University of Castilla-La Mancha

A publication of the **Department of Computer Science**

A Strategy to Reduce the Delay and Implementation Cost of QoS Support in Clusters

by

Alejandro Martínez, Francisco J. Alfaro, José L. Sánchez, José Duato

Technical Report **#DIAB-04-12-2** December 2004

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

Contents

A Strategy to Reduce the Delay and Implementation Cost of QoS Support in Clusters[1]

Alejandro Martínez, Francisco J. Alfaro, José L. Sánchez
Dept. de Informática
Escuela Politécnica Superior
Universidad de Castilla-La Mancha
02071- Albacete, Spain
{alejandro, falfaro, jsanchez}@info-ab.uclm.es

José Duato
Dept. de Informática de
Sistemas y Computadores
Universidad Politécnica de Valencia
46071- Valencia, Spain
jduato@gap.upv.es

[1]This work was partly supported by the Spanish CICYT under grant TIC2003-08154-C06 and Junta de Comunidades de Castilla-La Mancha under grant PBC-02-008.

Abstract

Current interconnect standards providing hardware support for QoS use 16 or even more virtual channels (VCs) for this purpose. However, most implementations do not offer so many VCs because it is too expensive in terms of chip area. We believe that the number of required VCs can be significantly reduced if the system is considered as a whole rather than each element being taken separately. Reduction in QoS support complexity can be achieved by eliminating certain redundancies in traffic processing. In particular, some of the scheduling decisions made at network interfaces can be easily reused at switches without significantly altering the global behavior of the schedulers. Therefore, the number of VCs per switch port can be reduced and flow control and buffer organization simplified.

In this paper we show that it is enough to use two VCs at each switch port: One of them for QoS traffic and another one for best-effort traffic. Results show that our proposal can achieve performance similar to the one for systems with a larger number of VCs but with a great reduction in processing delay and in the number of VCs required.

1 Introduction

The last decade has witnessed a vast increase in the variety of computing devices as well as in the number of users of those devices. In addition to the traditional desktop and laptop computers, new handheld devices like pocket PCs, personal digital assistants (PDAs), and cellular phones with multimedia capabilities have now become household names.

The main reasons for the widespread use of computing devices are the availability of cheaper and more powerful devices and, even more importantly, the huge amount of information and services available through the Internet. These services rely on applications executed in many servers all around the world. Due to the dramatic increment in the number of clients concurrently requiring these services, clusters of PCs have emerged as a cost-effective platform to implement these services and run the required Internet applications. These clusters provide service to thousands or tens of thousands of concurrent users. Many of these applications are multimedia applications, which usually present bandwidth and/or latency requirements [17]. These are known as quality of service (QoS) requirements.

In the next section we shall be looking at some of the proposals to provide QoS in clusters. All of them incorporate 16 or even more VCs, devoting a different VC to each traffic type. This increases the switch complexity and also prevents the use of these VCs for other purposes (for instance, to provide adaptive routing [13] or fault tolerance [18]). Moreover, it seems that, when the technology enables it, the trend is to increase the number of ports instead of increasing the number of VCs per port [16, 7]. This, in fact, is the trend followed nowadays by companies in their new products launched to the market [3, 1, 2].

In this paper we show that it is enough to implement two VCs in each switch port for the provision of QoS. One of these VCs is used for QoS traffic and the other one for best-effort traffic. This can be achieved by reusing in the switches some of the scheduling decisions made at network interfaces. Simulation results show that applications achieve a similar QoS performance, but using fewer VCs and with a reduced processing delay.

The remainder of this paper is structured as follows. In the following section the related work is presented. In Section 3 we present our strategy to reduce the number of VCs required for QoS support. Details on the experimental platform and the performance evaluation are presented in Section 4. Finally, Section 5 summarizes the results of this study and identifies directions for future research.

2 Related Work

The importance of network QoS is widely accepted by both the research community and the commercial service providers. However, the problem is that existing networks are not so well prepared for the new demands. Implementing QoS is a work in progress with multiple possible solutions competing against each other. Depending on the network architecture, different techniques have to be taken into consideration. Many research efforts are today performed around the main aspects related to QoS in different environments.

As mentioned earlier, the increasing use of the Internet has been the dominant contribution to the need of QoS. For this reason, it is not surprising that most of the studies are focused on delivering QoS on the Internet [12, 20]. Many of the services available through the Internet are provided by applications running on clusters. Therefore, the researchers are also proposing mechanisms for providing QoS on these platforms, as will be shown later in this section.

More recently, with the advent of different types of wireless technologies, wireless devices are becoming increasingly popular for providing Internet access to users. It is possible to transmit data with them but also voice, or executing multimedia applications for which QoS support is essential. The QoS mechanisms proposed for wired networks are not directly applicable to wireless networks. Therefore specific approaches are being proposed [9, 8]. Although there has been a lot of effort in the QoS field, in this paper we will focus on the clusters used to build Internet servers.

During the last decade several cluster switches with QoS support have been proposed. All of them incorporate VCs in order to provide QoS support. In these proposals, different scheduling algorithms are used to arbitrate between the different existing traffic flows, providing each with QoS according to its requirements.

Switcherland [11] was proposed in 1997 and is an approach similar to ATM. It uses packet switching, but without some of the overheads associated with ATM. However, this design is too simple and does not provide adequate treatment for VBR real-time traffic. Moreover, the crossbars used must be n times faster than the links, where n is the number of ports.

The Multimedia Router (MMR) [10] is a hybrid router. It uses pipelined circuit switching for multimedia traffic and virtual cut-through for best-effort traffic. Pipelined circuit switching is connection-oriented and needs one VC per connection. This is the main drawback of the proposal, because the number of VCs per physical channel is limited by the available buffer area and there may not be enough VCs for all the possible existing connections. Therefore, the number of multimedia flows allowed is limited by the number of VCs. Moreover, the scheduling among hundreds of VCs is a complex task.

MediaWorm [21] was proposed to provide QoS in a wormhole router. It uses a refined version of the Virtual Clock algorithm [23] to schedule the existing VCs. These VCs are divided into two groups: One for best-effort traffic and the other for real-time traffic. Several flows can share a VC, but 16 VCs are still needed to provide QoS. Besides, it is well known that wormhole is more likely to produce congestion than virtual cut-through. In [22], the authors propose a pre-emption mechanism to enhance MediaWorm performance, but in our view that is a rather complex solution.

InfiniBand was proposed in 1999 by the most important IT companies to provide present and future server systems with the required levels of reliability, availability, performance, scalability and QoS [14]. Specifically, the InfiniBand Architecture (IBA) proposes three main mechanisms to provide the applications with QoS. These are traffic segregation with service levels, the use of VCs (IBA ports can have up to 16 VCs) and the arbitration at output ports according to an arbitration table. Although IBA does not specify how these mechanisms should be used, some proposals have been made to provide applications with QoS in InfiniBand networks [6].

Finally, PCI Express Advanced Switching (AS) architecture is the natural evolution of the traditional PCI bus [4]. It defines a switch fabric architecture that supports high availability, performance, reliability and QoS. AS ports incorporate up to 20 VCs that are scheduled according to some QoS criteria. In the AS specifications, three possible arbiters are proposed, one of them being table-based.

Today's trends in traffic arbitration (IBA, AS) seem to use table-based Weighted Round Robin (WRR) [15] for the arbitration of the service levels. The idea is to fill in a table with rows (slots) indicating VCs. When arbitration is needed, the table is cycled through sequentially and a packet is transmitted from the VC indicated in the current table entry. The bandwidth guarantees are achieved by allocating a suitable number of slots for each VC and by assigning a weight to each slot, where the weight is the number of data units to be transmitted when a slot is processed. On the other hand, deadline guarantees are provided by the reservation of slots at a certain distance in the table, thus bounding the number of data units from other VCs that can be transmitted between two consecutive transmissions from a given VC.

The table-based WRR is appropriate for bandwidth guarantees, but not for deadline guarantees. Due to the relationship between both requirements, if we want to provide strict deadline guarantees and the deadline is short, we must reserve many rows of the table and thereby also provide a high bandwidth [5]. On the other hand, priority based algorithms are much more suitable for deadline requirements, although they do raise two problems: There is no way to reserve bandwidth and they may cause starvation. However, both issues can be solved if we use a priority based algorithm with Connection Admission Control (CAC) to reserve bandwidth. The provision of bandwidth guarantees is obvious and starvation is averted because we are not allowing more traffic than we can handle. Of course, there would be no reservation for best-effort traffic, apart from a possible minimum bandwidth specifically designed for this purpose. Moreover, as we are using a priority system, we can guarantee that best-effort traffic would never interfere with QoS traffic.

In summary, all of the proposals studied use VCs to provide QoS support. Most of them use 16 VCs, but the MMR uses hundreds. Such a large number of VCs would require a significant fraction of chip space and would make packet processing a more time-consuming task. In all cases, the VCs are used to segregate the different traffic classes. Therefore, it is not possible to use the available VCs to provide other functionalities like adaptive routing or fault tolerance when using all the VCs to provide QoS support.

3 A Strategy to Reduce the Number of VCs for QoS Support

In this section we present our proposal to use only two VCs in the switch ports to provide QoS. In this way, packet processing can be speeded up because the scheduler has to consider a smaller number of VCs, the switch complexity can be reduced and/or the remaining VCs could be used for other purposes. This reduction of the complexity and the associated speed-up do not imply a reduction in functionality, because it is based on the elimination of redundancies. The basic idea consists in reusing in the switches some of the scheduling decisions made at network interfaces.

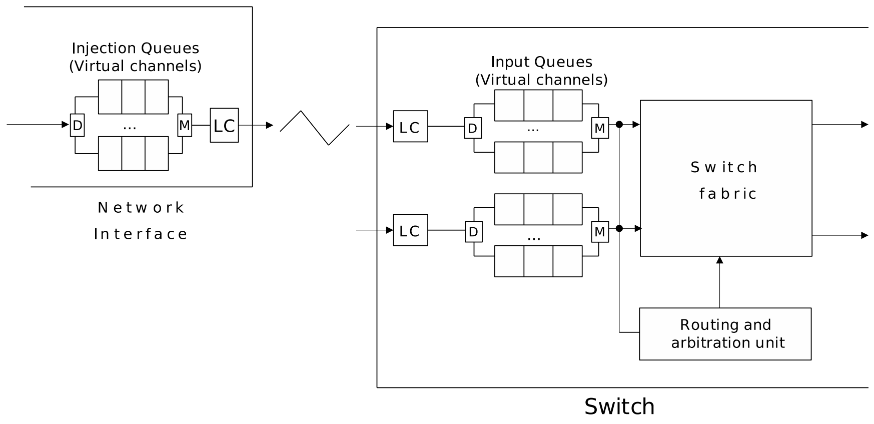


Figure 1: QoS support at the network interface and the switch.

Figure 1 shows an example of a network interface that is connected to a switch. Note that at both the network interface and the switch input port, there are several VCs. Therefore, when a packet arrives at the switch, the header is analyzed and the packet is then usually stored in a VC according to the flow or class to which it belongs to.

When packets arrive at different ports of the switch and have to cross it, conflicts may arise because several packets might request the same output port. These conflicts are solved by an arbiter that considers the QoS requirements of the traffic. Our intention is to make it possible to implement algorithms in hardware that are more powerful than WRR, especially when we do not have as many VCs as service levels. In order to accomplish this we will focus on reusing part of the scheduling decisions made at the network interfaces. This is achieved by keeping the packet arrival order, instead of splitting the incoming flow into VCs, and thereby losing the order in which the packets arrive at the switch ports. In this way we are both saving work and simplifying the design. Note that traditional scheduling techniques that consider priorities require comparing the priority of all the packets in each queue. Furthermore, we made the minimum separation into VCs to guarantee the necessary QoS.

Our work is based on the following observation. If QoS traffic does not require more bandwidth than available at any link in the corresponding path then:

- Queuing delays for QoS traffic will be very short, and therefore, the packet ordering established at network interfaces does not need to be changed at any switch in the path.

- When two flows arriving at different input ports of a switch request the same output port, the scheduler just needs to compare the priorities of one packet from each flow. Taking into account that packets are ordered according to their priority, this means that the scheduler only needs to consider the first packet from each queue. Note that we are assuming that there is no link oversubscription.

- The remaining bandwidth in each link (i.e. the one not used by QoS traffic) can be filled by best-effort traffic.

Thus, our proposal consists in ordering the QoS traffic at the network interfaces and guaranteeing that it will not consume all the bandwidth. Then we fill the gaps with best-effort traffic, which is not ordered. For that reason we use two VCs at the switches, one for QoS traffic, maintaining the arrival order, and one for best-effort traffic so that it does no interfere with the QoS traffic. Note that although we assume that QoS traffic does not oversubscribe any link, no assumption is made about best-effort traffic. Thus, if we did not separate QoS traffic and best-effort traffic, the total bandwidth demand for a given output link could exceed the available bandwidth.

In order for this proposal to be effective we need several assumptions. The first assumption we make is that a static priority criterion exists to order packets. In this way, every packet would be stamped with a service level. This is necessary because if we are to maintain the incoming order, it is necessary for this order to be correct. The second assumption is that there must be a CAC for the traffic with QoS requirements so that no link is oversubscribed. This requirement is needed to solve the problems associated with strict priority systems, that is, bandwidth guarantee and starvation. Besides, this is a solution that is not more expensive than table-based WRR.

Let us suppose that several packets arrive at a switch from a network interface. If the interface implements a priority based arbiter, the first packet should be the one with the highest priority. So, instead of separating the packets among several VCs according to their traffic class, we put them all in the same queue in the arrival order. Later, when the switch must decide which packets should leave, it will seek in the input queues. In this case, it is only necessary to look at the first packet in each queue. The reason is that if it is placed before the others in the FIFO queue, it is because it had a higher priority when it left the network interface.

If the previous reasoning were always true, then the behavior would be the same as when separating the traffic into VCs and applying priority arbitration. However, the network interface cannot arbitrate among all the packets, but only over those it holds at a given moment. Therefore, when no more high-priority packets are available, a low-priority packet will be transmitted. It is important to note that as best-effort packets have their own VC, they will never interfere with QoS traffic. So, coming back to our example, a low-priority packet is still a QoS packet. If this low-priority packet has to wait at a switch input queue, and other packets with higher priority are transmitted from the network interface, they would be stored in the same VC as the low priority packet, and be placed after it in the FIFO queue. Thus, the arbiter would penalize the high-priority packets, because they would have to wait until the low-priority packet is served. But this situation has a small impact on performance because there is bandwidth reservation for QoS packets. This means that all the QoS packets will flow with short delay, so the high-priority packet is guaranteed a short delay.

Summing up, our proposal consists in reducing the number of VCs in the switch ports that are needed to provide flows with QoS, making the network elements more cooperative. Instead of having a VC per flow or service level, we propose to use only two VCs: one for QoS packets and another one for best-effort packets. Moreover, some of the scheduling decisions performed at network interfaces are reused at switches. Furthermore, the arbiter only needs to look at the first packet in each VC, thus reducing the scheduling time. In order for this strategy to work, we must guarantee that there is no link oversubscription by using a suitable CAC strategy.

4 Performance Evaluation

Here we evaluate the proposal presented in the previous section. For this purpose a detailed simulation tool has been developed. Firstly, we describe the simulation parameters and the modeling considerations we have used in the simulations. Secondly, we show and analyze the results obtained.

4.1 Simulated architecture

The network used to test the proposals was a multi-stage interconnection network (MIN). However, our proposal is valid for any network topology, including both direct networks and MINs. The switches use a combined input and output buffer architecture, with a crossbar to connect the buffers. Virtual output queuing was implemented to solve the head-of-line blocking problem at the switch level. The parameters of the network elements used in this performance study are given in Table 1. We simulated a network with 32 nodes. The remaining parameter values are typical for this kind of study. Note that we are assuming some internal speed-up, as is usually the case in most commercial switches.

In Table 1 we also show the amount of memory necessary to implement each VC. We propose to use 4 kbytes because that permits to store two whole packets. Note that switches using our proposal save memory (and thus chip area) at the ports by a factor of 8 compared with the typical 16 VCs.

The arbitration time depends on the number of packets to be processed: A base time of 10 ns plus 2 ns per packet. Furthermore, as the memory bandwidth of the buffers is limited, we assume that when arbitration is in progress, the buffers cannot send packets either through the crossbar or the output link. However, it is possible to

Switch size	8 ports
Packet size	128 to 2048 bytes
Header size	8 bytes
Control message size	8 bytes
VC size	4 Kbytes
Channel bandwidth	1 Gbps
Crossbar bandwidth	2 Gbps
Number of network interfaces	32

Table 1: Simulation parameters.

receive packets during arbitration. We believe these assumptions are very reasonable, taking into account how switches are designed nowadays. With this data we can show that the typical switch with 16 VCs needs at most 42 ns to arbitrate an output port. However, with our proposal, the same task would take at most 14 ns. This means a speed-up of 3 in the worst case.

4.2 Simulation conditions

The workload was composed of 12 different service levels, each with increasing priority, such that level 0 is the highest priority and level 11 is the lowest. Moreover, we also assumed that levels 8-11 were best-effort traffic, and thus were not subject to bandwidth reservation.

The traffic mix that was injected into the network was composed by 8.33% of traffic from each service level, that is, the same amount for each category. The destination pattern was uniform and the packets were generated according to different distributions:

- Service levels from 0 to 3: Audio traffic composed of CBR 64 kbps point-to-point connections. The packet size was fixed to 128 bytes.
- Service levels from 4 to 7: Video traffic composed of MPEG-4 750 kbps sequences transmitted on point-to-point connections. The packet size varied up to 2 kbytes.
- Service levels from 8 to 11: Best-effort traffic with self-similar pattern. The packet size was 2 kbytes.

There are three traditional QoS indices for this kind of evaluation. The first of these, latency, is the time taken by a packet to arrive at its destination. The throughput is the number of delivered messages per time unit. However, we will represent it as a percentage of injected traffic. Finally, it is also usual to measure jitter, which is the difference between the delays of packets. This metric is only meaningful among the packets of a connection, and so, it is only measured for audio and video traffic. No packets are dropped because we use credit based flow control.

4.3 Simulation results

In this section, the performance of the proposal presented in this paper will be shown. We have tested several scenarios, varying several switch parameters:

- Priority arbitration with 12, 3 and 2 VCs.
- WRR table arbitration with 12 VCs.
- Priority arbitration with 12 VCs and a fixed ideal arbitration time.

The implementation with 12 VCs represents the traditional approach, while the implementations with 2 and 3 VCs use our proposal to avoid sweeping the entire queues when arbitrating. As indicated above, we have compared our proposal (with 2 and 3 VCs) with the traditional approach (with 12 VCs) using two different arbiters (priority-based and WRR). We decided to perform the test with 12 VCs because this number matches the number of service levels, therefore excluding the possibility of order errors. In other words, it is not possible for a low-priority packet to be stored in a queue before a high-priority one because there is a queue for each priority level. Note that this favors the traditional approach because, as a consequence of this, it is not necessary to sweep the entire queue for the traditional approach. We also chose 3 VCs to be tested because this number matches up the traffic classes (audio, video and best-effort), and so, there would only be order errors inside each class. Furthermore, packets in the same traffic class have similar characteristics and requirements.

In order to accurately point to the sources of delay in the different scenarios, we also tested an ideal switch design, with 12 VCs. However, in this case the arbitration time was fixed to 10 ns, whatever the number of packets to be arbitrated.

It is important to note that the proposal in this paper does not aim at achieving a higher performance but, instead, at drastically reducing buffer requirements while keeping performance.

Finally, in addition to the uniform packet destination distribution, we also tested a hot-spot distribution, disabling the CAC. Our aim is to test our proposal in the worst conditions to study how it performs and to confirm experimentally the necessity of a CAC.

Our proposal aims to reduce the VCs of the switches ports. However, we have modeled an entire network, including the network interfaces at the end points. These interfaces must implement a 12 VC priority arbiter in order of our proposal to work. This is necessary because our switches can reduce their complexity because they reuse part of the scheduling decisions made at the networks interfaces. So, in all the tests we have conducted, the network interfaces have remained the same, with 12 VCs and priority arbitration. Furthermore, we have supposed an infinite memory capacity at the interfaces. This is because these devices typically implement a big memory and have access to memory hierarchy. However, when the network interface needs to perform arbitration, it only considers the first packet of each VC.

4.3.1 Priority arbiter

In this case, we have varied the number of VCs in the switches, but in all the cases the arbitration is priority based. The most complex design uses 12 VCs and so, has the same number of VCs as service levels, and is thus order error free. In other words, it is not possible for a low-priority packet to be stored in a queue before a high-priority one because there is a queue for each priority level. The other two proposals do not have as many VCs as service levels and can thus suffer the consequences of these order errors, but they benefit from a simpler arbiter that needs less time and memory bandwidth to operate.

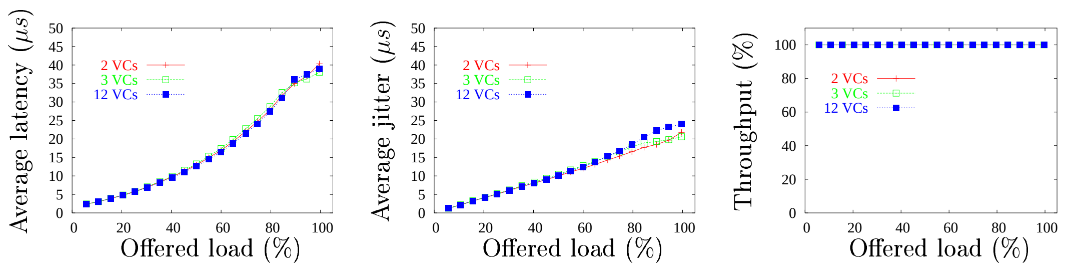
Figure 2 (a) shows the performance of service level 0 (audio traffic) for the different priority arbiters. It can be seen that the three cases achieve very similar performance in the two metrics. In other words, the use of a smaller number of VCs together with an arbiter that only looks at the first packet in each VC does not degrade performance. The small differences between the *Priorities 2 VCs* and *Priorities 12 VCs* cases are produced by two factors: The absence of order errors in the 12 VCs model and the shorter arbitration in the 2 VCs model. Obviously, the same goes for the *Priorities 3 VCs* model. Moreover, the proposals with 2 and 3 VCs are slightly better in terms of jitter. The reason for this is, again, the shorter arbitration time.

In the SL 3 (audio traffic), shown in Figure 2 (d), it is the *Priorities 2 VCs* and *Priorities 3 VCs* proposals which have the best performance. This is because the model with order errors gives a better-than-necessary QoS to the lowest priority audio levels. Note that the difference is really anecdotic and the best-effort traffic is which suffers this error. This results show that the use of a smaller number of VCs together with an arbiter that only looks at the first packet in each VC does not degrade performance. The other audio SLs achieve a performance at midway of the two SLs shown here. That is a good point in favor of our proposal.

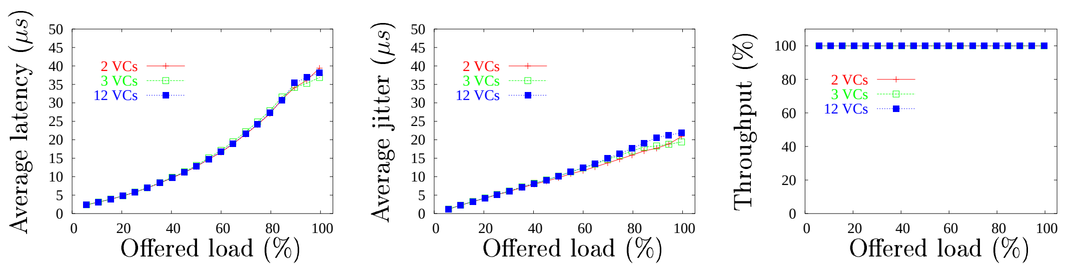
Figure 3 (a) shows the average latency and jitter for SL 4 (video traffic). In this case there is a clear difference between the *Priorities 2 VCs* and case and the other two cases. This difference is due to the smaller buffer space. However, note that: First, the audio traffic, which is the most latency-sensitive, achieved an ideal performance; secondly, the video traffic is not so latency sensitive; and finally, the performance that it is achieved is very acceptable because latency and jitter keep in low values and there is not a saturation point for video traffic. It should be noted that our proposal is based on the assumption that QoS traffic will not oversubscribe link bandwidth. However, bandwidth reservation for video traffic is based on average bandwidth requirements, thus suffering a small performance degradation during peak traffic.

Even if these results are not good enough, our other proposal with 3 VCs achieves a performance very similar to the traditional arbiter with 12 VCs. In this case, the reduction of VCs is still noticeable.

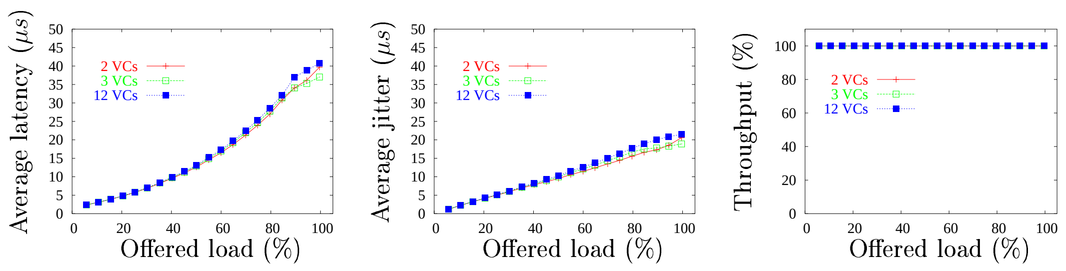
For the rest of the video SLs, the performance is very similar to that for SL 4. The only difference is that the average latency increases for the less priority SLs. However, in neither case it reaches a saturation point, so the performance is always acceptable,



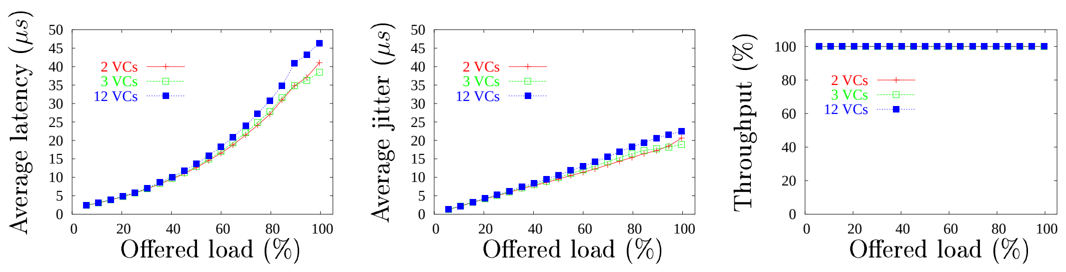
(a) Audio 0



(b) Audio 1



(c) Audio 2

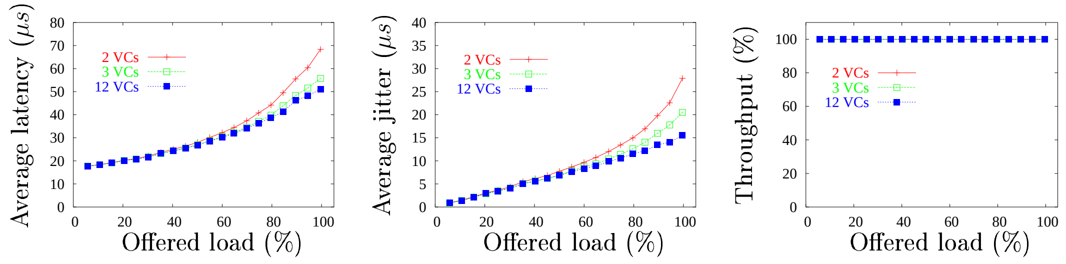


(d) Audio 3

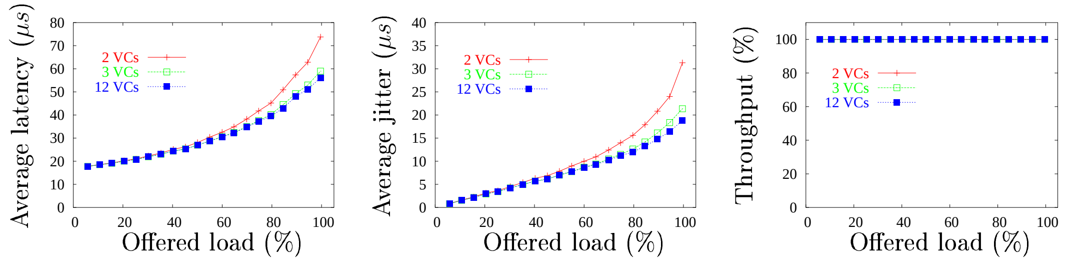
Figure 2: Latency, jitter and throughput for audio traffic for a priority arbiter.

even with the reduced number of VCs of our proposal.

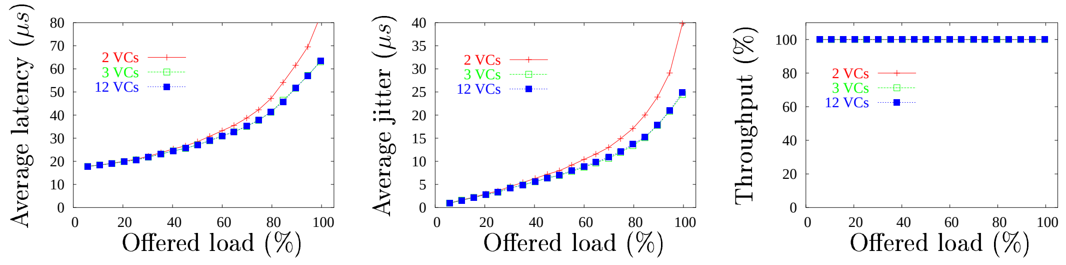
Figure 4 (a) shows the average latency and throughput for SL 8 (best-effort traffic). Although the performance is very similar, the *Priorities 2 VCs* and *Priorities 3 VCs* cases provide a degraded performance. Note that this SL would translate in some kind of preferential best-effort and thus, it would be reasonable that this kind of traffic



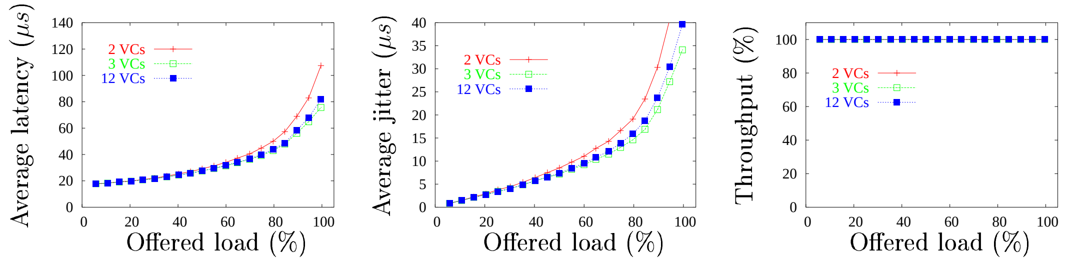
(a) Video 0



(b) Video 1



(c) Video 2



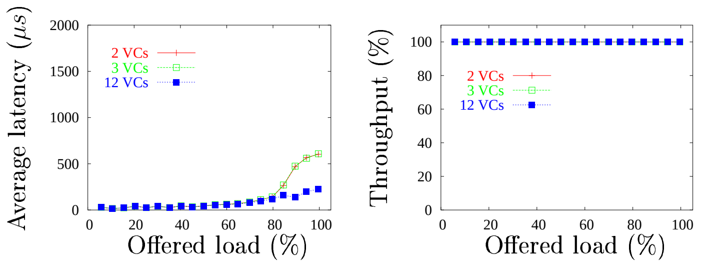
(d) Video 3

Figure 3: Latency, jitter and throughput for video traffic for a priority arbiter.

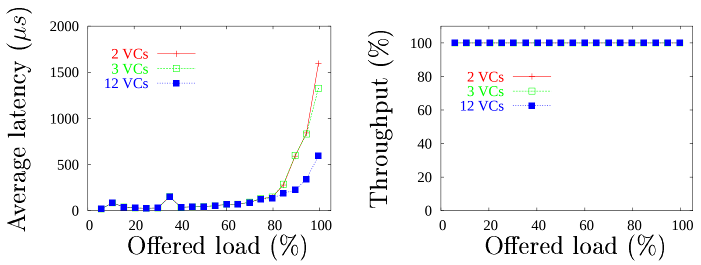
achieved a good performance. As we can see in the figure, the three models achieve a 100% throughput and the latency is not too big in neither of them. So, although the performance is slightly worse with our proposal, note that it is still an acceptable result. Also note that our proposals are very close to the 12 VC one without most of the complexity associated with such a proposal.

In Figure 4 (d) we can see the average latency and throughput for SL 11. This is the less priority SL of the best-effort and it would model back-up traffic and similar issues. Note that the three models have a very similar performance. The other two intermediate best-effort SLs (SLs 9 and 10) achieve in both cases a 100% throughput and a similar latency. In other words, we achieve a similar performance although we have reduced the number of VCs and the buffer space. Note that although we are putting all the best-effort traffic in a single SL, we are still able to differentiate among the different SLs of this category. The explanation is that all the reason we gave in the Section 3 for the QoS traffic are valid too for the best-effort traffic. That is, we are reusing the scheduling decisions made for the best-effort traffic.

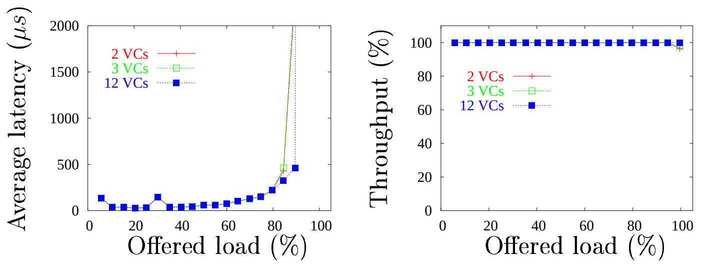
We can conclude at this point that our proposal is able to provide an adequate QoS to multimedia traffic. The switch model we have proposed reduces the number of VCs, also reduces the associated memory space and the arbitration time. Moreover, we achieve not only an adequate performance, but also a performance very similar to that obtained with the more complex traditional arbiter.



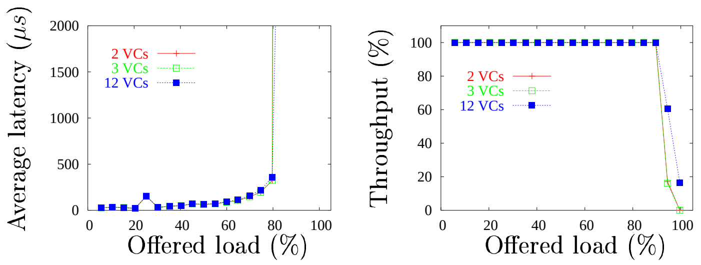
(a) BE 0



(b) BE 1



(c) BE 2



(d) BE 3

Figure 4: Latency and throughput for best-effort traffic for a priority arbiter.

4.3.2 Table arbiter

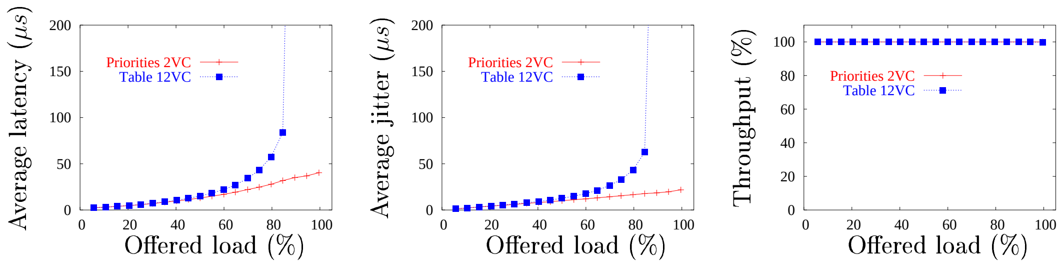
In this section, we have tested the table-based WRR arbiter to compare its performance with the results obtained with our proposals. At the introduction section, we claimed that the table-based WRR arbiters are not suitable for providing latency guarantees if the required deadline is short. We have modeled an arbiter of this kind and provided it with a suitable arbitration table. As we are injecting the same amount of traffic from each service level, there must appear the same number of entries of each SL. The only latency guarantee we can provide is that the arbiter will provide a packet of each SL for every 12 packets (remember that there are 12 SLs). If we needed to provide stricter latency guarantees, then not all the bandwidth of the links would be assigned because we would take too many entries of the table. However, in the results that we will show we will see that the different SLs achieve a differentiated service. This is due to the network interfaces, which in all the cases implement a 12 VCs priority arbiter.

Figure 5 (a) shows the performance of service level 0 (audio traffic). The first thing that is clear is that this kind of arbiter can guarantee bandwidth (100% throughput) but the obtained latency and jitter are not so adequate. This is because it is not possible to reserve slots in the table to provide the latency guarantees and at the same time reserve bandwidth for the rest of the traffic (video and best-effort). Note how our proposal clearly beats the performance of the table-based WRR arbiter, even with less VCs and buffer space. For the rest of the audio SLs, the results are very similar.

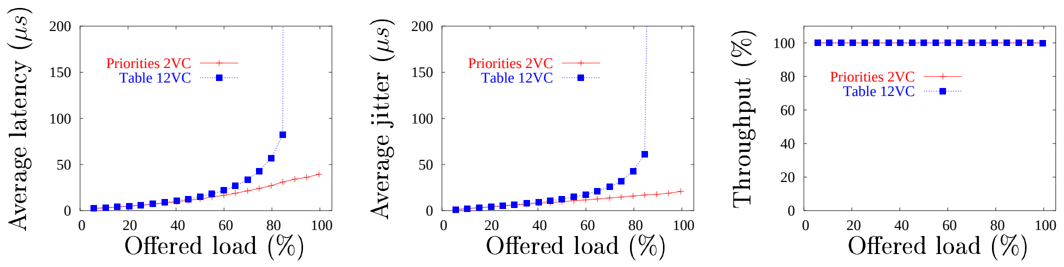
Figure 6 (a) shows the service level 4 (video traffic) results. In this case the performance is very similar according to latency for the two models. However, our proposal is clearly better in terms of jitter. This is due to the reduced arbitration time we have with only 2 VCs to arbitrate and for the strict priority politic we apply, which effectively segregates QoS traffic from best-effort traffic. The other video SLs achieve very similar results.

Figure 7 (a) shows the service level 8 (best-effort) traffic performance. It is important to note that the most priority best-effort traffic has a better performance with table-based WRR than with a priority arbiter. This is because the former assigns a minimum bandwidth even for the lowest priority flows, while our proposal uses a strict priority system. However, the difference is not very important and the performance is still acceptable, as we explained in the former section.

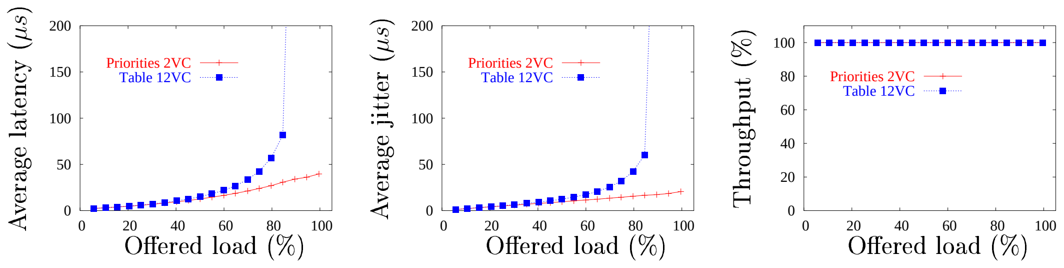
In Figure 7 (d) we show the service level 11 (best-effort) traffic performance. Remember that this SL was the least priority SL. In this case, the performance is very similar for the two arbiters. The rest of the best-effort SLs achieve a similar performance in the two cases.



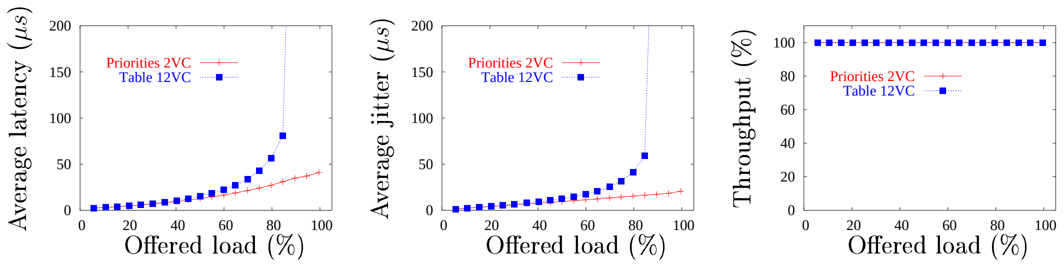
(a) Audio 0



(b) Audio 1

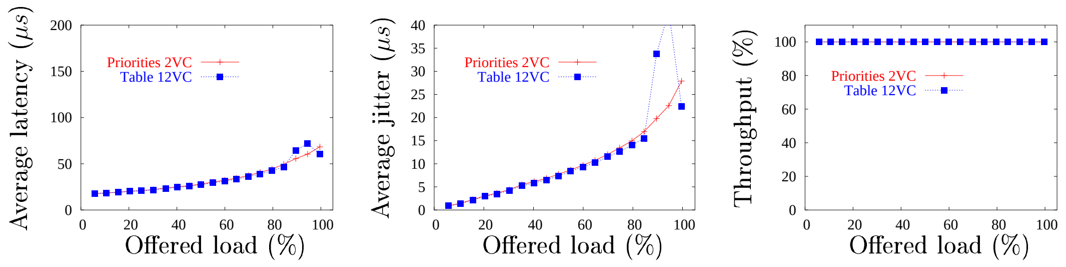


(c) Audio 2

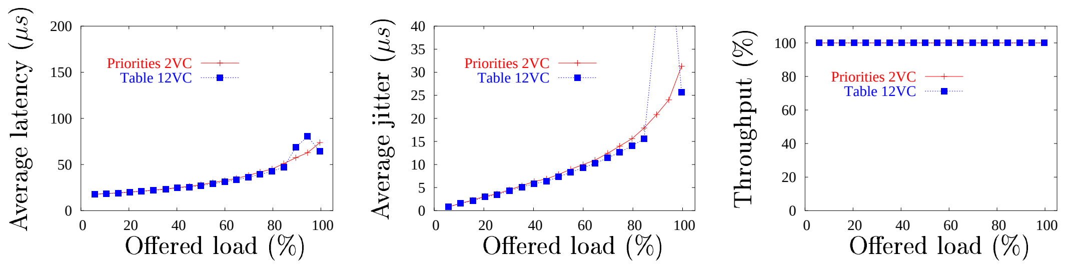


(d) Audio 3

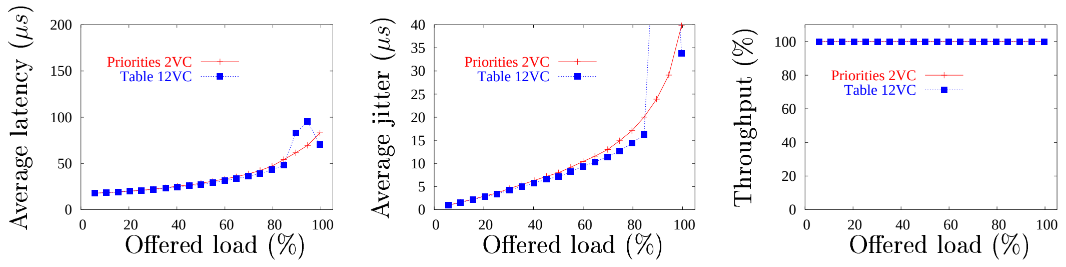
Figure 5: Latency, jitter and throughput for audio traffic for a table arbiter.



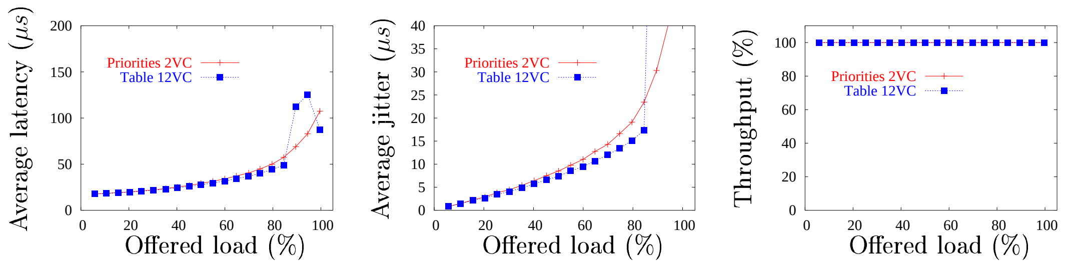
(a) Video 0



(b) Video 1

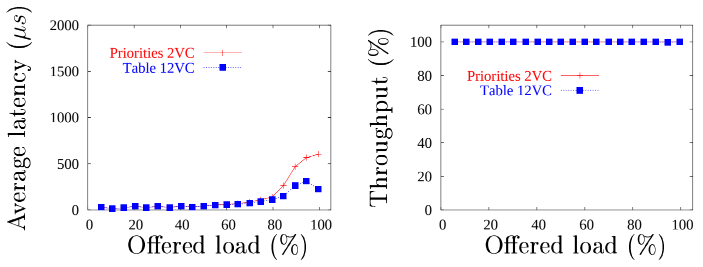


(c) Video 2

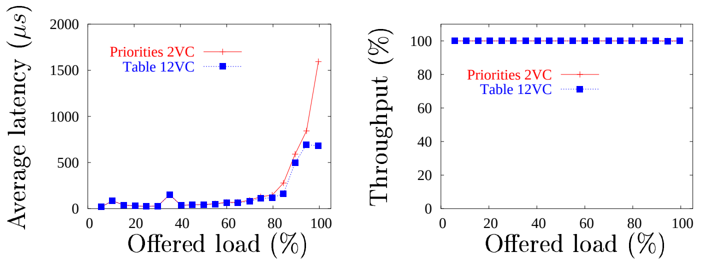


(d) Video 3

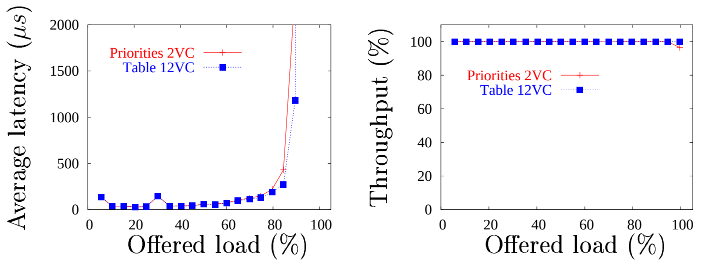
Figure 6: Latency, jitter and throughput for video traffic for a table arbiter.



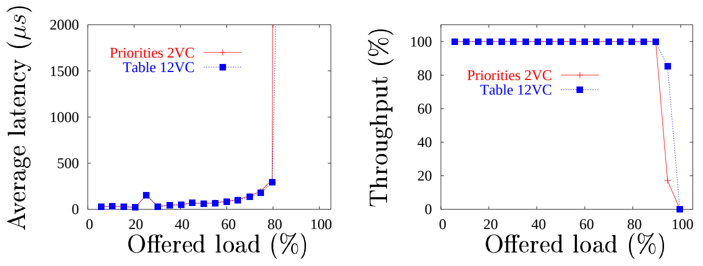
(a) BE 0



(b) BE 1



(c) BE 2



(d) BE 3

Figure 7: Latency and throughput for best-effort traffic for a table arbiter.

4.3.3 Ideal architecture

In this section we shall study the source of delay in the priority model. To accomplish that objective, we have defined and tested another architecture. It is noted as “Ideal" in the figures and implements priority arbitration with 12 VCs. However, it has a constant arbitration time of 10 ns, whatever the number of packets to be processed. That is the reason why we call it “Ideal". We have tested that architecture in order to accurately point to the sources of delay by comparison with this ideal architecture.

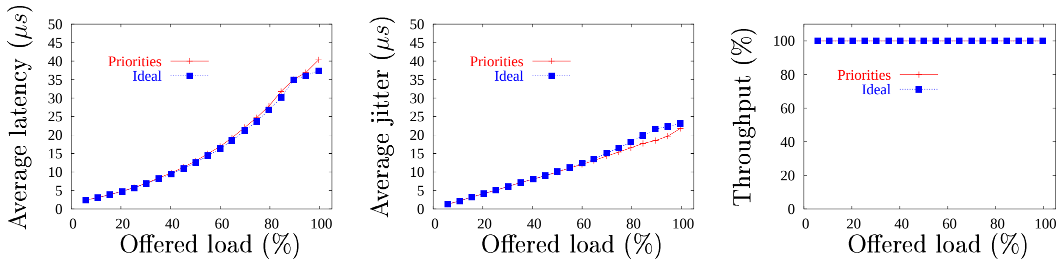
Figure 8 (a) shows the service level 0 (audio traffic) performance. We can see that it is very similar in all the cases. We can appreciate the amount of delay introduced by order errors in the priorities case as the difference between the *Priorities 2 VCs* and the *Ideal 12 VCs* cases. Obviously, this difference is neglect able and it confirms that is is not necessary to separate the incoming traffic that arrives at a switch into VCs. It is wiser to reuse the scheduling decisions taken before.

For the service level 3 (audio traffic), the results are very similar to those we shown in the Section **??**. Again, our proposal achieves a better performance. The other audio SLs also achieve a performance at midway of the SLs 0 and 3.

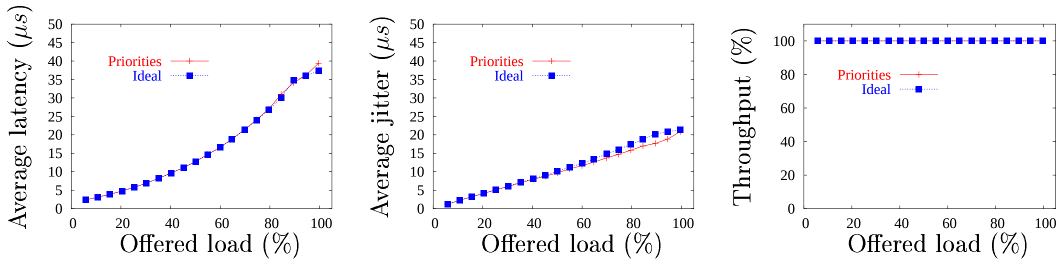
In Figure 9 (a) we can see the average latency and jitter for service level 4. We can appreciate again that our proposal does not achieves the ideal performance. However, it is not a problem, as we have discussed before. Furthermore, the issue is solved with our proposal with 3 VCs. The rest of the video traffic results are very similar.

Figure 10 (d) shows the average latency and throughput for service level 11. Note that the ideal arbiter achieves a performance very similar to that we obtain with our proposal. The other best-effort SLs achieve a very similar performance in both cases. However, in the SL 8 there is a small performance degradation, as we discussed in Section **??**

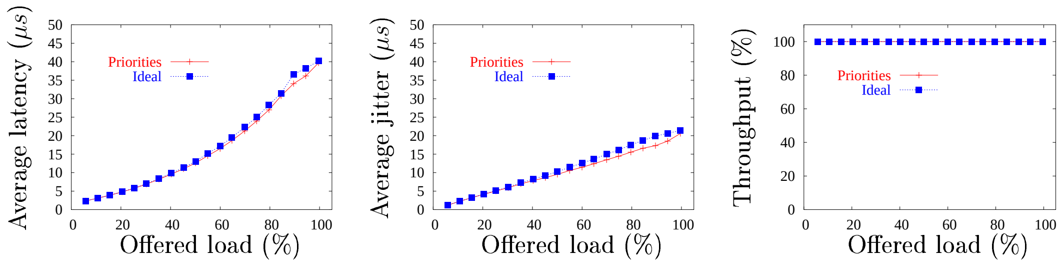
According to the results we have obtained, we can conclude that our proposal can provide an adequate QoS. We only need 2 VCs at the switches, 8 times less memory at the ports and a simple arbitration algorithm. It reduces the peak arbitration time on the output ports from 42 ns to only 14 ns. This is simpler than today’s trends but as powerful as the more complex arbiters with many more VCs.



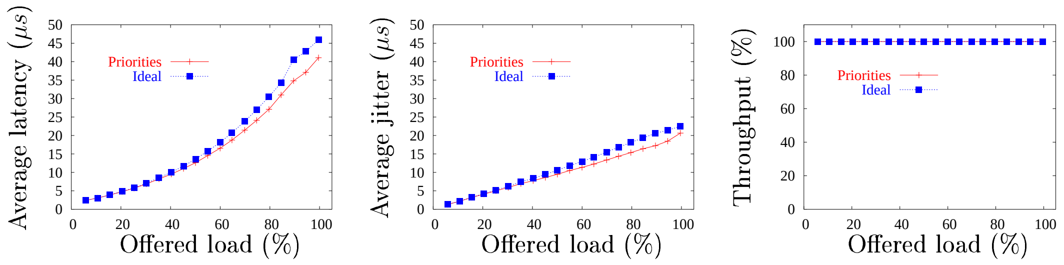
(a) Audio 0



(b) Audio 1

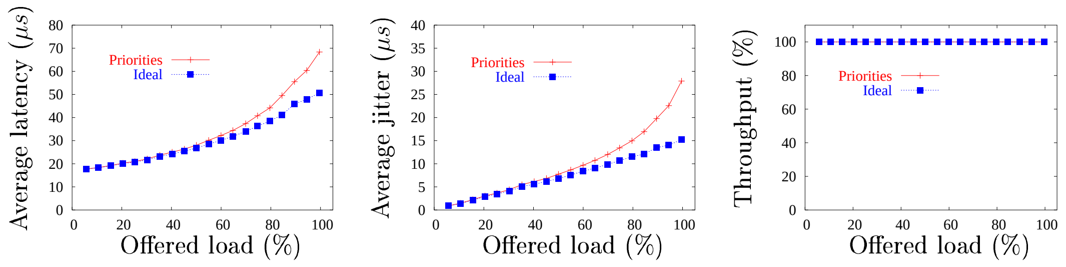


(c) Audio 2

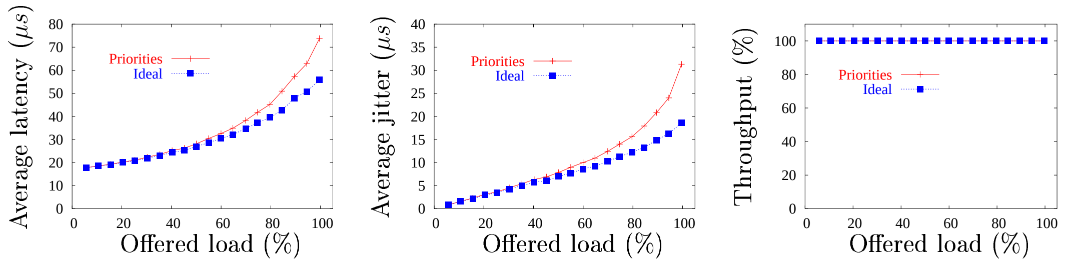


(d) Audio 3

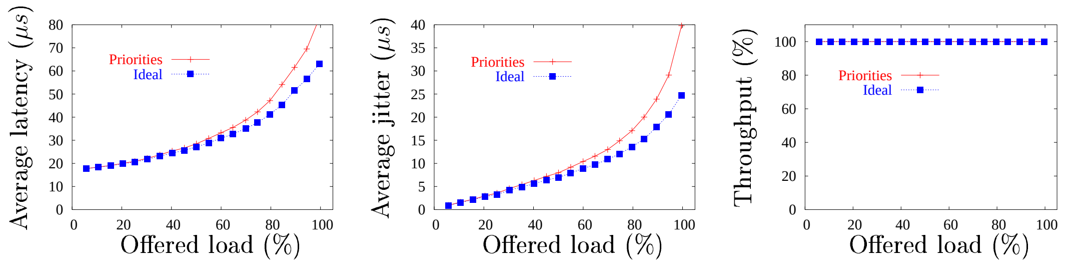
Figure 8: Latency, jitter and throughput for audio traffic for ideal arbiters.



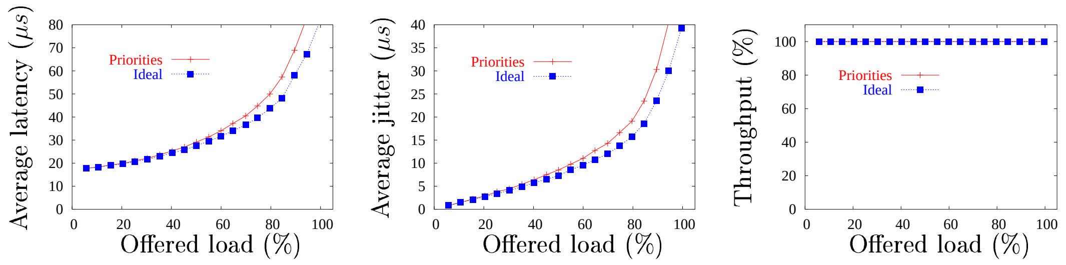
(a) Video 0



(b) Video 1

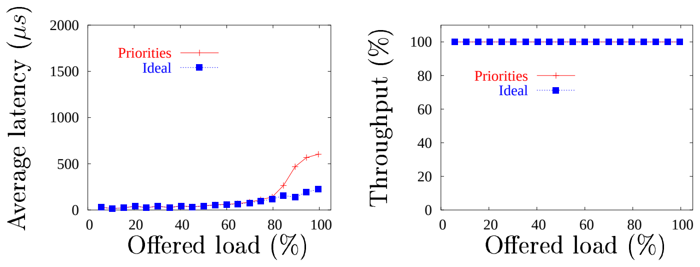


(c) Video 2

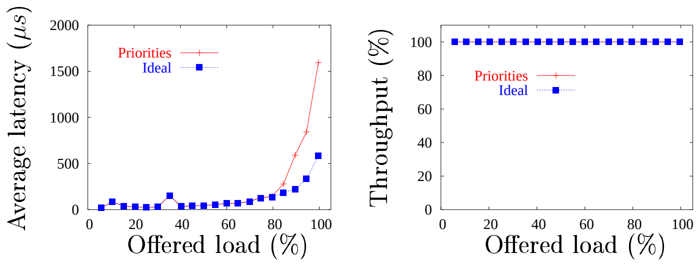


(d) Video 3

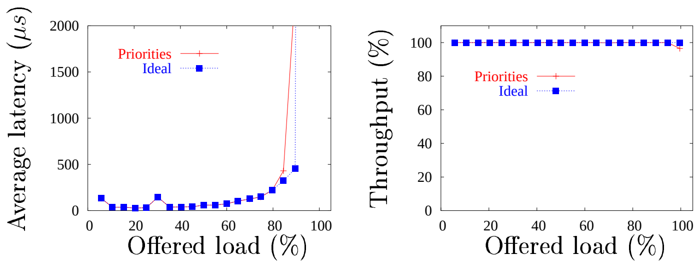
Figure 9: Latency, jitter and throughput for video traffic for ideal arbiters.



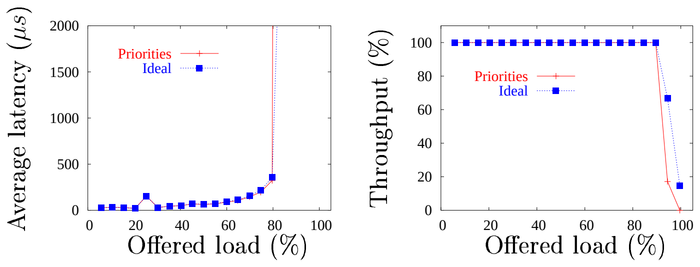
(a) BE 0



(b) BE 1



(c) BE 2



(d) BE 3

Figure 10: Latency and throughput for best-effort traffic for ideal arbiters.

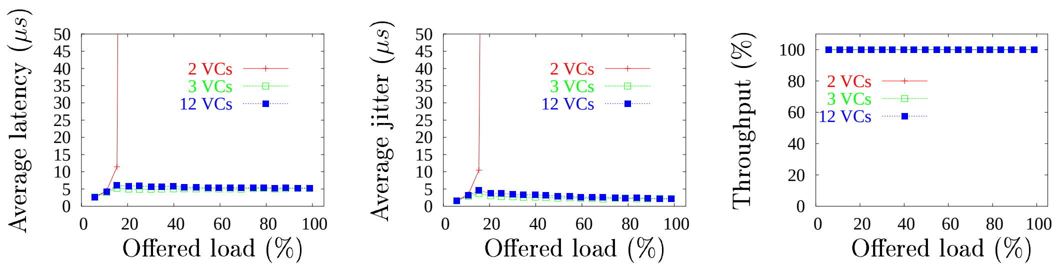
4.3.4 Hot Spots scenario

To complete our study, we have removed the CAC and tested a hot-spots scenario. This is the worst case for our proposal. We have defined randomly two destinations to be the hot-spot nodes which have received 10% of each traffic source. That means a 160% utilization of the links that connect these hot-spot nodes to the network. This is possible because we have given up the CAC. Obviously, not all the traffic, even the one with QoS requirements, could be accommodated. This leads to a congestion situation that affects the entire network, not only the traffic targeted to the hot-spot nodes. Moreover, the global utilization of the network elements is very much lower than that obtained with a uniform distribution of the destinations.

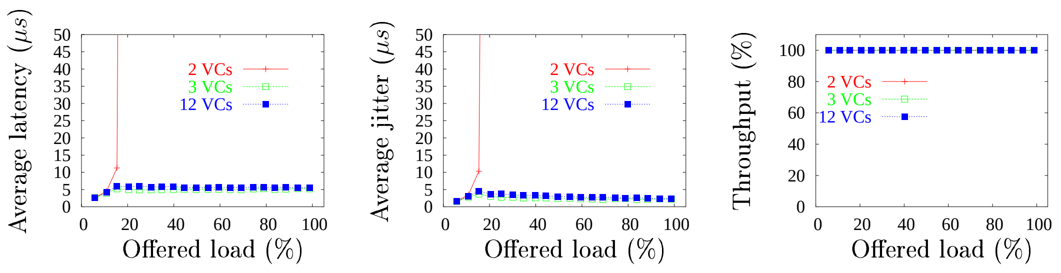
Average Figure 11 shows the performance of audio traffic. It is immediately apparent that the 2 VC proposal becomes congested very soon, while the 3 and 12 VC cases achieve an almost ideal performance. The reason is that although the hot-spot nodes cannot accept all the QoS traffic, they are capable of coping with the audio traffic. As the 3 and 12 VC models can effectively separate the audio traffic from the rest, this traffic is not affected by the congestion. That shows how important the CAC is for our proposal. As in the 2 VC model all the QoS traffic uses only one VC, we must be sure that there is enough bandwidth for it, or otherwise we will not be able to provide the adequate QoS. Note, however, that even with this traffic pattern, which is the most unfavorable for our proposal, all the audio traffic is also able to arrive at its destination in the 2 VC proposal (as was stated, the throughput is 100%). It receives a differentiated service from the low-priority video traffic (Figure 12), which at a certain point ceases to arrive, simply because there is not enough bandwidth in the channels to accommodate all the QoS traffic. It is also worth noting that our proposal of 3 VCs achieves a performance very similar to the typical arbiter with 12 VCs (with buffers 4 times bigger).

Figure 12 shows the average latency and throughput for video traffic SLs. In this case, we can see that the 2 and 3 VC models achieve a very similar performance. Note that only the 12 VC model can differentiate between the video levels, accommodating those with the highest priority as far as possible. We can see that around 30% of load, the video 0 (SL 4) still achieves an acceptable latency, while video 3 (SL 7) is clearly congested. The SLs 5 and 6 achieve a performance halfway between these two. Note that at a certain point there is not enough bandwidth for all the video traffic and its throughput begins to decline. However, these results are very similar in the three cases, with a slightly better behavior in the 12 VC model.

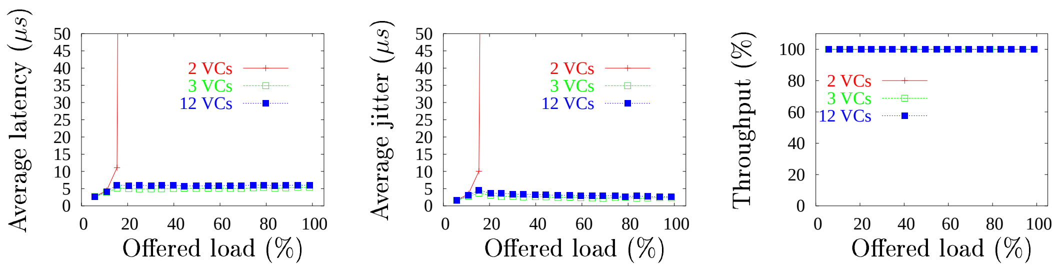
The best-effort performance is depicted in Figure 13 and, as one can imagine, is very poor. More important is that it is the same for the three architectures. We can conclude that the use of 12 VCs to provide QoS can be beneficial if there is no CAC, but even in this case, our proposal with only 2 VCs can provide the appropriate throughput with reduced buffer space and arbitration time. This is noticeable if we consider that all the QoS traffic uses only one oversubscribed VC. Our view is that it



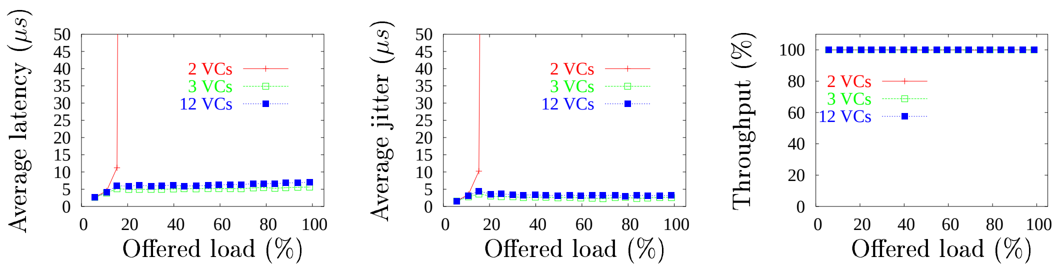
(a) Audio 0



(b) Audio 1



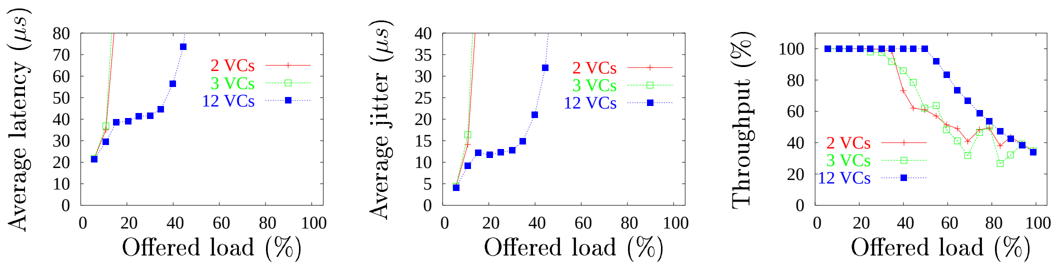
(c) Audio 2



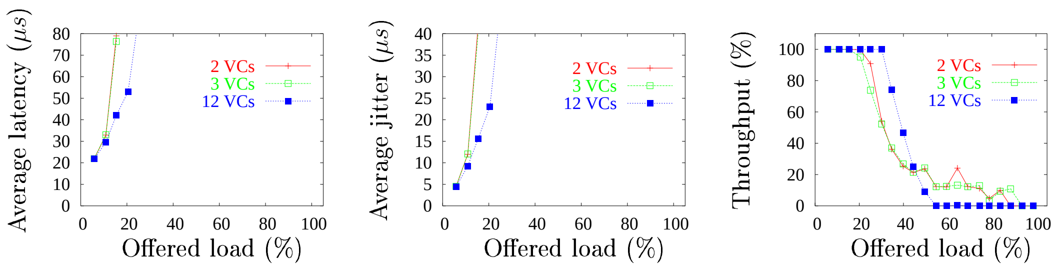
(d) Audio 3

Figure 11: Latency, jitter and throughput for audio traffic with hot-spots.

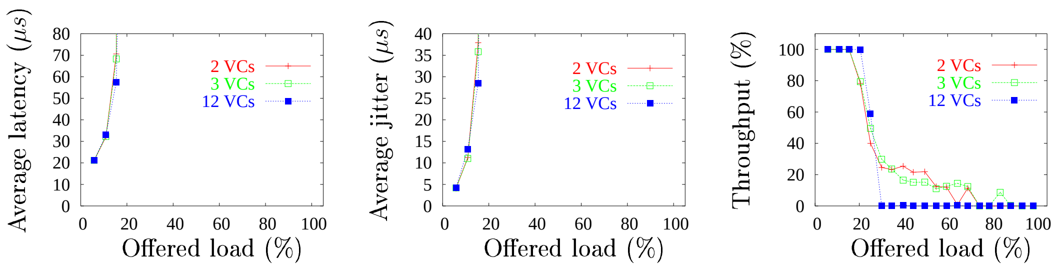
is much wiser to use a CAC and fewer VCs than to implement a larger number of VCs without CAC. The reason for this is that, as we have shown here, more VCs alone do not guarantee a good performance for the QoS traffic, which means that the CAC is necessary anyway.



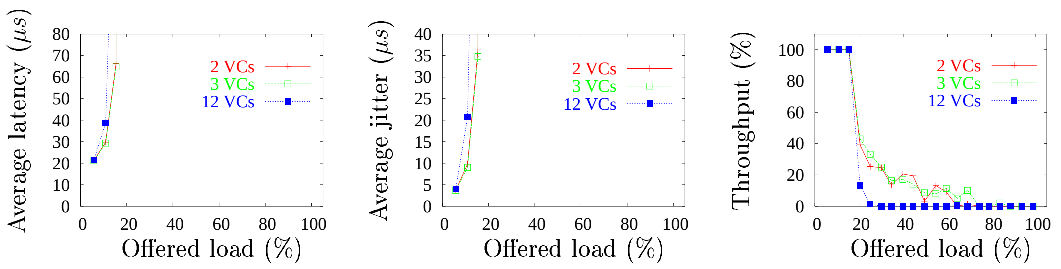
(a) Video 0



(b) Video 1

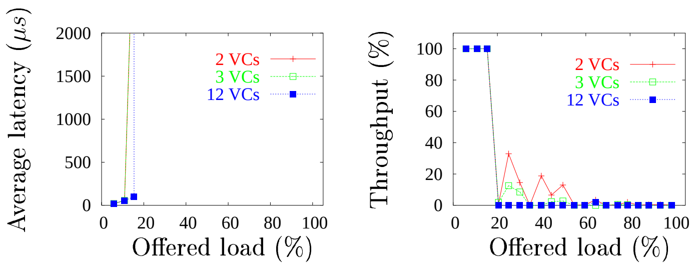


(c) Video 2

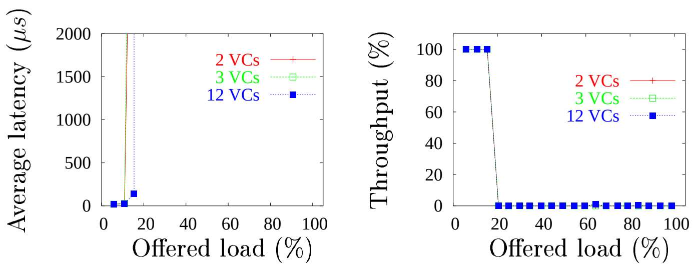


(d) Video 3

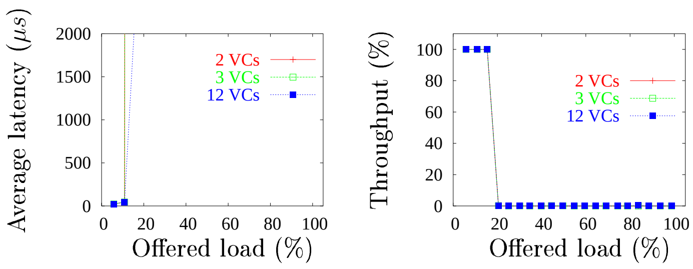
Figure 12: Latency, jitter and throughput for video traffic with hot-spots.



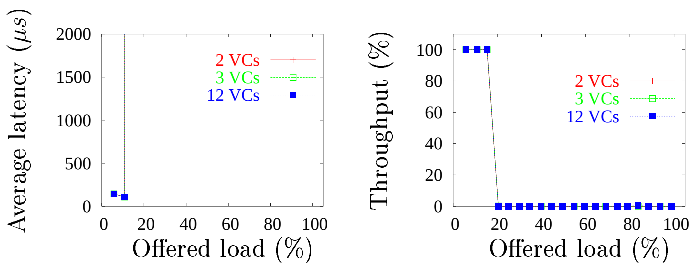
(a) BE 0



(b) BE 1



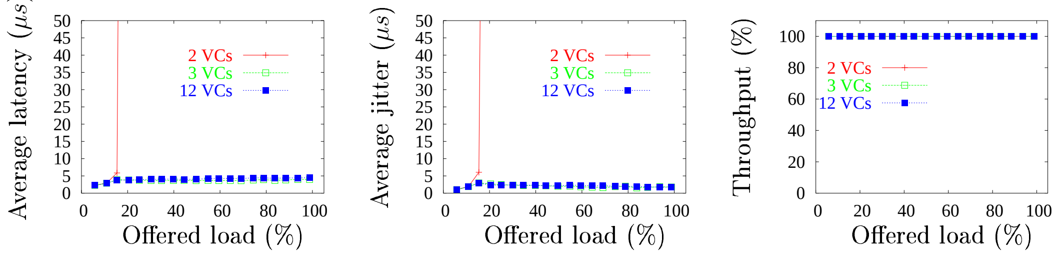
(c) BE 2



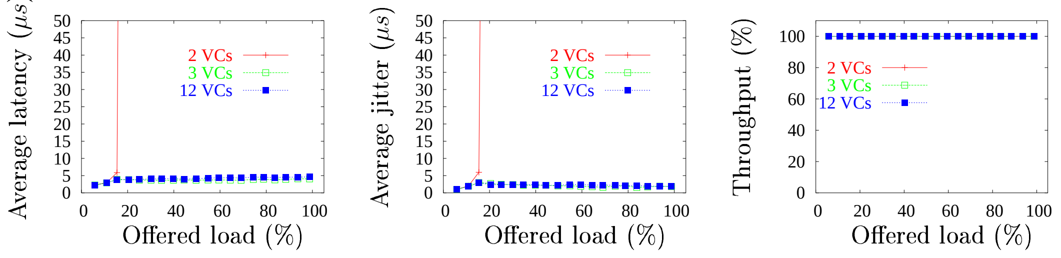
(d) BE 3

Figure 13: Latency and throughput for best-effort traffic with hot-spots.

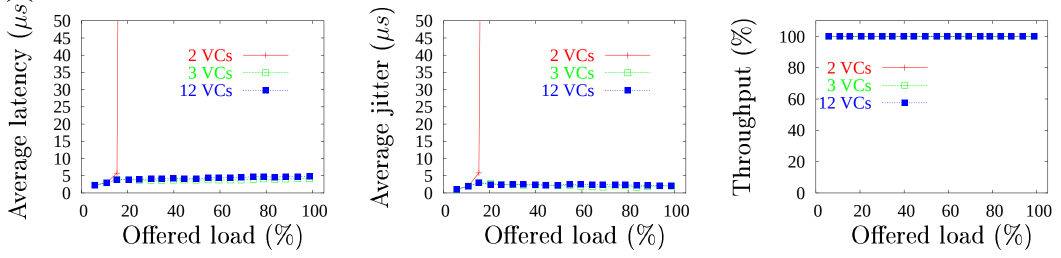
Free destinations In the next figures we provide a closer perspective of the hot-spots scenario. We show the results obtained by the traffic not targeted to the hot-spots. Even if a packet does not travel directly to a hot-spot node, it can also suffer from congestion. As is well known, when congestion trees appear, they can affect the entire network [19]. For that reason, the results shown in figures 14, 15 and 16 are very similar to the average results. The only noteworthy difference is that the average latency for the audio packets is smaller, because they have to stay in the congested area for a shorter time than those directly targeted to the hot-spot nodes.



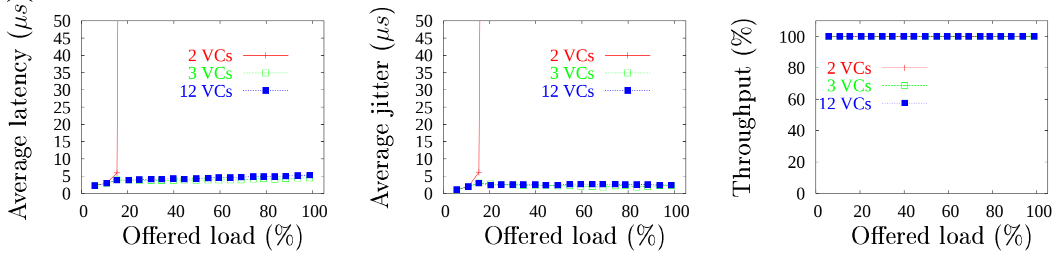
(a) Audio 0



(b) Audio 1

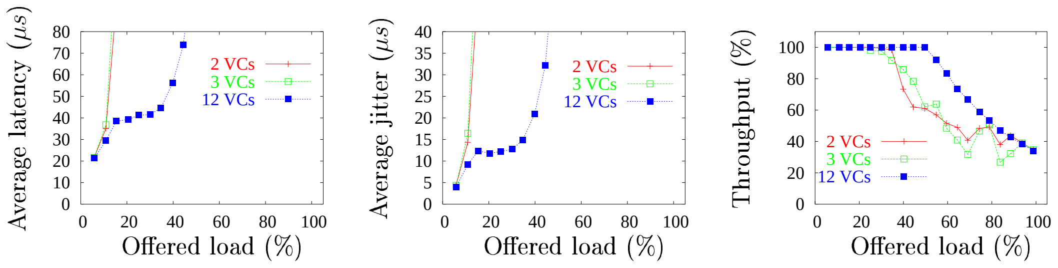


(c) Audio 2

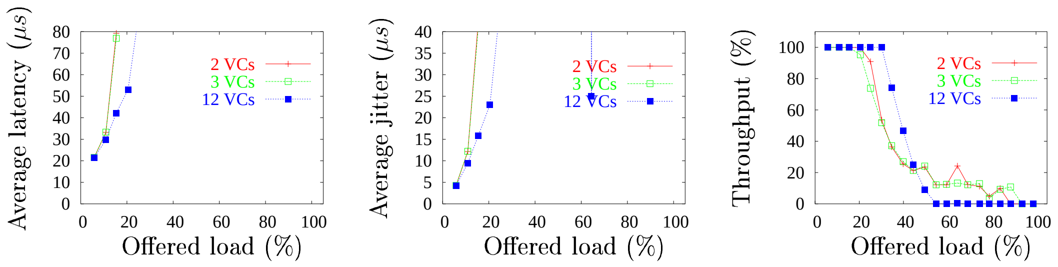


(d) Audio 3

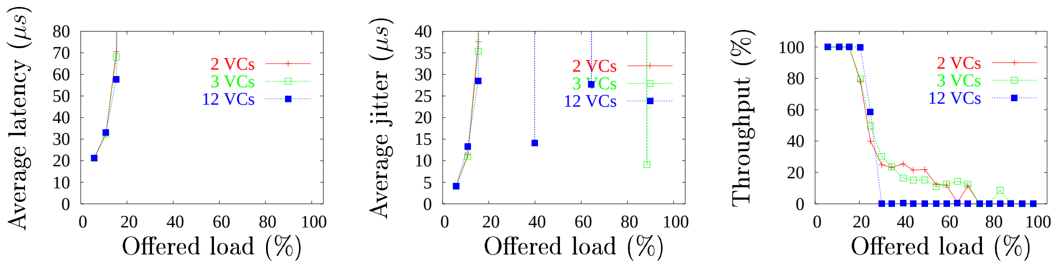
Figure 14: Latency, jitter and throughput for audio traffic with hot-spots.



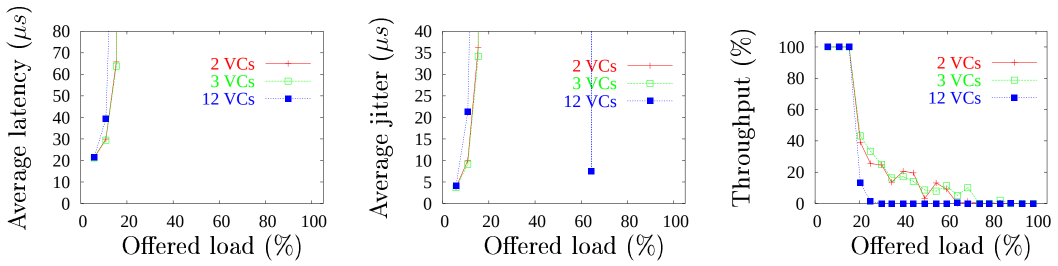
(a) Video 0



(b) Video 1

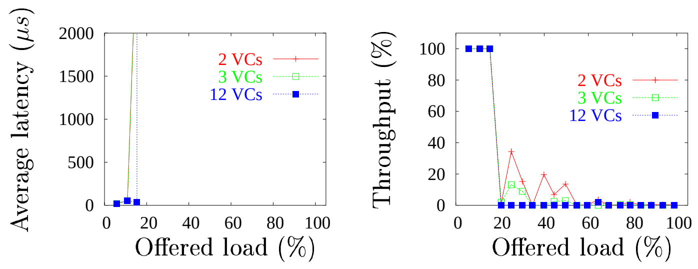


(c) Video 2

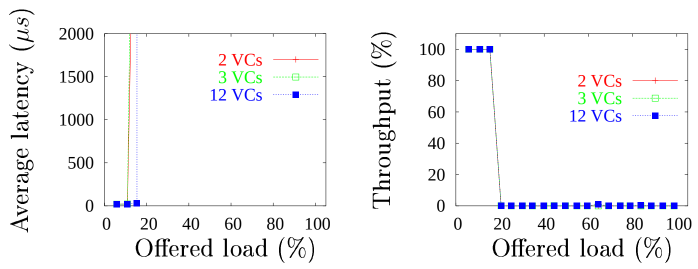


(d) Video 3

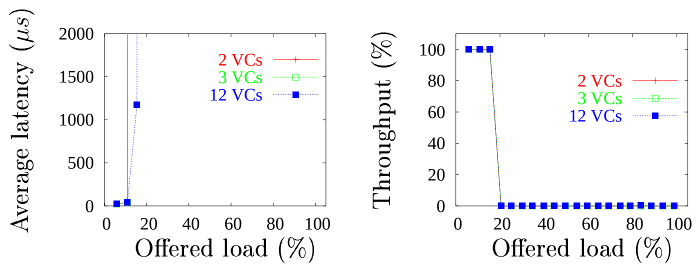
Figure 15: Latency, jitter and throughput for video traffic with hot-spots.



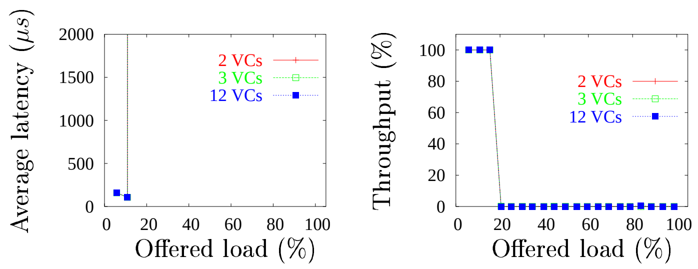
(a) BE 0



(b) BE 1



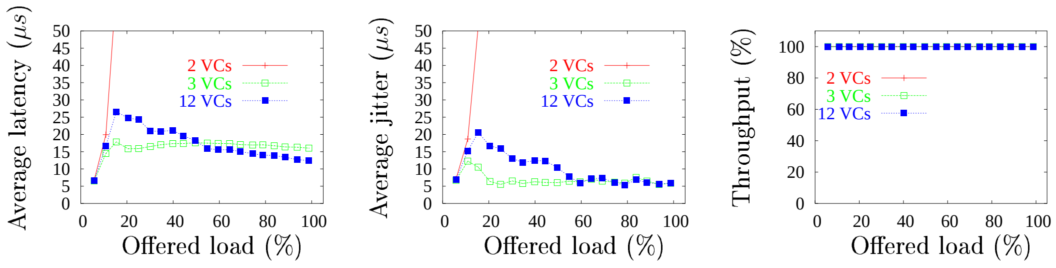
(c) BE 2



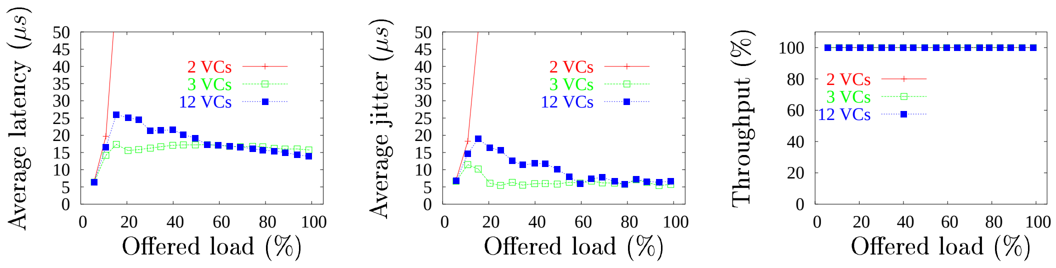
(d) BE 3

Figure 16: Latency and throughput for best-effort traffic with hot-spots.

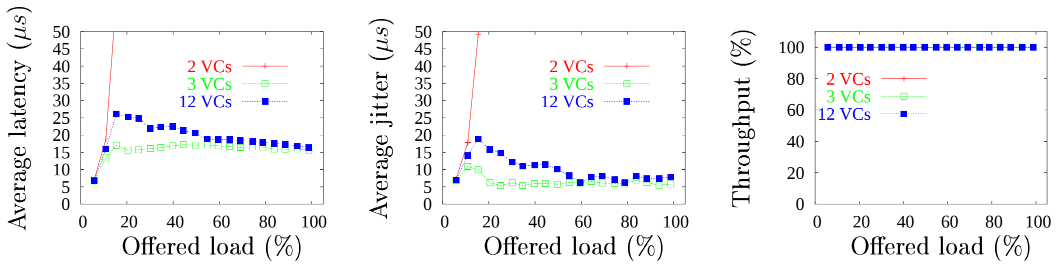
Congested destinations For completeness, we also show the performance of traffic in the hot-spots scenario, but now only with the packets traveling directly to the hot-spots. Figures 17, 18 and 19 show this. Again, the only noticeable difference is that the average latency for the audio packets is greater, because of the heavy congestion along the way to the hot-spot nodes.



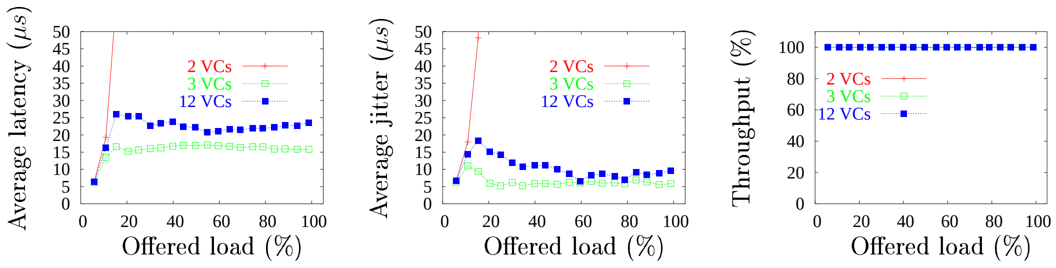
(a) Audio 0



(b) Audio 1

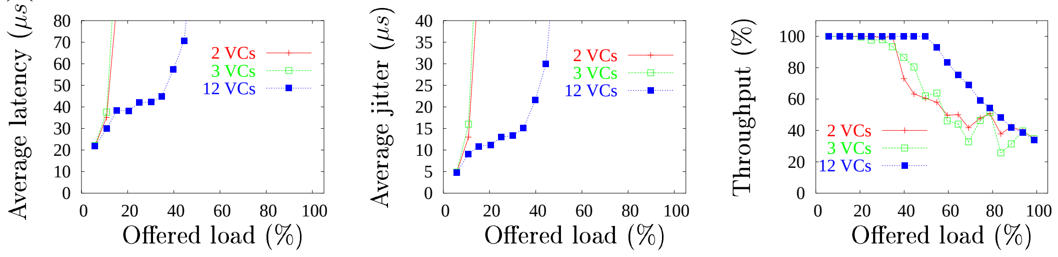


(c) Audio 2

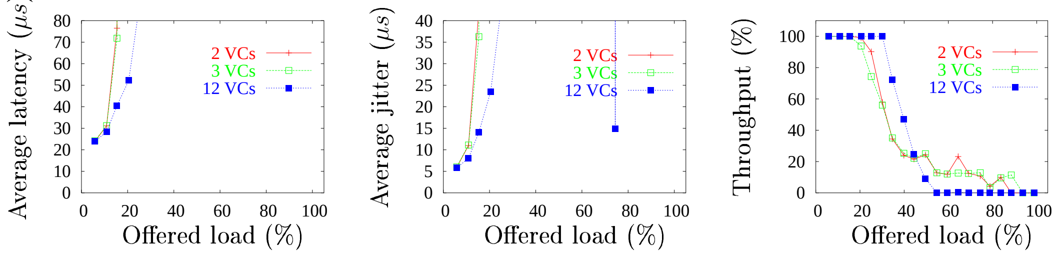


(d) Audio 3

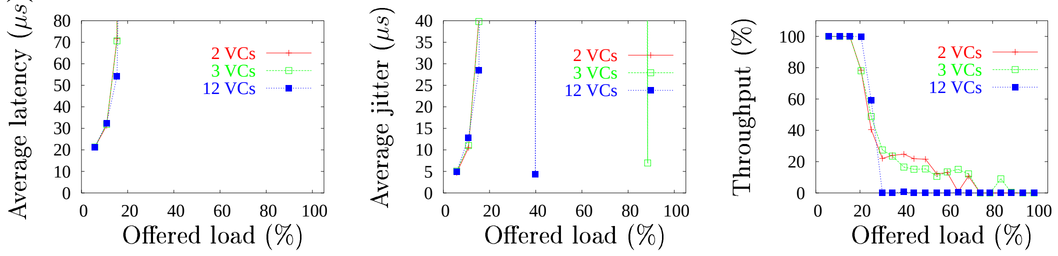
Figure 17: Latency, jitter and throughput for audio traffic with hot-spots.



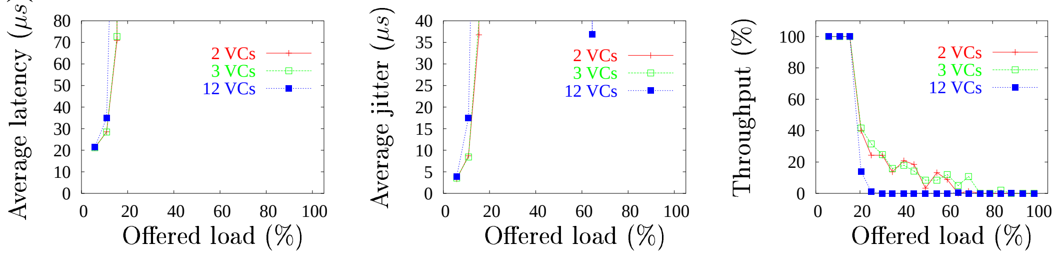
(a) Video 0



(b) Video 1

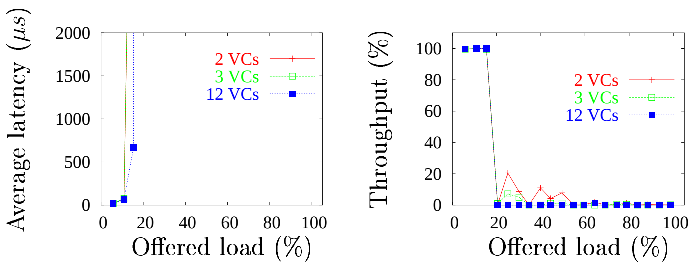


(c) Video 2

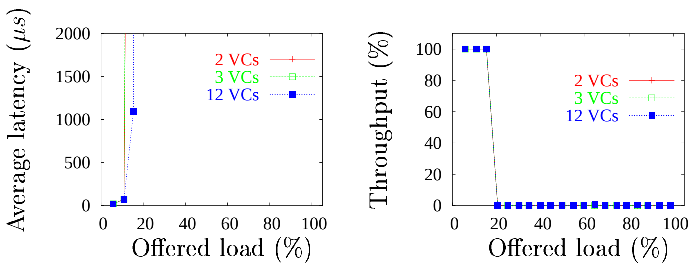


(d) Video 3

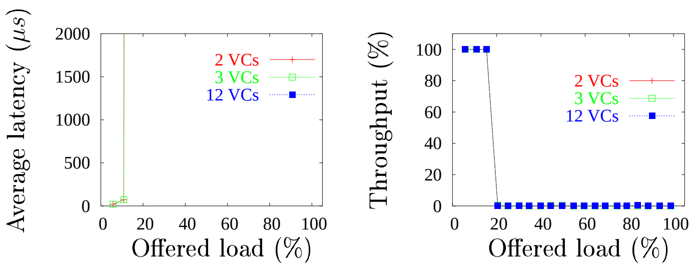
Figure 18: Latency, jitter and throughput for video traffic with hot-spots.



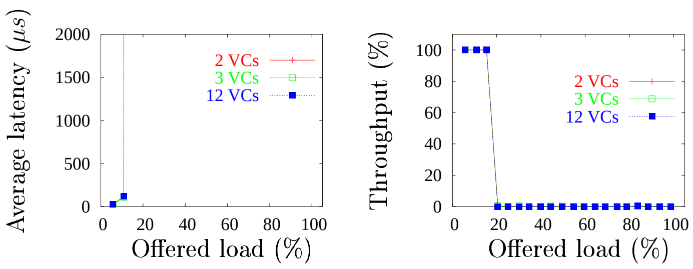
(a) BE 0



(b) BE 1



(c) BE 2



(d) BE 3

Figure 19: Latency and throughput for best-effort traffic with hot-spots.

5 Conclusions

The proposal of this paper consists in making the network elements cooperate, building together ordered flows of packets. Consequently, the switches try to respect the order in which packets arrive at the switch ports, which is probably correct. This allows a drastic reduction in the number of VCs required for QoS purposes at each switch port.

We have shown that WRR arbiters are only useful to provide throughput guarantees. Due to the relationship between both requirements, if we want to provide strict deadline guarantees and the deadline is short, we must reserve many rows of the table and thereby also provide a high bandwidth. This has been experimentally confirmed and the performance of our proposal was superior to that of the WRR arbiter, even with less resources.

The design we propose is a remarkable improvement over today's trends, where a meaningful number of VCs are required to provide QoS and table-based WRR arbiters are used. The network architecture presented here clearly reduces the number of VCs needed and thus the memory required to implement them. It uses simple scheduling policies which reduce the peak arbitration time, and it provides a performance comparable with that obtained using more complex solutions.

This study has shown that it is possible to achieve a more than acceptable QoS with only two VCs. This opens up the possibility of using the remaining VCs for other concerns, like adaptive routing or fault tolerance. Furthermore, it is also possible to reduce the number of VCs supported at the switches, thereby simplifying the design, or increasing the number of ports.

In this paper we have also tested our proposal in the worst situation, that is, when there are several hot-spot nodes and the CAC is omitted, leading to a situation where some links are oversubscribed. These tests have shown that even in these conditions our proposal can provide an acceptable performance. However, results have also shown that for an optimal operation a CAC that assures the bandwidth reservation is necessary.

The results we have shown here confirm that the CAC is a key element of our proposal to provide QoS with low resources. Moreover, we have also shown that the CAC is very important for any QoS support scheme, because even if we provide a large number of VCs, if it cannot be guaranteed that the links will have enough bandwidth, we cannot provide an adequate QoS. Hence the use of a CAC, as in our proposal, can be extremely advantageous. The confirmation of this necessity was mandatory before we could further progress in our study of reducing the costs of QoS support.

We are currently examining a number of possible extensions to the work here presented. First, we are exploring a more formal approach which can analytically prove the results we have obtained from simulation experiments. Secondly, we intend to code the switches and network interfaces with a hardware description language, which can then be implemented in FPGAs to examine the practicality of our models. Finally, we are considering more complex switch models that can benefit from our proposal, such as switches using EDF arbitration.

Bibliography

[1] Myrinet. Web page, Myricom, Inc. `http://www.myrinet.com`.

[2] QsNet overview. W. paper, Quadrics Ltd. `http://www.quadrics.com`.

[3] Infiniband switch systems. Product overview, Mellanox Technologies Inc., 2004. `http://www.mellanox.com/products/switch.html`.

[4] Advanced switching core architecture specification. Technical report, available at `http://www.asi-sig.org/specifications` for ASI SIG members.

[5] F. J. Alfaro, J. L. Sánchez, and J. Duato. A strategy to manage time sensitive traffic in InfiniBand. In *Proceedings of Workshop on Communication Architecture for Clusters (CAC'02)*, April 2002.

[6] F. J. Alfaro, J. L. Sánchez, and J. Duato. QoS in InfiniBand subnetworks. *IEEE Trans. Parallel Distrib. Syst.*, 15(9):810–823, September 2004.

[7] S. Berry. *Advanced Bus and Interface Markets and Trends.* Electronic Trend Publications, 2004.

[8] D. Bull, N. Conagarajah, and A. Nix. *Insights into Mobile Multimedia Communications.* Academic Press, 1999.

[9] D. Chalmers and M. Sloman. A survey of quality of service in mobile computing environments. *IEEE Communications Surveys and Tutorials*, 2(2), 1999.

[10] 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 *Proceedings of the 5th Symposium on High Performance Computer Architecture (HPCA-5)*, January 1999.

[11] H. Eberle and E. Oertli. Switcherland: a QoS communication architecture for workstation clusters. In *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pages 98–108, 1998.

[12] P. Ferguson and G. Huston. *Quality of service: delivering QoS on the Internet and in corporate networks.* John Wiley & Sons, Inc., 1998.

[13] J. Flich, P. López, J. C. Sancho, A. Robles, and J. Duato. Improving InfiniBand routing through multiple virtual networks. *Lecture Notes in Computer Science*, 2327:49–58, 2002.

[14] InfiniBand Trade Association. *InfiniBand architecture specification volume 1. Release 1.0*, October 2000.

[15] M. Katevenis, S. Sidiropoulos, and C. Corcoubetis. Weighted round-robin cell multiplexing in a general-purpose ATM switch. *IEEE J. Select. Areas Commun.*, October 1991.

[16] C. Minkenberg, F. Abel, M. Gusat, R. P. Luijten, and W. Denzel. Current issues in packet switch design. In *ACM SIGCOMM Computer Communication Review*, pages 119–124, January 2003.

[17] D. Miras. A survey on network QoS needs of advanced internet applications. Technical report, Internet2 - QoS Working Group, 2002.

[18] J. M. Montañana, J. Flich, A. Robles, P. López, and J. Duato. A transition-based fault-tolerant routing methodology for InfiniBand networks. In *Proceedings of Workshop on Communication Architecture for Clusters (CAC'04)*, 2004.

[19] G. Pfister and A. Norton. Hot spot contention and combining in multistage interconnection networks. *IEEE Transactions on Computers*, C-34, 10:943–948, 1985.

[20] X. Xiao and L.M. Ni. Internet QoS: A Big Picture. *IEEE Network Magazine*, pages 8–18, March 1999.

[21] K. H. Yum, E. J. Kim, C. R. Das, and A. S. Vaidya. MediaWorm: A QoS capable router architecture for clusters. *IEEE Trans. Parallel Distrib. Syst.*, 13(12):1261–1274, December 2002.

[22] K.H. Yum, E.J. Kim, and C.R. Das. QoS provisioning in clusters: An investigation of router and NIC design. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*. IEEE Computer Society, July 2001.

[23] L. Zhang. VirtualClock: A new traffic control algorithm for packet switched networks. *ACM Transaction on Computer Systems*, 9, 2:101–124, 1991.