

# On the Throughput of Input-Queued Cell-Based Switches with Multicast Traffic

M. Ajmone Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri

**Abstract**—In this paper we discuss the throughput achievable in input-queued cell-based switches loaded with multicast traffic. The switch architecture is assumed to comprise a synchronous broadcast switching fabric, where fixed-size data units, called cells, can be transferred in one slot from one input to any set of outputs. The switch scheduler must select the time slots for transfers of non-conflicting cells, i.e., cells neither coming from the same input nor directed to the same output. Contrary to the case of unicast traffic, for which input-queued switches were proved to yield the same throughput as output-queued switches, we show by simulation experiments and analytical modeling that throughput limitations exist in input-queued switches loaded with multicast traffic.

**Keywords**—Input queued architectures, IQ switch, multicast traffic.

## I. INTRODUCTION

The trend in networking today is to integrate traditional communication services (telephony, radio and TV broadcasting, etc.) into IP networks, thus making IP networks the candidate of choice for the implementation of an information transfer infrastructure capable of satisfying all communication needs of end users. This trend is producing a number of effects, the most evident and most frequently quoted of which are the explosive increase of IP traffic, the need for techniques to guarantee the quality of the services offered to end-users in IP networks, and the need for terabit channels and routers. Another effect of the integration of traditional communication services, and in particular of radio and TV broadcasting, into IP networks is the need for efficient multicast support.

While the multicast problem already received considerable attention at the algorithmic (or network) level, e.g., with the development of multicast protocols for IP networks [1], [2], [3], the techniques to support multicast traffic within IP routers have not received much attention yet; this is instead a crucial problem, especially when considering IP routers with input-queued (IQ) architectures. In this paper we consider techniques to support multicast traffic in IQ *cell-based* IP routers, and we show that the presence of multicast traffic can lead to severe performance degradations within such architectures.

The reason why we consider IQ IP routers is that they have received a considerable attention in the recent literature, seeming the most promising architecture for the implementation of extremely fast switching engines [4], thanks to the fact that the aggregate bandwidth required in their switching fabric is not larger than the sum of the data rates of input links. Instead, output-queued (OQ) router architectures with  $N$  input/output ports require switching fabrics and output memories whose bandwidth must be up to  $N$  times the sum of the data rates of input links. We normally say that IQ architectures can operate at speed-up

equal to 1, while OQ architectures require a speed-up equal to  $N$ .

The consideration of a cell-based paradigm within IP routers comes from the synchronous nature of the majority of such machines, that operate by transferring fixed-size data units from inputs to outputs, and from the availability of chip sets developed for ATM switches (this is why we use the term cell, even if our results are not limited to the case of the 53-bytes ATM cells; the time necessary for the transfer of a cell will be called a 'time slot'). Examples of commercial or experimental IP routers adopting a cell-based IQ architecture are found in [5], [6], [7], [8], [9], [10], [11].

In an IQ router with speed-up 1, the main issue concerning multicast support is the capability of transferring a multicast cell from an input queue to different output ports. In order to solve this problem, several architectures have been proposed [12], which are based either on internal copy networks or recirculating lines, so that multicast cells are replicated at inputs and treated like unicast cells, or on redundant switching paths, that allow the parallel transfer of multicast cells to their destinations. From an architectural point of view, the availability within the router of a switching fabric with *intrinsic multicast capabilities* is extremely important to reduce the cost of multicast traffic support. For example, switching fabrics implemented with a bus or a crossbar offer the possibility of transferring a cell from one input port to many output ports at no extra cost; the cell injected into the switching fabric by the input port can reach any number of output ports within one time slot. In this paper we refer to IQ routers whose internal fabrics have such intrinsic multicast capabilities.

The problem of scheduling multicast traffic was defined and modelled in [13], using a theory based on stochastic ordering and majorization. In that work, the optimal scheduling discipline is fully characterized for a switch with two and three input ports, based on a queueing structure with only one FIFO queue for each input. No results about the maximum achievable throughput are provided. Several theoretical studies [14], [15], [16], [17] investigate the maximum throughput achievable when arrivals are generated according to a Poisson process, and random services of input queues are assumed. Moreover, cells at the head of input queues are assumed to be served independently across the different inputs, as well as from slot to slot. These models show that the maximum normalized throughput for uniform traffic is always less than one, and that it depends on the multicast traffic distribution.

In a previous paper [18] we discussed some basic issues concerning the transfer of multicast traffic within IQ switching architectures. We described architectural alternatives for switches with multicast support, and we discussed the constraints on input traffic to guarantee finite queue lengths in the switch. Ex-

The authors are with the Dipartimento di Elettronica, Politecnico di Torino, Torino, Italy. Email: {ajmone,bianco,giaccone,leonardi,neri}@polito.it. This work was supported by the Italian Ministry for University and Research through the MQoS Project.

amples of traffic patterns for which IQ switches exhibit intrinsic performance losses with respect to OQ switch architectures were provided, and the optimal multicast scheduling discipline was defined.

In this paper we consider packet switches/routers where arriving unicast and multicast fixed- or variable-size packets are fragmented into fixed-size cells and stored into buffers at input ports. Cells are transferred from input to output ports through a switching fabric with multicast capabilities, following a scheduling discipline that must avoid contentions (no more than one cell can be extracted from an input port in one time slot, and no more than one cell can be delivered to an output port in one time slot). We define a simple greedy multicast scheduling algorithm with low complexity and good performance, and apply it in our simulation study of different switch configurations, revealing the existence of traffic patterns leading to high throughput, but also of traffic patterns producing unexpectedly low throughput values. Working with the latter traffic patterns, we then analytically prove that with some switch configurations and traffic patterns, a wide class of scheduling algorithms leads to poor performance.

This result is quite relevant, since it shows that, with multicast traffic, IQ architectures are inferior to OQ architectures, contrary to the case of unicast traffic, for which IQ and OQ switches were proved to be equivalent [19].

## II. MULTICAST TRAFFIC IN IQ SWITCHES

We introduce in this section some basic definitions on multicast switching, and define the switch architecture considered in the paper. Unless otherwise specified, we refer to switches with  $N$  input and  $N$  output ports, in which all input and output lines run at the same data rate. The switching fabric has intrinsic broadcasting capabilities, i.e., the cost of transferring in a time slot a cell from an input to either one or several outputs, is the same.

Any multicast cell is characterized by its *fanout set*, i.e., by the set of switch output ports (destinations) to which the cell is directed. The *fanout* [16], [17] is defined as the number of different destinations of a multicast cell, i.e., the cardinality of the fanout set. We say that a cell has *fanout destination*  $j$  when output port  $j$  belongs to the fanout set of the cell. A unicast cell has fanout equal to one, and its fanout destination is the only output port to which the cell is destined.

In the sequel we will use the following notation.

- $O = \{o_k\}$  is the set of switch output ports;  $|O| = N$
- $I = \{i_k\}$  is the set of switch input ports;  $|I| = N$
- $I_a \subseteq I$  is the set of active switch input ports, i.e. input ports with cells waiting to be switched;  $|I_a| = N_a$
- $R = \{r_k\}$  is the *request set* of (multicast) cells waiting to be switched at input queues
- $D_l = \{d_{lk}\}$  is the set of destinations of the  $l$ -th cell waiting to be switched
- $D_{lk} \subseteq D_l$  is the set of destinations of the  $l$ -th cell that are reached with the  $k$ -th cell transfer through the switch;  $\cup_k D_{lk} = D_l$ , and  $D_{lk} \cap D_{ln} = \emptyset, \forall k \neq n$

We now define a particular class of traffic patterns, which was already introduced in [18], and which is useful for our simulations and analytical models.

Input	Fanout sets	
$i_1$	$\{o_1, o_2, o_3\}$	$\{o_1, o_4, o_5\}$
$i_2$	$\{o_2, o_4, o_6\}$	$\{o_3, o_5, o_6\}$

TABLE I

EXAMPLE OF A 2-COMPLEX REQUEST SET FOR A  $6 \times 6$  SWITCH WITH 2 ACTIVE INPUT PORTS

*Definition 1:* A request set  $R$  is said  **$k$ -complex**, with<sup>1</sup>  $k \in \mathbb{N}$ ,  $k > 1$ , if:

- $k$  cells are associated with each active input  $i_l \in I_a$ ;
- $k$  cells are directed to each output  $o_l \in O$ ;
- for each sub-set of  $R$  comprising  $k$  cells, a destination exists to which all the cells in the subset are directed.

Table I reports an example of a 2-complex request set for a  $6 \times 6$  switch, where only inputs  $i_1$  and  $i_2$  are active.

In an  $N \times N$  switch where  $N_a$  input ports are active, with  $N = \binom{kN_a}{k}$  and  $N_a \geq 2$ , a  $k$ -complex request set  $R$  of size  $kN_a$  can be generated with the following algorithm:

- **Step 1.** Assign the first  $k$  cells in  $R$  to input 1, the second  $k$  cells to input 2, and so on, until the last  $k$  cells have been assigned to input  $N_a$ .
- **Step 2.** Form all the possible  $N = \binom{kN_a}{k}$  different sub-sets of  $R$  whose size is  $k$ , and create an arbitrary injective correspondence from subsets of cells to destinations.
- **Step 3.** To each cell in  $R$  assign all the destinations that correspond to sets containing the cell itself.

Note that, for each cell, the resulting fanout is equal to  $\binom{kN_a-1}{k-1} = \frac{N}{N_a}$ .

The average amount of traffic at each input (output) is called input (output) load. In this paper we normalize input (output) loads to line rates: a load equal to 1 means a fully utilized input (output) line. The traffic at the input of a switch is called *admissible* if no input load is larger than 1, and no output load is larger than 1. An input traffic is called *sustainable* if it can be transferred through the switch.

It is known that under multicast traffic, buffer space may be used more efficiently by IQ switches than by OQ switches, in the sense that any multicast cell currently in the switch is stored only in one buffer position. Several different ways of organizing the input queue system can be envisaged; the choice of the queue structure affects the scheduling discipline. We suppose to use at each input a set of FIFO queues, since they can be easily implemented and can provide high performance in terms of speed. The algorithm that computes the scheduling of cell transfers is supposed to scan only the cells at the heads of the input queues and only those cells may be scheduled for transfer from input to output ports.

In an IQ switch, when unicast cells are stored in just one FIFO queue per input port, the cell at the head of the queue can block the access to the switching fabric of subsequent cells, leading to the well-understood [20] head-of-the-line (HOL) blocking effect, which limits the maximum throughput achievable by IQ switches. In the case of unicast traffic only, a common approach for avoiding HOL blocking in IQ switches consists in using at

<sup>1</sup>In this paper  $\mathbb{N}$  denotes the set of non negative integers,  $\mathbb{R}$  denotes the set of real numbers, and  $\mathbb{R}^+$  denotes the set of non-negative real numbers.

each input separate queues for each output; this is called *Virtual Output Queueing* (VOQ). For multicast traffic, HOL blocking can be completely avoided using at each input separate FIFO queues for each one of the  $(2^N - 1)$  possible fanout configurations.

We considered in [18] a queue architecture with  $(2^N - 1)$  FIFO queues at each input, and we based on this architecture the definition of the optimal scheduling algorithm for multicast traffic. Unfortunately, the optimal algorithm is complex, and does not guarantee the orderly delivery of cells belonging to a multicast flow. On the contrary, in this paper we consider a per-multicast-flow queuing architecture, and assume that HOL blocking exists within flows. This choice seems justified by the fact that most of the routers on the market today adopt some sort of per-flow queuing architecture; moreover, the presence of HOL blocking within flows, which guarantees in-order delivery, seems the most reasonable choice, specially for the real-time services that are most likely to use multicast services.

Since we consider switching fabrics with intrinsic multicast capabilities, the cost of the transfer of one multicast cell equals the cost of the transfer of one unicast cell, provided that only a cell is transferred to each output. At each time slot, cells stored in input queues contend for access to the switching fabric in order to reach output ports. The decision about which cells can be transferred is made by the switch scheduler, which implements a *scheduling discipline*. The fact that multicast cells have multiple destinations implies that some scheduling disciplines may elect to transfer in just one time slot the multicast cell to all destinations, while others may elect to transfer the cell in several time slots, reaching non-overlapping and exhaustive subsets of destinations. In the latter case, a *partial service* is adopted when a cell reaches a subset of its remaining destinations with its current transfer, whereas a *total service* is adopted when a cell reaches all its remaining destinations with its current transfer. In the case of partial service, the *residue* [16], [21] is defined as the set of fanout destinations that have not yet been reached after a multicast cell is transferred towards output ports. Note that, given the fanout set of multicast cells, the same overall residue cardinality is produced by any work-conserving scheduling discipline. Each scheduling discipline with partial service uses a specific way of distributing or concentrating the residue among all contending inputs.

Although a hybrid approach [14], [22] can be adopted, one of the two following basic strategies is normally used by the scheduling discipline:

- *No fanout splitting* – Any multicast cell is transferred through the switching fabric once, only when all fanout destinations can be reached in the same time slot. If any of the fanout destinations cannot be reached because the multicast cell loses contention for an output port, the cell cannot be transferred, and it will contend again in a future time slot (normally the next one). This strategy is non-work-conserving, since it may happen that no cell is delivered to an available output because of contention. This discipline favors cells with small fanout.
- *Fanout splitting* [14], [16], [21], [23] – Multicast cells may be delivered to output ports over a number of time slots. Only those fanout destinations that could not be reached in previous time slots are considered in the next time slot.

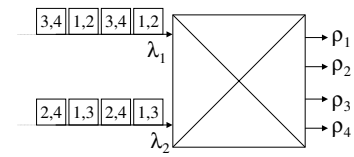


Fig. 1. Example traffic pattern leading to poor throughput with fanout splitting policies

Note that scheduling disciplines with no fanout splitting imply that a total service is always adopted, whereas multicast cells can receive either total or partial service if a fanout splitting scheduling discipline is used.

In the case of fanout splitting, every partial service causes an increase of the input load, leading to performance penalties. This effect is missing if a no fanout splitting discipline is adopted, but in this case another form of performance penalty is introduced, due to the non work-conserving service. A good scheduling discipline should therefore find the best compromise between these two major performance impairments.

With multicast traffic, the constraints for traffic sustainability are well-defined only for OQ switch architectures: no input load and no output load must be larger than 1. In the case of IQ switches, instead, while the best scheduling discipline for unicast traffic can sustain the same load of an OQ switch, it is easy to find a traffic pattern for which the best fanout splitting algorithm cannot reach the maximum throughput. When fanout splitting is allowed, a cell is scheduled in an average number of time slots equal to  $\alpha$ , with  $\alpha \geq 1$ . This fact increases the number of time slots necessary to schedule the cell transfers and thus either the input load must be lowered by the same factor  $\alpha$  ( $\alpha$  can be seen as a factor of bandwidth reduction) or a minimum internal speed-up equal to  $\alpha$  is required. This performance degradation due to excessive splitting, which produces additional load, was observed in [17].

The example shown in Figure 1 gives an indication of the problems that can arise with unfortunate traffic patterns. Inputs are fed with a sequence of back-to-back cells with alternating multicast destinations; numbers inside cells show their fanout destinations. Using an OQ switch architecture, it is possible to sustain the considered traffic pattern since the load offered to each input and output is smaller than one. Instead, with IQ switches and the best fanout splitting strategy,  $3/2$  cells can be scheduled in each time slot in the whole switch. This means that, for each input,  $3/4$  cells are scheduled in each time slot, so that each cell requires  $4/3$  time slots to be scheduled, and the bandwidth reduction factor is  $\alpha \approx 1.33$ .

Thus, with the setup shown in Figure 1, while an OQ switch can sustain the considered traffic pattern until  $\lambda_1 < 1$  and  $\lambda_2 < 1$ , IQ switches must impose  $\lambda_1 < 0.75$  and  $\lambda_2 < 0.75$ .

In [18] we presented some first theoretical results to investigate the impact of traffic patterns like the one in Figure 1; here we continue in this direction, and present further results obtained either from simulation (in Section III), or from analytical models (in Section IV).

Simulation results are obtained with a heuristic scheduling algorithm, because the optimum scheduler defined in [18] for multicast traffic is too complex for practical implementations, since it requires  $(2^N - 1)$  queues at each input and a complex re-queuing algorithm of partially served multicast cells. Indeed, any partial service of a multicast cell implies the re-queuing of the same cell at the input queue corresponding to the cell residue.

The heuristic scheduling algorithm that we used in simulations is based on a per-multicast-flow queueing architecture, where each queue is associated with a multicast flow, hence with one input port and a set of output ports, those toward which the multicast flow is directed. When a multicast cell is partially served, it remains at the head of its queue, and will contend for the residual set of destinations in the next time slot, thus blocking the following cells of the same flow (HOL blocking exists within a multicast flow).

#### A. Multicast Scheduling

The heuristic multicast scheduling algorithm is named **Multicast Greedy**. It uses as metrics the numbers of cells in input queues, and aims at serving each output to which at least one multicast cell is directed.

The scheduler handles a list of multicast-flow queues, sorted by queue lengths, i.e., by the number of cells stored in each queue. At each time slot, all input and output ports are initially *unmatched*. The scheduler orderly examines all queues of unmatched input ports, starting from the longest queue. The cell at the head of the examined queue is selected for a (possibly partial) transfer from the corresponding unmatched input port to all the unmatched output ports belonging to its fanout set. The input and output ports selected in this step are declared as *matched*. The algorithm iterates the above process, by orderly examining all queues of unmatched input ports, until either all output ports are matched, or no more non-empty queues exist. All the selected cells can be transferred across the switching fabric to their destinations matched by the scheduler in one time slot.

Note that the scheduler tries to transfer each selected cell toward the largest possible number of output ports. If an output port is in the fanout set of the selected multicast cell and it is unmatched, it is selected as a matched destination in the current time slot, and it is subtracted from the fanout set of the considered cell.

The complexity of the algorithm depends on the maximum number of queued multicast flows, and on the efficiency in keeping an ordered structure among queues. Note that, at each time slot, at most  $N_a$  queues change their lengths by one unit; this fact can be exploited to design efficient sorting techniques. Moreover, at each slot, the algorithm only requires the inspection of  $N^2$  queues (the longest one for each input-output pair), independently of the number of active multicast flows.

#### B. Traffic Patterns

The traffic patterns used in simulation experiments are described by a request set, identified by a list of input ports (the multicast flow sources), and, for each multicast flow, by a fanout

TABLE II  
QUASI-2-COMPLEX MULTICAST REQUEST SET

Input	Fanout sets	
$i_1$	$\{o_1, o_2, o_3, o_4\}$	$\{o_5, o_6, o_7, o_8\}$
$i_1$	$\{o_9, o_{10}, o_{11}, o_{12}\}$	$\{o_{13}, o_{14}, o_{15}, o_{16}\}$
$i_2$	$\{o_1, o_5, o_9, o_{13}\}$	$\{o_2, o_6, o_{10}, o_{14}\}$
$i_2$	$\{o_3, o_7, o_{11}, o_{15}\}$	$\{o_4, o_8, o_{12}, o_{16}\}$
$i_3$	$\{o_1, o_6, o_{11}, o_{16}\}$	$\{o_2, o_7, o_{12}, o_{13}\}$
$i_3$	$\{o_3, o_8, o_9, o_{14}\}$	$\{o_4, o_5, o_{10}, o_{15}\}$
$i_4$	$\{o_1, o_7, o_{10}, o_{13}\}$	$\{o_2, o_8, o_{11}, o_{14}\}$
$i_4$	$\{o_3, o_5, o_{12}, o_{15}\}$	$\{o_4, o_6, o_9, o_{16}\}$

set. Request sets correspond to probability mass functions from which random arrivals are generated. Unicast is considered as a special case of multicast, with fanout equal to one. In general, only a subset  $I_a$  of the switch input ports can be active, i.e., originate at least a multicast flow, in a given traffic pattern.

During simulation experiments, cells are randomly generated, according to the request set specification, uniformly loading all input ports. The fanout set of each multicast cell is randomly selected among the possibilities defined by the request set specification.

We used three different traffic patterns:

- *k-complex* ( $\Upsilon^k$ ): this traffic pattern refers to a  $k$ -complex request set, as described in Definition 1; this is one of the most critical traffic loads for the switch, and it can be defined only for very large switch sizes;
- *quasi-2-complex* ( $\Upsilon^Q$ ): this traffic pattern refers to a  $16 \times 16$  switch with  $N_a = 4$ , and its request set is specified by Table II; it is similar to a 2-complex traffic pattern, but is adapted to a small switch size;
- *random* ( $\Upsilon^R$ ): contrary to the two previous traffic patterns, that were constructed so as to be difficult to schedule, this traffic pattern is purely random; it may be an example of a traffic that is much simpler to schedule. It is based on a randomly generated request set, from which cell arrival probabilities are defined.

During our simulation experiments, the load offered to each active input port is chosen to be equal to the largest load which asymptotically avoids overloading the inputs. Note that, for  $\Upsilon^k$  and  $\Upsilon^Q$ , the load at the outputs is equal to the load offered to each input, so that the chosen input offered load is equal to 1.0. Instead, for  $\Upsilon^R$ , the input load must be reduced in order to obtain an admissible load for each output.

Simulation experiments were run until the confidence interval of the estimate of the average throughput for each destination reached with probability 0.95 a relative width smaller than 3%. The estimation of the confidence interval width is obtained with the batch means approach. Each queue is assumed to have finite capacity, equal to 1000 cells.

#### C. Simulation Results

Table III shows throughput results for a 2-complex traffic pattern  $\Upsilon^2$ , for different switch sizes and numbers of active inputs  $N_a$ . The throughput can be observed to decrease as the switch size increases. For  $N_a \geq 16$ , throughput values are less than 0.5; this is an indication of the fact that, using our greedy heuristic

TABLE III  
THROUGHPUT FOR  $\Upsilon^2$  TRAFFIC

$N_a$	$N$	Max Throughput
2	6	0.679
4	28	0.588
8	120	0.521
16	496	0.479
32	2016	0.437
64	8128	0.405
128	32640	0.378
256	130816	0.355

TABLE IV  
THROUGHPUT FOR  $\Upsilon^k$  TRAFFIC AND FIXED  $N_a = 8$

$k$	$N$	Max Throughput
2	120	0.521
3	2024	0.396
4	35960	0.334
5	658008	0.296

algorithm, a speed-up larger than 2 is necessary to transfer a 2-complex multicast traffic pattern without loss. Remember that OQ architectures can sustain any  $k$ -complex traffic pattern, but they require a speed-up equal to the number of input/output ports  $N$ .

Table IV reports throughput values for a fixed number of active input ports,  $N_a = 8$ , and variable  $k$ , for a  $k$ -complex traffic pattern  $\Upsilon^k$ . For increasing  $k$ , the throughput can be seen to decrease considerably. This is again an indication of the fact that a significant speed-up may be necessary to transfer  $k$ -complex traffic patterns.

With the quasi-2-complex traffic pattern  $\Upsilon^Q$  of Table II we obtained a throughput equal to 0.562, in the case  $N = 16$ ,  $N_a = 4$ ; this throughput value is lower than the one obtained for  $N_a = 4$  and a 2-complex traffic pattern, as can be seen by Table III. Although this result may seem strange, since the  $k$ -complex traffic pattern was defined as a worst-case scenario, an explanation is possible. Define an operator  $\Gamma_d(\Upsilon)$  which gives the probability that  $d$  cells (randomly chosen from different inputs under traffic  $\Upsilon$  at unitary load) have at least one destination in common, i.e., that they are in conflict, and thus cannot be completely transferred in the same slot time.  $\Gamma_d(\Upsilon)$  can be seen as a measure of the ‘difficulty’ of scheduling  $d$  cells in the same time slot, since when  $d$  cells are in conflict, their transfer must be split among at least  $d$  different time slots;  $\Gamma_d(\Upsilon)$  is also a degree of ‘similarity’ to the  $d$ -complex traffic pattern. Note that, given the definition of a  $k$ -complex traffic pattern

$$\Gamma_d(\Upsilon^k) = \begin{cases} 1 & \text{for } 1 < d \leq k \\ 0 & \text{for } d > k \end{cases}$$

For the quasi-2-complex traffic scenario,  $\Gamma_2(\Upsilon^Q) = 0.92$ ,  $\Gamma_3(\Upsilon^Q) = 0.25$  and  $\Gamma_4(\Upsilon^Q) = 0.063$ . The fact that  $\Gamma_2(\Upsilon^Q) = 0.92$  would imply that  $\Upsilon^Q$  is slightly less difficult to schedule than a 2-complex traffic pattern, but since  $\Gamma_3(\Upsilon^Q) = 0.25$ ,  $\Upsilon^Q$

has a non negligible probability of behaving like a 3-complex traffic pattern, thus being more difficult to schedule than  $\Upsilon^2$ . This is an indication of the fact that multicast traffic patterns exist, which are extremely difficult to schedule in IQ switches.

We also investigated by simulation the switch throughput for the traffic pattern  $\Upsilon^R$  in a  $16 \times 16$  switch, when all input ports are active, and the offered traffic is composed partly by unicast traffic and partly by multicast traffic; the ratio between the multicast and unicast offered load is a random variable uniformly distributed between 0.5 and 1.5. The unicast traffic is uniform across input and output ports; instead, the multicast traffic is uniform across inputs, but the fanout set of every multicast cell is selected at random, and destinations are randomly chosen with uniform distribution. Over 200 random traffic patterns  $\Upsilon^R$  were studied; for all of them our heuristic greedy algorithm reaches at least 99% throughput at each output. Note that, for the set of considered random traffic patterns, we obtained that  $0.0084 \leq \Gamma_2(\Upsilon^r) \leq 0.022$  and  $0.0005 \leq \Gamma_3(\Upsilon^r) \leq 0.001$ .

From the observation of simulation results it is possible to conclude that, for some multicast traffic patterns, IQ switches provide very good performance. However, it is also easy to identify traffic patterns that lead to very low throughput values. The presence of such traffic patterns led us to a theoretical investigation of scheduling algorithms in IQ switches loaded with multicast traffic.

#### IV. ANALYTICAL RESULTS

Given the evidence about the existence of traffic patterns that are quite difficult to schedule in IQ switches, we proceeded to an analytical investigation of the limit throughput in IQ switch architectures loaded with unicast and multicast traffic. However, since the optimal scheduling defined in [18] is impractical, and the Multicast Greedy algorithm is difficult to analyze, we concentrated on a different class of scheduling algorithms, i.e., *frame-based* schedulers. This means that we assume that a fixed number of slots is grouped in a frame, and the scheduler operates within the frame, by allocating slots for the transmission of multicast flows, based upon the situation of input queues at the beginning of the frame. The requirement for the absence of packet losses is that, on the average, all cells queued at the beginning of a frame be scheduled within the frame slots. Note that an IQ switch achieves 100% throughput if it is able to sustain every admissible traffic pattern. Frame-based scheduling provides an upper bound to the switch performance when the frame size grows to infinity; indeed, in such case the scheduler has a complete visibility of an infinite amount of traffic that can be scheduled over an infinite number of time slots, with much less constraints with respect to slot-by-slot schedulers. Instead, frame-based scheduling is not very significant with frame size equal to 1, since in this case all cells that arrive in a slot must be scheduled in the following slot; this is quite a strict requirement. Hence, for frame size growing to infinity, the ratio between the number of slots in a frame and the number of scheduled cells per port coincides with the required switch speed-up for 100% throughput. Note also that frame-based schedulers can be a very interesting approach for the provision of delay guarantees to real-time traffic flows.

We first modify and extend the definitions introduced at the

beginning of Section II in the following way:

- $R = \{r_k\}$  is the request set of (multicast) cells waiting to be switched at input queues at the beginning of a time frame
- $F = \{f_k\}$  is the set of time slots in a switching frame;  $|F| = L$ ;  $L$  is the frame length, whose minimum value necessary for the transfer of all cells in  $R$  will be discussed in this section; note that at most  $NL$  cells can be transferred in a frame of length  $L$
- $T = \{t_k\}$  is the set of cells switched in a frame; we look for scheduling algorithms such that  $T = R$

Each (multicast) cell  $r_l \in R$  must be transferred from its input port to all its destination output ports. Thus, each  $r_l \in R$  is associated with a pair  $(i_l, D_l)$ , where  $i_l \in I$ ,  $D_l \subseteq O$ .

**Definition 2:** Two multicast cells  $r_l$  and  $r_k$  are said to be in conflict if they either are associated with the same input port ( $i_l = i_k$ ), or have at least one destination in common ( $D_l \cap D_k \neq \emptyset$ ).

In the switch architecture we consider, a subset of the cells in  $R$  can be transferred from input ports to output ports in the same time slot, provided that no cells in the subset are in conflict. Conflicts may force the transfer of a cell  $r_l$  to disjoint subsets of  $D_l$  in different time slots.

**Definition 3:** A Scheduling  $\mathcal{S}[R]$  is defined as a function whose domain is  $R$  and whose image is  $2^{I \times 2^O \times F}$ .  $U$  is the set of transfer units, whose elements  $u_{lk}$  associate with each multicast cell  $r_l \in R$  a triple  $(i_l, D_{lk}, f_k)$ , where  $i_l \in I$  is the cell input port,  $D_{lk} \subseteq D_l$  is a non-empty subset of the cell destinations, and  $f_k \in F$  is a time slot in the frame.

Informally,  $\mathcal{S}$  assigns to each multicast cell  $r_l \in R$  a set of transfer units  $u_{lk} = (i_l, D_{lk}, f_k) \in U$ ; each transfer unit allows a (possibly partial) transfer of the multicast cell, i.e. a transfer toward a subset of the cell destinations.

**Definition 4:** A Scheduling  $\mathcal{S}[R]$  is *admissible* if, given any two elements  $(i_1, D_{1k}, f_k) \in U$  and  $(i_2, D_{2n}, f_n) \in U$ , such that  $f_k = f_n$ , then

- $i_1 \neq i_2$ , and
- $D_{1k} \cap D_{2n} = \emptyset$

i.e., they are non conflicting.

In other words, no (possibly partial) transfer of two different multicast cells can occur in the same time slot of the frame if the two cells are in conflict.

**Definition 5:** An Admissible Scheduling  $\mathcal{AS}[R]$  is *complete* if  $\cup_k D_{lk} = D_l$ ,  $\forall r_l \in R$ , i.e., if it achieves a complete transfer of all cells belonging to  $R$ , using a subset of slots belonging to  $F$ .

**Definition 6:** A Time Slot Assignment  $\mathcal{TSA}[R]$  is defined as a function whose domain is  $R$  and whose image is the power set of  $F$  ( $\mathcal{TSA}[R] : R \rightarrow 2^F$ ).

Informally, a  $\mathcal{TSA}$  defines a correspondence between each cell  $r_l \in R$  and the set of time slots in the frame in which copies of the cell are transferred.

A more complex, but more useful definition of a  $\mathcal{TSA}[R]$  will allow us to relate a Time Slot Assignment  $\mathcal{TSA}[R]$  to an Admissible Scheduling  $\mathcal{AS}[R]$ .

**Definition 7:** A Timing function  $\mathcal{T}[2^U] : 2^U \rightarrow 2^F$  is a function that, given a set  $\{(i_l, D_{lk}, f_k)\}$ , returns a set containing all the third components  $f_k$  of each element.

<sup>2</sup>Given a set  $A$ ,  $2^A$  denotes the power set of  $A$ .

Informally, the timing function  $\mathcal{T}$  extracts the time slot information related to (possibly partial) multicast cell transfers.

**Definition 8:** A Time Slot Assignment  $\mathcal{TSA}[R] = \mathcal{T}(\mathcal{AS}[R])$  is a composition of two functions, a Timing function  $\mathcal{T}$  applied to the image of an Admissible Scheduling function  $\mathcal{AS}$ .

Although a time slot assignment  $\mathcal{TSA}[R]$  does not completely specify the scheduling of multicast cells to be transferred from sources to destinations (it only defines the set of used time slots, with no information about sources and destinations), we will concentrate our attention only on  $\mathcal{TSA}[R]$ .

**Definition 9:** A time slot assignment  $\mathcal{TSA}[R]$  is said to be *complete* if there exists a complete Admissible Scheduling  $\mathcal{AS}$  such that  $\mathcal{TSA}[R] = \mathcal{T}(\mathcal{AS}[R])$ .

**Definition 10:** For each sub-set  $A \subseteq R$ , let  $\mathcal{I}_{\mathcal{TSA}}(A)$  be the image of  $A$  through  $\mathcal{TSA}[R]$ , i.e., the set comprising all time slots in which some element of  $A$  is transmitted.

**Definition 11:** A time slot assignment  $\mathcal{TSA}[R]$  is said to satisfy the *one-shot property* if  $|\mathcal{I}_{\mathcal{TSA}}(\{r_l\})| = 1, \forall r_l \in R$ .

Informally, a time slot assignment  $\mathcal{TSA}[R]$  satisfies the one-shot property if all multicast cells are transmitted with no fanout splitting.

**Theorem 1:** Let  $R$  be a  $k$ -complex set ( $k \geq 2$ ) with  $|R| = kN_a$ . If  $L < kN_a$ , then a complete  $\mathcal{TSA}$  satisfying the one-shot property does not exist.

*Proof:* The proof is immediate, since at most one cell can be scheduled in each time slot because  $R$  is  $k$ -complex. ■

**Lemma 1:** Let  $R$  be a  $k$ -complex request set. Consider a complete  $\mathcal{TSA}[R]$  and any set  $D \subseteq R$  such that  $|D| = k$ ; then,  $|\mathcal{I}_{\mathcal{TSA}}(D)| \geq k$ .

*Proof:* Suppose that there exists a subset  $D \subseteq R$ , with  $|D| = k$ , such that  $|\mathcal{I}_{\mathcal{TSA}}(D)| < k$ ; since the request set is  $k$ -complex, there exists a destination  $o^*$  to which all cells in  $D$  are directed. Copies of the cells in  $D$  directed to  $o^*$  must be transmitted in different time slots in order to avoid conflicts. Thus, at least  $k$  slots are necessary to transmit all the cells in  $D$ . As a consequence, the schedule  $\mathcal{TSA}[R]$  is not complete. ■

As a trivial consequence:

**Lemma 2:** Let  $R$  be a  $k$ -complex request set. Consider a  $\mathcal{TSA}[R]$  on  $F$ ; if there exists a set  $D$  s.t.  $D \subseteq R$ ,  $|D| = k$  and  $|\mathcal{I}_{\mathcal{TSA}}(D)| < k$ , then  $\mathcal{TSA}[R]$  is not complete.

We now introduce a lemma that will be used in subsequent proofs.

**Lemma 3:** For a strictly positive integer-valued random variable  $X$  whose average value  $E[X] = \mu$  does not exceed  $m$ , the  $\frac{100}{m}$ -th percentile does not exceed  $m$ .

*Proof:*

Since  $\mu \leq m$ , by definition:

$$\begin{aligned} E[X] = \mu &= \sum_{i=1}^{\infty} i \Pr\{X = i\} \\ &= \sum_{i=1}^m i \Pr\{X = i\} + \sum_{m+1}^{\infty} i \Pr\{X = i\} \\ &\geq \sum_{i=1}^m \Pr\{X = i\} + (m+1) \sum_{m+1}^{\infty} \Pr\{X = i\} \\ &= \Pr\{X \leq m\} + (m+1)(1 - \Pr\{X \leq m\}) \end{aligned}$$

But, since  $\mu \leq m$ :

$$\Pr\{X \leq m\} + (m+1)(1 - \Pr\{X \leq m\}) \leq m$$

and finally

$$\Pr\{X \leq m\} \geq \frac{1}{m}$$

We also need the following result in subsequent proofs. ■

*Lemma 4:* If  $u$  transfer units (objects) are transmitted in  $t$  time slots (containers), it is possible to find a subset  $C$  of containers, such that  $|C| = c$  in which at least  $\lceil \frac{u}{t}|C| \rceil$  objects are contained for each integer positive number  $c < t$ .

## V. MINIMUM FRAME LENGTHS FOR COMPLETE MULTICAST SCHEDULES

The main analytical result of this paper is in the following theorem.

*Theorem 2:* Let  $R$  be a  $k$ -complex request set, such that  $|R| = kN_a$ . For any finite speed-up value  $S$ , there exists an integer  $N_0$  such that  $\forall N_a > N_0$  no complete  $\mathcal{TSA}[R]$  exists with frame length  $L \leq Sk$ .

Note that  $S$  is equivalent to an internal switch speed-up, since the  $k$ -complex request set has  $k$  cells per output and  $k$  cells per active input, hence it can be transferred on the output lines in  $k$  time slots in an OQ switch. Thus, the theorem states that, if the number of active input ports is sufficiently large, no speed-up value is sufficient to obtain a complete  $\mathcal{TSA}$ , hence to achieve 100% throughput. This result will be proved in the cases  $S = 2$ ,  $N_0 = 32$ , and  $S = 3$ ,  $N_0 = 2187$ ; the proofs can be extended to larger values of  $S$ .

*Proof:* We consider in this proof the case  $S = 2$ , so that we must show that no complete  $\mathcal{TSA}$  exists if  $L \leq 2k$ . Obviously, if no complete  $\mathcal{TSA}$  exists for a frame length  $L = 2k$ , no complete  $\mathcal{TSA}$  can be found for shorter frame lengths. Thus, we consider the case  $L = 2k$ .

We will prove that, for any possible  $\mathcal{TSA}[R]$ , a subset  $D$  of  $R$  can be found, such that  $|D| = k$  and  $|\mathcal{I}_{\mathcal{TSA}}(D)| < k$ . Then, for Lemma 2, we will conclude that the considered  $\mathcal{TSA}[R]$  is not complete.

The identification of set  $D$  will require a number of steps, defining a chain of subsets of  $R$ :  $E \subseteq D \subseteq C \subseteq B \subseteq R$ , as well as a subset  $W$  of the time slots in the frame  $F$ .

- Set  $B$  comprises all cells whose transfer is scheduled either once or twice in the frame (not a larger number of times).
- Set  $W$  comprises a number of slots not larger than  $k/2$ , chosen so as to maximize the number of cells in  $B$  transferred (possibly partially) within  $W$ .
- Set  $C$  comprises all cells whose transfer is scheduled at least once within  $W$  (but not more than twice, being  $C \subseteq B$ ).
- Set  $E$  comprises all cells whose transfer is scheduled only within  $W$  (either once or twice, being  $E \subseteq B$ ).
- Set  $D$  is obtained by adding cells to  $E$  (if necessary) in order to obtain  $|D| = k$ .

It is possible to visualize these sets by considering a scheduling as a  $L \times N$  matrix  $\mathcal{S}$ , where rows correspond to time slots in the frame  $F$ , and columns correspond to switch input ports in  $I$ . The elements of  $\mathcal{S}$  are  $s_{nl} = 0$  if no cell from  $i_l$  is scheduled

for transfer in slot  $f_n$ , and  $s_{nl} = m$  if in slot  $f_n$  the transfer of cell  $r_m$  is scheduled from  $i_l$  to a subset of its destinations  $D_{mk}$ . By so doing, the definition of the above sets becomes:

- $B$  comprises all cells whose identifier appears either once or twice in  $\mathcal{S}$  (but no more than twice)
- $W$  comprises the rows of  $\mathcal{S}$  with the largest numbers of elements of  $B$  (selecting at most  $k/2 - 1$  rows)
- $C$  comprises all cells of  $B$  whose identifier appears in the rows of  $\mathcal{S}$  within  $W$  (the same identifier may appear also once outside  $W$ )
- $E$  comprises all cells of  $B$  whose identifier appears only in the rows of  $\mathcal{S}$  within  $W$  (either once or twice)

To obtain a complete  $\mathcal{TSA}[R]$  it is necessary that  $\mathcal{I}_{\mathcal{TSA}}(\{r_l\}) \neq \emptyset, \forall r_l \in R$ ; thus, we will consider only the class of  $\mathcal{TSA}[R]$  for which  $\mathcal{I}_{\mathcal{TSA}}(\{r_l\}) \neq \emptyset, \forall r_l \in R$ .

If  $L = 2k$ , then the average size of the image of individual cells in  $R$  through  $\mathcal{TSA}[R]$  cannot be greater than 2 (because the copies of  $kN_a$  cells must fit into  $2kN_a$  slots; considering matrix  $\mathcal{S}$ , at most all its  $2kN_a$  elements can be nonzero, thus scheduling on the average twice the transfer of each one of the  $kN_a$  cells), i.e.:  $E_{r_l} [|\mathcal{I}_{\mathcal{TSA}}(\{r_l\})|] \leq 2$ . As a consequence, from Lemma 3,  $|\mathcal{I}_{\mathcal{TSA}}(\{r_l\})| \leq 2$  for at least half of the cells in  $R$ .

Let  $B \subset R$  be the set of cells for which  $|\mathcal{I}_{\mathcal{TSA}}(\{r_l\})| \leq 2, \forall r_l \in B$ ; for Lemma 3,  $|B| \geq |R|/2 = kN_a/2$ . Thus, the average number of cells in  $B$  that are scheduled in each slot of the frame is never less than  $\frac{|B|}{L} \geq \frac{kN_a/2}{2k} = \frac{N_a}{4}$ .

For Lemma 4, there exists a set  $W$  of time slots with  $|W| < k/2$  in which a set  $C$  of cells belonging to  $B$  (i.e.,  $C \subseteq B$ ) is scheduled, and  $|C| \geq |W|N_a/4$ .

Since it will be later necessary to obtain  $|C| \geq 4k$ , this implies  $N_a > 32$ , and  $k$  must be sufficiently large to produce enough cells to significantly fill the frame with transmissions.

Let  $E$  be the set of cells  $r_l \in C$  for which  $\mathcal{I}_{\mathcal{TSA}}(\{r_l\}) \subseteq W$ . Note that  $E$  comprises all the cells in  $C$  for which  $|\mathcal{I}_{\mathcal{TSA}}(\{r_l\})| = 1$ , and that  $E$  may be a null set.

Now we consider three cases:

- $|E| \geq k$ . We can set  $D = E$ , obtaining  $|D| \geq k$  and  $|\mathcal{I}_{\mathcal{TSA}}(D)| < k$ .
- $k/2 \leq |E| < k$ . Let  $D$  be a set of cells such that  $E \subseteq D \subset C, |D| = k$ . We define a set of slots  $X = \mathcal{I}_{\mathcal{TSA}}(D \setminus E) \cap W$ . Since  $\mathcal{I}_{\mathcal{TSA}}(E) \subseteq W$ , by construction:

$$\mathcal{I}_{\mathcal{TSA}}(D) \subseteq W \cup [\mathcal{I}_{\mathcal{TSA}}(D \setminus E) \setminus X]$$

$$|\mathcal{I}_{\mathcal{TSA}}(D)| \leq |W| + |\mathcal{I}_{\mathcal{TSA}}(D \setminus E) \setminus X|$$

Since  $\mathcal{I}_{\mathcal{TSA}}(r_l), \forall r_l \in (D \setminus E)$  has at most one element in  $[\mathcal{I}_{\mathcal{TSA}}(D \setminus E) \setminus X]$ :

$$\begin{aligned} |\mathcal{I}_{\mathcal{TSA}}(D \setminus E) \setminus X| &= \\ &= |\cup \mathcal{I}_{\mathcal{TSA}}(\{r_l | r_l \in (D \setminus E)\}) \setminus X| \leq \\ &\leq \sum |\mathcal{I}_{\mathcal{TSA}}(\{r_l | r_l \in (D \setminus E)\}) \setminus X| \leq k/2 \end{aligned}$$

Since  $|W| < k/2$ , we have  $|\mathcal{I}_{\mathcal{TSA}}(D)| < k/2 + k/2 = k$ .

- $0 \leq |E| < k/2$ . Define  $h = k - |E|$ ; obviously,  $k/2 < h \leq k$ . Each one of the cells in  $C \setminus E$  (at least  $3k + h$ ) will be scheduled once in  $F \setminus W$  (referring to matrix  $\mathcal{S}$ , the cells in  $C \setminus E$  appear once in the rows within  $W$  and once in rows not belonging to

$W$ ). On the average, at least  $\frac{3k+h}{|F \setminus W|}$  cells belonging to  $C \setminus E$  will be scheduled in each time slot in  $F \setminus W$  (we have set  $C \geq 4k$ , and we have  $|E| = k-h$ ; thus  $|C \setminus E| \geq 4k - (k-h)$ ). Note that  $\frac{3k+h}{|F \setminus W|} > \frac{h}{k/2}$ , because  $3k+h \geq 4h$  and  $|F \setminus W| < |F| = 2k$ . Thus, for Lemma 4, it is possible to find a set of cells  $E'$ , s.t.  $E' \subseteq (C \setminus E)$ ,  $|E'| = h$ , and  $|\mathcal{I}_{\mathcal{TSA}}(E') \setminus (\mathcal{I}_{\mathcal{TSA}}(E') \cap W)| \leq \frac{k}{2}$  (referring to matrix  $\mathcal{S}$ ,  $E'$  comprises the cells that belong to the  $k/2$  rows not in  $W$  containing the largest numbers of cells in  $C \setminus E$ ).

Finally, let  $D = E \cup E'$ . By construction  $|D| = k$  and

$$|\mathcal{I}_{\mathcal{TSA}}(D)| = |\mathcal{I}_{\mathcal{TSA}}(E)| + |\mathcal{I}_{\mathcal{TSA}}(E')| < k$$

When the size of the switch grows, it is possible to generalize the proof for frame sizes which are integer multiples of  $k$ . We briefly sketch the proof for  $L = 3k$ .

*Proof:*

Also in this case we will prove that, for each  $\mathcal{TSA}[R]$ , a subset  $A$  of  $R$  can be found, such that  $|A| = k$  and  $|\mathcal{I}_{\mathcal{TSA}}(A)| < k$ . Then, for Lemma 2 we conclude that the considered  $\mathcal{TSA}[R]$  is not complete.

If  $L = 3k$ , then the average size of the image of individual cells in  $R$  through  $\mathcal{TSA}[R]$  cannot be greater than 3, i.e.:  $E_{r_i} [|\mathcal{I}_{\mathcal{TSA}}(\{r_i\})|] \leq 3$ . As a consequence, from Lemma 3,  $|\mathcal{I}_{\mathcal{TSA}}(\{r_i\})| \leq 3$  for at least one third of the cells in  $R$ .

Let  $B \subseteq R$  be the set of cells for which  $|\mathcal{I}_{\mathcal{TSA}}(\{r_i\})| \leq 3$ ,  $\forall r_i \in B$ ; for Lemma 3,  $|B| \geq |R|/3 = kN_a/3$ . Thus, the average number of cells in  $B$  that are scheduled in each slot of the frame is never less than  $\frac{|B|}{L} \geq \frac{kN_a}{3} \frac{1}{3k} = \frac{N_a}{9}$ .

For Lemma 4, there exists a set  $W$  of time slots with  $|W| < k/3$  in which a set  $C$  of cells belonging to  $B$  (i.e.,  $C \subseteq B$ ) is scheduled, and  $|C| \geq |W|N_a/9$ .

Since it will be later necessary to obtain  $|C| \geq 81k$ , this implies  $N_a > 2187$ , and  $k$  must be sufficiently large to produce enough cells to significantly fill the frame with transmissions.

Let  $E$  be the set of cells  $r_i \in C$  for which  $|\mathcal{I}_{\mathcal{TSA}}(\{r_i\})| \leq 1$ . Note that  $E$  comprises all the cells in  $C$  for which  $|\mathcal{I}_{\mathcal{TSA}}(\{c_i\})| = 1$ , and that  $E$  may be a null set.

For simplicity, we assume that  $E = \emptyset$ . The proof, however can be extended to the more general case. In this case each one of the  $|C|$  cells in  $C$  will be scheduled once or twice in the residual time slots belonging to  $T \setminus W$ .

On the average,  $\frac{|C|}{|T \setminus W|}$  cells belonging to  $C$  will be scheduled in each time slot belonging to  $T \setminus W$ . Note that  $\frac{|C|}{|T \setminus W|} > 27$ . Thus, for Lemma 4, there exists a set of  $W'$  time slot, with  $|W'| = k/3$ , in which at least  $9k$  cells in  $C$  are scheduled. Let  $F$  the set comprising these cells.

Let  $E'$  be the set of cells  $r_i \in F$  for which  $|\mathcal{I}_{\mathcal{TSA}}(\{r_i\})| \leq 2$ . Note that  $E'$  comprises all the cells in  $F$  for which  $|\mathcal{I}_{\mathcal{TSA}}(\{r_i\})| \leq 2$ . Again, we suppose that  $E' = \emptyset$ .

In this case each one of the  $|F|$  cells in  $F$  will be scheduled once in the residual time slots belonging to  $T \setminus (W \cup W')$ . On average  $\frac{|F|}{|T \setminus (W \cup W')|}$  cells belonging to  $F$  will be scheduled in each time slot belonging to  $T \setminus (W \cup W')$ . Note that  $\frac{|F|}{|T \setminus (W \cup W')|} > 3$ . Thus, for Lemma 4, there exists a set of  $W''$  time slot, with  $|W''| = k/3$ , in which at least  $k$  cells in  $F$  are scheduled.

Let  $G$  be the set comprising these cells. Since  $|G| = k$  and  $\mathcal{I}_{\mathcal{TSA}}(G) \subseteq (W \cup W' \cup W'')$ , we get  $|\mathcal{I}_{\mathcal{TSA}}(G)| < k$ . ■

## VI. CONCLUSIONS

In this paper we presented results concerning the throughput achievable in IQ cell-based switches and routers loaded with multicast traffic, and we showed with simulation experiments and analytical models that throughput limitations exist in such architectures, contrary to the case of unicast traffic, for which IQ switches were proved to yield the same throughput as OQ switches.

Our simulation study of different switch configurations allowed the identification of multicast traffic patterns leading to good performance in IQ switches, but also of traffic patterns producing unexpectedly low throughput values.

Working with the latter traffic patterns, and with frame-based scheduling algorithms, we then analytically proved that with some switch configurations and traffic patterns, a wide class of scheduling algorithms leads to poor performance. In particular, we could prove that in large IQ multicast switches loaded with  $k$ -complex traffic patterns, given any finite speed-up, it is possible to find a number of active input ports that does not allow 100% throughput to be reached.

The fact that this result was proved under frame-based scheduling algorithms is also an indication of the fact that no delay guarantees can be provided to multicast flows in IQ switches if the input load approaches 100%.

The results that we obtained for scheduling algorithms in IQ switches can be actually applied to schedulers in a number of other communication systems; among those, we can mention all-optical networks with broadcast-and-select architectures, multi-channel wireless networks, and satellite systems.

## REFERENCES

- [1] Waitzman D., Deering S., Partridge C., "Distance Vector Multicast Routing Protocol", RFC 1075, Nov. 1988
- [2] Deering S., Estrin D., Farinacci D., Jacobson V., Liu C.G., Wei L., "The PIM Architecture for Wide Area Multicasting", *IEEE/ACM Transactions on Networking*, vol.4, n.2, pp.153-162, Apr. 1996
- [3] A.Ballardie, "Core Based Trees (CBT) Multicast Routing Architecture", RFC 2201, Sept. 1997
- [4] 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 (ETT)*, vol. 10, n. 2, Mar./Apr. 1999
- [5] "GFR MultiGigabit Routers", Product Overview, [www.lucent.com](http://www.lucent.com), Apr. 2000
- [6] "Cisco 12000 Gigabit Switch Router", Product Overview, [www.cisco.com](http://www.cisco.com), Apr. 2000
- [7] McKeown N., Izzard M., Mekkittikul A., Ellesick B., Horowitz M., "The Tiny Tera: a packet switch core", *IEEE Micro Magazine*, vol. 17, pp. 27-40, Feb. 1997
- [8] Anderson T., Owicki S., Saxe J., Thacker C., "High speed switch scheduling for local area networks", *ACM Transactions on Computer Systems*, vol. 11, n. 4, pp. 319-352, Nov. 1993
- [9] Hung A., Kesidis G., McKeown N., "ATM input-buffered switches with guaranteed-rate property", *IEEE ISCC'98*, Athens, Greece, pp. 331-335, July 1998
- [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*, Los Alamitos, CA, USA, vol. 1, pp.20-8, 1997
- [11] Partridge C., et al., "A 50-Gb/s IP router", *IEEE Transactions on Networking*, vol. 6, n. 3, pp. 237-248, June 1998
- [12] Guo M., Chang R., "Multicast ATM switches: survey and performance evaluation", *Computer Communication Review*, vol. 28, n. 2, pp. 98-131, Apr. 1998

- [13] Liu Z., Righter R., "Scheduling multicast input-queued switches", *Journal of Scheduling*, John Wiley & Sons, vol. 2, n. 3, pp.99-114, May 1999
- [14] Chen X., Lambadaris I., Hayes J., "A general unified model for performance analysis of multicast switching", *IEEE GLOBECOM'92*, New York, NY, USA, vol. 3, pp. 1498-502, 1992
- [15] Hayes J.F., Breault R., Mehmet-Ali M.K., "Performance analysis of a multicast switch", *IEEE Transactions on Communications*, vol. 39, n. 4, pp. 581-587, Apr. 1991
- [16] Hui J., Renner T., "Queueing strategies for multicast packet switching", *IEEE GLOBECOM'90*, New York, NY, USA, vol. 3, pp. 1431-7, 1990
- [17] Kim C.K., Lee T.T., "Performance of call splitting algorithms for multicast traffic", *IEEE INFOCOM'90*, Los Alamitos, CA, USA, vol. 2, pp. 348-56, 1990
- [18] Ajmone Marsan M., Bianco A., Giaccone P., Leonardi E., Neri F., "Optimal Multicast Scheduling in Input-Queued Switches", submitted for publication
- [19] Chuang S.T., Goel A., McKeown N., Prabhakar B. "Matching output queuing with a combined input/output-queued switch", *IEEE JSAC*, vol. 17, n. 6, pp.1030-39, June 1999
- [20] Karol M., Hluchyj M., Morgan S., "Input versus output queuing on a space division switch", *IEEE Trans. on Communications*, vol. 35, n. 12, pp. 1347-1356, Dec. 1987
- [21] McKeown N., Prabhakar B., "Scheduling multicast cells in an input-queued switch", *INFOCOM'96*, San Francisco, CA, USA, vol. 1, pp. 261-278, Mar. 1996
- [22] Chen W., Chang Y., Hwang W., "A high performance cell scheduling algorithm in broadband multicast switching systems", *IEEE GLOBECOM'97*, New York, NY, USA, vol. 1, pp. 170-4, 1997
- [23] Prabhakar B., McKeown N., Ahuja R., "Multicast scheduling for input-queued switches", *IEEE Journal on Selected Areas in Communications*, vol. 15, n. 5, pp. 855-66, June 1997