Size-Based Flow Scheduling in a CICQ Switch

Dinil Mon Divakaran, Fabienne Anhalt, Eitan Altman, Pascale Primet

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Size-Based Flow Scheduling in a CICQ Switch

Dinil Mon Divakaran, Fabienne Anhalt, Eitan Altman, Pascale Vicat-Blanc Primet

N° 7183

Jan. 2010

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7183--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
GRENOBLE – RHÔNE-ALPES

Size-Based Flow Scheduling in a CICQ Switch

Dinil Mon Divakaran, Fabienne Anhalt, Eitan Altman, Pascale Vicat-Blanc Primet

Thème : THCom
Équipe-Projet RESO

Rapport de recherche n° 7183 — Jan. 2010 — 10 pages

Abstract: In the context of flow-aware networking, size-based (SB) scheduling policies have been shown to improve response times of small flows, without degrading the performance of large flows. But these differentiating policies are designed for Output-queued (OQ) switch architecture, which is known to have scalability issues. On the other hand, the buffered-crossbar (BX) switch architecture is currently being pursued as a potential next-generation scalable switch architecture. This work looks into the problem of performing SB scheduling in BX switches. In particular, the design goals, with respect to each output port, are (i) to transmit high-priority packet(s) as long as there is at least one present, and (ii) to respect the FIFO order among high-priority packets. In this direction, we propose a CICQ switches using a single PIFO queue at each crosspoint to schedule packets according to the priority assigned. pCICQ-1 switch uses a simple design to guarantee that packet-priorities are respected once they are in the crosspoint queues. But it does not maintain the FIFO order of high-priority packets, besides letting a bounded number low-priority packets to depart through an output, when there are one or more high-priority packets for the same output. To solve this, we propose an enhancement in pCICQ-2 switch, that uses a sequence controller to respect packet-priorities as well as arrival order for high-priority packets.

Key-words: scheduling, switch, piority, CICQ

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Ordonnancement de flux basé sur la taille dans un switch CICQ

Résumé : Dans un contexte du traitement par flux, il a été montré que les politiques d'ordonnancement basées sur la taille (size-based (SB)) améliorent le temps de réponse des petits flux, sans dégrader la performance des autres flux. Ces politiques de différenciation sont conues pour des architectures de switch ayant des files d'attente aux ports de sortie (output queued (OQ)), et dont on sait qu'ils sont limités dans le passage à l'échelle. Comme solution à ce problème de passage à l'échelle, l'architecture de switch avec une matrice de commutation équipée de buffers aux points d'intersection (buffered crossbar (BX)) est actuellement vue comme pouvant être l'architecture des switches de demain. Ce travail examine le problème de réaliser un ordonnancement basé sur la taille dans de tels switches ayant une matrice de commutation avec des buffers. Plus particulièrement, les objectifs au niveau de chaque port de sortie sont (i) de toujours transmettre les paquets prioritaires en premier, et (ii) de respecter l'ordre FIFO entre les paquets prioritaires. Dans ce but, nous proposons un switch CICQ qui utilise une file d'attente PIFO à chaque point d'intersection, afin d'ordonnancer les paquets en fonction de leur priorité. Le switch pCICQ-1 utilise un concept simple pour garantir le respect des priorités des paquets une fois qu'ils ont atteint les files d'attentes aux points d'intersection. Parcontre, il ne maintient pas l'ordre FIFO entre les paquets prioritaires et peut admettre un nombre limité de paquets non prioritaires à un port de sortie, même s'il existe un ou plusieurs paquets prioritaires en attente dans le système et qui sont destinés à ce même port de sortie. Pour résoudre ceci, dans le switch pCICQ-2, nous proposons une amélioration introduisant un contrleur de séquence qui assure le respect d'une part des priorités des différents paquets, et d'autre part de l'ordre d'arrivée des paquets prioritaires.

Mots-clés : ordonnancement, switch, priorité, CICQ

I. Introduction

In the recent years, along with the growth of the Internet, the focus has moved from controlling and providing QoS at packet-level to, what is now known as, flow-aware networking [1], examples showing the impetus include [2] and [3]. Explicit arguments supporting the need for such a move were provided by Bonald *et al.* [4]. In this context, several works concentrated on improving flow response times using queueing models, once it was revealed that queues in the Internet can be modelled as processor-sharing (PS) queues at flow level [5]. In particular, size-based (SB) flow scheduling is advocated to improve response times of small flows without hurting the performance of large flows [6], [7], [8].

An SB scheduler, in general, gives priority to small flows. For example, in the $PS+PS$ model proposed in [8], the first θ packets of all flows are sent to a high-priority queue, and the rest are queued in a low-priority queue. The low-priority queue is served only when the high-priority queue is empty. This way, small flows with size less than or equal to θ packets complete transfers quickly. Though the two queues are PS queues at flow level, at packet level, they are FIFO queues. Therefore, packets within each queue depart in FIFO order.

Like almost all QoS guaranteeing scheduling algorithms, SB scheduling policies have also been designed to be used at an Output-Queued (OQ) switch. OQ switch has queues only at the output ports, requiring packets to be switched instantly, without any delay, on their arrivals at the input ports. As this requires the switch fabrics as well as the output buffers to run at N times the line rates (where N is the number of switch ports), OQ architecture is unable to scale up with the product of line rates and port density. Nevertheless, it is used as a reference architecture, and designers of practical switch architectures, try to 'emulate' OQ switch.

In this work, we study how SB flow scheduling can be implemented on a scalable switch architecture, namely the Combined Input-Crosspoint Queued (CICQ) architecture [9]. As the name suggests, CICQ switches have buffers at the crosspoints of a crossbar. This crosspoint-buffering decouples the inputs and outputs, thereby removing the need for complex centralized arbitration algorithms used in unbuffered crossbar switches. Therefore, such buffered-crossbar (BX) switch architecture is regarded as a potential candidate for the next generation scalable high-speed switch architecture.

SB scheduling at flow level is implemented as priority scheduling at packet level. In this framework, [10] and [11] propose solutions for supporting multiple priorities in BX switches. The former uses a single memory at each crosspoint, shared between multiple queues, thus causing buffer-hogging. The latter focuses on mapping more than two priorities on to reduced number of queues at the crosspoints. There has also been studies that showed how a BX can emulate an OQ switch [12], in addition to those that study FIFO and priority queueing policies using *competitive analysis* [13], [14]. But these works assume queues also at the output, in addition to input and crosspoint queues.

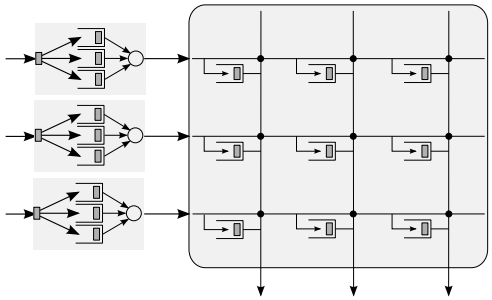

Fig. 1. Architecture of a CICQ switch

We propose a switch design for SB flow scheduling using PIFO crosspoint queues. The design objectives for such a switch are (i) to have strict priority scheduling at packet level, and (ii) to respect the FIFO order among high-priority packets. The initial simple design, pCICQ-1 switch, respects the priority of packets within an input-output pair, and also among different crosspoint queues for an output. But, under certain conditions, we observe that a low-priority packet can depart when there exists a high-priority packet in the switch for the same output. Besides, the high-priority packets destined to an output do not depart in the order of arrivals. To solve these, we propose an enhanced design, pCICQ-2 switch, that uses a sequence controller to respect the FIFO order of high-priority packets destined to the same output. In addition, we show that pCICQ-2 switch respects the priorities of all packets.

The next section briefs the necessary background. Section III describes the modification required at the input and crosspoint buffers for doing SB flow scheduling. The designs of pCICQ-1 and pCICQ-2 switches are described in sections IV and V, respectively. Section VI analyzes the two switch designs, before concluding in Section VII.

II. Background

A. CICQ switch architecture

A CICQ switch architecture is shown in Fig. 1. There are small buffers at the crosspoints, that enables running independing processors on each input, thus eliminating the need for a centralized scheduler. Besides, scheduling can be performed on variable-size packets, removing the need for packet segmentation and reassembly required in unbuffered crossbars due to synchronous scheduling [15]. Therefore internal speedup is no more required, which in turn facilitates the removal of output buffers.

CICQ switch also has VOQs at each input. A VOQ at input (port) i for output (port) j is denoted as $\text{VOQ}_{i,j}$. Similarly, the buffer at the crosspoint formed by input i and output j is denoted as $\text{B}_{i,j}$.

B. SB flow scheduling

Though literature has quite a few ways of prioritizing small flows so as to improve their response times, we focus on the $PS+PS$ model proposed by Avrachenkov *et. al.* [8]. As said before, the $PS+PS$ model assumes an OQ switch. The queue

P^h	A high-priority packet
P^l	A low-priority packet
$A(P)$	Arrival time of packet P
$D(P)$	Departure time of packet P
$\text{VOQ}_{i,j}$	VOQ for output j at input i
$\text{VOQ}^h_{i,j}$	A high-priority VOQ for output j at input i
$\text{VOQ}^l_{i,j}$	A low-priority VOQ for output j at input i
$\text{B}_{i,j}$	Queue at crosspoint $CP_{i,j}$
R_j	Register at $CP_{i,j}$
ASH_j	Arrival sequence of high-priority packets for output j

TABLE I
TABLE OF NOTATIONS (I).

at the output is divided into two: a high-priority queue, and a low-priority queue. Packets destined to an output, as they arrive, are classified into either of these queues depending on the threshold θ. The fist θ packets of each flow are 'tagged' as high-priority packets, and they go to the high-priority queue, and the other packets are tagged as low-priority packets, and are sent to the low-priority queue. As for the scheduler, the low-priority queue is served only when the high-priority queue is empty. At the packet level, this becomes strict priority scheduling. Since the design is for OQ switches, packets of a priority level depart the queue in FIFO order.

III. Components for buffering packets

In this section, we describe the modifications needed at the input and crosspoint queues of a CICQ switch, to facilitate SB flow scheduling. These components for buffering packets are the same for both pCICQ-1 and pCICQ-2 switches.

Though the proposed switches can operate on variable-size packets, for making the analysis simpler, we assume packets are of fixed-size, and arrive at the beginning of a time-slot. A time-slot is the time required to transfer the packet at the switch line-rate. Hence, time is taken as discrete slots. At any given time-slot t, not more than one packet can arrive at any input, and not more than one packet can depart through an output. Similarly, the time to transfer a packet from the input to the crosspoint is also one time-slot. The traffic is assumed to be admissible in the long run. The arrival time and departure time of an arbitrary packet P are denoted by $A(P)$ and $D(P)$ respectively. Table I gives the list of notations used here.

A. VOQ

The first step is to divide each $\text{VOQ}_{i,j}$ into two queues: $\text{VOQ}^h_{i,j}$ and $\text{VOQ}^l_{i,j}$. That is, there are $2N$ VOQs at each input. Packets of $\text{VOQ}^h_{i,j}$ are high-priority packets destined to output j at input i; and for the same input-output pair, $\text{VOQ}^l_{i,j}$ holds the low-priority packets. On arrival, a packet is sent to $\text{VOQ}^h_{i,j}$, if the ongoing size of its flow is less than or equal to θ; or else, it is sent to $\text{VOQ}^l_{i,j}$ (assuming packets of the same flow arrive at the same input). Every VOQ is a FIFO queue.

Each input i has an associated scheduler, IS_i, that selects a VOQ to be dequeued at every scheduling instance. We elaborate more on this later. At this point, it suffices to say

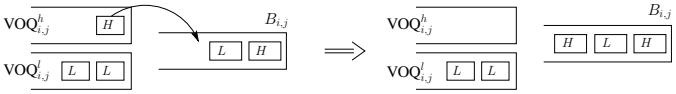

Fig. 2. Input scheduling, and queueing at the crosspoint FIFO queue

that a high-priority VOQ will be selected for dequeueing as long as there is a high-priority packet at the input.

B. Crosspoint queue

There are N^2 crosspoints queues. If these crosspoint queues are FIFO queues, then it is possible that a high-priority packet might be blocked by a low-priority packet in front of it. For example, consider a time when there are only low-priority packets at $\text{VOQ}_{i,j}$. When $\text{VOQ}_{i,j}$ is selected for dequeueing, the packet at the head of $\text{VOQ}^l_{i,j}$ is sent to the corresponding crosspoint queue $\text{B}_{i,j}$. While the packet is still in $\text{B}_{i,j}$, a packet of high priority arrives at input i for the same output j. The next time IS_i selects the j^{th} VOQ, the high-priority packet will be sent to $\text{B}_{i,j}$. But, this high-priority packet will still be behind the low-priority packet that was scheduled earlier, as the crosspoint queues are FIFO queues. Hence, the low-priority packet will depart the switch before the high-priority packet. This is illustrated in Fig. 2, where L is a low-priority packet, and H is a high-priority packet.

We propose to use a Push-In-First-Out (PIFO) queue at the crosspoints [16]. With a PIFO queue, an arriving packet can be 'pushed' into an arbitrary position depending on a criteria. The dequeueing operation is performed at the queue head. By using such a queue, one can implement many QoS guaranteeing policies, as most of them determines a specific departure order of packets in the queue; and this can be achieved using a PIFO queue. This also removes the need for multiple queues for multiple priority levels, at the crosspoints. Though a PIFO queue is more complex than a FIFO queue, recent research works have shown that a highly-scalable PIFO architecture can now be realized in hardware [17].

To make use of the PIFO queue, we need one bit of information in every packet: a bit to denote if the packet is of high priority or low priority. An arriving packet is classified as high priority or low priority; this is indicated in its header using a one-bit information (for example, '1' indicating high priority and '0' indicating low priority). The input scheduler, as usual, selects the head-of-the-line (HoL) packet from $\text{VOQ}^h_{i,j}$, or from $\text{VOQ}^l_{i,j}$ if the former is empty, and sends it to be queued at the crosspoint $\text{B}_{i,j}$.

Definition 3.1: PIFO queueing policy

The crosspoint logic reads the priority bit; and enqueues the packet at the end of the queue if it is of low priority, or else, the packet is queued behind all the high-priority packets, but ahead of all low-priority packets. For dequeueing, the HoL packet is read out.

This ensures that, within an input-output pair, all packets of the same priority, leave in the order they arrived. Observe that this can lead to drops at crosspoints, which otherwise can

be avoided by coordinating with the input[1]. If the crosspoint queue is full with low-priority packets, an arrival of a high-priority packet 'knocks off' the low-priority packet in the tail. Though this avoids the buffer hogging by low-priority packets, dropping packets in the crosspoints is not advisable. The frequency of such drops can be avoided during low-traffic regime, by input-crosspoint coordination and use of reserved space. During times of high load, the 'knocking off' behaviour is justified in SB policies.

We revisit the previous example, this time with PIFO queues at the crosspoints. The example is illustrated in Fig. 3.

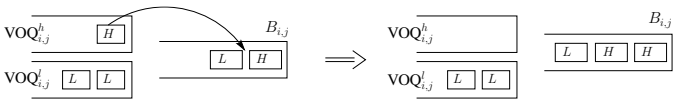

Fig. 3. Input scheduling, and queueing at the crosspoint PIFO queue

IV. pCICQ-1: CICQ USING PRIORITY-INDICATOR VECTOR

There are two schedulers: input scheduler and output scheduler. The input scheduler IS_i associated with each input i, selects and dequeues one of the VOQs at each scheduling instance. Output scheduling refers to the selection of one of the N crosspoint queues to dequeue for one particular output. OS_j is the output scheduler for the output port j.

A. Output scheduling

Once a queue $\mathrm{B}_{i,j}$ is selected by OS_j, the packet at the head of $\mathrm{B}_{i,j}$ is dequeued and transmitted through the output line. The scheduling policy used by the output scheduler determines the order in which the crosspoint queues are selected, and hence, also determines the order in which packets depart the switch. Note that, since the queues at the crosspoints are PIFO queues, no low-priority packet from $\mathrm{B}_{i,j}$ will depart before the departure of high-priority packets in B_i (if any). But depending on the scheduling policy, a low-priority packet from $\mathrm{B}_{i,j}$ can depart before a high-priority packet from $\mathrm{B}_{k,j}$, where $k \neq i$. This, for example, can take place if a simple round-robin (SRR) scheduler is used for output scheduling. In SRR scheduling, if the currently served queue is i, the next queue to be served is $(i+1)$ mod N. Fig. 4 shows one such scenario using SRR scheduling, where the scheduler does not respect the priority of packets. If the SRR scheduler dequeues in the order $\mathrm{B}_{1,j}, \mathrm{B}_{2,j}, \mathrm{B}_{3,j}, \mathrm{B}_{4,j}$, then the low-priority packet at the head of $\mathrm{B}_{2,j}$ will depart before the high-priority packet at the head of $\mathrm{B}_{3,j}$.

The mis-ordering of priority packets at the output caused by the simple RR scheduler can be avoided by using a one-bit priority indicator, $\mathrm{PI}_{i,j}$, for each crosspoint. The logic associated with the crosspoint $CP_{i,j}$ updates $\mathrm{PI}_{i,j}$, when a packet departs from $\mathrm{B}_{i,j}$. If the HoL packet is a high-priority packet, $\mathrm{PI}_{i,j}$ is set to '1', or else, it is set to '0'.

[1]The drops at crosspoints can be avoided, if the crosspoint logic notifies the input of every packet departure from that crosspoint, and the input uses this information during scheduling.

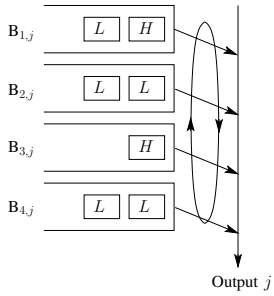

Fig. 4. Simple round-robin scheduler for an output port

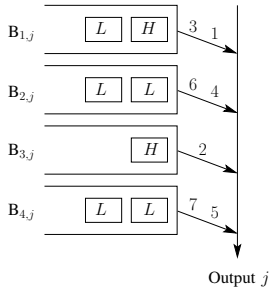

Fig. 5. Output scheduling using $\mathbb{PI}$ vector and with $\chi_1 = \chi_2 =$ RR

Let $\mathbb{PI}_j$ denote the vector $< \mathrm{PI}_{1,j}, \mathrm{PI}_{2,j}, \ldots, \mathrm{PI}_{N,j} >$. After dequeueing a packet, scheduler OS_j reads the $\mathbb{PI}_j$ vector to take the scheduling decision.

Definition 4.1: Output scheduling using $\mathbb{PI}$ vector

At each scheduling instance, OS_j, selects a queue from a set $\mathcal{H}_j$ based on a criterion χ_1, where $\mathcal{H}_j = \cup_{i=1}^{N}\{B_{i,j}|PI_{i,j} = 1\}$. If $\mathcal{H}_j$ is empty, it selects a queue from another set $\mathcal{L}_j$ based on a criterion χ_2, where $\mathcal{L}_j = \cup_{i=1}^{N}\{B_{i,j}|PI_{i,j} = 0\}$. The scheduler dequeues from the selected crosspoint queue.

The criteria χ_1 and χ_2 to select a queue from a set of queues can be, for example, *queue holding Oldest-HoL-Packet first*, or *Longest-Queue-First*, or *round-robin within the set of queues*. Fig. 5 illustrates the scheduling, where $\chi_1 = \chi_2$, and the criterion used is round-robin within a queue-set. The order of departures $(\mathrm{B}_{1,j}, \mathrm{B}_{3,j}, \mathrm{B}_{1,j}, \mathrm{B}_{2,j}, \mathrm{B}_{4,j}, \mathrm{B}_{2,j}, \mathrm{B}_{4,j})$ is shown on the directed lines, for a period where there is no new arrival.

Within a priority level, even if the packets come from different inputs, once they are in the crosspoint queues, the priority scheduler ensures that, a low-priority packet does not leave when there exists at least one high-priority packet in the crosspoint queues.

B. Input scheduling

The input scheduler should complement the output scheduler with an appropriate policy to minimize the number of low-priority packets departing at an output when there are one or more high-priority packets destined to the same output. As a necessary criterion, the input scheduler should schedule a high-priority VOQ as long as there is a high-priority packet at the input. The decision regarding which among the high-priority VOQs should be dequeued, can be based on some criterion like discussed above.

Definition 4.2: Input scheduling policy

IS_i selects a VOQ for dequeueing, from a set of non-empty high-priority VOQs ($VOQ^h_{i,*}$), based on a criterion χ_1. If there is no such (non-empty) high-priority VOQ, then the scheduler selects a VOQ from the set of low-priority VOQs ($VOQ^l_{i,*}$), based on a criterion χ_2. The scheduler dequeues from the selected VOQ.

The criterion χ_1 influences the delay for an HoL packet.

Definition 4.3: Δ^h: Maximum waiting time (in number of time-slots) for an HoL packet at a high-priority VOQ

Δ^h is the maximum time (in number of time-slots) an HoL packet of a high-priority VOQ waits to be enqueued in the corresponding crosspoint queue. This time is dependent on the criterion χ_1, and also the time needed to transfer the packet from the input to the crosspoint queue.

C. Discussion

The pCICQ-1 switch has a simple design that respects the priority of packets in the crosspoint queues. The additional memory required when compared to a CICQ switch is only N bits for the priority indicators. The pCICQ-1 switch still does not guarantee that a low-priority packet will not leave through an output port, when there exist one or more high-priority packets destined to the same output in the switch. For example, consider the following scenario. If there are only low-priority packets destined to an output j, the output scheduler will send those packets one after the other. A new high-priority packet arriving at the input for output j at time t, has to wait for at least one time-slot to reach the the crosspoint, during which a low-priority packet will depart through output j. Note that, such a scenario will not occur in an OQ switch, as the high-priority packet will be available at the output queue instantly at the arrival time t, thus allowing the transfer of no more low-priority packet (until the high-priority packet is transferred).

Besides, the output scheduling using $\mathbb{PI}$ vector does not meet the second design goal of maintaining the FIFO order among high-priority packets destined to the same output. In the following section, we propose means to achieve this.

V. pCICQ-2: CICQ using Sequence Controller

Observe that, what the output scheduler requires for dequeueing high-priority packets in the order of arrivals, is the relative order of arrivals of those packets, and not the exact arrival timestamps. Therefore, a simple method is maintaining the order of packet arrivals at the output scheduler using an identifier for each input. Interestingly, the output scheduler does not have to bother about the arrival sequence of packets belonging to a single crosspoint queue. The input scheduler along with the PIFO queueing policy does the necessary ordering of packets within a crosspoint queue, from which an HoL packets are dequeued.

We propose the following solution. There is a *sequence controller*, SC_j, for each output j. It maintains a FIFO queue called ASH_j that holds the input port identifiers to reflect the *A*rrival *S*equence of *H*igh-priority packets destined to output j. Each input card, on the reception of a high-priority packet, informs the SC corresponding to the output, the packet-arrival event. This can be achieved in a distributed way, for scalability.

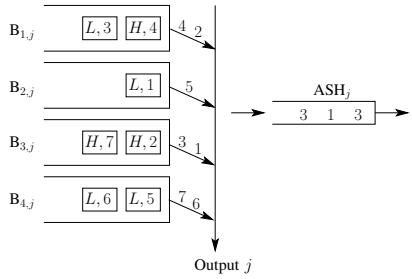

Fig. 6. Scheduling using Sequence Controller for an output port

Let δ be an interval less than or equal to a time-slot.

Definition 5.1: Two-phase SC operation

Every crosspoint $CP_{i,j}$ maintains a one-bit register $R_{i,j}$. Input i informs $CP_{i,j}$, when a high-priority packet arrives for output j. $CP_{i,j}$ updates $R_{i,j}$ — ensuring $R_{i,j}$ is '1' if there was a signal from the input, or '0' otherwise. The above operations are done in the first $\delta/2$ period. In the second $\delta/2$ period, SC_j reads these registers ($R_{*,j}$) for arrival-event notification. For each i such that $R_{i,j} = 1$, it enqueues the identifier i into ASH_j queue.

The two-phase SC operation is done in every time-slot. Observe that the SC operation can be done while the packet is being buffered at the input, as the input needs only the header information to decide the priority of the packet.

A. Output scheduling

Definition 5.2: Output Scheduling using SC (OSSC)

If ASH_j is not empty, OS_j dequeues from ASH_j, the index of the crosspoint queue to be dequeued. If ASH_j is empty, it selects one of the crosspoint queues based on a criterion χ. It then dequeues from the selected crosspoint queue.

We illustrate the working using an example. Consider Fig. 6. Each packet is represented as $\ll \pi(P), A(P) \gg$ where $\pi(P) \in \{L, H\}$ indicates the priority, and $A(P)$ is the arrival time of packet P [2]. The arrival times show the arrival order of the packets currently in the crosspoint queues. This means, packet $\ll L, 1 \gg$ in $\text{B}_{2,j}$ is the first to arrive at the switch among all other packets currently in the crosspoints $\text{B}_{*,j}$. Also, observe the arrival order of packets in $\text{B}_{1,j}$. The second packet, $\ll L, 3 \gg$, of $\text{B}_{1,j}$ arrived at the switch before the first packet $\ll H, 4 \gg$. But as the latter has higher priority than the former, it is pushed ahead of the low-priority packet.

There are three high-priority packets in the crosspoints, one at $\text{B}_{1,j}$ and two at $\text{B}_{3,j}$; and their relative arrival order is stored in ASH_j using the corresponding input port identifiers. This means, among the high-priority packets in the switch, the first packet that arrived is queued in $\text{B}_{3,j}$, the second in $\text{B}_{1,j}$, and the third in $\text{B}_{3,j}$. The scheduler in this case, first schedules $\text{B}_{3,j}$, then $\text{B}_{1,j}$, and finally $\text{B}_{3,j}$. During this time, if no high-priority packet arrived to the switch, the scheduler

[2] Arrival times here, are not used for scheduling, but only for explanation.

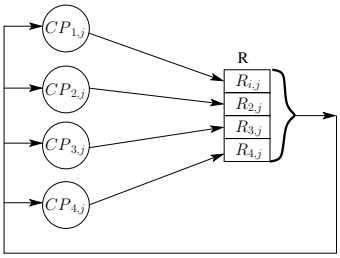

Fig. 7. Independent updates and retrievals of registers by crosspoints

will select one of the crosspoint queues based on the criterion for dequeueing low-priority packets. In the example here, we assume round-robin policy when there is no high-priority packet in the crosspoints. Hence, the order of dequeueing of low-priority packets is shown as $\mathrm{B}_{1,j}, \mathrm{B}_{2,j}, \mathrm{B}_{4,j}, \mathrm{B}_{4,j}$.

B. Input scheduling

For the OSSC to be work-conserving, the queues whose identifiers are at the head of the ASH queue should be non-empty. That is, if OS_j dequeued i from its ASH_j, then the input scheduler IS_i should already have sent the corresponding high-priority packet to have it queued at (the head of) $\mathrm{B}_{i,j}$. This can be achieved using the following.

During the two-phase SC operation, at the end of first $\delta/2$ time period, crosspoints $CP_{*,j}$ read the set of registers $\mathrm{R}_{*,j}$ (refer Fig. 7). Then, using the length of ASH_j ($len(\mathrm{ASH}_j)$) at the beginning of the current two-phase period and values in the register set, each crosspoint $CP_{i,j}$ computes the 'dequeue time' DT for the packet P_h.

Definition 5.3: Dequeue time of a packet, DT

Dequeue time of a packet P_h that arrives at time t at input i for output j,

$$DT(i,j,t) = len(ASH_j(t)) + \sum_{k=1}^{i-1} R_{i,j}(t)$$

where $R_{i,j}(t)$ is the value of $R_{i,j}$ at time t.

If multiple inputs had arrivals of high-priority packets for an output at the same time, the following rule is adopted — the one with the lower input port identfier leaves earlier. This is computed by each crosspoint logic independently. For example, let $len(\mathrm{ASH}_j)$ be M at the time of arrivals of high-priority packets at inputs $1, 3$ and 4 for output j. Then, $CP_{1,j}, CP_{3,j}$ and $CP_{4,j}$ will respond to inputs $1, 3$ and 4, respecitively, with $M, M+1$ and $M+2$.

Next, we define a time-line for every high-priority packet P^h on its arrival.

Definition 5.4: Time-line of a high-priority packet: τ

For a high-priority packet P^h that arrived at input i for output j, $\tau(P^h, A(P^h)) = DT(i,j,A(P^h))$, obtained from $CP_{i,j}$. Time-line of a packet decreases one per time-slot. For completeness, if P^h has not arrived until time t, $\tau(P^h,t) = \infty$; similarly, if P^h has departed before time t, $\tau(P^h,t) = -1$.

The low-priority packets are transferred only when there is no high-priority packet in the input. Hence, the selection of a low-priority VOQ can be based on some criterion, e.g. LQF.

Consider the special case when $CP_{i,j}$ responds with $DT = 0$ for a high-priority packet P^h. This means that, P^h is the next packet to be switched out through the output j. But, the packet will take at least one time-slot to be switched from the input to the crosspoint buffer, thus delaying its departure. A solution to this is to use 'cut-through' switching, where a packet from the input can go directly to the output, without being queued in the intermediate crosspoint buffer.

Observe that, at any time-slot, there is not more than one arrival at an input. Though there can be multiple inputs which have arrivals at the same time for the same output, only one will get a response $DT = 0$ (as only one packet can be transmitted through the output during one time-slot). Therefore, at any time-slot, there is not more than one cut-through switching associated with each input and each output.

Next, we define the input scheduling policy. Before that, we define the time-line of $\mathrm{VOQ}^h_{i,j}$ as the time-line of the HoL packet of $\mathrm{VOQ}^h_{i,j}$.

Definition 5.5: Input scheduling policy

At every scheduling instance, the scheduler IS_i at input i does either of the following:

- *If there arrived a high-priority packet P^h, such that $\tau(P^h, A(P^h)) = 0$, the packet is sent directly to the output using cut-through switching;*
- *Else, from among $\mathrm{VOQ}^h_{i,*}$, the VOQ with minimum time-line is selected. In the absence of high-priority packets at the input, the scheduler selects a VOQ from the low-priority VOQs based on a criterion χ. The scheduler dequeues the HoL packet of the selected VOQ.*

We assume the input scheduling achieves the following:

Assumption 5.6: Switching high-priority packets at inputs

For every high-priority packet P^h with $A(P^h) = t$,

- *If $\tau(P^h, t) = 0$, the input scheduler uses cut-through switching to send it directly to the output;*
- *If $\tau(P^h, t) \neq 0$, the input scheduler completes transfer of the packet to the corresponding crosspoint queue latest by time $\hat{t}$, such that $\tau(P^h, \hat{t}) \geq 0$.*

The above assumption is valid for a work-conserving input scheduler or and infinite size crosspoint queues.

C. Discussion

As we will show in the next section, pCICQ-2 respects the priority of packets conditioned on the Assumption 5.6, and also maintains FIFO order for high-priority packets.

The disadvantage of this mechanism is the size requirement for the ASH queue. It should have enough size to hold the indicators for the maximum number of high-priority packets that can be in a switch for a particular output. If each input together with its corresponding crosspoint queues can have a maximum of M packets for each output at any instance in an $N \times N$ switch, then the size of ASH queue (independent of the packet size) should be $log(N) \times N \times M$ bits. For example,

$V_{i,j}(t)$	Set of packets in $\text{VOQ}_{i,j}$ at time t
$V^h_{i,j}(t)$	Set of packets in $\text{VOQ}^h_{i,j}$ at time t
$V^l_{i,j}(t)$	Set of packets in $\text{VOQ}^l_{i,j}$ at time t
$B_{i,j}(t)$	Set of packets in $\text{B}_{i,j}$ at time t
$B^h_{i,j}(t)$	Set of high-priority packets in $\text{B}_{i,j}$ at time t
$H_{i,j}(t)$	$V^h_{i,j}(t) \cup B^h_{i,j}(t)$
$H^j(t)$	$\cup_{i=1}^{N} H_{i,j}(t)$
$\text{ASH}_j(t)$	Set of packets in ASH at time t
$I(P,Q,t)$	Index of packet P in queue Q at time t

TABLE II
TABLE OF NOTATIONS (II).

for a 32×32 switch with $M = 1000$ packets, the size of ASH queue for one output should be 20 KB.

VI. Analysis

Here, we analyse the pCICQ-1 and pCICQ-2 proposed for SB scheduling. Table II gives the notations for time-dependent variables we use further in this work.

A. pCICQ-1 switch

Lemma 6.1: For any two packets, $P^h_1, P^h_2 \in H_{i,j}(t)$, if $A(P^h_1) < A(P^h_2)$, then $D(P^h_1) < D(P^h_2)$

Proof:

$$A(P^h_1) < A(P^h_2) \Rightarrow \forall t \in \mathbb{N} \quad I(P^h_1, \text{VOQ}^h_{i,j}, t) < I(P^h_2, \text{VOQ}^h_{i,j}, t) \quad (1)$$

Therefore, by the definition of input scheduling policy (Def. 4.2) and PIFO queueing policy (Def. 3.1), P^h_2 will be pushed behind P^h_1 at the crosspoint buffer $\text{B}_{i,j}$. That is,

$$\forall t \in \mathbb{N} \quad I(P^h_1, \text{VOQ}^h_{i,j}, t) < I(P^h_2, \text{VOQ}^h_{i,j}, t) \Rightarrow \forall t^{'} \in \mathbb{N} \quad I(P^h_1, \text{B}_{i,j}, t^{'}) < I(P^h_2, \text{B}_{i,j}, t^{'}) \quad (2)$$

From output scheudling policy (Def. 4.1),

$$\forall t \in \mathbb{N} \quad I(P^h_1, \text{B}_{i,j}, t) < I(P^h_2, \text{B}_{i,j}, t) \Rightarrow D(P^h_1) < D(P^h_2) \quad (3)$$

From (1), (2) and (3),

$$A(P^h_1) < A(P^h_2) \Rightarrow D(P^h_1) < D(P^h_2) \quad \forall P^h_1, \forall P^h_2 \in H_{i,j}(t)$$

■

Lemma 6.2: The output scheduler OS_j will select for dequeueing at time t, a queue having a low-priority packet at the head, only when $\cup_{i=1}^{N} B^h_{i,j}(t) = \emptyset$.

Proof: The proof is trivial from the definition of output scheduling policy using PI vector, Def. 4.1.

■

Lemma 6.3: The maximum number of low-priority packets that will depart through output j, when there exist one or high-priority packets destined to output j, is bounded by Δ^h, the waiting time of a high-priority HoL packet in a VOQ.

Proof: Let t be the earliest time when OS_j has finished dequeueing high-priority packets during a busy period, and selects a low-priority queue for dequeueing the next packet. By Lemma 6.2,

$$\bigcup_{i=1}^{N} B^h_{i,j}(t) = \emptyset$$

That is, there is no high-priority packet in the crosspoint queues, $\text{B}_{*,j}$, at time t. So, if there is a high-priority packet in the switch destined to the same output j, it is in the VOQs. Let it be in $\text{VOQ}^h_{i,j}$. This means, $V^h_{i,j}(t) \neq \emptyset$. But, the maximum time a high-priority HoL packet has to wait before being transferred to the crosspoint queue $\text{B}_{i,j}$ is Δ^h by Def. 4.3. Therefore,

$$B^h_{i,j}(t + \Delta^h) \neq \emptyset$$

Once a high-priority packet reaches a crosspoint queue, the output scheuduler (by Def. 4.1) will no more choose a low-priority packet for dequeueing. Therefore, the maximum number of low-priority packets that can depart the switch when there is one or more high-priority packets is Δ^h.

■

B. pCICQ-2 switch

Lemma 6.4: In pCICQ-2 switch, for any two packets, $P^h_1, P^h_2 \in H_{i,j}(t)$, if $A(P^h_1) < A(P^h_2)$, then $D(P^h_1) < D(P^h_2)$

Proof: The proof here is similar to that for Lemma 6.1 with the input and output scheduling policies replaced by that of pCICQ-2 switch.

■

Lemma 6.5: If a packet P_h that arrived at input k destined to output j has $\tau(P^h, A(P^h)) = n$, then,

- the number of high-priority packets that depart through output j before P^h and after time $A(P^h)$ is n;
- P^h departs at time $A(P^h) + (n+1)$.

Proof: Assume P_h arrived at input k destined to output j, such that $A(P^h) = t$.

For the first part, from definitions 5.3 and 5.4, $\tau(P^h, t) = n$ means, excluding P^h there are n high-priority packets in the switch destined for output j and arrived not after t. These packets form a set,

$$\mathcal{H}_j = \{P | (D(P) > t) \wedge \big((A(P) < t) \vee \big((A(P) = t) \wedge (id(P) < id(P^h))\big)\big)\} \quad (4)$$

where $id(P)$ is the input port at which packet P arrived. For any packet $P \in \mathcal{H}_j$, $\tau(P, A(P^h)) < n$. Hence, the input scheduler (Assumption 5.6) guarnatees that these packets reach their corresponding crosspoint queues within their time-lines, which is less than n.

Let n_i be the number of occurances of i among the first n elements in ASH_j at time t. That is, $n = n_1 + n_2 + \cdots + n_N$, and $0 \leq n_i \leq n, \forall i \in \{1, \ldots, N\}$. By definition of OSSC (Def. 5.2), $CP_{i,j}$ will be selected for dequeueing n_i times in the first n time-slots following t. In addiation, there are n_i packets ($\in \mathcal{H}_j$) in the switch at time t, that arrived at input i for output j (from Def. 5.1). The PIFO queueing policy ensures that all the n_k packets in $\mathcal{H}_j$ from input k reach $CP_{i,j}$ before P^h. Therefore P^h will not be dequeued in any of the first n_k

times $CP_{i,j}$ is selected after time t by the output scheduler. Hence, P^h will be dequeued only at the $(n_k+1)^{th}$ selection instance of $CP_{i,j}$ by the output scheduler since time t; and this by the definition of SC operation and OSSC will happen only after n time-slots after t during which all the n packets in $\mathcal{H}_j$ will be dequeued. Therefore, for all $P \in \mathcal{H}_j, D(P) < D(P^h)$.

For the proof of the second part: From the definitions of SC operation (Def. 5.1) and time-line (Def. 5.4), the entry at position $(n+1)$ in ASH_j at time t is k. Going by the same arguments as for the first part, after all the n packets are dequeued at their crosspoints, the output scheduler selects $CP_{i,j}$ for dequeueing. Due to the input scheduling (Assumption 5.6) and PIFO queueing policy (Def. 3.1), and the first part of this proof, P^h will be at the head of $CP_{i,j}$. Hence, P^h will be dequeued at time $t+(n+1)$.

■

Lemma 6.6: For any two packets, $P_1^h, P_2^h \in H^j(t)$, if $A(P_1^h) < A(P_2^h)$, then $D(P_1^h) < D(P_2^h)$

Proof: Without loss of generality, assume P_1^h arrived at input i_1, and P_2^h at input i_2. Since $P_1^h, P_2^h \in H^j(t)$, both are destined to the same output, and also arrived at or before time t. There are two possible scenarios:

- $i_1 = i_2$: By Lemma 6.4, $D(P_1^h) < D(P_2^h)$.
- $i_1 \neq i_2$:

$$A(P_1^h) < A(P_2^h) \Rightarrow \quad \forall t \in \mathbb{N} \quad \tau(P_1^h, t) < \tau(P_2^h, t) \quad (5)$$

Let $t_1 = A(P_1^h)$ and $t_2 = A(P_2^h)$. $t^{'} = t_2 - t_1$. Let $\eta_1 = \tau(P_1^h, t_1)$. Since time-line of a packet decreases one per time-slot (Def. 5.4), $\tau(P_1^h, t_2) = \eta_1 - t^{'}$. Let $\eta_2 = \tau(P_2^h, t_2)$. Clearly, from (5), $\tau(P_1^h, t_2) < \tau(P_2^h, t_2)$; i.e., $(\eta_1 - t^{'}) < \eta_2$ From Lemma 6.5, P_1^h departs at time $t_1 + \eta_1 + 1$ which is equal to $t_2 + (\eta_1 - t^{'}) + 1$; and P_2^h departs at time $t_2 + +\eta_2 + 1$. As $(\eta_1 - t^{'}) < \eta_2$, $D(P_1^h) < D(P_2^h)$.

■

Therefore, once a high-priority packet P^h enters the pCICQ-2 switch for an output j, it is sure that no high-priority packet that arrives after P^h for the same output, will leave the switch before P^h.

Lemma 6.7: The output scheduler OS_j will select for dequeueing at time t, a queue having a low-priority packet at the head, only when $\cup_{i=1}^{N} B_{i,j}^h(t) = \emptyset$.

Proof: The proof is trivial from the definition of output scheduling policy, Def. 5.2.

■

Theorem 6.8: In pCICQ-2 switch,

- high-priority packets depart through an output j in the order of their arrivals,
- a low-priority packet departs the switch through output j, only when there exists no high-priority packet destined to output j in the switch.

Proof: The first part comes from Lemma 6.6.

To prove the second part, assume that a low-priority packet departs the switch through output j at time t, when there is at least one high-priority packet destined to the same output. Then, from Lemma 6.7,

$$\cup_{i=1}^{N} B_{i,j}^h(t) = \emptyset$$

Therefore, there is no high-priority packet at the crosspoints. Besides, according to the output scheduling policy, Def. 5.2,

$$\text{ASH}_j(t) = \emptyset$$

Therefore, there exists no high-priority packet in any VOQ for output j at time t; i.e.,

$$\cup_{i=1}^{N} V_{i,j}^h(t) = \emptyset$$

Hence, there exists no high-priority packet in the switch at time t destined to output j, which is a contradiction. Therefore, a low-priority packet departs a switch only when there exists no high-priority packet in the switch destined to the same output.

■

VII. Conclusions

The analysis for pCICQ-1 switch shows that high-priority packets between an input-output pair follow FIFO order. The low-priority packets do not depart through an output if there is at least one high-priority packet in the crosspoints. Depending on the criteria used for input scheduling policy, one or more low-priority packets can depart through an output port, during the time a high-priority packet takes to reach from the input to the crosspoint. This time is can be reduced using cut-through switching, as it nullifies the time to transfer the high-priority packet from the input to the crosspoint queue.

The pCICQ-2 is shown to maintain the FIFO order of high-priority packets destined to the same output, thanks to the sequence controller. For the stricter constraint of not allowing any low-priority packet to pass through, when there exists a high-priority packet for the same output, we see that an input scheduler dequeueing packets before their time-lines is needed. In future, we will look into specific scheduling policies that achieve this task.

With PIFO queues at crosspoints, it is not difficult to envision implementation of multi-priority strategies in a CICQ switch, though it would require PIFO queue size in propotion to the number of priority levels. This is another intesting aspect of the study that we plan to pursue.

References

[1] Abdesselem Kortebi, Luca Muscariello, Sara Oueslati, and James Roberts, "Minimizing the overhead in implementing flow-aware networking," in *ANCS '05: Proceedings of the 2005 ACM symposium on Architecture for networking and communications systems*. 2005, pp. 153–162, ACM.

[2] "The OpenFlow Switch Consortium," www.openflowswitch.org.

[3] Adam Greenhalgh, Felipe Huici, Mickael Hoerdt, Panagiotis Papadimitriou, Mark Handley, and Laurent Mathy, "Flow processing and the rise of commodity network hardware," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, pp. 20–26, 2009.

[4] T. Bonald, S. Oueslati-Boulahia, and J. Roberts, "IP traffic and QoS control: the need for a flow-aware architecture," in *World Telecommunications Congress*, Sep. 2002.

[5] S. Ben Fred, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts, "Statistical bandwidth sharing: a study of congestion at flow level," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 111–122, 2001.

[6] Leonard Kleinrock, *Queueing Systems, Volume II: Computer Applications*, Wiley Interscience, 1976.

[7] Idris A. Rai, Guillaume Urvoy-Keller, Mary K. Vernon, and Ernst W. Biersack, "Performance analysis of las-based scheduling disciplines in a packet switched network," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 106–117, 2004.

[8] Konstantin Avrachenkov, Urtzi Ayesta, Patrick Brown, and Eeva Nyberg, "Differentiation Between Short and Long TCP Flows: Predictability of the Response Time," in *IEEE INFOCOM*, 2004, vol. 2, pp. 762–773.

[9] Masayoshi Nabeshima, "Performance Evaluation of a Combined Input- and Crosspoint-Queued Switch," *IEICE Transactions on Communications*, vol. vE83-B, no. 3, pp. 737–741, Mar 2000.

[10] F. Abel, C. Minkenberg, R.P. Luijten, M. Gusat, and I. Iliadis, "A four-terabit packet switch supporting long round-trip times," *Micro, IEEE*, vol. 23, no. 1, pp. 10–24, Jan/Feb 2003.

[11] N. Chrysos and M. Katevenis, "Multiple priorities in a two-lane buffered crossbar," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM '04*, Nov.-3 Dec. 2004, vol. 2, pp. 1180–1186.

[12] S.-T. Chuang, S. Iyer, and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," in *Proceedings IEEE INFOCOM 2005, 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, March 2005, vol. 2, pp. 981–991.

[13] Alex Kesselman, Kirill Kogan, and Michael Segal, "Packet mode and QoS algorithms for buffered crossbar switches with FIFO queuing," in *PODC '08: Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing*, 2008, pp. 335–344.

[14] Alex Kesselman, Kirill Kogan, and Michael Segal, "Best Effort and Priority Queuing Policies for Buffered Crossbar Switches," in *SIROCCO '08: Proceedings of the 15th international colloquium on Structural Information and Communication Complexity*, 2008, pp. 170–184.

[15] Manolis Katevenis, Georgios Passas, Dimitrios Simos, Ioannis Papaefstathiou, and Nikos Chrysos, "Variable packet size buffered crossbar (CICQ) switches," in *IEEE International Conference on Communications (ICC 2004)*, Jun 2004, pp. 1090–1096.

[16] S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE J. Selected Area in Commun.*, vol. 17, no. 6, pp. 1030–1039, Jun 1999.

[17] S. Young, I. Arel, and O. Arazi, "Pi-PIFO: A scalable pipelined PIFO memory management architecture," in *10th International Conference on Telecommunications, ConTEL 2009*, June 2009, pp. 265–270.

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399