

## SCHEDULING IN INPUT-QUEUED CELL-BASED PACKET SWITCHES\*

M. Ajmone Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri  
 Dipartimento di Elettronica, Politecnico di Torino, Italy  
 e-mail: {ajmone,bianco,giaccone,leonardi,neri}@polito.it

**Abstract**

Input queuing switch architectures must be controlled by a scheduling algorithm, which solves contentions in the transfer of data units from inputs to 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. We show that, in the case of packet switches, input queuing architectures can provide advantages over output queuing architectures.

**1 Input vs. Output Queueing**

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 B-ISDN [1]. Those research activities brought on the market efficient chip-sets currently used as building blocks for high-performance Internet routers. As a consequence, for many advanced Internet routers, the switching fabric internally operates on 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.

Most of the old designs 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, assumed an output queuing architecture. Input queuing switch architectures [2] have recently received increasing attention, and they are currently considered by many designers as the best solution when the line speed is pushed to technological limits.

The main disadvantage of output queuing (OQ) is that both the switching fabric and the 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 interfaces is large or the line rate is high, this makes OQ impractical. With respect to input queuing (IQ) schemes, OQ has the advantage that delays through the switch can be more easily controlled, and the implementation of (concentrated) fair queuing algorithms at the output is relatively easy and well understood [3].

IQ schemes permit all the components of the switch (input interfaces, switching fabric, output interfaces) to operate

\*This work was supported by a research contract between CSELT and Politecnico di Torino and by the Italian Ministry for University and Research

at a speed which is compatible with the data rate of input and output lines. One of the reasons why IQ was almost ruled out by the research efforts of the ATM community is the performance reduction due to head-of-the-line blocking in the case of a single queue per input interface [4]. Recently, Virtual Output Queuing (VOQ) or Destination Queuing schemes, that largely reduce this problem, were proposed [5]: in each interface card, input buffers are organized into a set of isolated queues, each queue storing cells directed toward a specific output interface.

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 algorithm in order to avoid contention<sup>1</sup>. Several scheduling algorithms for IQ cell switches were proposed and compared in the literature [2], [7], [8], [9], [10], [11]. They provide performance close to OQ architectures. We consider some of these proposals, and develop novel variations to deal with variable-size packets; more precisely, 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 reassembly 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, will be compared by simulation.

**2 Logical Architecture****2.1 Cell switches****2.1.1 Input queuing cell switches**

Fig. 1 shows the logical structure for an input queuing 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 generic term *cells* to identify the internal fixed-size data units. We as-

<sup>1</sup>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 [6]. *Switching matrix schedulers* decide which input interface is enabled to transmit in an input queuing switch; 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.

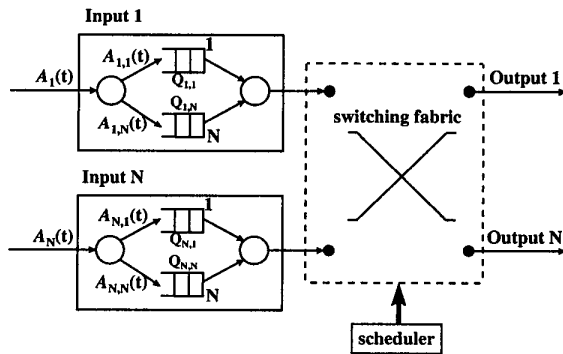


Figure 1: Logical structure of an input queuing cell switch

sume that the switch takes switching decisions at equally-spaced time instants. The distance between two switching decisions is called *slot*, and the slot is the granularity in allocating switch resources.

The switch can have in general  $m$  inputs and  $n$  outputs, but we focus on the more realistic case  $m = n = N$  (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.

Cells arrive at input  $i$ ,  $1 \leq i \leq N$ , according to a discrete-time random process. 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. 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.

The average rate of the arrival process  $A_{ij}(t)$  at input  $i$  for output  $j$  is denoted by  $\lambda_{ij}$ . For later use, we define the cell arrival rate matrix  $\Lambda = [\lambda_{ij}]$ , 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; 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 into input interfaces, we have a speed-up factor equal to 1. The scheduling algorithm decides which cells can be transferred from the inputs to the outputs of the switch in every slot.

### 2.1.2 Output queuing cell switches

The output queuing 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 towards each output must be

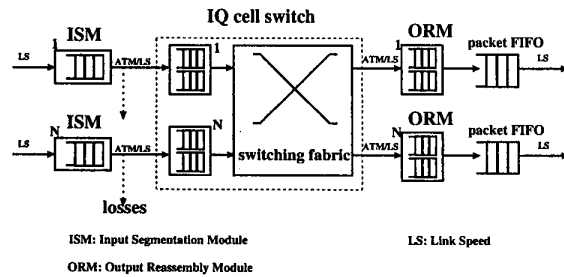


Figure 2: Logical architecture for an IQ packet switch

equal to the sum of the bandwidths available on all input lines. We say that the switching fabric must have a speed-up factor equal to  $N$ .

We can assume that the slot time is subdivided into  $N$  minislots, and that each input can use a different minislot to transfer a cell to a given output. The allocation of minislots 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 queuing 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 (in IQ, we assume that  $L$  is the capacity of each virtual queue).

## 2.2 Packet switches (or routers)

We consider **packet** switch architectures based on a cell switching matrix, considering as reference application the development of a high-performance IP router around an ATM cell-switch. Each router port has line interfaces where any data-link and physical layer protocol can be used to receive and transmit IP datagrams. Input IP datagrams are segmented into cells, that will be transferred to output ports by a high-performing switching fabric. Once cells are delivered to an output port, they are reassembled into IP datagrams, which are transmitted on the output line.

### 2.2.1 IQ packet switches

The logical architecture is shown in Fig. 2. At each input an Input Segmentation Module (ISM) segments the incoming packet into cells. Since it 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 ATM/LS) equal to the line speed (LS) 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 it has the advantage of ease of implementation, and of avoiding the transmission of packet segments 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  $ATM/LS$ . 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 sent to an output packet queue, called *packet FIFO*, from which packets are sequentially transmitted onto the output line.

### 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 as we shall see, by enforcing a constraint on the scheduling algorithm. Indeed, the cells belonging to the same packet are stored contiguously in the input queue of the internal cell-switch. As shown 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 started towards the corresponding output interface, no cells belonging to other packets can be transferred to that output. Cells belonging to the same packet are thereby kept contiguous also in the output queue, and the ORM modules are not necessary any longer (or at most one per output is used). We call this class of scheduling algorithms *packet-mode scheduling*. Note that enforcing packet contiguity is not possible in OQ switches, where cell interleaving in output queues cannot be avoided.

If packet-mode scheduling is adopted in IQ packet switches, the logical architecture can be simplified by removing both the ORM module and the output packet FIFO from the scheme of Fig. 2. 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, in principle, 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.

### 2.2.3 OQ packet switches

Fig. 3 shows the logical architecture of an OQ packet switch. Packets arrive at input ports where they are seg-

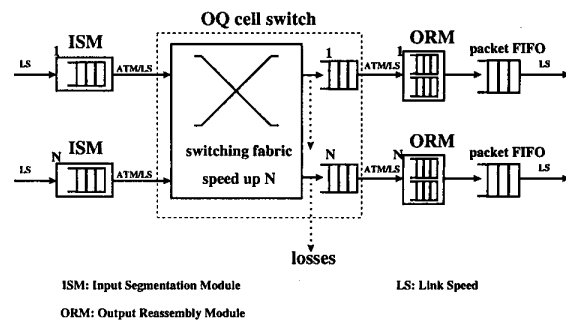


Figure 3: Logical architecture for an OQ packet switch

mented by ISM modules, similarly to what happens in IQ routers. The cells obtained with the segmentation are sent to the cell-switch at speed  $ATM/LS$ , and are immediately transferred to the output queues of the cell-switch, thanks to a speed-up  $N$  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  $ATM/LS$  to an ORM module for reassembly. Once a packet is completed, it is queued to 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 to be unable 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 remain in the output queues of the cell-switch, unnecessarily using system resources.

## 3 Scheduling algorithms in IQ switches

### 3.1 Problem definition

The scheduling algorithm selects 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. 4)

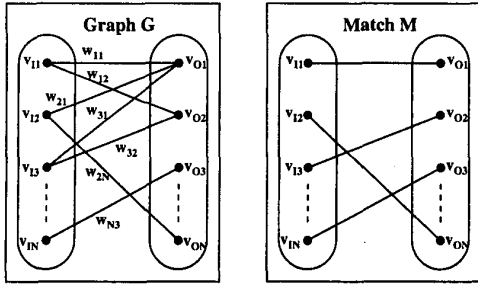


Figure 4: Bipartite graph description of the scheduling problem

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 metrics that will be denoted by  $w_{nm}$ . The adopted metrics 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.

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.

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. The complexity is  $O(N^3 \log N)$  for the maximum weight matching [12, Chapt. 8] algorithm, that 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 the simpler and less efficient maximum size matching algorithm [12], [13], [14].

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 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 recently appeared in the technical literature. In this paper we consider four such proposals, namely iSLIP [2], [10], iOCF [2], [5], MUCS [8] in its weighted version, and RPA [7]. 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.

For the sake of brevity we do not provide a description of these algorithms. The reader is referred to the original works for detailed descriptions. In this section we recall the general taxonomy for scheduling algorithms proposed in [11], and classify accordingly the considered proposals.

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:

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

Any IQ scheduling algorithm can be viewed as operating according to following three phases. The first phase is preliminary to the other two, that are instead interlaced.

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 metrics depending on the state of the corresponding queue.

2. *Heuristic matching.* Computation of the cell transfer matrix  $\mathbf{T}(k) = [t_{ij}(k)]$ . This phase must try to maximize  $\sum_{i=1}^N \sum_{j=1}^N t_{ij}(k)w_{ij}(k)$ , with the constraints  $\sum_{i=1}^N t_{ij}(k) \leq 1$  and  $\sum_{j=1}^N t_{ij}(k) \leq 1$ . 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 edges of  $G$  can be added to  $M$  without violating the above constraints, 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 metrics, 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.

As we shall see in Section 4.2, 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.

As regards metrics, 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 metrics adopted by iSLIP. 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.
- **QL** (*Queue length*). The metrics in this case is the number of cells in the queue:  $w_{ij} = L_{ij}$ . This is the metrics adopted by RPA.
- **CA** (*Cell age*). The metrics in this case is the time already spent in the queue by the cell at the queue head. This is the metrics adopted by iOCF.
- **ML** (*MUCS Length*). MUCS uses a metrics that is derived from queue lengths as:

$$w_{ij} = \frac{L_{ij}}{\sum_{k=1}^N L_{ik}} + \frac{L_{ij}}{\sum_{k=1}^N L_{kj}} \quad (2)$$

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  $j$ ). These 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 heuristics is adopted by both 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 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.
- **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

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

TABLE I  
CHARACTERIZATION OF THE CONSIDERED IQ SCHEDULING ALGORITHMS

whose original reservations were overwritten. This is the heuristics adopted by RPA.

In Table I, we summarize the characteristics of the considered IQ scheduling algorithms.

The computational complexity of an IQ scheduling algorithm is a crucial parameter, since the algorithm must be run at every cell time. We do not tackle this issue here (the interested reader is referred to [11] for details) but it is worth noting that the complexity obtained with the different heuristics is still significant, being not much less than those of the optimum match determination.

### 3.3 Packet-mode scheduling algorithms

We claimed in the last part of Section 2.2 that the architectural complexity of IQ packet switches 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 goal, 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 enabled also for the following  $k - 1$  slots.

This is equivalent to have an infinite weight on the corresponding connection (i.e., edge of graph  $G$ ). 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 store a boolean variable at each input to flag over-prioritized connections. Furthermore, maximal size matching algorithms must operate on ternary weights, i.e.,  $w_{ij} = 0$  if  $L_{ij} = 0$ ,  $w_{ij} = 1$  if  $L_{ij} > 0$ , and  $w_{ij} = \infty$  if a packet is being transferred.

## 4 Simulation results

To evaluate the performance of IQ and OQ switching architectures, we conducted a 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. Only unicast traffic flows are considered.

In IQ switches, each input queue  $Q_{ij}$  can store up to  $L = 1000$  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

In our simulation model we represent the arrivals of IP datagrams at the input of the packet-switch as arrivals of cell bursts at the input of the internal cell-switch. These cell bursts are assumed to be originated by the segmentation of a packet.

The cell arrival processes at input  $i$  is generated 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 [15], 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 generated. The probability that the OFF state is equal to  $j$  slots is  $P(j) = p(1-p)^j$ , with  $j \geq 0$ . The average idle period duration 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 normalized cell arrival rate matrices:

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

*Hot-spot traffic.* All input interfaces are equally loaded, but one of the outputs receives twice as much traffic as all others; this produces a normalized cell arrival rate matrix of the following type (using  $N = 4$  as an example):

$$\Gamma = \frac{1}{N^2 + N} \begin{pmatrix} 2 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \end{pmatrix}$$

Results are presented with graphs where the performance indices defined next are plotted versus the switch load. The load is defined as the ratio between the input traffic load and the total capacity of input/output lines; load values are comprised between 0 and 1.

- **Cell delay.** This is the time spent by cells in the cell-switch queues. For this metrics 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 exit of the first cell of the packet from the ISM module until the exit of the last cell from the final packet FIFO. Constant components to the delay 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  $k - 1$  slots, due to wait in the packet FIFO. We shall consider the average value and the standard deviation of packet delay.

- **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 or output queues.

- **Maximum number of reassembly machines.** This is the maximum number of reassembly machines simultaneously active, i.e. assembling packets, in all the  $N$  ORM modules. In the case of IQ architectures, the reassembly machines always act on packets that will be completed. In the OQ case, instead, a reassembly machine can start to assemble a packet that will suffer cell losses; we assume that the reassembly machine immediately discards the entire packet when the first loss occurs. As a consequence, the maximum number of simultaneously active reassembly machines can never be larger than  $N^2$ . For packet-mode scheduling algorithms in IQ switches, at most one reassembly machine can be active at each output, hence the largest value taken by this performance index is  $N$ .

Since the aim of the performance evaluation is a comparison of IQ and OQ architectures, we usually consider *relative* performance indices, i.e., we divide the absolute value taken by a performance index in the case of IQ 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 reaches 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. 5 shows, for the four considered scheduling algorithms *not operating in packet mode*, curves of the normalized average packet delay, i.e., the average packet delay of IQ switches divided by the average packet delay of an OQ switch loaded with the same traffic. Note that delay values are always larger than 1 (the reference value for OQ), i.e., IQ switches always yield longer delays, but differences are limited within a factor 2.5.

For loads less than 0.90, 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

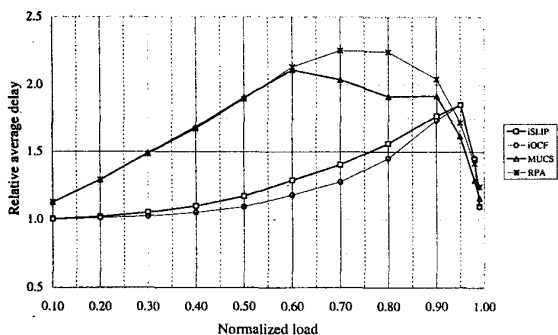


Figure 5: Relative average packet delay under uniform traffic

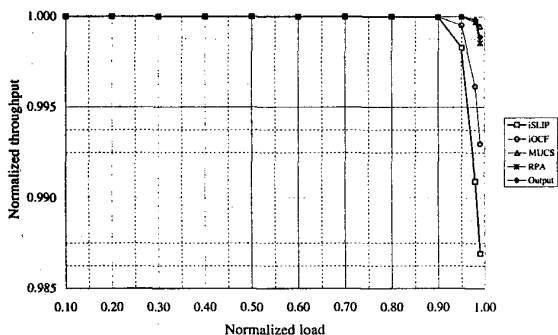


Figure 6: Packet throughput under uniform traffic

(at other inputs) that store small packets directed to the same output suffer a temporary starvation. iOCF, instead, provides the lowest delays.

For loads above 0.90, losses occur, as shown in Fig. 6, where the packet throughput is plotted. Note that MUCS and RPA now provide the highest throughput (i.e., less losses); this is again due to the used metrics. iSLIP exhibits the largest losses: the QO metrics tends in general to provide low delays but large losses. iOCF probably achieves in this scenario the best compromise between delay and losses. The OQ architecture produces fewer losses than all IQ architectures.

Fig. 7 shows curves of the relative standard deviation of

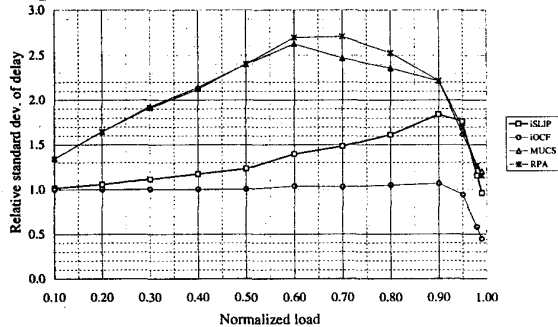


Figure 7: Relative standard deviation of packet delays under uniform traffic

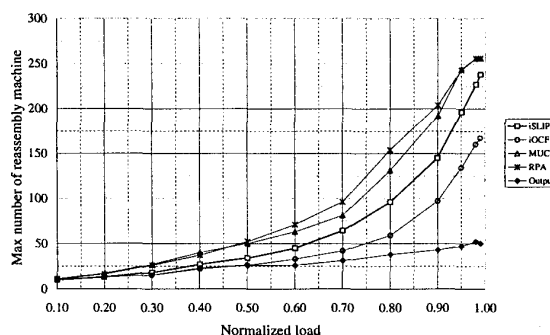


Figure 8: Maximum number of simultaneously active reassembly machines under uniform traffic

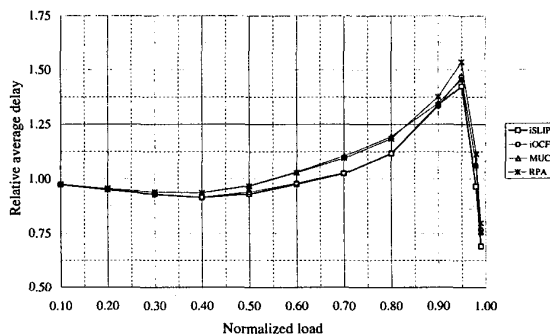


Figure 9: Relative average packet delay in packet mode under uniform traffic

packet delays. Note the very good behavior of iOCF, which provides delay variations very close to the OQ architecture.

Fig. 8 shows curves of the maximum observed number of simultaneously active reassembly machines. OQ switches perform best, with a maximum below 50 for all admissible loads. This is due to the fact that OQ architectures only interleave two packets directed to the same output when the first cell of one packet arrives (at another input) before the last cell of the other packet. This is not true for IQ architectures, in which cells may be delayed at the input.

The metrics used by MUCS and RPA leads to the largest values for the maximum number of reassembly machines, reaching values around 256, which is the total number of reassembly machines available in the ORM modules of our 16 × 16 switch.

#### 4.2.2 Uniform traffic – Packet-mode scheduling

Performance results are significantly different from what we observed above if the packet-mode scheduling proposed in this paper is considered. 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, as discussed in Section 2.2.2.

Curves of the relative average delays are shown in Fig. 9. Differences between the four algorithms are limited, but we observe advantages over OQ switches for loads up to around

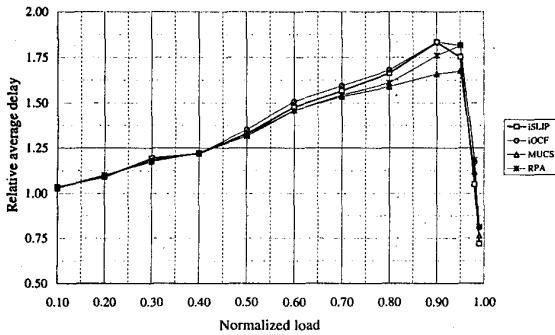


Figure 10: Relative average cell delay in packet mode under uniform traffic

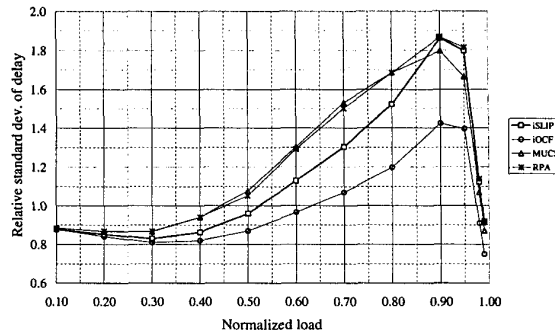


Figure 12: Relative standard deviation of packet delay in packet mode under uniform traffic

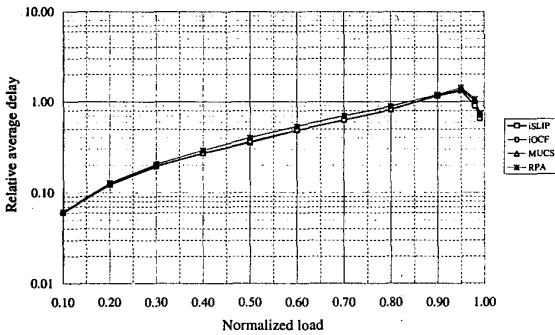


Figure 11: Relative average packet delay in packet mode under uniform traffic for the optimized IQ architecture

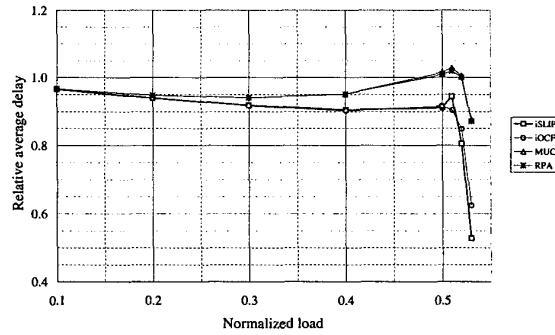


Figure 13: Relative average packet delay in packet mode under hot-spot traffic

0.6. IQ exhibits the largest delay advantage over OQ at loads around 0.4. Note that cell-level delays in this same scenario (shown in Fig. 10) are larger for all IQ architectures with respect to OQ at all loads where losses do not occur.

The reductions in packet delays are interesting, specially if we consider the negligible additional cost of implementing packet-mode schedulers. 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 delays over OQ becomes much larger, as shown in Fig. 11. At low loads, IQ architectures gain more than one order of magnitude in average packet delays over OQ (note the logarithmic vertical scale).

It can be observed in Fig. 9 that the metrics used by MUCS and RPA tend to provide longer delays than iOCF and iSLIP. Throughput and loss results, not reported here, show instead a behavior very similar to that of Fig. 6: iSLIP and iOCF suffer larger losses than MUCS and RPA. Again, the OQ architecture produces less losses than all IQ setups: this is due to a beneficial buffer sharing effect at each output, as previously mentioned.

The graph in Fig. 12 shows curves of the relative standard deviation of packet delays for packet-mode scheduling algorithms. Smaller variations than OQ are observed for low and

moderate traffic. Similarly to Fig. 7, iOCF is very effective in achieving small delay variations.

#### 4.2.3 Hot-spot traffic – Packet-mode scheduling

This scenario concentrates traffic towards one output. The maximum admissible switch load can be analytically computed: it is equal to  $17/32 \approx 0.53$  when  $N = 16$ .

Fig. 13 shows curves of the relative average packet delay in the hot-spot traffic scenario. The four considered algorithms, modified to operate in packet mode, exhibit smaller delays than the OQ switch for all loads where losses do not occur (throughput results, not reported here, show that losses occur when the load is equal to 0.5 or larger). Average delays are smaller than those of OQ both for the heavily loaded output and for the  $N - 1$  lightly loaded outputs.

Fig. 14 shows curves of the relative standard deviation of packet delays in the hot-spot traffic scenario. Delay variations are similar to those produced by OQ, with marginal reductions (ratios are around 0.8) at loads not larger than 0.3. At larger loads, while iOCF always provides smaller variations than OQ, MUCS and RPA, and iSLIP at a lesser extent, tend to exhibit larger variations (ratios are however limited to less than 1.4).

These results are very encouraging, and tell us 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 ar-

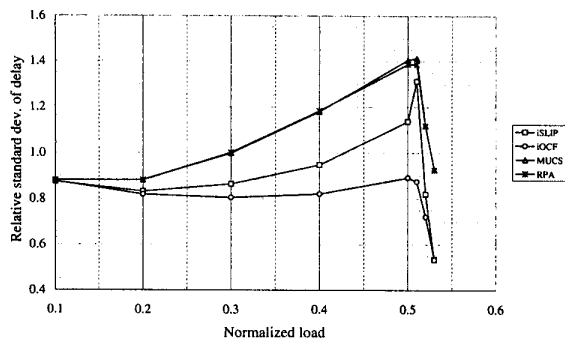


Figure 14: Relative standard deviation of packet delay in packet mode under hot-spot traffic

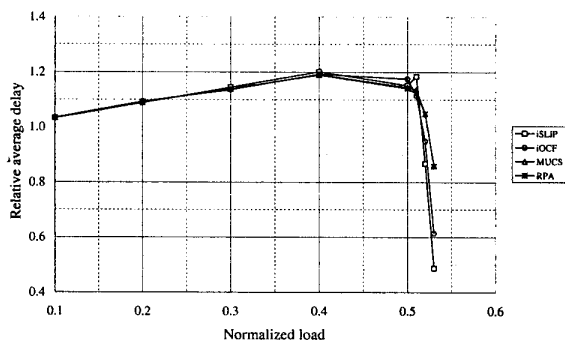


Figure 15: Relative average packet delay in cell mode under hot-spot traffic

chitectures. Similarly, the uniform traffic scenario is a worst case scenario when measuring the merits of packet-mode vs cell-mode scheduling algorithm, as can be seen comparing Fig. 13 with Fig. 15.

## 5 Conclusions

The paper focused on architectures and scheduling algorithms for input queuing packet switches, comparing them with output queuing switches in the case of variable-size data units. Similar performance results were observed for IQ, and OQ, with a tradeoff between algorithmic complexity of the scheduling algorithm on one side, and hardware complexity due to the internal speed-up on the other side.

We considered four previously proposed scheduling algorithms for the transfer of fixed-size data units in input queuing architectures. These algorithms exhibited performances very close to output queuing switches.

We proposed novel modifications of 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 queuing architectures. This is an important result, which can impact the design of next-generation packet switches.

Note that packet-mode scheduling can in principle avoid the segmentation of variable-size packets into fixed-size

cells: the activation of the scheduling algorithm must occur at instants that are spaced in time by fixed time intervals (called slots), and each packet "freezes" an input/output connection for a sufficient number of slots.

Obviously, a drawback of packet-mode scheduling is that long packets keep busy an input/output pair for a long time, thereby possibly delaying short, time-sensitive packets. This behavior is typical of packet switches and can be avoided by permitting either preemption and retransmission of lower priority packets, or interruption and restart of the transfer of lower priority packets using one output buffer per packet priority.

## References

- [1] Awdeh Ra'ed Y., Mouftah H.T., "Survey of ATM Switch Architectures", *Computer Networks & ISDN Systems*, vol. 27, n. 12, May 1995, p. 1567-1613
- [2] McKeown N., "Scheduling Algorithms for Input-Queued Cell Switches", *Ph. D. Thesis*, University of California at Berkeley, 1995
- [3] Golestani S., "A Self-Clocked Fair Queueing Scheme for Broadband Applications", *IEEE INFOCOM'94*, Toronto, Canada, June 1994,
- [4] 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
- [5] McKeown N., Mekkittikul A., "A Starvation Free Algorithm for Achieving 100% Throughput in an Input Queued Switch", *ICCCN'96*, Washington, DC, Oct. 1996
- [6] Stiliadis D., Varma A., "Providing Bandwidth Guarantees in an Input Buffered Crossbar Switch", *IEEE INFOCOM'95*, Boston, MA, April 1995
- [7] Ajmone Marsan M., Bianco A., Leonardi E., "RPA: A Simple Efficient and Flexible Policy for Input Buffered ATM Switches", *IEEE Communications Letters*, vol. 1, n. 3, May 1997, p. 83-86
- [8] 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*, Kobe, Japan, April 1997
- [9] McKeown N., Mekkittikul A., "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches", *IEEE INFOCOM'98*, San Francisco, CA, March 1998
- [10] McKeown N., Anderson T.E., "A Quantitative Comparison of Scheduling Algorithms for Input-Queued Switches", <http://tiny-tera.stanford.edu/~nickm/papers.html>
- [11] Ajmone Marsan M., Bianco A., Filippi E., Giaccone P., Leonardi E., Neri F., "On the Behavior of Input Queuing Switch Architectures", *European Transactions on Telecommunications*, vol. 10, n. 2, March/April 1999, p. 111-124
- [12] Tarjan R.E., *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Pennsylvania, Nov. 1983
- [13] Even S., Kariv O., "An  $O(n^{2.5})$  Algorithm for Maximum Matching in General Graphs", *16<sup>th</sup> Symposium on Foundations of Computer Science*, 1975, p. 100-112
- [14] 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, n. 4, Dec. 1973, p. 225-231
- [15] Atkinson R., "RFC 1626: Default IP MTU for use over ATM AAL5", May 1994