QoS Support for Video Transmission in High-speed Interconnects

A. Martínez[1], G. Apostolopoulos[2], F. J. Alfaro[1], J. L. Sánchez[1], and J. Duato[3]

[1] Dept. de Sist. Inform., Univ. de Castilla-La Mancha, 02071 - Albacete, Spain
{alejandro,falfaro,jsanchez}@dsi.uclm.es
[2] CARV Lab., Inst. of Computer Science - FORTH, 71110 - Heraklion, Crete, Greece
georgeap@ics.forth.gr - Member of HiPEAC
[3] GAP - DISCA, Univ. Politécnica de Valencia, 46071 - Valencia, Spain
jduato@disca.upv.es - Member of HiPEAC

Abstract. Multimedia traffic presents some special requirements that are unattainable with a best-effort service. Current interconnect standards provide mechanisms to overcome the limitations of the best-effort model, but they do not suffice to satisfy the strict requirements of video transmissions. This problem has been extensively addressed at the general networking community. Several solutions have arisen, but they are too complex to be applied to high speed-interconnects. In this paper, we propose a network architecture that is at the same time compatible with the requirements of high-speed interconnects and provides video traffic with the QoS it demands.

Keywords: *QoS, Switch Design, Scheduling, Virtual Channels, Clusters*

1 Introduction

The last decade has witnessed a vast increase in the amount of information and services available through the Internet. Clusters of PCs have emerged as a cost-effective platform to implement these services. They provide service to thousands or tens of thousands of concurrent users. These users usually demand specific quality of service (QoS) requirements [1].

In the next section, we will introduce the InfiniBand and PCI AS high-speed interconnect standards. These technologies provide mechanisms for QoS support that consist of the segregation of the traffic in traffic classes (TCs), virtual channels (VCs), and a mechanism to map the TCs to the VCs and then provide scheduling for the VCs. However, the scheduling algorithms proposed [2, 3] are fairly simplistic and fail to provide certain kinds of traffic with the requirements they demand. For instance, video traffic is usually very concerned about jitter, and much less about latency [4].

There has been a very substantial body of work on mechanisms for providing QoS guarantees for packet switches[4]. Usually these works assume that the packet

[4] We will use the terms *packet switches* and *packet networks* to refer to general networking technologies.

switch has significant amount of resources, in particular large random access buffers. Moreover, in packet networks packets may be dropped when buffering capacity is exceeded and latencies can be large. Thus, most of the scheduling policies focus on controlling packet losses and delays. On the other hand, in high-speed interconnects, switches are single-chip and, thus, have considerably more limited resources, latencies are very small due to the small geographical extent of the interconnect, and typically flow control is employed preventing packet drops. As a result, the interconnect environment requires special attention when developing QoS scheduling policies.

In this work, we introduce a QoS architecture that is tailored for the interconnect environment. By moving the complexity of the packet scheduling to the host network interfaces we are able to keep the packet processing in the switches very simple. We show how with a simple QoS architecture, we can support the QoS requirements of multiple different types of traffic: high-priority control traffic, medium priority video traffic, and low priority best-effort. All the types of traffic can coexist and receive the desired QoS without interfering with each other. At the same time, we are able to achieve high utilization of the interconnect and all these without requiring more than two queues per switch port. This allows us a significant reduction in the switch complexity, which is critical if we want to scale up the switch port densities.

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 offer QoS support. Details on the experimental platform are in Section 4 and the performance evaluation is presented in Section 5. Finally, Section 6 summarizes the results of this study and identifies directions for future research.

2 Related Work

In this section, we will review the special characteristics of video traffic and its requirements. Next, we will analyze the two most recent technologies for high-speed interconnects (InfiniBand and PCI AS) and how they can provide QoS. Finally, we will review some algorithms for the provision of QoS to multimedia flows in general networking.

2.1 Video traffic's characteristics and requirements

Video sequences are composed of a set of video frames that are generated at regular intervals. Compression algorithms produce frame patterns in which some frames are smaller than others. More specifically, there are intra-coded frames, which are basically normal pictures compressed with an algorithm like JPEG; besides, there are inter-coded frames, which only encode the differences with some neighbor frames. Therefore, frame size presents a lot of variability [5].

Ideally, the receiver should receive a complete frame exactly each inter-frame interval (usually 40 milliseconds). This is measured by *jitter*, the variation of the latency of two consecutive packets of the same flow [6]. This is important

because if frames arrive too late they are obviously useless, but if they arrive too soon, they can overflow the reception buffer.

Furthermore, a latency of less than 100 milliseconds is desirable for interactive video [4]. This includes video-conference and video on demand, when the watcher has the ability to stop and peek through the sequence. Moreover, although there is some tolerance to packet loss, it should be very reduced.

2.2 QoS support for multimedia traffic in packet networks

Over the last years there has been extensive work on how to schedule resources of a packet switch to provide guaranteed performance to traffic. The switch resources that need to be scheduled are buffer space (usually at the outgoing port) and link capacity. Both are managed through a *service discipline*. Typically, packet switch buffers are fairly large and support random access. When buffers become full, packets are dropped. Thus, general packet switches can introduce packet loss when their resources are oversubscribed. Performance guarantees usually include bounds on packet loss, delay, jitter, and transmission rate or throughput. A large number of service disciplines have been proposed (see [7] for an overview) each specifically targeted for providing certain types of guarantees.

The service disciplines operate at the flow level and consequently can provide different QoS guarantees to individual flows. An example of such flow oriented QoS architectures is the QoS architecture of ATM [8] and the Integrated-Services model [9] that was proposed for Internet QoS in mid-90s. Since such per-flow scheduling can prove a bottleneck as the number of flows grows, aggregate-QoS architectures have been proposed where QoS is provided collectively to all flows that belong to a certain class of service. There are only a few such classes of service, but flows now get only aggregate and not individual QoS. An example of such a QoS architecture is Differentiated Services [10], which is used to provide limited QoS in parts of the Internet today.

2.3 QoS support in new high-speed interconnects

When compared with a generic packet switch, high-speed interconnect switches have some important differences mostly because of their much simpler and compact implementation. Firstly, flow control is commonly used to throttle the incoming traffic, and thus usually there are no packet drops due to running out of buffer space. Buffers themselves may be smaller than what one would expect from a generic packet switch. Furthermore, access to these buffers may be more restricted and random access may not be possible due to the strict time limitations. Similarly, the number of different queues may be limited.

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 [2]. 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 [11].

On the other hand, PCI Express Advanced Switching (AS) architecture is the natural evolution of the traditional PCI bus [3]. It defines a switch fabric architecture that supports high availability, performance, reliability and QoS. AS ports incorporate up to 20 VCs (16 unicast and 4 multicast) that are scheduled according to some QoS criteria. Is is also possible to use a connection admission control implemented in the fabric management software.

These proposals, therefore, permit to use a significant number of VCs to provide QoS support. However, implementing a great number of VCs would require a significant fraction of silicon area and would make packet processing slower. Moreover, there is a trend of increasing the number of ports instead of increasing the number of VCs per port [12]. In general, the number of queues per port can have a significant effect on the overall complexity and cost of the interconnect switch. It is important to attempt to provide effective QoS with a number of queues as small as possible. Indeed, our proposal addresses this very effectively.

3 Architecture for QoS Support

Deadline[5]-based policies are among the most effective scheduling policies in packet networks. These policies operate as follows: each packet is labeled with a deadline and, thereafter, the switches solve all the scheduling and output conflicts choosing always the packet with the smallest deadline. This usually requires that all the packets in the buffers are taken into account for scheduling and, thus, random access buffers are needed. Another alternative is to set up heap buffers, that always keep at the top the packet with the lowest deadline [13, 14]. However, these implementations are too expensive for high-speed interconnects. As far as we know, nobody has tried to adapt this kind of algorithms to this environment.

Note that in packet networks (like Internet) the deadline would be recomputed at each hop. This is not reasonable in this case. For high-speed networks, which span over a much shorter area, deadline would be computed once at the interfaces.

When traffic is regulated, the switches can avoid random access buffers and just take into account the first packet at each input buffer. The idea is that traffic coming from the interfaces has already been scheduled and is coming in descending order of deadlines. This being so, it is possible to just consider the first packet at each queue, being confident that packets coming afterward have higher deadlines.

The behavior of the switch would be analogous to a sorting algorithm: if the switch has as input sorted chains of packets and has to produce at the output a sorted sequence, it only needs to look at the first packet of each input.

[5] We will use the term *deadline* as a tag contained in the header of packets used for scheduling.

Let us have a look at the possible limitations of this algorithm:

- The traffic must be regulated. If a link is oversubscribed, we cannot guarantee that flows will get the demanded QoS. However, regulation on the traffic is always mandatory to provide strict guarantees on latency and throughput.
- The deadlines must not be recomputed. If we allowed the deadlines to change during the life of the packet, we would not be able to assure that the order established at the interfaces would be valid. However, in the high-speed interconnects environment latencies are expected to be very short and there is no need to recompute these deadlines.
- The packets are not coming always in order from the interfaces. The above scheduling policy assumes that packets arrive from interfaces ordered according to their deadlines. This may not be true all the time though. For example, immediately after a packet with large deadline has departed from the interface, a high priority small deadline packet arrives and is sent behind the large deadline packet. This will violate our assumptions and degrade the service offered to the high-priority packet. In order to limit the occurrences of these out-of-order packets, we use a non-work conserving deadline scheduling policy. For video traffic, an eligibility time (the minimum time when packets are allowed to leave) is also provided to reduce jitter. In this way, by bounding the cycle where packets are available for transmission, we can also bound the maximum distance between the deadlines of two out-of-order packets. In any case, as we will see in the evaluation section, the impact of out of order packets is rather limited mostly due to the low latency of the interconnect for regulated traffic.

In order to support control traffic, which is usually unregulated, we can safely assume that it will never congest any link by itself. This kind of traffic usually requires negligible bandwidth but demands low latencies. We can mix it with video traffic by providing small deadline tags.

Best-effort traffic must also be supported. In this case, high bandwidth is demanded and congestion may appear since this traffic is not regulated. Therefore, in order to not disturb regulated traffic, it requires a separate VC. Moreover, absolute priority should be given to regulated traffic over unregulated traffic.

Summing up, our proposal consists in a network architecture able to deal with three different classes of traffic: control traffic, which demands little bandwidth but low latency; video traffic, which demands guaranteed throughput, bounded latency and low jitter; and best-effort traffic, which demands as much bandwidth as possible. This is achieved with only two VCs and a feasible implementation on a single chip, as is usually the case in high-speed interconnects.

3.1 Generating deadlines for video traffic

We propose a simple scheme to label video packets at the interfaces in order to provide these requirements. Each packet that belongs to a particular frame would receive a deadline covering the whole inter-frame period. This would smooth the

burst along this period of time (see Figure 1 for an example). In other words, the first packet of the frame would receive a deadline near to the actual clock cycle, while the last one would receive a deadline equal to the clock plus the inter-frame time. The intermediate packets would be uniformly distributed between these.

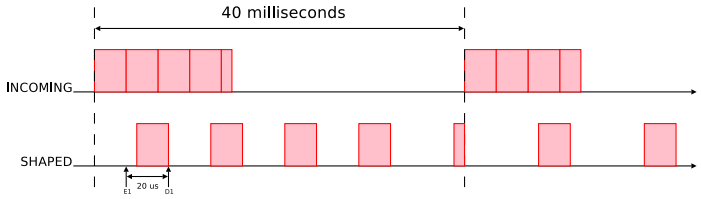


Fig. 1. Traffic shaping at network interfaces.

In addition to deadlines, each packet would have an eligibility time. No packet would be allowed to leave the interface before this time has passed, in order to guarantee that the jitter would be as close to 0 as possible. We are computing the eligibility time of a packet as its deadline minus a constant value. We have found that 20 μs works well for this value.

With this strategy, buffers must have capacity for one whole frame per active video connection in the worst case. Note that this amount is also required in a work conserving alternative: the worst case is the same.

Therefore, the next packet to be chosen at a network interface would be the one with the lowest deadline from those which are eligible (the eligibility cycle has passed). This requires to keep two ordered queues at the interface, one with non-ready packets, in eligibility order, and another with eligible packets, in deadline order. This is affordable in these devices, since significant memory and processing capacity are available here.

4 Simulation conditions

We have performed the tests considering three cases. First, we have tested the performance of our proposal, which uses 2 VCs at each switch port. It is referred to in the figures as *New 2 VCs*. We have also performed tests with switches using ideal, but impractically expensive random access buffers. In this case, it is referred to in the figures as *RAM buffers*. Note that we assume the same delays for the switch as in our proposal, which is not realistic, but serves us to examine which is the impact of order errors. Finally, we have also tested a traditional approach, based on the specifications of InfiniBand and PCI AS, with 4 VCs, noted in the figures as *Traditional 4 VCs*. In this case, there is a VC for each traffic class considered, both at the switches and at the network interfaces.

In the three cases, we have used 16 port switches, 8 Gbits/s links, and 8 Mbits of total buffering at each switch. To cope with the inefficiencies of the scheduler and packet segmentation overheads[6], the crossbar core operates twice as fast as the external lines (internal speed-up of 2.0).

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

The network used to test the proposals is a butterfly multi-stage interconnection network (MIN) with 64 end-points. The actual topology is a folded (bidirectional) perfect-shuffle. We have chosen a MIN because it is a usual topology for clusters. However, our proposal is valid for any network topology, including both direct networks and MINs. No packets are dropped at the switches because we use credit-based flow control at the VC level. However, if a video packet has to wait more than 100 milliseconds at the interface, it is completely useless and is, therefore, dropped.

In Table 1, the characteristics of the modeled traffic are included. Traffic consists in three categories: network control, video, and best-effort. The first category models short control messages that require short latency but demand negligible bandwidth. Video traffic is taken from actual MPEG-4 video sequences, which produce a video frame each 40 milliseconds, approximately. This is very bursty traffic and will heavily degrade the performance of the network. The required results for video traffic according to [4] are guaranteed bandwidth, latency below 100 milliseconds and jitter as short as possible. Finally, we have modeled two classes of best-effort traffic: *Best-effort* and *Background*. The former demands as much bandwidth as possible, while the latter would require the remaining bandwidth, if any.

Table 1. Traffic injected per host.

TC	Name	% BW	Packet size	Notes
0	Network Control	1	[64,512] bytes	self-similar
1	Video	49	[64,2048] bytes	750 KByte/s MPEG-4 traces
2	Best-effort	25	[64,2048] bytes	self-similar, burst = 20
3	Background	25	[64,2048] bytes	self-similar, burst = 20

The self-similar traffic is composed of bursts of packets heading to the same destination. The packets' sizes are governed by a Pareto distribution, as recommended in [15]. In this way, many small size packets are generated, with an occasional large size packet. The periods between bursts are modeled with a Poisson distribution. If the burst size is long, it should show worst-case behavior because, at a given moment in time, many packets are grouped going to the same destination.

5 Simulation results

In this section, we show the performance of our proposal. We have considered three common QoS metrics for this performance evaluation: throughput, latency, and jitter. Note that packet loss is only possible at the interfaces for video packets, thereafter there is a credit-based flow control.

The performance of *Network Control* traffic is shown in Figure 2. We can see that the three alternatives offer good latency results, both average and maximum. The differences between the *Traditional 4 VCs* case and the two based

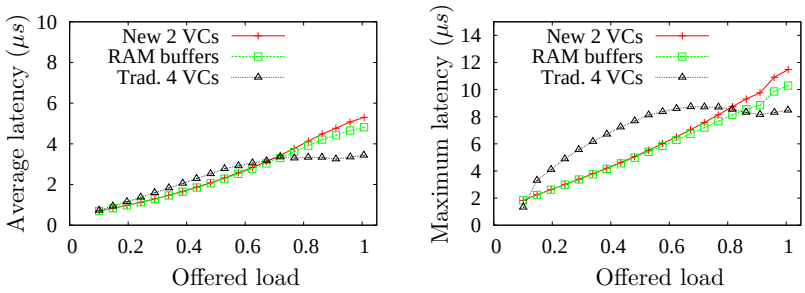


Fig. 2. Performance of *Network Control* traffic.

on deadlines come from the fact that in the latter ones only two VCs are used. Therefore, the *Network Control* traffic shares the VC with the *Video* traffic. We can conclude that this reduction of VCs is not causing problems, there is only a rather small performance loss at high load.

On the other hand, the small differences that can be observed between our proposal and the *RAM buffers* case are due to the order errors we discussed in Section 3. However, we see that the impact of this issue is very limited, even after mixing a few of high-priority packets with lots of low-priority video traffic.

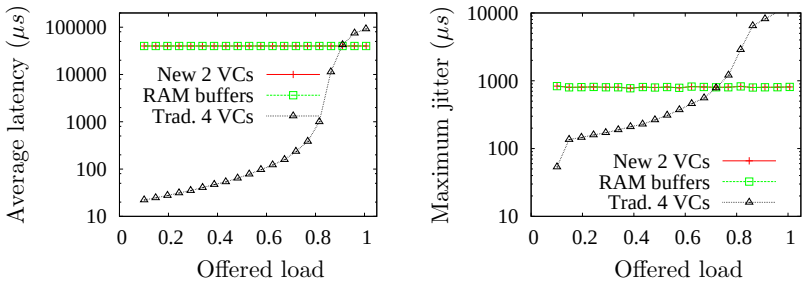
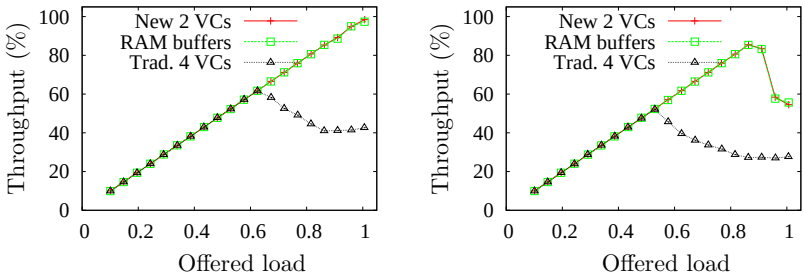


Fig. 3. Performance of *Video* traffic.

Figure 3 shows the performance of video traffic in terms of average latency and maximum jitter. The deadline-based alternatives succeed in providing a constant latency of 40 milliseconds for all the video frames independently of the load. This is also reflected in jitter, which is low at all load levels. Note that this results are referred to the full video frames. Individual packets have much lower latency, of course. On the other hand, results for *Traditional 4 VCs* case are not so good, because latency varies with load and at high load the latency and jitter reach unacceptable values. Our two-VC scheme, with the traffic shaping based on deadlines, offers much better performance with less VCs.

To finish this part of the study, we will look at best-effort traffic results (Figure 4). We can see that the two deadline-based alternatives offer much better throughput to this kind of traffic than the *Traditional 4 VCs* architecture. Moreover, note that although we are using just one VC with our proposal for both best-effort TCs, there is QoS differentiation between the two classes of best-effort



(a) Best-effort (b) Background

Fig. 4. Performance of best-effort traffic.

traffic: *Best-effort* traffic keeps good performance at high load, while *Background* decays.

We can conclude at this point that our proposal offers as good latency for *Control Traffic* as the other two options; offers as good jitter for video traffic as the unfeasible *RAM buffers* architecture due to the traffic shaping, while the *Traditional 4 VCs* case fails in this; and offers as good throughput as the other alternatives for bursty, unbalanced best-effort traffic.

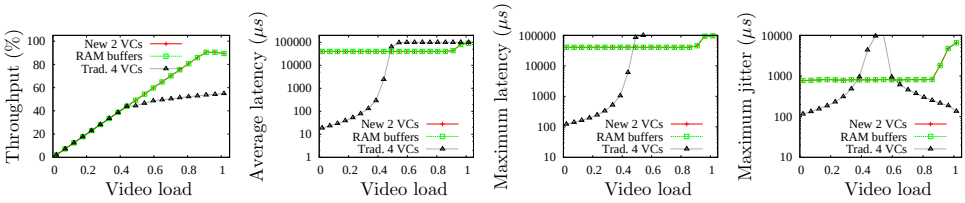


Fig. 5. Performance of full *Video* traffic injection.

In the next experiment, we examine the capacity of the three alternatives to deal with video traffic. We vary the load from 0 to full input injection of video sequences, as can be seen in Figure 5. We observe that the *Traditional 4 VCs* case offers good performance up to a video load of 45%. Afterwards, latency is so bad that packets start being dropped after having waited 100 millisecond at the network interfaces. This decreases the throughput results. As a side effect, jitter actually decreases when the load is very high and almost all the packets that leave the interface have waited near 100 milliseconds.

On the other hand, the deadline based alternatives can cope with a video load of 85% before the performance in terms of latency, jitter or throughput is affected. Note that before that point, average and maximum latency is 40 milliseconds (for the full frames) and, therefore, jitter is very low.

6 Conclusions

In this paper, we propose a novel technique for supporting very efficiently deadline-based scheduling policies in a high-speed interconnect. Based on these policies, we are able to offer excellent performance compared with traditional solutions proposed in specifications like InfiniBand or PCI AS. Moreover, we show that the performance of our proposal is not far from what would be obtained using expensive random access buffers. We are able to use only 2 VCs per port, reducing considerably the cost and complexity of the interconnect switch. Even with only 2 VCs, we are able to provide QoS differentiation between multiple different classes of traffic and improve network utilization.

References

1. Miras, D.: A survey on network QoS needs of advanced internet applications. Technical report, Internet2 - QoS Working Group (2002)
2. InfiniBand Trade Association: InfiniBand architecture specification volume 1. Release 1.0. (2000)
3. Advanced switching core architecture specification. Technical report, (available online at `http://www.asi-sig.org/specifications` for ASI SIG members)
4. IEEE: 802.1D-2004: Standard for local and metropolitan area networks. `http://grouper.ieee.org/groups/802/1/` (2004)
5. Moving Picture Experts Group: Generic coding of moving pictures and associated audio. Rec. H.262. Draft Intl. Standard ISO/IEC 13818-2 (1994)
6. Duato, J., Yalamanchili, S., Lionel, N.: Interconnection networks. An engineering approach. Morgan Kaufmann Publishers Inc. (2002)
7. Guerin, R., Peris, V.: Quality-of-service in packet networks: basic mechanisms and directions. Comput. Networks **31** (1999) 169–189
8. Forum, A.: ATM Forum traffic management specification. Version 4.0. (1995)
9. Braden, R., Clark, D., Shenker, S.: Integrated Services in the Internet Architecture: an Overview. Internet Request for Comment RFC 1633, Internet Engineering Task Force (1994)
10. Blake, S., Back, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services. Internet Request for Comment RFC 2475, Internet Engineering Task Force (1998)
11. Alfaro, F.J., Sánchez, J.L., Duato, J.: QoS in InfiniBand subnetworks. IEEE Transactions on Parallel Distributed Systems **15** (2004) 810–823
12. Minkenberg, C., Abel, F., Gusat, M., Luijten, R.P., Denzel, W.: Current issues in packet switch design. In: ACM SIGCOMM Computer Communication Review. (2003)
13. Ioannou, A., Katevenis, M.: Pipelined heap (priority queue) management for advanced scheduling in high speed networks. In: Proceedings of the IEEE International Conference on Communications (ICC'2001). (2001)
14. Yun, K.Y.: A terabit multiservice switch. IEEE Micro **21** (2001) 58–70
15. Jain, R.: The art of computer system performance analysis: techniques for experimental design, measurement, simulation and modeling. John Wiley and Sons, Inc. (1991)