University of Castilla-La Mancha

A publication of the

Department of Computer Science

Traffic Scheduling Solutions with QoS Support for an Input-Buffered MultiMedia Router

by

Blanca Caminero, Carmen Carrión, Francisco J. Quiles,

José Duato, Sudhakar Yalamanchili

Technical Report **#DI-03-02-34** January, 2003

A shorter version of this paper has been submitted to the *IEEE Transactions on Parallel and Distributed Systems*.
José Duato is with the Deptartment of Information Systems and Computer Architecture (DISCA), Universidad Politécnica de Valencia.
Sudhakar Yalamanchili is with the School of Electrical and Computer Engineering, Georgia Institute of Technology (USA).

DEPARTAMENTO DE INFORMÁTICA
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34.967.599200, Fax +34.967.599224

Traffic Scheduling Solutions with QoS Support for an Input-Buffered MultiMedia Router

Blanca Caminero, Carmen Carrión, Francisco J. Quiles
Dept. of Computer Science. Escuela Politécnica Superior.
Universidad de Castilla-La Mancha. 02071 - Albacete, SPAIN
{blanca, carmen, paco}@info-ab.uclm.es

José Duato
Dept. of Information Systems and Computer Architecture (DISCA)
Univ. Politécnica de Valencia. 46071 - Valencia, SPAIN
jduato@gap.upv.es

Sudhakar Yalamanchili
School of Electrical and Computer Engineering.
Georgia Institute of Technology. Atlanta, 30332-0250 Georgia, USA
sudha@ece.gatech.edu

Abstract

Quality of Service (QoS) support in local and cluster area environments has become an issue of great interest in recent years. Most current high-performance interconnection solutions have been designed to enhance conventional best-effort traffic performance, but are not well-suited to the special requirements of the new multimedia applications. The MultiMedia Router (MMR) tries to offer hardware-based QoS support within a compact interconnection element. One of the key elements in the MMR architecture are the algorithms used in traffic scheduling. These algorithms are responsible for the order in which information is forwarded through the switch fabric. Thus, they are closely related to the QoS-provisioning mechanisms. In this work, several traffic scheduling algorithms developed for the MMR architecture are described. Their general organization is motivated by chances for parallelization and pipelining, while providing the necessary support both to multimedia flows and to best-effort traffic. Performance evaluation results show that the QoS requirements of different connections are met, in spite of the presence of best-effort traffic, while achieving high link bandwidth utilizations.

Index Terms

Link/switch scheduling, Quality of Service (QoS), router architecture, multimedia transmissions

This work was partially supported by the Spanish CICYT under Grant TIC2000-1151-C07, and by the Consejería de Ciencia y Tecnología of the Junta de Comunidades de Castilla-La Mancha (regional government). The first author is in charge of correspondence.

I. Introduction and Motivation

Over the past years a rapid drop in cost of high-speed processors and mass storage media has led to the widespread use of multimedia applications. Numerous examples can be highlighted from web-based applications, streaming media, interactive simulations, visualization, virtual meeting, and collaborative design environments [1][2][3][4]. This has meant an increase both in the amounts of transmitted data, and in the sensitivity of that data to transmission delay and its variation (jitter). As a response to the new requirements imposed by workload, new mechanisms have been implemented in the Internet infrastructure in order to address the Quality of Service (QoS) needs of these applications [5][6][7][8].

Besides, the important increase in using clusters as back-end servers makes also necessary to provide QoS guarantees within these systems. But the router technologies developed for high-speed multiprocessor interconnection networks or high-performance local/system area networks (LANs/SANs) were optimized for providing low latency to best-effort traffic [9][10]. These networks are not designed to permit concurrent guarantees of communication performance to multiple applications. For example, nowadays available commercial SAN/LAN fabrics such as IBM SP2 [11], Myricom Myrinet [10], HAL Mercury [12], or Tandem ServerNet [13], are not designed to efficiently support real-time traffic. However, contemporary router technology should provide the capacity to fit the high bandwidth and timing requirements demanded by current applications.

Developing QoS-aware interconnects for high-speed system/local area network (SAN/LAN) fabrics is becoming critical. As an example, the recently proposed InfiniBand^{SM} architecture [14][15], provides some sort of QoS capabilities within its switches, although details on how these capabilities should be used are not specified. Thus, several current research efforts focus on providing proper support for the multimedia flows within the InfiniBand framework [16] [17].

As it will be detailed in Section II, the problem of providing architectural QoS support within switching elements in cluster and local area environments remains practically unsolved. Thus, the MultiMedia Router (MMR) architecture [18][19][20] arises as a solution to provide hardware-based QoS support within an interconnection element targeted for use in cluster and LAN environments. Also, conventional best-effort traffic must be seamlessly integrated in the proposed solution, and link bandwidth utilization should be maximized. Small control messages can also be generated by the applications (i.e., user commands to govern the advance of a film: rewind, fastforward, pause, . . .) or by the network itself (i.e., dynamic bandwidth management, updating of routing information, . . .). As these messages modify the behavior of the data flows, or the routers, they must be transmitted with the maximum urgency. Nevertheless, their bandwidth requirements are almost negligible, since the are usually generated very infrequently. Adequate support for these control messages must also be included into the proposed router architecture.

Efficient traffic scheduling algorithms are of key importance to fulfill these goals. These algorithms decide which data must be transmitted at each time, so their behavior will determine whether QoS guarantees are fulfilled or not.

The MMR organization is based on input queues and a multiplexed crossbar switch fabric (further details on the MMR architecture will be given in Section III). Most of the scheduling solutions appeared in the literature for such organization seek to maximize the link and switch fabric utilization [21][22][23][24], and do not address QoS issues. Some recent research on scheduling algorithms try to offer both high throughput and QoS support [25][26][27]. However, these are almost theoretical solutions. Compact and fast hardware implementations of these algorithms are hardly feasible, which prevents their use in high-speed interconnection networks. Moreover, most scheduling solutions for input buffered switches need to run at speeds higher than links to provide QoS guarantees and high link utilization, or lack the needed flexibility to concurrently accommodate different connection requirements [28].

Thus, in this paper different link and switch scheduling algorithms, suitable for the MMR architecture, as well as others alike, are proposed and evaluated. These algorithms are shown to provide high-throughput and QoS guarantees to the different multimedia flows, according to their reservations. They are also amenable for a simple low-cost hardware implementation.

The rest of the paper is organized as follows. First, Section II summarizes the related work. Next, Section III outlines the main characteristics of the Multimedia Router architecture. Then, the resource scheduling algorithms implemented in the MMR are explained in Sections IV–VI. Extensive evaluation results follow, which reveal the effectiveness of the proposed link and switch scheduling algorithms. Finally, some conclusions and final remarks are given.

II. Related Work

As it was pointed out in the previous Section, efficient router designs have been developed in the past few years for high-speed multiprocessor networks. The traditional techniques employed in this multiprocessor and multicomputer systems have also been applied into LAN/SAN systems. Examples of several commercial routers can be found, such as Myrinet [10], ServerNet [13], SGI Spider [29], or Cray T3E [30]. A common characteristic of these interconnecting elements is the use of the wormhole or virtual cut-through (VCT) switching techniques. Besides, most of them support virtual channels. However, these routers are targeted to minimize average latency and maximize the network throughput. That is, they are optimized for *best-effort traffic* transmissions. Nevertheless, widespread use of clustered systems in diverse application environments poses different communication demands from their interconnects.

The explosive increase in multimedia applications implies new requirements on LAN/SAN systems. Examples of multimedia applications include the transmission of delay-sensitive voice, bandwidth intensive video, or data applications. Voice transmission requires a small but assured amount of bandwidth, low delay, and low jitter, whereas a file transfer needs more bandwidth but can tolerate the delay and the jitter. Thus, the network must serve as a transport for a multitude of applications with different requirements. Networks must therefore provide the differing QoS requirements to offer efficient, predictable services to multimedia flows.

From the interconnection elements mentioned above, only the ServerNet switch [13] provides an arbitration mechanism, called *ALU-biasing*, for managing bandwidth allocation among the different input links. But this simple mechanism is not sufficient to support media streams. It is essential to assure that critical applications are guaranteed the network resource they need, despite varying network traffic load.

Nowadays, the growing necessity of offering hardware support to provide QoS in high-speed interconnection networks has led to some interesting proposals. Some of them will be summarized in the next paragraphs.

The *Switcherland switch* [31] includes some mechanisms to support QoS for multimedia flows. This switch was developed to interconnect memories belonging to a multiprocessor system. It uses a packet switched mechanism similar to ATM and in its design two types of traffic were considered, Constant Bit Rate (CBR) and Variable Bit Rate (VBR). While the CBR traffic class is equivalent to the one defined by the ATM Forum [32], the VBR traffic class as defined by the developers of Switcherland, corresponds to best-effort traffic, i.e., it has no bandwidth nor delay requirements. CBR traffic is given absolute priority over VBR traffic. Thus, delay bounds are not provided for this so-called VBR traffic. That is, the Switcherland architecture is only well suited to support traffic mixes composed of CBR and best-effort traffic.

Another interesting proposal is the *Yuni router* [33]. Yuni supports QoS guarantees to the applications by using input and output buffers with a different scheduler in each of them. The router dedicates several channels to implement a Time Division Multiplexing (TDM) service, with absolute guarantees of bandwidth and no jitter at all. Thus, the architecture tries to serve as a mixing point for circuit-switching networks and packet-switching networks. It is a sophisticated router architecture designed to work in platforms much more complex than LAN/SAN environments.

Lastly, the router proposal presented in [34], called the *MediaWorm router*, explores the feasibility of providing QoS by using wormhole switching for both multimedia and best-effort traffic. The MediaWorm router uses a rate-based bandwidth allocation mechanism, called Virtual Clock [35], to schedule network resources for different traffic classes. The proposed architecture is kept quite simple but, to assure QoS requirements are met, some admission control strategy must be implemented. In such case, the switching technique would no longer be wormhole switching, but it would become some sort of Pipelined Circuit Switching (PCS) [36].

The proposals of QoS-aware router architectures described above are aimed at providing QoS guarantees to multimedia applications. But it can be seen as there is no definite solution to the problem for clustered environments.

Considering the current necessities in LAN/SAN environments, the main objectives any new proposal of router architecture should address can be summarized in the following points: a) providing QoS guarantees to the multimedia flows, b) making an efficient utilization of the non-allocated bandwidth by the best-effort traffic, c) maximizing link utilization. Also, these goals should be met with a simple hardware implementation. The MMR architecture was proposed carrying these objectives in mind.

III. Overview of the Multimedia Router Architecture

The main goal pursued by the MultiMedia (MMR) project is to design a single-chip router able to efficiently handle multimedia flows and best-effort traffic. In order to achieve this goal, solutions to many difficult resource management and scheduling problems must be provided, while keeping into account that these solutions must be simple enough to permit effective single-chip implementation. In the following paragraphs, the main architectural characteristics of the MMR will be briefly described. The interested reader is referred to [18] [20] for a more detailed description.

The MMR requires the use of a *switching technique* which efficiently handles both multimedia flows and conventional best-effort traffic. Moreover, best-effort traffic must not affect the QoS received by multimedia flows. The only suitable switching techniques able to offer QoS guarantees to the data flows, are connection oriented ones. In this way, resources are allocated to the flows in the connection setup phase, and, when data are effectively transmitted, delays caused by busy resources will be bounded. On the other hand, cut-through techniques are the best suited to maximize performance for conventional best-effort traffic (high throughput, low average latency). Therefore, the MMR uses a *hybrid switching technique*, where the most suitable technique is used for each kind of traffic: a connection-oriented scheme (Pipelined Circuit Switching (PCS) [37]) for the multimedia flows, and Virtual Cut-Through (VCT) [38] for best-effort messages.

The PCS *connection establishment process* will be slightly modified in the MMR. Each time a multimedia connection establishment phase begins, the source node generates a routing probe that will try to set up a path from source to destination, reserving link bandwidth and buffer space. The probe carries information about the bandwidth requirements of the connection measured in flit cycles per round. A flit cycle is the time taken for a flit to be transmitted through the router and across the physical link. In the MMR, link bandwidth and switch port bandwidth are split into flit cycles. Flit cycles are grouped into *rounds*. The number of flit cycles in a round is an integer multiple K ($K > 1$) of the number of virtual channels per link.

Different conditions are checked in the establishment of the connection depending on the type of traffic. A CBR connection will only be accepted if the total number of flit cycles that have been allocated by all the connections using that link, including the one being set-up, does not exceed the number of flit cycles in a round. An additional test is performed when establishing a VBR connection, besides the previous one: the total peak bandwidth requested by all the VBR connections does not exceed the product of the number of flit cycles in a round and a *concurrency factor* [39]. The concurrency factor is a trade-off between the ability to make QoS guarantees, the number of connections that can be concurrently serviced, and link utilization.

The MultiMedia Router avoids losing data due to buffer overflow. This is achieved by using *per connection flow control* at the link level. For both switching schemes, PCS and VCT, the flow control unit has the same size and will be referred to as a *flit*. The selected scheme is *credit-based flow control* [40]. The main advantage of the credit-based flow control technique is that does not require large buffers even though links were long. Note that InfiniBand^{SM} has also selected this flow-control method [15]. On the other hand, its main drawback is the introduction of some control overhead. Therefore, in order to better amortize this overhead, large flits are preferred. In fact, simulation tests carried out in the MMR showed that the use of large 1024-bit flits enhances overall performance [39][20].

The general organization of the Multimedia Router is depicted in Figure 1(a). The basic building components that can be distinguished within the MMR will be described now.

a) Storage Buffers: In order to support a large number of multimedia connections, the storage buffers at each input link are organized as a set of virtual channels. For each connection, an input virtual channel is provided in order to consider the QoS of each flow. Additionally, HOL-blocking [41] is also avoided in this way. However, one important drawback is that many buffers, one per virtual channel, need to be implemented. Thus, an optimized implementation is required. Figure 1(b) illustrates the buffer structure proposed for the MMR. Buffers are organized as modules of RAM memory, interleaved with a simple scheme similar to the one used by vectorial and pipelined computers. This scheme is explained with greater details in [18].

b) Routing Unit: This unit executes the routing algorithm. The purpose of this algorithm is to compute the paths that must be followed by the connection establishment probes, and by the control and best-effort messages, in their travel through the network. Due to the fact that in PCS the probes travel independently of the data, the Exhaustive Profitable Backtracking (EPB) routing algorithm [42] is deployed when establishing connections. The EPB algorithm performs an exhaustive search of the minimal paths in the network until a valid path is found or the probe backtracks to the source node. On the other hand, best-effort messages are routed according to a fully adaptive routing algorithm [43].

The routing unit also keeps the channel mappings among input and output virtual channels for the established connections. Direct and reverse mappings are needed. The first ones are used for forwarding the data flits, and the second ones are used by the backtracking probes and the acknowledgments. These matchings are also used to propagate status information (credits, dynamic modifications in bandwidth allocation, . . .).

c) Switch Unit: One of the main concerns regarding switch organization is hardware cost both in silicon area and time delay. Thus, due to the large number of virtual channels, the MMR uses a *multiplexed crossbar* with as many ports as physical channels. This crossbar organization implies that the MMR must perform several arbitration tasks. First, arbitration is needed at the input side to select the virtual channel that will make use of the crossbar input port in the next flit cycle. A second arbitration is needed within the switch because several input channels might request the same output link for the same flit cycle. These arbitration tasks are carried out by the *link and switch scheduling algorithms*, respectively.

Arbitration can be a moderately time-consuming task, specially if the QoS needs of every flow have to be considered when computing valid schedules. However, its delay is hidden by performing scheduling in parallel with flit transmission. That is, while flits are being forwarded though the switch fabric, arbitration is concurrently being performed for the next flit cycle. This means that the time available for solving arbitration is one flit cycle. As large flits in the MMR are preferred, there is a moderately wide range of time available for the execution of link and switch scheduling algorithms. It should also be noted that the selected flits from every input link are forwarded synchronously through the MMR. This makes easier to take scheduling decisions according to the QoS needs of the applications.

d) Link Scheduler: The MMR has a link scheduling unit associated with each input link (labeled as *LS* in Figure 1(a))). In this way, link scheduling can be performed in parallel for every input port.

The purpose of the link scheduler is to solve conflicts among the virtual channels that share an input link for the use of the crossbar input ports. These conflicts must be solved taking into account the QoS requirements of the connections sharing the link. So, every link scheduler will select a small number of virtual channels, according to their QoS requirements. This is done by assigning a priority value to the head flit of every virtual channel, and selecting those virtual channels whose head flits have the highest priorities.

e) Switch Scheduler: The switch scheduler challenge is to compute switch settings to establish connections between input and output crossbar ports at speeds comparable to the time to transfer a flit through the switch. Crossbar arbitration, also called switch scheduling, algorithms must be carefully designed in order to provide QoS guarantees to the multimedia connections.

Recall that in the MMR, an efficient arbitration scheme must maximize the throughput of the network while taking into account both the priority of the traffic and the QoS delivered to each flow.

The choice of the switch scheduling algorithm, as well as the link scheduling algorithm, are critical parameters for the MMR. The authors have proposed several link and switch scheduling algorithms [44] [45] [46] [47]. A detailed description of the behavior of these resource scheduling algorithms will be given in the next Sections.

IV. Traffic Scheduling Algorithms for the MMR

QoS provisioning within the MMR architecture is addressed as providing solutions to three basic problems, namely, bandwidth reservation, link scheduling, and switch scheduling. The *bandwidth reservation scheme* used in the MMR, which acts together with the PCS connection set-up phase, has already been described in Section III. Thus, this section will focus on the other two problems, which can be globally referred to as *traffic scheduling algorithms*. They must cooperate to guarantee that the bandwidth allocated to each connection is available during data transmission. Besides, these algorithms must be well suited to parallelization and pipelining.

In order to increase chances for a parallel and pipelined implementation, the scheduling algorithms proposed for the MMR are partitioned into three basic decisions: *Candidate Selection*, *Port Ordering* and *Arbitration*.

Candidate Selection corresponds to link scheduling, and is performed in parallel by all the physical links. Every link scheduler considers the set of virtual channels in each input link and generates a set of one or more virtual channels, called *candidates*. The *candidates* are obtained as the result of some operations that allow to identify the virtual channels that should transmit a flit during the next flit cycle. More precisely, only virtual channels with flits ready for transmission and credits available are considered. Among them, those with the highest priorities are selected as candidates. The authors have proposed several ways to compute priorities [44][45]. They will be described in Section V.

Then, the switch scheduler must tackle the remaining two phases, Port Ordering and Arbitration. The purpose of the *Port Ordering* phase is to select a crossbar output port for matching, among the ones that were requested by one or more of the candidates. The *Arbitration* phase is applied in order to break conflict situations over a given output port. That is, if there are more than one requests for a given output port, only one of them is chosen. The authors have proposed two different algorithms to perform switch scheduling [46] [47]. They will be described in Section VI.

V. Solutions for Link Scheduling

The link scheduling algorithm must be executed before the switch scheduling algorithm. The link scheduler will chose a small set of candidates per input physical link. Then, the switch scheduler algorithm will work on the chosen candidates. Every link scheduler computes a *candidate vector* with the information on the selected candidates. For every candidate, at least its priority and the output port it requests are stored. These data are needed by the switch scheduler to carry out its scheduling decisions. Items in the candidate vector are sorted into *levels*, according to their priorities. The candidate with the highest priority is a first level candidate, the next belongs to the second level of candidates, and so on.

Candidates are chosen on the basis of a *biased priority scheme* [48][13][49] . Every head flit stored in a virtual channel has an associated priority value that is updated over the time, as will be described later. In each flit cycle, the selected virtual channel or *candidate* is the one whose header flit has the highest priority. In order to increase chances for port matching when switch scheduling is performed, more than one candidate per input port can be selected. The maximum number of candidates per input port is the same as the number of output ports in the crossbar.

The priority associated to every head flit is computed by considering both the QoS required by the connection and the QoS it is receiving. Priorities are biased combining the effect of the scheduler (measured as the delay or the jitter experienced by a flit in the queue) with the QoS requirements (measured as the bandwidth requested by the connection). This approach differs from others, such as the algorithm based on aging counters, in which the priority of the connections is carried out without having into account the different QoS requirements of the applications. By updating the priorities following the idea outlined before, the more QoS requirements the connection has, the faster its priority value grows. As a result, link bandwidth is distributed among connections in such a way that it depends also on the QoS requirements, rather than simply on the time spent by the packet in the network.

The MMR is designed to support not only multimedia traffic but also best-effort and control traffics. Hence, the router needs a link scheduling algorithm that gives support to all of these types of traffic. First, the best-effort traffic does not require any QoS guarantees. It is just enough to offer low latency and high throughput, but without interfering to the QoS of the multimedia traffic. Thus, flits belonging to best-effort traffic are assigned the lowest priority, and it will not be modified over the time.

On the contrary, control messages are treated with the maximum priority. Recall that these messages carry control information from applications, or from the network itself. Thus, they should get to their destination as soon as possible, because they can modify the applications or the network behavior. These messages are short and they do not appear too frequently in the network. Thus, although they do not use a high percentage of bandwidth, they must arrive at their destination as soon as possible. Therefore, the MMR assigns the maximum priority value to control messages. In this way, they will be surely chosen as first level candidates.

With the previous ideas in mind, some algorithms proposed to compute the priority are going to be described next.

A. Bandwidth-Based Algorithms

In a CBR connection, the QoS requirements are directly related to the allocated bandwidth. So, the priority of a flit can be computed as the ratio between the queuing delay, and the established Inter-Arrival Time (*IAT*) of consecutive flits in the connection. The *Inter-Arrival Biased Priority (IABP) algorithm* works using this idea. More precisely, the priority of a virtual channel head flit is computed according to Equation 1:

$$Priority = \frac{Queue\ Delay}{IAT} \tag{1}$$

The resulting behavior is that, when a flit remains in an input queue of a given router, its delay grows and therefore, the flit priority grows. Moreover, priority grows faster for those flits belonging to high-bandwidth consuming connections, that is, there are more chances that they will be forwarded sooner through the router.

This algorithm can also be applied to VBR connections. In such case, the QoS requirements of the connection can be expressed as its average value, or can be modified during the data transmission by using control words.

In the first case, the average inter-arrival time among arrivals would be used to compute the priority. This will make that some flits get assigned a higher priority than needed while flits belonging to a traffic burst get a lower one. As a result, the flow shape will be likely smoothed as it leaves the router.

On the other hand, dynamically modifying the IAT of the connection can be a useful approach when the characteristics of the generated traffic are known. This is the case when using compressed video such as MPEG-2 traffic [50]. In MPEG-2 video transmission, images (or frames) must be transmitted at regular intervals. As every image has a different size, this leads to a VBR traffic pattern, where the bandwidth required by the connection varies from image to image. If image sizes are known, the IAT can be dynamically modified for every image previously to its injection in the network.

B. Jitter-Based Algorithms

A second biasing function proposed to compute the priority is the *Jitter-Biased Priority algorithm (JBP)*. In this case, the priority is calculated as the ratio between the accumulated jitter experienced by the flits in a connection, and the inter-arrival time for the flits in the connection. The accumulated jitter is computed as the sum of all the successive flit delay differences. Equation 2 shows the operation to perform in the JBP algorithm:

$$Priority \; = \; \frac{Accumulated\,jitter + Current\,jitter}{IAT} \tag{2}$$

The goal of this algorithm is to make all the flits in a connection experience the same delay, i.e., minimize jitter. To get this objective, the algorithm reduces the priority of a given flit if it is too early to be transmitted by considering the average delay suffered by the previous flits of the connection. Once the flit has experienced a delay equal or greater than the media value, its priority grows. The more bandwidth demanding is the connection, the faster the priority of the flit grows. Finally, when the flit goes ahead through the crossbar, its delay is recorded, and the connection's accumulated jitter is updated.

C. Practical Considerations: Hardware Cost

The link scheduling algorithms presented before, the IABP and the JBP, need as many priority computing engines as virtual channels are needed in the MMR. Then, a hardware implementation block will calculate the initial priority of a connection and will also update its value. Due to the large number of virtual channels implemented in the MMR, a complex implementation of this block might make impossible to get a compact and fast design. As a result, special emphasis must be put on the cost of the hardware dedicated to compute priorities.

The implementation of the proposed link scheduling algorithms needs storing the delay suffered by every flit. This delay is directly used by the IABP algorithm, or is used to compute the jitter in the JBP algorithm. Thus, a counter to store the queuing delay of every flit is needed. At this point it can be highlighted that previous studies on the quantitative parameters of the MMR have proved that a single flit buffer per virtual channel achieves the best performance results [51] [20]. Thus, a single counter is needed per virtual channel. This counter is increased in every router cycle, and reset when a new flit enters in the input buffer. Also, another register to store the bandwidth requirements of the connection and a divider module that computes the priority, are required for the IABP algorithm. Some additional circuitry is needed to implement the jitter calculation in JBP.

To clarify the hardware requirements of the IABP and JBP algorithms for every virtual channel, Figure 2(a) and (b) present a block level diagram of one possible design for both algorithms, respectively. It can be easily seen as the most expensive elements in those hardware designs are the dividers. So, a new algorithm has been devised in order to simplify hardware cost.

As the idea behind IABP is more intuitive than JBP, and besides it has been shown to achieve a wider useful workload range than JBP [44], IABP has been taken as a starting point to devise a more practical implementation. Thus, in the *Simple IABP (SIABP) algorithm*, the priority value for each head flit is computed in a different way. The idea is to apply the same rationale introduced by the IABP algorithm, that is, to relate the bandwidth required by the connection with the experienced queuing delay, but replacing the division with some other less expensive operation.

In the SIABP algorithm the queuing delay of a flit is stored into a counter an updated every router cycle, in the same way as in the IABP algorithm. On the contrary, the priority biasing function is performed as follows. The initial value for the priority is the bandwidth required by the connection. But, instead of representing this value by the IAT of the connection, the number of slots per scheduling round reserved to service the average bandwidth of the connection is used. One advantage of this approach compared with considering the IAT is that this magnitude is an integer value. Next, the priority of the flit could be computed in a similar way to the ratio shown in Equation 1, as the product between the queuing delay and the bandwidth requirements. But, in order to achieve a simpler hardware design, the product is replaced by shifting operations. To be more precise, in the SIABP the priority value is updated by shifting to the left its current value (i.e., it is multiplied by 2), each time the queuing delay becomes greater than $1, 2, 4, \ldots 2^n$, i.e., every time a bit in the queuing delay counter is set for the first time since it was reset. In this way the QoS needed (represented by the initial priority value) is also related to the QoS received by the flit (the queuing delay). Thus, this new approach can be implemented with very low hardware complexity, by using just a shifter and some combinatorial logic to replace the divider used in IABP (see Figure 2(c)).

A VHDL description [52] of the IABP and SIABP modules was carried out. The results obtained with the Synopsys [53] design tool, using the MIETEC 0.35 μm Europractice libraries, showed that the SIABP algorithm achieves a reduction of 22 times in terms of silicon area, and of 38 times in terms of estimated delay over an integer approximation to the IABP algorithm[1]. Of course, the advantage achieved by this reduction in hardware complexity must not threaten the QoS guarantees of the multimedia flows. This will be one of the performance issues analyzed in Section VII.

VI. Solutions for Switch Scheduling

The switch scheduling algorithm challenge is to find a conflict-free matching among the input and the output ports of the switch fabric. An efficient switch scheduling scheme must maximize the throughput of the network while guaranteeing the QoS requirements of the connections.

In the Introduction to this work, it was noted that many researchers have addressed the arbitration problem over the last years [21][22][23][24]. Most of those hardware efficient solutions are aimed at maximizing crossbar utilization. Solutions that take into account some kind of QoS guarantees whether are too complex to be efficiently implemented in hardware [25][26][27], or are limited by lack of flexibility, or by scalability issues [28].

Switch scheduling algorithms in the MMR try to find a schedule that involves as many switch ports as possible. This schedule will be used later to configure the crossbar for synchronous flit forwarding, The clue to get a successfully matching is to have into account both the number of conflicts for output ports and the priorities computed by the link scheduler, in order to maximize switch throughput and guarantee QoS requirements of multimedia flows. Hence, two critical decisions must be taken by the switch scheduling algorithm. The first one is the order in which the

[1]Note that IABP was originally formulated to work with floating point magnitudes, i.e., the IAT of the connection.

output ports are considered, and the second one is the arbitration scheme to apply when candidates from several different input ports compete for the use of the same output port.

The *port ordering function* is important because once an input port is connected to an output port, the other requests from the same input port are dropped. This can modify the number of conflicts at other ports, and affect the order in which the ports are considered, consequently impacting final switch utilization. On the other hand, an *arbitration function* must be defined such that in case of conflicts for an output port, the flit with the highest priority is chosen. Priorities computed by link schedulers are re-used.

As it was described in previous Sections, *candidate selection* is performed during link scheduling. Every link scheduler computes a candidate vector, that stores the information regarding the flits with the highest priorities from every input link. The switch scheduler takes these candidate vectors as inputs to compute a conflict-free matching among the crossbar input and output ports.

In this Section, two switch scheduling algorithms designed by considering the QoS parameters of the connections will be described in detail. These algorithms are named *Candidate-Order Arbiter (COA)* and *Candidate-Conflict Arbiter (CCA)*. They differ in the criterion used to perform the port ordering phase.

A. The Candidate-Order Arbiter (COA)

The first algorithm proposed for solving the switch scheduling problem in the MMR is the Candidate-Order Arbiter (COA). The criterion used to perform the Port Ordering function is to choose output ports first by level, and then, in increasing order of conflicts within a level. Ties are broken by randomly selecting one of the ports. The rationale is that ports with the most conflicts should be matched last since those ports have the most opportunities to be matched to an input port. Besides, preference should be given to the candidates from lower levels when computing matchings, because they are the ones with higher priorities among the candidates.

In order to apply this criterion, the information from the candidate vectors is conveniently organized. The candidates are ordered by levels, according to their priority values. The information related to all the candidates from every input port is arranged into a *selection matrix*, which has $N \times C$ rows, and N columns. The first N rows store the requests made by the highest priority candidate for every input link, that is, the level one candidate requests. The next N rows store the level two candidate requests, and so on. Also, a *conflict vector* is computed. This vector has

$N \times C$ items, and stores the number of non-null entries on every row of the selection matrix. That is, the conflict vector identifies the number of conflicts that every output link has at every candidate level. Figure 3(a) shows an example selection matrix considering two levels of candidates, for a 4 port crossbar. The rows and the columns are labeled with the corresponding output and input port, respectively. Note that the first 4 rows record the highest priority candidates for each input while the next 4 rows record the level two candidates. For example, it can be seen as the candidate with the highest priority from input port 0 is requesting output port 0. This candidate is one of the Level 1 candidates. The second candidate with the highest priority (i.e., a Level 2 candidate) from the same input port is directed to output port 1. On the right hand side of Figure 3(a), the corresponding conflict vector has also been depicted.

Once an output port has been selected for matching as a result of the port ordering phase, if there are several requests for it, one of them must be chosen. This is done by applying the *Arbitration* phase. The criterion to perform this is to select the candidate with the highest priority.

Each time an input port and an output port are matched, all the requests involving those ports are dropped, and the selection matrix and conflict vector are re-computed.

Figures 3(b)-(d) show an example of the execution of the COA algorithm, for a 4×4 crossbar with 2 levels of candidates. Priorities for the candidates are shown, in case they are needed in the arbitration phase. When applying the port ordering function for the first time (Figure 3(b)), it can be seen as there is a tie between output ports 0 and 3. Port 3 is randomly selected. As there are two input ports with candidates requesting output port 3 the arbitration phase must be applied. As a result, input port 1 is chosen, and this leads to the first match between output port 3 and input port 1. The selection matrix is updated by clearing all the other requests involving these ports, and also the conflict vector is consequently recomputed. Now, as there are still pending requests among the candidates of the first level, the port ordering function and arbitration phases are applied again. They lead to the second match, between input port 0 and output port 0. Now, no more matches can be found among level 1 candidates, so the process moves on to consider level 2 candidates. In this way, one additional match is achieved: output port 2 with input port 3. After this, there are no more pending requests, and no more levels of candidates, so the scheduling process ends. Figure 3(e) depicts the final matching. Note that only 3 ports out of 4 are scheduled for transmitting flits. If more levels of candidates were considered, perhaps the remaining ports would have been also matched, thus increasing average crossbar throughput. But increasing the number of levels of candidates would also increase the complexity of the scheduler. Thus, a trade-off must be met.

B. The Candidate-Conflict Arbiter (CCA)

As well as COA, the CCA algorithm also tackles the last two phases of traffic scheduling, i.e., port ordering and arbitration. While the arbitration phase is performed in the same way than in the COA algorithm, the CCA approach differs in the criterion used to compute the port ordering phase. The ordering function used in CCA selects output ports in increasing order of conflicts. As in COA, ties are broken by randomly selecting one of the ports.

This algorithm differs from the previous one in the fact that all the selected candidates are considered at the same time, instead of splitting them into levels. Thus, the information from all the levels is recorded together, and thus conflicts for output ports refer to the total number of candidates from all the levels, requesting every output port. Consequently, in this case, the *selection matrix* has N rows, and N columns, and each column stores all the requests made by one input physical link. The *conflict vector* is also computed from the selection matrix. Accordingly to its dimensions, the conflict vector has N items, and stores the number of non-null entries on every row of the selection matrix. One example selection matrix and its corresponding conflict vector is depicted in Figure 4(a). This matrix holds the same information as the matrix shown in Figure 3(a). Note that in this case, requests from both levels of candidates are mixed in a smaller matrix. Requests from level 1 candidates are labeled as '`X1`', and the ones from level 2 candidates are labeled as '`X2`'.

To clarify the behavior of this algorithm, let us consider the following example. Figure 4(b) shows the selection matrix considering two levels of candidates, for a 4 port crossbar, where priorities have been recorded for each candidate. The information recorded in this matrix corresponds to the same example shown in Figure 3 for the COA algorithm. Note that it might occur that there are more than one candidate from the same input port, requesting the same output port. In that case, only the priority of the candidate from the lowest level would be recorded. The rows and the columns are labeled with the corresponding output and input port, respectively. Figures 4(b)-(e) illustrate the execution of the algorithm, where every step involves the application of the port ordering phase, and, in some cases, the arbitration phase too. It can be seen as, except for the way the information from candidate vectors is organized, this algorithm takes decisions in a similar way as COA does.

The matching obtained at the end of the scheduling process is shown in Figure 4(f). Note that in this case, the application of the CCA algorithm has led to a full crossbar utilization, i.e., all the ports have been scheduled to transmit one flit in the next cycle, and there are no idle ports. However, the fact of considering all the levels of candidates at once might decrease the opportunities for a pipelined operation, since in this case the extraction of information from the candidate vectors cannot be made while the scheduling algorithm is processing the candidates from the previous level.

A more detailed discussion of the hardware requirements for both COA and CCA algorithms can be found in [19]. In that work, a block-level hardware implementation of the COA and CCA algorithms is carried out, and chances for a parallel and pipelined implementation are identified, in order to estimate the feasibility of a hardware implementation of these algorithms. Estimations show that only 2 or 3 router cycles are needed to compute switch settings in the worst case (i.e., with the maximum number of levels of candidates), respectively, for a 4×4 router with 1.24 Gbps 16-bit wide links (i.e., the router cycle is 12.9 nanoseconds).

Estimations on the candidate selection phase, which involves sorting a moderately high number of 16-bit values, show that 8 more cycles should be added to the delay introduced by the switch scheduler, when the SIABP priority biasing function is used. These results lead to a minimum flit size of 160–176 bits to permit concurrent operation of the traffic scheduler with flit transmission. As the use of larger flits (as long as 1024 bits) have been shown to improve the MMR performance [39][20], it is clear that there is time enough for the execution of the proposed traffic scheduling algorithms.

VII. Performance Analysis

Now it is time to assess whether the objectives that should be addressed by the link and switch scheduling algorithms are fulfilled. On that purpose, a C++ discrete event simulator has been developed. This tool will be described next, before presenting the results obtained from extensive simulation tests. The traffic models used in simulations will also be described in Section VII-B.

A. Simulation Framework

The simulation tool models with great detail the behavior of one single MMR with one Network Interface Card (NIC) attached to every input/output port. This will allow to focus the analysis on the behavior of the proposed schedulers, independently of other matters, as routing or network topology.

All the experiments presented here have been carried out for a single 4×4 router, with full-duplex 1.24 Gbps 16 bit-wide links. This gives a router cycle of 12.9 nanoseconds. The number of virtual channels per input link is 128. Flits are 1024 bit long. The MMR buffers have capacity to store one flit per virtual channel. And finally, the scheduling round size, determined by the K parameter, has been been fixed with $K = 16$ [20].

Input workload is expressed as a percentage of the bandwidth. Simulated percentages range from medium to high loads. Several traffic models have been used in the simulations: multimedia, best-effort, and control traffic. They will be described in the next Section. Multimedia workload is specified as a list connection requirements stored in an external file, that must be read at the beginning of the simulation. Before starting flit injection, the bandwidth reservation tests described in Section III are checked. In this process, a connection can be rejected if it fails the corresponding test(s). Once the connection setup phase has ended, traffic sources begin to inject their data flits. Best-effort workload is defined by specifying a number of best-effort traffic sources. When control traffic is included, one traffic source of this type per NIC is configured.

Traffic sources inject their flits into buffers located in the corresponding NIC. These buffers are considered to be infinite because the host main memory can also be used if the NIC buffers become full.

The physical link controller located in the NIC forwards the flits belonging to the different flows in a demand-driven round-robin fashion, i.e., it performs round-robin among all the connections with both flits and credits available.

No statistical information is gathered until some scheduling rounds have been completed in order to get data only when the system is stable. Simulations have been carried out for long enough to record significant data of the MMR state.

Due to space limitations, only the most significant results are presented in this paper. More extensive evaluation results can be found in [19].

B. Traffic Models

Several traffic types have been considered in simulations. These include multimedia connections (both CBR and VBR), best-effort traffic, and also control traffic. For all of them, destination ports have been randomly selected from uniform distribution.

- **Multimedia Traffic**: The *CBR traffic model* is composed of a mix of synthetic connections with 64 Kbps, 1.54 Mbps and 55 Mbps average bandwidth. These are representative of several applications such as audio, video, and high-definition video transmission, respectively. Besides, they impose a wide range of different bandwidth requirements to the MMR.
  The *VBR traffic models* are based on video traces, obtained from real MPEG-2 video sequences. This is a typical type of multimedia flow. The MPEG-2 video coding standard [50] encodes the video streams as a sequence of different frame types, I, P, and B, ordered with a predefined and repetitive pattern, called GOP (Group Of Pictures). I frames encode independent frames, that is, I frames do not need any extra information to be decoded. P frames need the previous I frames in the sequence to be decoded, because the data they hold is related to that on the I frame. Finally, B frames need information from both previous and following P or I frames to be decoded. The bandwidth needed for each type of frame is different. I frames are the most bandwidth demanding, and the B frames are the lest consuming . All the flits belonging to a frame have to be generated every 33 milliseconds. Flits can be injected in different ways (see [20] for more details). In the results presented here, the flits belonging to the same frame are evenly distributed over the 33 milliseconds. This leads to a flit Inter-Arrival Time (IAT) which varies from frame to frame.
- **Best-effort Traffic**: Multimedia traffic may coexist with best-effort traffic, such as the data generated by scientific applications, and other applications such as web traffic. This is the type of traffic that is willing to make use of any available bandwidth on the network. The generator of best-effort traffic is based on SURGE (Scalable URL Reference Generator) [54]. SURGE is a realistic web workload generator that shows self-similar properties. It mimics a set of real users accessing a server. The MMR assigns the flits of this class of traffic the lowest priority values.
- **Control Traffic**: Control messages carry information used in network management, such as, network configuration and reconfiguration, congestion control, bandwidth management, and

so on. They are also used for carrying application-level control information, thus controlling the behavior of the multimedia connections. They must be transmitted as soon as possible, and the MMR assigns the highest priority to its flits. Since flits are large, a control message is composed of just one flit. The generation rate of this traffic is quite low, one message is generated approximately every 33 msecs [34], with a tolerance margin of $\pm 10\%$.

In a similar way as InfiniBand does, virtual channel 0 has been reserved for control traffic. Virtual channels 1 to 4 are reserved for best-effort traffic, one virtual channel per output port. Best-effort flits are enqueued in these virtual channels following a Virtual Output Queuing scheme. The remaining virtual channels can be used by multimedia flows, both CBR and VBR traffic.

C. Link Scheduling Evaluation

Evaluation results regarding the performance of the proposed link scheduling algorithms will be presented now. All the tests carried out in this Section have been done with the COA switch scheduler. Only the link scheduling algorithm, the number of candidates and the workload have been varied.

First of all, the influence of the number of candidates generated by link schedulers is analyzed for the IABP and JBP algorithms. Results are presented in Figure 5(a)-(f), for both CBR and VBR workloads. Average crossbar utilization is a measure of the amount of traffic the router is able to handle. The plots in Figure 5(a)-(c) show, for both CBR and VBR traffic and for the IABP and JBP algorithms [2], that an increase in the number of candidates produces an increase in the amount of workload admitted by the router before entering saturation. Nevertheless, as the number of candidates is increased, diminishing returns are obtained. It can also be noticed as the crossbar is very poorly utilized when using only one level of candidates. As in this case there are little chances for every input port to get matched, performance is degraded, specially when using the JBP algorithm and CBR traffic.

In order to know whether the multimedia flows are receiving the requested QoS, some more results are needed. For CBR traffic, the plots shown in Figure 5(d)-(e) depict the distribution of flit delay for the most demanding CBR connections, for one of the highest workload levels simulated (81% of link utilization). This distribution is computed by comparing the delay suffered by each flit with a set of thresholds related to the IAT of the connection. In this way, thresholds are related

[2] The plot for VBR traffic and JBP is not shown because it is almost identical to that with IABP

to the QoS needs of the connection. It can be seen that, at this load, the four possible levels of candidates are needed when using the IABP algorithm to obtain reasonable deadlines for all the flits. In that case, all the flits suffer delays lower than twice their IAT (IAT = 19 microsecs). However, if the JBP algorithm is used, even when using four levels of candidates, there is still a portion of flits that is not able to fulfill large deadlines. For example, 10% of the generated flits suffer delays higher than $8 \times IAT$, i.e., 151.4 microsecs. Of course, at lower workload levels, tighter deadlines can be met when using less candidates per input link.

Regarding VBR traffic, its performance metrics are related to the application data units, i.e., video frames. Thus, Figure 5(f) shows the distribution of frames delays since generation when the IABP algorithm is used, at a workload level of 84% of link utilization. This distribution is computed in a similar way to CBR traffic, but this time thresholds are related to the frame time (33 millisecs). Again, four candidate levels are needed to achieve reasonable deadlines. In that case, all the frames suffer delays lower than 132 millisecs. Note that a common delay bound used in MPEG-2 video transmissions is one second between endpoints [55], so this bound leaves place for some more hops within the network. When a lower number of candidates is used, there is a moderate percentage of frames (up to 10%) that cannot traverse the router due to saturation. This means that this workload level should not be reached with this router configuration, because QoS cannot be guaranteed.

From all the analysis above, it can be concluded that the router with IABP is able to achieve higher link utilizations, while providing QoS to multimedia flows, than when using JBP. However, as JBP is aimed at providing low jitter, this metric should also be analyzed (only results for 4 levels of candidates will be considered now). Regarding the jitter suffered by CBR flits, JBP effectively enhances its value for the lower bandwidth connections (64 Kbps) (see Figure 5(g)), while similar results are obtained for the most consuming ones before saturation is reached (see Figure 5(h)). On the contrary, frame jitter, i.e., the variation in delay experienced by the VBR connections data unit (Figure 5(i)), is quite similar for IABP and JBP before saturation occurs. The reason is that JBP tries to minimize *flit* jitter, and this does not impact the jitter suffered by the full frame.

As a conclusion, when the JBP algorithm is used, the jitter suffered by CBR flits is improved, at the cost of a narrower useful workload range. Besides, with any algorithm at least two candidates must be generated per input port to reach reasonable link utilizations. Best results, i.e., the widest useful workload range, are obtained when using the IABP algorithm with 4 levels of candidates.

As it was explained in Section V-C, the implementation of the IABP algorithm needs a floating point divider, which dramatically increases hardware costs. Thus, a more hardware-effective version of the IABP algorithm was devised. This is the SIABP algorithm. Now, it is the turn for evaluating how the hardware reductions introduced by SIABP impact the performance of the multimedia flows. The plots in Figure 5(j)-(k) show the distribution of flit delay since generation for the lowest and highest bandwidth consuming connections, respectively, for a workload level of 84%. It can be seen as hardly any difference can be found between the performance of the original algorithm and the simplified one. The distribution of VBR frame delay is shown in Figure 5(l). In this case, the maximum delay achieved with the SIABP algorithm is higher that with IABP, although there is a significantly higher amount of frames (around 60%) that achieve delays lower than the tightest threshold. These results show that the useful workload range is similar for both algorithms, IABP and SIABP. Hence, the hardware reductions will greatly improve the feasibility of a hardware implementation of the MMR, without degrading performance.

Lastly, and for reference purposes, the performance obtained when using the practical SIABP algorithm is compared to that obtained when introducing a classical algorithm acting as a link scheduler. The widely known Virtual-Clock (VC) [35] algorithm has been chosen for this purpose. As well as SIABP does, VC algorithm also assigns priorities to flits according to the bandwidth requirements of the connection they belong to. Most significant results are achieved with CBR traffic. The plots in Figure 5(m)-(n) show the distribution of flit delay for a workload level of 84%, for the CBR connections with the lowest and the highest requirements, respectively. It can be seen as, while little differences are found for the 55 Mbps connections, the flits from the 64 Kbps ones suffer much greater delays when VC is used. Also, the jitter suffered by these flits is negatively impacted when using VC, as can be seen in Figure 5(o). The conclusion is that when workload is composed of connections with very different bandwidth requirements, the use of the VC algorithm to compute flit priorities degrades performance of the connections with less strict requirements. This fact limits the useful workload range achieved when using the VC algorithm as link scheduler in the MMR.

D. Switch Scheduling Evaluation

All the tests presented in this section have been carried out with the SIABP link scheduler, with four levels of candidates. Thus, a certain level of QoS provisioning is already provided. It is now necessary to assess whether QoS support is also needed within the switch scheduler to offer QoS guarantees to the multimedia flows.

In order to calibrate both the COA and CCA approaches for the switch scheduling algorithm, they are compared to a well-known conventional symmetric crossbar arbiter called the Wave Front Arbiter (WFA) [22], with no QoS support.

WFA is based on the propagation of an arbitration “wave” across an array of arbitration cells, one cell per crosspoint. At every crosspoint, it is checked whether the corresponding input or output port have already been matched. If they are still unmatched, and there is a request involving the corresponding input and output ports, they are matched. In [22] it is shown that WFA is able to achieve nearly the same performance as more complex theoretical schemes, which cannot be practically implemented. Moreover, the analysis in [56] shows that WFA performs better than other scheduling algorithms such as PIM [21] or iSLIP [24], under uniform and non-uniform traffic patterns, at a lower hardware cost. Because of these reasons, WFA has been chosen for comparison with the proposed switch scheduling algorithms.

The plots shown in Figure 6(a) and (b) correspond to the average crossbar utilization obtained with the three switch schedulers, for CBR and VBR traffic, respectively. It can be seen as the worst performance in terms of amount of forwarded data is achieved by WFA. When this algorithm is used, the MMR is only able to reach 75% of link bandwidth utilization (and even less for VBR traffic), that is, around 15–20% lower than the other two proposals. It can also be observed as CCA offers better results than COA, specially when workload is composed of VBR traffic. These results suggest that a QoS-aware switch arbiter is necessary in order to better utilize link and switch bandwidth. Note that as WFA does not take into account the QoS requirements of the connections when computing matchings, it may schedule for transmission flits with lower priority instead of others with higher priority. Consider the case of a connection with high bandwidth requirements. If its flits are not scheduled in time, because its QoS requirements are not being considered by the switch scheduler, its flits eventually will start to enqueue at the MMR buffers and within the network interface card (NIC). This impacts negatively the amount of workload forwarded by the router.

Some measures of the QoS received by the applications are plot in Figure 6(c)-(d) for CBR traffic, and in Figure 6(e)-(f) for VBR traffic. The plots show the average flit or frame delay since generation for two workload levels close to saturation. Regarding CBR traffic, results are shown for the most demanding connections. It can be seen as a workload level of 81% of link utilization can be reached, while still guaranteeing reasonable deadlines to the connections, only if a QoS-aware switch scheduler (COA or CCA) is used. If the WFA scheduler is used, around 10% of the generated flits are not even able to traverse the router. When workload is increased up to 87%, only the use of CCA permits offering QoS guarantees to the CBR connections. When COA is used, the router enters saturation at this workload level. Results for VBR frames exhibit a similar behavior. As before, when WFA is used, there are frames that are not forwarded by the router, and remain at the corresponding NIC buffers. CCA also offers better performance than COA, and is able to reach higher utilization levels before entering saturation.

The conclusion that can be drawn from this study is that a QoS-aware switch scheduler is necessary in order to make an efficient use of link and switch bandwidth, while guaranteeing the needed performance to the applications. Both COA and CCA beat WFA in terms of the achieved useful workload range. Besides, CCA is able to support slightly higher link bandwidth utilizations than COA before entering saturation, while still providing an adequate service to the applications.

E. Global Evaluation

In order to perform a global evaluation of the proposed architecture and algorithms, results mixing different traffic classes will be presented in this Section. More precisely, control and best-effort traffic sources will be introduced together with multimedia traffic sources, and overall performance will be analyzed.

Workload has been generated in a different way for the tests carried out in this Section. Now, a certain percentage of link bandwidth will be filled with a mix of multimedia connections, both CBR and VBR, and increasing amounts of best-effort traffic will be introduced, up to filling the whole link bandwidth. There will also be a control traffic source per NIC, but its bandwidth use is negligible. This study is aimed at assessing several issues. First of all, the QoS guarantees the applications get must not be affected by the presence of best-effort traffic. Second, control traffic must be forwarded with the minimum delay, independently of the workload level. And last, best-effort traffic must use the remaining bandwidth not used by multimedia connections.

The plots in Figure 7(a)-(b) show the average crossbar utilization, for both the COA and CCA switch schedulers. Utilization due to multimedia connections is depicted separately, as an horizontal line. The difference between this line, and the line corresponding to the total utilization corresponds to best-effort traffic. As it was shown in the previous Section, CCA achieves a slightly higher crossbar utilization than COA.

Now, the QoS received by the multimedia connections must be assessed in the presence of best-effort traffic. As the results for lower demanding connections showed no impact when including best-effort traffic, only the plots of the distribution of flit delay since generation for the most demanding CBR connections, are shown in Figures 7(c)-(d). Workload is the highest simulated level, near 100% of link utilization. For reference purposes, also the curve corresponding to the same multimedia workload, but without best-effort traffic, is shown, labeled as (ref). It can be noticed a slight increase in the number of flits that fail to fulfill the tightest deadlines when best-effort traffic is present. This is due to the fact that, although best-effort flits cannot be selected as candidates by the link scheduler before any multimedia flit, they can affect the number of conflicts per output port, thus modifying the computed matching. This effect shows up as an increase in the delay suffered by some flits from the most sensitive applications, i.e., the most demanding ones. This impact is less noticeable when the COA switch scheduler is used, because in this case, ports are first considered by level of candidates, and then, by number of conflicts. Thus, the effect of the presence of best-effort flits is softened. Nevertheless, the minimum deadline fulfilled by all these CBR flits is the same than when there is no best-effort traffic, both for the COA and the CCA schedulers. On the other hand, performance of VBR video connections is not altered when introducing best-effort traffic, as can be inferred from the distribution of frame delay shown in Figure 7(e). From all the above, it follows that the the proposed traffic scheduling algorithms, within the framework of the MMR architecture, are able to fulfill the requirements of the multimedia connections in spite of the presence of best-effort traffic.

Control flits must reach their destination as soon as possible. In order to achieve this, they are assigned the maximum priority in link scheduling, so that they are always selected as first level candidates. As the proposed switch schedulers make its first decision by considering number of conflicts, this could lead to select an output port for matching different than one requested by a control flit. This is solved by matching ports with pending control flits previously to the execution of the switch scheduler. As control flits are generated with a very low rate, approximately one flit every 33 milliseconds, this does not impact the performance of multimedia connections. As a result, the *maximum delay* suffered by control flits is not affected by the presence of other types of traffic, regardless of the switch scheduling algorithm used, as can be seen in Figure 7(f).

Finally, the performance achieved by best-effort traffic is depicted in Figure 7(g), as its average flit delay since generation. It can be noticed as saturation occurs at around 85% of link bandwidth utilization, for both switch scheduling algorithms. This means that best-effort traffic is effectively taking advantage of the link bandwidth not used by multimedia applications.

VIII. Conclusions and Final Remarks

Multimedia applications have become of widespread use in the last few years, due to the availability of low cost, high performance personal computers. Many of these applications are distributed in nature, or require the computational resources of a cluster of workstations. In any case, a new need for QoS support within cluster and local environments has been born. Within this context, the MultiMedia Router architecture arises as a solution for providing efficient hardware-based QoS support for the new applications, as well as including support for conventional, best-effort traffic.

One of the key elements in fulfilling the goals pursued by the MMR architecture is the traffic scheduling algorithm. The work presented here has been focused on providing several traffic scheduling solutions within the MMR architecture framework. More precisely, the traffic scheduling problem has been split into two parts, namely, link scheduling and switch scheduling. For both of them, several different algorithms have been described, all of them being suitable for a parallel and pipelined operation. Evaluation results show that, within the MMR architecture, the proposed algorithms are able to provide the necessary QoS required by nowadays multimedia applications.

Regarding link scheduling, two approximations to the problem of assigning priorities to the flits have been presented, as well as some practical considerations regarding their hardware implementation. These considerations have led to the proposal of a improved hardware-efficient link scheduler which has been shown to provide the needed QoS to multimedia flows with a wide range of bandwidth requirements.

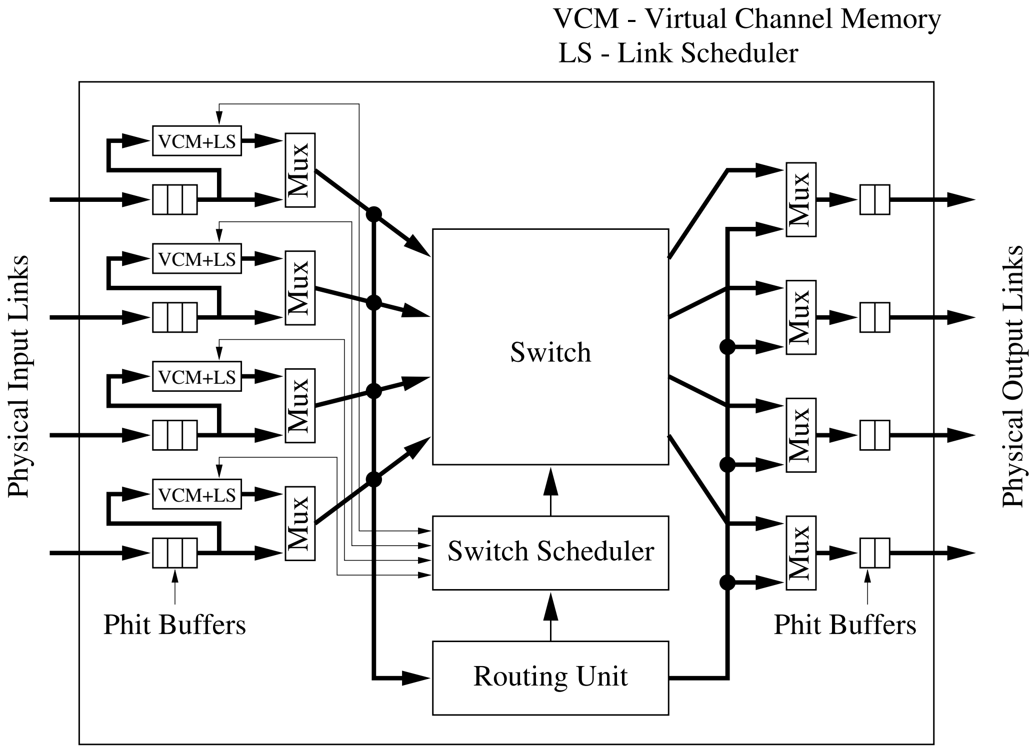
Regarding switch scheduling, two algorithms aimed at solving the crossbar port matching problem have been presented. Their main distinguishing features is that they take into account the QoS needs of the multimedia flows when computing matchings, and that they are amenable for a hardware-efficient implementation. It has been highlighted that the fact of considering QoS in the switch scheduling process is of key importance in achieving high link utilizations while still providing the necessary QoS to the applications, in addition to the use of a proper link scheduling algorithm.

Finally, it has been assessed that the performance obtained by the multimedia flows is not altered by their coexistence with best-effort traffic, while this type of traffic is able to efficiently use the remaining link bandwidth not used by multimedia applications.

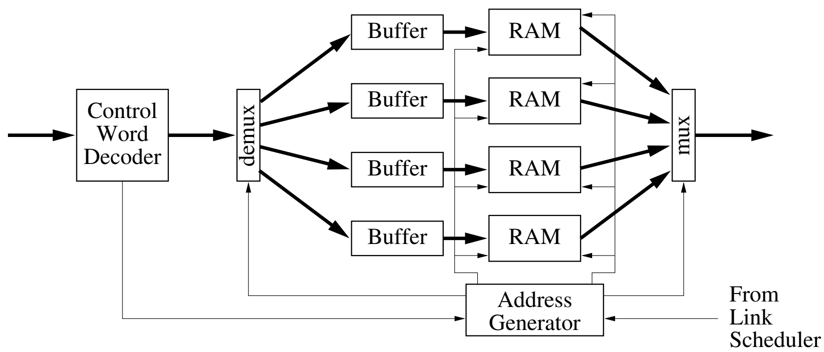
As a final consideration, it should be noted that the algorithms presented here can be easily adapted to other router architectures, with organizations similar to that of the MMR's, i.e., based on input buffers and a multiplexed crossbar.

Appendix

Figures

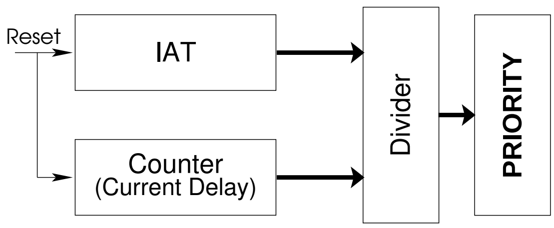


(a) Router organization

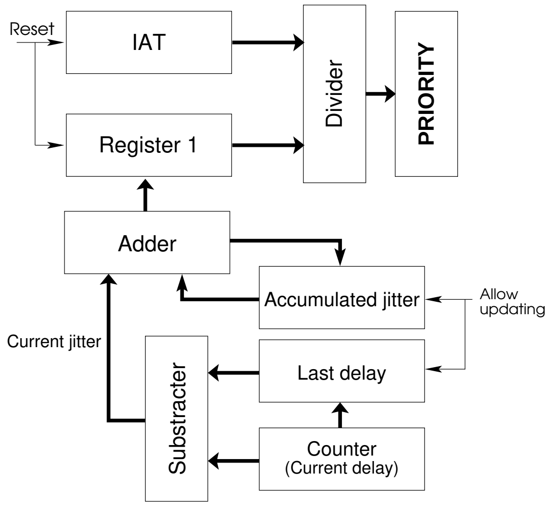


(b) Virtual channel memory organization

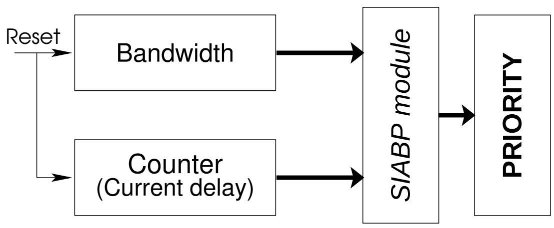
Fig. 1. MultiMedia Router architecture.



(a) IABP algorithm



(b) JBP algorithm



(c) SIABP algorithm

Fig. 2. Block diagrams for the priority computing engines of the different link scheduling algorithms.

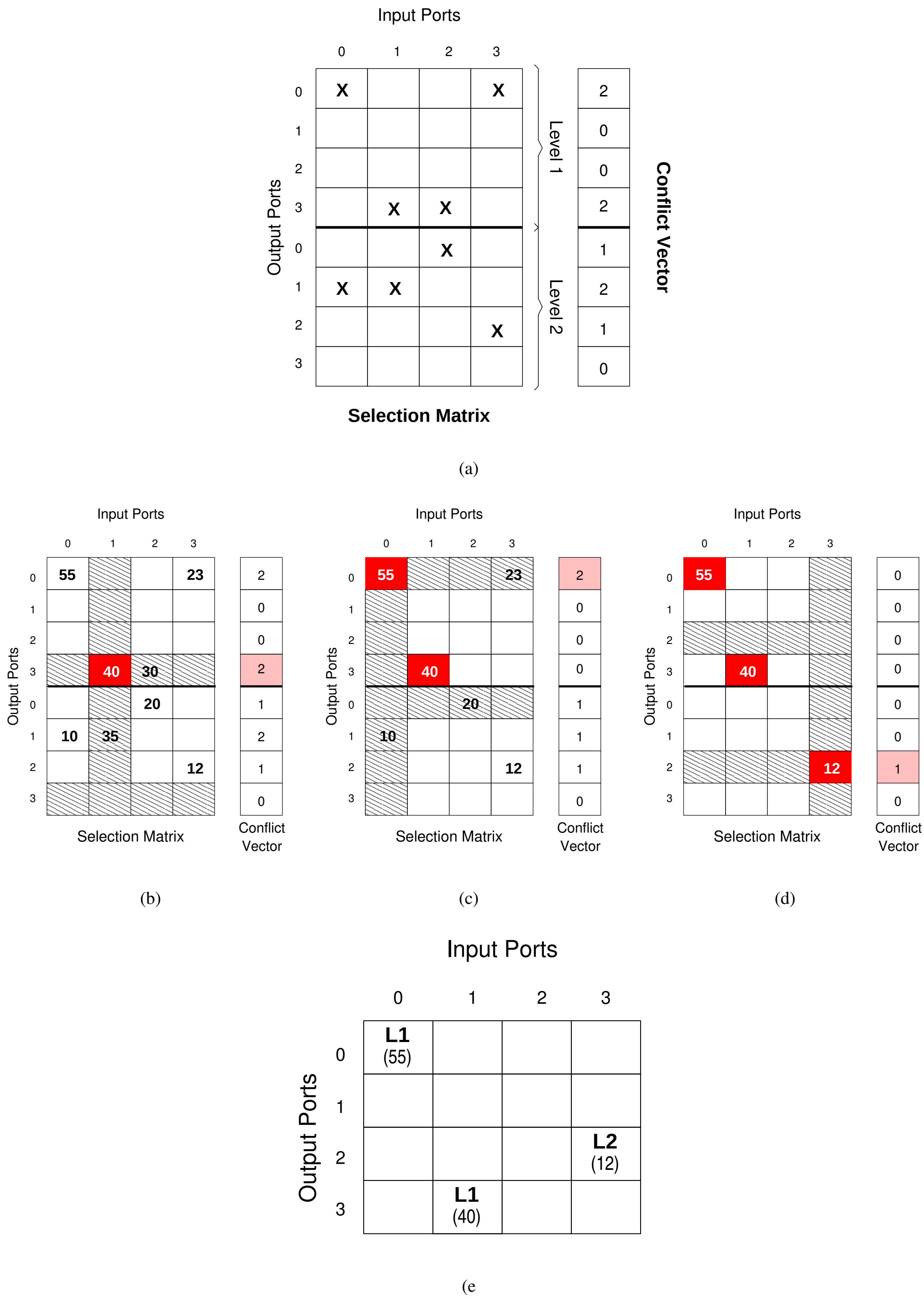


Fig. 3. Example of the COA algorithm.

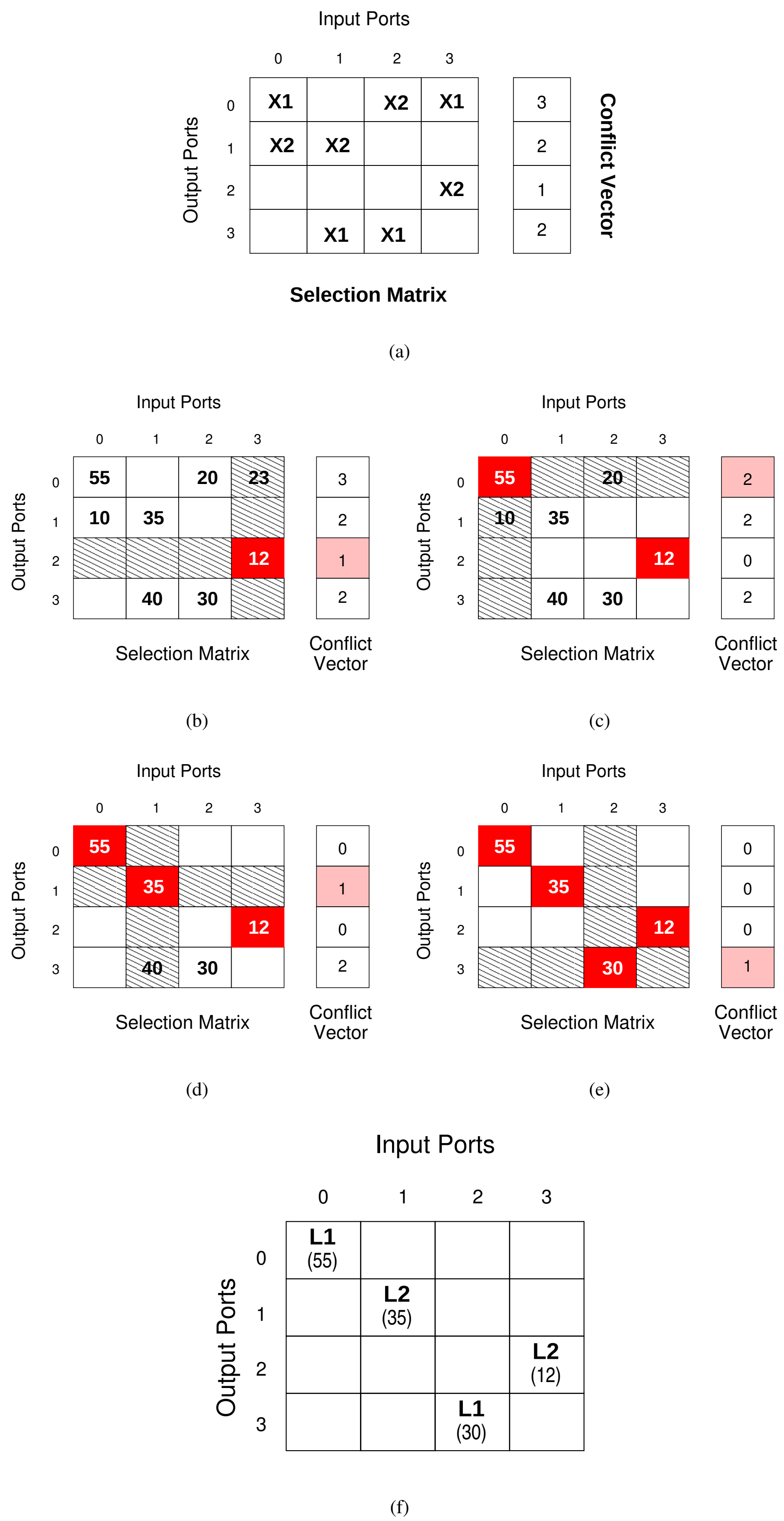
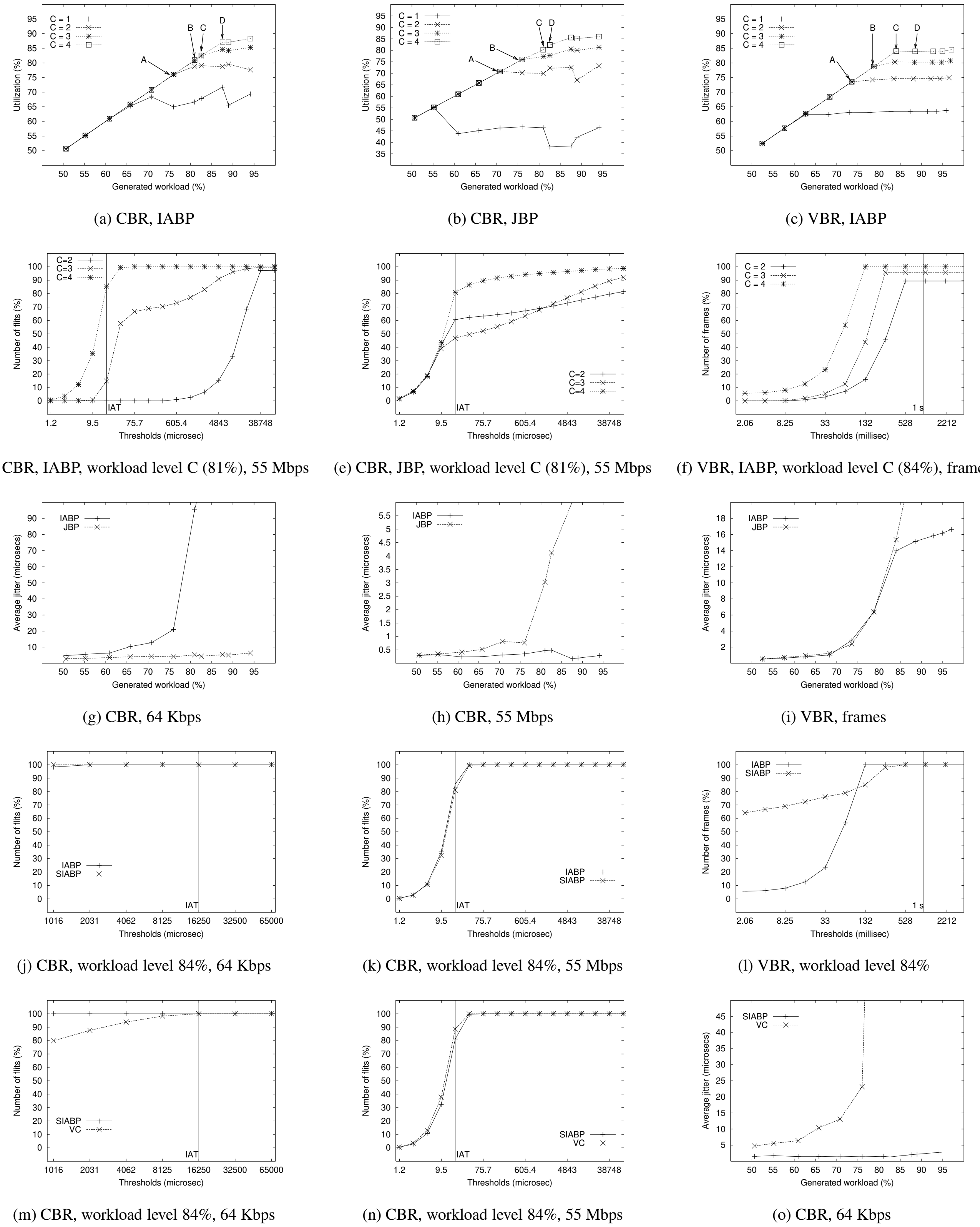
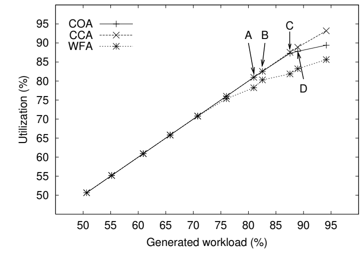


Fig. 4. Example of the CCA algorithm.

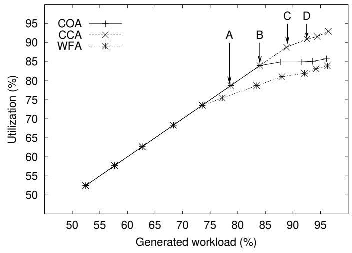


(a) CBR, IABP
(b) CBR, JBP
(c) VBR, IABP
(d) CBR, IABP, workload level C (81%), 55 Mbps
(e) CBR, JBP, workload level C (81%), 55 Mbps
(f) VBR, IABP, workload level C (84%), frames
(g) CBR, 64 Kbps
(h) CBR, 55 Mbps
(i) VBR, frames
(j) CBR, workload level 84%, 64 Kbps
(k) CBR, workload level 84%, 55 Mbps
(l) VBR, workload level 84%
(m) CBR, workload level 84%, 64 Kbps
(n) CBR, workload level 84%, 55 Mbps
(o) CBR, 64 Kbps

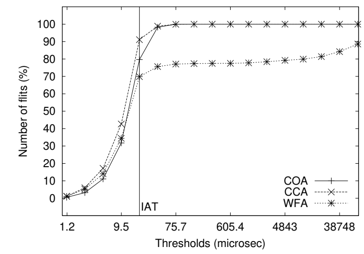
Fig. 5. Link scheduling performance results: Average crossbar utilization for IABP and JBP ((a)-(c)), distribution of CBR flit delay or VBR frame delay since generation for IABP and JBP ((d)-(f)), average CBR flit jitter or VBR frame jitter for IABP and JBP ((g)-(i)), distribution of CBR flit and VBR frame delay since generation for IABP and SIABP ((j)-(l)), distribution of CBR flit delay since generation for SIABP and VC ((m)-(n)), and average CBR flit jitter for SIABP and VC (o).



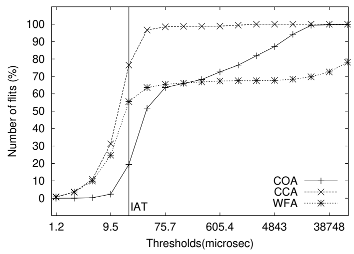
(a) CBR



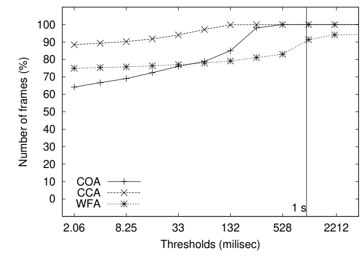
(b) VBR



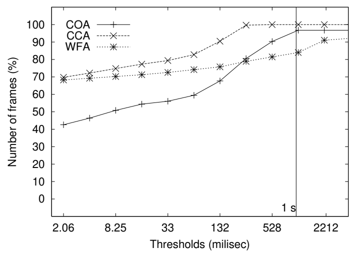
(c) CBR, workload level A (81%), 55 Mbps



(d) CBR, workload level C (87%), 55 Mbps

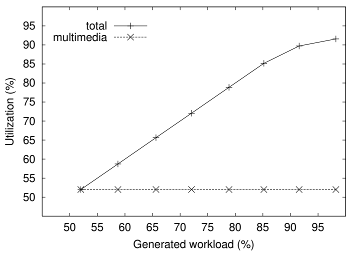


(e) VBR, workload level B (84%), frames

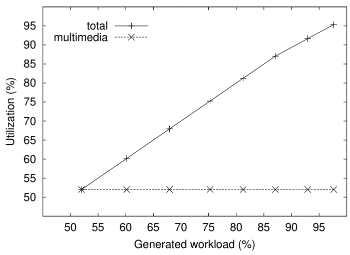


(f) VBR, workload level C (89%), frames

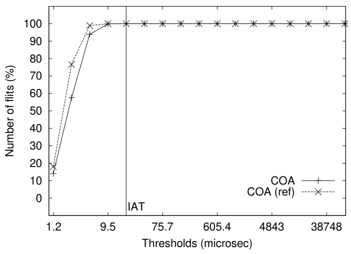
Fig. 6. Switch scheduling performance results: Average crossbar utilization ((a)-(b)), distribution of flit delay since generation for CBR traffic ((c)-(d)), and distribution of frame delay since generation for VBR traffic ((e)-(f)).



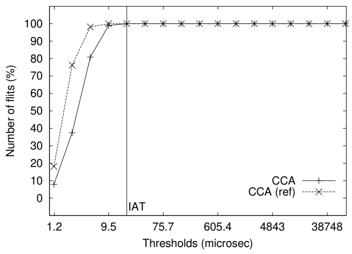
(a) COA



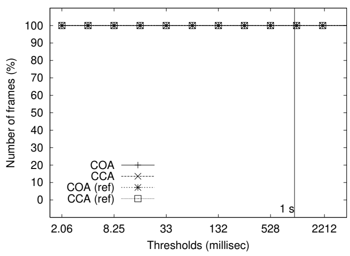
(b) CCA



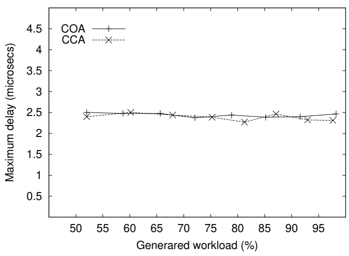
(c) COA, maximum workload, CBR, 55 Mbps



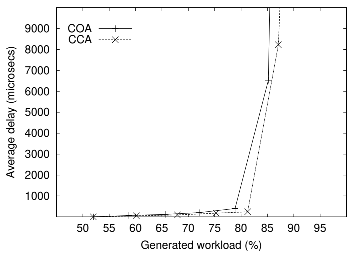
(d) CCA, maximum workload, CBR, 55 Mbps



(e) Maximum workload, VBR, frames



(f) Maximum delay of control flits



(g) Average delay of best-effort flits

Fig. 7. Global performance results: Average crossbar utilization ((a)-(b)), distribution of flit delay since generation for CBR traffic ((c)-(d)), distribution of frame delay since generation for VBR traffic ((e)), and performance of non-multimedia traffic ((f)-(g)).

REFERENCES

[1] K. Dienfendorff and P. Dubey, "How multimedia workloads will change processor design," *IEEE Computer*, vol. 30, no. 9, September 1997.

[2] C.E. Kozyrakis and D. Patterson, "A new direction for computer architecture research," *IEEE Computer*, vol. 31, no. 11, November 1998.

[3] J. F. Kurose and K. W. Ross, *Computer Networking: A top-down approach featuring the Internet*, Addison-Wesley, 2000.

[4] G. Raczkowski, "The new face of streaming media: Accessing rich content on the Internet," WorldCom White Paper, available at `http://www.worldcom.com/es/resources/whitepapers/`, 2002.

[5] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet architecture: An overview," Internet Request for Comments (RFC) 1633, June 1994.

[6] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services field (DS field) in the IPv4 and IPv6 headers," Internet Request for Comments (RFC) 2474, December 1998, proposed standard.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for Differentiated Service," Internet Request for Comments (RFC) 2475, December 1998.

[8] D. Grossman, "New terminology and clarifications for Diffserv," Internet Request for Comments (RFC) 3260, April 2002.

[9] W.J. Dally et al., "Architecture and implementation of the Reliable Router," in *Hot Interconnects II*, 1994.

[10] N. Boden et al., "Myrinet: A gigabit per second LAN," *IEEE Micro*, February 1995.

[11] C. B. Stunkel et al., "The SP-2 high-performance switch," *IBM Systems Journal: Scalable Parallel Computing*, vol. 34, no. 2, 1995.

[12] W.D. Weber et al., "The Mercury interconnect architecture: A cost-effective infraestructure for high-performance servers," in *24th International Symposium on Computer Architecture*, 1997.

[13] D. Garcia and D. Watson, "Servernet II," in *Workshop on Parallel Computer Routing and Communication*, June 1997.

[14] InfiniBand Trade Association, "InfiniBand Architecture specification, volume 1, release 1.0," October 2000.

[15] G. Pfister, *High Performance Mass Storage and Parallel I/O*, chapter 42: An Introduction to the InfiniBand Architecture, IEEE Press and Wiley Press, 2001.

[16] J. Pelissier, "Providing quality of service over Infiniband^{TM} architecture fabrics," in *Hot Interconnects*, 2000.

[17] F.J. Alfaro, J.L Sánchez, J. Duato, and C.R. Das, "A strategy to compute the InfiniBand arbitration tables," in *International Parallel and Distributed Processing Symposium (IPDPS)*. April 2002, IEEE Computer Society.

[18] J. Duato, S. Yalamanchili, M. B. Caminero, D. Love, and F. J. Quiles, "MMR: A high-performance multimedia router - Architecture and design trade-offs," in *International Symposium on High Performance Computer Architecture (HPCA-5)*. 1999, IEEE Computer Society.

[19] M. B. Caminero, *Design of a Router Oriented to Multimedia Traffic in LAN Environments*, Ph.D. thesis, Escuela Politécnica Superior, University of Castilla-La Mancha (Spain), July 2002, in Spanish.

[20] M. B. Caminero, C. Carrión, F. J. Quiles, J. Duato, and S. Yalamanchili, "MMR: A MultiMedia Router architecture to support integrated workloads," Tech. Rep. DIAB-02-02-28, Computer Science Department. University of Castilla-La Mancha, October 2002, available from `http://raap.info-ab.uclm.es/`.

[21] T. E. Anderson et al., "High speed switch scheduling for local area networks," *ACM Transactions on Computer Systems*, 1993.

[22] Y. Tamir and H.C. Chi, "Symmetric crossbar arbiters for VLSI communication switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 1, 1993.

[23] R. Sivaram, C. B. Stunkel, and D. K. Panda, "HIPIQS: A high-performance switch architecture using input queuing," Tech. Rep. OSU-CISRC-10/97-TR45, Ohio State University, 1998.

[24] N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches," *IEEE Transactions on Networking*, vol. 7, no. 2, April 1999.

[25] B. Prabhakar and N. McKeown, "On the speedup required for combined input and output switching," Tech. Rep. CSL-TR-97-738, University of Stanford, November 1997.

[26] I. Stoica and H. Zhang, "Exact emulation of an output queuing switch by a combined input and output queuing switch," in *IEEE/IFIP International Workshop on QoS (IWQoS'98)*, May 1998.

[27] S.T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queuing with a combined input output queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, June 1999.

[28] G. Nong and M. Hamdi, "On the provision of Quality-of-Service guarantees for input queued switches," *IEEE Communications Magazine*, December 2000.

[29] M. Galles, "Scalable pipelined interconnect for distributed endpoint routing: The SGI SPIDER Chip," *Proceedings of Hot Interconnects Symposium IV*, August 1996.

[30] S. L. Scott and G. M. Thorson, "The Cray T3E network: Adaptive routing in a high performance 3D torus," in *Symposium on High Performance Interconnects (Hot Interconnects 4)*, 1996.

[31] H. Eberle and E. Oertli, "Switcherland: A QoS communication architecture for workstation clusters," in *Intl. Symposium on Computer Architecture (ISCA)*, June 1998.

[32] "Web page "ATM Forum", URL: `http://www.atmforum.com/`,".

[33] K. Y. Yun, "A terabit multi-service switch with Quality of Service support," in *Hot Interconnects*, 2000.

[34] K. H. Yum, A. Vaidya, C. R. Das, and A. Sivasubramanian, "Investigating QoS support for traffic mixes with the MediaWorm router," in *High-Performance Computer Architecture (HPCA-6)*, 2000.

[35] L. Zhang, "Virtual Clock: A new traffic control algorithm for packet switching networks," *ACM Transaction on Computer Systems*, May 1991.

[36] P. T. Gaughan and S. Yalamanchili, "Pipelined Circuit-Switching: A fault-tolerant variant of wormhole routing," in *4th IEEE International Symposium on Parallel and Distributed Processing*, December 1992.

[37] P. T. Gaughan and S. Yalamanchili, "A family of fault-tolerant routing protocols for direct multiprocessor networks," *IEEE Transactions on Parallel and Distributed Systems*, May 1995.

[38] P. Kermani and L. Kleinrock, "Virtual Cut-Through: A new computer communication switching technique," *Computer Networks*, vol. 3, 1979.

[39] M.B. Caminero, F.J. Quiles, J. Duato, D. Love, and S. Yalamanchili, "Performance evaluation of the Multimedia Router with MPEG-2 video traffic," in *Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing (CANPC 99) - Lecture Notes on Computer Science*. January 1999, vol. 1602, Springer-Verlag.

[40] M. Katevenis et al., "ATLAS I: A single-chip ATM switch for NOWs," in *Workshop on Communications and Architectural Support for Network-based Parallel Computing (CANPC)*, 1997.

[41] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queuing on a space division packet switch," *IEEE Transactions on Communications*, December 1987.

[42] P. T. Gaughan and S. Yalamanchili, "Adaptive routing protocols for hypercube interconnection networks," *IEEE Computer*, May 1993.

[43] F. Silla and J. Duato, "Improving the efficiency of adaptive routing in networks with irregular topology," in *Conference on High Performance Computing (HiPC)*, 1997.

[44] D. Love, S. Yalamanchili, J. Duato, M.B. Caminero, and F.J. Quiles, "Switch scheduling in the Multimedia Router (MMR)," in *International Parallel and Distributed Processing Symposium (IPDPS)*. May 2000, IEEE Computer Society.

[45] M.B. Caminero, C. Carrión, F.J. Quiles, J. Duato, and S. Yalamanchili, "A cost-effective hardware link scheduling algorithm for the Multimedia Router (MMR)," in *Lecture Notes on Computer Science: International Conference on Networking (ICN'01)*. July 2001, vol. 2094, Springer-Verlag.

[46] M.B. Caminero, C. Carrión, F.J. Quiles, J. Duato, and S. Yalamanchili, "A MultiMedia Router architecture to provide high performance and QoS guarantees to mixed traffic," in *International Conference on Multimedia and Expo (ICME'2002)*. August 2002, IEEE Computer Society.

[47] M.B. Caminero, C. Carrión, F.J. Quiles, J. Duato, and S. Yalamanchili, "A new switch scheduling algorithm to improve QoS in the MultiMedia Router," in *IEEE International Workshop on Multimedia Signal Processing (MMSP'02)*. December 2002, IEEE Computer Society, to appear.

[48] A. A. Chien and J. H. Kim, "Approaches to quality of service in high-performance networks," in *Parallel Computer Routing and Communications Workshop (PCRCW)*, 1997.

[49] J. H. Kim, *Bandwidth and latency guarantees in low-cost, high-performance networks*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1997.

[50] "Generic coding of moving pictures and associated audio. Recomendation H.262. Draft International Standard ISO/IEC 13818-2," 1994.

[51] M.B. Caminero, C. Carrión, F.J. Quiles, J. Duato, and S. Yalamanchili, "Tuning buffer size in the Multimedia Router (MMR)," in *Workshop on Communication Architecture for Clusters (CAC'01) - International Parallel and Distributed Processing Symposium (IPDPS)*. April 2001, IEEE Computer Society.

[52] "IEEE Standard VHDL; Language Reference Manual. IEEE STD1076," 1993.

[53] "Synopsys Reference Manual, version 1998.02," .

[54] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *ACM Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'98)*, 1998.

[55] M. Schwartz and D. Beaumont, "Quality of Service requirements for audio-visual multimedia services," Tech. Rep. ATM94-0640, ATM Forum, 1994.

[56] A. Mekkittikul, *Scheduling Non-uniform Traffic in High Speed Packet Switches and Routers*, Ph.D. thesis, University of Stanford, 1998.