University of Castilla-La Mancha

A publication of the

Department of Computer Science

On the Buffer Size Requirements of the Multimedia Router (MMR)

by

Blanca Caminero, Carmen Carrión, Francisco J. Quiles,
José Duato, Sudhakar Yalamanchili

Technical Report **#DIAB-02-02-29** October 2002

A shorter version of this paper has been submitted to the *Workshop on Communication Architecture for Clusters. IEEE Computer Society*.

José Duato is with the Department 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

On the Buffer Size Requirements of the Multimedia Router (MMR)

*

B. Caminero[1] C. Carrión[1] F. J. Quiles[1] J. Duato[2] S. Yalamanchili[3]

[1] Dept. of Computer Science. Escuela Politecnica Superior. Univ. de Castilla-La Mancha. 02071 - Albacete, SPAIN.
E-mail: {blanca, carmen, paco}@info-ab.uclm.es

[2] Dept. of Computer Engineering (DISCA). Univ. Politecnica de Valencia. 46071 - Valencia, SPAIN.
E-mail: jduato@gap.upv.es

[3] School of Electrical and Computer Engineering. Georgia Institute of Technology. Atlanta, Georgia 30332-0250.
E-mail: sudha@ece.gatech.edu

Abstract

The primary objective of the Multimedia Router (MMR) project is the design and implementation of a compact router optimized for multimedia applications. The router is targeted for use in cluster and LAN interconnection networks, which offer different constraints and therefore differing router solutions than WANs. One of the key design parameters is the amount of buffer space, which is closely related to the silicon area required to implement the router. In this paper, the MMR performance obtained when varying the size of the input buffers is explored. Results show that buffers as small as one flit large suffice to guarantee QoS to both CBR and VBR traffic, thanks to the use of flow control and short links. These results have been obtained with a relatively complex function to compute priority in the link scheduling algorithm. Moreover, the authors have also developed another version of the priority computing function which is more suitable for efficient hardware implementation, while achieving the same or better performance than the original proposal. Simulations show that with this new function, the conclusions about the minimum buffer size needed to provide QoS to multimedia traffic still hold.

1 Introduction

The Multimedia Router (MMR) project is aimed at the design and implementation of a compact router optimized for multimedia applications. The goal is to provide architectural support to enable a range of quality of service (QoS) guarantees at latencies comparable to state-of-the-art multiprocessor cut-through routers.

The design of this Multimedia Router involves the tuning of several qualitative and quantitative parameters. While some of the qualitative parameters have already been addressed in previous work [8, 17], in this paper we will focus on one of the quantitative parameters, namely, the study of the influence of buffer size on MMR performance. The goal is to establish the minimum buffer size needed to guarantee QoS to CBR (Constant Bit Rate) and VBR (Variable Bit Rate) traffic. Small buffers in the router are

*This work was partially supported by the Spanish CICYT under Grants TIC97–0897–C04 and TIC2000-1151-C07

preferred, because they will make a more compact router implementation feasible. But this might lead to a degradation in the QoS received by the multimedia flows. Note that ATM switches implement large buffers in order to fulfill the QoS requirements of the multimedia flows [5].

Studies relating the buffer size needed to provide QoS to the different multimedia flows are presented in this work. The final goal is to make the router as small and compact as possible. Due to the large number of virtual channels the MMR implements, buffer size is a key factor in achieving this goal. Also, the hardware required by other modules (i.e., link schedulers) that need to be replicated for each virtual channel needs to be reduced. The authors have recently presented a proposal towards this goal [3], namely, a hardware cost-effective link scheduling algorithm. The studies in this paper are extended in order to consider this important hardware simplification.

The paper is organized as follows. First of all, a brief motivation of the work is presented. Second, the Multimedia Router architecture is outlined, and its main characteristics are discussed. Next, the resource scheduling algorithms implemented in the MMR are explained, together with a description of the new link scheduling algorithm. Then, simulation results are presented and discussed. Finally, there are some conclusions and future work.

2 Motivation

The cost of both processor and network components has been continuously decreasing over the past few years. This has led to a tighter integration of computation and communication. As a consequence, an explosion of network-based applications, which are characterized by the processing and delivery of continuous data streams and dynamic media [7, 16] in addition to the handling of traditional static data, has taken place. Numerous examples can be drawn from web-based applications, interactive simulations, gaming, visualization, virtual meetings, and collaborative design environments. The changing nature of the workload and cost/performance trade-offs have prompted the development of a new generation of scalable media servers structured around networks of workstations (NOWs) interconnected by high-speed system/local area network (SAN/LAN) fabrics. Systems such as real-time media servers need to service hundreds, and possibly thousands, of clients. Typically, each client has its own distinct quality of service (QoS) requirements, such as packet dropping vs. reliable message transmission, low latency vs. jitter, or throughput vs. latency. Diverse service requirements must be concurrently met with the same set of hardware and software resources.

The need for higher communication bandwidth is becoming especially important in local and storage area environments. There are certain kinds of applications, like 3-D image rendering or video encoding, that need the storage capacity or the computing power of a cluster of workstations in order to achieve good performance, thus increasing the need for higher communication bandwidth [4]. Moreover, the astonishing growth of the Internet in the last few years, and the increasing request for multimedia information, has caused the need for more powerful and sophisticated multimedia servers, usually implemented as clusters of workstations, or even clusters of processing elements together with storage elements [12]. This bandwidth must be provided over a network shared by multiple applications, i.e., not only multimedia flows but also traditional best-effort traffic generated by conventional applications.

Traditional router technology developed for high-speed multiprocessor interconnection networks or high-performance local area networks (LANs) are optimized for low latency for best-effort traffic [1, 6]. These networks are not designed to permit concurrent guarantees of communication performance for multiple applications. Even InfiniBand$^{(SM)}$ [12], the new attempt for high-performance LAN/SAN interconnection devised by a consortium of computing industry's leaders such as Compaq, Dell, Hewlett-Packard, IBM, Intel, Microsoft, Sun Microsystems and others alike, seems to be focused on the transmission of best-effort data. They claim that their switches will provide some sort of QoS capabilities, though. This will be done by using a weighted round-robin virtual lane scheduling algorithm with two levels of priority. Besides InfiniBand switches, the only known approach to providing QoS with a simple router aimed at local environments is the MediaWorm Router [20]. This router uses wormhole switching for both multimedia and best-effort traffic, and uses Virtual Clock [21] for bandwidth allocation among the connections.

On the other hand, ATM provides support for quality of service (QoS), but it has been optimized for wide area networks (WANs). As a consequence, most of the solutions adopted in ATM switches are not valid for LAN environments. In particular, ATM switches require large buffers to provide QoS guarantees. The reasons for using large buffers are twofold: 1) ATM does not implement flow control; therefore, large buffers are required to minimize data losses; and 2) ATM switches for WANs usually support thousands of connections concurrently. The only known compact implementation of ATM in a LAN environment is ATLAS I [14].

The main goal pursued by the Multimedia Router (MMR) project [8] is to design compact single-chip routers able to support a large number of multimedia connections while allocating the remaining bandwidth to best-effort traffic. The MMR should handle this hybrid traffic efficiently, satisfying the QoS requirements of multimedia traffic, minimizing the average latency of best-effort traffic, and maximizing link utilization until network reaches saturation. In order to allow for single-chip implementations, the MMR can only provide small buffers. In spite of using small buffers, and due to the use of flow control, flits are not lost, since they can eventually be stored at the buffers placed in the Network Interface Card (NIC). However, buffering flits at the NIC may difficult resource scheduling and may increase flit delay, possibly increasing the percentage of flits that do not meet their deadlines, thus degrading QoS. Therefore, it is necessary to analyze the behavior of the MMR when most data are buffered at the NIC, determining the minimum buffer size required in the router to guarantee QoS. This analysis should consider VBR traffic.

3 The Multimedia Router Architecture

The general organization of the Multimedia Router is shown in Figure 1-(a). The key issues regarding its architecture are summarized in the following paragraphs. The interested reader is referred to [8] for a more detailed description of the MMR, as well as the design trade-offs.

Switching Technique The only suitable switching technique to be able to offer QoS guarantees to the data flows, must be *connection oriented*, because in this way resources are allocated to the flows in the connection establishment phase, and, when data are effectively transmitted, delays caused by busy resources will not occur. The overhead introduced by the connection establishment phase is paid off with the transmission of a large amount of information, as multimedia flows do.

On the other hand, conventional best-effort traffic will coexist with multimedia flows. Cut-through techniques are the most suitable ones to maximize the performance for this kind of traffic (high throughput, low average latency). However, best-effort traffic must not affect the QoS received by multimedia flows.

As there is no single switching technique suitable for both kinds of traffic, the MMR uses a hybrid approach, where the most suitable switching technique is used for each kind of traffic: a connection-oriented scheme (Pipelined Circuit Switching [10, 11]) for the multimedia flows, and Virtual Cut-Through for best-effort messages. For both schemes, the flow control unit has the same size and will be referred to as a flit. Flits are synchronously forwarded through the crossbar.

Input Buffers The MMR is able to support a large number of connections. For each of them, a virtual channel is provided. This means that a large amount of buffers will be needed. In order to optimize their implementation, the buffers are organized as modules of RAM memory, interleaved with a simple scheme (see Figure 1-(b)). Also, as each virtual channel is assigned to a single connection (in the connection establishment phase previous to flit transmission), all the flits stored in it are headed to the same destination. Thus, the head flit destination port is the same than the flits queued up behind it, and HOL-blocking [13] is avoided. In this paper, the influence of using a different buffer size on MMR performance will be analyzed.

Flow Control The MMR has been designed to avoid data losses. In order to achieve this, flow control is used. The selected scheme is *credit-based flow control* [14]. InfiniBand [12] has also selected this flow-control method. Although stop-and-go flow control [1] is more suitable than credit-based flow control for long wires, as is the case in local area networks, we selected credit-based flow control because it does not require large buffers when links are long.

Note also that, due to the fact that the number of virtual channels per physical link is relatively high (on the order of one to two hundred), link utilization is not likely to degrade, because flow control is performed per connection, and not per physical link. This implies that the probability of having at least one flit ready for transmission (i.e., with credits available) among all the connections sharing a link every flit cycle is quite high, even with long links.

This flow control method introduces some control overhead, which is best paid off with large flits (1024 bits).

Multiplexed Crossbar Due to the large number of virtual channels, the MMR uses a multiplexed crossbar with as many ports as physical channels. A drawback of this organization is that arbitration is needed every time an input link changes from one virtual channel to another one. Arbitration is needed at the input side (*link scheduling*), to select one virtual channel from each physical channel, but it is also needed within the switch (*switch scheduling*), because several input channels might request the same output link. Arbitration is made concurrently with flit transmission. Large flits will also help to hide arbitration delay and amortize crossbar reconfiguration delay. The use of large flits will increase flit latency. However, this is avoided by pipelining flit transmission at the phit level.

4 Resource Scheduling Algorithm

4.1 Bandwidth Allocation

Link bandwidth and switch port bandwidth are split into flit cycles. Flit cycles are grouped into *rounds* also referred to as *frames*. The number of flit cycles in a round is an integer multiple K ($K > 1$) of the number of virtual channels per link. Bandwidth for a connection is allocated as an integer number of flit cycles. The allocated flit cycles will be assigned to the requesting connection every round.

4.2 Scheduling Algorithm: Priority Biasing

At the core of the MMR, there is one key element to provide QoS guarantees to the multimedia flows: the link and switch scheduling algorithm. The authors have proposed a link and switch scheduling algorithm, based on the concept of biased priorities [4, 9, 15] that is well suited to parallelization and pipelining [17].

The key point in this scheme is that priorities are biased according to the ratio between the QoS a flit is receiving and the one it should receive. This approach combines 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).

Note that at least a counter is needed to store the queue delay experienced by every flit. This means that one of these counters has to be included for every flit that can be queued at the MMR input buffers, i.e., the number of counters will be the same as the buffer size for every virtual channel.

Two different biasing functions were proposed:

Inter-Arrival Biased Priority (IABP) The priority of a flit is computed as the ratio between the queuing delay, and the inter-arrival time for the flits in the connection. The effect is that the priority grows as queuing delay 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 switch.

Jitter Biased Priority (JBP) The priority value is computed 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. The goal of this algorithm is to make all the flits in a connection experience the same delay, i.e., zero jitter.

In a previous paper [2], the biasing function used to determine the priority was IABP. In the next section, some drawbacks inherent to this function are indicated, and a simplification of it is described.

Regardless of the biasing function used, link and switch scheduling work as follows.

Link scheduling is carried out by selecting the C virtual channels whose head flits have the highest priorities, among all the virtual channels in a physical link ($1 \leq C \leq N$, N = number of ports). Priorities are computed as outlined above. All the physical links perform link scheduling in parallel.

These candidates are forwarded to the *switch scheduler*, which computes a conflict-free matching among input and output ports, in a way motivated by chances for parallelization and pipelining.

Link and switch scheduling are partitioned into a set of basic decisions: *candidate selection*, *port ordering*, and *arbitration*.

Candidate selection is performed during link scheduling, as described in previous paragraphs. The candidates are ordered by levels, according to their priority values. Every input link produces a *candidate vector*, where the requested output port and the priority are stored for every candidate.

All the candidates selected by every port are 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. Then, 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 2 shows an example selection matrix with its corresponding conflict vector.

Next phase is *port ordering*. The ordering function selects output ports first by level and then in increasing order of conflict 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.

The last phase is *arbitration*: once an output port has been selected for matching, if there are several requests for it, one of them must be chosen. 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 recomputed.

At the end of the process, a conflict-free matching between input and output ports of the switch will have been obtained. Also, at most one virtual channel from every physical link will have been selected to transmit its head flit. All the selected flits will be then forwarded synchronously through the crossbar. At the same time, the link/switch scheduling algorithm starts a new execution. See [17] for more details.

4.3 The SIABP (Simple-IABP) Biasing Function

In the first switch scheduling proposal for the MMR [17], link scheduling was performed by using the concept of *biased priorities* [4, 9, 15]. The priority value for each flit was related to the QoS requested by the connection, and was biased depending on the QoS received by it. One of the proposed biasing functions was IABP (Inter-Arrival Based Priority). With IABP, the priority value was computed as the ratio between the queuing delay experienced by the flit, and the Inter-Arrival Time (IAT) specified for the connection. Results showed that this approach provides good performance for CBR traffic, differencing the QoS received by each type of connection. The main problem with the IABP algorithm is that it is a theoretical approach not suitable for a practical implementation because of the way priorities are computed, since a division operation is needed for every virtual channel. Hardware implementations of dividers are slow and expensive, and hardly fit into our fast, compact router. So, it is necessary to devise an alternative algorithm. The idea is to apply the same rationale introduced by IABP, that is, relate the bandwidth required by the connection to the experienced queuing delay, but replacing the division with some other operation that is faster and cheaper to implement.

In the new algorithm, called *SIABP* (*Simple-IABP*), the priority value for each flit is computed in a different way. The queuing delay for a flit is stored as a counter that is incremented every router cycle, that is, the queuing delay is measured as a number of router cycles.

The priority biasing function is implemented as follows. The initial value for the priority is the bandwidth required by the connection. Instead of representing this by the IAT of the connection, the number of slots per frame reserved to service the average bandwidth of the connection is used. The advantage of this approach is that this magnitude is an integer value.

It must be noted that computing the ratio between the queuing delay and the IAT (as IABP does) is equivalent to compute the product between the queuing delay and the bandwidth requirements. But if the product could be replaced with some shift operation, the implementation would be much simpler. So, the priority value is shifted to the left (that is, it is multiplied by two) every time the queuing delay value becomes greater than $1, 2, 4, \ldots, 2^{n}$, that is, every time a bit in the queuing delay counter is set for the first time since it was last reset. In this way, the QoS required (represented by the initial priority value) is also related to the QoS received by the flit (the queuing delay). Thus, the SIABP function can be implemented in hardware with just a shifter and some combinational logic, while the IABP function needs a floating point divider. Note however, that those circuits have to be replicated for each flit buffer. Therefore, larger buffers also imply a larger amount of control logic.

The reduction in hardware complexity and delay achieved with the SIABP algorithm was determined by using VHDL design tools. Results showed a reduction of 22 times in terms of silicon area, and of 38 times in terms of delay, while still satisfying the QoS needs of the multimedia flows [3].

5 Experimental Results

5.1 Input Model and Simulation Configuration

Simulations have been carried out with a single MMR router configuration, with one Network Interface Card (NIC) attached to every input link. Traffic sources inject their flits into buffers located in the corresponding NIC (see Figure 3). These buffers are considered to be infinite because the host main memory can also be used if the NIC buffers become full. The MMR buffer size is limited to a few flits per virtual channel. With this configuration, accurate results regarding the delay introduced by the router, and the delay caused by the different virtual channels competing for the use of the physical link connecting to the MMR are obtained.

As the buffer size in the MMR is limited, and flit loss needs to be avoided, credit-based flow control has been simulated between the NIC and the MMR. Note that links are quite short. Thus, the delay introduced by them when transmitting flow control information is very short, compared to the time needed to transmit a full flit. Also, note that flow control information is transmitted in a single phit (which takes a few nanoseconds), while a full flit requires many phits (which takes several hundreds of nanoseconds).

The physical link controller located in the NIC only considers for forwarding to the router the flits corresponding to virtual channels which have credits available, that is, the flits that have free space in their corresponding MMR buffer. This link controller forwards the flits belonging to the different connections in a demand-driven round-robin fashion, i.e., it performs round-robin among all the connections with both flits and credits available. Simulation results suggest that this simple scheme suffices for forwarding flits to the MMR while guaranteeing QoS. The reason is that when the scheduling algorithm guarantees QoS in the router, the NIC simply adapts flit forwarding to the needs of the router, thanks to the use of small buffers and flow control.

Simulations have been carried out on a 4×4 router. Simulations consider 1024-bit flits, and 1.24 Gbps 16 bit-wide links. The link/switch scheduling algorithm is implemented with four levels of candidates (see Section 4.2).

The first simulations were carried out with a CBR-only workload. Results did not show any influence when varying buffer size, since flits are able to traverse the router without filling the buffers in the corresponding virtual channel as long as there is no saturation, due to their constant generation rate. Thus, the study focused on the performance of the router with VBR traffic. In this model, the traffic generation rate varies over time for all the connections, and can cause saturation over short periods of time. In this case, buffer size may influence QoS.

The VBR traffic models used in our simulations are based on MPEG-2 video traces. This is a typical type of multimedia flow. MPEG-2 video coding standard [22] 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). The GOP used is IBBPBBPBBPBBPBB. I frames encode independent frames, that is, I frames do not need any other information but themselves to be decoded. P frames need the previous I frame 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 consuming, because they carry more information, and B frames are the least bandwidth consuming. Figure 4 illustrates the traffic pattern of a typical MPEG-2 video sequence.

Every 33 milliseconds, a frame must be injected. A frame is composed of a number of flits. The number of flits that compose every frame has been extracted from real MPEG-2 video traces. Some data about their frame sizes are shown in Table 1.

Two ways of injecting these flits into the NIC buffers have been considered:

Back-to-Back (BB) model. In this model, all the flits are transmitted at a peak bandwidth, common to all the connections. The *peak bandwidth* is such that it allows the injection of the largest frame among all the connections within 33 millisecs. Transmission of each frame starts at a frame time boundary. All the flits that compose the frame are injected at the selected peak rate, and then, the source becomes idle until the next frame time boundary. The peak bandwidth in the experiments is 50 Mbps for all the connections. Figure 5-(a) depicts this model. IAT_p stands for the Inter-Arrival Time (IAT) related to the peak bandwidth.

Smooth-Rate (SR) model. In this model, the flits that compose a frame are transmitted with a different IAT for every frame, in such a way that the flits of a frame are evenly distributed within the frame time. The IAT has been computed as the ratio between 33 msecs and the number of flits that compose the frame. A graphical representation is shown in Figure 5-(b).

Simulations have been carried out until four complete GOPs (Group Of Pictures) from every connection have been forwarded through the router. The connections sharing the same physical link have been randomly aligned, that is, they start at a random time within a GOP time. The destination ports for the connections have been chosen randomly.

5.2 Simulation Results

First of all, average crossbar utilization is shown in Figure 6 for both VBR models (SR and BB). Results with buffer sizes of 1, 4 and 1024 flits are shown. Other intermediate sizes were also tested, but the results showed an intermediate behavior between the presented curves. The reason for testing such a wide range of sizes is to check if a buffer with capacity for a whole video frame would impact performance significantly. The largest video frame used in these simulations contains 1000 flits, approximately. Also, small buffer sizes were tested in order to establish the minimum buffer size necessary to guarantee QoS to MPEG-2 flows. Note that small buffers are required to keep the router compact.

The behavior for buffer sizes of 1 and 4 flits is very similar. In both cases, the network enters saturation at 81% load (approximately) without showing any performance degradation beyond saturation. The sources generated up to 100% of workload but, beyond 81% of utilization, flow control makes the flits that do not fit into the MMR buffers (which are full) stay at the NIC buffers. The Connection Admission Control algorithms should not admit connections over that level of workload (we deactivated those algorithms to run the simulations). Surprisingly, utilization drops heavily for buffers with 1024 flits and the SR injection model. One reason for this is the degradation of the priority mechanism (which is based on queue delays) when the queues in the MMR become long. Note that priority takes into account the queue delay, that is, there is a counter associated to every flit in every buffer that stores the delay experienced by the flit in the MMR queues. Consider the case when contention occurs and flits start to form queues. When buffers are large, flits are more likely to wait for their turn for transmission within the MMR input buffers. This implies that their queue delay will be greater than the one experienced if buffers were smaller. In the latter case, flits would spend more time in the NIC buffers, and their queue delay in the router would not be increased as much. So, when using large buffers, those counters are more likely to overflow, as queue delay can reach higher values, thus corrupting the priority mechanism. This degradation also affects slightly the performance obtained by the BB injection model. Besides, for the SR injection model, the Inter-Arrival Time (IAT) for the connection, which is used in the IABP scheduling algorithm for computing priorities, changes for every video frame. The modification in the IAT for the connection is transmitted across routers as a control message. These messages are transmitted independently from the flits belonging to the connection, and may arrive at a router before all the flits from the previous frame have been forwarded. When the router is near saturation, and the flits stay for long at the MMR queues, flits belonging to an I frame are treated as if they belonged to a B frame. So, their priority increases more slowly, and they are not forwarded as soon as they should. This effect, coupled with the degradation of the priority mechanism, explains the performance degradation observed for 1024 flit buffers and the SR injection model.

Next, the Quality of Service (QoS) received by the connections must be assessed. As the considered VBR models are MPEG-2 video streams, the QoS metrics are related to the performance obtained by the application data units, i.e., the *frames*, rather than individual flits. First of all, the average frame delay since generation is considered. Frame delay was computed as the delay suffered by the last flit from the frame, because in this way, the measure is independent of the injection model used. Delays are measured since generation, thus computing the global time the frame spends in the network. Results are shown in Figure 7. On the left side, results for the SR injection model are shown, while the plot on the right side corresponds to the BB injection model. Notice the logarithmic scale on the vertical axis.

For both injection models, it can be seen that, with workloads up to 71%, frame delays are pretty low. For the next workload point (77% load, approximately), frames experience an important increase in their average delay, although saturation has not been still reached. There are two reasons for this. First, there is more contention for the output ports within the crossbar, and the flits take longer to be scheduled. Second, and most important, as traffic is not uniform, the router enters saturation when I frames are being transmitted, causing greater delays. Later, as the rest of frames are being transmitted, the scheduler is able to transmit all the flits with lower delays. Anyway, frames are still able to meet their deadlines, as will be shown later. For workloads higher than 80% load, the router clearly saturates, and frames are no longer able to meet any deadline.

Besides, for the SR injection model it can be seen that average frame delay is not affected by the variation in buffer size when the workload is low. On the other hand, when the BB injection model is used, average frame delays before saturation are higher than the ones obtained with the SR injection model, and get worse when buffer size is increased. The reason is that the BB injection model concentrates flit injection at the beginning of every frame time, causing a transient saturation period. It has been previously shown that smaller buffer sizes perform better when saturation is reached. Thus, this explains the better average delays obtained for one and four flit buffers. As sources finish injecting their frames, the saturation fades away until the next frame boundary. But the flits already injected will suffer from this congestion, as can be observed by the increase in the average frame delay (Figure 7, right-hand side).

It is also interesting to compute the distribution of frame delays, in order to know how many frames suffer from delays longer than a certain deadline. This will give a more accurate measure of how many frames get the QoS they need, compared to the average delay. Results are shown in Figure 8 for both the SR and the BB injection models. Each point in the plots shows the ratio of frames received at the output ports of the router whose delay is lower than the corresponding deadline to the total number of frames generated by the sources. The plots show the distribution of frame delay since generation for the workload levels labeled as A, B, and C in Figure 6, which are close to or beyond saturation. The deadlines considered are related to a frame time (FT, 33 milliseconds), and are set as $FT/16$, ..., $FT/2$, FT, $2 \times FT$, ..., $64 \times FT$, that is, they range from 2.06 to 2212 milliseconds. All the plots show a vertical line at 1 second, for reference purposes.

It can be seen that best performance is obtained for small buffer sizes. For the A workload point, that is, right before saturation, and with one flit buffers, all the frames suffer delays lower than 8.25 milliseconds when the SR injection model is used. For four-flit buffers, the lower deadline fulfilled by all the frames is 16.5 milliseconds, while for 1024 flit buffers, the frames are only able to fulfill the 66 milliseconds deadline.

These differences hold for the BB injection model. As the BB injection model is more demanding, its performance is poorer than the one obtained with the SR injection model, for one flit buffers. Anyway, all the frames suffer delays lower than 8.25 milliseconds when workload is just below saturation, and single flit buffers are used. When larger buffers are used, the lower deadline fulfilled by all the frames is 16.5 milliseconds. The reason of this behavior is the degradation on the priority mechanism used in the resource scheduling algorithm for large buffers, when the router is near saturation, and flits enqueue in the router buffers.

For the workload point labeled as B, the router is clearly saturated for 1024-flit buffers. Note that, for the SR injection model, approximately 15 % of the generated frames are not able to traverse the router even with the longest deadline, and stay at the NIC buffers or the MMR input buffers. However, performance is still tolerable for one and four-flit buffers, when the SR injection model is used, since deadlines of 66 millisecs can be met by all the generated frames. With the BB injection model, 1024-flit buffers perform better, but still worse than small buffers. Nevertheless, performance with small buffers is also inadmissible, since the lowest delay than can be met by all the frames is just 132 millisecs for one-flit buffers and 264 millisecs for four-flit buffers.

For the higher workload point considered, C, and for both injection models, saturation is reached for all the buffers sizes (there are always some remaining frames at the NIC buffers or at the MMR input buffers by the end of the simulation), although small buffers perform better, again. This workload point should not be reached, because QoS guarantees can not be fulfilled for the multimedia flows. Proper Connection Admission Control (CAC) strategies should be applied in order to avoid this.

These results are rather encouraging, because a typical deadline for MPEG-2 video transmission is one second between end-points. This is the value for the CTD (Cell Transfer Delay) recommended by the ATM Forum for video distribution services, using MPEG-2 [19].

Another interesting measure of the QoS received by the multimedia connections is jitter, that is, the variation in the delay experienced by two adjacent flits belonging to the same connection. As we are dealing with MPEG-2 video connections, and as we did previously, we will apply this parameter to the application data units, that is, the video frames. Figure 9 shows the average frame jitter for both injection models. Let's focus on the useful workload range previously discussed, that is, workloads under the B workload point (81% approximately). It can be seen that for both injection models, average frame jitter does not unboundly grow when 1 flit buffers are used. In this case, average jitters are under 10 microsecs for the SR injection model, and under 4 microsecs for the BB injection model. This is due to the fact that in this case, the priority mechanism performs correctly, and thus, all the frames are treated in a similar way.

These are quite encouraging results, because the jitter allowed in MPEG-2 video transmission is around several milliseconds, that is, jitter must be low enough so that a person can see the video sequence smoothly, at a regular rate. Note that jitter values of up to several milliseconds can be absorbed at the destination by using jitter absorption techniques [18].

5.3 Comparison between IABP and SIABP Biasing Functions

As was shown previously, when using the IABP algorithm as the biasing function in the link scheduler, buffers as small as one flit suffice to guarantee QoS to the multimedia flows. But the IABP is a theoretical approximation unfeasible for a hardware implementation, and thus some simplifications had to be done in order to make a compact, fast and hardware cost-effective implementation feasible. The result is the so-called *SIABP biasing function* (SIABP stands for Simple-IABP) (see Section 4.3). This simplification in hardware must not degrade the QoS received by the different multimedia flows. Thus, simulations with both biasing functions have been carried out to assess that there is no degradation in the performance achieved by the traffic when applying this simpler version of the algorithm.

Since results in the previous section were obtained with VBR-only workloads, only results with the same kind of workload are presented. More detailed results with also CBR workloads can be found in [3].

First of all, the average crossbar utilization is shown in Figure 10. For both injection models, it can be noticed that in spite of its simpler implementation, SIABP achieves a slightly better crossbar utilization than IABP when saturation is reached (workload greater than 87%). For lower loads, their performance in terms of amount of forwarded traffic is the same.

Let's check now the QoS received by the connections. As in the previous section, results regarding the performance obtained by video frames are shown. Figure 11 shows the average frame delay since generation. Note the logarithmic scale in the vertical axis. It can be noticed that for medium loads, both approaches behave quite similarly, and when workload increases over 75 % and becomes near saturation, SIABP performs slightly better than IABP. This holds for both flit injection models.

The distribution of frame delays since generation has also been computed, as in the previous section. Results are plotted in Figure 12 for both injection models. Only results for the workload levels right before (workload point B) and right after saturation (workload point C) are shown.

With the SR injection model, all the frames experience delays lower than 132 millisecs for both algorithms, with a workload of 81% (workload point B). Tighter thresholds can be met by more frames when using the SIABP algorithm. Even when workload has entered saturation (workload point C, 87%), both algorithms make frame delays fit into deadlines of around 500 milliseconds. If SIABP is used, more frames are able to fit into lower delay bounds. It should also be noted that with the IABP algorithm, more frames fail to traverse the router than when using the SIABP algorithm. On the other hand, with the BB injection model, results are quite similar for both algorithms. Deadlines of 132 millisecs right before saturation, and around 500 millisecs right after saturation can be met both with IABP and SIABP. These are quite reasonable values, since a typical deadline for MPEG-2 video transmission is one second between end-points. This is the value for the CTD (Cell Transfer Delay) recommended by the ATM Forum for video distribution services, using MPEG-2 [19].

The reason for this behavior is that the flits injected by SR sources usually have to spend less time in buffers, because they are injected at a lower rate than the ones from BB sources. Within that period of time, their priority increases quicklier. Note that as queue delay continues increasing, priority is doubled less frequently. On the other hand, BB sources inject their flits as bursts with a fixed peak rate. So, their flits have usually to wait longer, and their priority values get to increase slower and slower, in a more similar way to what IABP does. This effect is most noticeable for workloads over 70%.

Lastly, regarding frame jitter (see Figure 13), SIABP achieves clearly better results than IABP for the SR injection model, specially with workload levels greater than 70%, while with BB model SIABP performs slightly worse. Anyway, average jitters are always under 8 microsecs, which is a quite bearable value. Recall that even jitter values of several millisecs can be absorbed at the destination by using jitter absorption techniques [18].

5.4 Buffer Size Influence with the SIABP Biasing Function

Now it is necessary to assess the conclusions obtained for the IABP algorithm regarding minimum buffer size requirements in the MMR. For that purpose, the same tests previously done have been carried out just replacing the biasing function used in link scheduling.

First of all, the QoS received by the MPEG-2 video frames is presented in Figure 14 for both injection models. It can be seen that now buffer size has little influence on the average delay experienced by the video frames, specially for the BB injection model. This is due to the fact that when queuing delay in the router increases, which is more probable when buffers are large, priority increases more slowly than with IABP, and thus overflows that corrupt the priority system are not likely to happen.

To get more detailed information on frame delays, their distribution is shown in Figure 15, for the workload level labeled as B in Figure 14, just when the router is entering saturation (83% workload, approximately).

For the SR injection model (plot at the left of Figure 15), it can be seen that, although approximately 10% more frames are able to meet the tightest thresholds when using the largest buffers, beyond the 66 microsecs threshold, performance trend swaps, and is for the smallest buffer size when better performance is obtained. Notice that in this case all the generated frames are able to traverse the router, while there are some frames that cannot do it when increasing buffer size. It seems that when using the largest buffer size frames are not treated evenly: there are a few of them that suffer from very long delays, as can be deducted from this plot. This is confirmed by the jitter results (Figure 16, left), where it can be seen that jitter remains stable for single flit buffers, while it grows unboundly when larger buffers are used.

For the BB injection model (plot at the right of Figure 15), large buffers perform slightly better in terms of delay, although beyond the 16.5 millisecs threshold, performance is quite similar regardless of buffer size. Again, jitter grows heavily for large buffer sizes ((Figure 16, right) , showing the great differences in delay experienced by some frames. It can be concluded that, although some frames experience better performance in terms of delay when using this injection model, this performance gain does not pay off for the extremely large amount of silicon area needed for those large buffers. Recall that a typical deadline for MPEG-2 video transmission is one second between end-points [19].

6 Conclusions and Future Work

The Multimedia Router is intended to provide QoS guarantees to multimedia traffic in local area environments. One of the quantitative parameters involved in its design is the size of the input buffers. Thus, its influence on QoS has been analyzed in this work. Also, the study has been extended in order to consider hardware simplifications made in the link scheduling modules of the router. The overall goal is to make a router as compact and small as possible, while still satisfying the QoS needs imposed by the multimedia flows.

The analysis has been carried out through simulations. Both CBR and VBR traffic types have been studied. For CBR traffic, the buffer size has no influence in performance. For VBR traffic, the workload was composed of MPEG-2 video streams. Two different injection models were tested. Simulations show that the use of small buffers does not degrade performance when the traffic is composed of VBR MPEG-2 streams. Moreover, performance is slightly worse when buffers are large enough to store a whole video frame. The preliminary conclusion is that, thanks to flow control, buffers as small as one flit long suffice to provide QoS to both types of traffic. What is more, when replacing the relatively complex priority biasing algorithm initially used with a more hardware cost-effective one, performance does not degrade when small buffers are used. If this still holds for more complex topologies, it will make a compact router implementation feasible.

Moreover, when using small buffers, if congestion occurs, flits would mainly queue at the buffers placed in the NIC at the source node of the connection. Thus, the source can make decisions on whether to tear down the connection or discard flits when congestion is detected. This would be easily done by checking the occupation of the NIC buffers. In this way, if a connection is torn down, there are few flits from it in the network, and minimum network resources will be wasted to transmit flits from an aborted connection. Another remarkable conclusion is that, by combining small buffers with the new link scheduling proposal, that is, with a quite cost-effective implementation, the router is able to reach more than 75% of utilization while still providing QoS to the MPEG-2 flows. Recall that commercial ATM switches implement several Megabytes of memory for buffering [23, 24], while our proposed router requires only a few tens of Kilobytes.

This study must be further extended to a network composed of several MMR's, in order to assess the conclusions obtained in this paper. Another related issue is the Connection Admission Control (CAC) algorithm. Once the workload level at which the router enters saturation has been established, the target of the CAC algorithms becomes clear: connections must be accepted or rejected on the basis of not feeding the router with workload over that level, so that all the ongoing connections can get their QoS requirements fulfilled.

References

[1] N. J. Boden, et al., "Myrinet - A gigabit per second local area network," *IEEE Micro*, pp. 29–36, February 1995.

[2] B. Caminero, C. Carrión, F. J. Quiles, J. Duato, and S. Yalamanchili, "Tuning buffer size in the Multimedia Router (MMR)," *Workshop on Communication Architecture for Clusters*, held in conjunction with *2001 International Parallel and Distributed Processing Symposium (IPDPS'2000)*, April 2001.

[3] B. Caminero, C. Carrión, F. J. Quiles, J.Duato, and S. Yalamanchili, "A Cost-effective Hardware Link Scheduling Algorithm for the Multimedia Router (MMR)," *IEEE International Conference on Networking (ICN'01)*, July 2001.

[4] A. Chien, J.H. Kim, "Approaches to Quality of Service in High Performance Networks," *Proceedings of the Workshop on Parallel Computer Routing and Communication*, Lecture Notes in Computer Science, Springer-Verlag, pp.1-19, June 1997.

[5] P. Cuenca, L. Orozco-Barbosa, F.J Quiles, A. Garrido, "Loss-Resilient ATM Protocol Architecture for MPEG-2 Video Communications," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, June 2000.

[6] W. J. Dally, et al., "The Reliable Router: A reliable and high-performance communication substrate for parallel computers," *Proceedings of the Workshop on Parallel Computer Routing and Communication*, pp. 241–255, May 1994.

[7] K. Dienfendorff, P. Dubey, "How multimedia workloads will change processor design," *IEEE Computer*, vol. 30, no. 9, pp.43-45, September 1997.

[8] J. Duato, S. Yalamanchili, M.B. Caminero, D. Love, and F.J. Quiles, "MMR: A high-performance multimedia router. Architecture and design trade-offs," *Proceedings of the 5th Symposium on High Performance Computer Architecture (HPCA-5)*, pp. 300-309, January 1999.

[9] D. Garcia, D. Watson, "ServerNet II," *Proceedings of the Workshop on Parallel Computer Routing and Communication*, pp. 119-136, June 1996.

[10] P. T. Gaughan and S. Yalamanchili, "Adaptive routing protocols for hypercube interconnection networks," *IEEE Computer*, vol. 26, no. 5, pp. 12–23, May 1993.

[11] P. T. Gaughan and S. Yalamanchili, “A family of fault-tolerant routing protocols for direct multiprocessor networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 5, pp. 482–497, May 1995.

[12] D. Pendery, J. Eunice, “InfiniBand Architecture: Bridge Over Troubled Waters,” *Research Note*, available from the web page: `http://www.infinibandta.org`

[13] 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.

[14] M. G. H. Katevenis, et al., “ATLAS I: A single-chip ATM switch for NOWs,” *Proceedings of the Workshop on Communications and Architectural Support for Network-based Parallel Computing*, February 1997.

[15] J.H. Kim, “Bandwidth and latency guarantees in low-cost, high-performance networks,” Ph. D. Thesis, Department of Computer Sciences, University of Illinois at Urbana-Champaign, 1997.

[16] C.E. Kozyrakis, D. Patterson, “A new direction for computer architecture research,” *IEEE Computer*, vol.31, no. 11, pp. 24-32, November 1998.

[17] D. Love, S. Yalamanchili, J. Duato, M.B. Caminero, and F.J. Quiles, “Switch Scheduling in the Multimedia Router (MMR),” *Proceedings of the 2000 International Parallel and Distributed Processing Symposium (IPDPS’2000)*, May 2000.

[18] M. Perkins and P. Skelly, “A hardware MPEG clock recovery experiment for variable bit rate video transmission,” ATM Forum, ATM94-0434, May 1994.

[19] M Schwartz and D. Beaumont, “Quality of service requirements for audio-visual multimedia services,” ATM Forum, ATM94-0640, July, 1994.

[20] K.H. Yum, A. Vaidya, C.R. Das, A. Sivasubramanian, “Investigating QoS Support for Traffic Mixes with the MediaWorm Router,” *Proceedings of the 6th Symposium on High Performance Computer Architecture (HPCA-6)*, January 2000.

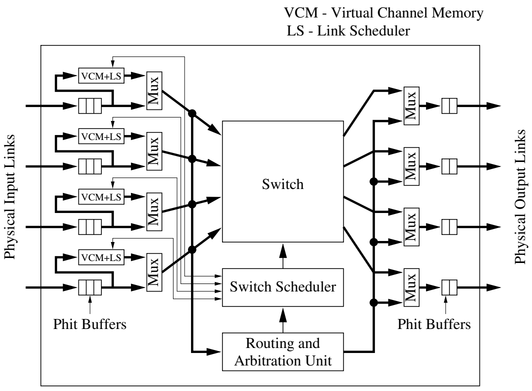
[21] L. Zhang, “Virtual Clock: A new traffic control algorithm for packet switching networks,” *ACM Transactions on Computer Systems*, May 1991.

[22] “Generic coding of moving pictures and associated audio,” Recommendation H.262, Draft International Standard ISO/IEC 13818-2, March, 1994.

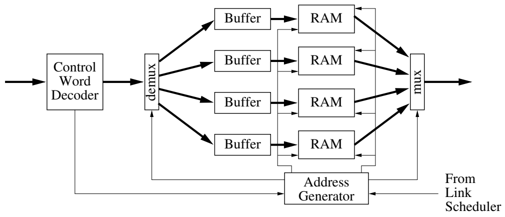
[23] Alcatel Carrier Internetworking Division, `http://www.cid.alcatel.com/`

[24] Cisco 12000 Series - Gigabit Switch Routers, `http://www.cisco.com/`

Appendix: Figures and Tables



(a) Router architecture



(b) Virtual Channel Memory Organization

Figure 1. MMR Architecture

Table 1. MPEG-2 video sequence statistics

	Image Size (bits)		
Video Sequences	Max.	Min.	Average
Ayersroc	535030	148755	232976
Hook	454560	159622	272738
Martin	444588	116094	199880
Flower Garden	900139	308411	497126
Mobile Calendar	970205	412845	600742
Table Tennis	933043	260002	440547
Football	590532	340246	441459

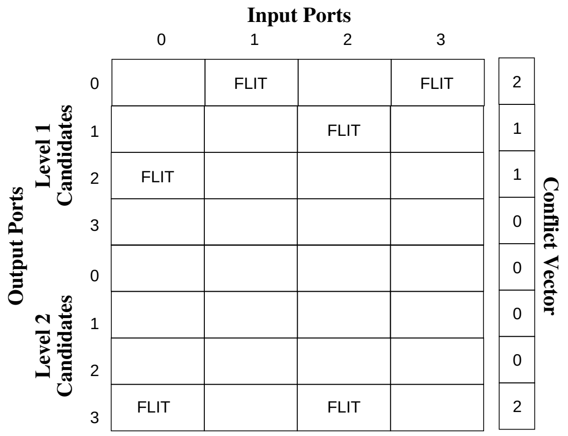


Figure 2. An example selection matrix and its corresponding conflict vector

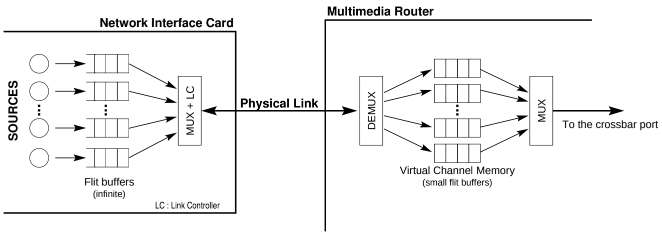


Figure 3. Conceptual representation of the NIC and MMR buffers

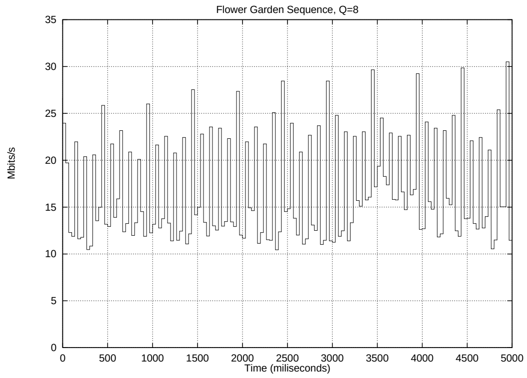
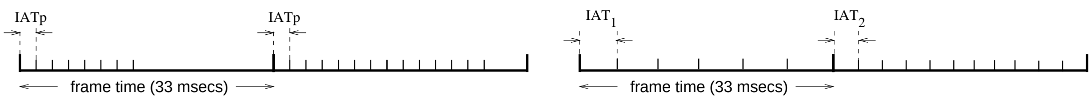


Figure 4. Example of a typical MPEG-2 video sequence



(a) Back-to-Back (BB) injection model (b) Smooth-Rate (SR) injection model

Figure 5. VBR injection models

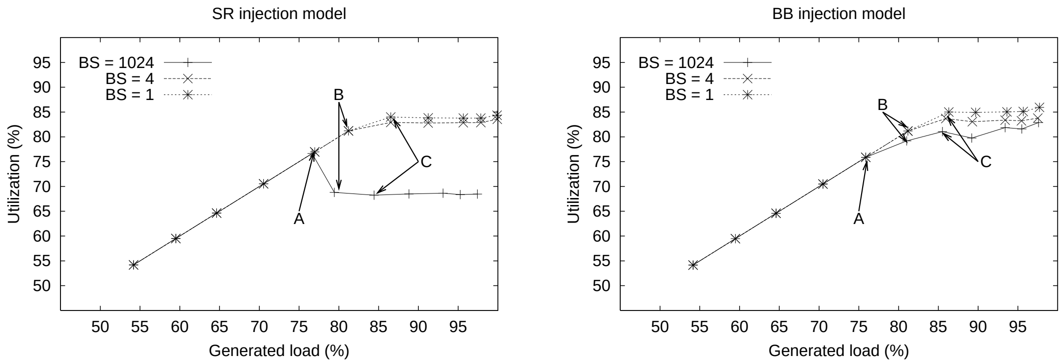


Figure 6. IABP algorithm: Average crossbar utilization

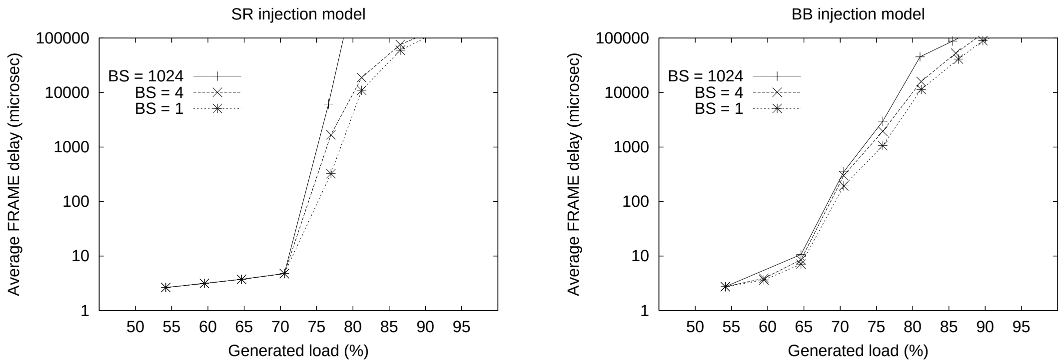
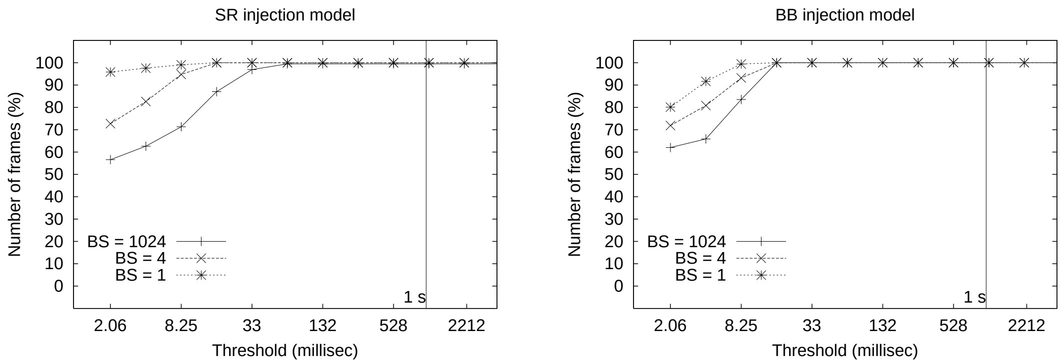
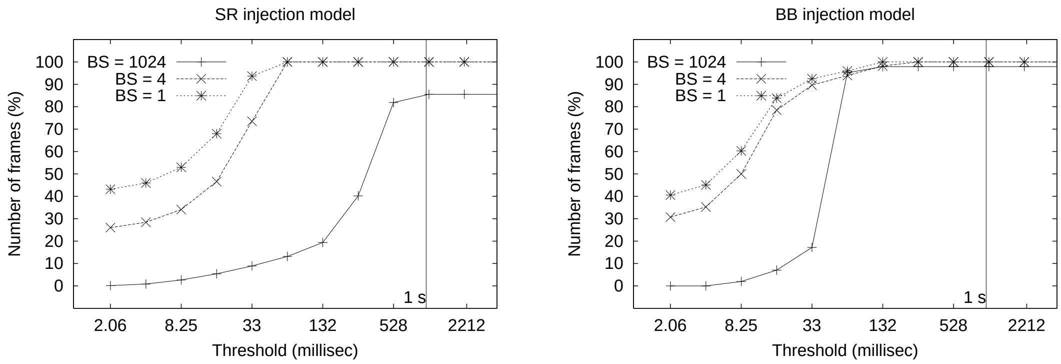


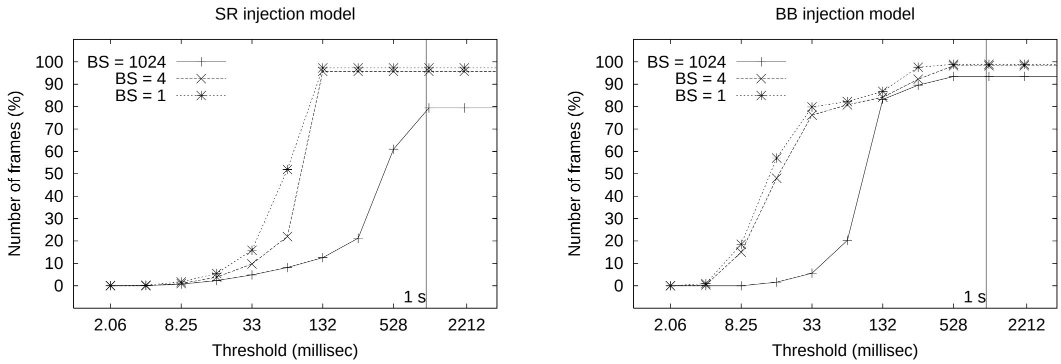
Figure 7. IABP algorithm: Average frame delay since generation



(a) Workload level A



(b) Workload level B



(c) Workload level C

Figure 8. IABP algorithm: Distribution of frame delay since generation

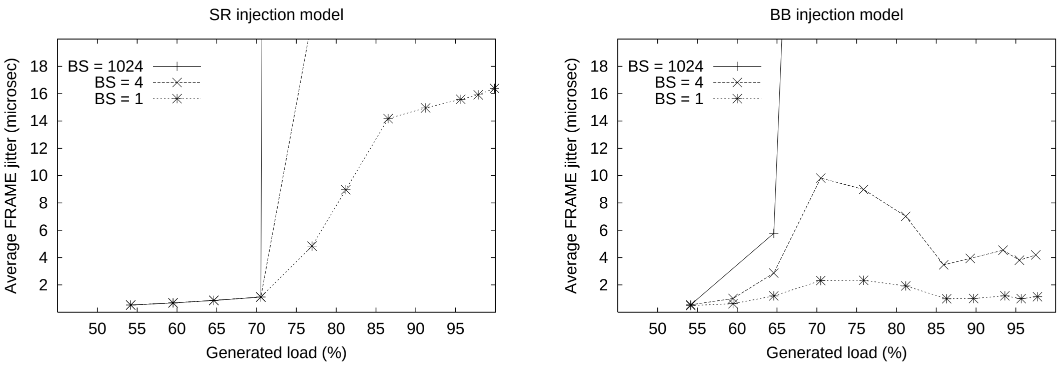


Figure 9. IABP algorithm: Average frame jitter

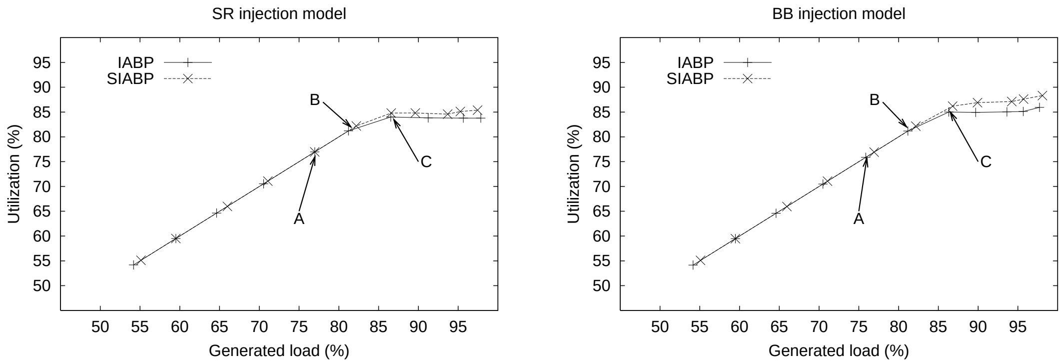


Figure 10. IABP vs. SIABP: Average crossbar utilization

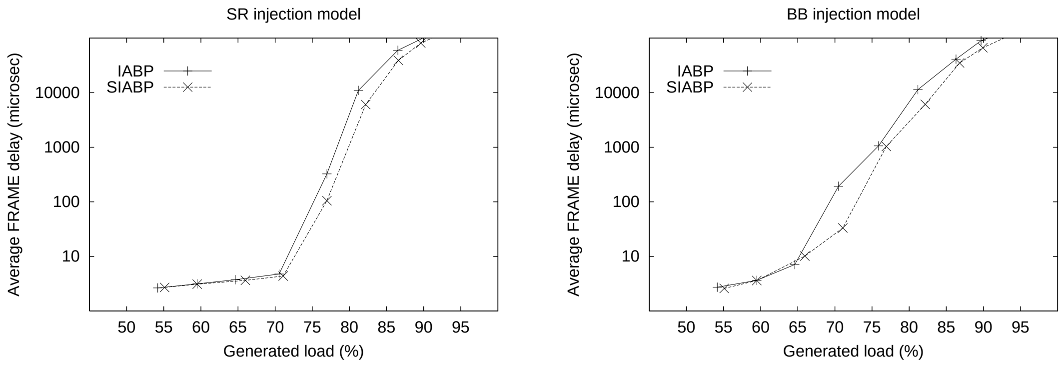
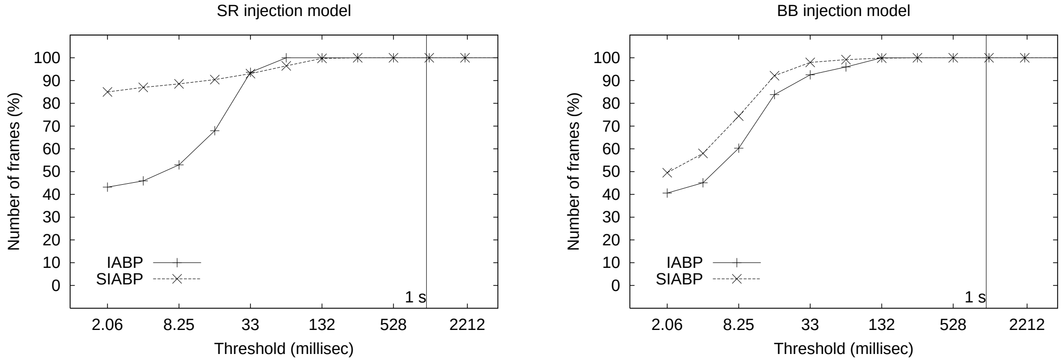
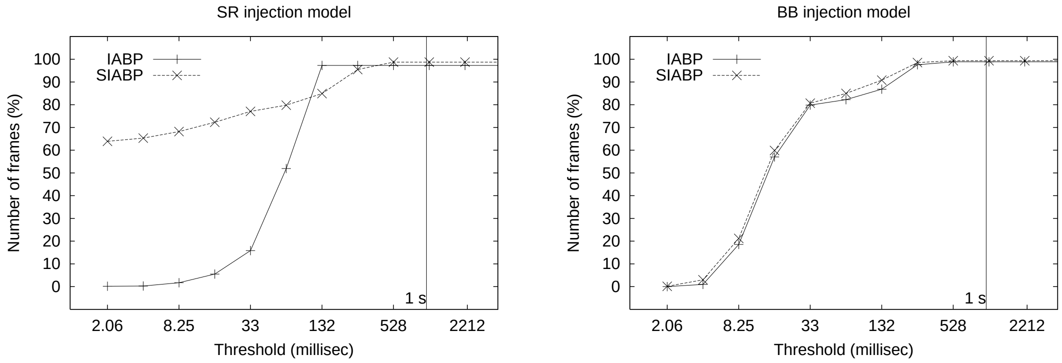


Figure 11. IABP vs. SIABP: Average frame delay since generation

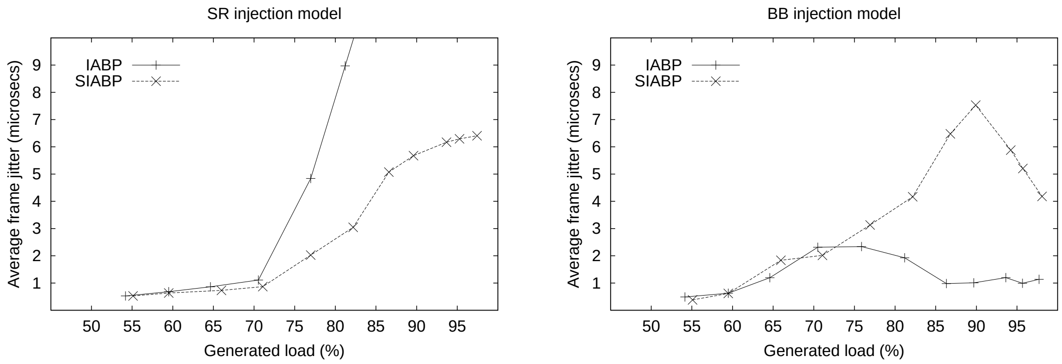


(a) Workload level B



(b) Workload level C

Figure 12. IABP vs. SIABP: Distribution of frame delay since generation



SR traffic model BB traffic model

Figure 13. IABP vs. SIABP: Average frame jitter

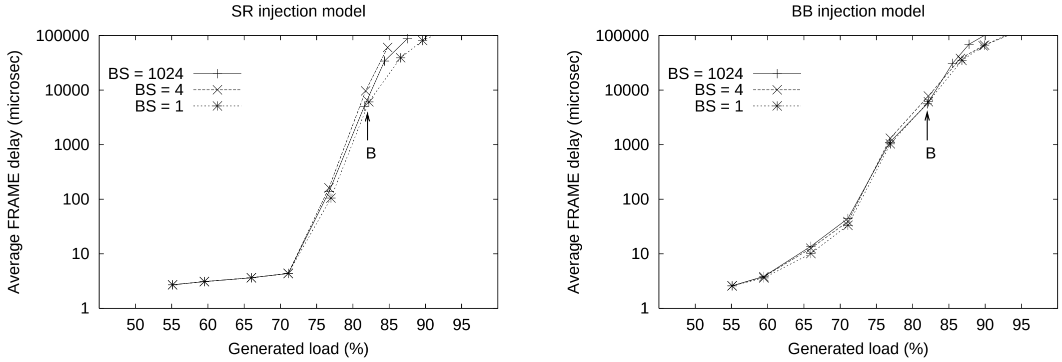


Figure 14. SIABP algorithm: Average frame delay since generation

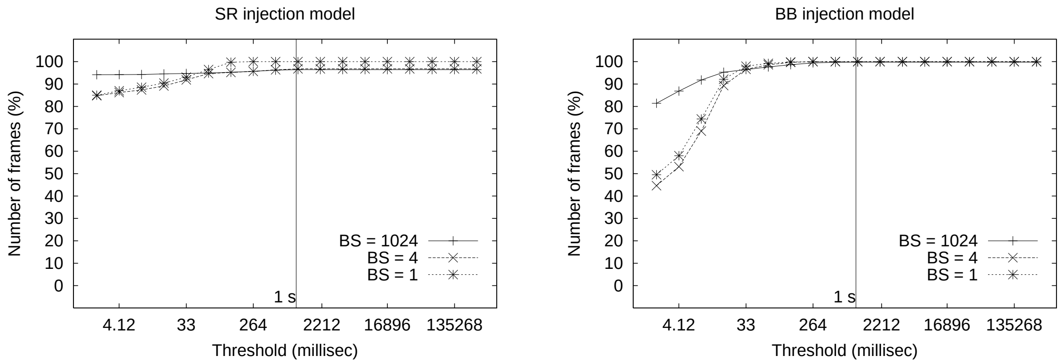


Figure 15. SIABP algorithm: Distribution of frame delay since generation for the B workload level

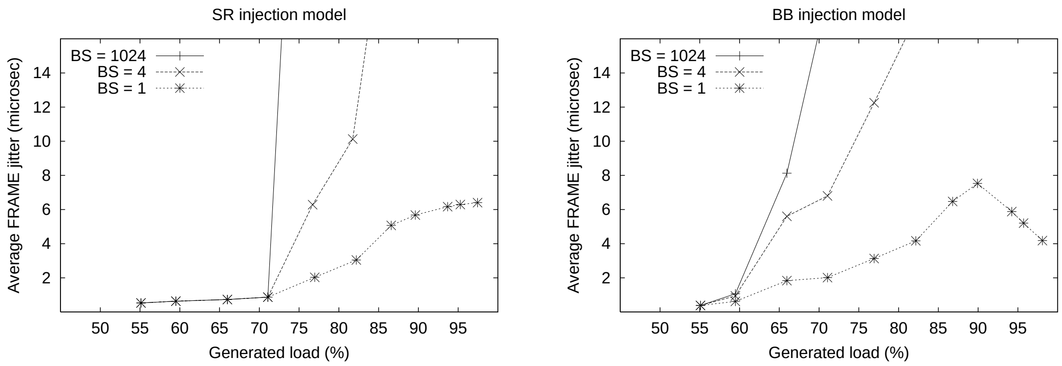


Figure 16. SIABP algorithm: Average frame jitter