



ELSEVIER

Computer Networks 37 (2001) 541–559

**COMPUTER
NETWORKS**

www.elsevier.com/locate/comnet

Input-queued router architectures exploiting cell-based switching fabrics

M. Ajmone Marsan^{*}, A. Bianco, P. Giaccone, E. Leonardi, F. Neri

Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy

Abstract

Input queued and combined input/output-queued architectures have recently come to play a major role in the design of high-performance switches and routers for packet networks. These architectures must be controlled by a packet scheduling algorithm, which solves contentions in the transfer of data units to switch outputs. Several scheduling algorithms were proposed in the literature for switches operating on fixed-size data units. In this paper we consider the case of packet switches, i.e., devices operating on variable-size data units at their interfaces, but internally operating on fixed-size data units, and we propose novel extensions of known scheduling algorithms for input queued and combined input/output-queued architectures. We show by simulation that, in the case of packet switches, input queued and combined input/output-queued architectures can provide performance advantages over output-queued architectures. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Input-queued switches; Router architectures; Scheduling algorithms

1. Background: input vs. output-queued switches

A key component of a packet network is the switching node: switches in Ethernet LANs, ATM switches in the B-ISDN, IP routers in the Internet. Given the diversity of user applications and the ever-growing bandwidth demands, nodes must keep increasing their performance and improving

their ability to contribute to the provision of quality-of-service (QoS) guarantees.

In the recent past, significant research efforts were devoted by both the academic and the industrial communities to the design of efficient ATM switches for the B-ISDN [2]. Those research activities initially concentrated on issues related to the design of the switching fabric (i.e., of the part of the switch that is in charge of the transfer of data units from input to output line interfaces) normally assuming an output-queued architecture. More recently, much attention was paid to the design of input/output interfaces (or line cards), with the development of efficient table look-up schemes to implement label swapping and routing algorithms [3–5], and of fair queueing

^{*} Corresponding author. Tel.: +39-11-564-4032; fax: +39-11-564-4099.

E-mail addresses: ajmone@polito.it (M. Ajmone Marsan), bianco@polito.it (A. Bianco), giaccone@polito.it (P. Giaccone), leonardi@polito.it (E. Leonardi), neri@polito.it (F. Neri).

schemes [6–9] to guarantee some degree of QoS to end users.

The effort in the design of ATM switches brought efficient chip-sets to the market, currently used as building blocks for high-performance Internet routers; as a consequence, for many advanced Internet routers the switching fabric internally operates on fixed-size data units (called cells), and input IP datagrams are internally segmented into ATM-like cells that are transferred to output interfaces, where they are reassembled into variable-size IP datagrams.

Although output-queued switch architectures were for some time the generally accepted approach, in the last few years, input-queued architectures [10] have come back into the arena as packet switching engines, and they are currently considered by many designers as the best solution when the line speed is pushed to technological limits. Most architectures that are being studied today for the implementation of Gigabit routers in the Internet are not purely output queued. The main disadvantage of output queueing (OQ) is that both the internal interconnect (i.e., the switching fabric) and output queues in line cards must operate at a speed equal to the sum of the rates of all input lines. In applications where the number of ports is large or the line rate is high, this makes OQ impractical. With respect to input queueing (IQ) schemes, OQ retains the advantage that delays through the switch can be more easily controlled, and the implementation of (concentrated) fair queueing algorithms at the output is relatively easy and well understood [11].

IQ schemes permit all the components of the switch (input interfaces, switching fabric, output interfaces) to operate at a speed which is compatible with the data rate of input and output lines. Examples of cell-based routers and switches adopting switching fabrics with input buffers can be found both in commercial products and laboratory prototypes: the Lucent GRF [12], the Cisco GSR [13], the Tiny-Tera [14], the AN2/DEC [10,15], the iPoint [16], the MGR/BBN [17], and the IBM Prizma [18].

A major issue in the design of IQ switches is that the access to the switching fabric must be controlled by some form of scheduling algo-

rithm.¹ Several scheduling algorithms for IQ cell switches were proposed and compared in the literature [16,20–25]. They provide performance very close to the more hardware-demanding OQ architecture. We consider some of these proposals, and modify them in order to deal with variable-size packets, in the sense that we constrain the scheduling algorithm to deliver contiguously all the cells deriving from the segmentation of the same packet. In other words, variable-size packets are transformed into “trains of cells”, and the transfer of the cells belonging to the same train is scheduled in such a way that they remain contiguous in the delivery to the output card, i.e., they are not interleaved with the cells of another train. We shall see that this constraint permits significant savings in memory and complexity, since the re-assembly of packets at the output becomes much easier. The performance of the considered scheduling algorithms, both in the case of cell scheduling and in the case of packet scheduling, is compared by simulation.

Scheduling algorithms for IQ architectures are always relatively demanding in terms of computing power and control bandwidth. It has been shown [26] that this complexity can be partly reduced when the switching fabric and the output memory bandwidth have a moderate speed-up with respect to the data rate of input/output lines. A speed-up of two, independent of the number of switch ports, can be shown [27–31] to be sufficient to provide the same performance as an OQ architecture. As soon as the switch takes advantage of an internal speed-up, buffering is required also at output ports; we thus use the term “combined input/output queueing” (CIOQ) in the remainder of the paper. Obviously, when the speed-up is such

¹ The term “scheduling algorithm” for switching architectures is used in the literature for two different types of schedulers: switching matrix schedulers and flow-level schedulers [19]. Switching matrix schedulers decide which input port is enabled to transmit in a switch without pure output queueing; they avoid blocking and solve contentions within the switching fabric. Flow-level schedulers decide which cell flows must be served in accordance to QoS requirements. In this paper the term scheduling algorithm is only used to refer to the first class of algorithms.

that the internal bandwidth equals the sum of the data rates on input lines, input buffers become useless, and we obtain a pure OQ architecture.

In this paper we study purely input-queued and combined input/output-queued switching architectures, taking output-queued switches as a reference for comparisons. We concentrate on switch architectures that internally operate in a “synchronous” fashion, i.e., in which switching decisions are taken at equally spaced time instants, but in general accept at their inputs variable-size data units. Moreover, we limit our attention to memoryless switching fabrics, and to a control of conflicts in accessing the switching fabric entirely performed by the scheduling algorithm, although some recent proposals consider the possibility of resolving conflicts inside the switching fabric, making use of distributed buffering and back-pressure techniques (see, e.g., Ref. [32]). We devote some attention to switching fabrics with a speed-up equal to two, where input buffers are organized into a single FIFO queue, and the scheduling algorithm is particularly simple. A comparison of this setup with the previously considered IQ architectures is presented.

The paper is organized as follows. Section 2 describes the logical switch architecture and its major components, both in the case of fixed-size and of variable-size packets. Section 3 briefly

overviews the considered cell scheduling algorithms, and presents our modifications to turn them into packet scheduling algorithms. Section 4 presents simulation results for cell and packet scheduling, in the case of both pure input-queued, and combined input/output-queued architectures with speed-up equal to two. Finally, Section 5 concludes the paper.

2. Logical architecture

We describe now the logical structure of cell and packet switches, based on IQ, OQ and CIOQ switching architectures. Cell switches, i.e., switches operating on fixed-size data units, are introduced first; packet switches will then be built using cell switches as an internal building block.

2.1. Cell switches

2.1.1. Input-queued cell switches

Fig. 1 shows the logical structure for an input-queued cell switch. The switch operates on fixed-size data units, which can be ATM cells, or have any other convenient format. Borrowing from the ATM jargon, we shall use the term cells to identify the internal fixed-size data units. Actually, our

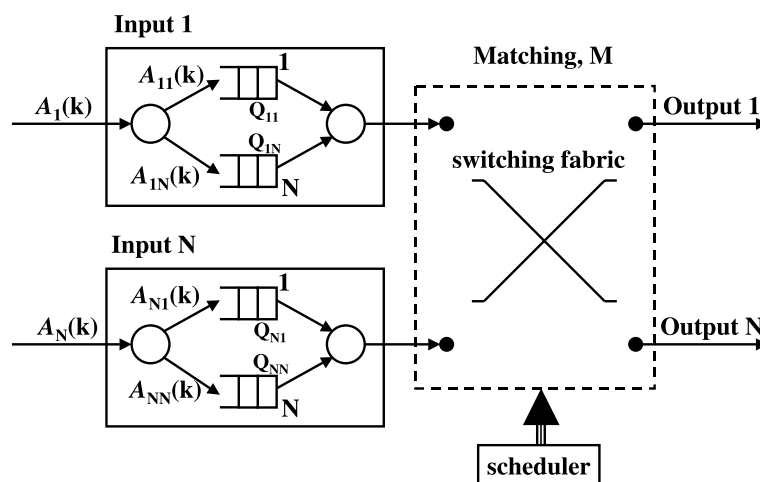


Fig. 1. Logical structure of an input-queued cell switch.

results do not strictly require that fixed-size cells are used, but more generally refer to any switch that takes switching decision at equally spaced time instants. The distance between two switching decisions is called slot, and the slot is the granularity in allocating switch resources.

We do not deal with the problem of partial slot filling due to the variable size of IP packets arriving at inputs, even if its impact on performance may be significant, depending on the slot size and the input packet length distribution.

We consider a switch with N inputs and N outputs (indeed, one input and one output interface usually reside on the same “line card”). We also assume for simplicity that all input and output lines run at the same speed.

Packets are stored at input interfaces. Each input manages one queue for each output, hence a total of $N \times N = N^2$ queues are present. This queue separation avoids performance degradations due to head-of-the-line blocking [33], and is called virtual output queuing (VOQ) or destination queueing [10,34–36].

Cells arrive at input i , $1 \leq i \leq N$, according to a discrete-time random process $A_i(k)$. At most one cell per slot arrives at each input, i.e., the data rate on input lines is no more than 1 cell per slot. Enough addressing information is contained in each arriving cell in order to be able to know which is the output j , $1 \leq j \leq N$, to which it must be transferred. The association between inputs and outputs is performed by routing functions, and typically requires the inspection of a routing table. When a cell with destination j arrives at input i , it is stored in the FIFO queue Q_{ij} . The number of cells in Q_{ij} at time k is denoted by $L_{ij}(k)$. These FIFO queues have limited capacity: each queue can store at most L cells.

We call $A_{ij}(k)$ the arrival process at input i for output j ; the average arrival rate is denoted by λ_{ij} . The aggregation of all arrival processes is $A(k) = \{A_i(k), 1 \leq i \leq N\}$. $A(k)$ is termed admissible if no input and no output is overloaded, i.e., if $\sum_{i=1}^N \lambda_{ij} < 1$, $\forall j$ and $\sum_{j=1}^N \lambda_{ij} < 1$, $\forall i$. Otherwise $A(k)$ is inadmissible. For later use, we define the cell arrival rate matrix $A = [\lambda_{ij}]$, and the normalized cell arrival rate matrix $\Gamma = [\gamma_{ij}]$, with $\gamma_{ij} = \lambda_{ij} / (\sum_{i=1}^N \sum_{j=1}^N \lambda_{ij})$.

The switching fabric is non-blocking and memoryless, and introduces no delay: at most one cell can be removed from each input and at most one cell can be transferred to each output in every slot. Since the speed at which cells are fed into output interfaces is equal to the speed at which cells are fed to input interfaces, we have a speed-up factor (see Section 2.2) equal to 1. The scheduling algorithm decides which cells can be transferred from the inputs to the outputs of the switch in every slot. Scheduling algorithms will be discussed in detail in Section 3.

2.1.2. Output-queued cell switches

The output-queued cell switch needs no input buffers (neglecting buffers used at the input to store the newly arrived cell) because the switching fabric has enough capacity to transfer to the desired output all the cells received in one time slot. In the worst case (i.e., when a cell arrives at each input, and all cells are directed to the same output), this means that the bandwidth to each output must be equal to the sum of the bandwidths available on all input lines. We thus say that the switching fabric must have a speed-up factor equal to N . This internal speed-up can be obtained in different manners. We assume here that it is achieved in the time domain, by setting the switching fabric clock N times faster than on input/output line interfaces. We therefore assume that the slot time is subdivided into N mini-slots, and that each input can use a different mini-slot to transfer a cell to a given output. The allocation of mini-slots to inputs can be fixed, time-varying, or random. We implemented a fixed round-robin of inputs in our simulation programs.

At each output, cells are stored in a single FIFO queue. For a fair comparison with input-queued switches, we assume that the total amount of buffer space is kept constant, i.e., that the FIFO output queue can store $N \times L$ cells. This assumption gives some advantage to the OQ schemes, which can exploit some degree of buffer sharing.

2.1.3. Combined input/output-queued cell switches

CIOQ architectures are a compromise between IQ and OQ, for which some degree of speed-up is

available, but not as much as for OQ. These architectures require buffering at both input and output line interfaces.

The buffer space at each input can be organized in either one FIFO queue, or several queues, as in the case of VOQ. We call S_{in} the number of cells per slot that can be read from the queue(s) at each input, and S_{out} the number of cells per slot that can be written into the output queue at each output.

Several CIOQ architectures were proposed in the literature. Unfortunately, different authors often assume different definitions of speed-up. The following cases are possible:

(1) $S_{in} = 1$ and $S_{out} = S$ [37–41]. The switch can transfer up to S cells to the same output, but no more than a cell can be read from each input in one slot. This definition $S^{(1)}$ of speed-up derives from input bus switching architectures, in which a shared bus can be read by all output ports, but can be accessed by each input for the transmission of at most one cell per slot. A well-known example of this architecture is the Knockout switch [42].

Analytical models [2] show that the maximum throughput for $S^{(1)} = 2$ equals 0.885 in uniform traffic conditions when $N \rightarrow \infty$. To achieve a throughput larger than 99%, $S^{(1)}$ must be larger than 4. Only with $S^{(1)} = N$ it is possible to reach the maximum throughput in general traffic conditions.

(2) $S_{in} = S$ and $S_{out} = S$ [26,41]. When this speed-up $S^{(2)}$ equals S , up to S cells can be read from each input and written to each output of the switch in one time slot.

(3) $S_{in} = S$ and $S_{out} = 1$. This definition $S^{(3)}$ of speed-up was investigated in Ref. [44]. It is dual with respect to $S^{(1)}$, in the sense that at most one cell per slot can reach an output card. When coupled with suitable scheduling algorithms for variable-size packet switching, this form of speed-up has the advantage of preventing the interleaving of cells belonging to different packets.

Of course, when $S^{(1)} = 1$, or $S^{(2)} = 1$, or $S^{(3)} = 1$, we have an IQ architecture. We define the speed-up according to $S^{(2)}$ in the remainder of the paper. A CIOQ architecture with VOQ and speed-up $S^{(2)} = S$ is shown in Fig. 2.

It is well-known that the maximum normalized throughput achievable in an IQ switch using one

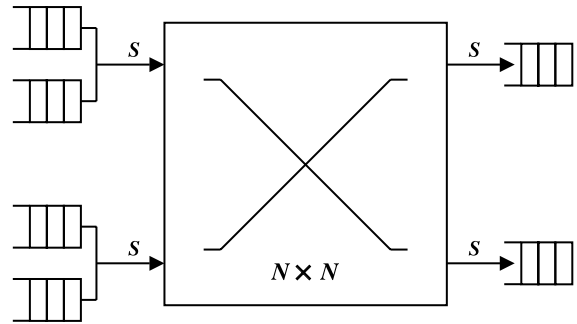


Fig. 2. CIOQ architecture with VOQ and speed-up $S^{(2)} = S$.

FIFO queue per input port, with uniform traffic and an infinite number of ports, is limited to 0.586 by the head-of-the-line blocking phenomenon [33]. One can argue that, by executing the scheduling algorithm twice in each slot, thereby using a speed-up $S^{(2)} = 2$, the maximum throughput becomes 100% [41], as it is for OQ architectures. This simple statement holds only for uniform traffic, and does not provide guarantees related to delays.

It has been shown that a CIOQ architecture with VOQ can mimic the OQ behavior if a speed-up $S^{(2)} = N/2$ is available, and if a (non-trivial) scheduling algorithm called Home Territory Algorithm is implemented [26]. With the same architecture, an exact emulation of OQ under general traffic patterns has been obtained using a complex scheduling algorithm in Ref. [27].

In this paper we shall compare IQ, OQ, and CIOQ architectures. With the aim of being fair in the comparison, we accept high scheduling complexity for IQ, high hardware complexity (i.e., speed-up N) for OQ, and intermediate scheduling and hardware complexity for CIOQ. We therefore limit our attention to CIOQ switches with limited speed-up increase $S^{(2)} = 2$, and very simple FIFO scheduling.

Although a more detailed discussion of scheduling algorithms for IQ architectures will be provided in Section 3, we anticipate here the description of the simple algorithm that we consider in the CIOQ case. We assume that each input and each output of the CIOQ switch maintains a single FIFO queue, and that the following scheduling algorithm is executed twice in each slot.

At every execution of the algorithm, inputs are cyclically scanned, starting from a different input every time, selected by a round-robin scan. In this scan each input attempts to transfer the cell at the head of its input FIFO queue. If the corresponding output was not already engaged by a preceding input in the round-robin scan, the transfer is enabled, otherwise it is deferred to the next execution of the algorithm. Since the algorithm is executed twice in each slot time, at most the first two cells are removed from input queues, and at most two cells are delivered to each output queue in each slot. We call this scheduling algorithm FIFO-2.

2.2. Packet switches (or routers)

Given the cell-switch logical architectures described in previous sections, we can now build a packet switch around them. We use as a reference model the case of a high-performance IP router built around an ATM cell-switch, but the same analysis can be referred also to router built around a generic (possibly proprietary) cell-based switch.

Each router port has line interfaces where any data-link and physical layer protocol can be used to receive and transmit IP datagrams. The IP protocol sits on top of the data-link protocol at the input and at the output of the router. Within the IP layer at the input, routing functions are activated to associate an output port with the destination IP address (we neglect here issues related to

the implementation of these functions, such as table look-up). Input IP datagrams are segmented into ATM cells, that will be transferred to output ports by a high-performance ATM switching fabric. Once cells are delivered to an output port, they are reassembled into the IP datagram, which is transmitted on the output line according to possibly different line formats.

Fig. 3 emphasizes the possible speed variations inside the architecture of an IP router incorporating an ATM switching fabric. We take as a reference the bit rates on the input and output lines, which we assume to be the same and equal to 1.

- **SPEEDUP-IP-IN** is the speed at which cells are transferred from the input IP module to the input of the ATM switch. $\text{SPEEDUP-IP-IN} = 1$ means that, if an input IP datagram is segmented in k cells, all these cells are sequentially transferred to the cell-switch input in k time slots. Note that this takes into account segmentation overheads and partial filling of the last cell.
- **SPEEDUP** is the number of cells per slot that can be read from the inputs of the cell-switch. Since we assume the definition $S^{(2)}$ of speed-up for the cell-switch, this is also the number of cells per slot that can be written onto a switch output.
- **SPEEDUP-IP-OUT** is the speed at which cells are transferred from the output of the cell-switch to the output IP module.

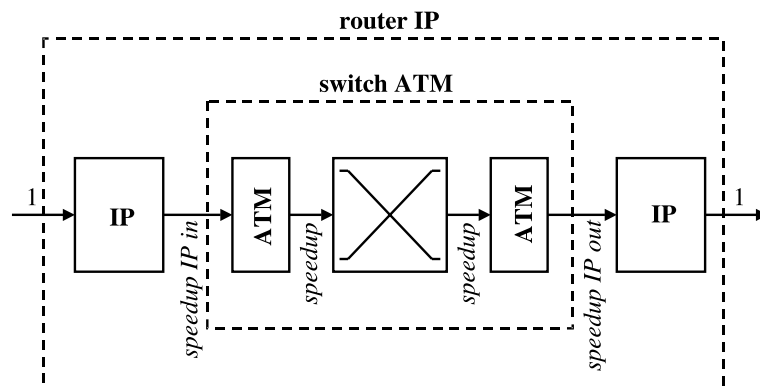


Fig. 3. Speed-up definitions for the packet switch built around a cell switch.

Input and output IP modules in Fig. 3 consist of queues for variable-size packets, and operate in store-and-forward mode; they comprise segmentation and reassembly functions. ATM modules comprise cell queues at the input and/or at the output of the switch.

In the next subsections, we detail the router architecture in the case of IQ, OQ and CIOQ cell switches.

2.2.1. IQ packet switches

The logical architecture for an IQ packet switch is shown in Fig. 4. At each input an input segmentation module (ISM) segments the incoming packet into cells. PLS is the external packet line speed. Since the ISM operates in store-and-forward mode, it must be equipped with enough memory to store a maximum-size packet, and the segmentation process starts only after the complete reception of the packet.

The cells resulting from the segmentation are transferred to the cell-switch input at a speed (called ILS) equal to the line speed PLS incremented to account for segmentation overheads. The capacity of input queues at the cell switch is limited to L , hence losses can occur. We assume that the entire packet is discarded if the input queue of the cell switch does not have enough free space to store all the cells deriving from the segmentation of the packet when the first of these cells hits the queue. This is of course a pessimistic assumption, but has the advantage of ease of implementation, and of avoiding the transmission of incomplete packet fractions through the switch.

The cell-based switching fabric transfers cells from input to output queues, according to a scheduling algorithm, as previously discussed. These cells are delivered to the output reassembly module (ORM) at speed ILS. Here packets (i.e., IP datagrams) are reassembled. In general, cells belonging to different packets can be interleaved at the same output, hence more than one reassembly machine can be active in the same ORM. However, at most one cell reaches each ORM in a slot time, hence at most one packet is completed at each ORM in a slot.

Once a packet is complete, it is logically added to an output packet queue, called packet FIFO in the figure, from which packets are sequentially transmitted onto the output line. Note that the packet FIFO functionality is typically implemented by imposing a sequential transfer from the suitable ORM to the output line of all the cells belonging to the same reassembled packet. No internal speedup is required to support ISM and ORM, but for compensating internal overheads.

2.2.2. Optimization of IQ packet switches

In the case of IQ packet switches, it is possible to further simplify the structure of the switch, and to improve its performance, by enforcing additional constraints on the scheduling algorithm. Indeed, the cells belonging to the same packet are contiguous in the input queue of the internal cell switch. As we shall see in Section 3, it is possible to modify some well-known scheduling algorithms in such a way that, once the transfer through the switching fabric of the first cell of a packet has

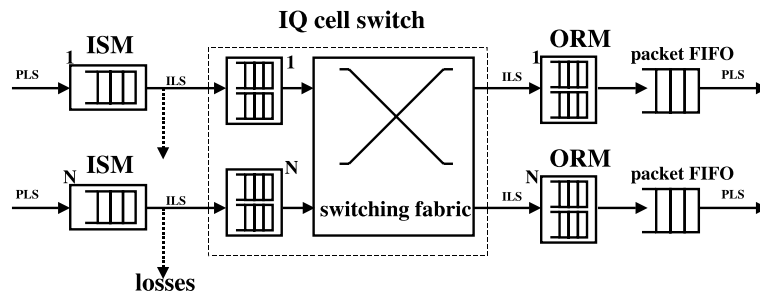


Fig. 4. Logical architecture for an IQ packet switch.

started towards the corresponding output port, no cells belonging to other packets can be transferred to that output. In such a way, cells belonging to the same packet are also kept contiguous in the output queue, and the ORM modules are not necessary. We call this class of scheduling algorithms packet-mode scheduling.

Packet-mode scheduling is not possible in OQ switches, since the interleaving of cells in output queues cannot be avoided.

If packet-mode scheduling is adopted in IQ packet switches, the logical architecture can be simplified as shown in Fig. 5: both the ORM module and the output packet FIFO are removed with respect to Fig. 4. The removal of the packet FIFO can be achieved only if no format conversion is required in order to transmit the packet on

the output link. Since each module operates in store-and-forward mode, hence introduces a delay equal to the packet size, the delays through the switch are reduced by twice the packet duration. Note also that the segmentation of the packet into cells is no longer strictly required: the only necessary information is the (integer) number of slot times used for the transfer of the packet.

These observations show that important additional advantages with respect to cell switches are exhibited by IQ architectures with respect to OQ architectures (see next paragraph) for the implementation of packet switches.

2.2.3. OQ packet switches

Fig. 6 shows the logical architecture of an OQ packet switch. Packets arrive at input ports where

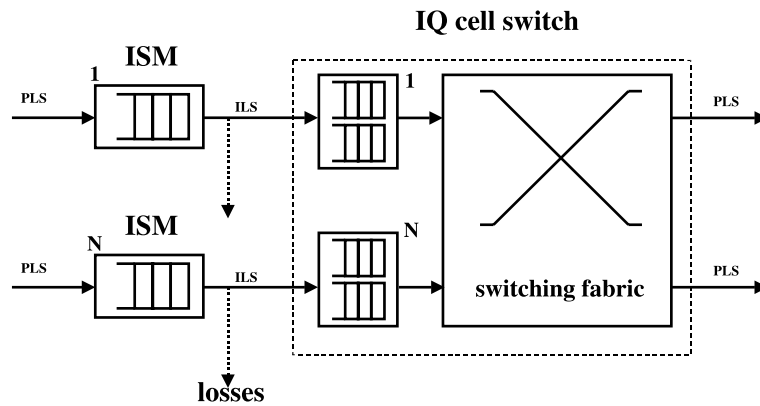


Fig. 5. Simplified architecture for an IQ packet switch with contiguous cells.

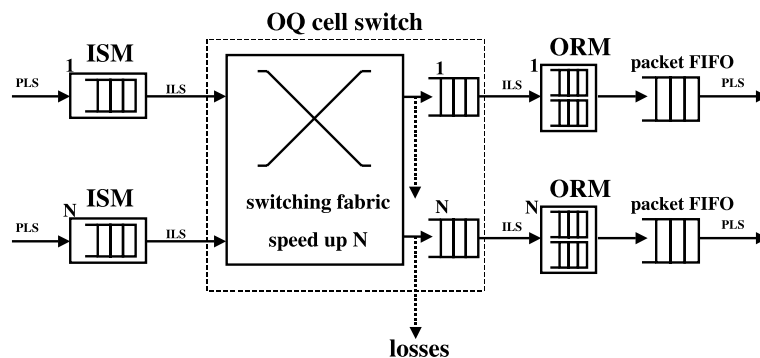


Fig. 6. Logical architecture for an OQ packet switch.

they are segmented by ISM modules, similarly to what happens in IQ routers. The cells obtained with the segmentation are sent to the cell switch at speed ILS, and are immediately transferred to the output queues of the cell switch, thanks to a speed-up equal to N in the switching fabric. Losses may occur at output queues, whose capacity is limited to $N \times L$ for each queue, since we are assuming the same buffering capacity for OQ and IQ switches.

From the output queues of the cell switch, cells are delivered at speed ILS to an ORM module for reassembly. Once a packet is completed, it is queued in a packet FIFO queue similar to what was seen for the IQ case.

An important difference between OQ packet switches and IQ packet switches resides in the way of handling losses. Cells belonging to different packets can be interleaved in output queues. When a cell at the output of the switching fabric does not find room in the output queue of the cell switch, it must be discarded. The other cells belonging to the same packet of the discarded cell may be in the output queue, or already in the ORM. We assume it is impossible to identify and discard the other cells in the output queue: those cells will be discarded by the ORM module, which has knowledge of the existence of the packet. This means that cells belonging to packets that suffered partial losses unnecessarily use system resources.

The IQ architecture has the advantage that all the cells belonging to the same packet are contiguous in input queues, where losses occur. In the OQ case, losses occur in queues where cells of different packets are interleaved.

2.2.4. CIOQ packet switches

CIOQ packet switches are built around a CIOQ cell switch. For the sake of conciseness, we do not show the logical architecture in this case, but it can be easily obtained by combining Figs. 4 and 6. The only difference is that we do not consider VOQ in the CIOQ case, hence only one FIFO queue is available at each input (and at each output) of the cell switch. Losses may occur at both input and output interfaces.

3. Cell and packet scheduling algorithms in IQ switches

3.1. Problem definition

This section describes scheduling algorithms. Remember that we used this term to refer to switching matrix schedulers, i.e., to the set of rules used to decide which input port is granted access to the switching fabric in a non-purely output-queued switch. Scheduling algorithms avoid blocking and solve contention within the switching fabric.

The scheduling algorithm selects a matching M , i.e., a set of input–output pairs with no conflicts, such that each input is connected with at most one output, and each output is connected with at most one input. In each slot, if input i is connected with output j , a cell is removed from Q_{ij} , and transferred to output j by properly configuring the non-blocking switching fabric.

In the technical literature, scheduling algorithms in IQ cell switch architectures are described as solutions of the matching problem in bipartite graphs. The switch state can be described as a bipartite graph $G = [V, E]$ (see Fig. 7) in which the graph vertices in set V are partitioned in two subsets: subset V_I , whose elements v_{Ik} correspond to input interfaces, and subset V_O , whose elements v_{Ok} correspond to output interfaces. Edges indicate the needs for cell transfers from an input to an output (an edge from v_{In} to v_{Om} indicates the need for cell transfers from input n to output m), and can be labeled with a metric that will be denoted

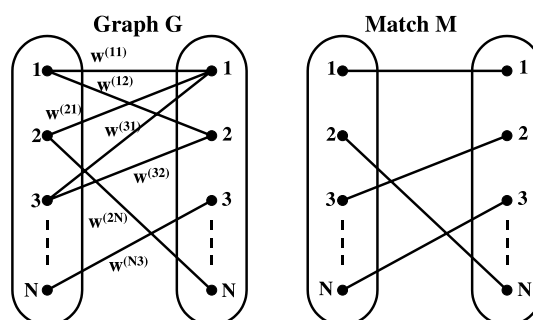


Fig. 7. Bipartite graph description of the scheduling problem.

by w_{nm} . The adopted metric is a key part of the scheduling algorithm, as we shall see; it can be constant (or equal to 1, and usually not specified on the graph) to simply indicate that at least one cell to be transferred exist, or it can refer to the number of cells to be transferred, to the time waited by the oldest cell, or to other state indicators.

A matching M is a selection of an admissible subset of edges. A subset of edges is admissible if no vertex has two connected edges; this means that it never happens that two cells are extracted from the same input, or that two cells are transferred to the same output. A matching has maximum size if the number of edges is maximized; a matching has maximum weight if the sum of the edge metric is maximized.

The different IQ cell switch architectures that we shall discuss differ in their scheduling algorithms, hence in their matching algorithms. The need for good matching algorithms derives from the fact that the optimal solutions of the problem have very high complexity, so that a compromise between complexity and performance is sought for. The complexity is $O(N^3)$ for the maximum weight matching ([45, Chapter 8]) algorithm, that can be proved to yield the maximum achievable throughput using as metric either the number of cells to be transferred, or the time waited by the oldest cell; it is $O(N^{5/2})$ for the simpler and less efficient maximum size matching algorithm [46]. Well known maximum size matching algorithms were proposed by Dinic [45] and Hopcroft [47].

The $N \times N$ matrix whose elements are the edge metric in graph $G = [V, E]$ is called the weight matrix, denoted by $\mathbf{W} = [w_{ij}]$. This weight matrix \mathbf{W} varies with time, according to the changes in the system parameters from which its elements are computed. When necessary, denoting by k the current slot, we shall write $\mathbf{W}(k) = [w_{ij}(k)]$. We assume $w_{jk} = 0$ for missing edges in G , i.e., when no cells from input j to output k are waiting in input queues.

3.2. Classification of cell scheduling algorithms

A number of scheduling algorithms for IQ switch architectures have appeared in the technical litera-

ture. In this paper we consider four such proposals, namely iSLIP [20], iOCF [36,43], MUCS [16] (in its weighted version), and RPA [21]. iSLIP has been chosen for its simplicity, iOCF due to its metric based on cell age, RPA for both simplicity and good performance under unbalanced traffic patterns, and MUCS for the very good performance obtained in different traffic scenarios.

The reader is referred to the original works for a detailed description of these algorithms. In this section we recall the general taxonomy for scheduling algorithms proposed in Ref. [24], and classify the considered proposals accordingly.

The output of the scheduling algorithm at slot k is a cell transfer matrix $\mathbf{T}(k) = [t_{ij}(k)]$, whose elements provide the result of the matching computation:

$$t_{ij} = \begin{cases} 1 & \text{if a cell is transferred from } i \text{ to } j, \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Any IQ scheduling algorithm can be viewed as operating according to three phases:

(1) *Metrics computation.* Computation of the weight matrix $\mathbf{W}(k) = [w_{ij}(k)]$. Each one of the possible N^2 edges in the bipartite graph is associated with a metric depending on the state of the corresponding queue (the metric associated with the edge from v_{i1} to v_{Oj} , w_{ij} , depends on the content of Q_{ij} at slot k). This metric will act as a priority for the cell transfer.

(2) *Heuristic matching.* Computation of the cell transfer matrix $\mathbf{T}(k) = [t_{ij}(k)]$. This phase must try to maximize the following sum:

$$\sum_{i=1}^N \sum_{j=1}^N t_{ij}(k) w_{ij}(k) \quad (2)$$

with the constraints

$$\sum_{i=1}^N t_{ij}(k) \leq 1 \quad \sum_{j=1}^N t_{ij}(k) \leq 1. \quad (3)$$

Since the cost for the computation of the optimum matching (maximum size or maximum weight) is too high, all scheduling algorithms resort to heuristics with variable effectiveness and complexity. When the matching is not optimum, but no cells can be added without violating the constraints (3),

the terms maximal size matching and maximal weight matching are used in the literature.

(3) *Contention resolution*. In the execution of the heuristic algorithm for the determination of a maximal match, a strategy is necessary to solve contentions due to edges with equal metric, source or destination. The contention resolution typically is either random (RO, random order), or based on a deterministic scheme; round-robin (RR) and sequential search (SS) are frequent choices in the latter case, the difference lying in the starting point chosen in the selection process, which is state-dependent for RR, and state-independent for SS.

The first phase is preliminary to the other two, which are instead interlaced.

As we shall see in Section 4.2, the phase that has the most profound impact on performance is the metric computation, whereas the different heuristics to obtain good matches have a deep impact on the algorithm complexity.

As regards metric, we consider the following alternatives:

QO (*Queue occupancy*). In this case $w_{ij} = u(L_{ij})$ where $u(\cdot)$ is the unit step function. This is the metric adopted by iSLIP. The adoption of this metric leads to the search for a maximal size match. All other metrics lead to the search for a maximal weight match, which can differ in the weights.

QL (*Queue length*). The metric in this case is the number of cells in the queue: $w_{ij} = L_{ij}$. This is the metric adopted by RPA.

CA (*Cell age*). The metric in this case is the time already spent in the queue by the cell at the queue head. This is the metric adopted by iOCF.

ML (*MUCS length*). MUCS uses a metric that is derived from queue lengths as

$$w_{ij} = \frac{L_{ij}}{\sum_{k=1}^N L_{ik}} + \frac{L_{ij}}{\sum_{k=1}^N L_{kj}}. \quad (4)$$

As regards the heuristic matching, the choices adopted in the considered IQ scheduling algorithms are the following:

IS (*Iterative search*). In this case, during a first step, each input interface sends all its transmission requests with the associated metric to the relevant output interfaces ($w_{ij}(k)$ is sent from input interface i to output interface j). These select one

among the arriving requests by choosing the largest metric value, and resolving ties according to a contention resolution scheme (output contention). The accepted requests are then sent back to the input interfaces. If an input interface receives more than one acceptance, it selects one by choosing that with the largest metric value, and resolving ties according to a contention resolution scheme (input contention). The successive steps are equal to the first one, but they concern only the transmission requests from input interfaces that received no acceptance, as well as those that were satisfied in previous steps (the repetition of these requests is necessary to progressively freeze the match configuration). This heuristic was proposed in Ref. [36], and it is adopted by iSLIP and iOCF.

MG (*Matrix greedy*). Consider the $N \times N$ matrix $\mathbf{W} = [w_{ij}]$. In this case the algorithm consists of (up to) N steps, in each of which the largest element(s) w_{ij} of \mathbf{W} is (are) selected, and the corresponding cell transmissions are enabled (provided that no conflict arises if ties for the largest metric exist; otherwise a conflict resolution is necessary) before reducing the matrix by eliminating the entries corresponding to enabled cell transfers. This is the heuristic adopted by MUCS.

RV (*Reservation vector*). In this case the algorithm is based on a sequential access to a reservation vector with N records, where input interfaces sequentially declare their cell transfer needs and the associated metric, possibly overwriting requests with lower metric values. A second sequential pass over the vector allows the confirmation of requests, or the reservation of other transfers for the inputs whose original reservations were overwritten. This is the heuristic adopted by RPA.

Table 1 gives a synoptic view of the considered IQ scheduling algorithms, where for each algorithm we give the type of metric used, the heuristic algorithm adopted for the identification of matches, and the approach used for contention resolution.

3.3. Packet-mode scheduling algorithms

We claimed in the last part of Section 2.2 that the architectural complexity of IQ packet switches

Table 1
Characterization of the considered IQ scheduling algorithms

Algorithm	Metric	Heuristic	Contention resolution	
			Input	Output
iSLIP	QO	IS	RR	RR
iOCF	CA	IS	RO	RO
MUCS	ML	MG	SS	RO
RPA	QL	RV	RO	SS

can be reduced by introducing packet-mode scheduling algorithms, since both the ORM module and the output packet FIFO can be removed. The additional constraint in this case is to keep the cells belonging to the same packet contiguous, also in output queues. To achieve this, the scheduling algorithm must enforce that, once the transfer through the switching fabric of the first cell of a packet has started towards the corresponding output port, no cells belonging to other packets can be transferred to that output, i.e., when an input is enabled to transmit the first cell of a packet comprising k cells, the input/output connection must be also enabled for the following $k - 1$ slots.

This is equivalent to having an infinite weight on the corresponding connection. Note that no conflicts can arise between infinitely weighted connections, since no more than one cell can reach a given output in one slot, hence two connections directed to the same output cannot simultaneously have an infinite weight.

We propose to extend the four considered scheduling algorithms in order to operate in packet mode. The only complexity increase in the implementation is to maintain a boolean variable at each input to flag over-prioritized connections.

4. Simulation results

To evaluate the performance of IQ, OQ, and CIOQ switching architectures, we conducted quite a large number of simulation experiments. We report results only for packet switches with $N = 16$ input/output interfaces, assuming that all input/output line rates are equal, and that only unicast traffic flows are present. Additional results are available in Ref. [44].

In IQ switches, each input queue Q_{ij} can store a finite number of cells; when a cell directed to output j arrives at input i , and queue Q_{ij} is full, the cell is lost. No buffer sharing among queues is allowed.

4.1. Traffic scenarios and performance indices

Our simulation models do not explicitly describe the arrival of IP datagrams at the packet-switch inputs. We instead model the arrival of cell bursts at the inputs of the internal cell-switch. These cell bursts are assumed to originate from the segmentation of a packet.

The cell arrival processes at input i , $A_i(k)$, is characterized by a two-state model:

ON state. When the input line is in this state, a packet is being received. The duration in slots of the ON state, i.e., the size in cells of the packet, is a discrete random variable uniformly distributed between 1 and 192. The upper value comes from the Maximum Transmission Unit (MTU) of IP over ATM [48], which is equal to 9180 octets (9140 octets of user payload, and 20 + 20 octets of TCP and IPv4 headers), to which 8 octets are added for LLC/SNAP encapsulation. AAL5 turns 9188 octets into 192 ATM cells.

OFF state. In this state no cells are received. The probability that the OFF state lasts j slots is

$$P(j) = p(1 - p)^j, \quad j \geq 0.$$

The average idle period duration in slots is $E_{\text{OFF}} = (1 - p)/p$. The parameter p is set so as to achieve the desired input load.

We shall consider two types of input/output flows, which can be described through their corresponding Γ matrix (see its definition in Section 2.1):

Uniform traffic. In this case $\gamma_{ij} = 1/N^2$, $\forall i = 1, \dots, N$ and $\forall j = 1, \dots, N$.

Hot-spot traffic. This case corresponds to a situation where all input interfaces are equally loaded, but one of the outputs receives twice as much traffic as all others; i.e., $\gamma_{1j} = 2/(N^2 + N)$ and $\gamma_{ij} = 1/(N^2 + N)$, $\forall i = 2, \dots, N$ and $\forall j = 1, \dots, N$.

Results are presented with graphs where the following performance indices are plotted versus the switch load. The latter is defined as the ratio between the input traffic load and the total capacity of input/output lines (hence, the switch load is comprised between 0 and 1).

Cell delay. This is the time spent by cells in the cell-switch queues. For this metric we shall consider the average value only.

Packet delay. This is the overall delay of a packet, considering the ISM module, the internal cell-switch queues, the ORM module, and the final packet FIFO. It is computed only for packets completely delivered at switch outputs, measuring the time from the ingress of the last cell of the packet into the ISM module until the egress of that same last cell from the final packet FIFO. Constant delay components are removed; hence a single-cell packet traverses an empty packet switch in null time, and a packet comprising k cells has a best-case delay equal to $2(k - 1)$ slots, due to wait in the segmentation and reassembly phases. We shall consider only the average value of packet delay.

Since the aim of the performance evaluation is a comparison of IQ and CIOQ architectures with OQ switches, we usually consider relative performance indices, i.e., we divide the absolute value taken by a performance index in the case of IQ and CIOQ architectures by the value taken by the same performance index in an OQ packet switch loaded with the same traffic pattern.

Simulation runs were executed until the estimate of the average cell delay reached with probability 0.95 a relative width of the confidence interval equal to 2%. The estimation of the confidence interval width is obtained with the batch means approach.

4.2. Performance of IQ switches

4.2.1. Uniform traffic – cell-mode scheduling

Fig. 8 shows, for the four considered scheduling algorithms operating in cell-mode, curves of the normalized average packet delay. Note that normalized delay values are always larger than 1, i.e., IQ switches always yield longer delays, but differences are limited to a factor 2.5, for load smaller

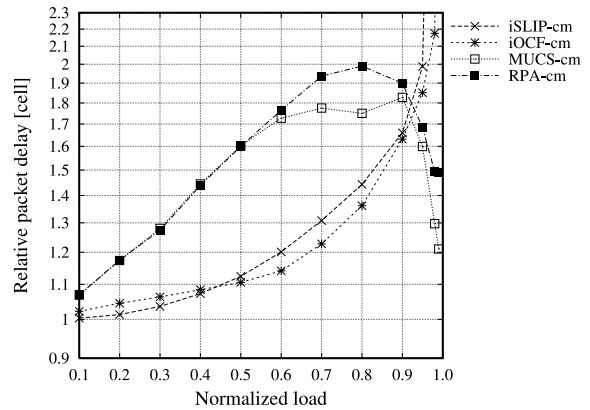


Fig. 8. Relative average packet delay for cell-mode scheduling in the uniform traffic scenario.

than 0.95. No losses were experienced with input queue lengths equal to 30,000 cells.

For loads less than 0.9, MUCS and RPA generate longer delays than iSLIP and iOCF. This is mainly due to the QL and ML metrics, which tend to equalize queue lengths: when a large packet arrives at a particular queue, the queues (at other inputs) that store small packets directed to the same output suffer a temporary starvation. iOCF, instead, provides the lowest delays among weight-aware algorithms.

4.2.2. Uniform traffic – packet-mode scheduling

Performance results are significantly different if the packet-mode scheduling proposed in this paper is considered. Curves of the relative average delays are shown in Fig. 9. Differences between the four algorithms are limited, and we observe (small) gains over OQ switches for loads up to around 0.6. IQ exhibits the largest delay advantage over OQ at loads around 0.4. To be fair in the comparison with OQ switches, we take into account, for IQ switches, delays due to ORM modules and to packet FIFOs, although these modules are not necessary in packet-mode operation, and at most one reassembly machine is active at each output, as discussed in Section 2.2. Cell delays in this same scenario, not shown here, are larger for all IQ architectures with respect to OQ at all loads.

The reductions in packet delays are interesting, specially if we consider the negligible additional

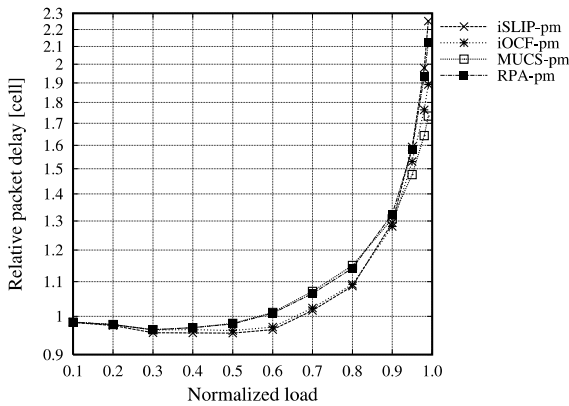


Fig. 9. Relative average packet delay for packet-mode scheduling in the uniform traffic scenario.

cost of implementing packet-mode schedulers. A simple queueing model proposed in Ref. [49] justifies this performance improvement, which is not very intuitive; it can be shown to be related with the packet length distribution. Note that more complex scheduling algorithms, better tailored to the statistics of packet traffic, could probably provide even larger gains.

If the IQ switch architecture is further simplified to take advantage of packet-mode scheduling by removing ORM modules and output packet FIFOs, the gain in average packet delay over OQ becomes much larger, since the delay necessary to reassemble packets is avoided.

4.2.3. Hot-spot traffic – cell/packet-mode scheduling

Hot-spot traffic scenarios give us the opportunity to observe the effectiveness of packet-mode scheduling when traffic is not uniform. Recall that these scenarios concentrate traffic towards one output, which has a load equal to twice the load of other outputs. The maximum admissible switch load can be analytically computed (see Ref. [44]): it is equal to $17/32 \approx 0.53$ when $N = 16$. No losses were experienced with queue lengths equal to 30,000 cells.

Figs. 10 and 11 show curves of the relative average packet delay in the hot-spot traffic scenario, with the algorithms working in cell mode and in packet mode, respectively. iSLIP and iOCF, operating in packet mode, exhibit smaller delays

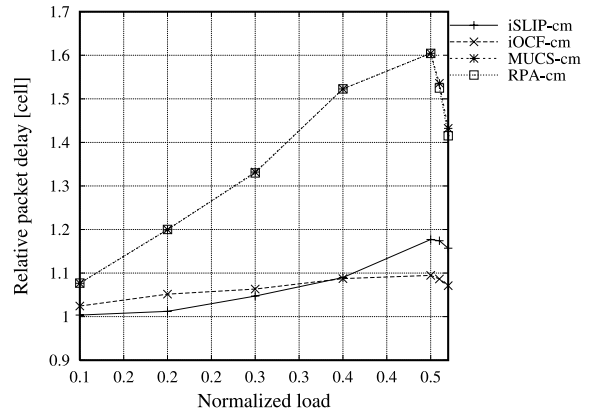


Fig. 10. Relative average packet delay for cell-mode scheduling in the hot-spot traffic scenario.

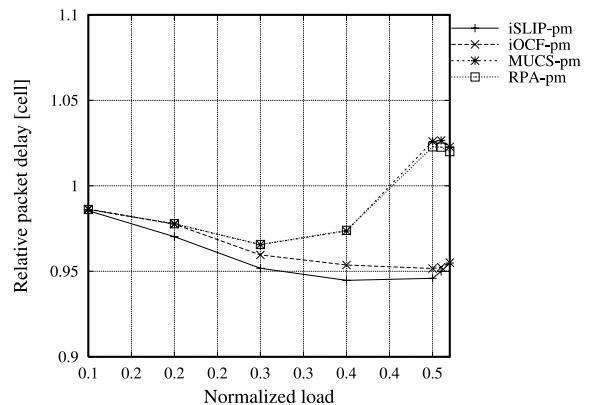


Fig. 11. Relative average packet delay for packet-mode scheduling in the hot-spot traffic scenario.

than the OQ switch for all considered loads. As mentioned above, this non-intuitive result can be shown [49] to be related to the packet length distribution. Average delays are smaller than those of OQ both for the heavily loaded output and for the $N - 1$ lightly loaded outputs. MUCS and RPA, operating in packet mode, behave similarly and exhibit a delay higher than the OQ switch for high input loads.

It is worth observing that for packet-mode algorithms, the maximum ratio between the (worst) algorithm with the maximum delay, and the (best) algorithm with the minimum delay, for the same load, is 1.08; for cell-mode algorithms this ratio is 1.47. When an algorithm operating in packet mode

is used, for high load the matching is computed only on a subset of the input-output pairs; so the metric and the heuristic matching have a reduced impact on performance.

These results are very encouraging, and tell us that simple algorithms as iSLIP can be used to very efficiently switch variable-length packets if packet mode is adopted. Without introducing non-binary weights and significant hardware complexity, iSLIP operating in packet mode achieves a partial “sensitivity” to the length of the queues, since all the cells (excluded the first one which contends for the matching) that belong to a long packet are transferred with highest priority, like in the case of a matching based on queue lengths.

Note also that the uniform traffic scenario, in which contentions involve all outputs in the same fashion, is a worst-case situation for the comparison of both IQ scheduling algorithms with OQ architectures. Similarly, uniform traffic is a worst case scenario when measuring the merits of packet-mode vs. cell-mode scheduling algorithms, as can be seen comparing Fig. 10 with Fig. 11.

4.3. Performance of CIOQ switches

This section presents simulation results for CIOQ switches using the FIFO–2 scheduling algorithm described in Section 2.1. A speed-up factor equal to 2, according to the $S^{(2)}$ speed-up definition, is supposed to be available in the internal cell-switch. We consider the uniform traffic scenario only. The capacity of all queues is taken to be unlimited for the results presented in this section, in order to be able to observe the unbalancement of queue lengths between input and output queues.

Fig. 12 plots curves of the average packet delay normalized to OQ values. The curves in the figure refer to the following switch setups, which differ in the values taken by the $SPEEDUP-IP-IN$ and $SPEEDUP-IP-OUT$ parameters (see Fig. 3 in Section 2.2), and in the operation mode (cell mode vs. packet mode) of the FIFO–2 scheduling algorithm (the extension of FIFO–2 to packet-mode operation is straightforward).

FF-121: basic FIFO–2, $SPEEDUP-IP-IN = SPEEDUP-IP-OUT = 1$. The cells belonging to IP

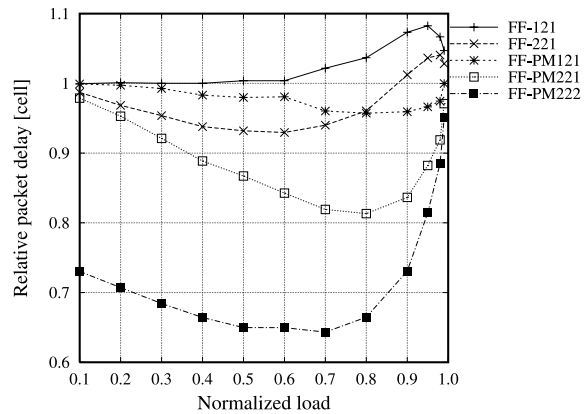


Fig. 12. Relative average packet delay in CIOQ architectures for in the uniform traffic scenario.

packets are fed to the cell-switch at the same speed of input and output lines.

FF-221: basic FIFO–2, $SPEEDUP-IP-IN = 2$, $SPEEDUP-IP-OUT = 1$. The cells belonging to IP packets are fed to the cell-switch at twice the speed of input and output lines.

FF-PM121: packet-mode FIFO–2, $SPEEDUP-IP-IN = SPEEDUP-IP-OUT = 1$. The cells belonging to the same packet are transferred to their output in contiguous slots. Note that up to two packets (received at different inputs) can be interleaved in output cell queues in this setup. An ORM module is therefore required at each output.

FF-PM221: packet-mode FIFO–2, $SPEEDUP-IP-IN = 2$, $SPEEDUP-IP-OUT = 1$. The packet-mode FIFO–2 scheduling algorithm has in this case the additional constraint that packets cannot be interleaved in output cell queues: this becomes possible without performance degradation because of $SPEEDUP-IP-IN = 2$, since two cells of the same packet can be switched to their output in one slot time. ORM modules are therefore not strictly necessary in this case: they are kept for a fair comparison with the other setups.

FF-PM222: packet-mode FIFO–2, $SPEEDUP-IP-IN = 2$, $SPEEDUP-IP-OUT = 2$. The same constraints of FF-PM221 apply to the FIFO–2 algorithm: packets are not interleaved in output queues.

In the notation above, “FF” stands for FIFO, “PM” stands for packet mode (recall the different

constraints on the scheduling algorithm when *SPEEDUP-IP-IN* equals 1 or 2), and the three digits refer to the values taken by *SPEEDUP-IP-IN*, $S^{(2)}$, and *SPEEDUP-IP-OUT*, respectively.

As expected, average packet delays at low loads (not plotted here) were observed to be around twice the average packet duration for the architectures with *SPEEDUP-IP-OUT* = 1 (where one ISM and one ORM module must be traversed by each packet), and to be around 1.5 times the average packet duration for FF-PM222 (for which the ORM can be traversed by two packets in one packet time, i.e., at double speed).

The curves in Fig. 12 show that all the considered architectures operating in packet-mode have better performance than OQ packet switches. As previously observed, IQ and CIOQ switches provide extra advantages over OQ switches when variable-size packets are considered.

Note that the FIFO-2 algorithm in packet-mode can be more efficient than output queueing also when no speed-up exists between the IP line interfaces and the internal cell-switch (i.e., for FF-PM121). The best-performing setups are FF-PM221 and FF-PM222 which avoid output packet interleaving. Delay ratios can be as small as 0.64 (i.e., 1.5 times in favor of CIOQ) when the ORM modules are present, and lower values can be achieved for the optimized architecture.

FF-121 exhibits delays slightly larger than OQ. Note that FF-121 can be considered as a simplified version of OQ, in which at most 2 (instead of N) cells can reach an output in a slot time.

It is also interesting to observe that FF-221 behaves better than FF-121, mainly thanks to the fact that the cells belonging to the same packet are kept closer to each other in the transfer through the switch.

Simulation results not presented here show that very similar behaviors were observed for CIOQ architectures also in the case of hot-spot traffic.

5. Conclusions

The paper focused on architectures and scheduling algorithms for input and combined input/output-queued packet switches, comparing them

with output queued switches in the case of variable-size data units. Similar performance results were observed for IQ, CIOQ, and OQ, with a tradeoff between the complexity of the scheduling algorithm on one side, and hardware complexity due to the internal speed-up on the other side.

Scheduling algorithms are required by all non-purely output-queued architectures. In the case of input queueing, the performance of the scheduling algorithms, and the amount of internal control information, are strongly affected by the metrics adopted by the heuristic matching. The latter aims at identifying a maximal set of packets to be transferred from input to output switch ports, avoiding conflicts in the access to the switching fabric and to output ports.

We considered four previously proposed scheduling algorithms for the transfer of fixed-size data units. These algorithms exhibited performance very close to output-queued switches.

We modified the scheduling algorithms in order to deal with variable-size packets, having in mind IP routers internally using ATM switching engines. These packet-mode scheduling algorithms require a negligible complexity increase with respect to cell scheduling, and yield performance advantages over output-queued architectures, especially when simple scheduling algorithms are adopted. This is an important result, which can impact the design of next-generation packet switches.

We also considered combined input/output architectures, which adopt a small internal speed-up (we considered a speed-up factor equal to 2), which is however sufficient to reduce the complexity of the scheduling algorithm and of the management of input queues. Our simulation results show that a speed-up equal to 2, coupled with a very simple scheduling algorithm, leads to performance very close to output-queued switches, and even better for packet-mode scheduling.

Our results can be summarized with the following two statements:

- Packet-mode scheduling in input and combined input/output-queued architectures hardly makes output-queued architectures attractive for the implementation of future-generation IP routers.

- Combined input/output-queued architectures with small speed-up factors are particularly promising. They provide better performance than output-queued architectures at lower hardware costs. When compared with input queued architectures, they move congestion to the output ports of the switch, where the implementation of schemes for QoS management is much easier and better understood.

Acknowledgements

This work was supported by the Italian Ministry for University and Research under the MQoS Project. A preliminary version of this paper was presented at the QoS-IP workshop in Rome in January 2001 [1].

References

- [1] M. Ajmone Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri, Router architectures exploiting input-queued cell-based switching fabrics, International Workshop on QoS in Multiservice IP Networks (QoS-IP 2001), January 2001, Rome, Italy, pp. 223–238.
- [2] Y. Awdeh Ra'ed, H.T. Mouftah, Survey of ATM switch architectures, *Comp. Networks ISDN Sys* 27 (12) (1995) 1567–1613.
- [3] M. Dagermark, A. Brodnik, S. Carlsson, S. Pink, Small forwarding tables for fast routing lookups, ACM SIGCOMM'97, September 1997, Cannes, France, pp. 3–14.
- [4] M. Waldvogel, G. Varghese, J. Turner, B. Plattner, Scalable high-speed IP routing lookups, ACM SIGCOMM'97, September 1997, Cannes, France, pp. 25–36.
- [5] P. Gupta, S. Lin, N. McKeown, Routing lookups in hardware at memory access speed, IEEE INFOCOM'98, April 1998, San Francisco, CA, pp. 1240–1247.
- [6] A.K. Parekh, R.G. Gallager, A generalized processor sharing approach to flow control in integrated services networks – the single node case, IEEE/ACM Trans. Networking 1 (3) (1993) 344–357.
- [7] A. Demers, S. Keshav, S. Shenker, Analysis and simulation of a fair queueing algorithm, ACM SIGCOMM'89, September 1989, Austin, TX, pp. 3–12.
- [8] S. Golestani, A self-clocked fair queueing scheme for broadband applications, IEEE INFOCOM'94, June 1994, Toronto, Canada, pp. 636–646.
- [9] J.C.R. Bennett, H. Zhang, WF²Q: Worst-case fair weighted fair queueing, IEEE INFOCOM'96, March 1996, San Francisco, CA, pp. 120–128.
- [10] T. Anderson, S. Owicki, J. Saxe, C. Thacker, High speed switch scheduling for local area networks, *ACM Trans. Comp. Sys.* 11 (4) (1993) 319–352.
- [11] H. Zhang, Service disciplines for guaranteed performance service in packet-switching networks, *Proc. IEEE* 83 (10) (1995) 1374–1396.
- [12] GFR MultiGigabit Routers, Product Overview, www.lucent.com, April 2000.
- [13] Cisco 12000 Gigabit Switch Router, Product Overview, www.cisco.com, April 2000.
- [14] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellesick, M. Horowitz, Tiny Tera: a packet switch core, *IEEE Micro Mag.* 17 (1) (1997) 27–40.
- [15] A. Hung, G. Kesidis, N. McKeown, ATM input-buffered switches with guaranteed-rate property, IEEE ISCC'98, July 1998, Athens, Greece, pp. 331–335.
- [16] H. Duan, J.W. Lockwood, S.M. Kang, J.D. Will, A high performance OC12/OC48 queue design prototype for input buffered ATM switches, IEEE INFOCOM'97, April 1997, Kobe, Japan, pp. 20–28.
- [17] C. Partridge et al., A 50-Gb/s IP router, *IEEE Trans. Networking* 6 (3) (1998) 237–248.
- [18] C. Minkenberg, T. Engbersen, A combined input and output queued packet-switched system based on PRIZMA switch on-a-chip technology, *IEEE Commun. Mag.* 38 (12) (2000) 70–77.
- [19] D. Stiliadis, A. Varma, Providing bandwidth guarantees in an input buffered crossbar switch, IEEE INFOCOM'95, April 1995, Boston, MA, pp. 960–968.
- [20] N. McKeown, iSLIP: a scheduling algorithm for input-queued switches, *IEEE Trans. Networking* 7 (2) (1999) 188–201.
- [21] M. Ajmone Marsan, A. Bianco, E. Leonardi, L. Milia, RPA: a flexible scheduling algorithm for input buffered switches, *IEEE Trans. Commun.* 47 (12) (1999) 1921–1933.
- [22] N. McKeown, A. Mekkittikul, A practical scheduling algorithm to achieve 100% throughput in input-queued switches, IEEE INFOCOM'98, April 1998, New York, NY, pp. 792–799.
- [23] R. Schoenen, G. Post, G. Sander, Weighted arbitration algorithms with priorities for input-queued switches with 100% throughput, BSS'99, Third IEEE International Workshop on Broadband Switching Systems, Kingston, Canada, June 1999.
- [24] M. Ajmone Marsan, A. Bianco, E. Filippi, P. Giaccone, E. Leonardi, F. Neri, On the behavior of input queuing switch architectures, *Eur. Trans. Telecommun.* 10 (2) (1999) 111–124.
- [25] N. McKeown, T.E. Anderson, A quantitative comparison of scheduling algorithms for input-queued switches, *Comp. Networks ISDN Sys.* 30 (24) (1998) 2309–2326.
- [26] S.-T. Chuang, A. Goel, N. McKeown, B. Prabhakar, Matching output queueing with a combined input/output-queued switch, *IEEE J. Selected Areas Commun.* 17 (6) (1999) 1030–1039.
- [27] I. Stoica, H. Zhang, Exact emulation of an output queueing switch by a combined input output queueing switch, IWQoS'98, Sixth International Workshop on Quality of Service, Napa, CA, May 1998, pp. 218–224.

- [28] P. Krishna, N.S. Patel, A. Charny, R. Simcoe, On the speedup required for work-conserving crossbar switches, IWQoS'98, Sixth International Workshop on Quality of Service, Napa, CA, May 1998, pp. 225–234.
- [29] A. Charny, P. Krishna, N.S. Patel, R. Simcoe, Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup, IWQoS'98, Sixth International Workshop on Quality of Service, Napa, CA, May 1998, pp. 235–244.
- [30] M. Ajmone Marsan, E. Leonardi, M. Mellia, F. Neri, On the stability of input-buffer cell switches with speed-up, IEEE INFOCOM 2000, March 2000, Tel Aviv, Israel.
- [31] J.G. Dai, B. Prabhakar, The throughput of dataswitches with and without speedup, IEEE INFOCOM 2000, March 2000, Tel Aviv, Israel.
- [32] F.M. Chiussi, Y. Xia, V.P. Kumar, Backpressure in shared-memory-based ATM switches under multiplexed bursty sources, IEEE INFOCOM'96, March 1996, San Francisco, CA, pp. 830–843.
- [33] M. Karol, M. Hluchyj, S. Morgan, Input versus output queuing on a space division switch, IEEE Trans. Commun. 35 (12) (1987) 1347–1356.
- [34] Y. Tamir, G. Frazier, High performance multi-queue buffers for VLSI communication switches, 15th Annual Symposium on Computer Architecture, June 1988, Washington, DC, pp. 343–354.
- [35] Y. Tamir, H.C. Chi, Symmetric crossbar arbiters for VLSI communication switches, IEEE Trans. Parallel Distributed Sys. 4 (1) (1993) 13–27.
- [36] N. McKeown, Scheduling algorithms for input-queued cell switches, Ph.D. Thesis, University of California, Berkeley, 1995.
- [37] J.S. Chen, T.E. Stern, Throughput analysis optimal buffer allocation and traffic imbalance study of a generic non-blocking packet switch, IEEE J. Selected Areas Commun. 9 (3) (1991) 439–449.
- [38] A.K. Gupta, N.D. Georganas, Analysis of a packet switch with input and output buffers and speed constraints, IEEE INFOCOM'91, Bal Harbour, FL, April 1991, pp. 694–700.
- [39] I. Iliadis, W.E. Denzel, Performance of packet switches with input and output queuing, IEEE ICC'90, April 1990, Atlanta, GA, pp. 747–753.
- [40] Y. Oie, M. Murata, K. Kubota, H. Miyahara, Performance analysis of non blocking packet switch with input and output buffers, IEEE Trans. Commun. 40 (8) (1992) 1294–1297.
- [41] Y. Oie, T. Suda, M. Murata, H. Miyahara, Survey of the performance of non blocking switches with FIFO input buffers, IEEE ICC'90, April 1990, Atlanta, GA, pp. 737–741.
- [42] Y. Yeh, M.G. Hluchyj, A.S. Acampora, The Knockout switch: a simple modular architecture for high performance packet switching, IEEE J. Selected Areas Commun. 5 (1987) 1274–1283.
- [43] N. McKeown, A. Mekikittikul, A starvation free algorithm for achieving 100% throughput in an input queued switch, ICCCN'96, October 1996, Rockville, MD, pp. 694–700.
- [44] P. Giaccone, Tecniche di accodamento e trasferimento in architetture di commutazione a larga banda, Laurea Thesis, Politecnico di Torino, May 1998 (in Italian).
- [45] R.E. Tarjan, Data Structures and Network Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, November 1983.
- [46] S. Even, O. Kariv, An $O(n^{2.5})$ algorithm for maximum matching in general graphs, 16th Symposium on Foundations of Computer Science, October 1975, Berkeley, CA, pp. 100–112.
- [47] J.E. Hopcroft, R.M. Karp, An $n^{2.5}$ algorithm for maximum matching in bipartite graphs, Soc. Ind. Appl. Math. J. Comput. 2 (4) (1973) 225–231.
- [48] R. Atkinson, RFC 1626: Default IP MTU for use over ATM AAL5, May 1994.
- [49] M. Ajmone Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri, Packet scheduling in input-queued cell-based switches, IEEE INFOCOM'01, April 2001, Anchorage, AL.



Marco Ajmone Marsan is Full Professor at the Electronics Department of Politecnico di Torino, in Italy. He was born in Torino, Italy, in 1951. He holds degrees in Electronic Engineering from Politecnico di Torino and University of California, Los Angeles. Since November 1975 to October 1987 he was at the Electronics Department of Politecnico di Torino, first as a researcher, then as an Associate Professor. From November 1987 to October 1990 he was a Full Professor at the Computer Science Department of the

University of Milan, in Italy. During the summers of 1980 and 1981 he was with the Research in Distributed Processing Group, Computer Science Department, UCLA. During the summer of 1998 he was an Erskine Fellow at the Computer Science Department of the University of Canterbury in New Zealand. He has coauthored over 200 journal and conference papers in the areas of Communications and Computer Science, as well as the two books “Performance Models of Multiprocessor Systems” published by the MIT Press, and “Modelling with Generalized Stochastic Petri Nets” published by John Wiley. He received the best paper award at the Third International Conference on Distributed Computing Systems in Miami, FL., in 1982. His current interests are in the fields of performance evaluation of communication networks and their protocols. M. Ajmone Marsan is a Fellow of IEEE.



Andrea Bianco is Assistant Professor at the Electronics Department of Politecnico di Torino, in Italy. He was born in Torino, Italy, in 1962. He holds a Dr. Ing. degree in Electronics Engineering since 1986 and a Ph.D. in Telecommunications Engineering since 1993, both from Politecnico di Torino. Since 1994 he was an Assistant Professor at Politecnico di Torino, first in the Dipartimento di Sistemi di Produzione, later in the Dipartimento di Elettronica. In 1993 he visited Hewlett-Packard Labs in Palo Alto, CA. He

has co-authored over 50 papers published in international

journals and presented in leading international conferences. His current research interests are in the fields of protocols for all-optical networks and switch architectures for high-speed networks. A. Bianco is a member of IEEE.



Paolo Giaccone is a Ph.D. student at the Electronics Department of Politecnico di Torino, in Italy. He was born in Torino, Italy, in 1973. He holds a Dr. Ing. degree in Telecommunications Engineering from Politecnico di Torino since 1998. During the summer 1998, he visited the High Speed Networks Research Group at Lucent Technology, Holmdel. He is currently visiting the research group of Prof. Prabhakar at Stanford University. His main research interest is in the field of scheduling algorithms for input

queued switches. P. Giaccone is a student member of IEEE.



Emilio Leonardi is Assistant Professor at the Electronics Department of Politecnico di Torino, in Italy. He was born in Cosenza, Italy, in 1967. He received a Dr. Ing degree in Electronics Engineering in 1991 and a Ph.D. in Telecommunications Engineering in 1995, both from Politecnico di Torino. In 1995, he spent one year at the Computer Science Department of the University of California, Los Angeles (UCLA), where he was involved in the Supercomputer-SuperNet (SSN) project aimed at the design of a high capacity optical network architecture. During the summer 1999,

he joined the High-Speed Research Group of Lucent Technologies, in Holmdel, NJ, where he worked on the design of scheduling algorithms for high capacity switch architectures. He has co-authored over 50 papers published in international journals and presented in leading international conferences. His research interests are in the fields of all-optical networks, switching architectures, and queueing theory. E. Leonardi is a member of IEEE.

he joined the High-Speed Research Group of Lucent Technologies, in Holmdel, NJ, where he worked on the design of scheduling algorithms for high capacity switch architectures. He has co-authored over 50 papers published in international journals and presented in leading international conferences. His research interests are in the fields of all-optical networks, switching architectures, and queueing theory. E. Leonardi is a member of IEEE.



Fabio Neri is Full Professor at the Electronics Department of Politecnico di Torino, Turin, Italy. He was born in Novara, Italy, in 1958. He received his Dr. Ing. and Ph.D. degrees in Electrical Engineering from Politecnico di Torino in 1981 and 1987, respectively. From 1991 to 1992 he was with the Information Engineering Department at University of Parma, Parma, Italy, as an Associate Professor. From 1982 to 1983 he was a visiting scholar at George Washington University in Washington, DC. He has been a visit-

ing researcher at the Computer Science Department of the University of California at Los Angeles, CA (Summer 1995), at the Optical Data Networks Research Department of Bell Laboratories/Lucent Technologies in Holmdel, NJ (Summer 1998), and at British Telecom Advanced Communication Research in Ipswich, UK (Summer 2000). His research interests are in the fields of performance evaluation of communication networks, high-speed and all-optical networks, packet switching architectures, discrete event simulation, and queueing theory. He has recently been involved in several European projects on WDM networks. He has co-authored over 100 papers published in international journals and presented in leading international conferences. Fabio Neri is a member of the IEEE Communications Society and served several IEEE conferences and journals.