A Switch Architecture Guaranteeing QoS Provision and HOL Blocking Elimination

Alejandro Martínez, Pedro J. García, Francisco J. Alfaro, José L. Sánchez, José Flich, Francisco J. Quiles, and José Duato

Abstract—Both QoS support and congestion management techniques become essential to achieve good network performance in current high-speed interconnection networks. The most effective techniques traditionally considered for both issues, however, require too many resources for being implemented. In this paper, we propose a new cost-effective switch architecture able to face the challenges of congestion management and, at the same time, to provide QoS. The efficiency of our proposal is based on using the resources (queues) used by RECN (an efficient Head-of-Line blocking elimination technique) also for QoS support, without increasing queue requirements. The provided results show that the new switch architecture is able to guarantee QoS levels without any degradation due to congestion situations.

Index Terms—High-speed interconnection networks, quality of service, congestion management.

1 Introduction

The use of high-speed interconnection networks has become a major need on the design of several computing and communication systems, including systems for parallel computing. They provide the low-latency and high-performance demanded by parallel applications. The proliferation of systems based on high-speed networks has increased the researchers' interest on developing techniques for improving the performance of such networks. Moreover, due to the increase in network components' cost and power consumption, it is nowadays very important to propose efficient and cost-effective techniques, trying to use a minimum number of network resources while keeping network performance as high as possible.

In that sense, many techniques have been proposed for solving the problem of network performance degradation during congestion situations. Actually, congestion dramatically degrades network performance due to the appearance of Head-of-Line (HOL) blocking [1] during congestion situations. This phenomenon happens when a packet at the head of a queue temporarily blocks,[1] preventing other packets at the same queue from advancing, even if they request available resources further ahead. This may cause that data flows not contributing to congestion advance at the same speed as congested flows, thereby degrading network performance.

However, although both congestion and HOL blocking are well-known phenomena in interconnection networks, efficient proposals for managing congestion in modern high-speed networks are rare. On one hand, traditional simple solutions are not suitable for modern interconnects. For instance, network overdimensioning is not currently feasible due to cost and power consumption constraints. On the other hand, more elaborated techniques that have been specifically proposed for solving the problems related to congestion have not been really efficient until very recently. For instance, if Virtual Output Queuing (VOQ) [5] is used, there must be at each port as many queues as end nodes in the network, and any incoming packet is stored in the queue assigned to its destination. The aim of this policy is to prevent flows addressed to different destinations from sharing the same queue, thereby avoiding HOL blocking. However, although this scheme is very effective, it is not efficient (and even not feasible for medium or large networks) since it requires a considerable number of queues at each switch port, and the silicon area required for implementing such a number of buffers strongly increases the cost. A variation of VOQ uses as many queues at each port as output ports in a switch [6], thereby reducing queue requirements. However, this scheme does not completely eliminate HOL blocking since only switch-level HOL blocking is eliminated.

Recently, a new HOL blocking elimination technique has been proposed: Regional Explicit Congestion Notification (RECN) [7]. RECN completely eliminates HOL blocking while requiring a small number of resources regardless of network size, thus being a really efficient, scalable, and cost-effective technique. Specifically, in order to avoid HOL blocking between congested and noncongested flows, RECN identifies congested flows and puts them in special dynamically-assigned set aside queues (SAQs). Moreover, it uses a status-based flow control to avoid a congested flow from taking all the buffer space.

1. We are considering lossless networks like InfiniBand [2], Quadrics [3], or Myrinet [4].

- *A. Martínez is with Intel Barcelona Research Center, Intel Labs, Universitat Politècnica de Catalunya C/ Jordi Girona 29, Edificio Nexus II, 3a planta 08034, Barcelona, Spain. E-mail: AlejandroX.Martinez@intel.com.*
- *P.J. García, F.J. Alfaro, J.L. Sánchez, and F.J. Quiles are with the Computing Systems Department, Escuela Superior de Ingeniería Informática, Universidad de Castilla-La Mancha, Campus Universitario, s/n 02071, Albacete, Spain. E-mail: {pgarcia, falfaro, jsanchez, paco}@dsi.uclm.es.*
- *J. Flich and J. Duato are with the Department of Computer Engineering (DISCA), Technical University of Valencia, Camino de Vera, s/n 46071, Valencia, Spain. E-mail: {jflich, jduato}@disca.upv.es.*

Manuscript received 10 Aug. 2007; revised 15 Feb. 2008; accepted 2 Apr. 2008; published online 16 Apr. 2008.
Recommended for acceptance by M. Ould-Khaoua.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2007-08-0271.
Digital Object Identifier no. 10.1109/TPDS.2008.62.

Another way for improving network performance, from an application point of view, is to use techniques for providing quality of service (QoS). If we want to provide communication services for several types of applications, the QoS consists in assuring a minimum performance to each application, despite the behavior of the rest of traffic classes. For instance, if we want to guarantee a minimum bandwidth to each traffic class, this minimum must be provided even if there are sources injecting more traffic than they are allowed to.

The usual solution to this problem is to provide a separate virtual channel (VC) for each traffic class [8], [2], [9] and perform a correct output scheduling at switches. These VCs provide also separate domains of flow control, i.e., there is a separate credit counter for each VC. In this way, two objectives are achieved: first, there is no HOL blocking between traffic classes, and second, a single traffic class cannot take all the available buffer space, so buffer hogging is avoided. Consequently, the performance of different traffic classes only depends on the scheduling and on the amount of injected traffic of each class. Unfortunately, VCs are costly to implement, since each new VC requires more buffer space and a more complex scheduler (more packets to be considered at each scheduling cycle). To alleviate this problem,[2] some techniques have been proposed for reducing the number of VCs while keeping QoS guarantees. For instance, in [10], a technique for providing full QoS support with only two VCs was proposed. Moreover, in [11], this technique was combined with an RECN by duplicating all the RECN queue structures in two VCs, in order to obtain a switch architecture offering QoS provision and congestion management at the same time.

However, we think that it is possible to achieve full QoS provision in an even more cost-effective efficient way. Specifically, we propose in this paper a switch architecture that improves the basic RECN mechanism so it can provide also QoS guarantees without introducing additional VCs or queues. The newly proposed architecture takes into account the clear parallelism between the tasks performed by RECN and the use of VCs for QoS provision and exploits the RECN queue structure from a new original approach that efficiently uses these resources for managing congestion while providing QoS at the same time. The benefits of the proposal are obvious since these two important issues on interconnect design would be afforded by a single and very efficient architecture.

The rest of the paper is organized as follows: In Section 2, RECN is described. Next, in Section 3, the proposed switch architecture is explained in detail. Section 4 shows an evaluation of the new proposal, based on simulation results. Finally, in Section 5, some conclusions are drawn.

2 Regional Explicit Congestion Notification

RECN is based on a key idea: if HOL blocking is completely eliminated, congestion may exist while being harmless. Like VOQ, RECN tries to separate congested and noncongested flows by storing them into different queues, with the aim of eliminating HOL blocking. Specifically, RECN adds a set of additional queues (SAQs) to the standard queue at every input and output port of a switch. While standard queues store noncongested packets (RECN assumes that packets from noncongested flows can be mixed in the same buffer without producing significant HOL blocking), SAQs are dynamically allocated for storing packets passing through a specific congested point. SAQs at each switch port are controlled by means of a Content Addressable Memory (CAM), in such a way that each CAM line contains information for managing an associated SAQ, including the information required for addressing a congested point.

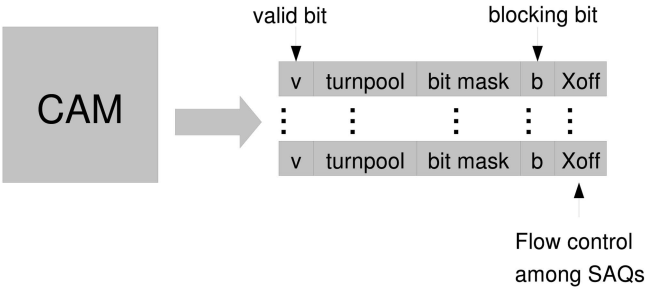

Fig. 1. RECN CAM organization for ASI networks.

In that sense, RECN addresses network points by means of the routing information included at the packet header, assuming that source routing is used. For instance, Advanced Switching Interconnect (ASI) [9] packet headers include a turnpool made up of 31 bits, which contains all the turns (offset from the input port to the output port) for every switch in a route. Therefore, in ASI networks, CAM lines include turnpools addressing congested points,[3] as can be seen in Fig. 1, and these turnpools can be compared to the turnpool of the incoming packet, in order to know if it will cross the corresponding congested point. By doing this, congested packets can be easily detected.

In order to identify a network point as congested, RECN implements different congestion detection mechanisms at each switch input and output port. At any output port, whenever the standard queue fills over a given threshold, congestion is detected at this point, and a notification is sent to any switch input port sending packets to the congested switch output port (Fig. 2a). These notifications include the turnpool required to reach the congested point from the input port receiving the notification. Upon reception of a notification, an input port must allocate a new SAQ and fill the corresponding CAM line with the received turnpool.

On the other hand, at input ports, the standard queue is divided into several detection queues, one per output port. Whenever a detection queue fills over a given threshold, congestion is detected at the corresponding output port, and a new SAQ associated to this congested point is automatically allocated at the input port. Immediately, a notification including the turnpool of the newly allocated SAQ is sent to the output port of the upstream switch, where a new SAQ should be subsequently allocated.

Note that for both input and output congestion detection mechanisms, detection threshold must be tuned for achieving an optimal performance [7].

2. Note that for high-speed single-chip switches proposals requiring many VCs could be considered if external DRAM is available for implementing the buffers. However, in this case, the low latencies demanded by the QoS-requiring traffic could not be provided.

3. Note, however, that RECN can be applied to other interconnect topologies as long as they use source routing, so in these case, CAM lines structure would vary.

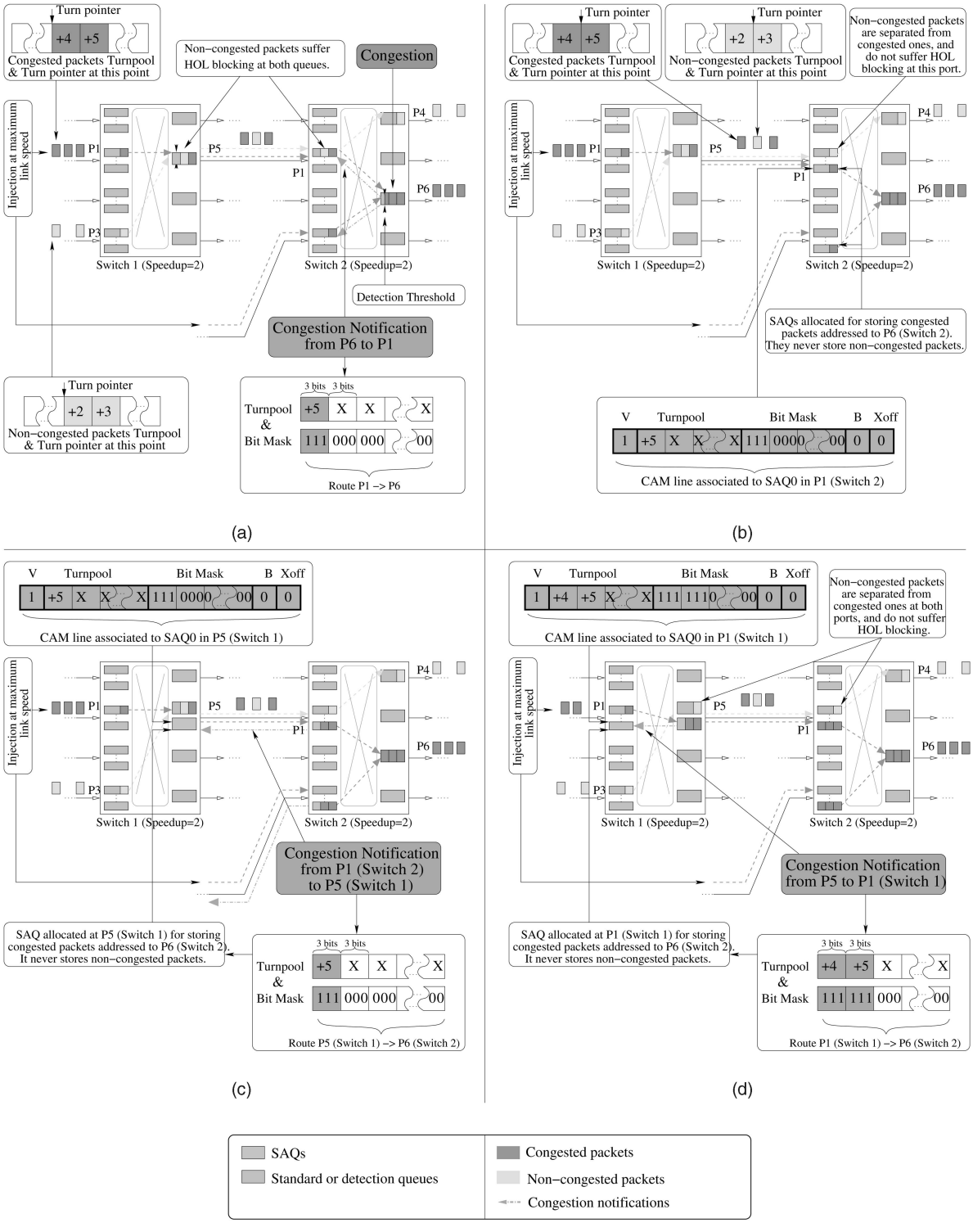

Fig. 2. RECN congestion detection and allocation of SAQs.

Every packet received at any port with allocated SAQs is stored in an SAQ if it will pass through the congested point associated to that SAQ. Otherwise, the incoming packet is stored in the standard (or detection) queue. In this way, noncongested packets are always separated from the congested ones, thereby preventing the appearance of HOL blocking at the port (Fig. 2b).

Furthermore, if any SAQ becomes congested, a notification is sent upstream, and the receiving input or output port allocates a new SAQ (Figs. 2c and 2d). This procedure can be repeated until these notifications reach the sources. Therefore, there will be SAQs for storing congested packets at every point where, otherwise, these packets could produce HOL blocking. Moreover, congested packets cannot fill completely the port memory, since RECN uses an SAQ-specific X_{on}/X_{off} (Stop & Go) flow-control mechanism. An X_{off} bit at each CAM line controlling an active SAQ is used to implement the flow-control mechanism. Packets can be

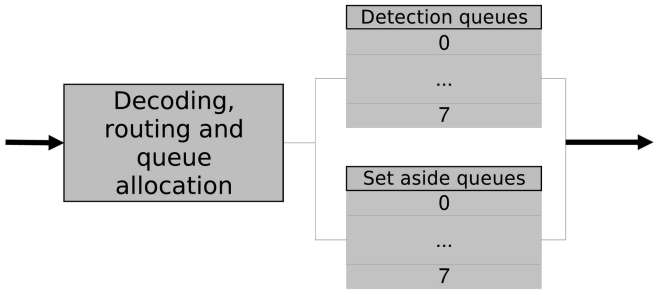

Fig. 3. Logical input port organization.

forwarded from an SAQ only if the associated X_{off} bit is reset.

RECN also detects congestion vanishing at any point, in such a way that SAQs can be dynamically deallocated. Specifically, the conditions for deallocating an SAQ are the following: it must be empty, and it must be in X_{on} state (the associated X_{off} bit is reset). Note that these conditions allow a distributed SAQ deallocation, in such a way that an SAQ can be deallocated independently of other SAQs. Since deallocated SAQs can be reallocated for storing packets addressed to new congested points, this policy reduces the number of SAQs per port required for completely eliminating HOL blocking. Further details about RECN can be found in [7].

2.1 RECN and QoS

A high-performance interconnect must offer many features in order to be effective. Apart from congestion control, this paper also focuses on QoS provision, which is very important if there is a variety of applications sharing the network, becoming necessary to offer guarantees on performance. Since an efficient congestion control technique would allow to achieve the best performance near the saturation point, a combination of both features would improve network performance significantly.

However, when network designers want to combine different techniques, it may happen that resources are multiplied, and the resulting design is completely infeasible. For instance, if we want to support 16 VCs for different traffic classes and, at the same time, we want to provide VOQ at the switch level, with 16-port switches, the resulting switch design requires 256 queues per switch port, which is not usually possible to implement.

Therefore, the motivation of this work is to obtain an integrated solution for providing congestion management and QoS support. In particular, we use the same set of resources for both purposes. In that sense, the proposal that

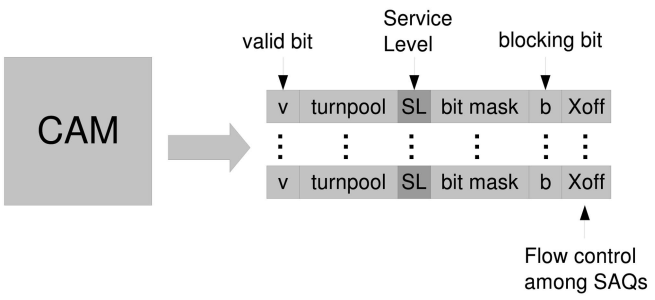

Fig. 4. New CAM organization.

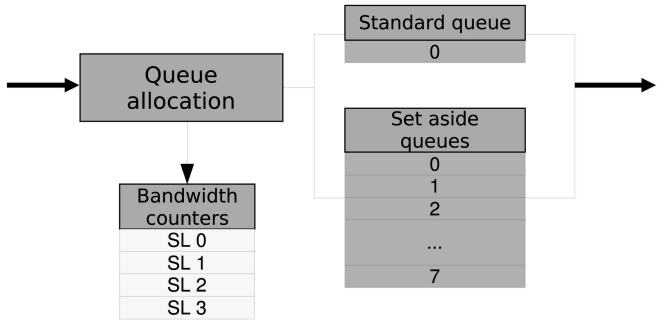

Fig. 5. Logical output port organization.

we present in Section 3 uses the RECN queue structure described in this section for both HOL blocking elimination (thereby turning congestion harmless) and QoS support. This is possible since this queue structure may be configured in such a way that RECN performs traffic isolation while still dealing with congestion. Thus, our proposal is an integrated cost-effective and (as we will show in the following sections) efficient solution for both congestion management and QoS support.

3 New Proposal for QoS Provision and HOL Blocking Elimination

As we have already mentioned, our proposal tries to exploit the resources RECN uses for eliminating HOL blocking (as explained above), in such a way that QoS could also be provided without increasing queue requirements. The following sections explain both the minimum changes necessary to introduce in the RECN architecture and the criteria that will allow us to use the modified architecture for providing both QoS and congestion management.

3.1 Proposed Architecture

The switch organization consists of a combination of input and output buffering, which is a usual design [12], [13]. Note that all the switch components are intended to be implemented in a single chip. This is necessary in order to offer the low cut-through latencies demanded by current applications.

The logical organization of an input port can be seen in Fig. 3. This is the standard scheme for a RECN input port, so there are as many detection queues as output ports (eight in this case[4]) for storing noncongested packets and a group of SAQs for storing congested packets. There are eight of these SAQs since it has been shown that eight or less SAQs are enough for eliminating HOL blocking almost completely [7]. The use of these queues is discussed later. A CAM is required at each input or output port in order to manage the set of SAQs. CAM organization for ASI networks[5] can be seen in Fig. 4. Each CAM line contains all the fields defined by RECN, plus a new field for storing the service level the corresponding SAQ is assigned to.

Fig. 5 shows the logical organization of output ports. In this case, detection queues are not needed, and a unique

4. For the sake of simplicity, we assume 8-port switches and four service levels. Anyway, architectures with a different number of ports or service levels could be easily deducted.

5. Note that our proposal could be applied to any interconnect technology using source routing, so CAM organization could vary.

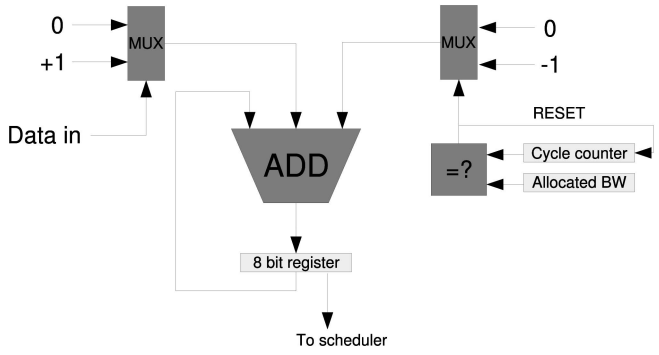

Fig. 6. Bandwidth counter.

standard queue is used for storing noncongested packets. Following also the RECN scheme for output ports, a set of SAQs (8) is also used at each output port. In addition to this RECN queue structure, our proposal introduces at the output ports a set of bandwidth counters, one per service level. These counters must dynamically compute the difference between the reserved bandwidth and the current bandwidth used for each service level. These counters are used for two purposes: first, to perform deficit round-robin (DRR) scheduling at the switches and, second, for congestion detection (both issues are detailed in the following sections).

The bandwidth counters' structure can be seen in Fig. 6. Basically, their behavior is the following: each time a block of 64 bytes from a packet[6] is scheduled to cross toward the output port of a switch, the bandwidth counter corresponding to the service level of the packet is increased by 1. We have assumed an 8-bit register for implementing the bandwidth counter, so its range of values is from—Kbytes to 8 Kbytes.[7] On the other hand, the same register must be automatically decremented at a rate that matches the bandwidth we guarantee to the corresponding service level. For instance, if 128 Mbytes/s have been guaranteed to the service level and the internal clock is 100 MHz, the bandwidth counter must be decreased by one every 50 cycles. This decrease is implemented by configuring the "Allocated BW" register with the appropriate number of cycles. When the cycle counter matches the configured cycles, the bandwidth counter register is decreased by one, and the cycle counter is reset. All these operations effectively allow to measure the difference between reserved and consumed bandwidth, while they are simple and do not introduce a significant delay or require much silicon area.

3.1.1 Switch Scheduler

The switch scheduler implements a DRR algorithm based on the bandwidth counters' information. This algorithm consists in giving priority to flows that are consuming less bandwidth than the amount reserved for them. Following this scheduling, for each packet ready to cross the crossbar, the scheduler checks the value of the bandwidth counter corresponding to the output port and the service level of the packet. Afterwards, packets are served in the order of the value of these registers: first, packets with the smallest values and, later, packets with higher values.

It is possible to use a simplified version of this scheduler, in such a way that a threshold value is set, and the scheduler performs two rounds of scheduling: first, a round for packets with counter values lower than the threshold and, later, another round with the unassigned outputs and the rest of the packets. For our tests, we tuned the value of this threshold, obtaining an optimal value of 4 Kbytes.

Since we assume Virtual Cut-Through switching, our scheduling decisions are made for whole packets (*packet-mode* scheduling [14]). 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.

This scheduler also needs a new feature if compared to a typical one. Specifically, our scheduler needs to know whether a packet at an input port is going to be stored in an output port SAQ. This is known by comparing the turnpool of the packet with the information at the corresponding CAM. In the case of a match, packets can only be selected if the receiving SAQ is not filled over a given threshold. By applying this rule, buffer hogging is prevented.

Moreover, the scheduler does not treat specially packets coming from SAQs. However, if such packets are contributing to congestion, the bandwidth counter will have a high value, and they will be penalized by the scheduling algorithm. On the other hand, if packets are no longer contributing to congestion, the bandwidth counter will have a small value, and they will achieve a high priority. This promotes unnecessary SAQs to be emptied as soon as possible, allowing so the deallocation of these SAQs (SAQs must be empty for being deallocated).

3.1.2 Congestion Management

RECN detects congestion both at input and output ports of switches always by measuring the number of packets mapped at queues. Our new proposal also detects congestion at input and output ports, but in this case, these detections do not depend only on queue occupancy. Specifically, in the new proposal, input detections occur when a detection queue in an input port fills over (in terms of stored information) a certain threshold,[8] and the value of the bandwidth counter associated to the packet service level at the requested output port is over another threshold (bandwidth counter threshold, see previous section).

Note, however, that detection thresholds and bandwidth counter threshold are completely independent: detection thresholds determine how quickly the mechanism reacts against congestion, while the bandwidth counter one determines the mechanism tolerance regarding "oversubscribing" traffic classes.

The actions to take after a congestion detection at the input port are, first, an SAQ with turnpool equal to the output port and the appropriate service level is allocated in the input port and, in addition to this, another SAQ, with empty turnpool but associated to the service level, is allocated at the corresponding output port. These SAQs with empty turnpool (not considered in original RECN) will store any packet belonging to the associated service level.

6. The unit used in our counter is 64 bytes because in ASI, each credit is 64 bytes, and thus, packets come in 64-byte increments.

7. These values are adequate to monitor instantaneous traffic behavior.

8. Like in RECN, this threshold and the output detection one were tuned in our experiments until achieving optimal performance.

This is necessary in order to avoid that packets from a single service level completely fill an output buffer.

On the other hand, the conditions for a detection of congestion at an output port are similar. If a packet arrives from an input port and causes the occupancy of the standard queue at this output port to be over a certain threshold, and the bandwidth counter of the service level of the incoming packet is over another threshold, then the detection of congestion takes place. The actions after an output detection are the same as in an input detection: allocation of two SAQs, one at the input port the packet came from and another with empty turnpool (but associated to the service level) at the output port.

If congestion persists and SAQs start to fill over a certain threshold (known as the propagation level), then information about the corresponding turnpool and service level is propagated backwards the congestion flow in order to allocate new SAQs for storing packets belonging to the flow wherever these packets are. In the case of propagation from an SAQ at an output port, SAQs are allocated at the input ports that cause the overflow of the output port SAQ. Of course, these input SAQs will have an associated turnpool with one more hop than the output SAQ. SAQs with empty turnpools (only allocated at output ports) may also produce the allocation of SAQs at the input ports, which in this case will have a one-hop turnpool.

If SAQs at input ports fill over the propagation level, a control packet is sent to the preceding switch. This packet includes the turnpool and service level associated to the filled input SAQ, in order to also have an allocated SAQ associated to this information at the receiving output port.

In this way, there will be SAQs at any point where they are necessary in order to store congested packets, thereby eliminating HOL blocking. SAQs can be deallocated following the same conditions for SAQ deallocation considered in the RECN [7].

3.1.3 QoS Provision

In order to provide QoS guarantees, each traffic class has an assigned percentage of link bandwidth (or weight). For instance, if there are four traffic classes, each one could have 25 percent. Of course, total assigned bandwidth must not exceed the link bandwidth. At end nodes, we assume a traditional DRR implementation with a VC per traffic class, which is feasible in these devices.

Provided that end-nodes implement a QoS policy and as long as there is no contention, we have observed that packets pass through the switches in the same proportions as they are injected into the network. The reason is that switches do not introduce any significant delay when links are not oversubscribed.

However, since there is no admission control, it may happen that any link in the network becomes oversubscribed. In this situation, congestion appears because at this point, one or more traffic classes introduce more traffic than their assignation.

A traditional congestion management technique would penalize all traffic regardless of its traffic class. From this point of view, all packets are equally contributing to congestion. However, with the bandwidth counters we have proposed and the switch scheduler we have presented before, only traffic classes injecting more than their allowance are handled by the congestion management technique. In this case, this is done by storing "guilty" packets into SAQs. Note that any traffic class may still inject additional traffic if there is unused bandwidth. Thus, problems only arise as a consequence of oversubscribed links.

Therefore, QoS guarantees are achieved in the sense that if traffic from a class is injected up to its allowed bandwidth, it will achieve maximum throughput and experience short delay. The scheme proposed for QoS provision uses the same resources provided for congestion management. Therefore, we can offer a satisfactory solution for both problems, as we will confirm in the following sections.

TABLE 1
Switch Organization Comparison

Organization	HOL blck. eliminated?	QoS?	Queues per switch port
Basic	no	no	1
VOQ_{sw}	partial	no	s
VOQ_{net}	yes	no	n
RECN	yes	no	$s + SAQs$
Traditional VC	no	yes	t
New 2 VC	no	yes	2
VOQ_{sw} + QoS	partial	yes	$s \times t$
VOQ_{net} + QoS	yes	yes	$n \times t$
RECN + QoS	yes	yes	$s + SAQs$

Notes: s = ports per switch, n = end-nodes in the network, t = traffic classes, SAQs = number of SAQs implemented in RECN.

3.2 Switch Complexity

In this section, we compare the complexity of different switch organizations. We consider the solutions for HOL blocking elimination such as RECN and VOQ (at network level—VOQ_{net}—or at switch level—VOQ_{sw}—). We also consider those techniques that provide QoS support using VCs, including the "traditional" approach (1 VC per traffic class) and the recent proposal [10] using just two VCs. Finally, we also consider switch organizations able to provide both HOL blocking elimination and QoS support.

Table 1 shows some characteristics of the different switch organizations considered. As can be seen, only the last two switch architectures offer full QoS and HOL blocking elimination. Table 1 also shows the expressions for calculating the number of queues per switch port that are necessary to implement in each case. From these expressions, it is possible to obtain the exact number of queues by considering specific values for the number of port switches, network ports, traffic classes, and SAQs per port.

Moreover, we have calculated for each case offering HOL blocking elimination and QoS the number of required queues per port by assuming typical values of eight port switches, 128 end nodes in the network, eight traffic classes, and eight SAQs per port. Table 2 shows the resulting numbers, along with an estimation for the required area. In order to obtain this estimation, first, the minimum data memory per port has been calculated for each case as the minimum storage requirements for Virtual Cut-Through

TABLE 2
Queues and Data Memory Area Required at Each Port

Organization	Queues per port	Data Memory Area per port (mm^2)
VOQ_{sw} + QoS	64	15.00
VOQ_{net} + QoS	1024	240.00
RECN + QoS	16	3.75

Notes: ports per switch=8, end-nodes=128, traffic classes=8, SAQs = 8, minimum storage per queue= $2KB$.

switching (one packet per queue, allowing a maximum packet size of 2 Kbytes). Then, the area required in each case has been estimated from the calculated minimum data memory using the SRAM modeling tool CACTI [15] and assuming the use of two-port SRAM memories (0.133 μm technology) with a 64-bit organization.

Note that, for these assumptions, the $VOQ_{sw} + \text{QoS}$ architecture would need 64 queues, the $VOQ_{net} + \text{QoS}$ approach would need 1,024 queues, while our proposal presented in this paper would just need 16 queues per port. As a consequence, the estimated data memory area per port is much smaller for our proposal. Obviously, this would allow to produce simpler and cheaper switches.

Summing up, we can see clearly why a straightforward implementation of VOQ at the network level and the typical VC-based QoS support is a bad idea. An integrated solution such as the one we propose is more interesting, since the resources, queues in this case, are reused instead of multiplied.

4 Performance Evaluation

In the following, we evaluate the performance of the proposal presented in this paper. We have followed the methodology proposed in [16] using an ad-hoc simulator.

4.1 Simulation Model

In most cases, we have assumed a workload of four service levels. Each one has been assigned a guaranteed bandwidth, but for the sake of simplicity, in many cases, we assume that each traffic class has been guaranteed 25 percent of the link bandwidth.

We have considered different network topologies: multistage interconnection networks (MINs, the usual topology for computer clusters) with different sizes and also direct networks (3D meshes). However, our proposal is valid for any other network topology. Other assumptions, like link bandwidth or packet size, are based on the ASI specifications [9]. Another assumption is the use of source routing, since it is needed by RECN.

We have run simulations for several switch architectures. In all cases, switches have 16 ports. We have considered the following switch architectures:

- VOQ_{sw} **1 VC**. A simple switch design with VOQ at switch level but without VCs. In this case, the number of queues per port is 16.
- VOQ_{sw} **4 VC**. An architecture with one VC per traffic class, and each VC is further divided in VOQs at the switch level. The total number of queues per switch port is 64.
- VOQ_{net} **4 VC**. A switch design with classic VOQ at network level and also one VC per traffic class. This is a complex architecture, since it requires 256 queues per switch port.
- **RECN + QoS**. The switch architecture we have proposed in this paper. Since it integrates QoS and congestion management with the same resources, the number of queues per port is 24: 16 detection queues plus eight SAQs.

Regarding traffic scenarios, we have studied the following:

- *Hot-spot scenario*. We test the different alternatives with a static congestion traffic pattern.
- *Multimedia traffic scenario*. In this case, we introduce congestion dynamically. To achieve this, we transmit video sequences, which are very bursty.
- *Multimedia traffic scenario with priorities*. This scenario is similar to the previous one, but we vary the bandwidth assigned to each service level.

4.2 Simulation Results

In the following sections, we show results for several (four) tests. The first three tests correspond to the three different traffic scenarios considered, applied in a MIN with 64 end nodes, while in the last test, network topology and size are varied.

In each test, we show the advantages of the different architectures from two different points of view: congestion management and QoS support. However, validating RECN as a congestion management technique is out of the scope of this paper since it has been done in previous works [17], [18], [7]. Consequently, we just confirm that our proposed integrated architecture still offers the expected performance regarding congestion.

Regarding QoS results, we have already stated that the aim of this architecture is not to offer strict QoS guarantees, but a compromise from the network to offer good performance if traffic injection is inside the configured bounds. Moreover, since all traffic shares the same VC, some additional delays are unavoidable when the congestion detection takes place. For instance, in order to detect that a traffic class is injecting too much, a buffer must be filled up to a certain level, which implies that packets using this buffer will be delayed until the SAQs start storing congested packets. Therefore, latency results will not be as good as if there were a dedicated VC for regulated traffic. Nevertheless, this architecture has the advantage of not requiring connection admission control (CAC) at all.

4.2.1 Hot-Spot Scenario

The objective of this scenario is to show that our proposal is able to identify and isolate congested traffic, just as the original RECN technique. Traffic from service levels that are injecting less traffic than their reserved bandwidth should not be affected by this congestion. Moreover, congested traffic should get at least as much bandwidth as reserved.

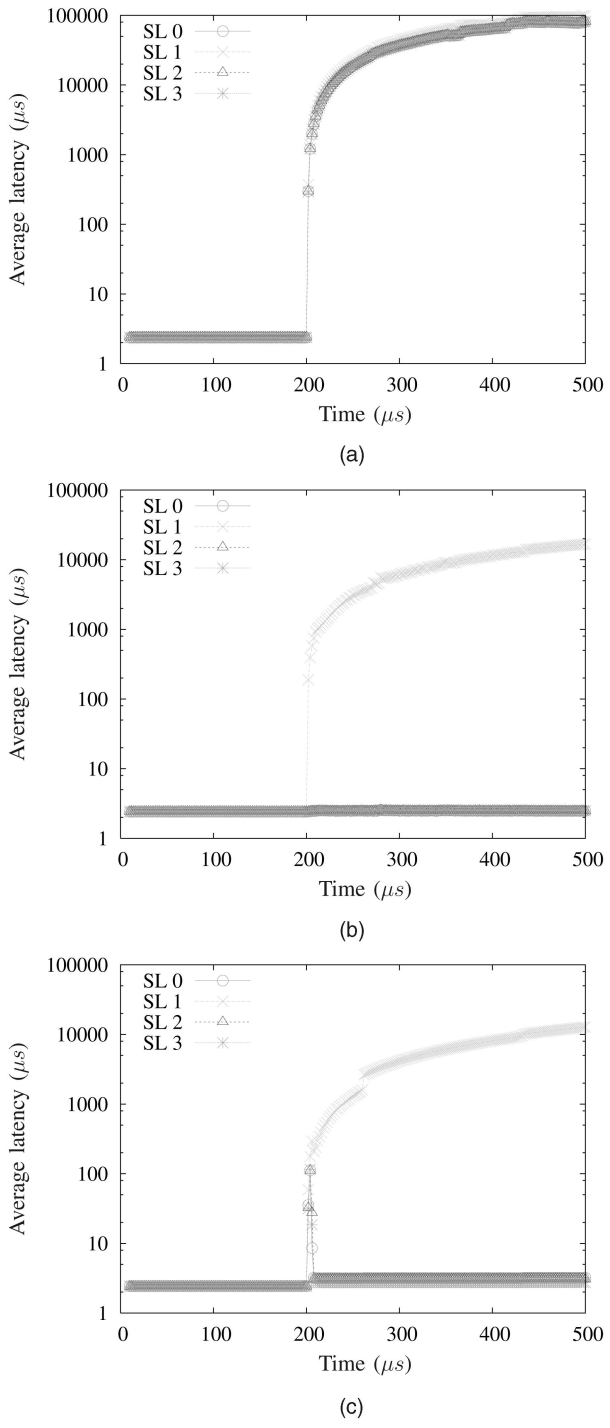

Fig. 7. Results for hot-spot scenario. (a) VOQ_{sw}. (b) VOQ_{net}. (c) RECN+QoS.

In this scenario, there is uniform traffic belonging to four service levels. However, during a small period of time, there is a sudden burst of traffic toward a hot spot coming from a single service level. Without loss of generality, we assume that the hot spot is located at end node 5 and service level 1. In this way, in addition to the uniform traffic, some of the interfaces inject traffic of service level 1 toward end node 5.

All the results of this scenario have been obtained for a MIN with 64 end nodes. In Fig. 7, latency results of the four aforementioned traffic classes can be seen. We show performance for three different architectures: classic VOQ at switch level and a single VC (Fig. 7a); classic VOQ at network level, with also one VC per traffic class (Fig. 7b); and our RECN+QoS proposal (Fig. 7c).

As can be observed, performance is very poor when using the VOQ_{sw} organization: all the traffic classes are affected by congestion. The VOQ_{net} architecture offers very good performance since all the traffic classes and destinations are completely isolated but is a very expensive (even infeasible) solution. When using our proposal, we can see that traffic classes that are not producing congestion are only slightly affected by the hot spot (specifically, for all these classes, latency slightly increases during a short interval), even though they share the same VC with the congested traffic.

From this test, we can conclude that our proposal is still able to provide several traffic classes with their assigned bandwidth, even if there is another traffic class flooding the network.

4.2.2 Multimedia Traffic Scenario

After studying the performance of the different architectures under static congestion, we look at results when congestion is dynamically generated. In real life, congestion rarely appears regarding a single hot spot, with the rest of the traffic behaving properly. Rather, hot spots appear and disappear quickly as bursts of packets are injected into the network. An excellent example of this is multimedia traffic, like video sequences.

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 intracoded frames, which are basically normal pictures compressed with an algorithm like JPEG; besides, there are intercoded frames, which only encode the differences with some neighbor frames. Therefore, frame size presents a lot of variability [19].

In this way, every 40 ms a long burst of packets is produced. Moreover, if many of these sources are multiplexed, they can quickly deplete buffers in the network and lead to a very poor performance.

In this scenario, all traffic comes from video traces of MPEG-4 sequences. Each sequence has an average throughput of 750 Kbytes/s, but bursts can be as large as 300 Kbytes. There are four traffic classes, each one with a guaranteed 25 percent of the throughput.

Again, the results shown in this case have been obtained for a MIN with 64 end nodes. For this scenario, we can see in Fig. 8 the performance of the different switch architectures we are considering.

Specifically, Fig. 8c shows the overall throughput results. We can see that our proposal offers a performance identical to the ideal architecture (VOQ_{net}). Regarding the VOQ_{sw} architectures, performance is very bad, with a throughput peak near 65 percent. Note that this is even the case of the four VC architecture, which has many more queues than our proposal.

In Figs. 8a and 8b, we show overall latency results, both in linear and logarithmic scale at the y-axis. In this case, the best results are for our RECN+QoS architecture, even better

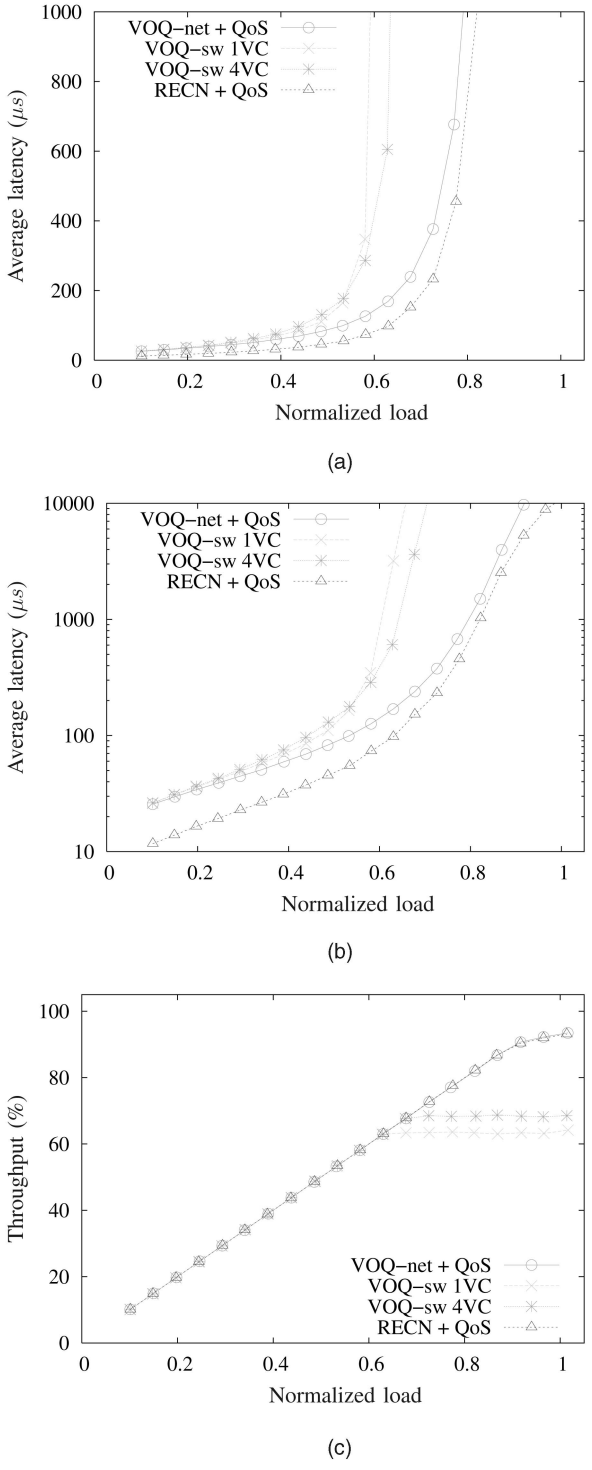

Fig. 8. Performance in video transmission scenario. (a) Global latency. (b) Global latency, logarithmic scale. (c) Global throughput.

than the VOQ_{net} case. The reason is that our RECN+QoS proposal is able to cope better with congestion at any point in the network, while VOQ_{net} is designed to handle congested end nodes. As a consequence, large bursts of packets (like big video frames) creating congestion inside the network progress faster when our proposal is used, thereby obtaining the best frame-level latency results.

We can conclude that our proposal is ready to cope with very demanding traffic, like multimedia traffic.

TABLE 3
Scheduler Configuration

Traffic Class	Bandwidth	Percentage	Table Entries
Class 1	3 Gbit/s	37.5%	24
Class 2	2.5 Gbit/s	31.25%	20
Class 3	1.5 Gbit/s	18.75%	12
Class 4	1 Gbit/s	12.5%	8
Total	8 Gbit/s	100.0%	64

4.2.3 Multimedia Traffic Scenario with Priorities

Up to this point, we have shown that our proposal handles congestion properly. Moreover, it is able to isolate one traffic class from the others, in such a way that it can get a guaranteed minimum throughput at every link. In this scenario, we vary the weights of the traffic classes (see Section 3.1.3). This can be done in our switch model just by configuring the weights but also in "traditional" switches by assigning different number of arbitration table entries. As can be seen in Table 3, we use four different bandwidth assignments to each (four) traffic classes.

In this test, we also inject multimedia traffic, just like in the previous test. In this way, in addition to varying priority, we also have a very bursty traffic to be handled by the switches. Despite the bandwidth assignments, exactly the same amount of traffic is injected from every class (25 percent). In this way, we should see that classes 4 and 3 are the first ones to get congested, while class 1 should not be affected.

The performance results for this scenario of the different switch architectures are shown in Fig. 9 (in all the cases, network is a MIN with 64 end nodes). In the left column, latency results for the different classes are shown, while in the right column, the throughput ones are shown. Once again, our proposal offers performance identical to ideal architecture regarding throughput. We can also appreciate the differences between the four traffic classes. We can see that classes 1 and 2 obtain 100 percent throughput, while classes 3 and 4 congest at lower loads. Note that all these traffic classes share a unique VC.

Regarding VOQ_{sw}, the traffic classes obtain very poor performance, similar to that of the previous test. Note that even the VOQ_{sw} 1VC scheme obtains some differentiation between the four traffic classes. This is because, in all the cases, we are considering ideal network interfaces at end nodes, with as many queues as traffic classes times network destinations.

Finally, latency results are as expected (from throughput results). Note that, once again, the best latency performance corresponds to our proposed architecture.

4.2.4 Direct Networks and Scalability

In this section, we explore the performance of the considered architectures in 3D meshes and MINs with different network sizes. This will give us an estimation about how our proposal scales and adapts to other environments. The results shown in this section have been obtained considering the multimedia traffic (without priorities) scenario.

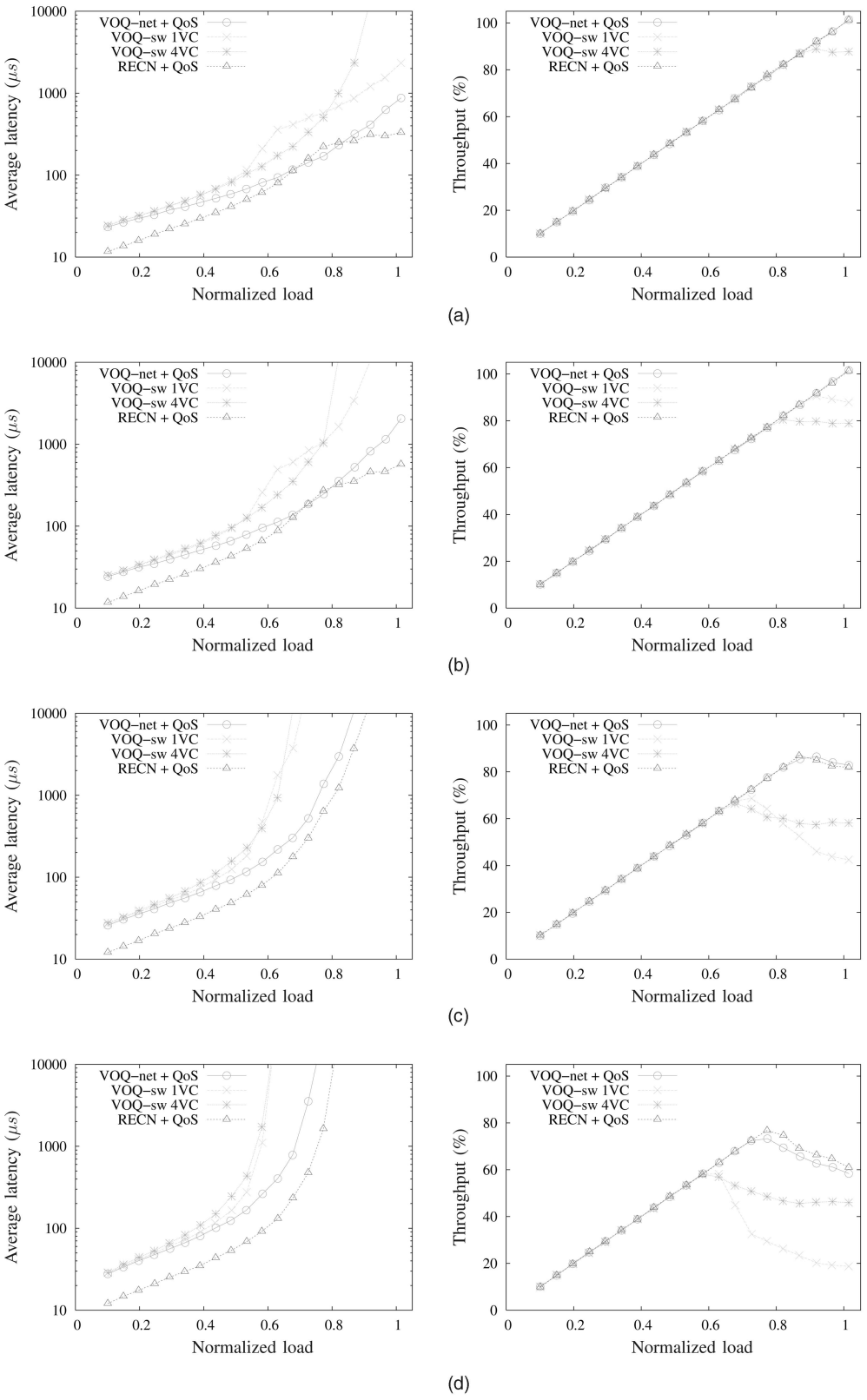

Fig. 9. Results for multimedia traffic with different weights. (a) Class 1. (b) Class 2. (c) Class 3. (d) Class 4.

First, Fig. 10 shows the normalized overall network throughput achieved for the different mechanisms in different MIN sizes (64, 128, 256, and 512 end nodes). In all the cases, our proposed solution obtains better performance than the rest. Note that, for all the architectures, there is a slight performance degradation with network size. This is because multimedia traffic is very bursty, as discussed in former sections.

Second, Fig. 11 shows the performance achieved for the different mechanisms in a 3D mesh network with 64 nodes.

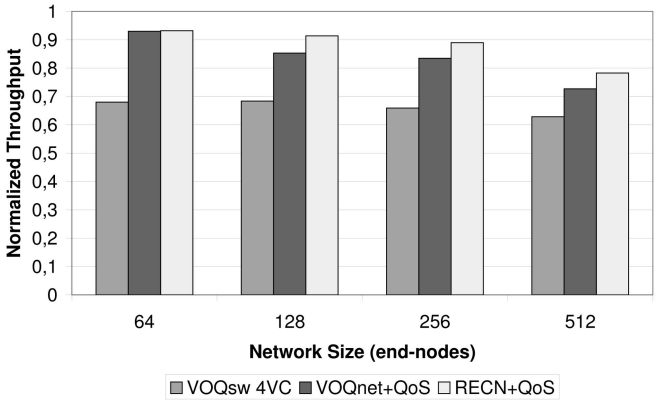

Fig. 10. Performance for different MIN network sizes.

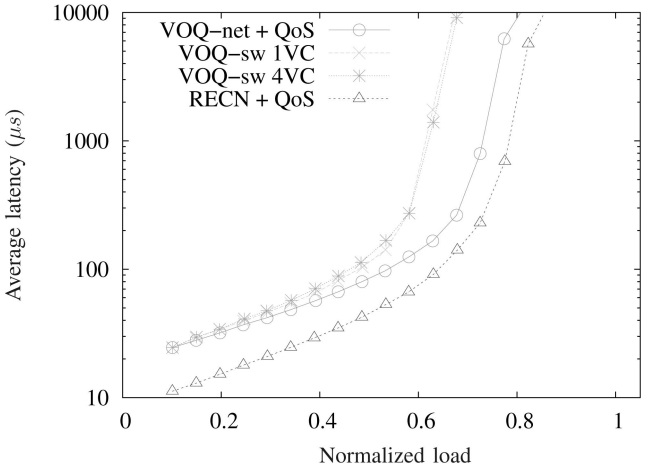

Fig. 11. Performance in video transmission scenario in a 3D mesh with 64 end nodes.

Specifically, overall latency results are shown (please note logarithmic scale). As can be seen, results obtained are very similar to the ones achieved for the MIN case with 64 nodes (see Fig. 8a), thus we can conclude that our proposal is also valid for direct networks.

5 Conclusions

Due to cost and power consumption constraints, current high-speed interconnection networks cannot be over-dimensioned. Therefore, some solutions are needed in order to handle the problems related to high link utilization. In particular, both QoS support and congestion management techniques have become essential for achieving good network performance. However, most of the techniques proposed for both issues require too many resources.

In this paper, we propose a new switch architecture able to face the challenges of congestion management and, at the same time, QoS provision, while being more cost-effective than other proposals. Our proposal is based on the combination of two novel techniques that provide HOL blocking elimination (turning congestion harmless) and QoS support while requiring a reduced number of resources.

According to the results presented in this paper, we can conclude that our proposal provides an adequate QoS level while properly dealing with congestion. Since this is achieved with a reduced number of resources, this architecture would also reduce network cost.

Specifically, we have shown that our proposal obtains excellent results in both QoS provision and HOL blocking elimination for different traffic scenarios and for different topologies and network sizes. Regarding QoS, we have seen that our proposal can efficiently isolate the traffic of each service level, in such a way that each one can obtain its guaranteed throughput, regardless of the behavior of the other service levels. Moreover, latency results are better when using our proposal than when using a much more complex (and expensive) VOQ_{net} architecture. Regarding HOL blocking elimination, the performance was excellent even when we injected static intensive hot spots.

Summing up, this outstanding performance was achieved with a cost-effective architecture, which integrates the QoS and HOL blocking elimination mechanisms in an efficient way.

Acknowledgments

This work has been jointly supported by the Spanish MEC and European Commission FEDER funds under Grants Consolider Ingenio-2010 CSD2006-00046 and TIN2006-15516-C04-0X and by Junta de Comunidades de Castilla-La Mancha under Grant PBC08-0078-9856.

References

[1] M.J. Karol, M.G. Hluchyj, and S.P. Morgan, "Input versus Output Queueing on a Space-Division Packet Switch," *IEEE Trans. Comm.*, vol. 35, pp. 1347-1356, 1987.

[2] *InfiniBand Architecture Specification Volume 1, Release 1.0,* InfiniBand Trade Assoc., Oct. 2000.

[3] *QsNet Overview,* white paper, Quadrics Ltd., http://www.quadrics.com, 2005.

[4] *Myrinet,* Myricom Inc., http://www.myrinet.com, 2005.

[5] W.J. Dally, P. Carvey, and L. Dennison, "Architecture of the Avici Terabit Switch/Router," *Proc. Sixth Symp. High-Performance Interconnects (Hot Interconnects '98),* pp. 41-50, 1998.

[6] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High-Speed Switch Scheduling for Local-Area Networks," *ACM Trans. Computer Systems,* vol. 11, no. 4, pp. 319-352, Nov. 1993.

[7] P.J. García, J. Flich, J. Duato, I. Johnson, F.J. Quiles, and F. Naven, "Efficient, Scalable Congestion Management for Interconnection Networks," *IEEE Micro,* vol. 26, no. 5, pp. 52-66, Sept. 2006.

[8] J. Duato, S. Yalamanchili, M.B. Caminero, D. Love, and F.J. Quiles, "MMR: A High-Performance Multimedia Router. Architecture and Design Trade-Offs," *Proc. 11th Symp. High Performance Computer Architecture (HPCA '99),* Jan. 1999.

[9] *Advanced Switching Core Architecture Specification, Revision 1.1,* Advanced Switching Interconnect Special Interest Group, Mar. 2005.

[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," *Proc. 12th Int'l Conf. High Performance Computing (HiPC '05),* pp. 158-169, http://investigacion.uclm.es/portali/documentos/it_1131561750-HiPC05.pdf, Dec. 2005.

[11] A. Martínez, P.J. García, F.J. Alfaro, J. Flich, J.L. Sánchez, F.J. Quiles, and J. Duato, "A Cost-Effective Interconnection Architecture with QoS and Congestion Management Support," *Proc. European Conf. Parallel Computing (EuroPar '06),* Aug. 2006.

[12] J. Duato, S. Yalamanchili, and N. Lionel, *Interconnection Networks: An Engineering Approach.* Morgan Kaufmann, 2002.

[13] W.J. Dally and B. Towles, *Principles and Practices of Interconnection Networks.* Morgan Kaufmann, 2003.

[14] M.A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Packet-Mode Scheduling in Input-Queued Cell-Based Switches," *IEEE/ACM Trans. Networking*, vol. 10, no. 5, pp. 666-678, 2002.
[15] P. Shivakumar and N.P. Jouppi, "CACTI 3.0: An Integrated Cache Timing, Power, and Area Model," technical report, Compaq Western Research Laboratory, 2001.
[16] I. Elhanany, D. Chiou, V. Tabatabaee, R. Noro, and A. Poursepanj, "The Network Processing Forum Switch Fabric Benchmark Specifications: An Overview," *IEEE Network*, pp. 5-9, Mar. 2005.
[17] J. Duato, I. Johnson, J. Flich, F. Naven, P.J. García, and T. Nachiondo, "A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks," *Proc. 11th Symp. High Performance Computer Architecture (HPCA)*, 2005.
[18] P.J. García, J. Flich, J. Duato, I. Johnson, F.J. Quiles, and F. Naven, "Dynamic Evolution of Congestion Trees: Analysis and Impact on Switch Architecture," *Proc. Int'l Conf. High Performance Embedded Architectures and Compilers (HiPEAC '05)*, pp. 266-285, Nov. 2005.
[19] *Generic Coding of Moving Pictures and Associated Audio*, Moving Picture Experts Group, Rec. H.262., Draft Int'l Standard ISO/IEC 13818-2, 1994.

Alejandro Martínez received the MS degree in computer science and the PhD degree from the University of Castilla-La Mancha in 2003 and 2007, respectively. He is currently working at Intel Barcelona Research Center. His research interests include high-performance interconnections, quality of service, high-performance computing, and processor microarchitecture.

Pedro J. García received a degree in communication engineering from the Technical University of Valencia, Spain, in 1996, and the PhD degree in computer science from the University of Castilla-La Mancha, Spain, in 2006. In 1999, he joined the Computer Systems Department (DSI), University of Castilla-La Mancha, Spain, where he is currently an assistant professor of computer architecture and technology. His research interests are focused on high-performance interconnection networks, mainly congestion management and deadlock-avoidance techniques and also reconfiguration and routing algorithms.

Francisco J. Alfaro received the MS degree in computer science from the University of Murcia in 1995 and the PhD degree from the University of Castilla-La Mancha in 2003. He is currently an assistant professor of computer architecture and technology in the Computer Systems Department, Castilla-La Mancha University. His research interests include high-performance local area networks, QoS, design of high-performance routers, and design of on-chip interconnection networks for multicore systems.

José L. Sánchez received the PhD degree from the Technical University of Valencia, Spain, in 1998. Since November 1986, he has been a member of the Computer Systems Department (formerly Computer Science Department), University of Castilla-La Mancha. He is currently an associate professor of computer architecture and technology. His research interests include multicomputer systems, quality of service in high-speed networks, interconnection networks, and parallel algorithms and simulation.

José Flich received the MS and PhD degrees in computer science from the Technical University of Valencia (Universidad Politécnica de Valencia), Spain, in 1994 and 2001, respectively. He joined the Department of Computer Engineering (DISCA), Universidad Politécnica de Valencia, in 1998, where he is currently an associate professor of computer architecture and technology. His research interests are related to high-performance interconnection networks for multiprocessor systems, cluster of workstations, and networks on chip. He has served as a program committee member in different conferences, including ICPP, IPDPS, HiPC, CAC, ICPADS, and ISCC. He is currently the cochair of the CAC and INA-OCMC workshops and the vicechair (high-performance networks track) of the EuroPar conference.

Francisco J. Quiles received the degree in physics (electronics and computer science) and the PhD degree from the University of Valencia, Spain, in 1986 and 1993, respectively. In 1986, he joined the Computer Systems Department, University of Castilla-La Mancha, where he is currently a full professor of computer architecture and technology and the vicerector of research at the University of Castilla-La Mancha. He has developed several courses on computer organization and computer architecture. His research interests include high-performance networks, parallel algorithms for video compression and video transmission, and DVC. He has published more than 160 papers in international journals and conference proceedings.

José Duato is a professor in the Department of Computer Engineering (DISCA), Polytechnic University of Valencia, Spain. His research interests include interconnection networks and multiprocessor architectures. He has published more than 350 papers. His research results have been used in the design of the Alpha 21364 microprocessor and the Cray T3E and IBM BlueGene/L supercomputers. He is the first author of the book "Interconnection Networks: An Engineering Approach." He served as associate editor for the *IEEE Transactions on Parallel and Distributed Systems* and *IEEE Transactions on Computers* and is serving as an associate editor for the *IEEE Computer Architecture Letters*. He was the general cochair of ICPP 2001, the program chair of HPCA 2004, and the program cochair of ICPP 2005. Also, he served as the cochair, steering committee member, vicechair, or program committee member in more than 55 conferences, including HPCA, ISCA, IPPS/SPDP, IPDPS, ICPP, ICDCS, Europar, and HiPC.