Multiple Priorities in a Two-Lane Buffered Crossbar

Nikos Chrysos and Manolis Katevenis†
Institute of Computer Science - Foundation for Research and Technology - Hellas (FORTH)
ICS-FORTH, P.O. Box 1385, Vassilika Vouton, Heraklion, Crete, GR-711-10 Greece
http://archvlsi.ics.forth.gr/bufxbar/ - {nchrysos,katevenis}@ics.forth.gr

September 2003

***Abstract*— A significant advantage of buffered crossbars is that they can directly switch variable-size packets. However separate queues (or lanes) per priority at each crosspoint are required to prevent HOL blocking and buffer hogging. In this paper we study a variable-size-packet buffered crossbar that effectively supports multiple priorities with only two such lanes per crosspoint, by dynamically changing the effective priority of crosspoint queues. Through simulations we show that when eight priorities are supported, even under a highly irregular traffic pattern, our system will not increase the average delay of any priority by more than 75 percent compared to the ideal system (which requires 4 times more buffering). Under realistic traffic, the two systems perform almost identically. We also compare multipriority variable-size-packet buffered crossbars to fixed-cell ones; through simulations we verify that the latter need significant speedup to reach the formers' performance.**

1 . INTRODUCTION

The crossbar is the simplest and most popular organization for high performance (internally non-blocking) switches; it is also the building block for switching fabrics. The crossbar scheduling problem, inherent in all unbuffered crossbar architectures, consists of selecting a conflict-free match of inputs to outputs; this match can only be computed in a centralized fashion. Existing, practical crossbar schedulers ([1] [2] [3]) cannot simultaneously support high crossbar-utilization and sophisticated Quality of Service (e.g. multiple priority levels or WRR scheduling): when some connections are preferentially selected over others, multiple iterations are required to produce near maximal matches [4]. The solution commonly used today is to provide significant internal speedup i.e., *combined-input-output queueing* or CIOQ [5]. In CIOQ architectures (fig. 1), WRR or strict-priority scheduling is implemented in the egress line-cards where traffic normally accumulates. Speedup considerably increases the cost of all major parts of the switch and severely limits the maximum line rate that can be supported.

The *combined-input-crosspoint-queueing* (CICQ, or *buffered crossbar*) architecture significantly simplifies scheduling [6] [7] [8] [9]. The 2· N schedulers in CICQ switches –N at the input and N at the output lines of the crossbar–, work *independently* of each other, since each of them deals with only a single resource. They are still coordinated, but only over longer timescales, through backpressure feedback from the crosspoint buffers. In

† The authors are also with the Dept. of Computer Science, University of Crete, Heraklion, Crete, Greece.

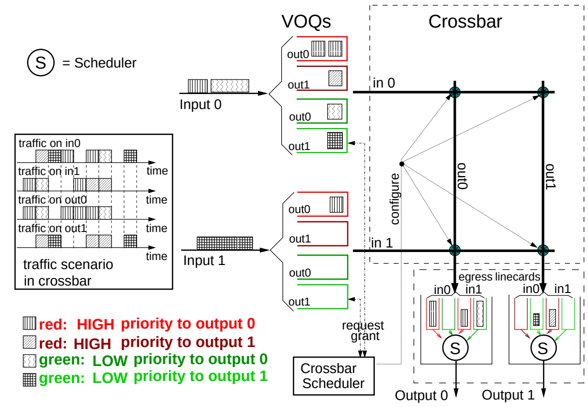

Fig. 1. The CIOQ switch with unbuffered crossbar

buffered crossbars, WRR and strict priority scheduling can be implemented at each of the input and output crossbar lines, without negatively affecting switch throughput [8]; thus no speedup is required to compensate for scheduling inefficiencies. From the implementation point of view, an advantage of buffered crossbars is that they do not require clock or cell-time synchronization among the ingress lines [10].

A related significant advantage of buffered crossbars is their capacity to directly switch variable-size packets [6] [11]. Since the 2· N schedulers in a buffered crossbar can operate asynchronously, there is no global "time-frame", that constraints the system to transmit packets in fixed-size units (segments or cells) –fig. 2. This does not hold for the unbuffered architecture, where the centralized scheduling has only be shown to operate efficiently when it manipulates fixed-size cells (fig. 1). In turn, directly switching variable-size packets eliminates the other reason for internal speedup i.e., to compensate for the segmentation overhead when the packet size is not an integer multiple of the segment size. It also removes the cost of large memories and reassembly mechanisms in the egress line-cards. Concerning power, variable size packets will generally reduce consumption, since operations like scheduling and forwarding configuration will be made on larger units (packets) and hence less frequently. For a 32×32 crossbar that directly switches all network packets of size up to 1500 bytes, 16Mbits of on-chip memory are required (with 2 Kbyte memory per crosspoint). In a companion paper [12], we demonstrate that this is typically feasible with current ASIC technology.

Multiple priorities are desirable to provide means for service differentiation; e.g. IEEE 802.ID/Q defines eight classes of service by means of priorities. Proper priority mapping, on the application or packet level can provide efficient utilization of network resources and increase the "user-perceived" utility [14]. A CICQ switch supporting multiple priorities, requires separate crosspoint buffers/queues (or lanes) per priority to prevent HOL blocking and buffer hogging [11]. These effects can make a high priority packet receive low-priority service (sec. 2.2). However, each additional such queue incurs a significant cost, hence we want to economize on their number.

This paper studies a buffered crossbar that directly switches variable-size packets and supports multiple priorities while economizing on the number of lanes per crosspoint. Section 2 presents our baseline architecture, regarding flow-control, queuing, scheduling and buffer dimensioning. In section 3, we propose a method, SQP, that resolves, in the short term, HOL blocking and buffer hogging which occur when many different priorities are multiplexed on a shared lane; even with one queue per crosspoint, SQP prevents unjustified starvation of high priority flows. Section 4 proposes an enhanced version of SQP, 2B-ADAPT, that effectively supports multiple priorities with two lanes per crosspoint. In section 5 we show through simulation, that when eight priorities are supported, even under highly irregular traffic, 2B-ADAPT will not increase the average delay of the HIGHEST priority level by more than **75** percent, when compared to a system with a distinct lane per priority (i.e., four (**4**) times more buffering); intermediate priorities experience much smaller discrepancies. Under smoother patterns, like typical network traffic or Poisson arrivals, we find that the two systems perform almost identically. We also compare the multipriority variable-packet-size buffered crossbar to a fixed-size-cell one and to pure output queuing with per priority output queues. In section 5 we show that the only relatively complex functionality of our methods is located in the ingress line-cards.

To the best of our knowledge, this is the first published study of how to effectively map multiple priority levels onto a reduced number of queues in a buffered crossbar –and perhaps in any kind of switch in general. The importance of the paper stems from this novelty and from the importance of buffered crossbars –especially variable-packet-size ones– as the probable emerging architecture of choice for future commercial crossbar products.

2. System & Methods

2.1. Baseline Architecture

The system we consider in this paper is a buffered crossbar that directly switches variable-size packets (fig. 2). In order to effectively support multiple priorities, separate virtual-output-queues (VOQ) per priority are maintained at the input line-cards. Credit-based flow-control is employed, to provide lossless transmission at the link level between the input line-cards and the buffers within the crossbar. Ideally, a separate FIFO queue with distinct buffer space, is allocated for each

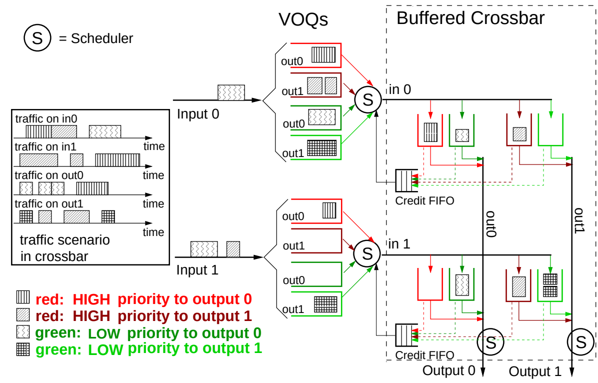

Fig. 2. A CICQ switch with separate crosspoint lane per priority

priority at each crosspoint; we name this system multiple-queue-distinct-buffer (in short, MQDB).

As we demonstrate in [12], each separate crosspoint FIFO is placed in a dedicated address range of a dual-ported SRAM, where it is implemented as a circular queue; we support cut-though at the crosspoints. When a packet is selected for service at the output, a credit informing about the packet departure is generated inside the crossbar chip. Credits destined to the same input are sent in FIFO order and the credit rate per input port is set equal to one credit every minimum-packet-size (in short, MnPS) time.

For the credit-based flow-control to operate correctly, the size of each separate crosspoint FIFO, B, must be at least equal to the maximum-packet-size (in short, MxPS) allowed in the network. In order to sustain full output-link utilization even when a single flow is active, B must be greater or equal to MxSP + RTT$\cdot$ LR, where LR stands for the line rate and RTT stands for the delay from the generation of a credit till the first word of a packet, that utilized this credit at the input, starts being transmitted on the output lines of the crossbar [12].

The scheduling discipline assumed at the input and the output links is a combination of non-preemptive, priority scheduling and advanced Round-Robin (RR): when queues of different priorities are eligible for service, the highest eligible priority, l, is selected; if multiple queues with priority l are eligible, we use RR to select one of them. Scheduling at a link starts when a flow (i.e., a distinct input/output/priority-level triplet) becomes eligible, if the link is currently idle, or one scheduling delay before the current transfer ends [13].

2.2. HOL blocking & Buffer Hogging

Separate lanes per priority, at each crosspoint are desired in order to provide flow isolation and protection. If a single crosspoint queue, Q, is shared among multiple priorities –single-queue switch (in short, SQ)– and low priority packets have been enqueued in Q first, a higher priority packet that is inserted after them can experience unacceptable long delay until it becomes the head of Q; this phenomenon is known

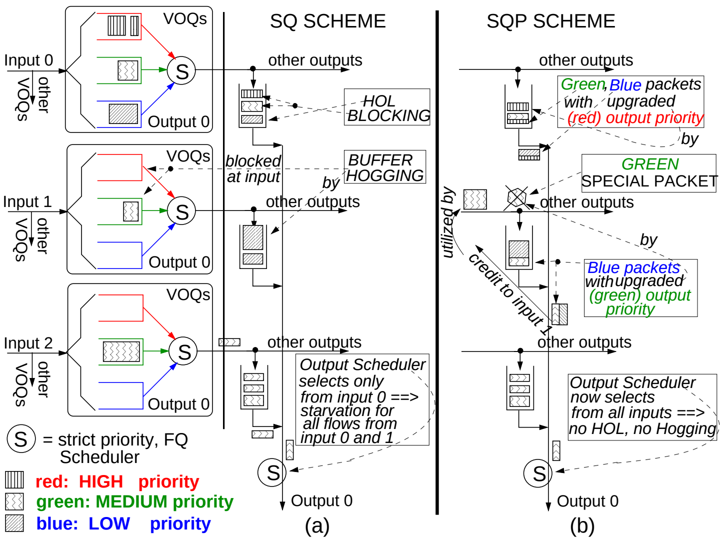

Fig. 3. **(a)** HOL blocking & Buffer Hogging in a SQ switch **(b)** SQP

as Head Of Line (HOL) blocking –fig. 3(a). Even if multiple (logical) queues, one for each priority, are maintained at each crosspoint but these queues share a common buffer space –multiple-queue-shared-buffer switch (in short, MQSB)– similar effects can occur if low priority packets have exhausted the shared buffer space: higher priority packets, that are ineligible for service at the input due to flow-control, will have to wait until the low priority queues are served, so these can experience a delay comparable to that of low priority traffic; this effect is known as buffer hogging. Note that buffer hogging can also appear in the SQ scheme, while MQSB by definition prohibits HOL blocking.

Both HOL blocking and buffer hogging can have destructive effects on the higher priorities. Even a flow in the HIGHEST priority level, that normally receives preferential service, can experience starvation, which is indirectly induced by congestion of flows with intermediate priority –fig. 3(a).

An additional disadvantage of MQSB, is the control circuit, that is required at each crosspoint in order to maintain the logical queues. Besides the fragmentation that can occur if variable size packets are to be stored inside the queues, this scheme significantly increases the complexity of the already heavily loaded crossbar chip. Instead, circular queues, implemented in private buffer space, require much simpler control (e.g. a *head* and a *tail* counter) and do not incur fragmentation. A MQSB system, switching cells, is examined in [10].

2.3. Single Queue with Push-Forward

In the first scheme that we explore, like in SQ, all flows from the same input that go to the same output share a single crosspoint queue. In order to resolve HOL blocking we employ the following discipline at the crosspoints: the "effective priority" of a crosspoint queue Q, for output scheduling purposes, is the highest of the priorities of the packets currently enqueued in it; in this way, Q drains with the priority of the most "urgent" packet currently inside it. By doing that, low priority packets at the HOL can no any longer cause starvation to high priority packets behind them. We name this scheme single-queue-push-forward (in short, SQP), see fig. 3(b).

To resolve buffer hogging, the input in SQP sends a special packet to signal this priority upgrade, in lieu of the actual packet which cannot be sent due to flow-control constraints. Special packets are not stored in crosspoint queues, neither they are forwarded to outputs, but they are subject to scheduling at the input and are transmitted like normal packets, through the same interface. Their size has been set equal to a MnPS, so that the scheduling rate is not affected.

Consider now the case, where an input, i, has not yet received credit from a crosspoint queue, Q, for a packet of priority l^*, higher than the highest eligible priority, l, in the VOQs that correspond to Q.

This fact indicates probable heavy traffic at level l^* or higher in the crossbar. Under such conditions, if a priority l packet is enqueued in Q, this will need to wait until all packets in front of it are served and probably even more if output congestion actually occurs; this increases the probability that buffer hogging or HOL blocking will appear if a packet with priority higher than l arrives later at input i for the same output.

So we performed the following optimization to SQP (SQP-opt): when packets of priority l^* or higher are still pending (i.e., not yet acknowledged to the input by means of credits) at a crosspoint queue, Q, the input is not allowed to send a packet with priority lower than l^* toward Q. Although SQP-opt is not work-conserving, we show through simulations, that compared to SQP, it significantly reduces the delay of all high priorities, without considerably affecting the crossbar utilization. Another advantage of SQP-opt is that it simplifies the priority upgrade mechanism that must be implemented at the crosspoints.

Although with the aforementioned methods persistent buffer hogging and HOL blocking that may appear under worst-case scenaria are resolved, this comes at the cost of occasionally transmitting packets with higher priority than their specification, which increases the load at higher priorities, hence, potentially increases the average delay of high priorities.

2.4. Two Buffers with Adaptive Mapping

Our second method, 2-buffers-with-adaptive-mapping (in short, 2B-ADAPT), uses SQP-opt ideas in a more sophisticated way. We assume two separate buffer/queues (lanes) per crosspoint;

UP stands for one of these lanes and DOWN stands for the other. An algorithm, Alg-2B-ADAPT, running independently at each input line-card, decides through which lane a packet can make it to the output. Alg-2B-ADAPT should not be confused with the input scheduler; essentially, Alg-2B-ADAPT monitors whether a flow f is eligible for service: it returns UP or DOWN when it finds f eligible –the packet at the HOL of f's VOQ can then use the respective lane– and NONE otherwise. The input scheduler, as described in sec 2.1, selects one of the eligible flows. Alg-2B-ADAPT is adaptive, since it processes packets based on the run-time state of the two lanes at the corresponding crosspoint.

The output scheduler in 2B-ADAPT considers all the non-empty UP and DOWN queues in a column of the crossbar and serves the one with the highest effective priority (l); when both the UP and DOWN queues in a crosspoint have the effective priority l, the scheduler selects UP. The main idea in 2B-ADAPT is the following.

Consider all flows from a given input to a given output. We try to use only the DOWN lane, but when multiple priorities are active, we allocate the UP lane to the highest priority among them. The packets of the other active flows, either use DOWN, or are kept at the VOQs. For all flows of priority l, lower than the HIGHEST supported, the algorithm is conservative when it uses UP: we want to keep the UP queue usually empty, so that a packet of priority higher than l, that arrives later at the input, is able to go through UP without experiencing the delay of traffic with priority l. Whenever HOL blocking appears in any of the two queues, or buffer hogging appears at UP, the mechanisms of SQP apply. Thus, Alg-2B-ADAPT can also return SpUP, meaning that the flow is eligible but only to send a special packet UP.

We assume the following data structures at each input, for each output; the number of flows corresponding to this pair is L, the number of priority levels. Two bits for each flow f, FQ [f], indicating whether the most recently sent and still pending packet of this flow was sent UP or DOWN, or NONE (if f has no pending packet). Similarly, PackUP [f] counts the number of f's pending UP packets; and TimeUP [f] keeps track of the time that has elapsed since the input started transmitting the oldest pending UP packet. Finally, two registers, EffPr [UP] and EffPr [DOWN], estimate the "effective" output-priority of UP and DOWN respectively, by keeping track of the highest pending priority in the corresponding queue; when no packet is pending UP or DOWN, the respective register equals NONE[1].

Suppose that a packet p, from flow f with priority l, is processed by Alg-2B-ADAPT. As fig. 4(a,b) shows, the HIGHEST supported priority, l= 0, is mapped only UP whereas, the LOWEST, l= L$-$1, is mapped only DOWN. Intermediate priority levels can be mapped either UP or DOWN –fig. 4(c).

1) Justification and Alternative Policies: Like SQP-opt, 2B-ADAPT never sends a packet to a queue where packets of higher priority are pending. Regarding intermediate priorities, when (a) FQ [f] equals NONE, i.e. f has no pending packet, 2B-ADAPT first tries to map p DOWN, but if it finds that EffPr [DOWN] is lower than l, it tries to map it UP, to save the delay of the packets pending DOWN. When (b) FQ [f] equals DOWN, 2B-ADAPT will use only the DOWN lane; this property, combined with the discipline at the output, ensures in-order transmission of packets within a flow [13].

Finally, (c) when FQ [f] equals UP, p is found eligible (UP)

[1]When we compare priorities, the NONE value for EffPr [{UP,DOWN}] translates either as HIGHEST, or as LOWEST –see fig. 4(d)

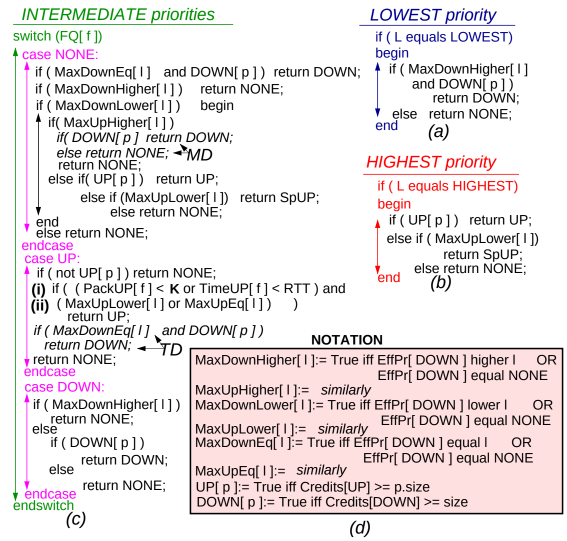

Fig. 4. Alg-2B-ADAPT policy for: **(a)** LOWEST priority; **(b)** HIGHEST priority; **(c) Intermediate priorities. (d) Notation. (The lines in *italics* correspond to alternatives policies and are not activated in 2B-ADAPT.)**

only if (i) PacketUP [f] $< K$ or TimeUP [f] $<$ RTT; and (ii) EffPr [UP] $\geq l$, i.e. no flow of priority l^*, higher than l, is using UP. We use criterion (i) in order to keep the UP queue relatively empty, and (ii) to reserve the UP lane for level l^* [2]. In 2B-ADAPT, K is set equal to one (1).

If we do not include the "TimeUP [f] $<$ RTT" condition in criterion (i), (policy 2B-ADAPT-noRTT –in short, noRTT) then an intermediate priority flow, which uses the UP lane and sends small packets, may not be able to reach full link rate. For instance, if f sends MnPS packets and K is less than $\frac{RTT \cdot LR}{2 \cdot MnPS}$, then noRTT will bound f's rate to $\frac{LR}{2}$. The noRTT policy limits the number of packets that are pending UP, and thus, normally, reduces the delay of the higher priorities.

In either case when criteria (i) or (ii) fail, instead of blocking p, we can try to send it DOWN (policy 2B-ADAPT-TD –in short, TD) –see *TD* marker in fig. 4. Similarly to policy noRTT, 2B-ADAPT-TD has the potential to reduce the delay of the higher priorities – remember that while FQ [f] equals DOWN, f cannot use UP. However, the TD policy can be unjustifiably rushy and aggressive if it causes packets that were queued DOWN to be upgraded to priority l, because TD send p as DOWN while criteria (i) or (ii) had failed:

Normally, the UP queue will drain before DOWN starts receiving service [3]; hence, before p reaches the HOL of DOWN, (i) and (ii) will probably not fail any more. In this case, p would be able to go through the UP lane with similar or smaller delay and without upgrading any other packets. Furthermore, the failure of either criterion indicates congestion at level l or higher within the crossbar, that can delay p in spite

[2]We assume that a flow's packet arrivals exhibit temporal locality

[3]EffPr [UP] (i.e. l^*, or l) will be normally higher or equal to EffPr [DOWN] (i.e. at most l, after p is enqueued); this is the effort of the algorithm.

of the priority upgrades. In this case, sending p DOWN can also cause future buffer hogging or HOL blocking, if in the meanwhile that p is queuing DOWN, packets of priority higher to l appear at the input for the same output. Alg-2B-ADAPT simply blocks p, until no packet is pending UP, when (ii) fails, or until one credit for f is received, when only (i) fails.

Another policy that we examined activates the code lines in fig. 4 marked *MD*. The medium-priority-down policy (or 2B-ADAPT-MD –in short, MD) maps a packet, p, DOWN when (1) FQ $[f]$ equals NONE, and (2) l is higher than EffPr [DOWN] and lower than EffPr [UP]. 2B-ADAPT simply blocks p in this case. Policy MD has similar advantages and disadvantages with policy 2B-ADAPT-TD; through simulations we observed that these alternatives can reduce the delay of the HIGHEST priority, but at the expense of intermediate priorities experiencing considerably higher delays.

3 . SIMULATIONS

We created an event-driven simulator to experiment with our methods. In this paper, we simulate a 32×32 switch with port speed 10 Gbps; for simplicity we assume no internal packet-header and thus no speedup [4]. The VOQs and the crosspoint queues implement cut-through. The RTT is set equal to 400ns (or 500byte time), resulting as the sum of the following delays: (1) one propagation delay (or PD [5], 100ns) and one transfer delay (equal to a MnPS time, 32ns), before a credit, c, reaches the input; (2) one scheduling delay (or SD, 30ns) at the input, plus one VOQs memory access delay (76ns) and one PD, before a packet, p, that used c at the input, reaches the crosspoint; last (3) one SD at the crossbar output, before p starts being transmitted on the output lines of the crossbar [13].

We explicitly model variable-size packet arrivals. Packets' output destinations are uniformly distributed and all priorities arrive with equal probability. We use **Poisson**, and **Bursts60** packet arrivals with burst length sixty back-to-packet packets and exponentially distributed idle periods (average burst size 23 Kbyte). The **Bursts60** pattern models worst-case real traffic scenarios. Packets within a burst have the same destination and the same priority. The size of each packet is always selected independently using a Pareto distribution: average-packet-size (or AvPS) 400, MnPS 40 and MxPS 1500 (bytes). We plot packets' average delay just-before-output-service (or, equivalently, packets' waiting-time), for each priority in separate, versus the aggregate input load; we present plots only for the most interesting priority levels (p0 stands for the HIGHEST).

2) OQ vs MQDB: We start by comparing the MQDB architecture, with 2 Kbyte buffer space per crosspoint queue, to the pure output queuing system (OQ[6]). We also present the performance of OQ, computed using the theory of *non-preemptive*, priority scheduling, that can be found in [15].

[4]At 10Gbps pure payload link rate, a 64Byte cell-time equals 0.0512 usec

[5]PD includes time-of-flight on the interconnect between the crossbar-chip and the input line-cards, plus interfacing and pipeline delays within the chips.

[6]In the OQ model, the output queues, which are are organized per input and per priority, have infinite size and the minimum delay of a packet is $\frac{RTT}{2}$

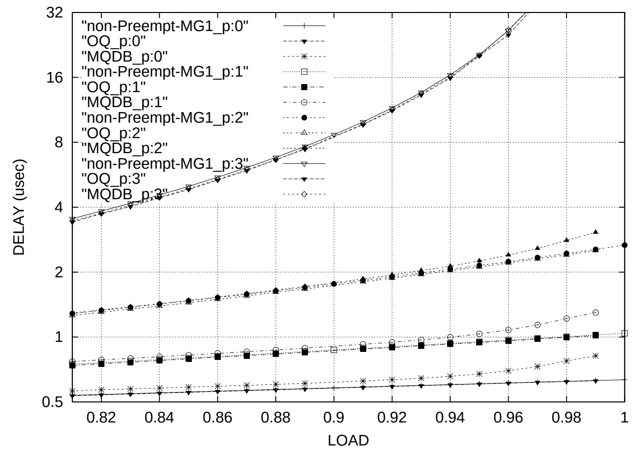

Fig. 5. 4 priority levels (p0 HIGHEST), **Poisson** arrivals. Y axis in logscale.

This theory, for **Poisson** arrivals, shows that the average delay of priority level l, D_l, depends primarily on the load of levels l^*, higher or equal to l. D_l loosely relates to the load (r_m) of a lower level, m, through the delay that a priority l packet, which just became eligible, can experience due to an ongoing transfer of a packet with priority m. Applying the respective formulas in [15] for the OQ system and for the Pareto packets' size distribution that we use, we find that:

$$D_l = \frac{0.3051 \cdot (\sum_{l^*=0}^{l} r_{l^*} + \sum_{m=l+1}^{L} r_m)}{(1 - \sum_{l^*=0}^{l}(1-r_{l^*})) \cdot (1 - \sum_{l^*=0}^{l-1}(1-r_{l^*}))} \text{ usec.}$$

As diagram 5 shows, the analytical model perfectly matches the simulated OQ system. It also shows that at input loads up to 0.94, MQDB performs identically to OQ. At higher loads, small discrepancies occur, that are more prominent for the higher priorities; these can be attributed to the *two* non-preemptive schedulers that each packet has to pass in MQDB, and to small inefficiencies of MQDB (input-queuing vs OQ).

3) MQDB vs Shared Buffer/Queue: Here, we examine the performance of MQSB, SQ, SQP and SQP-opt, all with crosspoint buffer space equal to 2 Kbyte. To model MQSB (see sec. 2.2), we use a single credit counter at each input for each corresponding crosspoint, that counts for the available space (2 Kbyte) of all logical queues in that crosspoint. In an actual MQSB system, memory fragmentation could worsen performance.

In diagram 6 we compare MQDB, MQSB, SQ and SQP-opt under **Poisson** arrivals; we see that at 0.81 input load, SQP-opt and MQSB perform close to MQDB while SQ cannot discriminate well low from high priority traffic: this is due to HOL blocking. With increasing load, all systems except MQDB downgrade similarly with SQ; in SQP-opt this happens at lower load than in MQSB, but MQSB downgrades more sharply at higher loads (i.e. near 0.97): this is due to buffer hogging. Our results in [13] indicate, that under this scenario, SQP-opt assigns lower delay to levels {p0, p1, p2} compared to SQP ({5, 8, 20} and {8, 12, 25} usec respectively, at 0.99 load), while p3 receives almost the same service.

In diagram 7, where we compare MQDB, MQSB, SQP-opt

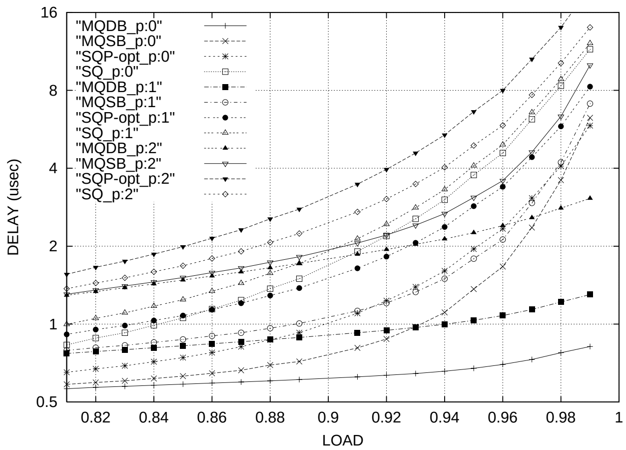

Fig. 6. 4 priority levels (p0 HIGHEST), **Poisson** arrivals. Y axis in logscale.

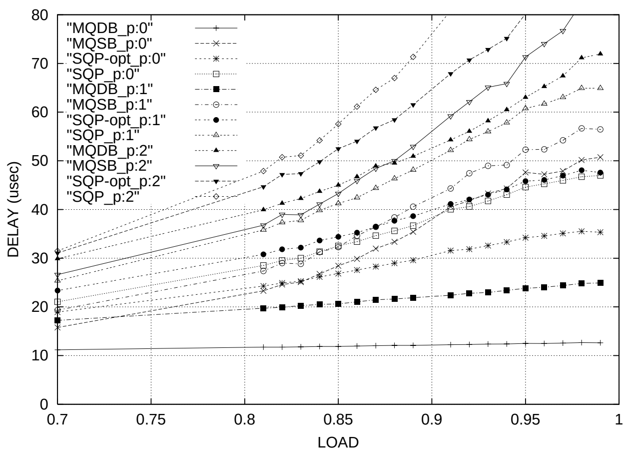

Fig. 7. 4 priority levels (p0 HIGHEST), **Bursts60** arrivals.

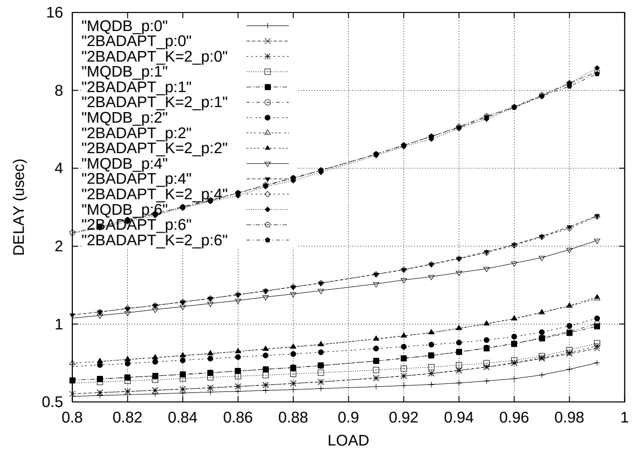

Fig. 8. 8 priority levels (p0 HIGHEST), **Poisson** arrivals. Y axis in logscale. 2B-ADAPT-K2 plots start at 0.8 load.

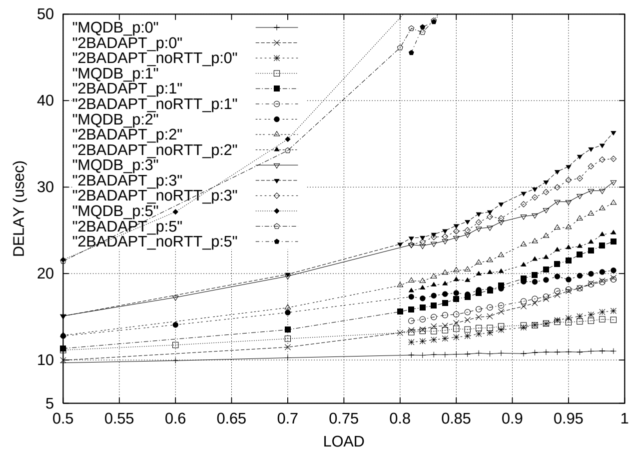

Fig. 9. 8 priority levels (p0 HIGHEST), **Bursts60** arrivals. 2B-ADAPT-noRTT plot starts at 0.8 load

and SQP under **Bursts60** arrivals, performance degradation in all shared-memory systems is more evident and the superiority of SQP-opt compared to SQP is apparent. At low load (i.e. 0.5 to 0.7), MQSB performs better than SQP and SQP-opt since it employs multiple queues that eliminate HOL blocking, and buffer hogging is not harmful when the crosspoints' buffers are relative empty; it performs much worse though, at higher input load, where the crosspoint buffers fill more frequently.

4) MQDB vs 2B-ADAPT: Following, we examine the performance of 2B-ADAPT, with 2 Kbyte buffer space assigned to each crosspoint queue. With four priority levels we found, that under **Poisson** arrivals 2B-ADAPT performs *identically* to MQDB, while, under **Bursts60** arrivals, small discrepancies occur at loads higher than 0.8; these discrepancies grow with increasing load, but in our results never exceed 50%, that appears at p0's average delay (18 vs 12 usec, at 0.99 load) [13].

Diagr. 8 is for eight (**8**) priorities and **Poisson** arrivals. It compares 2B-ADAPT to MQDB and to 2B-ADAPT-K2 (in short, K2), i.e. an alternative policy that sets K in criterion (i) equal to two (sec. 2.4.1). The diagram shows that 2B-ADAPT and K2 perform similarly under this scenario. Compared to MQDB, 2B-ADAPT exhibits negligible discrepancies that are observable only at input load higher than 0.85. The maximum discrepancy, under all priorities, is near 15% (p0's average delay, 0.8 vs 0.7 usec, at 0.99 load). For **Bursts60** arrivals, our results in [13] show, that 2B-ADAPT assigns smaller average delay, when compared to K2, to all priority levels except p7.

Diagr. 9 compares 2B-ADAPT to MQDB and to 2B-ADAPT-noRTT, under eight priorities and **Bursts60** arrivals; it shows that the noRTT policy reduces the delay of all high priorities. Compared to MQDB, the maximum discrepancies occur at the average delay of priority p0: 75% and 35% in 2B-ADAPT and in noRTT respectivelly, at 0.99 load. Regarding lower priority levels, the corresponding discrepancies are considerably smaller (less than 33%). A noteworthy point, relative to sec. 2.4.1, is that in the average case, the noRTT policy will not bound flows' rate, if $K \cdot$ AvPS $>$ RTT$\cdot$ LR.

Next we use a synthetic traffic pattern, **SynthBackb**, that tries to emulate as much as possible backbone, realistic IP traffic. In synopsy, under the **SynthBackb** pattern, the packet arrivals at an input line-card are generated by multiplexing thousands of interactive (IC) and bulk (BC) "conversations"

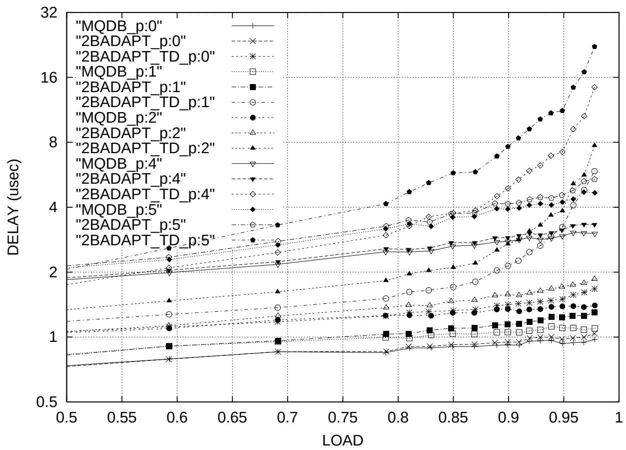

Fig. 10. 8 priority levels (p0 HIGHEST), **SynthBackb** arrivals. Y axis in logscale.

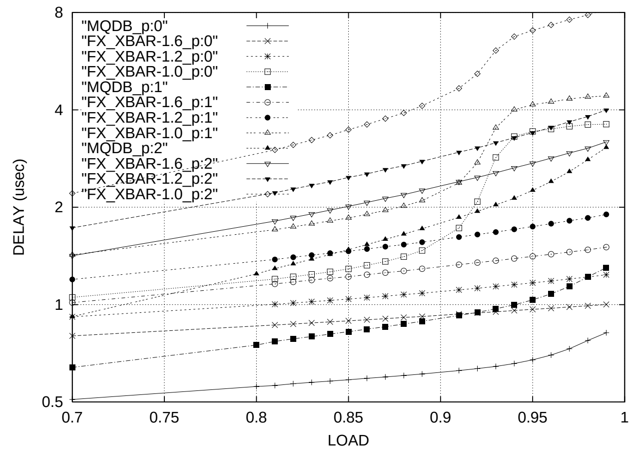

Fig. 11. 4 priority levels (p0 HIGHEST), **Poisson** arrivals. Y axis in logscale.

in a FIFO queue; an IC is modeled as a Poisson process that sends 125 packets with size 40 to 44 bytes, while BC as a burst with average size 8 Kbyte [12]. Diagr. 10 contains plots for MQDB, 2B-ADAPT and 2B-ADAPT-TD under **SynthBackb** arrivals and eight priority levels; each generated conversation is independently mapped to one level, using a uniform distribution. The diagram shows 2B-ADAPT to perform very close to MQDB. The TD alternative policy performs very poorly for the reasons described in sec. 2.4.1: compared to 2B-ADAPT, TD dramatically increases the delay of all priorities except p7. Under **Bursts60** and **Poisson** arrivals, we found that 2B-ADAPT-TD performs better than 2B-ADAPT regarding p0 and p7, and worse for all other priorities [13].

5) Packets vs Cells: Diagr. 11 compares MQDB with a similar *cell* system, FX-XBAR. FX-XBAR segments packets in the ingress and reassemblies in the egress line-cards. In the FX-XBAR model, we assumed 64 byte cells, buffer size per crosspoint queue equal to RTT×IR (IR stands for the internal line rate), and we used speedup factors 1.0, 1.2 and 1.6×; the scheduling delay is set equal to one cell time (or CT, 64byte/IR) and the credit rate per input port equal to one credit every CT.

As the diagram shows, FX-XBAR-1.0× saturates near 0.9 load, where all priorities perform purely. In FX-XBAR-1.6×, the average delay of p0 is not affected by the load, but is higher than in MQDB: by constrast with MQDB, in the FX-XBAR the delay of a p0 packet, p, includes one transmission delay of p toward reassembly. Normally, non-preemptive scheduling has lesser impact with cell switching: a high priority flow that now becomes eligible at the input or the crossbar-output scheduler, cannot not be delayed by more than one CT, due to an ongoing lower priority transfer. This does not hold in the egress line-cards, where the reassembled packets are served non-preemptively. Finally observe that by measuring packets' waiting-time, we favored FX-XBAR: if two packets arrive at time t from different inputs in an idle FX-XBAR, the smaller one will be reassembled firsts, due to smaller transmission delay, and thus will be served first; it is known that a smaller-packet-first server reduces packets' average waiting time.

HARDWARE COST

The SQP and 2B-ADAPT methods, require circuits inside the crossbar only to calculate queues' effective priority. If the inputs do no sent priority l packets, toward a queue Q with effective priority higher than l, like in SQP-opt and 2B-ADAPT, then one register maintaining the highest priority, under all packets currently inside Q, suffices. If this does not hold, like in SQP, then a FIFO per crosspoint is required. Regarding the complexity of Alg-2B-ADAPT, note that it is structured by a switch-statement and few number of comparisons between small values. At a packet arrival or departure, Alg-2B-ADAPT has to "run" for a single flow; at a credit, c, arrival, it must compute the eligibility of L flows, i.e. those that can utilize c. Currently, while we are designing Alg-2B-ADAPT in ASIC, we investigate architectural methods that eliminate this intricacy.

CONCLUSIONS

We presented a novel multiple priority CICQ switch, with very good performance, that employs a minimal number of lanes at each crosspoint, i.e. one or two. Our results indicate, that implementing more than two crosspoint lanes does not worth the associated, high incremental cost, since our system with two such lanes performs very close to the ideal system, which employs a separate crosspoint lane for each priority: under a worst case scenarion the maximum relative discrepancy is 75% whereas, in the typical case it is only 15%. Finally, we demonstrated that the only considerable, additional cost of our system is located in the ingress line-cards.

REFERENCES

[1] T. Anderson, S. Owicki, J. Saxe, C. Thacker: "High-Speed Switch Scheduling for Local-Area Networks", *ACM Trans. on Computer Systems*, vol. 11, no. 4, Nov. 1993, pp. 319-352.

[2] R. LaMaire, D. Serpanos: "Two-Dimensional Round-Robin Schedulers for Packet Switches with Multiple Input Queues", *IEEE/ACM Trans. on Networking*, vol. 2, no. 5, Oct. 1994, pp. 471-482.

[3] N. McKeown: "The iSLIP Scheduling Algorithm for Input-Queued Switches", *IEEE/ACM Trans. on Networking*, vol. 7, no. 2, April 1999, pp. 188-201;
[4] N. Ni, L. N. Bhuyan: "Fair scheduling for Input Buffered Switches", citeseer.nj.nec.com/482342.html
[5] P. Krishna, N. Patel, A. Charny, R. Simcoe: "On the Speedup Required for Work-Conserving Crossbar Switches", *IEEE J. Sel. Areas in Communications*, vol. 17, no. 6, June 1999, pp. 1057-1066.
[6] D. Stephens, H. Zhang: "Implementing Distributed Packet Fair Queueing in a scalable switch architecture", *Proc. INFOCOM'98 Conf.*, San Francisco, CA, March 1998, pp. 282-290.
[7] R. Rojas-Cessa, E. Oki, and H. Jonathan Chao: "CIXOB-k: Combined Input-Crosspoint-Output Buffered Switch", *Proc. IEEE GLOBECOM*, 2001, vol. 4, pp. 2654-2660.
[8] N. Chrysos, M. Katevenis: "Weighted Fairness in Buffered Crossbar Scheduling", *Proc. IEEE Workshop High Perf. Switching & Routing (HPSR 2003)*, Torino, Italy, June 2003, pp. 17-22; http://archvlsi .ics.forth.gr/bufxbar/
[9] G. Georgakopoulos: "Few buffers suffice: Explaining why and how crossbars with weighted fair queuing converge to weighted max-min fairness", http://archvlsi.ics.forth.gr/bufxbar/
[10] F. Abel, C. Minkenberg, R. Luijten, M. Gusat, I. Iliadis: "A Four-Terabit Packet Switch Supporting Long Round-Trip Times", *IEEE Micro Magazine*, vol. 23, no. 1, Jan./Feb. 2003, pp. 10-24.
[11] K. Yoshigoe, K. Christensen: "A Parallel-Polled Virtual Output Queued Switch with a Buffered Crossbar", *Proc. IEEE Workshop High Perf. Switching & Routing 2001*, Dallas, TX, USA, May 2001, pp. 271-275; http://www.csee.usf.edu/∼christen/hpsr01.pdf
[12] Manolis Katevenis, Giorgos Passas, Dimitris Simos, Giannhs Papaefstathiou and Nikos Chrysos "Variable Packet Size Buffered Crossbar (CICQ) Switches" http://archvlsi.ics.forth.gr/bufxbar
[13] Nikos Chrysos: "Design Issues of a Multiple-Priority, Variable-Size-Packet Buffered Crossbar" *Technical Report, ICS FORTH Hellas, Department of Computer Science, University of Crete, October 2004*; http://archvlsi.ics.forth.gr/bufxbar
[14] Wu-chang Feng et. al. "Adaptive Packet Marking for Providing Differentiated Services in the Internet", *Proc. of Int. Conf. on Network Protocols* Oct. 1998;
[15] L. Kleinrock, "Queueing Systems, vol. 2", *Wiley, New York, 1975*