission phase.

Claim 1 *No new potentially extra cell is formed 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 cell out of OQ_j^{OPT} and thus the difference between $|OQ_j^{OPT}|$ and $|OQ_j^{GU}|$ cannot increase. Hence, there is no new potentially extra cell in OQ_j^{OPT} whose position is greater than the length of $|OQ_j^{GU}|$. The claim follows.

We will show that if a OPT accepts the first cell of a packet that is rejected by GU, then the corresponding queue of GU is more than half full.

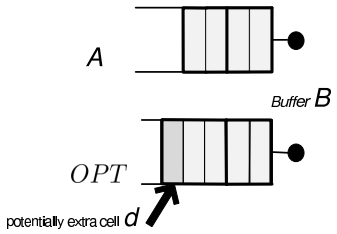

Fig. 4 Example of a potentially extra cell.

Claim 2 *Consider an arrival phase. For any virtual output queue, $VOQ_{i,j}^{OPT}$, if OPT accepts the first cell of a packet that is rejected by GU, then $VOQ_{i,j}^{GU}$ is more than half full.*

Proof Consider a virtual output queue $VOQ_{i,j}^{OPT}$. If OPT accepts the first cell of a packet p rejected by GU then $BI_{i,j} - |VOQ_{i,j}^{GU}| < l(p) \leq l_{max}$. Since by our assumption $BI_{i,j} \geq 2l_{max}$, it follows that $|VOQ_{i,j}^{GU}| > BI_{i,j}/2$.

In the following claim we bound from above the number of new potentially extra cells that are formed during an input scheduling subphase.

Claim 3 *For any input port i, the number of new potentially extra cells in virtual output queues $VOQ_{i,*}^{OPT}$ and crosspoint queues $CQ_{i,*}^{OPT}$ that are formed at the end of input scheduling subphase is at most two.*

Proof Consider an input scheduling subphase for an input port i. New potentially extra cells may be formed only in $VOQ_{i,j}^{OPT}$ if GU transfers a packet from $VOQ_{i,j}^{GU}$ and in $CQ_{i,k}^{OPT}$ if OPT transfers a cell to this queue, $j \neq k$. That establishes the claim.

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

Claim 4 *For any output port j, the number of new potentially extra cells in the crosspoint queues $CQ_{*,j}^{OPT}$ and output queue OQ_{j}^{OPT} that are formed at the end of output scheduling subphase is at most one.*

Proof Consider an output scheduling subphase t for an output port j. One new potentially extra cell may be formed in $CQ_{i,j}^{OPT}$ if GU transfers a cell from this queue. At the same time, if a new potential extra cell is formed in OQ_{j}^{OPT}, then GU does not transfer a cell during output scheduling subphase t, which establishes the claim.

The following routine in Figure 5 maps all potentially extra cells to the cells 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.

First we make the following observation concerning potentially extra cells.

Observation 1 *All potentially extra cells 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 1 no new potentially extra cell is formed during a transmission phase.

Mapping Routine: During each phase or subphase do:

Step 1 Arrival Phase: For each $VOQ_{i,j}^{OPT}$, map each new potentially extra cell to a cell in $VOQ_{i,j}^{GU}$ unmapped by this step of the mapping routine (local mapping).

Step 2 Scheduling Phase: (The next sub-steps are repeated S times for each scheduling cycle.)

Step 2.1 Input Scheduling Subphase: For each input port i:
Map a new potentially extra cell (if any) in a queue $VOQ_{i,j}^{OPT}$ to the cell transferred by GU from this queue. If OPT transfers a cell d to $CQ_{i,j}^{OPT}$, which becomes a new potentially extra cell, then proceed as follows: If $CQ_{i,j}^{GU}$ is less than half full, then map d to the cell transferred by GU from input port i. Otherwise, if $CQ_{i,j}^{GU}$ is more than half full, map d to a GU cell unmapped by this step of the mapping routine in $CQ_{i,j}^{GU}$ (local mapping).

Step 2.2 Output Scheduling Subphase: For each output port j:
Map a new potentially extra cell (if any) in a queue $CQ_{i,j}^{OPT}$ to the cell transferred by GU from this queue. If OPT transfers a cell d to OQ_{j}^{OPT}, which becomes a new potentially extra cell, map d to a GU cell unmapped by this step of the mapping routine in OQ_{j}^{GU} (local mapping).

Step 2.3: Unmap all OPT cells that ceased to be potentially extra in all OPT queues.

Fig. 5 Mapping routine for GU policy

We say that a mapping is *local* if an OPT cell is mapped to a GU cell in the same queue and a GU cell is *available* for local mapping if currently it is not locally mapped at the queue it sojourns in (see Figure 6). Now we will derive a bound on the number of local mappings that each GU cell may receive while sojourning in a queue at one of the three levels of buffering and establish the feasibility of the local mapping.

Lemma 1 *Each GU cell sojourning in a queue at any buffering level can be mapped at most once to an OPT cell in the same queue and there always exist GU cells available for local mapping when the mapping routine is executed.*

Proof Consider a queue Q at any of the three buffering levels and let B be the capacity of Q. We argue that as long as an OPT cell d is locally mapped to a GU cell in Q, d cannot leave Q earlier than its GU mate. It follows from the fact that we remove the mapping of all OPT cells that cease to be potentially extra at Step 2.3 while an OPT cell transferred from a non-empty GU queue is clearly not a potential extra one.

Consider a step of the mapping routine when a local mapping in Q is done and let x be the number of potentially extra OPT cells in Q that are currently locally mapped. By Claim 2 and the construction of the mapping, Q is more than half full in GU. We have that $x + 1 \leq B/2$ and the number of locally mapped GU cells in Q is at most x since locally mapped GU cells leave earlier than their OPT mates. Therefore, there always exists a GU cell available for mapping and a GU cell sojourning in a queue can be mapped at most once. The lemma follows.

The next theorem shows that the mapping routine is feasible and at most three potentially extra cells are mapped to a packet transmitted out of the switch by GU.

Theorem 1 *The mapping routine is feasible and no GU cell is mapped more than* 6 *times by the mapping routine prior to transmission out of the switch for any value of the speedup S.*

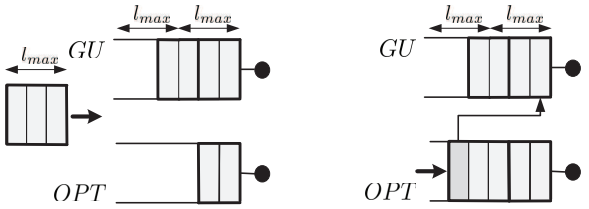

Fig. 6 Example of local mapping. *GU* drops the arriving packet due to lack of space, while the packet is accepted by *OPT*. Thus, the last cell of the packet (in *OPT*) is potentially extra and is the mapped to the first cell of the packet at the head of *GU*'s buffer.

Proof According to Lemma 1, the total number of local mappings received by each *GU* cell is at most 3 and the local mapping is feasible. Note that the arrival phase is completely covered by the local mapping.

By Claim 3, the number of new potentially extra cells in virtual output queues and crosspoint queues corresponding to an input port i that are formed at the end of input scheduling subphase is at most two. If *OPT* transfers a cell to a crossbar queue that is more than half full in *GU* and this cell becomes a new potentially extra cell, then this cell is locally mapped. We argue that no new potentially extra cell that is not locally mapped may be formed in $CQ_{i,*}^{OPT}$ if *GU* does not transfer a cell from input port i. In this case, the potentially extra cell appearing in a crosspoint queue $CQ_{i,j}^{OPT}$ must have already been potential extra at the beginning of the input scheduling subphase under consideration. Note that by our assumption, a queue that is at most half full always has enough free space to accept a packet of maximum length and thus $VOQ_{i,j}^{GU}$ is empty. Otherwise, *GU* should have transferred a cell to $CQ_{i,j}^{GU}$. Thus, Step 2.1 of the mapping routine is feasible and no *GU* cell is mapped by a non-local mapping more than twice.

Claim 4 implies that the number of new potentially extra cells in crosspoint queues and the output queue corresponding to an output port j that are formed at the end of an output scheduling subphase is at most one. We will show that a new potentially extra cell that may be formed in OQ_j^{OPT} must be locally mapped if *GU* does not transfer a cell to OQ_j^{GU}. In this case, the potentially extra cell appearing in OQ_j^{OPT} transfered by *OPT* from $CQ_{i,j}^{OPT}$ must have already been potential extra at the beginning of the output scheduling subphase under consideration. Again, a queue that is less than half full always has enough free space to accept a packet of maximum length and hence $CQ_{i,j}^{GU}$ is empty. Therefore, Step 2.2 of the mapping routine is feasible and no *GU* cell is mapped more than once.

The *GU* policy does not drop packets that have been admitted to the switch. Thus, the mapping is persistent and all mapped *GU* packets are eventually sent out of the switch. Moreover, no *GU* cell is mapped more than 6 times in total.

Now we are ready to show that *GU* achieves a competitive ratio of 7.

Theorem 2 *The competitive ratio of GU is at most* 7 *for any speedup value.*

Proof Fix an input sequence σ. Clearly, the number of cells sent by *OPT* is bounded by the number of cells sent by *GU* plus the number of extra cells. Observe that every extra cell at first becomes a potential extra cell prior to transmission. By Lemma 1, the number of extra cells is bounded by six times the number of cells transmitted by *GU*. Therefore, $V^{OPT}(\sigma) \leq 7V^{GU}(\sigma)$.

4.2 Variable Value Packets

In this section we study the case of variable value packets where all packets have a unit length. The goal is to maximize the total value of packets that cross the switch. We will demonstrate that β-PGV achieves a competitive ratio of $6+4\beta+\beta^2+3/(\beta-1)$ for any value of speedup. To prove the competitive ratio of β-PGV we will assign value to the packets sent by β-PGV so that no packet is assigned more than $6+4\beta+\beta^2+3/(\beta-1)$ times its value and then show that the value assigned is indeed at least $V^{OPT}(\sigma)$.

For the analysis, we assume that OPT maintains $FIFO$ order and never preempts packets.

The assignment routine presented on Figure 7 specifies how to assign value 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. Intuitively, when the matching routine considers a packet p scheduled by OPT it either matches p to itself if it is transmitted by β-PGV or finds another packet q at least as valuable as p sent by β-PGV if p has been dropped. Observe that when p is dropped by β-PGV, its buffer is full of packets that have the same or larger value. Furthermore, these packets can be later preempted by yet more valuable packets. Careful case analysis allows us to show that we can always find at least one packet available for mapping.

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

Claim 5 *A β-PGV packet is assigned at most once its own value before it leaves a virtual output queue.*

Proof Initially, a β-PGV packet q' in a virtual output queue can be assigned its own value by Sub-Step 3.3. If q' is later preempted by a packet q, then q is re-assigned the value that was assigned to q' by Step 4. Obviously, q is assigned at most its own value as $V(q) > V(q')$. Note that if q will be later assigned its own value by Sub-Step 3.3, then the value assigned to q by Step 4 is going to be re-assigned by Step 2 since condition (ii) must be true. The claim follows.

The next claim shows that when Step 2 re-assigns the value assigned to a β-PGV packet located at an virtual output queue, the value of the head-of-line packet in this queue is at least equal to the value that needs to be re-assigned.

Claim 6 *If condition (ii) of Step 2 is true and we re-assign the value assigned to the packet p' in $VOQ_{i,j}^{PGV}$ by Step 4, then we have that $V(p)$ is at least the value to be re-assigned, where p is the head-of-line packet in $VOQ_{i,j}^{PGV}$.*

Proof Consider the time step at which p' arrived and was accepted by both β-PGV and OPT. If condition (ii) of Step 2 is true, p' must have preempted another packet q' in $VOQ_{i,j}^{PG}$ and was re-assigned the value that had been previously assigned to q' by Step 4. Since β-PGV always preempts the least valuable packet from a virtual output queue, all packets in $VOQ_{i,j}^{PGV}$ preceding p' must have a value of at least $V(q')$. Moreover, according to Claim 5, q' had been assigned at most its own value. That establishes the claim.

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 *twice* its own 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 head-of-line packet in $VOQ_{i,j}^{PGV}$ (if any) or a dummy packet with zero value otherwise.

Step 2*:* If p is *not eligible* for transmission and either
i $V(p') \leq V(p)$ or
ii $V(p') > V(p)$, p' is present in $VOQ_{i,j}^{PGV}$, and p' has been assigned some value by Step 4,
then proceed as follows. Consider 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 head-of-line packet 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 $V(p)/\beta^2$)). If (i), assign the value of p' to $\hat{p}$; if (ii), re-assign the value assigned to p by Step 4 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 β-PGV from $CQ_{i,j}^{PGV}$ prior to t'_s. Find a packet $\hat{q}$ 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). If (i), assign the value of $V(p')$ to $\hat{q}$; if (ii), re-assign the value assigned to p by Step 4 to $\hat{q}$.

Sub-Step 2.3: Otherwise, if (ii), $V(p'') \geq V(p)/\beta$ and p'' is eligible for transmission, remove the value assigned by Step 4 to p' (this value is re-assigned by Step 1 to p'').

Step 3*:* If $V(p') > V(p)$ then proceed as follows:

Sub-Step 3.1*:* If p' was already scheduled by β-PGV, assign the value of $V(p')$ to p'.

Sub-Step 3.2*:* Else if p' is not present in $VOQ_{i,j}^{PGV}$, consider the set of packets with value at least $V(p')$ that are scheduled by β-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).

Sub-Step 3.3*:* Else if p' is present in $VOQ_{i,j}^{PGV}$, assign the value of $V(p')$ to p'.

Step 4*:* If a packet q preempts a packet q' at a virtual output, crosspoint or output queue of β-PGV, re-assign to q the value that *was or will* be assigned to q'.

Fig. 7 Assignment routine for β-PGV policy - executed each scheduling cycle t_s.

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

Lemma 2 *The assignment routine is feasible and the value of each packet scheduled by OPT is assigned to a β-PGV packet so that no β-PGV packet is assigned more than $6 + 4\beta + \beta^2 + 3/(\beta - 1)$ times its value. 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, Sub-Step 3.3, Step 4 is clearly feasible. Consider Sub-Steps 2.1, 2.2 and 3.2.

Sub-Step 2.1(see Figure 8). Let p'' be the first packet with the largest value in $CQ_{i,j}^{PGV}$ at the beginning of t'_s and suppose that p'' is not eligible for transmission. If $V(p'') \geq V(p)/\beta$ then, by the definition of β-PGV, the minimal value among the packets in OQ_j is at least $V(p'')/\beta \leq V(p)/\beta^2$ and OQ_j is full. Thus, during the following BO_j time steps β-PGV will send packets with value of at least $V(p/\beta^2)$ out

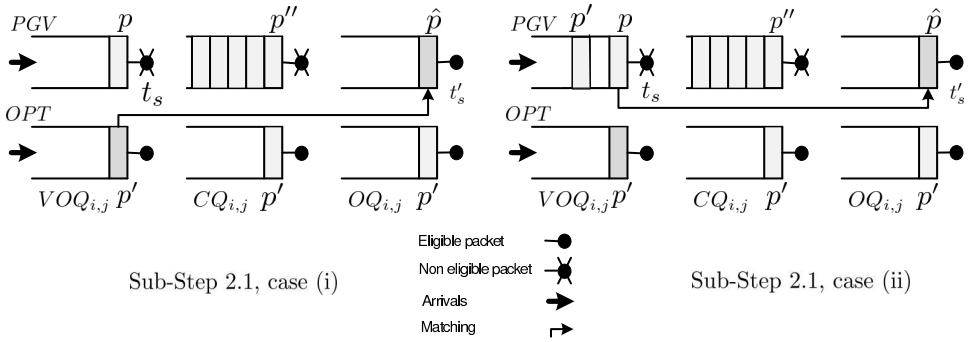

Fig. 8 Mapping on Sub-Step 2.1.

of OQ_j. 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 packet $\hat{p}$ of β-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 β-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 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 packet p' (scheduled by OPT from $VOQ_{i,j}^{OPT}$ during the input scheduling subphase of t_s) is dropped by β-PGV from $VOQ_{i,j}^{PGV}$ during the arrival phase $t_q < t_s$. Let $t_r \geq t_q$ 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 $VOQ_{i,j}^{PGV}$, $VOQ_{i,j}^{PGV}$ contains $BI_{i,j}$ packets with value of at least $V(p')$ at the end of this arrival phase. Let P be the set of these packets. Note that $p' \notin P$ because it has been already dropped by β-PGV by this time. We have that in $[t_r, t_s)$, β-PGV has actually scheduled all packets from P, since in $[t_r, t_s)$ no packet of value at least $V(p')$ has been dropped, and at time t_s all packets in $VOQ_{i,j}^{PGV}$ have value 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 currently present in $VOQ_{i,j}^{OPT}$. 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 $[t_r, t_s)$ (when it is scheduled). We now argue that OPT has scheduled at most $BI_{i,j} - 1 - x$ packets out of $VOQ_{i,j}$ in $[t_r, t_s)$, and thus P contains at least one available packet. To see this observe that the x packets from P that are present in $VOQ_{i,j}^{OPT}$ at the beginning of the scheduling cycle t_s, were already present in $VOQ_{i,j}^{OPT}$ at the end of the arrival phase t_r. The same applies to packet p' (recall that $p' \notin P$). Since OPT maintains $FIFO$ order, all the packets that OPT scheduled out of $VOQ_{i,j}^{OPT}$ in $[t_r, t_s)$ were also present in $VOQ_{i,j}^{OPT}$ at the end

of the arrival phase t_r. Therefore, the number of such packets is at most $BI_{i,j} - 1 - x$ (recall that the capacity of $VOQ_{i,j}$ is $BI_{i,j}$). We obtain that at least one packet from P is available for assignment at Sub-Step 3.2 since $|P| = BI_{i,j}$, x packets are unavailable 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.

The value of each packet scheduled by OPT is assigned to a β-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 (1) $V(p') \leq V(p)$ and p is eligible for transmission and (2) $V(p') \leq V(p)$, 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: in case (1), the value of p' is assigned during the input scheduling subphase when p is scheduled since $V(p) \geq V(p')$; in case (2), the value of p' is assigned during the output scheduling subphase when p'' is scheduled since $V(p'') \geq V(p)/\beta$. In addition, the value assigned by Step 4 and removed by Sub-Step 2.3 is re-assigned by Step 1 as Claim 6 implies. Observe that this assignment and cases (1) and (2) are mutually exclusive. If a β-PGV packet is preempted, the value assigned to it is re-assigned to the preempting packet by Step 4.

Finally, we demonstrate that no packet is assigned more than $6+4\beta+\beta^2+3/(\beta-1)$ times its own value. Consider a packet p sent by β-PGV. From Claim 5 it follows that p can be assigned at most once its own value by Sub-Step 3.3 and Step 4, before it leaves the virtual output queue. Evidently, p can be assigned at most $1 + \beta$ times its own value by Step 1 and at most $\beta + \beta^2$ times its own value by Step 2. Note that cases (i) and (ii) of Step 2 are mutually exclusive. 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 also 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 Sub-Step 3.2 at time 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 Sub-Step 3.1 and Sub-Step 3.2. Therefore, a packet that does not perform preemptions can be assigned at most $3 + 2\beta + \beta^2$ times its value.

Next we analyze Step 4 for crosspoint and output queues (we have already analyzed the value that can be assigned by Step 4 to a packet before it leaves the virtual output queue). Note that this assignment is done only to packets that are actually transmitted out of the switch (i.e. they are not dropped). 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 a 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 Sub-Step 3.3 and Step 4, 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 three times its own value. Secondly, p can preempt another packet q'' in an output queue such that $V(q'') \leq V(p)/\beta$. Observe that any preempted packet in an output queue is assigned at most once its own value by Sub-Step 3.3 and Step 4 in the virtual output queue, $1 + \beta$ times its value by Step 1, β its own value by Step 2, once its own value by Sub-Step 3.1 and Sub-Step 3.2, and $3/(\beta - 1)$ times its own value by Step 4 in the crosspoint queue.

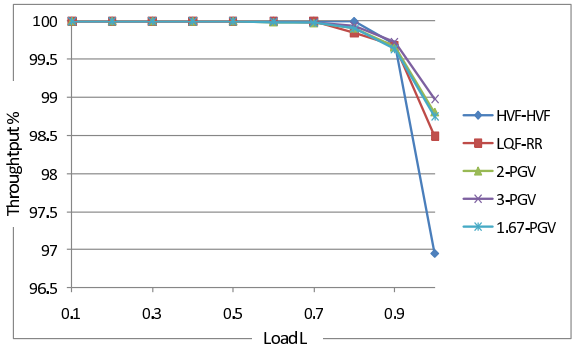

Fig. 9 Throughput of switch policies as a function of input loads for uniform distribution of output ports.

Hence, the total value that can be assigned to p by Step 4 in the crosspoint and output queues due to transitively preempted packets when p preempts q'' is bounded by $3 + 2\beta + 3/(\beta - 1)$ times its own value.

We obtain that in total no β-PGV packet is assigned more than $6 + 4\beta + \beta^2 + 3/(\beta - 1)$ times its own value.

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

Theorem 3 *The competitive ratio of the β-PGV policy is at most $6+4\beta+\beta^2+3/(\beta-1)$ for any speedup.*

The theorem follows from Lemma 2.

Corollary 1 *The competitive ratio of the β-PGV policy is at most 19.95 for preemption factor $\beta = 1.67$ and any value of speedup.*

5 Simulations

In this section we evaluate the throughput of β-PGV by means of simulation. We compare the performance of β-PGV with two natural switch policies, namely HVF-HVF and LQF-RR that are defined in Section 3. Note that LQF-RR disregards the values of packets. We consider a 32×32 switch with $BI = 4$, $BC = 2$, $BO = 8$.

We first examine the dependency of throughput on the input load during arrivals with a fixed speedup of 1. For each input port we generate a packet with a probability L, which determines the switch load. Then we choose an output port for the generated packet according to one of the distributions that we specify below. We also select weight for the generated packet uniformly at random from a range of $[1, 30]$. Each experiment runs for 1000 iterations.

Figure 9 demonstrates dependency of throughput on the input load, where an output port is chosen according to the uniform distribution. As can be seen a value of preemption factor β is not significant for this type of traffic distribution. Moreover, all considered policies perform close to the optimal way.

Figure 10 demonstrates dependency of throughput on the input load, where an output port is chosen according to Pareto distribution of order 1 and shape 1.6. The

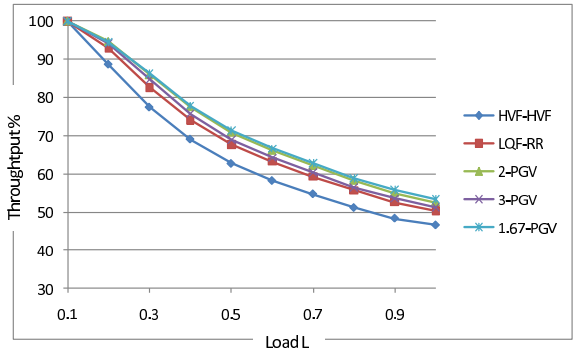

Fig. 10 Throughput of switch policies as a function of input loads for Pareto distribution

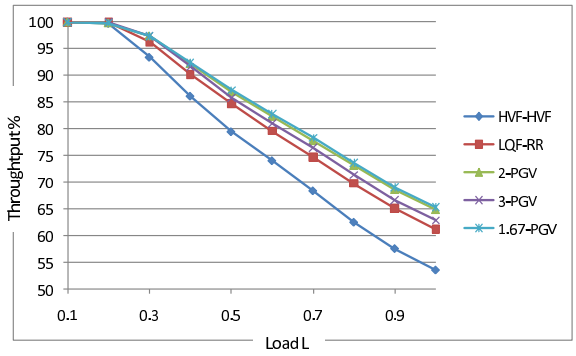

Fig. 11 Throughput of switch policies as a function of input loads for Gaussian distribution for each one of the potentially overloaded/underloaded groups of output ports.

simulation results demonstrate that β-PGV outperforms any other considered policies. Moreover, the value of preemption factor is more crucial than for the uniform distribution and 1.67-PGV outperforms β-PGV for $\beta = 2$ and $\beta = 3$.

Next, we fix a group of the first 8 output ports as potentially "overloaded". During arrival a packet p will be destined to the overloaded group with a fixed probability of 0.5. The output port in each group overloaded(underloaded) is chosen according to Gaussian distribution with mean of 4(20) and standard deviation of 4(12), respectively. Figure 11 depicts the simulation results for this type of traffic distribution. At least two observations follow from these simulations. Firstly, 1.67-PGV switch policy achieves better throughput than 2-PGV and 3-PGV. Secondly, β-PGV outperforms HVF-HVF and LQF-RR switch policies.

In addition, we evaluate throughput of switch policies as a function of speedup for a fixed moderate load of $L = 0.8$ and similar to the previous case traffic distribution. The results demonstrate that the increase of speedup has no significant impact on the throughput of the compared switch polices. The results are presented on Figure 12.

The performed simulations demonstrate that β-PGV performs much better than its theoretical worst-case bounds shown in Theorem 3 as well as outperforms the other evaluated policies.

In addition we simulate the throughput of one of the flavors of GU switch policy for packet mode scheduling RR-GU that is defined in Section 3. Recall that GU forwards

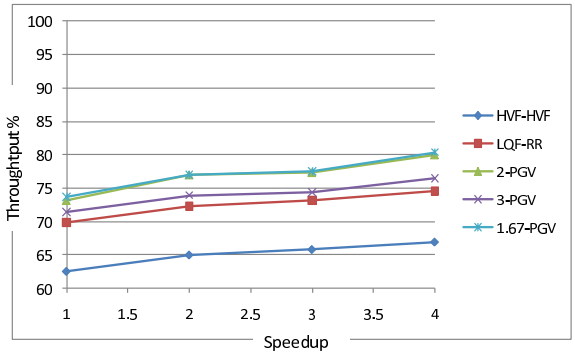

Fig. 12 Throughput of switch policies as a function of speedup for input load $L = 0.8$ and Gaussian distribution for each one of the potentially overloaded/underloaded groups of output ports.

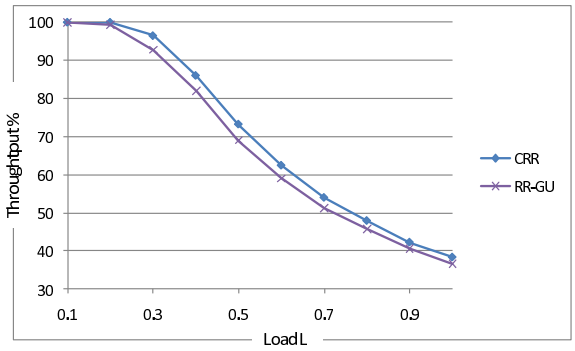

Fig. 13 Throughput of *RR-GU* switch policy for packet mode scheduling as a function of input loads for uniform distribution of output ports.

contiguously head-of-line packet, if there is a space to store it entirely in the buffer on the receiving side. We evaluate such property of GU by comparing with cell-based CRR policy (CRR is defined in Section 3) that is used as reference system. We do not perform reassembly for cells transmitted out of switch that is managed by CRR. Clearly, that CRR is superior to *RR-GU*. We consider a 32×32 switch with $BI = 4{*}10$, $BC = 2{*}10$, $BO = 8 * 10$ cells and a maximal packet size of 10 cells. Figure 13 demonstrates the result of such simulations for uniformly distributed traffic that is defined similarly to the weighted case, but now a weight corresponds to a length of packet.

The performed simulations demonstrate that *RR-GU* performs much better than its theoretical worst-case bounds shown in Theorem 2. In addition the property of GU that forwards contiguously head-of-line packet, if there is a space to store it entirely in the receiving buffer, does not affect significantly its performance even in comparison with superior a priory cell-based CRR policy.

6 Conclusions

As the switch speeds constantly grow, the development of distributed scheduling algorithms that are amenable to efficient hardware implementation becomes crucial for the

performance of the next generation high-speed switches. In this paper we consider competitive switch policies for buffered crossbars switches with FIFO 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. We propose preemptive greedy switch policy β-PGV with preemption factor β for the general case of unit size and variable value packets and arbitrary value of the switch fabric speedup. Our main result is an expression of competitive ratio of β-PGV as function on preemption factor β. In particular, the competitive ratio of the β-PGV policy is at most 19.95 for the preemption factor of $\beta = 1.67$. In addition, we demonstrate that performance of β-PGV outperforms some other natural switch policies and performs much better than its worst-case performance guarantee. We also propose a greedy switch policy operating in the packet-mode for the case of variable length packets with uniform value density that achieves a competitive ratio of 7. 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. It is interesting open question to compare throughput performance guarantees of packet- and cell-based schedulers for buffered crossbars.

Acknoledgements:

We thank anonymous reviewers for their valuable comments that helped to improve our paper.

References

1. The national laboratory for applied network research. wan packet size distribution. available online at: *http://www.nlanr.net/NA/Learn/packetsizes.html*.
2. W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosén. Dynamic routing on networks with fixed-size buffers. *Proc. SODA*, pages 771–780, 2003.
3. S. Albers and T. Jacobs. An experimental study of new and known online packet buffering algorithms. *Proc. ESA*, pages 754–765, 2007.
4. S. Albers and M. Schmidt. On the performance of greedy algorithms in packet buffering. *SIAM Journal on Computing*, 35(2), pages 278–304, 2005.
5. H. Attiya, D. Hay, and I. Keslassy. Packet-mode emulation of output-queued switches. *Proc. SPAA*, pages 138–147, 2006.
6. Y. Azar and M. Litichevskey. Maximizing throughput in multi-queue switches. *Algorithmica*, 45(1), pages 69–90, 2006.
7. Y. Azar and Y. Richter. Management of multi-queue switches in QoS networks. *Algorithmica*, 43(1-2), pages 81–96, 2005.
8. Y. Azar and Y. Richter. An improved algorithm for CIOQ switches. *ACM Transactions on Algorithms*, 2(2), pages 282–295, 2006.
9. D. Black, S. Blake, M. Carlson, Z. W. E. Davies, and W. Weiss. An architecture for differentiated services. *Internet RFC 2475*, December 1998.
10. A. Borodin and R. El-Yaniv. *Online computation and competitive Analysis*. Cambridge University Press, 1998.
11. S. Chuang, S. Iyer, and N. McKeown. Practical algorithms for performance guarantees in buffered crossbars. *Proc. IEEE INFOCOM*, pages 981–991, 2005.
12. 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*, 17, pages 1030–1039, 1999.
13. D. Clark and W. Fang. Explicit allocation of best effort packet delivery service. *IEEE/ACM Trans. on Networking*, 6(4), pages 362–373, 1998.

14. J. Dai and B. Prabhakar. The throughput of data switches with and without speedup. *Proc. IEEE INFOCOM*, pages 556–564, 2000.
15. P. Giaccone, E. Leonardi, B. Prabhakar, and D. Shah. Delay performance of high-speed packet switches with low speedup. *Proc. IEEE GLOBECOM*, pages 291–301, 2002.
16. D. Guez, A. Kesselman, and A. Rosén. Packet-mode policies for input-queued switches. *Proc. SPAA*, pages 93–102, 2004.
17. E. L. Hahne, A. Kesselman, and Y. Mansour. Competitive buffer management for shared-memory switches. *Proc. SPAA*, pages 53–58, 2001.
18. T. Javidi, R. Magill, and T. Hrabik. A high throughput scheduling algorithm for a buffered crossbar switch fabric. *Proc. IEEE International Conference on Communications*, 5, pages 1586–1591, 2001.
19. Y. Kanizo, D. Hay, I. Keslassy. The Crosspoint-Queued Switch. *Proc. IEEE INFOCOM* , pages 729–737, 2009.
20. K. Kar, T. Lakshman, D. Stiliadis, and L. Tassiulas. Reduced complexity input buffered switches. *HOT Interconnects*, 12(2), pages 13–20, 2000.
21. M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, and N. Chrysos, Variable packet size buffered crossbar (CICQ) switches. *Proc. IEEE Int. Conf. Communications*, pages 1090-1096, 2004.
22. M. Katevenis and G. Passas. Variable-size multipacket segments in buffered crossbar (CICQ) architectures. *Proc. IEEE Int. Conf. Communications*, pages 16-20, 2005.
23. A. Kesselman, K. Kogan, and M. Segal. Competitive policies for buffered crossbar switches. *Proc. 15th SIROCCO 2008*, Springer LNCS 5058, pages 170–184, 2008.
24. A. Kesselman and Y. Mansour. Harmonic buffer management policy for shared memory switches. *Theoretical Computer Science and Special Issue on Online Algorithms and In Memoriam: Steve Seiden*, 324(2-3), pages 161–182, 2004.
25. A. Kesselman, Z. Lotker, Y. Mansour, and B. Patt-Shamir. Buffer overflows of merging streams. *Proc. ESA*, pages 349–360, 2003.
26. A. Kesselman, Z. Lotker, B. P.-S. Y. Mansour, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3), pages 563–583, 2004.
27. A. Kesselman and A. Rosén. Scheduling policies for CIOQ switches. *Journal of Algorithms*, 60(1), pages 60–83, 2006.
28. A. Kesselman and A. Rosén. Controlling CIOQ switches with priority queuing and in multistage interconnection networks. *Journal of Interconnection Networks*, 9(1/2), pages 53-72, 2008.
29. M.A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri, Scheduling in input-queued cell-based packet switches. *Proc. IEEE GLOBECOM*, pages 1227–1235, 1999.
30. B. Magill, C. Rohrs, and R. Stevenson. Output-queued switch emulation by fabrics with limited memory. *IEEE Journal on Selected Areas in Communications*, pages 606–615, 2003.
31. V. Paxson and S. Floyd. Wide area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 1995.
32. R. Rojas-Cessa, E. Oki, and H. Jonathan Chao. CIXOB-k: Combined Input-Crosspoint-Output Buffered Packet Switch. *Proc. IEEE GLOBECOM*, pages 2654-2660, 2001.
33. D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *CACM 28*, 12(2), pages 202–208, 1985.
34. J. Turner. Strong Performance Guarantees for Asynchronous Buffered Crossbar Schedulers. *ACM/IEEE Transactions on Networking*, to appear, 2009.
35. A. Veres and M. Boda. The chaotic nature of TCP congestion control. *Proc. IEEE INFOCOM*, pages 1715–1723, 2000.
36. International Technology Roadmap for Semiconductors (ITRS), *Executive summary*, 2007