

A Comparison of Input Queuing Cell Switch Architectures

M.Ajmone Marsan¹, A.Bianco¹, E.Filippi²
P.Giaccone¹, E.Leonardi¹, F.Neri¹

¹ Dipartimento di Elettronica, Politecnico di Torino, Italy

² CSELT, Torino, Italy

Fax number: +39 011 5644099

e-mail: {ajmone,bianco,leonardi,neri}@polito.it, {filippi@cse.lt.it}

Abstract—

Many proposals of input queuing cell switch architectures have recently appeared in the literature. Some have even found application in commercial very high speed IP routers. In this paper we discuss the pros and cons of input and output queuing switch architectures, we provide a taxonomy of scheduling algorithms for input queuing switches, and we present comparative performance results for some of the recent proposals of input queuing cell switches. Performance is measured in terms of cell loss probability, as well as average, standard deviation, and 99-th quantile of the cell delay with different traffic patterns. The complexity of the algorithms, and the amount of control information to be exchanged inside the switch for their execution, are also discussed.

I. INTRODUCTION

In output queuing (OQ) packet switch architectures, packets arriving from input lines are not queued at input interfaces; rather, they cross the switching fabric, and join a queue at the switch output interface before being forwarded to the next node (see Fig. 1.a). In order to avoid contention either within the switching fabric or in the access to the output queue, it is necessary that the internal speed of an OQ switch, as well as the access rate of the memory at each output interface, be equal to the sum of all input line rates (i.e., they must equal the aggregate switch bandwidth).

On the contrary, in input queuing (IQ) packet switch architectures, packets arriving from input lines are queued at input interfaces. They are then extracted from input queues to cross the switching fabric and be forwarded to the next node, according to some algorithm that avoids contention within the switching fabric and at the input and output interfaces (no more than one packet at a time can be extracted from each input and reach each output). Note that the implementation of this algorithm requires coordination among input interfaces, hence the exchange of control information, which can either contend with user data for access to the switching fabric, or use a separate data path. The internal speed of an IQ switch need not be larger than the highest line rate (see Fig. 1.b), possibly incremented to account for extra packet headers added internally to the switch.

From this preliminary description of the IQ and OQ approaches, it is immediately evident that the intelligent packet scheduling of IQ compensates for the missing internal speedup of OQ. However, the comparison of the IQ and OQ approaches deserves deeper investigations.

This work was supported by a research contract between CSELT and Politecnico di Torino

If FIFO (first in first out) packet queues are used, IQ architectures suffer from the head of the line (HoL) blocking phenomenon, that severely limits their performance. In 1987, Karol, Hluchyj and Morgan [1] proved with an asymptotic analysis that under source- and destination-uniform traffic, HoL blocking limits the maximum throughput of IQ switches with an arbitrarily large number of input/output lines to $2 - \sqrt{2} \approx 0.586$. This result is quite disappointing, specially considering that, without any buffering in the switch (except for the obvious need to store one packet at the switch ingress and egress), i.e., by discarding packets that cannot be immediately switched, the throughput limit grows to 0.632 [2]. The HoL blocking phenomenon can be circumvented with more complex structures of the input queues, for example separating (either logically or physically) at each input the queues of packets referring to different outputs (see Fig. 1.c), as we shall see.

Moreover, OQ offers the possibility of rather easily controlling the delay of packets within the switch (with algorithms such as fair queuing), since the time needed to cross the switching fabric is known, and all packets following a given route can be found within the same (output) queue.

The situation is such that packet switch designs traditionally referred to OQ architectures, specially in the case of ATM (Asynchronous Transfer Mode) switches, because of their greater effectiveness, and because the requirement for an internal speedup was not critical.

The recent developments in networking, that produced a dramatic growth of line rates, have however made the internal speed requirements of OQ switches difficult to meet. This has generated great interest in IQ switch architectures, thus opening a lively new research line in switching.

A few implementations of IQ switches exist, either prototypical or commercial. Probably the most famous IQ cell switch prototype is the *Tiny Tera*, implemented at Stanford University in cooperation with Texas Instruments [3], [4]. The latest version of *Tiny Tera* can reach an aggregate bandwidth of 1 Tbps. A similar architecture is implemented in the recent line of Cisco Gigabit routers, the Cisco 12000 series, which adopts a core IQ switching fabric reaching an aggregate bandwidth equal to 60 Gbps. Other examples of commercial IQ switching fabrics are the Lucent Cajun Switch and the Yago PowerCast Switch.

A number of novel algorithms for IQ switch architectures have recently appeared in the technical literature, aiming at over-

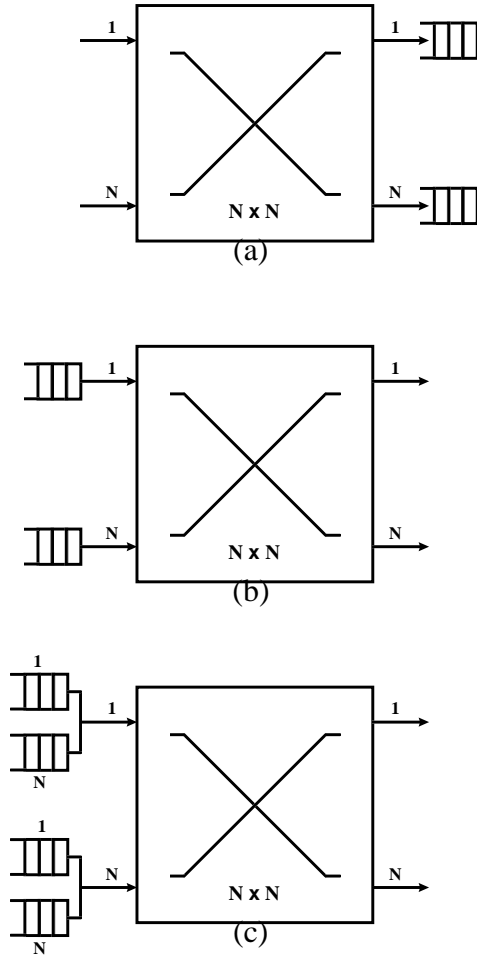


Fig. 1. Architectures of packet switches: a) one queue at each output interface b) one queue at each input interface c) separate queues at each input interface

coming the HoL blocking performance limitations with minimal complexity. In this paper we discuss and compare a number of such proposals in the context of packet switches operating on fixed-size data units. The term *cell* switches will be used, although the discussed switch architectures need not refer to ATM. The considered IQ cell-switch architecture proposals are: iSLIP [5], [6], iLQF [5], [7], iOCF [5], [7], iLPF [3], 2DRR [8], RC [9], MUCS [10], RPA [11], [12], and iZIP [13], a novel proposal.

II. A CLOSER LOOK AT IQ CELL SWITCH ARCHITECTURES

We consider IQ switch architectures with synchronous operations, where time is slotted, and where fixed-size data units are transferred in each slot from input interfaces to output interfaces. Borrowing from the ATM jargon, the term *cell* will be used to refer to data units, but the context that we consider is not restricted to ATM. 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.

For the operation of IQ switch architectures, input cells must

be stored into the memory available at input interfaces; the situation of input interfaces must then be communicated to a switch controller that executes a scheduling algorithm to decide which cells must be transferred from inputs to outputs. Thus, the key elements of an IQ cell switch architecture are

1. the input queue organization
2. the scheduling algorithm
3. the switch control scheme

These three elements are separately discussed below.

A. The input queue organization

The input queue organization is crucial to overcome the HoL blocking phenomenon: with a single FIFO queue at each input, it may happen that no packet can be transmitted from an input because of contention of the first packet in the FIFO, even when other packets that could be transmitted without contention lie behind in the same queue.

A frequently proposed queue organization directly derives from the observation that HoL blocking is due to the impossibility to extract from a FIFO queue a packet that is not at the head of the queue. With *n-window queuing*, also called *bypass queuing* or *look-ahead contention resolution*, if the packet at the head of the queue cannot be transferred to the intended output because of contention, the second packet is considered, and so on, up to the n -th queue position. Obviously, FIFO queuing is n -window queuing with $n = 1$.

The performance of IQ switches adopting this queue organization grows with n ; an approximate analysis [2] reveals that a complete visibility on the input queues is necessary to achieve the optimal switch throughput. For this reason, some recent proposals of IQ cell switch architectures adopt a queue organization with full separation at each input of the cells directed toward different outputs. This queue organization is called *destination queuing* or *virtual output queuing*. IQ switch architectures with N input/output lines and virtual output queuing comprise N queues at each input, thus a total of N^2 queues, each storing the cells going from one input to one output (see Fig. 1.c).

B. The scheduling algorithm

The term *scheduling algorithm* in IQ cell switch architectures can refer to two different types of schedulers:

1. *switching matrix schedulers*, that decide which input interface is enabled to transmit among those that need to transfer cells to a specific output interface; they avoid blocking and solve contention within the switching fabric;
2. *flow-level schedulers*, that decide which cell flows must be served in accordance to Quality of Service (QoS) requirements.

Actually, the traditional use of the term *scheduling algorithm* in OQ cell switches refers to flow-level schedulers, such as the different versions of fair queuing algorithms [14], whereas in IQ cell switches it refers to switching matrix schedulers, since the research work on flow-level schedulers for IQ switches started only very recently [?], [16].

In this paper the term *scheduling algorithm* will be used to refer to switching matrix schedulers.

In the technical literature, scheduling algorithms in IQ cell switch architectures are described as solutions of the matching problem in bipartite graphs. This happens because the switch

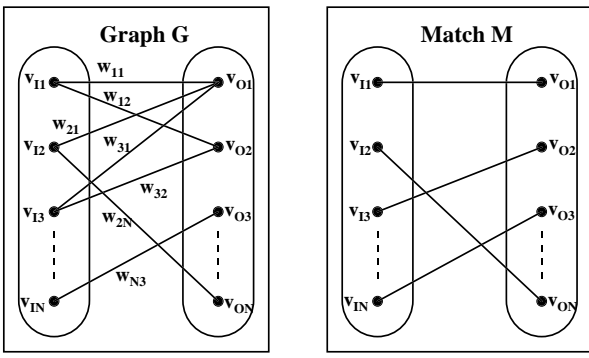


Fig. 2. Bipartite graph description of a packet switch

state can be described as a bipartite graph $G = [V, E]$ (see Fig. 2) in which the graph vertices in set V are partitioned in two subsets: subset V_I , whose elements v_{Ij} 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_{Ij} to v_{Ok} indicates the need for cell transfers from input j to output k), and can be labeled with a metric that will be denoted by w_{jk} . 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 exists, or it could refer to the number of cells to be transferred, to the time waited by the oldest cell, or to other state indicators, as we shall discuss. We assume $w_{jk} = 0$ for missing edges in G , i.e., when no cell from input j to output k are waiting in input queues.

A *match* 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 match has *maximum size* if the number of edges is maximized; a match has *maximum weight* if the sum of the edge metrics is maximized. A match is called *maximal* if no other admissible edge can be added to the current match in order to increase either its size or its weight.

The different IQ cell switch architectures that we shall discuss use different scheduling algorithms, hence different 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 $O(N^3 \log N)$ for maximum weight matching [17, Chapt. 8], which can be proved to yield the maximum achievable throughput using as metrics either the number of cells to be transferred or the time waited by the oldest cell; it is $O(N^{5/2})$ for maximum size matching [17], [18].

C. The switch control scheme

With the term *switch control scheme* we refer to

- the structure of the controller that executes the scheduling algorithm
- the control information used in the execution of the scheduling algorithm
- the data path and protocol used for the exchange of control information

The scheduling algorithm can be either executed by a centralized controller, or executed in a distributed manner over the input line interfaces. When the controller is centralized, it is necessary to transfer from input interfaces to the centralized controller the state information that the controller uses for the execution of the scheduling algorithm. The result of the scheduling must then be returned to input interfaces.

When the controller is instead distributed over line interfaces, it is necessary to transfer from each interface to all others the information that is required for the execution of the scheduling algorithm at each interface. The information and/or the computation required by each distributed controller can be less than the information and/or the computation required by the centralized controller.

Note that scheduling algorithms may be intrinsically centralized or distributed, in the sense that they were originally designed for either a centralized or a distributed implementation. However, an intrinsically centralized scheduling algorithm can be implemented on a distributed controller by letting each line interface operate on the same information of the centralized controller, hence executing in parallel at each interface the centralized algorithm over the same data. In this case there is no need to distribute the result of the scheduling (but the overall computational power required by the system is increased by a factor N in the worst case). Conversely, an intrinsically distributed scheduling algorithm can be implemented in a centralized controller only with an increase in complexity with respect to each distributed execution.

In the discussion of the different scheduling algorithms of Section IV, we shall indicate whether they are naturally matched with either a centralized or a distributed controller, but we shall always assume a centralized controller when presenting complexity results.

The control information exchanged in each slot for the execution of the scheduling algorithm can either refer to the present state of line interfaces, or describe the new events that induced a local state change in the last slot, so as to transfer information in an incremental fashion. This means that the line interfaces can either inform the controller (whether centralized or distributed) of their present state slot by slot, hence distributing redundant information, but being quite tolerant to errors or losses in the information transfer, or just distribute the information about events (cell arrivals, etc.) that modify their local state, thereby being more vulnerable to errors or losses in the information transfer. We consider the latter “incremental” case in this paper.

The amount of information that is sent to the centralized controller, in the case of a switch with N line interfaces with equal speed, independently of the scheduling algorithm, is $N \lceil \log_2(N + 1) \rceil$ bits, since each line interface must indicate

whether a new cell arrived, and what is its intended output. After the scheduling algorithm at the centralized controller has selected a maximal match, the match has to be transferred back to the enabled input interfaces. This requires again the transfer of $N \lceil \log_2(N+1) \rceil$ bits.

The control information can either contend with user data for the switching fabric or use a dedicated data path. Moreover, the transfer of control information can exploit a native broadcast facility, or require the transfer of multiple copies. The different alternatives may produce quite different performance of the IQ switch architectures. We shall assume that a dedicated data path is available, with broadcast capability.

III. NOTATION AND ASSUMPTIONS

As already mentioned, N will denote the number of input/output interfaces of the switch.

The $N \times N$ matrix whose elements are the edge metrics 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 cell time (called *slot*), we shall write $\mathbf{W}(\mathbf{k}) = [w_{ij}(\mathbf{k})]$.

Define the normalized weight matrix $\mathbf{M} = [\mu_{ij}]$, with $\mu_{ij} = \frac{w_{ij}}{\sum_{i=1}^N \sum_{j=1}^N w_{ij}}$.

The cell arrival process at input interface i , $1 \leq i \leq N$, is denoted by $A_i(k)$; this is a discrete-time process according to which, at the beginning of each slot, at each input interface up to one new cell can be generated.

Cell arrival processes will be taken to be Bernoulli: at each slot a cell arrives at an input interface with a given probability (say p_i at input i), and no cell arrives with the complementary probability $(1 - p_i)$. The value of p_i coincides with the input interface load.

Each incoming cell contains a field through which it is possible to identify the switch output interface j , $1 \leq j \leq N$, to which the cell must be transferred. We do not explicitly address issues related to table look-up and label conversion. A cell arriving at input i and directed to output j is stored in queue Q_{ij} , whose occupancy at time k is denoted by $L_{ij}(k)$. Any queue can contain up to L cells.

By separating within the cell arrival process $A_i(k)$ at input interface i the cells referring to different destinations we obtain the arrival processes $A_{ij}(k)$, whose average cell rates, relative to the line speed, are denoted by λ_{ij} . By aggregating all arrival processes $A_i(k)$ we obtain the overall switch arrival process $A(k)$. $A(k)$ is said to be *admissible* when no input or output is overloaded, i.e., $\sum_{i=1}^N \lambda_{ij} < 1$, and $\sum_{j=1}^N \lambda_{ij} < 1$; otherwise, $A(k)$ is said to be *inadmissible*.

Define the cell arrival rate matrix $\mathbf{\Lambda} = [\lambda_{ij}]$, and the normalized cell arrival rate matrix $\mathbf{\Gamma} = [\gamma_{ij}]$, with $\gamma_{ij} = \frac{\lambda_{ij}}{\sum_{i=1}^N \sum_{j=1}^N \lambda_{ij}}$.

We shall consider two types of cell arrival flows, that can be described through their corresponding normalized cell arrival rate matrices:

Uniform traffic. In this case $\gamma_{ij} = \frac{1}{N^2}$, $\forall(i, j)$, $1 \leq i \leq N$, $1 \leq j \leq N$.

Unbalanced traffic. In this case we use a normalized cell arrival rate matrix of the type:

$$\mathbf{\Gamma} = \frac{1}{N} \begin{pmatrix} \delta & 1 - \delta & 0 & 0 \\ 0 & \delta & 1 - \delta & 0 \\ 0 & 0 & \delta & 1 - \delta \\ 1 - \delta & 0 & 0 & \delta \end{pmatrix}$$

where $0 < \delta < 1$ is an unbalancement factor.

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

$$t_{ij} = \begin{cases} 1 & \text{if the match includes the cell transfer from } i \text{ to } j \\ 0 & \text{if the match does not include the cell transfer from } i \text{ to } j \end{cases} \quad (1)$$

IV. A TAXONOMY OF IQ SCHEDULING ALGORITHMS

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

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

2. *Heuristic matching.* Computation of the cell transfer matrix $\mathbf{T}(\mathbf{k}) = [t_{ij}(\mathbf{k})]$. 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.

3. *Contention resolution.* In the execution of the heuristic algorithm for the determination of an almost optimal match, a strategy is necessary to solve contentions due to edges with equal metrics, at the source or at the destination. The contention resolution typically is either random (random order – RO) or based on a deterministic scheme; in the latter case, we consider round robin (RR) and sequential search (SS), the difference being 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, that are instead interlaced.

The phase that has the most profound impact on performance is the metrics computation, whereas the different heuristics to obtain good matches have a deep impact on the algorithm complexity.

A. Metrics computation

As regards metrics, the most common choices are the following:

- **QO** (*Queue occupancy*). In this case $w_{ij} = u(L_{ij})$ where $u(\cdot)$ is the unit step function. This is the metrics adopted by iSLIP, 2DRR and RC. The adoption of this metrics leads to the search for a maximal size match. All other metrics lead to the search for a maximal weight match, with different weights.
- **QL** (*Queue length*). The metrics is the number of cells in the queue: $w_{ij} = L_{ij}$. This is the metrics adopted by iLQF, RPA, and iZIP.
- **CA** (*Cell age*). The metrics is the time already spent in the queue by the cell at the queue head. This is the metrics adopted by iOCF.
- **IL** (*Interface load*). The metrics is the total number of cells queued at the input and output interfaces:

$$w_{ij} = \sum_{k=1}^N L_{ik} + \sum_{k=1}^N L_{kj}$$

This is the metrics adopted by iLPF, with the aim of giving priority to most congested flows.

- **MB** (*MUCS Binary*). The binary metrics adopted by the MUCS-bin algorithm is defined as:

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

The rationale for this metrics is to consider first in the match definition, i.e., to provide with a larger metrics, those input/output connections for which less alternatives exist. The maximum value of the metrics is achieved when only one packet waits at input i , and it is the only packet in the switch directed to output j . That packet must be selected in the match if we do not want input i and output j to remain idle in the current slot. A packet waiting at an input where other packets with other destinations exist, and/or directed to an output at which packets from other inputs are also directed, receives a lower metrics value.

- **ML** (*MUCS Length*). The metrics adopted by the MUCS-len algorithm is similar in its rationale to that of MUCS-bin, but refers to queue lengths:

$$w_{ij} = \frac{L_{ij}}{\sum_{k=1}^N L_{ik}} + \frac{L_{ij}}{\sum_{k=1}^N L_{kj}}$$

B. Heuristic matching

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 metrics to the relevant output interfaces ($w_{ij}(k)$ is sent from input interface i to output interface j). Output interfaces select one among the arriving requests by choosing the largest metrics 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 metrics 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 is the heuristics adopted by iSLIP, iLQF, and iOCF

- **MRR** (*Matrix round robin*). Consider the $N \times N$ matrix $\mathbf{W} = [w_{ij}]$. In this matrix it is possible to identify $N!$ generalized diagonals (a generalized diagonal is a set of elements such that no two elements are in the same row or column), N of which are sufficient to cover the matrix. Consider a set of N covering generalized diagonals. The algorithm consists of N steps, and starts from one of the generalized diagonals, enabling the transmission requests that are found in its elements. When a transmission request is enabled, the corresponding row and column are put to zero in the matrix. Then, all other generalized diagonals are considered in sequence, enabling the transmission requests corresponding to nonzero elements. At the end of the algorithm, the starting point for the next search is chosen by changing the set of covering diagonals, so as to improve fairness. This is the heuristics adopted by 2DRR and RC (the two use different fairness approaches).

- **MG** (*Matrix greedy*). Consider again matrix \mathbf{W} . 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 metrics exist; otherwise a conflict resolution is necessary) before reducing the matrix by eliminating the entries corresponding to enabled cell transfers. This is the heuristics adopted by MUCS-bin, and MUCS-len.

- **PA** (*Preordering and arbitration*). Consider again matrix \mathbf{W} . In this case the algorithm, which is similar to MG, consists of two steps, in the first of which matrix \mathbf{W} is modified by permuting rows and columns so that their total weights (the sums of the weights in each row or column) are in decreasing order. With the second step, cell transmissions are enabled by sequentially considering inputs (rows) and outputs (columns), making sure that no conflict arises. This is the heuristics adopted by iLPF.

- **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 metrics, possibly overwriting requests with lower metrics 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 heuristics adopted by RPA-sta, and RPA-ada, which differ in the order of the sequential access.

- **RB** (*Reservation and blocking*). In this case the algorithm consists of a sequence of exactly N steps, in which inputs are orderly enabled, one at the time. Each step comprises two phases. In the first phase, the enabled input selects its queue with highest metrics, breaking possible ties with a (input) contention resolution algorithm, and sends a reservation information to the selected output interface. In the second phase the output interface that received the reservation, signals to the $N - 1$ input interfaces that are not active in this step that it was blocked. Successive inputs select their non blocked queue with highest metrics, and

Algorithm	Metrics	Heuristics	Contention resolution		Centralized or distributed
			Input	Output	
iSLIP	QO	IS	RR	RR	D
iLQF	QL	IS	RO	RO	D
iOCF	CA	IS	RO	RO	D
iLPF	IL	PA	RO	SS	C
2DRR	QO	MRR	SS	SS	C
RC	QO	MRR	SS	SS	C
MUCS-bin	MB	MG	SS	RO	C
MUCS-len	ML	MG	SS	RO	C
RPA-sta	QL	RV	RO	SS	D
RPA-ada	QL	RV	RO	RR	D
iZIP	QL	RB	RR	SS	D

TABLE I

CHARACTERIZATION OF THE CONSIDERED IQ SCHEDULING ALGORITHMS

Algorithm	Additions and subtractions	Products and divisions	Comparisons	Asymptotic complexity
iSLIP	-	-	$3N^2 \log_2 N$	$O(N^2 \log_2 N)$
iLQF	-	-	$3N^2 \log_2 N$	$O(N^2 \log_2 N)$
iOCF	-	-	$3N^2 \log_2 N$	$O(N^2 \log_2 N)$
iLPF	$2N^2$	-	$N^2 + 2N \log_2 N$	$O(N^2)$
2DRR	-	-	N^2	$O(N^2)$
RC	-	-	N^2	$O(N^2)$
MUCS	$\approx N^3$	$\approx \frac{1}{3}N^3$	$\approx \frac{1}{3}N^3$	$O(N^3)$
RPA	N^2	-	$2N^2$	$O(N^2)$
iZIP	-	-	$2N^2$	$O(N^2)$

TABLE II

COMPUTATIONAL COMPLEXITY OF THE DIFFERENT IQ SCHEDULING ALGORITHMS

proceed as described above.

This is the heuristics adopted by iZIP.

The above discussions are summarized in Table I, where for the considered IQ scheduling algorithms we give the type of metrics used, the heuristic algorithm adopted for the identification of matches, the approach used for contention resolution, and the most natural type of implementation (centralized or distributed), according to the original description of the algorithm.

V. COMPLEXITY OF IQ SCHEDULING ALGORITHMS

Recall that we assume that scheduling algorithms are executed by a centralized controller.

The computational complexity of an IQ scheduling algorithm is a crucial parameter, since the algorithm must be run at every cell time. In the computation of the complexity of the different algorithms we have considered the maximum number of elementary operations to be completed in a whole execution of the algorithm. By elementary operations we mean additions, subtractions, products, divisions, comparisons and orderings (the complexity of the ordering of k elements is taken to be $\log_2 k$). Initially we just provide the results for the total number of operations required to execute the algorithms on one centralized processor. Later we comment on the possibility of distributing and/or parallelizing the algorithms.

In Table II we report results about the complexity of the considered algorithms. It is worth noting that the asymptotic complexity obtained with the different heuristics is not much less than that of the optimum match determination (recall that the complexity of the computation of the maximum size matching is $O(N^{2.5})$, whereas that for the computation of the maximum weight matching is $O(N^3 \log_2 N)$).

A very appealing solution to speed the execution of the

scheduling algorithms consists in a parallel implementation at the centralized controller. Those algorithms that were conceived for a distributed implementation (iSLIP, iLQF, iOCF, RPA, and iZIP) can be easily parallelized at the controller over a number of processors equal to the number N of input/output interfaces. This can entail a performance improvement due to the reduction of the asymptotic complexity managed by each processor (reported in Table II). However, not all of these algorithms reduce their execution time when parallelized, mainly due to the presence of sequential choices: RPA and iZIP, in particular, do not benefit from a parallel implementation.

Also the algorithms that were conceived for a centralized implementation can take large advantage from a parallel implementation. This is true in particular for all those algorithms that perform matrix manipulations (iLPF, 2DRR, RC, and MUCS).

For the sake of conciseness, in this paper we do not dig further into the problem of parallel implementations of the scheduling algorithms.

VI. PERFORMANCE OF IQ SCHEDULING ALGORITHMS

A. Asymptotic results

Before considering the results of a simulative study of the different IQ scheduling algorithms, we present their analysis in asymptotic conditions, in order to gain some insight into the behavior of different classes of algorithms under various traffic patterns.

By asymptotic analysis we mean that the model that is proposed below does not capture the details of the heuristic algorithm execution, but is just based on the characteristics induced by the adoption of a specific metrics. The model considers switches with N input/output lines with equal rate, and assumes a stationary behavior, infinite queue sizes, an admissible workload close to saturation, and is just concerned with the management of input queues, thus disregarding the characteristics of the switching fabric and output interfaces (the constraints that no more than one cell can be extracted from an input interface and delivered to an output interface at any slot are not enforced).

The basic idea of the asymptotic analysis is quite simple. In the considered situation, if for some time cells are not extracted from input queues, the queue lengths L_{ij} grow proportionally to the cell arrival rates λ_{ij} . In order to contrast the queue size growth, the scheduler must allow the transmission of cells from queues at rate λ_{ij} . Thus, the metrics associated with queues must be proportional to arrival rates. If this is not the case, some queue sizes decrease at the expense of other queue sizes, that increase by the same amount.

Assume that a scheduler exists that can select the N cells to transfer at any slot with a frequency (or probability) proportional to the metrics associated with the queue. If this is the case, recalling the definitions of Section III, the normalized growth rate of L_{ij} is $\gamma_{ij} - \mu_{ij}$.

Due to the normalization imposed on Γ and \mathbf{M} , we have

$$\sum_{i,j} (\gamma_{ij} - \mu_{ij}) = 0 \quad (4)$$

This means that, as we noted, with the conditions we are assuming, the decrease in a queue length implies an increase of

the length of one or more other queues, with the total increase balancing the total decrease.

The average unbalancement in queue size growth is defined as:

$$\chi = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N |\gamma_{ij} - \mu_{ij}| \quad (5)$$

The stability of the system is achieved only when $\chi = 0$.

Two IQ scheduling algorithms will be said to be *asymptotically equivalent* for a given traffic pattern if they produce the same value of χ under the same normalized traffic pattern Γ .

Now we show that not all the considered IQ scheduling algorithms are asymptotically equivalent for all traffic patterns; rather, classes of algorithms can be identified with asymptotically equivalent behaviors. These groupings will also be reflected in the results of the simulation study that follows in the final part of this paper.

Note that when considering CA metrics in our stationary asymptotic conditions, we can assume that the age of the cell at the head of the queue is proportional to the queue length; this makes CA metrics always equivalent to QL metrics.

A.1 Uniform traffic

The symmetry of the traffic is sufficient to conclude that in asymptotic conditions all weights w_{ij} are equal.

As a consequence, from (5) we obtain that

$$\chi_{QO} = \chi_{QL} = \chi_{CA} = \chi_{IL} = \chi_{MB} = \chi_{ML} = 0 \quad (6)$$

Hence, all metrics can guarantee stable system operations for all uniform admissible traffic patterns.

A.2 Unbalanced traffic

Also in this case, from Γ we can simply derive the weight matrices for the different types of metrics:

$$\mathbf{W}_{QO} = \mathbf{W}_{MB} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{W}_{IL} \propto \begin{pmatrix} 2 & 2 & 0 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 2 \end{pmatrix}$$

$$\mathbf{W}_{ML} \propto \Gamma$$

$$\mathbf{W}_{QL} \propto \mathbf{W}_{CA} \propto \Gamma$$

hence $\mathbf{W}_{IL} \propto \mathbf{W}_{QO} = \mathbf{W}_{MB}$, and $\mathbf{W}_{QL} \propto \mathbf{W}_{CA} \propto \mathbf{W}_{ML} \propto \Gamma$.

From the above matrices it is trivial to obtain \mathbf{M} ; then, from (5) we obtain that

$$\chi_{QO} = \chi_{IL} = \chi_{MB} = \frac{2\delta - 1}{N^2} \quad (7)$$

$$\chi_{ML} = \chi_{QL} = \chi_{CA} = 0 \quad (8)$$

This shows that the QL, CA, and ML metrics can guarantee stable system operations for this type of traffic, whereas the QO, IL, and MB metrics cannot, except for $\delta = 0.5$, i.e. for balanced traffic on the two diagonals.

B. Simulation results

Simulation results are presented in the case of cell switches with $N = 16$ input/output interfaces, assuming that all input/output line rates are equal, and loading the cell switch with two types of input traffic:

- uniform Bernoulli traffic
- unbalanced Bernoulli traffic, with $\delta = 0.6$

Destination queueing is assumed, and each input queue Q_{ij} can store up to 1000 cells; when a cell directed to output j arrives at input i , and queue Q_{ij} is full, the cell is lost. For simplicity, no buffer sharing among queues is allowed.

Simulation runs execute until the estimate of the average cell delay reaches with probability 0.95 a relative width of the confidence interval equal to 1% for the first type of traffic, and equal to 5% for the second. The estimation of the confidence interval width is obtained with the batch means approach [19].

The curves that will be presented in the graphs refer to two performance indices:

- **Cell delay.** This is the time spent by cells in the switch queues. For this performance index we shall consider the average value, the standard deviation, and the 99-th quantile. We normally observe the *relative* cell delay, obtained by dividing the delay produced by IQ architectures by the delay produced by an OQ switch loaded with the same traffic pattern.
- **Throughput.** This is the ratio between the total number of cells forwarded to output interfaces, and the total number of cells arrived at input interfaces. It is essentially a measure of the cell loss probability at input queues.

Curves will always be plotted versus the normalized switch load, 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).

In order to ease the reading of the graphs, the markers on simulation curves are related to the metrics used by the scheduling algorithms: QO is associated with +, ×, and *, QL with triangles, CA with diamonds, IL with squares, and MB and ML with circles.

C. Uniform Bernoulli traffic

The first type of switch workload that we investigate is uniform Bernoulli traffic. We must immediately note that no cell losses were observed for all admissible input loads, so that throughput curves are not significant. Hence, we concentrate on cell delays.

In Fig. 3 we report the curves of the average cell delay normalized to the OQ cell delay values for the different IQ switch architectures that we considered in this paper. The most evident observation that emerges from the graphs is that, up to a switch load equal to 0.9, all IQ switch architectures produce average cell delays that are within a factor 3 of those of an OQ switch. Only for few scheduling algorithms (2DRR, RC, and iSLIP, which use the QO metrics and aim at a maximal size match), and switch loads above 0.95, the relative average cell delay grows larger than 4. Moreover, for some scheduling algorithms (most notably MUCS-len) the difference in the average cell delay with respect to OQ always remains very small.

Considering next the curves of the standard deviation of the cell delay, shown in Fig. 4, again with the normalization to the

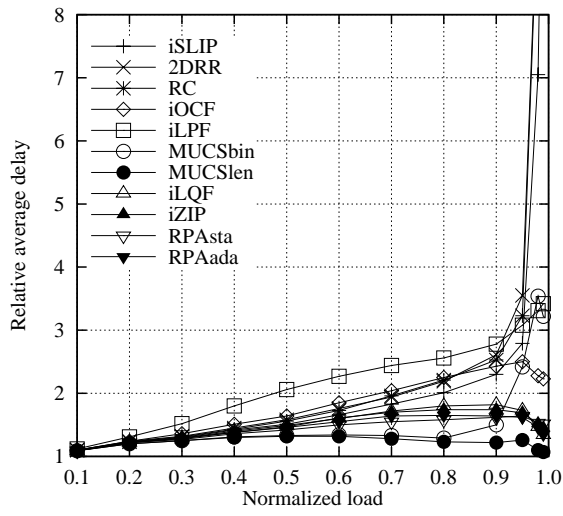


Fig. 3. Relative average cell delay under uniform traffic

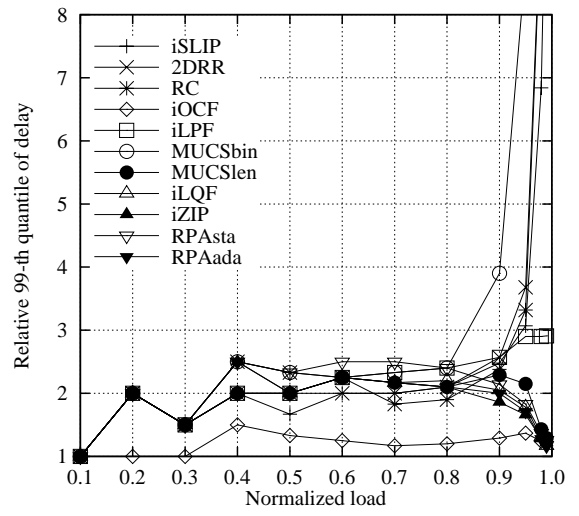


Fig. 5. Relative 99-th quantile of cell delay under uniform traffic

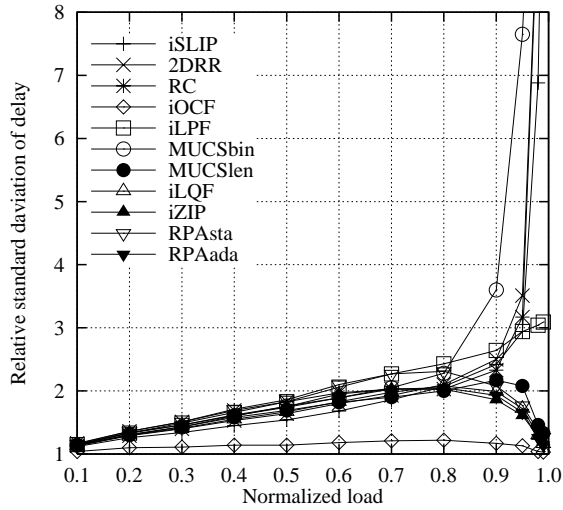


Fig. 4. Relative standard deviation of cell delay under uniform traffic

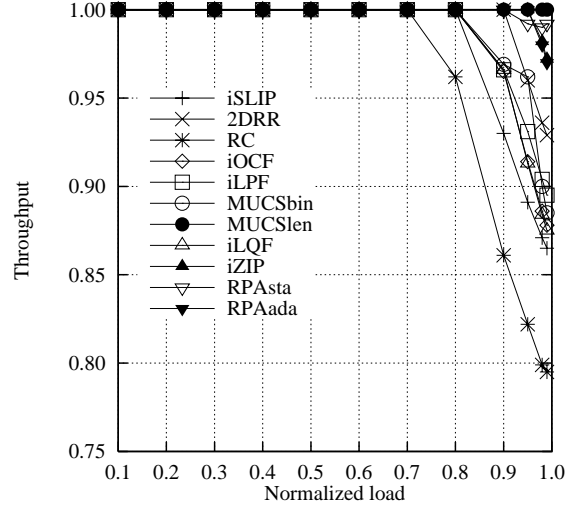


Fig. 6. Throughput under unbalanced traffic

OQ results, we see that, with the exception of MUCS-bin, also the standard deviation remains within a factor 3 of OQ up to a switch load equal to 0.9. In this case we also clearly identify a winner among IQ scheduling algorithms: iOCF is the only one that remains very close to OQ for all switch loads. This result was expected because of the particular metrics.

Very similar conclusions can be drawn from the observation of the curves of the 99-th quantile of the cell delay relative to OQ, shown in Fig. 5. Also for this performance index all algorithms, except MUCS-bin, remain within a factor 3 up to load 0.9, and iOCF is the algorithm providing results very close to OQ.

Overall, simulation results indicate that with uniform Bernoulli input traffic, IQ switch architectures can achieve an efficiency close to that of the traditional OQ switches. The performance differences among most of the algorithms appear to

be negligible, so that their implementation complexity could become the key factor in a choice about what algorithm to use in an IQ switch.

D. Unbalanced Bernoulli traffic

We now consider the case of unbalanced Bernoulli traffic, defined in Section III, assuming $\delta = 0.6$.

With this input traffic, cell losses are observed when saturation is approached. Throughput curves are reported in Fig. 6, where we can see that very high loss probabilities are incurred with the RC scheduling algorithm, as well as with iSLIP, iLQF, iOCF, MUCS-bin, 2DRR, and iLPF. This is in part as predicted by the simple asymptotic analysis of Section VI-A. However, the fact that all algorithms using the IS heuristic matching exhibit poor performance, independently of the considered metrics, indicates that with unbalanced traffic also the differences

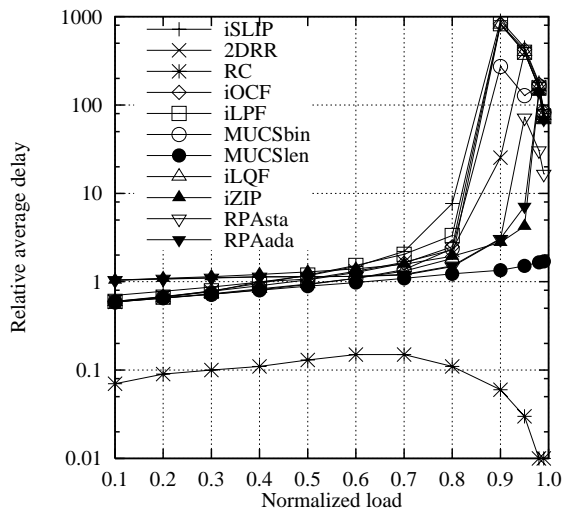


Fig. 7. Relative average cell delay for high-load cell flows

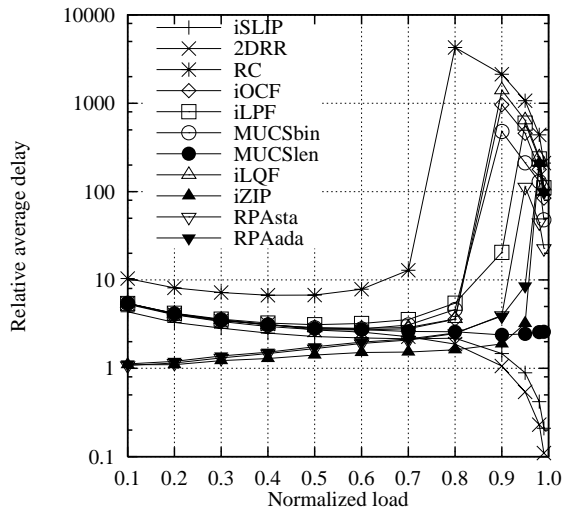


Fig. 8. Relative average cell delay for low-load cell flows

among the heuristics used to find the maximal match come into play. Note also that iOCF in this configuration produces high loss probabilities; this is in contrast with our asymptotic analysis, and is due to the poor performance of the IS heuristic. No losses are recorded only in the cases of OQ and MUCS-len. Small loss probabilities are produced by iZIP and RPA.

In Figs. 7 and 8 we report the curves of the average cell delay normalized to the OQ cell delay values for the different IQ switch architectures. The two figures refer respectively to the high load cell flows (Fig. 7) and to the low load cell flows (Fig. 8). In these figures for the first time we observe that average cell delays lower than those produced by OQ can be obtained. This is quite evident with RC for the high load cell flows, where the reduction is close to a factor 10. However, this reduction is paid with an increase of a similar factor, if not substantially higher, of the cell delay for low load cell flows. As a result,

a great unfairness exists with this algorithm. Other algorithms, like iSLIP, iLPF, iOCF, and iLQF, yield marginally lower average cell delays for the high load cell flows, but significantly higher average cell delays for the low load cell flows. Overall, the average cell delay of OQ remains lower than that of all IQ scheduling algorithms.

VII. CONCLUSIONS

The numerical results of our analysis of scheduling algorithms for IQ cell switches indicate that the choice of the metrics adopted by the scheduling algorithm is the major factor influencing performance with different traffic patterns. The best metrics are in general related to queue lengths; probably the metrics of MUCS-len emerges as a very interesting option, but its implementation complexity is high, since it requires floating point arithmetics.

The choice of the heuristics to approximate the maximum weight matching algorithm is mostly related to the hardware architecture of the switch, and has a significant impact on the time necessary to compute the cell transfer matrix. Since the computational complexities of the different heuristics remain very high, and not much less than those of the optimum match determination, simple heuristics should probably be preferred. It must be said that the choice of the heuristic matching has shown effects on performance in unbalanced traffic conditions.

Although not presented in this paper, results available in [13] show that the contention resolution strategy of the scheduling algorithm impacts fairness among input/output interfaces.

The amount of control information to be transferred for the coordination of the input interfaces is quite relevant, and in some cases this aspect may become the crucial factor in determining the system architecture.

It is worth noting that after this work was performed, an interesting and innovative proposal of a scheduling algorithm for IQ switches appeared in [20]. The proposal is based on the idea of exploiting the temporal correlation in the sequence of matchings: the matching at a given time slot is computed as a (random) variation with respect to the matching obtained at the previous slot, since at medium to high loads a large correlation between successive matchings is expected. The appealing feature of this proposal is in its simplicity (the author estimates a complexity linear with the number of input/output ports), probably paying some price in terms of performance. The quantification of the performance achievable with this new approach will be done in the prosecution of this work. Note that all the schemes considered in this paper compute the matching in a given time slot ignoring the matching computed in the previous time slot. An indirect correlation between matchings is provided by the metrics upon which the matching is computed, in a fashion that is strongly dependent on the type of metrics that is used: the QO metrics provides the least correlation, while CA or QL provide larger correlations. The exploitation of the temporal correlation between successive matchings can lead to important reductions in the complexity of scheduling algorithms.

The results presented in this paper show that IQ and OQ switch architectures can achieve similar performance figures. In the comparison, IQ architectures pay the price of a larger computational complexity (for the scheduling algorithm) and of

the transmission of more control information within the switch, while OQ architectures must face the hardware complexity due to the large internal speedup. A good compromise between the two approaches can be achieved with mixed input/output architectures, in which a limited speedup factor (typically equal to 2 [21]) permits a significant reduction in the complexity of the scheduling algorithm [13].

Cell switch architectures can be used as core switching engines within switches operating on variable-size packets, such as IP routers, if packets are internally segmented into cells. In this case it is easy to modify the scheduling algorithms in order to guarantee the contiguity of cells belonging to the same packet; with these modified scheduling algorithms, results presented in [13] show that performance advantages can be obtained by IQ architectures with respect to OQ architectures.

Finally, although in this paper we restricted our attention to switching matrix schedulers, an important issue in the design of high-performance next-generation IP routers is the design of flow-level schedulers, which serve cell flows according to QoS requirements. A large number of papers can be found in the literature regarding flow-level schedulers in the case of OQ switch architectures. Research on flow-level schedulers for IQ switches has not yet received much attention (see, for example, [15], [16]), but it is not difficult to foresee that the interest in the field will increase significantly in the near future.

REFERENCES

- [1] Karol M., Hluchyj M., Morgan S., "Input versus Output Queuing on a Space Division Switch", *IEEE Transaction on Communications*, vol.35, n.12, Dec.1987, p.1347-1356
- [2] Awdeh R.Y., Mouftah H.T., "Survey of ATM switch architectures", *Computer Networks & ISDN Systems*, vol. 27, 1995, pp. 1567-1613
- [3] McKeown N., Mekkittikul A., "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches", *INFOCOM 98*
- [4] McKeown N., Izzard M., Mekkittikul A., Ellesick B., Horowitz M., "The Tiny Tera: A Packet Switch Core", *IEEE Micro Magazine*, vol.17, Feb.1997, p.27-40
- [5] McKeown N., "Scheduling Algorithms for Input-Queued Cell Switches", *Ph. D. Thesis*, University of California at Berkeley, 1995
- [6] McKeown N., Anderson T.E., "A Quantitative Comparison of Scheduling Algorithms for Input-Queued Switches", <http://tiny-tera.stanford.edu/~nickm/papers.html>
- [7] McKeown N., Mekkittikul A., "A starvation free algorithm for achieving 100% throughput in an input queued switch", *ICCCN 96*, 1996
- [8] LaMaire R.O., Serpanos D.N., "Two dimensional round-robin schedulers for packet switches with multiple input queues", *IEEE/ACM Transaction on Networking*, vol.2, n.5, Oct.1994
- [9] Chen H., Lambert J., Pitsilledes A., "RC-BB switch. A high performance switching network for B-ISDN", *IEEE GLOBECOM 95*, 1995
- [10] Duan H., Lockwood J.W., Kang S.M., Will J.D., "A high performance OC12/OC48 queue design prototype for input buffered ATM switches", *IEEE INFOCOM 97*, 1997
- [11] Ajmone Marsan M., Bianco A., Leonardi E., "RPA: a simple efficient and flexible policy for input buffered ATM switches", *IEEE Communication Letters*, vol.1, n.3, p.83-86, May 1997
- [12] Ajmone Marsan M., Bianco A., Leonardi E., Milia L., "Quasi optimal algorithms for input buffered ATM switches", *ISCC 98*, 1998
- [13] Giaccone P., *Tecniche di accodamento e trasferimento in architetture di commutazione a larga banda*, Laurea Thesis, Politecnico di Torino, May 1998, in Italian
- [14] Golestani S., "A Self-Clocked Fair Queueing Scheme for Broadband Applications", *IEEE INFOCOM 94*, Toronto, Canada, June 1994
- [15] D.Stiliadis, A.Varma, "Providing bandwidth guarantees in an input-buffered crossbar switch", *IEEE INFOCOM 95*, 1995.
- [16] Stephens D.C., Zhang H., "Implementing distributed packet fair queueing in a scalable switch architecture", *IEEE INFOCOM 98*, 1998
- [17] Tarjan R.E., *Data structures and network algorithms*, Society for Industrial and Applied Mathematics, Pennsylvania, Nov.1983

- [18] Hopcroft J.E., Karp R.M., "An $n^{2.5}$ algorithm for maximum matching in bipartite graphs", *Society for Industrial and Applied Mathematics J.Comput.*, vol.2, p.225-231, 1973
- [19] Pawlikowski K., "Steady-state simulation of queueing processes: a survey of problems and solutions", *ACM Computing Surveys*, vol.22, n.2, June 1990, p.123-163
- [20] Tassiulas L., "Linear complexity algorithms for maximum throughput in radio networks and input queued switches", *IEEE INFOCOM 98*, 1998
- [21] Stoica I., Zhang H., "Exact emulation of an output queueing switch by a combined input output queueing switch", *6th International Workshop on Quality of Service (IWQoS'98)*, Napa, CA, May 1998, p. 218-224