Towards Efficient Switches with QoS Support*

Alejandro Martínez, Francisco J. Alfaro, José L. Sánchez
DSI - Univ. of Castilla-La Mancha
02071 - Albacete, Spain
{alejandro, falfaro, jsanchez}@dsi.uclm.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 explore a switch design that takes advantage of this reduction.

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 [10]. These are known as quality of service (QoS) requirements.

In the last few years, 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 [5] can support up to 16 VCs. On the other hand, AS architecture [1] incorporates up to 20 VCs. If a great number of VCs is implemented, it would require a significant fraction of silicon area and would make packet processing a more time-consuming task. 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 [9].

On the other hand, there have been proposals which use only two VCs. For instance, the Avici TSR [2] 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 [4]. So, although being able to differentiate two categories is a big improvement, it could be insufficient.

In [7], we have proposed a strategy to use just two VCs at each switch port for the provision of QoS that obtains very similar results as if we were using many more VCs. In this paper, we explore a switch design that takes advantage of this VC reduction, we study the hardware constraints of this design, and we evaluate it with realistic traffic models. Simulation results show a very similar performance compared with a traditional architecture with many more VCs. Moreover, the network built with our switches reduces greatly the component count.

2 Providing full QoS support with only two VCs

In [7], we have proposed a new strategy to use only two VCs at each switch port to provide QoS which achieves similar performance results to those using many more VCs. Here we review this proposal.

The key idea of our proposal is based on this observation: Assuming that the links are not oversubscribed, all the traffic flows through the switches seamlessly. Therefore, the basic idea of our proposal consists in using only two VCs at the switch ports. One of these VCs is used for QoS packets and the other for best-effort packets. Moreover, we propose to use a connection admission control (CAC) to guarantee that QoS

*This work was partly supported by the Spanish CICYT under grant TIC2003-08154-C06 and by the Spanish State Secretariat of Education and Universities under FPU grant.

traffic will not oversubscribe the links and we give QoS traffic absolute priority over best-effort traffic, which is not subject to the CAC.

The scheduler we propose considers the original priority of the packets at the head of the queues, instead of just whether they are regulated traffic or not. Note that this cannot lead to starvation on the regulated traffic because the CAC assures that there is enough bandwidth for all the regulated flows. Thereby, by using this scheduler, the switches achieve some reutilization of the scheduling decisions taken at network interfaces. This is because the order of the incoming messages is respected, but at the same time, the switches merge the flows to produce a correct order at the output ports.

A drawback of our technique is the switches not being able to reschedule the traffic as freely as if a different VC for each traffic class were implemented. In that case, the switches have the ability to push forward high-priority packets in detriment of packets with less priority at the same input port. Using our technique, it is only possible to do so with packets from different input ports.

Let us analyze the consequences of this handicap: It may happen that when no more high-priority packets are available, a low-priority packet is transmitted. We will assume that both packets are regulated traffic and, thus, will share the same VC. If the low-priority packet has to wait in 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 queue. Thus, the arbiter would penalize the high-priority packets, because they would have to wait until the low-priority packet is transmitted. This situation has a small impact on performance because there is bandwidth reservation for these packets. This means that all the QoS packets will flow with short delay.

On the other hand, the best-effort traffic classes only receive coarse-grain QoS, since they are not regulated. However, the interfaces are still able to assign the available bandwidth to the highest priority best-effort traffic classes and, therefore, some differentiation is achieved among them. If stricter guarantees were needed by a particular flow, it should be classified as QoS traffic. Therefore, although best-effort traffic can obtain a better performance using more VCs, the results do not justify the higher expenses.

Summing up, our proposal consists in reducing the number of VCs at each switch port needed to provide flows with QoS. Instead of having a VC per traffic class, we propose to use only two VCs at switches: One for QoS packets and another for best-effort packets. In order for this strategy to work, we guarantee that there is no link oversubscription for QoS traffic by using a CAC strategy.

3 Switch architecture

In this section we describe the proposed switch architecture. We study a 16 port, 8 Gb/s line rate, 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 network interfaces. For the rest of the design constraints, like packet size, routing, etc., we take PCI AS [1] as a reference model.

We will use a *combined input/output queued* (CIOQ) switch organization because it offers line rate scalability and good performance. Moreover, it can be efficiently implemented in a single chip. This is necessary in order to offer the low cut-through latencies demanded by current applications. Moreover, this also allows to provide some internal speed-up, without the need of faster external links.

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 speed-up). Thus, output buffers are needed, resulting in the CIOQ architecture. In this architecture, the memory access rate needed (including input and output accesses) is $(S+1) \times L$, where L is the external line rate and S is the speed-up factor (1 means no speed-up).

The organization that we propose for an input port consists in only two VCs: VC 0 is intended for QoS traffic, while VC 1 is intended for best-effort traffic. Each VC is further divided into 16 queues, which correspond to each switch output port. 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 queues, although sharing the same memory, are implemented in a static partition of the memory.

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

The switch is scheduled as follows. There is a strict precedence of VC 0 (QoS traffic) over VC 1 (best-effort traffic). Among the queues inside each VC, a simple round-robin algorithm is applied. The scheduling algorithm is very similar to iSLIP [8]. However, iSLIP was proposed as a *cell-mode* scheduler: External packets are splitted in fixed size internal cells which are scheduled ignoring which cell belongs to which packet. Packet reassembly is required at the switch output and cut-through cannot be used. Since we want to provide virtual cut-through switching, our scheduling decisions are made for whole packets (*packet-mode* scheduling [6]). 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.

4 Design evaluation

In the following, we study the silicon area, the power consumption, and the expected cut-through latency of the switch architecture proposed in the previous section. We consider 0.18 μm and 0.13 μm technologies, because they are popular in interconnection components and plenty of information is available about our interest topics.

In order to find out the area requirements of this design, we consider the individual components of the switch core. These are the buffers, the crossbar and the scheduler. Table 1 shows area estimates for each module.

Table 1. Area consumption.

Module	Tech. 0.18 μm	Tech. 0.13 μm
Buffers	64 mm^2	32 mm^2
Xbar and datapath	10 mm^2	5 mm^2
Scheduler	5 mm^2	3 mm^2
Total	79 mm^2	40 mm^2

We follow a similar methodology in order to figure out the power consumed by this design: we will analyze the power consumption of each individual component. Note that power consumption heavily depends on the activity of the different components and, therefore, on the load of the system. In Table 2, we see the estimates for the different elements considering worst-case power consumption.

Finally, we calculate the expected cut-through delay of our switch. We assume a pipelined design of the switch, as is usually the case in high performance switches. The stages are:

Table 2. Peak power consumption.

Module	Tech. 0.18 μm	Tech. 0.13 μm
Transceivers	9.0 Watt	6.4 Watt
Buffers	4.0 Watt	2.6 Watt
Xbar and datapath	3.0 Watt	1.8 Watt
Scheduler	0.9 Watt	0.5 Watt
Total	16.9 Watt	11.3 Watt

- Header decode/Routing/VC allocation.
- Block allocation.
- Writing and scheduling.
- Crossbar traversal.
- Output scheduling.

The latency of the first operation would be 1 cycle, which translates to 4 ns at 250 MHz clock frecuency. The latency of the three stages operating at 16 Gb/s over 64 bytes blocks (32 ns) is 3×32 ns $= 96$ ns. Finally, the output scheduling could also be performed in a single cycle. Therefore, the total latency would be $4+96+4 = 104$ ns; a complete implementation process would be necessary for more accurate delays.

After this hardware characteristics study, we procede to examine, through simulation, the performance of this switch architecture in the following section.

5 Performance evaluation

In this section, we show the behavior of our proposal and we compare it with the performance of traditional switches.

We have performed the tests considering three different cases. First, we have tested the performance of our proposal, which uses 2 VCs at each switch port. The second case would be a switch using 8 VCs (as many VCs as traffic classes). Finally, we have also tested a traditional approach with 2 VCs at switch ports and network interfaces, noted in the figures as *Traditional 2 VCs*. Therefore, we have two references to compare the performance of our proposal, one being the lower bound (*Traditional 2 VCs*) and the other the upper bound (*Traditional 8 VCs*).

Table 3 presents the characteristics of the traffic injected in the network. The workload is composed of 8 different TCs: Four QoS TCs and four best-effort TCs.

Table 3. Traffic injected per host.

TC	Name	% BW	Packet size	Notes
7	Network Control	1	[64,512] bytes	self-similar
6	Audio	16.333	128 bytes	CBR 64 KB/s connections
5	Video	16.333	[64,2048] bytes	750 KB/s MPEG-4 trc.
4	Controlled Load	16.333	[64,2048] bytes	CBR 1 MB/s connections
3	Excellent-effort	12.5	[64,2048] bytes	self-similar
2	Preferential Best-effort	12.5	[64,2048] bytes	self-similar
1	Best-effort	12.5	[64,2048] bytes	self-similar
0	Background	12.5	[64,2048] bytes	self-similar

Each TC has decreasing priority, such that TC 7 has the highest priority and TC 0 has the lowest. We follow the recommendations of The Network Processing Forum Switch Fabric Benchmark Specifications [3].

5.1 Simulation results

In this section, the performance of our proposals is shown. We first study the results of QoS traffic. In Figure 1, we show the latency of *Network Control* traffic. For this traffic, it is desireable a latency as low as possible. This is achieved by our architecture, with results very similar to those of the *Traditional 8 VCs* case. In Figure 1 (right), we see the CDF of latency at a load of 100%. In this case, our proposal offers a slightly more variable latency distribution compared with the *Traditional 8 VCs* case. The performance of the *Traditional 2 VCs* case is the worst for *Network Control* traffic. *Audio* traffic results are similar and not shown.

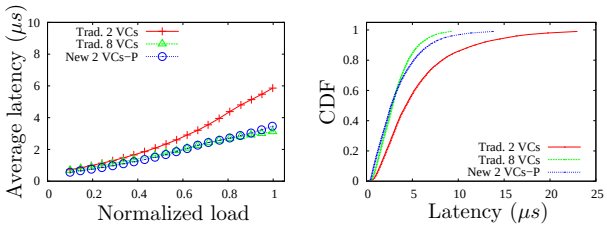

Figure 1. Results for Network Control traffic.

The *Video* traffic (Figure 2) is very bursty, since it involves the transmission of large video frames (around 40 Kbytes). This characteristic increases the latency and jitter results. However, note that even in this case, our proposal offers results very similar to those of the *Traditional 8 VCs* architecture. On the other hand, the three architectures we are studying offer 100% throughput to the *Controlled Load* traffic due to the CAC.

In Figure 3 we can see the throughput for the best-effort TCs. In these cases, the *Traditional 2 VCs* approach produces the same performance for all the TCs, which is an inadequate behavior, because *Excellent-effort* and *Preferential Best-effort* traffic should have better performance. The reason for this inadequate

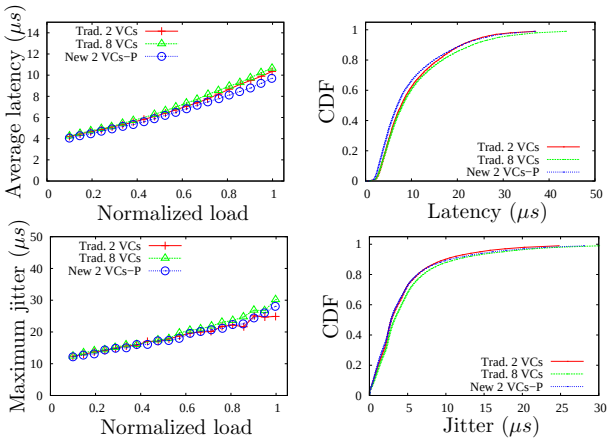

Figure 2. Results for Video traffic.

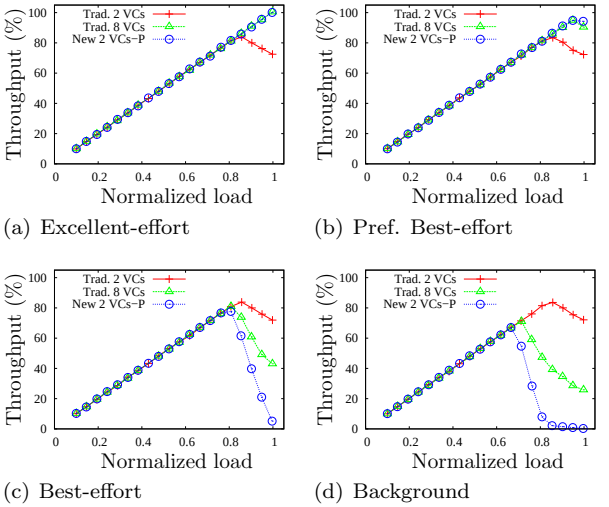

Figure 3. Throughput of best-effort traffic.

behavior is that in the *Traditional 2 VCs* model, all the best-effort classes look the same for the schedulers at both the network interfaces and the switches.

On the other hand, the arbiters using our technique take into account the priority of the packets, even if they share the same VC. For that reason, our proposal, which devotes a single VC in the switches for all the best-effort TCs, can provide a behavior similar to that of the *Traditional 8 VCs* approach, which uses 4 VCs for the best-effort TCs.

The *Best-effort* and *Background* TCs obtain a slightly worse performance with our technique if we compare it with the performance of the *Traditional 8 VCs* case. This is due to a lower global throughput of the network using our technique, but it only affects the TCs with the lowest priority, which is alright.

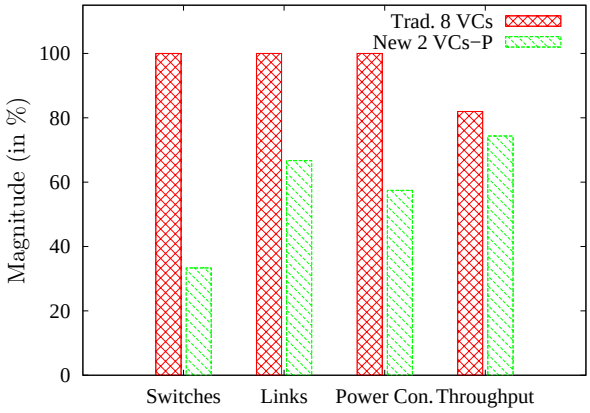

Figure 4. Summary of trade-offs of *New 2 VCs-P* proposal, compared with *Traditional 8 VCs* case.

Figure 4 summarizes the trade-offs of using our *New 2 VCs-P* proposal. As can be seen, there is a very noticeable reduction in chip and link counts and, therefore, in the associated power consumption of the interconnection network. On the other hand, the global throughput achieved is slightly lower (only a 10% reduction), but the reduction on the component count would lead to a much more cost-effective solution.

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 for QoS traffic and a small degradation in the performance of best-effort TCs.

6 Conclusions

In [7], we presented a proposal to use only two VCs at each switch port to provide QoS support. One 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 that paper, we showed preliminary results using multimedia traffic in an uniform scenario, without a clear study on how to apply this VC reduction.

In this paper, we explore a switch design which benefits from that proposal. We examine its feasibility as a single-chip switch and the hardware constraints that would have. We also compare, through simulation, the performance of this design with that of more traditional architectures. We find that we can provide performance very similar to a more complex architecture, but reducing the cost and the power-consumption of the interconnection network.

References

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

[2] 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.

[3] 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.

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

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

[6] 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.

[7] 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.

[8] N. W. McKeown. The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking*, 7:188–201, 1999.

[9] 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.

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