This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Performance Guarantees in Partially Buffered Crossbar Switches

Nikolaos Skalis
Computer Engineering Lab.
Delft University of Technology
Delft, the Netherlands
Email: N.Skalis@student.tudelft.nl

Lotfi Mhamdi
Computer Engineering Lab.
Delft University of Technology
Delft, the Netherlands
Email: L.Mhamdi@tudelft.nl

***Abstract*—Most of today's high capacity switches and Internet routers do not provide performance guarantees. This is attributed to their underlaying interconnection topology (i.e. the crossbar) and/or to their impractically complex scheduling algorithms. This paper derives a study for a Partially Buffered Crossbar (PBC) switch to practically provide throughput and fairness guarantees. We show how a PBC switch with a two–cell internal buffering per output and a speed–up of two can provide guaranteed performance on throughput under any admissible i.i.d input traffic. We propose a scheduling algorithm, named AF–DROP–PR, that enables us to derive bounds on the average cell latency and on the guaranteed fairness among the competing flows. AF–DROP–PR maximizes the total throughput while guaranteeing service fairness among all flows. We explore how the allocation bandwidth of individual flows is traded–off with the overall fairness index. We conjecture that the PBC switch, with its pipelined and distributed AF–DROP–PR scheduling, is attractive not only for providing performance guarantees, but also for its high capacity and low cost.**

I. Introduction

The Internet is widely considered the most reachable platform for the network infrastructure. Whereas the availability of unlimited bandwidth has triggered a plethora of services like file sharing and streaming media, there is no getting away from the fact that the Internet is based on statistical multiplexing which is facilitated through the packet mode or packet oriented communication. The statistical multiplexing principle implies the sharing of a link that is able to adapt in some way to the instantaneous traffic demands of the nodes connected to that link. Since resources are shared, traffic bandwidth must be able to be allocated on demand and fairly between different users of that bandwidth. Given that service providers wish for high capacity routers that provide guaranteed performance and that the latest backbone routers are designed to scale well [1], researchers are continually exploring faster switching technologies.

Routers usually employ a crossbar switching fabric, as it is non-blocking, memoryless and introduces no delay. The throughput of a switching architecture is mainly affected by the queueing architecture and the scheduling algorithm. Reviewing the buffering strategy in crossbar switches; in an output–queued (OQ) switch, packets are immediately forwarded to the destined output ports once they arrive at the inputs. Since in an OQ switch each output port has a memory, there is no contention from different outputs, the OQ switch has better quality-of-service (QoS) control ability, which comes at the cost of a memory speed constraint. Each memory must have an access rate equal to $(N + 1)$ times the line rate in order to cater for N writes and one read per timeslot, limiting, thus, the switch size.

Packet switches based on an input–queued (IQ) crossbar architecture with virtual-output-queueing (VOQ) are attractive for use in high speed networks, as the bandwidth of the input buffers is twice the line rate (at most one packet can be transferred to an input and at most one buffered packet can be transferred through the crossbar). So, in IQ switches, the switching fabric and the input line interface can operate at a rate that does not grow with the switch size. An unbuffered IQ switch requires a centralized scheduler to resolve two main blocking problems, namely input and output contention. Input contention results from the constraint that an input can send at most one packet every time slot. Similarly, output contention arises from the constraint that an output can receive at most one packet every time slot. These blockings make the task of the scheduler complex and the packets delay unpredictable, as the scheduler implements complex stable marriage algorithms [3].

The existence of crosspoint queueing [4] relaxes the output contention constraint, making the scheduling task much simpler. Buffered crossbars (CICQ) use distributed and independent schedulers (one per input/output port) to switch packets from the input to the output ports of the switch that do not have to resolve two constraints in a timeslot. A scheduling cycle consists of input scheduling, output scheduling and flow control to prevent crosspoint buffer overflow. The scheduling simplification comes at the expense of a costly crossbar, which is hard to scale, as the crossbar has to contain N^2 crosspoint buffers, where N is the number of input/output ports of the switch. The number of crosspoint buffers grows quadratically with the switch size and linearly with round trip delays [2]. This makes buffered crossbar switches highly expensive and hence less appealing.

The partially–buffered–crossbar (PBC) switch [5] was designed to be the best compromise between unbuffered crossbars and fully buffered crossbars. First, it overcomes the high cost of fully buffered crossbars that use N^2 internal

978-1-4244-5637-6/10/$26.00 ©2010 IEEE

buffers, by using a low number of internal buffers (B) per output irrespective of N. Second, it overcomes the scheduling complexity experienced by unbuffered crossbars by means of distributed and pipelined scheduling algorithms, whereas the input scheduling phase resembles a scheduling cycle in unbuffered crossbars, as it is based on request–grant–accept handshaking protocol.

In this paper, we describe how a PBC switch with only a two–cell internal buffering per output port coupled with an adaptive frame–based scheduler can achieve 100% throughput, guarantees a minimum level of fairness that a flow perceives. We show the trade–off between the Weighted–Max–Min–Fairness (WMMF) rates and the average cell latency compared to an OQ switch. Because we do not require a centralized scheduler and we do not impose any memory speed constraint with respect to the switch size, the proposed switching architecture and scheduling mechanism that allocates the available bandwidth with fairness in mind are considered practical. No commercial backbone router today can make hard guarantees on throughput [6].

The remainder of the paper is organized as follows; Section II presents the related work. In Section III, we describe the need for a speed–up of two in order to provide throughput guarantees. In Section IV, we show why a two–cell internal buffering per output suffices for a PBC switch to be work–conserving. In Section V, we propose a scheduler optimized for the switching architecture in question and in Section VI we discuss its fairness properties. Next, in Section VII we briefly discuss the implementation issues. Finally, Section VIII concludes the paper.

II. Related Work

Maximum–size–matching (MSM) finds the match containing the maximum number of edges, thus it can maximize the instantaneous bandwidth of the switch. It has been proved [7] that MSM can lead to instability and unfairness (if ties are broken randomly). In [8], an MSM scheduling algorithm that employs a weighting scheme, called Longest Port First (LPF), achieves 100% throughput in an IQ switch. The author in [9] devised a scheduler that uses the concept of adaptable-frame size coupled with round-robin arbitration for a buffered crossbar with one-cell crosspoint buffers. This scheme does not compare any weights or priorities and is able to provide nearly 100% throughput under the unbalanced traffic model. The characteristic of this scheduler is that it is based upon maintaining a balance between two competing interests; trying to maximize total throughput, while at the same time allowing all flows at least a minimal level of service. A study of the fairness properties and the factors that affect the stabilization delay after a change in the offered load and the weight factors in buffered crossbars is presented in [10]. In [11][12] rate and delay guarantees are provided using a complex and rather impractical scheduler. Dai and Prabhakar [13] studied how guaranteed performance on throughput can be obtained using a centralized MSM algorithm and a speed–up of two.

Our work differs from previous art both at the architectural and the scheduling level. The PBC architecture relies on separate internal buffers per output port interconnected by parallel multiplexers, dropping this way the requirement of an expensive shared memory (in terms of the memory access rate) per output. On the scheduling level, the synchronization between the grant and the accept schedulers is avoided by means of a distributed and pipelined scheduling algorithm. While emulating a PIFO–OQ switch is usually the way to show that a switch can provide guarantees, this is accomplished either by using a complex scheduler or by using an excessive memory bandwidth inside the crossbar. Our results show, that the underlying switching architecture is capable of achieving the same average cell latency and be completely fair, as an OQ switch but, the average cell latency must be traded–off with the optimal WMMF rates of the flows by regulating the amount of service that a flow receives.

III. A PBC switch with a speed–up of two

We consider $N \times N$ packet switches where arriving variable-size packets are segmented into fixed-size cells and reassembled back into packets upon their exit. We assume the traffic is Bernoulli independently and identically distributed (i.i.d.) and no input or output is oversubscribed. This section shows why the PBC architecture achieves 100% throughput and approximately the same average cell latency as an OQ switch with a speed–up two.

A. The Switch Model

Each input port contains N VOQs, one per output port. The crossbar fabric contains a small number of internal buffers (B) organized per output. The building blocks of a PBC switch are the input scheduler (IS), the grant scheduler (GS), the output scheduler (OS), the credit queue (CQ) and the grant queue (GQ), as depicted in Fig.1. Each input i maintains an input scheduler. IS_i is responsible for transferring the cells from the linecard to the internal buffers. Each output has a grant scheduler, GS_j, that tracks the grants an output sends and together with the IS_i are responsible for the flow control of the fabric 's internal buffers. The communication mechanism between the IS_i and the GS_j is the grant queue, GQ. $GQ_{ij} = 1$ when GS_j sends a grant to IS_i and when the IS_i accepts the grant then $GQ_{ij} = 0$. A credit queue with B entries is maintained per output, CQ_j, that tracks the availability of the internal buffers of that output. CQ_j is decremented whenever a grant is sent to an input, and incremented during output scheduling. Lastly, the output scheduler OS_j arbitrates the departure of cells from the internal buffers to the output j in a FCFS manner.

The succession of events during one timeslot in the scheduling process of a PBC switch is shown in Fig.1.

Note that the input schedulers (IS_i and IS_N) decisions (VOQ_{ik} and VOQ_{N1}) are based on past grant outcomes, excluding the current shown grant decisions (VOQ_{ij} and VOQ_{Nj}). Decoupling the input scheduling from the grant scheduling allows a two-stage pipelined arbitration, avoiding

978-1-4244-5637-6/10/$26.00 ©2010 IEEE

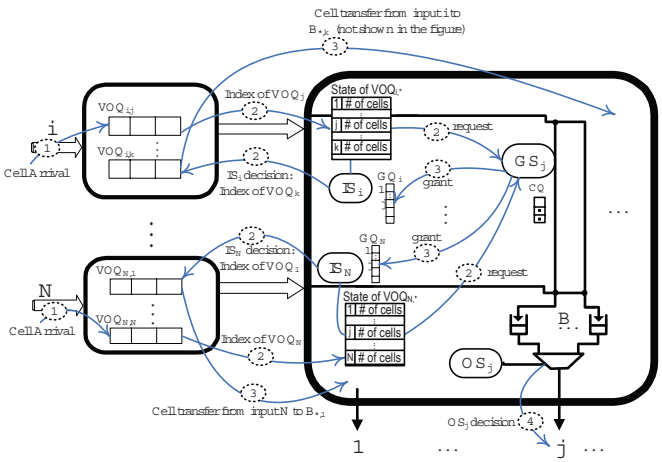

Fig. 1. The Partially Buffered Crossbar Switching Architecture [5].

the need for synchronized coordination between the grant and the accept schedulers on a timeslot basis [5].

B. On The Speed–up Required For Achieving 100% Throughput

An OQ switch is able to provide QoS by scheduling the departure of cells accordingly to latency constraints and ensuring that its outputs never idle (given that there are cells destined to them). We will require that the solution on the speed–up; 1) allows a PBC switch to achieve approximately the same average cell latency as an OQ switch and 100% throughput, 2) this is accomplished for any arbitrary input traffic pattern and 3) is independent of the switch size.

Theorem 1: A PBC switch can achieve 100% throughput with a speed–up of two for any Bernoulli i.i.d. admissible traffic.

Proof: Similar to [6], we define a super queue, $SVOQ_{ij}$ to track the evolution of each VOQ_{ij} . Let $SVOQ_{ij}$ denote the sum of the cells waiting at input i, and the cells destined to output j, comprised by the cells that reside in VOQ_{ij} and in B_j.

$$SVOQ_{ij} = \sum_k VOQ_{ik} + \sum_k (VOQ_{kj} + CQ_j)$$

What has to be shown is that the expected value of every super queue is bounded. Let 's assume, for now, that if VOQ_{ij} is not empty and j is an output with empty internal buffers ($CQ_j = B_j$), input i will send the HoL cell of VOQ_{ij} to the internal buffers of output j. We'll drop that assumption in the next section. There are two cases :

1) $0 < CQ_j \leq B_j$: output j will receive a cell, so $\sum_k (VOQ_{kj} + CQ_j)$ decreases by 1.
2) $0 \leq CQ_j < B_j$: Input i will send a cell, so $\sum_k VOQ_{ik}$ decreases by 1.

The number of cells sent to B_j during a timeslot is equal to $(B_j - CQ_j)$, so $\sum_k VOQ_{kj}$ decreases by $(B_j - CQ_j)$but CQ_j increases by the same quantity. Hence, $\sum_k (VOQ_{kj} + CQ_j)$ does not change after the input scheduling phase. During the output scheduling phase, output j will receive one more cell and $SVOQ_{ij}$ will decrease by two per timeslot. It is because the expected change in $SVOQ_{ij}$ is negative over a timeslot, given the traffic is admissible and Bernoulli i.i.d., the expected value of $SVOQ_{ij}$ is bounded. ■

It is important to mention at this point that the round–trip delay (RTT) is defined as the time from when a credit is generated until it is consumed by an input port and its associated cell is transferred to the internal buffer. Note that with our PBC request–grant protocol, only one RTT window is enough to achieve full-line rate to any requesting input. This is because of the shared credit queue (CQ) maintained by each grant scheduler and controlling the access from all inputs. As a result, the amount of buffering per output is required to be just $\geq B \times RTT$, and each output j must have an internal buffer of size B_j greater or equal to one RTT worth of cells. Next, we'll show that two cell times is the delay of a request going through both credit and grant scheduling.

IV. A WORK–CONSERVING PBC SWITCH

For a switch to be work–conserving, what is necessary is to ensure that its outputs will always serve cells as long as there are cells destined to them. Cell delay on average, is minimized, since cells leave earlier in a work–conserving switch than in any other switch. In the following, we'll find the conditions under which a PBC switch is work–conserving. In the context of switching theory, the pigeonhole principle, seminally described in [14], will be used in order to dictate these conditions. Note that, in a work–conserving switch the order of departure of cells is insignificant.

Theorem 2: A PBC switch is work–conserving with a total memory bandwidth of $2 \times N \times linerate$.

Proof: (Using constraint sets.) Let's denote as decision slots to be the slots that comprise a timeslot (if there are N cells to be scheduled, N decisions have to be made). If we denote the speed–up with S, the total memory bandwidth is $S \times N \times linerate$, where $S = 2$ as shown in the previous section. We define two constraint sets; when cells are written/read to/from the internal buffers of an output j (B_j).

- B_j Write Set (B_jWS) : The internal buffers of output j are busy for $\lceil B_j/S \rceil$ timeslots, when a cell is written into B_j. Let B_jWS be the set of internal buffers that store a new cell. In other words, B_jWS is the set of internal buffers that have initiated a write operation in the previous $\lceil B_j/S \rceil - 1$ decision slots. Hence, $|B_jWS| \leq \lceil B_j/S \rceil - 1$.
- B_j Read Set (B_jRS) : The internal buffers of output j are busy for $\lceil B_j/S \rceil$ timeslots, when a cell is read from B_j. Let B_jRS be the set of internal buffers that read a new cell. In other words, B_jRS is the set of internal buffers that have initiated a read operation in the previous $\lceil B_j/S \rceil - 1$ decision slots. Hence, $|B_jRS| \leq \lceil B_j/S \rceil - 1$.

Assuming that VOQ_{*j} have cells destined to output j, and the same case holds for all outputs. To keep all outputs busy,

when input scheduling, the internal buffers of the switch must meet the following constraints;

1) The internal buffer B_j must not be busy writing a cell. So $B_j \notin B_jWS$.
2) The internal buffer B_j must not be busy reading another cell; i.e., $B_j \notin B_jRS$.

Choosing an internal buffer B_j means that the following constraints must be met;

$$B_j \notin B_jWS \wedge B_j \notin B_jRS$$

A sufficient condition to satisfy this is :

$$B_j - |B_jWS| - |B_jRS| > 0 \Longrightarrow B_j - 2(\lceil B_j/S \rceil - 1) > 0$$

This is satisfied if $B_j \geq 2$. ■

The latter observation is confirmed by the fact that although the two pipelined scheduling phases in the PBC architecture are performed in parallel, phase 2 always has to wait 2 clock cycles in order to get the number of the remaining credits from phase 1.

V. The adaptive frame prioritized DROP scheduler (AF–DROP–PR)

We propose a round–robin frame-based input scheduling algorithm. The output scheduling used throughout this paper is based on FCFS policy. In DROP–PR [5], the grant pointers are initially set to different positions and are always incremented by one, and irrespective of the accept/drop outcome. When selecting a cell for input scheduling, DROP–PR takes into account the occupancy of the internal buffers belonging to an output. Note that when a grant scheduler grants an input request, it sends back the grant with an additional priority bit (P), which informs the granted input whether or not the grant comes from an output with empty internal buffers (prioritized output). During the input scheduling phase (second pipeline stage), priority is given to grants where $P = 1$. It is this prioritization given to outputs with empty internal buffers that ensures output j will receive a cell when an input i has a cell destined to it, and thus, drops the assumption we made earlier. In this way, we manage to balance the internal buffers utilization, achieving thus higher throughput, as we give priority to the outputs that are idle at that timeslot.

We define a frame to be the number of cells in a VOQ that are eligible for scheduling. The frame size is calculated in a regressive fashion, meaning that only after the previous frame has been fully served a new frame is determined. From now on, we'll denote the frame size of a VOQ as as captured–frame size ($|CF_{ij}|$) [9]. $|CF_{ij}|$ decreases each time a cell from VOQ_{ij} is selected and it is not the last cell of the frame under service. A VOQ can be either in ON–service status or in OFF–service status. A VOQ_{ij} is in ON–service, when $|CF_{ij}| \geq 2$ and the first cell belonging to this frame has been transmitted. A VOQ_{ij} is in OFF–service, when the last cell of the frame or no cell has been sent to the internals buffers or when $|CF_{ij}| = 1$.

The output scheduler is pointer persistent, meaning that if an accepted grant g_j points to input i, it keeps pointing to same input unless VOQ_{ij} is set to OFF–service. The specification of AF–DROP–PR is described below;

Algorithm 1 AF–DROP–PR

Grant Phase :
All output pointers, g_j, are initialized to different positions.
For each output, j, do
- Set CQ_j equals number of non-full internal buffers of output j.
- Set the priority bit, P, to the logic OR of CQ_j entries.
- While there are credits in CQ_j do
  - If output j is paired with an input i (VOQ_{ij} is in ON–service status), send a grant to input i (set $GQ_{ij} = 1$ and add bit P).
  - Else, starting from g_j index, send a grant to the first input, i, that requested this output (set $GQ_{ij} = 1$ and add bit P).
  - Decrement CQ_j by one.
- If VOQ_{ij} is in OFF–service status, move the pointer g_j to location $(g_j + 1)(mod\ N)$.

Input Scheduling Phase:
All input pointers, a_i, are initialized to different positions.
For each input, i, do
- If input i is paired with an output j (VOQ_{ij} is in ON–service status), send its HoL cell to the internal buffer.
- Else, starting from a_i index, select the first non empty VOQ_{ij} for which $GQ_{ij} = 1$ and bit $P = 1$ and send its HoL cell to the internal buffer.
- If no HoL cell is selected, Then
  - Starting from a_i index, select the first non empty VOQ_{ij} for which $GQ_{ij} = 1$ and send its HoL cell to the internal buffer.
- Drop the remaining grants (reset GQ: $GQ_{i*} = 0$).
- If VOQ_{ij} is in OFF–service status, move the pointer a_i to location $(a_i + 1)(mod\ N)$.

A. Enhanced AF–DROP–PR

Although the scheme presented above is a mixture of the DROP–PR [5] and the RR–FO [9] scheduling algorithms, it provides overall good performance while the amount of service a flow receives is almost proportional to that a flow demands, it has two drawbacks; 1) it does not discriminate the inputs when accepting grants and one of them is part of virtual circuit and 2) it is not a maximal algorithm concerning the number of virtual circuits that establishes. Next, we'll describe how these limitations can be overcome in order to fully exploit the underlying switching architecture when servicing VOQs that are in ON–status and ensure that the algorithm will do its best in order a flow to be serviced exactly as much as it demands as soon as possible.

We say that a virtual circuit between input i and output j is established when VOQ_{ij} is in ON–status. Given that $B_j = 2$ and $S = 2$, the first limitation is addressed by balancing the creation of the virtual circuits so that an output j should not be able to establish two virtual circuits in one iteration of the grant scheduling phase. This is accomplished simply by dropping the rest of the inputs that received a grant. The second limitation is addressed by adopting a prioritization mechanism similar to DROP–PR, but now considering to create up to N virtual circuits among all inputs. In other words, as DROP–PR tries to keep busy all the outputs of the switch using the priority bit mechanism, the same technique is applied in order to keep busy all outputs while making certain that each output is paired

to a different input and this input—output pair is a virtual circuit.

Subsequently, we'll explore how significant is the role of the captured–frame size both in the bandwidth a flow allocates and the fairness index and in the average cell latency of the traffic flowing through the PBC switch.

VI. Bounded bandwidth allocations

A common approach to provide throughput guarantees is to show that a switching architecture along with its scheduling algorithm is capable of emulating a PIFO–OQ switch. It is because WRR/WFQ scheduling functions in a PIFO manner. Mimicking a PIFO–OQ switch means that a cell must be switched out to the output port earlier than the departure time of the counterpart cell in the OQ switch. We argue that the latter can be accomplished by, using only a two–cell amount of buffering per output port in a PBC switch with a speed–up of two and employing the AF–DROP–PR scheduling algorithm that services the flows proportionally to their demand. The cost that we have to pay though is the time window in which this is accomplished is larger compared to the work in [6]. However, the time window is controlled by upper–bounding the frame size of a VOQ, $|CF_{ij}|$. We showed that when B equals two and with a speed–up two the architecture in question achieves 100% throughput and the same average cell latency as an OQ–switch. The phenomenon according to which the fair share of a flow at the output differs from its fair share at the input, is surpassed by establishing virtual circuits for as much time as is required in order to service that flow having its fair share in mind. Moreover, the use of virtual circuits alleviated the blockings between inputs contending for the same output. In this section, we'll study the fairness properties of the proposed switching architecture and the delay bounds achieved based on the guaranteed rate per flow.

We'll adopt the definitions originally presented in [15]. In a switch where flows make unequal demands for bandwidth, fairness is measured by closeness of the allocations to respective demands. If d_i is the demand of the i^{th} user and a_i is the corresponding allocation, then, the fraction of demand of the i^{th} flow is;

$$x_i = \begin{cases} \frac{a_i}{d_i} & \text{if } a_i < d_i \\ 1 & \text{Otherwise}[1] \end{cases}$$

The flow perception of fairness equals

$$f(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{x_i}{x_f}$$

where the fair allocation mark x_f equals

$$x_f = \frac{\sum_{i=1}^{N} x_i^2}{\sum_{i=1}^{N} x_i}$$

Thus, the scheduling algorithm is (x_i/x_f) fair concerning the i^{th} flow. When the enhanced AF–DROP–PR is used as a scheduling algorithm, the actual allocation of the bandwidth refers to the captured–frame size of VOQ_{ij}, $|CF_{ij}|$.

In the following, we'll consider the allocation of the excess bandwidth.

Theorem 3: If each user is given an additional amount c of the resource, their individual perception of fairness increases and so does the overall fairness index [15].

$$f(x_1 + c, x_2 + c, ..., x_n + c) \geq f(x_1, x_2, ..., x_n)$$

Theorem 4: If a single user j is given a small allocation Δx_j without changing other allocations, the new allocation is more (less) than before iff j is a discriminated (favored) user, i.e., [15]

$$f(x_1, x_2, ..., x_{j-1}, x_j + \Delta x_j, x_{j+1}, ..., x_n) > f(x_1, x_2, .., x_{j-1}, x_j, x_{j+1}, ..., x_n) \text{ if } x_j < x_i$$

and

$$f(x_1, x_2, ..., x_{j-1}, x_j + \Delta x_j, x_{j+1}, ..., x_n) < f(x_1, x_2, .., x_{j-1}, x_j, x_{j+1}, ..., x_n) \text{ if } x_j > x_i$$

What the above theorems imply is that, as a particular flow 's bandwidth allocation is increased from zero, the fairness at first increases, and then, it reaches a maximum after which additional bandwidth allocation to the same flow has as a consequence the rest of the flows to perceive unfairness. What has been just described is the optimal WMMF allocation metric.

A minimum level of fairness is guaranteed when we restrict the individual bandwidth allocations [15].

Theorem 5: If $x_{min} \leq x_i \leq x_{max}$ for all i and $x_{min} > 0$, then the fairness is guaranteed to be above a certain lower bound, and

a. The minimum fairness occurs when a fraction of γ of the users (i.e. γ n users in all) receive x_{min} and the remaining receive x_{max}. Here

$$\gamma = \frac{K}{K+1}, \text{ where } K = x_{max}/x_{min}$$

b. Fairness

$$f \geq \frac{4K}{(K+1)^2} \quad (1)$$

All in all, upper bounding the frame size at N, for example, the average cell latency is larger by $N^2/2$ timeslots, at most, compared to that of an OQ switch. Whereas, the minimum guaranteed fairness, in the worst case scenario, is given by Eq.1. The stabilization delay depends on the frame size and the frame size difference among VOQ_{*j}, meaning that smaller frame sizes and larger frame size differences incur faster stabilization.

VII. Implementation complexity

The enhanced AF–DROP–PR scheduling algorithm adopts a round–robin based arbitration scheme and the additional hardware that requires is the captured–frame size counters and the flags needed to indicate the service status of VOQ. In addition, the two–cell internal buffering per output is sufficient to make the switch deliver high performance. In the case of IQ switches, this is to be compared with the weight-based schemes where comparisons between the contending VOQs

[1] Allocating more bandwidth to a flow than what it demands is vain.

are needed in a timeslot basis, and in the case of fully–buffered crossbars, with the requirement for the amount of memory and its memory access rate.

The PBC switching architecture has to maintain up to B separate buffers per output, which run at the same bandwidth as the external line rate. This is done in order to maintain the low memory bandwidth requirement and it is to be compared with the shared internal buffers requiring a memory access rate of NxR in [6]. However, maintaining B separate buffers per output mandates the use of up to Bx(N-to-1) parallel multiplexers per output. The demultiplexer selects an internal buffer and sends the arriving cell to that buffer, where it is queued until it departs.

Given that: 1) the crossbar chip is bound by I/O count and power consumption and not by the crosspoints logic, implying the underutilization of the crossbar chip die [6][16], 2) using the crossbar chip extra (unused) logic for either N^2 distributed or B.N shared internal memories running B times the external line speed results is excessive power consumption and can be costly and 3) on-chip wires are inexpensive [17] and therefore meeting the bandwidth goal can be achieved by adding more wires and multiplexers in parallel. We believe that the cost and feasibility of the PBC (with BxN separate internal buffers and NxBx(N-to-1) parallel multiplexers), where $B = 2$, is lower than that of traditional CICQ design (with N^2 internal buffers and Nx(N-to-1) parallel multiplexers).

VIII. Conclusion

By introducing a small amount of buffering in the crossbar and employing a frame–based scheduling algorithm, the resulting PBC switching architecture is simple as it uses distributed and pipelined schedulers and is scalable as it requires much less memory bandwidth when compared to fully buffered crossbars. Note that the memory access rate of the internal buffers is decoupled of the switch size. AF–DROP–PR was devised to fully exploit the underlying architecture, yielding full output utilization and providing WMMF bandwidth allocations in a relaxed time period that depends on the frame sizes assigned to the flows and on the frame–size differences among them. Bounds on the average cell latency and the overall fairness index were derived, revealing this way a trade–off between these two metrics.

This paper conducts the first study on how to guarantee performance in PBC switches. Given the high potential of buffered crossbars, our work addresses the scalability issue of crossbar based routers.

References

[1] H. Jonathan Chao, Bin Liu, "High Performance Switches and Routers", Wiley-IEEE Press, 2007.

[2] F. Abel, C. Minkenberg, R. P. Luijten, M. Gusat, and I. Iliadis, "A Four-Terabit Packet Switch Supporting Long Round-Trip Times", IEEE Micro vol.23, no.1, pp. 10-24, Jan. 2003.

[3] D. Gale and L. S. Shapely, "College admissions and the stability of marriage", American Mathematical Monthly, 69:9–15, 1962.

[4] M. Nabeshima, "Performance Evaluation of Combined Input- and Crosspoint-Queued Switch", Proc. IEICE Trans. Comm., vol. B83-B, no. 3, pp. 737–741, Mar. 2000.

[5] L.Mhamdi, "PBC: A Partially Buffered Crossbar Packet Switch", Computers, IEEE Transactions on , vol.58, no.11, pp.1568-1581, Nov. 2009.

[6] S. Chuang, S. Iyer, and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars", In Proc. IEEE INFOCOM 05, pages 981991, 2005.

[7] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch", IEEE Transactions on Communications, vol. 47, no. 8, pp. 1260–1267, Aug. 1999.

[8] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input queued switches", In Proc. of IEEE INFOCOM 98, volume 2, pages 792–799, San Francisco, CA, April 1998.

[9] Rojas-Cessa, R., "High-performance round-robin arbitration schemes for input-crosspoint buffered switches", Workshop on High Performance Switching and Routing (HPSR), pp. 167–171, 2004.

[10] N. Chrysos, M. Katevenis, "Transient Behavior of a Buffered Crossbar Converging to Weighted Max-Min Fairness," Inst. of Computer Science, FORTH, Greece, August 2002.

[11] S-T. Chuang, A. Goel, N. McKeown,and B.Prabhakar, "Matching Output Queueing with a Combined Input Output Queued Switch", IEEE J. Select. Areas Commun., 17, No. 6, pp. 1030–1039. A short version appears in The Proceedings of Infocom 99.

[12] S. Iyer, R. Zhang, and N. McKeown, "Routers with a Single Stage of Buffering", ACM SIGCOMM 02, Pittsburgh, USA, Sep. 2002.

[13] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup", In Proc. of IEEE INFOCOM 00, pages 556564, Tel Aviv, Israel, March 2000.

[14] Sundar Iyer, "Load Balancing and Parallelism for the Internet", Ph.D. Thesis Report, Stanford University, July 2008.

[15] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems", DEC Research Report TR-301, September 1984.

[16] L. Mhamdi, C. Kachris, and S. Vassiliadis, "A Reconfigurable Hardware Based Embedded Scheduler For Buffered Crossbar Switches", ACM/SIGDA FPGA, pp. 143–149, Feb. 2006.

[17] J. Balfour and W. J. Dally, "Design Tradeoffs For Tiled CMP On-Chip Networks, ICS, pp. 187–198, 2006.

978-1-4244-5637-6/10/$26.00 ©2010 IEEE