Efficient Switches with QoS Support for Clusters*

Alejandro Martínez[1], Francisco J. Alfaro[1], José L. Sánchez[1], José Duato[2]

[1]DSI - Univ. of Castilla-La Mancha
02071 - Albacete, Spain
{alejandro, falfaro, jsanchez}@dsi.uclm.es

[2]DISCA - Tech. Univ. of Valencia
46071 - Valencia, Spain
jduato@disca.upv.es

Abstract

Current interconnect standards providing hardware support for quality of service (QoS) consider up to 16 virtual channels (VCs) for this purpose. However, most implementations do not offer so many VCs because they increase the complexity of the switch and the scheduling delays. We have shown that this number of VCs can be significantly reduced, because it is enough to use two VCs for QoS purposes at each switch port. In this paper, we cover the weaknesses of that proposal and, not only we reduce VCs, but we also improve performance due to the flexibility assigning buffer memory.

1 Introduction

The last decade has witnessed a vast increase in the amount of information and services available through the Internet. These services rely on applications executed in many servers all around the world. 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. These are known as quality of service (QoS) requirements.

Several cluster switches with QoS support have been proposed. All of them incorporate VCs in order to provide QoS support. Among the most recent ones are the industry standards InfiniBand and PCI Express Advanced Switching (AS). InfiniBand [7] can support up to 16 VCs. On the other hand, AS architecture [2] incorporates up to 20 VCs.

Therefore, recent proposals incorporate 16 or even more VCs, devoting a different VC to each traffic class. This increases the switch complexity and required silicon area. 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 [11].

On the other hand, there have been proposals which use only two VCs. For instance, the Avici TSR [4] is a well-known example of this. It is able to segregate premium traffic from regular traffic. However, it is limited to this classification and cannot differentiate among more categories. In the recent IEEE standards, it is recommended to consider seven traffic classes [6]. So, although being able to differentiate two categories is a big improvement, it could be insufficient.

In [10] we have proposed a strategy to use just two VCs at each switch port for the provision of QoS, emulating many more VCs. The idea consists in having a strict priority between traffic classes, such as the end-nodes would always inject packets with high priority before packets with low priority. This order of injection could be reused at the switches, producing an efficient design with good performance. However, inevitably, the end-nodes would inject some low-priority packets before packets with higher priority, leading to order-errors that would be propagated by the switches and would degrade performance.

The second weakness of that proposal is the requirement of strict priority among traffic classes, which is not the case in modern interconnects such as InfiniBand or PCI AS. Moreover, users usually require more sophisticated guarantees on latency that just the priority precedence.

In this paper, we review this proposal and fix these issues. In the next section, we present a switch design that offers complete QoS support using just two VCs. It does not require strict priority among traffic classes and allows to provide latency guarantees. Moreover, it is based in the table schedulers present both in the specifications of InfiniBand and PCI AS and, thus, can be implemented seamlessly

*This work was partly supported by the Spanish CICYT under CSD2006-46 and TIN2006-15516-C04-02 grants, by Junta de Comunidades de Castilla-La Mancha under grant PBC-05-005, and by the Spanish State Secretariat of Education and Universities under FPU grant.

in those network architectures.

The remainder of this paper is structured as follows. In Section 2 we explain our strategy to provide QoS support with only two VCs. Details on the experimental platform and the performance evaluation are presented in Section 3. Finally, Section 4 summarizes the results of this study.

2 Providing full QoS support with only 2 VCs

In modern interconnection technologies, like InfiniBand or PCI AS, the obvious strategy to provide QoS support consists in providing each traffic class with a separate VC. Only devoting a VC per traffic class at the switches is not enough to provide adequate QoS and other techniques and mechanisms are necessary.

More specifically, it is necessary to employ some kind of regulation on the traffic to provide strict guarantees on throughput and latency. Toward this end, a connection admission control (CAC) can guarantee that at no link the load will be higher than the available bandwidth.

Providing QoS with the scheduling at switches is not enough, there must be some scheduling at the output of the network interfaces as well. Thereby, these devices also need to implement VCs to separate the traffic classes.

Head-of-line (HOL) blocking and buffer hogging must be dealt with. There may be HOL blocking among the packets of a given VC if they are heading towards different destinations. This can be solved using virtual output queuing (VOQ) [4]. In this case, each input port has a queue per global destination of the network. However, this approach is generally an inefficient solution. A usual solution is to provide a separate queue per output of the switch. Although this could not solve completely the HOL blocking, it is an intermediate compromise between performance and cost. In that case, the number of queues required at each input port would be the number of traffic classes multiplied by the number of output ports of the switch.

We have observed that, once the previous conditions are met, traffic flows seamlessly through the network; congestion, if any, only happens temporarily. Therefore, regulated traffic flows with short latencies through the fabric. Therefore, to devote a different VC to each traffic class might be redundant.

On the other hand, when using *store and forward* or *virtual cut-through* switching, the minimum buffer space that is needed to achieve maximum throughput is one packet size plus a round-trip time (RTT) of data. However, depending on the characteristics of traffic, like burstiness or locality (hot-spots), more memory at the buffers is necessary to obtain acceptable performance.

VCs produce a static partition of buffer memory. That means that traffic of one VC cannot use space devoted to another VC, even if it is available. For that reason, although VCs provide traffic isolation, they may degrade overall performance under bursty traffic. We will see this at the performance evaluation section.

Based on all the previous observations, we propose that all the regulated (QoS) traffic that arrives at a switch port uses the same VC. We need a second VC for unregulated traffic, which should also be supported.

At the end-points, there are schedulers that take into account the QoS requirements of traffic classes. More specifically, in InfiniBand and PCI AS it is proposed to use a table based weighted round robin algorithm [8]. In [1] we proposed a sophisticated strategy to fill the table in such a way that we can guarantee both throughput and maximum latency. Briefly, we tune the table with two parameters: number of table entries per traffic class and maximum separation between the entries of that traffic class.

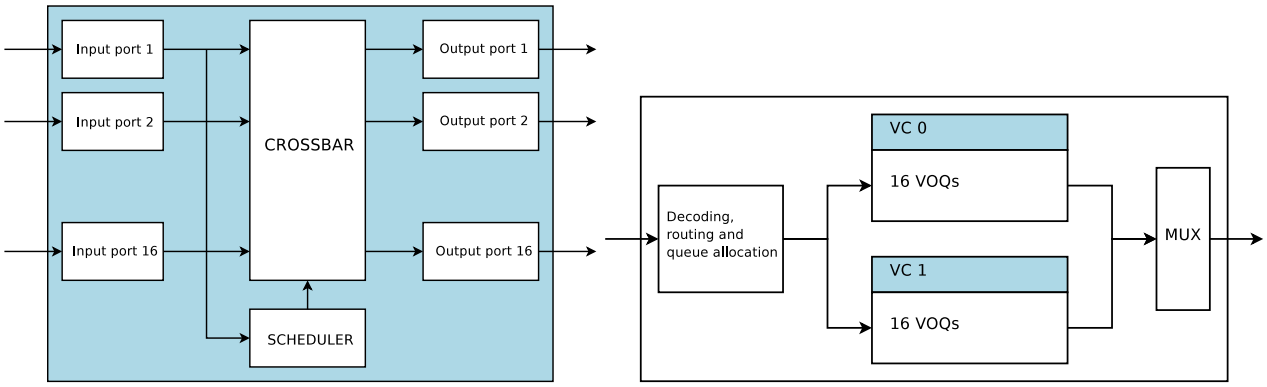
When the end-nodes implement this table-based schedulers, packets leaving the interfaces are ordered by the interface's scheduler. Therefore, if packet i leaves earlier than packet $i+1$, it is because it was the best decision to guarantee the QoS requirements of both packets, even if packet $i+1$ was not at the interface when packet i left. Therefore, we can assume that the order in which packets leave the interfaces is correct and there is no need for take-overs between packets coming from the same end-node.

For the purposes of the switches, it is safe to assume that in all the cases packet i has more priority than packet $i+1$. In this case, the switch is receiving at its input ports ordered flows of packets. Now, the switch's task is analogous to the sorting algorithm: it inspects the first packet at each flow and chooses the one whith the shorter arrival time, building another ordered flow of packets. Thereby, by using this scheduler, the switches achieve some reutilization of the scheduling decisions made at end-nodes. Note that we are talking about regulated traffic.

A drawback of our technique is that the switches are not able to reschedule traffic as freely as they would be with a technique where a different VC for each traffic class were implemented. This problem is attenuated by the connection admission, because connections are only allowed if we can guarantee their bandwidth and latency requirements all along the path of packets. That means that the connections are established as if all the VCs were implemented at the switches and there were also enough resources to attend the flows' requirements. In this way, we ensure that the required QoS load is feasible. We will not obtain exactly the same performance, but it will be very similar.

Another potential problem is that if an end-node is malfunctioning and injecting more than allowed by the CAC, it could disrupt the entire cluster. This could be solved by an adequate traffic policing and failure detection mechanism.

On the other hand, the best-effort traffic classes only receive coarse-grain QoS, since they are not regulated. How-



(a) Switch organization

(b) Switch input port organization

Figure 1. Switch architecture.

ever, the end-nodes are still able to assign the available bandwidth to the highest priority best-effort traffic classes and, therefore, some differentiation is achieved among them.

Note that this proposal does not aim at obtaining a better performance but, instead, at drastically reducing buffer requirements while achieving similar performance and behavior of systems with many more VCs. In this way, a complete QoS support can be implemented at an affordable cost.

2.1 Switch architecture

In this section, we describe the proposed switch architecture. We propose a single-chip, virtual cut-through switch intended for clusters/SANs. We assume QoS support for distinguishing two traffic categories: QoS-requiring and best-effort traffic. Credit-based flow control is used to avoid buffer overflow at the neighbor switches and end-nodes. For the rest of the design constraints, like packet size, routing, etc., we take PCI AS [2] as a reference model.

The block diagram in Figure 1 (a) shows the switch organization. We use a *combined input/output queued* (CIOQ) switch architecture because it offers line rate scalability and good performance [3]. In the CIOQ architecture, output conflicts (several packets requesting the same output) are resolved by buffering the packets at the switch input ports. Packets are transferred to the switch outputs through a crossbar whose configuration is synchronously updated by a central scheduler. To cope with the inefficiencies of the scheduler and packet segmentation overheads[1], the crossbar core operates faster than the external lines (internal speedup). Thus, output buffers are needed, resulting in the CIOQ architecture.

The organization that we propose for a switch input port can be seen in Figure 1 (b). There are only two VCs at the switch input ports: VC 0 is intended for QoS traffic, while VC 1 is intended for best-effort traffic. Each VC is further divided into several queues, which correspond to each switch output port (usually 8 or 16). These are logical queues which share the same physical memory and implement virtual output queuing (VOQ) at the switch level. The output ports of the switch are simpler: there are only three queues, one per VC plus one for the outgoing credits. These three queues, although sharing the same memory, are implemented in a static partition.

The switch is scheduled as follows. There is a strict precedence of VC 0 (QoS traffic) over VC 1 (best-effort traffic). Among packets belonging to the same VC, a simple FIFO scheduling is applied. Note that this cannot introduce starvation on the regulated traffic because the CAC assures that there is enough bandwidth for all the connections.

CAC may be a time-consuming task, not suitable for all traffic connections. However, for long communications (which take a significant amount of time to complete) it is feasible. Moreover, the CAC is contemplated and specified for popular cluster technologies, like InfiniBand and PCI AS.

Since we want to provide virtual cut-through switching, our scheduling decisions are made for whole packets (*packet-mode* scheduling [9]). In this way, once a packet is selected by the scheduler, the crossbar connection is kept until all cells of the packet have been delivered to the output. This allows the output port to start transmitting the packet on the line as soon as the first cell of the packet arrives at the switch output.

3 Performance evaluation

In this section, we show the behavior of our proposal. We have considered four switch architectures, varying two

[1] Crossbars inherently operate on fixed size cells and thus external packets are traditionally converted to such internal cells.

Table 1. Traffic injected per host.

TC	Name	% BW	Packet size	Regulated?	Notes
7	Network Control	1	[64,512] bytes	No	self-similar
6	Audio	16.333	128 bytes	Yes	CBR 64 KB/s connections
5	Video	16.333	[64,2048] bytes	Yes	750 KB/s MPEG-4 traces
4	Controlled Load	16.333	[64,2048] bytes	Yes	CBR 1 MB/s connections
3	Excellent-effort	12.5	[64,2048] bytes	No	self-similar
2	Preferential Best-effort	12.5	[64,2048] bytes	No	self-similar
1	Best-effort	12.5	[64,2048] bytes	No	self-similar
0	Background	12.5	[64,2048] bytes	No	self-similar

parameters: VCs and memory per port. Firstly, we have evaluated a traditional switch design with 8 VCs. On the other hand, we have evaluated our proposal of only 2 VCs. Regarding buffers, we have tested two types of switches: 32 Kbytes/port and 16 ports/switch, and 64 Kbytes/port and 8 ports/switch. Therefore, we have four switch architectures to compare: *8VC 16P*, *8VC 8P*, *2VC 16P*, and *2VC 8P*. The four cases would translate in single-chip switches of similar silicon area and, thus, implementation cost. However, 8VC cases are more complex to implement and would take higher delays, but this is not taken into account in our performance evaluation for the sake of clarity. Note that buffer space per VC varies at each case. Besides, when using 16 ports switches, less switches and links are necessary to connect the same number of end-points.

The network used to test our proposal is a butterfly multistage with 64 end-points. We have also tested other topologies, including direct networks, with similar results. The switches use a combined input and output buffer architecture, with a crossbar to connect the buffers. The CAC we have implemented is a simple one, based on average bandwidth. Each connection is assigned a path where enough resources are assured. No packets are dropped because we use credit-based flow control.

The scheduling at the 8 VC switches and at the end-nodes is weighted round robin based on a scheduling table. We provide enough entries and maximum entry separation to satisfy bandwidth and maximum requirements of regulated traffic classes. Moreover, we guarantee some bandwidth to best-effort traffic.

In Table 1, the characteristics of the traffic injected in the network are included. We have considered the traffic classes (TCs) defined by IEEE standard 802.1D-2004 [6] at the Annex G, which are particularly appropriate for this study. However, we have added an eighth TC, preferential best-effort, with a priority between excellent-effort and regular best-effort. It is possible that this mix of traffic is not actually present in a real-life cluster, but it serves perfectly to show the differences of the architectures we are testing. This evaluation follows the recommendations of the *Network Processing Forum Switch Fabric Benchmark Specifications* [5].

3.1 Simulation results

In this section, the performance of our proposals is shown. We have considered three traditional QoS indices for this performance evaluation: Throughput, latency, and jitter. Note that packet loss is not considered because no packets are dropped due to the use of credit-based flow control. We also show the cumulative distribution function (CDF) of latency and jitter, which represents the probability of a packet achieving a latency or jitter equal to or lower than a certain value. These results are obtained with an input load of 100% the capacity of links.

Firstly, we evaluate an initial scenario where the input QoS load is equal to the best-effort load. Afterwards, we will study which amount of QoS traffic can be allowed at each architecture before its performance is unacceptable. We consider QoS performance to be unacceptable when the bandwidth and maximum latency guarantees are not achieved.

We first study the results of QoS traffic. In Figure 2, we show the performance of *Network Control*, *Audio*, and *Video* traffic. The *Network Control* traffic demands very little bandwidth but a latency as short as possible. In this case, average results are very similar in the four cases. Maximum latency (which can be seen at the CDF plot) is 50 μs for 8 VC architectures and 200 μs for 2 VC architectures. Both results are acceptable, but, obviously, the first is better. Note that the guaranteed maximum latency by the table scheduler is approximately 400 μs according to [1], which is achieved in all cases.

Regarding *Audio*, and *Video* traffic, results are very similar in terms both of average and maximum latency. Moreover, jitter results (not shown) are also very similar.

Regarding *Controlled Load* and the best-effort traffic classes, results are almost the same in the four cases. However, we can see at Figure 3 (left) that maximum throughput is achieved with the *2VC 8P* architecture while worst results

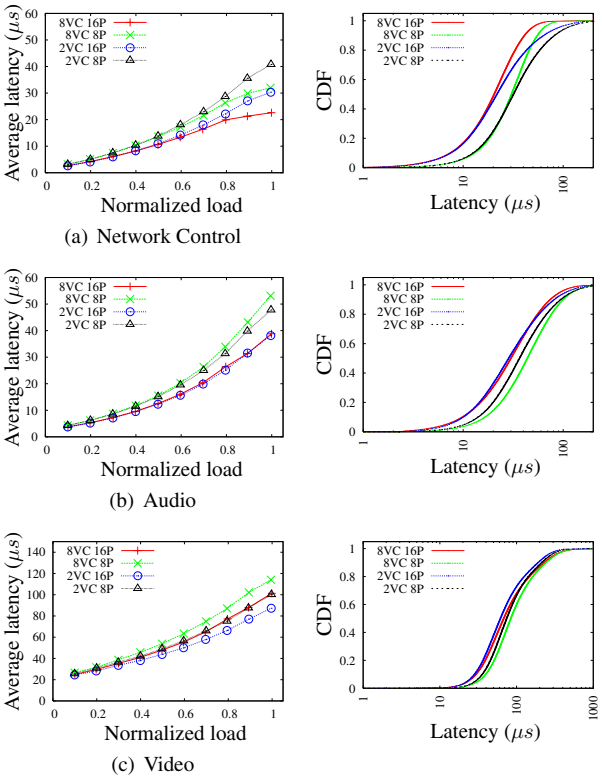


Figure 2. Latency of QoS traffic.

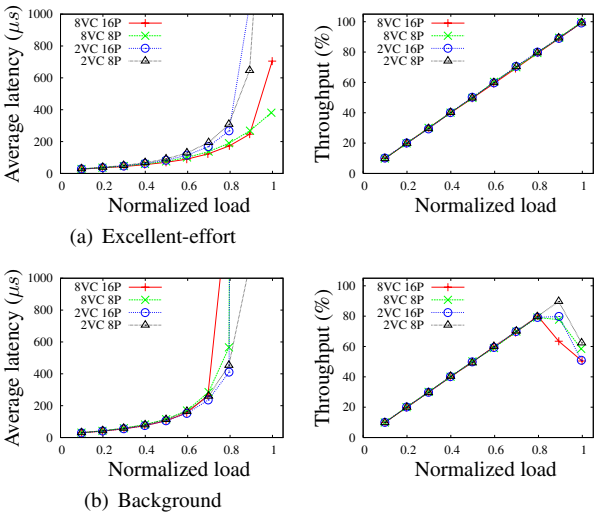


Figure 3. Latency and throughput of two best-effort traffic classes.

correspond to the *8VC 16P* case.

We can conclude at this point that the four architectures offer similar performance: 8 VC architectures have some advantage on the latency of *Network Control* traffic due to their better traffic isolation, whereas 2 VC architectures reduce the number of VCs required and increase global throughput due to the improved buffer management (shared space is larger).

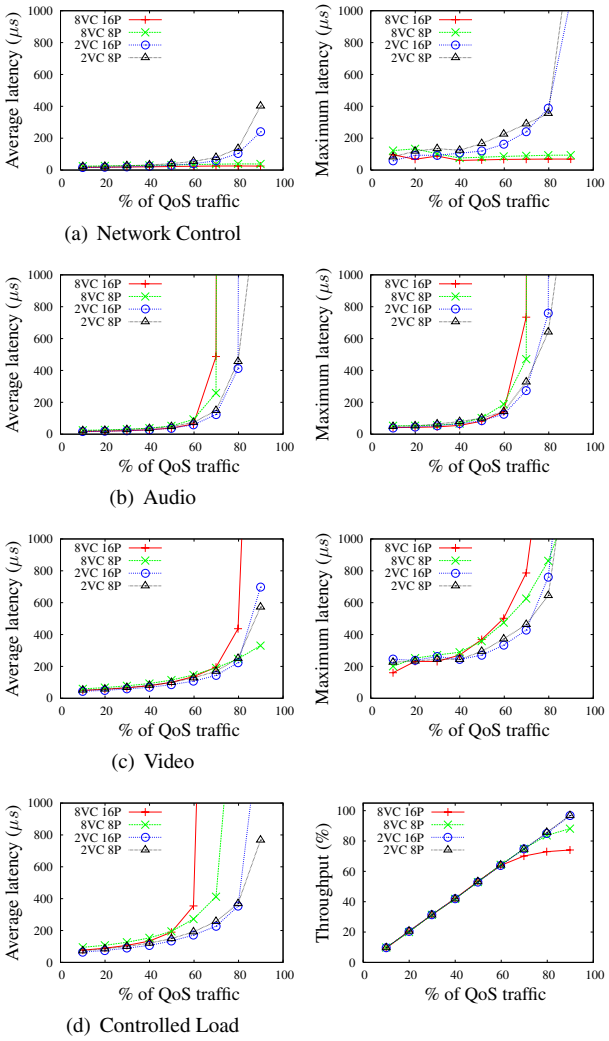


Figure 4. Performance of QoS traffic, varying QoS load.

In Figure 4 we vary the proportion of QoS traffic, from 10% to 90% of the total available network load. We fill the remaining bandwidth with best effort-traffic. The proportions among both groups are mantained (i.e. same amount of *Audio* and *Video*). We can see that the different traffic classes saturate at different points when using the four architectures.

We can see at Table 2 which is the maximum QoS load at which the different architectures yield acceptable results. We consider QoS performance to be unacceptable when

Table 2. Maximum QoS load with acceptable performance.

Traffic Class	8VC 16P	8VC 8P	2VC 16P	2VC 8P
Network Control	90%	90%	80%	80%
Audio	70%	70%	80%	80%
Video	70%	80%	80%	80%
Controlled Load	60%	70%	80%	90%
All QoS	**60%**	**70%**	**80%**	**80%**

the bandwidth and maximum latency guarantees are not achieved. For instance, performance of *Audio* traffic with the architecture *8VC 16P* is only acceptable up to a QoS load of 70%.

The last row of the table contains the minimum of the column, which means the maximum load where all the QoS requirements can be satisfied. We can see that both 2 VC architectures can accept up to 80% of QoS traffic, whereas the *8VC 16P* and *8VC 8P* cases can only accept 70% and 60% respectively.

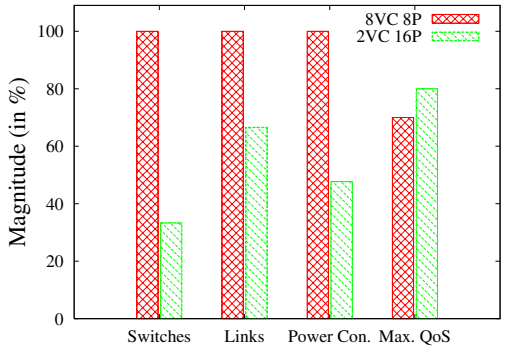


Figure 5. Comparison between *8VC 8P* **and** *2VC 16P*.

This is not the only advantage of our proposal. The *2VC 16P* case, since has 16 ports per switch, offers many advantages when compared with the *8VC 8P* architecture. This is summarized at Figure 5.

According to these results, we can conclude that our proposal can provide an adequate QoS performance. Using our switches, we greatly decrease the cost and power-consumption of the interconnection with excellent results.

4 Conclusions

In [10] we presented a proposal to use only two VCs at each switch port to provide QoS support. The first VC is used for QoS traffic and the other for best-effort traffic. In this way, we obtained a drastic reduction in the number of VCs required for QoS purposes at each switch port. In this paper, we improve that proposal in order to guarantee throughput and maximum latency. Moreover, we do so with the resources available at modern interconnection specifications like PCI AS and InfiniBand.

Therefore, we propose in this paper an efficient switch architecture for clusters with QoS support. It offers performance comparable to that of more complex switches but at the same time, reduces sillicon area requirements and is able to cope with more QoS requiring traffic.

References

[1] F. J. Alfaro, J. L. Sánchez, and J. Duato. QoS in InfiniBand subnetworks. *IEEE Transactions on Parallel Distributed Systems*, 15(9):810–823, Sept. 2004.

[2] ASI SIG. *Advanced switching core architecture specification*, 2005.

[3] S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching output queueing with a combined input output queued switch. In *INFOCOM (3)*, pages 1169–1178, 1999.

[4] W. Dally, P. Carvey, and L. Dennison. Architecture of the Avici terabit switch/router. In *Proceedings of the 6th Symposium on Hot Interconnects*, pages 41–50, 1998.

[5] I. Elhanany, D. Chiou, V. Tabatabaee, R. Noro, and A. Poursepanj. The network processing forum switch fabric benchmark specifications: An overview. *IEEE Network*, pages 5–9, Mar. 2005.

[6] IEEE. 802.1D-2004: Standard for local and metropolitan area networks. `http://grouper.ieee.org/groups/802/1/`, 2004.

[7] InfiniBand Trade Association. *InfiniBand architecture specification volume 1. Release 1.0*, Oct. 2000.

[8] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis. Weighted round-robin cell multiplexing in a general-purpose ATM switch. *IEEE J. Select. Areas Commun.*, pages 1265–1279, Oct. 1991.

[9] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri. Packet-mode scheduling in input-queued cell-based switches. *IEEE/ACM Trans. Netw.*, 10(5):666–678, 2002.

[10] A. Martínez, F. J. Alfaro, J. L. Sánchez, and J. Duato. Providing full QoS support in clusters using only two VCs at the switches. In *Proceedings of the 12th International Conference on High Performance Computing (HiPC)*, pages 158–169, Dec. 2005. Available at `http://investigacion.uclm.es/portali/documentos/it_1131561750-HiPC05.pdf`.

[11] C. Minkenberg, F. Abel, M. Gusat, R. P. Luijten, and W. Denzel. Current issues in packet switch design. In *ACM SIGCOMM Computer Communication Review*, volume 33, pages 119–124, Jan. 2003.