Ferrer Pérez, JL.; Baydal Cardona, ME.; Robles Martínez, A.; López Rodríguez, PJ.; Duato Marín, JF. (2012). Progressive congestion management based on packet marking and validation techniques. IEEE Transactions on Computers. 61(9):1296-1309. doi:10.1109/TC.2011.146.

Progressive Congestion Management Based on Packet Marking and Validation Techniques

Joan-Lluís Ferrer, Elvira Baydal, Antonio Robles, *Member*, *IEEE Computer Society*, Pedro López, *Member*, *IEEE Computer Society*, and José Duato

Abstract—Congestion management in multistage interconnection networks is a serious problem, which is not solved completely. In order to avoid the degradation of network performance when congestion appears, several congestion management mechanisms have been proposed. Most of these mechanisms are based on explicit congestion notification. For this purpose, switches detect congestion and depending on the applied strategy, packets are marked to warn the source hosts. In response, source hosts apply some corrective actions to adjust their packet injection rate. Although these proposals seem quite effective, they either exhibit some drawbacks or are partial solutions. Some of them introduce some penalties over the flows not responsible for congestion, whereas others can cope only with congestion situations that last for a short time. In this paper, we present an overview of the different strategies to detect and correct congestion in multistage interconnection networks, and propose a new mechanism referred to as Marking and Validation Congestion Management (MVCM), targeted to this kind of lossless networks, and based on a more refined packet marking strategy combined with a fair set of corrective actions, that makes the mechanism able to effectively manage congestion regardless of the congestion degree. Evaluation results show the effectiveness and robustness of the proposed mechanism.

Index Terms—Interconnection networks, congestion management, message throttling.

1 Introduction

Computing systems have experimented a dramatic growth due to the needs of new communication-intensive applications and the increasing demand of new services. Clusters of PC offer nowadays the best cost-performance ratio to build either low-cost supercomputers or high-performance servers. In these systems, computing nodes are interconnected with a high-performance interconnection network that uses point-to-point links and high speed switches. Among all the feasible interconnection topologies, Multistage Interconnection Networks (MINs) have become very popular because they are the recommended choice of several standard (InfiniBand, RapidIO) and nonstandard (Myrinet, Quadrics) interconnect technologies. Moreover, MINs are able to provide a high network throughput and take advantage of the availability of high-radix commercial switches.

As interconnects are expensive compared to processors, an easy solution to cut down costs is to reduce the number of components, increasing their usage (i.e., switches and links). On the other hand, to reduce power consumption some frequency/voltage scaling techniques [2], [17] are usually applied. Both factors, cost reduction and power saving, lead to a higher network utilization. In this situation, network congestion may arise strongly degrading network performance.

Network congestion appears when there is contention between several packets trying to use the same output link. If this situation remains for long, packets start to accumulate at the queues of the affected switches. As a consequence of the back pressure caused by the flow control mechanism, the packet advance in the previous switches is also delayed, generating the Head-Of-Line (HOL) blocking phenomenon, which prevents the advance of packets addressed to noncongested links. Note that in high-performance interconnects for clusters, the communication model assumes a lossless network, so packets cannot be dropped to deal with congestion, as happens in other interconnect environments.

If this situation spreads along the network, forming what it is known as a saturation tree, traffic in the switches will be blocked, causing the degradation of the overall network performance. Notice that, to avoid spreading the saturation tree along the network, it would be enough to stop at their origin hosts those packets responsible for the congestion situation. This fact would contribute to improve the overall network throughput. Therefore, it is mandatory to use an effective Congestion Management Mechanism (CMM) able to detect congestion early, and to apply efficient corrective actions to avoid HOL blocking and performance degradation. Notice that as MINs are lossless networks, packet loss cannot be used as an index to detect congestion. Therefore, CMMs, as the one used in TCP, cannot be applied in this kind of environments.

Recently, some CMMs have been proposed to detect and manage congestion. Unfortunately, these approaches do not guarantee, for all traffic distributions, that corrective actions

• *The authors are with the Department of Computer Engineering (DISCA), Universitat Politécnica de València Camino de Vera 14, València 46022, Spain. E-mail: jlferrer@gap.upv.es, {elvira, arobles, plopez, jduato}@disca.upv.es.*

Manuscript received 30 Sept. 2010; revised 9 July 2011; accepted 16 July 2011; published online 22 July 2011.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2010-09-0542.
Digital Object Identifier no. 10.1109/TC.2011.146.

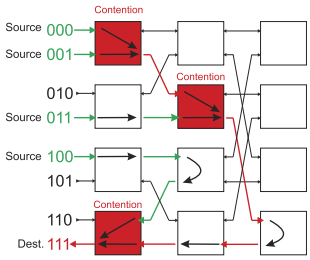

Fig. 1. A butterfly 2-ary 3-fly network.

are only carried out on those packet flows[1] causing congestion and with the appropriate intensity. As a consequence, they can penalize unnecessarily the network traffic and, therefore, degrade the overall network performance.

In this paper, we propose and evaluate in depth, a new cost-effective congestion management strategy for lossless MINs. Fig. 1 shows an example of a MIN (*k-ary n-fly*). Parameters k (radix) and n (number of stages) define the network structure and its depth. In particular, Fig. 1 shows a 2-ary 3-fly network with bidirectional links and butterfly interconnection pattern. This network is built with three stages of switches with a radix value of 2.

The proposed mechanism is able to accurately identify the flows responsible for congestion and to warn the corresponding origin hosts in different ways depending on the congestion level detected in the network, in order to stop or delay injecting packets. As a consequence, the resulting CMM is able to give a quick response in accordance to the congestion degree.

The rest of the paper is organized as follows: Section 2 presents a background of the general strategies applied to manage congestion. In Section 3, the proposed CMMs for MINs are described and analyzed. The new mechanism referred to as Marking and Validation Congestion Management (MVCM) is described in Section 4. The simulation scenario and the evaluation results are presented in Section 5. Finally, in Section 6 some conclusions are drawn.

2 Background

Congestion management in lossless networks has been widely studied over the years [14], generating a lot of research and proposals.

There are mainly two strategies to solve the problem: congestion prevention and congestion recovery. Congestion prevention requires to know in advance the network resources needed, reserving them before starting packet transmission. This strategy implies some overhead and is commonly used in protocols aimed at providing Quality of Service (QoS). On the other hand, mechanisms based on congestion recovery are able to detect and to solve the congestion on the fly, at the moment it arises. This strategy is usually based on three steps: detection, notification, and correction.

The congestion detection is frequently carried out by measuring the switch buffer occupancy [5], [7], [11], [15], [16], [19]. In some of them [7], [11], [15], [16], [19], if the occupancy at any buffer exceeds a predefined threshold, then packets crossing the switch will be marked. In this way, switches are

1. A flow identifies the traffic generated between an origin-destination pair of hosts.

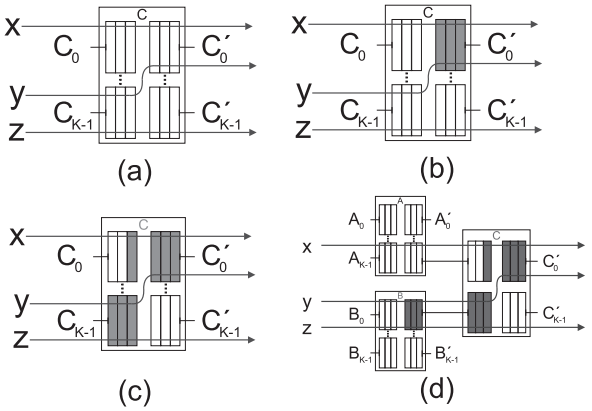

Fig. 2. Evolution of the congestion process.

able to detect and mark packets which are contributing to congestion. On the contrary, QCN [5] uses a more complex mechanism at the switches to detect congestion (the derivative of queue occupancy), but it does not use any packet marking strategy to warn the origin hosts.

After detecting congestion, the source hosts that are injecting too much traffic have to be warned in order to reduce evenly their injection rate. For this purpose, two techniques can be applied. The first one can be carried out by means of broadcast messages [19], [20]. This solution does not guarantee that notified sources are only those that are injecting traffic to the congested links. Therefore, hosts nonresponsible for congestion could reduce their injection rate and cause a decrement in network performance. Moreover, broadcasting control packets wastes network bandwidth, thus penalizing network throughput even more. On the other hand, the second technique, commonly known as Explicit Congestion Notification (ECN), has two basic strategies. The first one takes advantage of the acknowledgment (ACK packets) sent back to the source when a packet has reached its destination. In this technique [15], [16], ACK packets carry out the congestion information to the origins. In the second one [5], switches send specific control packets to warn the origin hosts. In any case, once the origin hosts are warned (with ACK or specific packets), they will apply message throttling to adjust its packet injection rate.

Other mechanisms try to solve the problem by temporarily separating the flows responsible for congestion from the flows that are not responsible of it in order to remove the HOL blocking effect [6]. To do so, it is necessary to incorporate a set of additional buffers. When the mechanism detects packets stopping the normal advance of packets into the network, the additional buffers will harbor those packets causing the HOL blocking phenomenon. Later, these packets may continue their trip to their destinations through some "slow roads" or, on the contrary, be reinstated again if congestion disappears.

Moreover, there are some proposals that try to reduce HOL blocking either at the switch level [3], [13], [18] or at the network level [4], [12], but do not actually solve it.

Next, for the sake of a more comprehensible insight into the different techniques, we expose the process of creating a congestion tree and how it spreads along the network from the root to the leaves.

Fig. 2 shows the process of creating a congestion tree. In Fig. 2a, an initial situation for three different flows (x, y,

and z) crossing the switch C is shown. Packets belonging to the x flow, cross the switch from the input channel C_0 to the output channel $C_0^{'}$. The y flow also advances from C_{k-1} toward $C_0^{'}$, and, finally, the z flow from C_{k-1} toward $C_{k-1}^{'}$. Notice that, in order to prevent switches from becoming a bottleneck, the internal bandwidth of the switch crossbar is usually higher than the channel bandwidth. As this bandwidth speedup increases switch complexity, often a maximum speedup of two is used [8].

Now, let us assume that the overall input traffic rate for $x + y$ flows is greater than the bandwidth of the output channel $C_0^{'}$. In this situation, packets belonging to x and y flows will have to compete for the output channel $C_0^{'}$ and, as a result of that, the output buffer will start to accumulate packets as Fig. 2b shows. Notice that, with a speedup equal to 1, packets would start to accumulate at their input buffers instead of at their output buffers. If this situation remains for long, and x, y, and z flows continue to inject traffic into the switch, packets may begin to accumulate at the input buffers, spreading the congestion along them. Fig. 2c shows a possible congestion situation, and due to the fact that incoming packets are stopped at the input buffer of the channel C_{k-1}, the HOL blocking phenomenon appears. Then, the advance of packets belonging to the z flow and addressed toward the noncongested channel $C_{k-1}^{'}$ could be delayed, causing the degradation of the switch performance. If this situation persists, congestion will be spread along the previous switches, stopping and accumulating packets at the output buffers of those switches. This situation provokes a new congested output channel at the previous switch. Fig. 2d shows how the congestion has been spread along the congestion tree toward its leaves, affecting several switches.

Notice that, when a congestion situation is created because of a set of origin hosts inject packets toward a hot-spot, as shown in Fig. 1, these flows do not only compete for the output channel at the last stage, but also at intermediate stages if some flows have to cross the same output channel. So, congestion may start at any stage.

3 Related Work

In this paper, we focus on CMMs proposed for MINs and based on ECN. Basically, current proposals define a packet marking technique combined with a packet injection limitation scheme that try to detect and palliate the effects of a congestion situation. In this section, first we show a brief analysis of the different packet marking techniques and the injection limitation strategies proposed until now, and second, we describe the current mechanisms already proposed.

3.1 Basic Strategies Applied by Current CMMs

1. The Packet Marking Techniques vary depending on the place where packets are marked. Basically, packets in transit are marked by setting 1 bit in the packet header. To this end, a threshold at input or output buffer is predefined in order to detect and mark packets provoking congestion.

If the applied strategy is based on marking packets at input buffers, we will refer to it as Input Packet Marking strategy (IPM). For this strategy, in the congestion process analyzed in Fig. 2, the congestion tree has to grow at least until reaching the input buffers placed at the same switch where congestion starts (as it is shown in Fig. 2c) before the packets stopped are detected and marked. This strategy produces a delay, because an output buffer has to be completely filled before any input buffer can detect congestion. When the congestion situation shown in Fig. 2c is reached, any incoming packet belonging to flows y and z will be marked regardless of its destination, that is, without considering if their output is the congested link $C_0^{'}$ or the noncongested link $C_{k-1}^{'}$.

Therefore, when applying the IPM strategy, two drawbacks appear. First, a delay in detecting congestion, and second an incorrect identification of the flows truly responsible for congestion.

The other packet marking strategy is based on marking packets at output buffers. We will refer to it as Output Packet Marking strategy (OPM). In this case, following the same example shown in Fig. 2b, all the packets belonging to the flows x and y, and addressed to the congested link $C_0^{'}$, will be marked when the output buffer occupancy exceeds the threshold. Notice that the contention in this switch has been provoked by the flows trying to cross the same output link toward the next switch (x and y flows), so all the packets belonging to the flows sharing the congested output link are responsible for this congestion regardless of their final destination. Hence, they will be marked in order to evenly reduce the injection rate at origins.

Due to the fact that this strategy only pay attention to the status of the output buffers, it also has a problem. On one hand, if contention has not been spread to the input buffers yet, as shown in Fig. 2b, the corrective actions that could be applied to the flows x and y because of marking their packets may not be necessary yet. So, although the OPM strategy detects contention sooner than IPM does, it is not able to detect whether the congestion is affecting other flows nonresponsible for the initial congestion or not. Therefore, marking packets and applying corrective actions prematurely could cause the reduction of the network performance. On the contrary, if contention is not reduced at this output link in time, and congestion spreads along the input buffers affecting other nonresponsible flows, as happens with z flow in Fig. 2c, the OPM strategy will not be able to mark packets belonging to those flows till the congestion reaches the output buffer at the previous switch, as shown in Fig. 2d. So, that delay in marking packets could affect in its turn other flows or even other switches not involved in the initial congestion. Additionally, given that both IPM and OPM strategies dedicate only 1 bit to mark packets in transit, they are not able to handle different levels of congestion and, as a consequence, to apply different corrective actions depending on the severity of the congestion.

2. The Corrective Actions will be applied when origin hosts receive an ACK packet warning about congestion. If this situation occurs, CMMs can apply a combination of the following basic techniques to evenly reduce the injection rate. Among the techniques commonly applied by the current corrective action schemes, we can refer to *window-based techniques* and *waiting interval insertion techniques*.

The *window-based technique* can be based either on a dynamic or static window. In particular, this strategy is

mainly used by TCP, which applies a dynamic window, but other techniques use a static one. Basically, this technique defines a window size to limit the maximum number of outstanding packets[2] per flow. The value of the window size depends on vendor criteria and it is kept fixed or can vary depending on network behavior.

The *waiting interval insertion technique* allows to reduce the injection rate in a progressive way by injecting Waiting Slots (WS) between two consecutive packets. Depending on the severity of congestion, the elapsed time between the injections of two consecutive packets will be increased or decreased. As long as the congestion vanishes, the waiting interval will be decreased until disappearing.

Basically, the technique works as follows: when an origin host receives a marked ACK packet, it waits a WS before injecting a new packet into the network. If more marked ACK packets are received, then more WS will be inserted enlarging the waiting interval. Notice that the injection of new packets for the same flow is forbidden along the waiting interval. Later, when unmarked ACK packets are received, the number of WS between packet injections will be progressively decreased.

Additionally, a timer can be defined to timeout if no more ACK packets arrive, and then, the waiting interval will be reduced or even reset.

3.2 Current Proposals for MINs

Recently, some CMMs based on ECN have been proposed to solve the congestion problem by using some of the basic strategies described in Section 3.1. Unfortunately, these approaches do not guarantee that either both packet marking and corrective actions are only carried out on flows responsible for congestion or they are not properly tuned. In this section, we describe the current CMMs based on the packet marking and the injection limitation techniques shown in the previous section. In particular, the Renato's proposal and the Pfister's implementation, both of them intended for InfiniBand [11].

3.2.1 Renato's Proposal

This CMM [16] applies an IPM strategy for packet marking and therefore to warn about a congestion situation. Switches need to have buffers at the input links in order to apply this strategy. In particular, the proposed IPM strategy operates in three steps. First, a switch input buffer triggers packet marking each time it becomes full. Second, any output link that is requested for at least one packet in such a full buffer, is classified as a congested link. Third, all packets stored at any input buffer at the switch that are destined to a congested output link will be marked.

In response to the reception of a marked packet, the origin hosts apply injection limitation based on a window-based technique combined with a waiting interval insertion technique. They proposed a static window size of one packet at any situation because a larger value completely saturates the network. In addition, an injection rate control based on inserting WSs is used. For rate control, they evaluate different functions to adjust the injection rate: Additive Increase Multiplicative Decrease (AIMD), Fast Increase Multiplicative Decrease (FIMD), and Linear Inter-Packet Delay (LIPD). As a result of their study, they propose two novel source response functions FIMD and LIPD for dynamic and static scenarios, respectively. In particular, the FIMD reduction function divides the injection rate by a constant of value $(m = 2)$ also using this value to increase the injection rate by using an exponential function. On the other hand, the LIPD response function applies a reduction based on the Inter-Packet Delay (IPD) design feature available in InfiniBand. Basically, it increases the IPD by one packet transmission time[3] each time a marked ACK arrive to origin hosts.

3.2.2 Pfister's Implementation

InfiniBand specs v1.2.1 [11] define a quite general proposal for congestion management where values for thresholds and other variables are left free to the vendor criteria. Pfister's implementation [15] is targeted to this scenario.

This CMM applies an OPM strategy for packet marking. Switches need to have output queues in order to apply this strategy. In particular, the defined threshold at output queues can be initialized with different values depending on the maximum level of congestion that can be tolerated in the network. In particular, a value between 0 and 15 could be programmed; 0 indicates that the switch is not going to mark any packet on this port, 1 indicates a high value of the threshold, so there is a high probability that congestion spreads, 15 indicates a low value of the threshold, so high probability that packets are marked, and values between 2 and 14 indicate a uniform distribution of decreasing threshold values. The exact meaning of a particular threshold setting is left to the switch manufacturer.

This proposal does not use any window-based technique to manage congestion. However, after receiving marked packets at origins host, sources reduce its injection rate by inserting WSs. In particular, the WS size applied is based on the Inter-Packet Delay (IPD) value. The amount of reduction is controlled using a table called Congestion Control Table (CCT). Each time a marked packet is received, an index into the table is incremented by a constant value depending on the Hotspot Degree (HSD) that indicates the number of sources contributing to the hotspot traffic. Additionally, each Host Channel Adapter (HCA) contains a timer that timeouts if no more marked packets arrive during a defined interval, in whose case, the index into the CCT is reduced.

4 Marking and Validation Congestion Management Mechanism

The Marking and Validation Congestion Management (MVCM) mechanism proposed in this paper consists of an end-to-end CMM based on the use of ECN. Unlike other approaches [15], [16], this new mechanism is not proposed for any standard interconnect in particular, but for MINs in general. However, the proposed mechanism could easily be applied to current standard interconnects, such as InfiniBand [11]. The main goal of this new CMM is to properly identify the flows responsible for congestion, in order to apply packet injection limitation only at the source nodes that are actually

2. Outstanding packets are those sent packets that have not been acknowledged yet.

3. The packet transmission time is the minimum time needed to send a packet plus to receive its ACK.

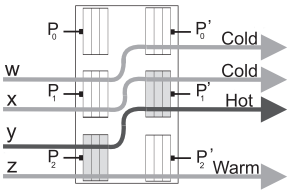

Fig. 3. Flows classification by applying MVPM.

causing congestion. Further, packet injection limitation is applied with the proper intensity in accordance with the congestion degree detected in the network, thus minimizing the negative effects over the flows nonresponsible for congestion. As a consequence, the available network resources are evenly distributed among the devices that demand them, improving network throughput.

In what follows, we describe the new proposal, paying attention to the strategies applied to detect congestion and the corrective actions involved.

4.1 Congestion Detection

As it was analyzed in Section 3.1, both IPM and OPM strategies, when applied separately, have some drawbacks to detect in time the flows that are provoking congestion. As a consequence, a new packet marking strategy based on a combination of both IPM and OPM can be proposed and analyzed in order to remove the detected weaknesses and to join the benefits of both marking strategies.

We propose a new packet marking strategy referred to as Marking and Validation Packet Marking (MVPM), that combines packet marking at input and output buffers, in such a way that packets are marked at input buffers and validated at output buffers. To this end, 2 bits are dedicated in the packet header. To implement the mechanism in a standard interconnect, we can use any of the header bits usually reserved by the specs for vendor applications. Notice that by dedicating 2 bits in the packet header to warn the origin hosts about congestion, a more effective four-level scheme of corrective actions can be carried out at the source nodes.

The MVPM strategy works as follows: first, packets arriving to an input buffer are marked if the number of stored packets in the buffer exceeds a threshold. This is performed by activating the Marking Bit (MB) in the packet header. In the same way, when a marked packet is forwarded through a saturated output link, even in a different switch, we proceed to validate it by activating a second bit in the packet header, the Validation Bit (VB). We assume that an output link is saturated when the number of packets stored in its buffer exceeds certain threshold. Notice that a packet can be marked or validated several times, but never unmarked. Moreover, a packet cannot be validated if it has not been previously marked.

The advantage of applying MVPM consists of its capability to detect different levels of congestion by classifying the network flows in three types: *Hot-Flows* (flows truly responsible for congestion), *Cold-Flows* (flows nonresponsible for congestion), and *Warm-Flows* (flows that were cold-flows at the beginning, but as congestion has spread along the network, they are becoming *hot-flows*). Fig. 3 shows all the possible situations where these three types of flows can be identified when applying the MVPM strategy.

The w flow is crossing the switch without contention at any buffer. Therefore, no marking actions will be carried out on its packets. So, this flow will be classified as a *cold-flow*. Flow x is also identified as a *cold-flow*, as it is crossing a congested output buffer but not a congested input one. So, although this flow is contributing with some packets to the congestion at the output link, congestion has not yet reached the input buffer, and then it does not affect other flows that could share the input buffer, such as flow w. Therefore, there is no risk yet of spreading congestion to the previous switch. This situation can be identified when the total injection rate of x and y flows surpasses the accepted traffic at the output link $P_1^{'}$, but the injection rate of x flow is not high enough to accumulate packets at the input buffer, thus affecting the flow w. Therefore, those packets belonging to flow x will not be marked, avoiding premature corrective actions on that flows that are not necessary yet. Notice that marking packets belonging to flow x could cause gaps in throughput, while still network is actually able to accept this traffic.

Next, y flow is crossing both input and output congested buffers. Therefore, both MV and VB bits will be set, as an indication that this flow is a *hot-flow*. Finally, flow z is crossing the congested input buffer but it is not traversing any congested output buffer. This situation indicates that congestion has spread into the input buffer and packets belonging to z flow are suffering HOL blocking at input buffer due to the high injection rate of the *hot-flow* y. Although the flow z is not directly responsible for the initial congestion at the output buffer, in order to stop the spreading of congestion tree to the previous switch, this flow will be classified as a *warm-flow*. So, packets will set only the MB bit. As a result of this flow classification, source hosts can apply different levels of corrective actions over their respective flows (see below).

The most important key in this new packet marking mechanism is to correctly detect congestion at the switch where the saturation starts in order to prevent the spreading of congestion along the previous switches, which would create a new root of congestion, and to avoid applying premature corrective actions that could penalize performance.

4.2 Congestion Correction

One of the most effective procedures to avoid congestion consists of reducing the injection rate of the responsible flows. To this end, we propose two phases of corrective actions at the origin hosts.

The first phase is based on adjusting the packet injection rate by using a Dynamic Window (DW). It is based on the idea of limiting, for each flow, the number of outstanding packets into the network by using a window-based mechanism. In this case, the window size is dynamic, allowing to fluctuate between the maximum value (DW_{max}) and the minimum value of 1.

If congestion persists after the window size is fixed to one, a second phase of actions will reduce even more the injection rate by introducing a waiting interval between the injections of two consecutive packets. Notice that the corrective actions are not indiscriminately applied over all the flows issued by a host, but only over those responsible for congestion. In order

TABLE 1
Corrective Actions Applied by MVCM

Ack bits			
MB	**VB**	**Types of flows**	**Actions**
0	**0**	*Cold-Flow*	No Actions
0	**1**	Not possible	
1	**0**	*Warm-Flow*	Moderate (DW)
1	**1**	*Hot-Flow*	Imminent (DW+WP)

to carry out the different corrective actions, we rely on the reception of ACK packets, which allow us to identify the flows and to warn about the detected congestion degree. Table 1 shows the defined corrective actions according to the congestion level identified by the values of the MB and VB bits received on ACK packets. We will refer to as "imminent" the actions performed on those flows whose ACK packets have both bits set. On flows whose packets have only the MB set, we take only "moderate" actions to prevent that congestion expands to the previous switch.

Note that the combination of bits $\mathrm{MB} = 0$ and $\mathrm{VB} = 1$ is not possible in this packet marking strategy.

4.2.1 Reducing the Injection Rate

Next, we explain the different types of actions applied.

1. **Moderate actions** are applied to *warm-flows*. Only the DW size is modified, by reducing its current value by subtracting one per each marked ($\mathrm{MB} = 1$, $\mathrm{VB} = 0$) ACK packet received. If window size reaches its minimum value (one) and more marked ($\mathrm{MB} = 1$, $\mathrm{VB} = 0$) ACK packets arrive, the mechanism will keep the window size equal to one and no additional actions will be taken. This situation will remain till a nonmarked ACK packet arrives ($\mathrm{MB} = 0$, $\mathrm{VB} = 0$). Then, window size will be increased by adding one per each nonmarked ACK packet received until the DW_{max} is reached. This way, the injection rate for *warm-flows* will be decreased during the strictly necessary period of time, thus stopping the spreading of congestion tree. So, DW size will be adjusted into the interval $[1..\mathrm{DW}_{max}]$.

2. **Imminent actions** will be applied when validated packets ($\mathrm{MB} = 1$, $\mathrm{VB} = 1$) are received. That is, packets belonging to *hot-flows*. At the beginning, the mechanism reacts by reducing the DW size as moderate actions do, but if the DW size reduction is not enough to stop congestion, and more validated packets ($\mathrm{MB} = 1$, $\mathrm{VB} = 1$) continue being received, harder actions will be applied intended to reduce even more the injection rate. This second phase of actions starts when the DW size becomes equal to one. Then, WS will be inserted between consecutive injected packets in such a way that every received ACK packet with both MB and VB set will increase the number of WS, whereas received ACK packets with $\mathrm{MB} = 0$ and $\mathrm{VB} = 0$ will reduce the number of WS. The duration of each WS is assumed as the minimum time needed to send a packet plus to receive its ACK, that is, the minimum Round-Trip Time delay (RTT). The waiting interval calculation mechanism is shown in detail in Section 4.3. Notice that both situations can occur at the same time. That is, *warm-packets* or *hot-packets* can arrive during a congestion process indistinctly. In that situation, the mechanism will combine the actions previously described. Anyway, the injection rate has to be reduced till a minimum value depending on the network topology. When this value is reached, injection rate is not reduced anymore, regardless of continuing to receive more marked packets. Keeping this minimum injection rate is important to prevent an undue message throttling. The minimum injection rate will be theoretically analyzed in Section 4.3.

4.2.2 Recovering the Injection Rate

The strategy to recover the injection rate must meet the tradeoff between achieving a fast response time when congestion has finished and avoiding injecting too much traffic when the network is still congested.

Unlike other mechanisms such as QCN, which does not explicitly warn about a *noncongestion situation*, the MVCM uses the reception of nonmarked ACK packets at the origin host ($\mathrm{MB} = 0$, $\mathrm{VB} = 0$) to allow the progressive recovery of the initial values of the parameters for the congestion control mechanism (i.e., full injection rate). In this case, recovery period will depend on the value the DW has reached and the WS applied. When the first nonmarked ACK packet is received, the waiting interval applied to that flow will be immediately eliminated, thus allowing for a fast recovering but keeping the window size equal to one. If more nonmarked ACK packets arrive, DW will recover the initial value by adding one for each nonmarked ACK packet received. As a result, after receiving DW_{max} packets with $\mathrm{MB} = 0$ and $\mathrm{VB} = 0$, the full injection rate will be available (one packet for removing WS and DW_{max}-1 packets for restoring the DW size at its maximum value). Notice that the elimination of waiting slots causes a fast recovery by setting the WS parameter to zero, but keeping the dynamic window equal to 1. So, although it may seem that this first phase of recovery could introduce oscillations, it does not actually happen due to the fact that the dynamic window provides a proportional and slow recovery based on an additive increase, thus avoiding suddenly flooding the network above the injection rate it can assimilate.

In order to speed up even more the removal of the corrective actions when congestion is no longer detected, all parameters (DW and WS) will be immediately set to their initial values if an origin host injects a packet into an empty injection queue. Notice that if a host temporarily stops injection because it does not have any packet to inject, it does not longer contribute to congestion. Moreover, if no data packets are injected, then no ACK packets will be received. Hence, although this host is not generating traffic, it would not be able to recover the initial values of the parameters as no ACK packets will be received, which may penalize the network throughput. Consequently, when this situation occurs, the parameters controlling the corrective actions applied for this flow will be reset. That is, $\mathrm{DW} = \mathrm{DW}_{max}$ and $\mathrm{WS} = 0$. With this action, we get an immediate recovery. Therefore, as can be seen, the proposed mechanism takes corrective actions immediately against serious situations that could cause congestion in the network, but, as commented before, the mechanism does not introduce oscillations. However, if congestion takes place only during a brief period of time, recovery is also very fast. As a consequence, the mechanism does not penalize the network performance in the absence of congestion.

4.3 Parameter Initialization

The correct initialization of the parameters in a CMM is key to achieve good results. In order to make the parameters adjustment easy, simple, and robust, the initial values of the parameters in MVCM will depend on the network configuration. In what follows, we explain the strategy followed to define the initial values for the MVCM mechanism. The set of parameters to initialize includes: the *Buffer Threshold*, that defines the threshold to mark and validate packets; the *Window Size*, that limits the maximum number of outstanding packets per flow, and finally the *Waiting Interval Calculation* that defines the duration of the waiting interval and the method to calculate it. The value of some of these parameters depends on the network traffic conditions near the saturation point. Therefore, in order to carry out an estimation, we propose to run some simulations of the network under such traffic conditions.

1. Buffer Threshold. The input and output buffer thresholds should be chosen with those values such that, for a uniform traffic pattern,[4] packets are allowed to cross the switch without being marked, regardless of the network load. To define the buffer thresholds, we have calculated the average buffer occupancy for a uniform traffic pattern with an injection rate near the saturation point. The occupation at both input and output buffers were traced separately. In particular, we have obtained an occupation, on average, about 66 and 33 percent of the input and output buffer capacity, respectively, for all network configurations, regardless of the buffer sizes. Therefore, we have assumed those values for the input and output thresholds, respectively. Notice that, as we are analyzing the network near saturation, a traffic burstiness will not affect the results due to two reasons. First, in a MIN, the accepted traffic is maintained when network enters saturation. Second, the own flow control would limit packet injection. A more detailed analysis can be seen in the *Appendix A*, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2011.146.

2. Window Size. There is a proposal with a static window fixed to the value of one [15]. While this choice seems appropriate to palliate the congestion in a general scenario, it may negatively impact network throughput (this would be the case, for example, when only one host sends packets toward a single destination host). Moreover, if a severe congestion appears and packets are stopped at origin hosts, network could not be able to consume all the waiting packets at origins when congestions disappear due to the fact that the static window has a fixed value smaller than the optimum, as it will be shown later. Therefore, it is necessary to identify the optimum value for the window, let us call it ω, regardless of what window-based technique is applied: static or dynamic window.

In order to maximize network throughput, the window size can be calculated as the number of packets from a flow that can be injected into the network until receiving the first ACK at the origin host in the absence of contention in the network. So, it can be established according to the minimum RTT of the packets, which depends on the network depth or number of stages. Hence, the minimum RTT can be calculated as the sum of the time required by the data packet to reach its destination (t_{data}) plus the time required to receive the ACK packet (t_{ack}) at the origin host:

$$RTT_{min} = t_{data} + t_{ack}.$$

Let us assume a simple approach to calculate the network base latency in networks under cut-through switching as follows:

$$t_{vct} = h * t_{hop} + \left(\frac{L}{B}\right),$$

where h defines the number of network hops, t_{hop} includes routing, switching, and link delays, L is the packet length, and B represents the channel bandwidth. Thus, the t_{data} and t_{ack} can be calculated as

$$t_{data} = h * t_{hop} + \left(\frac{L_{data}}{B}\right) \ \ and \ \ t_{ack} = h * t_{hop} + \left(\frac{L_{ack}}{B}\right),$$

where L_{data} and L_{ack} defines the data and ACK packet lengths, respectively. Therefore the minimum RTT can be obtained as:

$$RTT_{min} = 2 * h * t_{hop} + \left(\frac{L_{data} + L_{ack}}{B}\right).$$

The optimum value ω will be imposed by the maximum number of packets that can be sent by the source during the time interval defined by the RTT_{min}. Therefore,

$$\omega = \frac{RTT_{min}}{\frac{L_{data}}{B}}.$$

Now replacing RTT_{min} by its computed value,

$$\omega = \frac{2 * h * t_{hop} * B + L_{data} + L_{ack}}{L_{data}}.$$

Analyzing this formula, it can be observed that the value of $(2 * h * t_{hop} * B + L_{ack})$ results in a constant K. As a result of that, the value of ω will be bounded by the maximum and minimum allowed data packet sizes

$$\frac{K + Min(L_{data})}{Min(L_{data})} \leq \omega \leq \frac{K + Max(L_{data})}{Max(L_{data})},$$

where $L_{data} = L_{head}$[5] $+L_{payload}$ and $K = (2 * h * t_{hop} * B + L_{ack})$.

As an example, given a bidirectional network topology 4-ary 5-fly (512 hosts and 640 switches) where the number of stages is five ($h = 9$), and assuming the following fixed values: $t_{hop} = 3$ cycles, $B = 1$ byte/cycle, $L_{ack} = 22$ bytes, and $L_{head} = 22$ bytes, then ω will vary between two limits depending on $L_{payload}$. That is, if $L_{payload}$ tends toward ∞ , then $\omega \approx 1$, but if $L_{payload}$ tends toward zero then $\omega \approx \frac{98}{22} = 4,45$.

So, the optimal ω value will be bounded by the interval [1, 4.45]. Now, in particular, assuming a fixed data payload equal to 256 bytes, we obtain that the optimum ω value is:

4. In this traffic pattern, destination of packets is randomly chosen among all network nodes.

5. L_{head} comprises the control bytes in the header of all data packets.

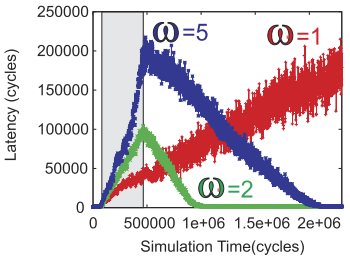

Fig. 4. Latency with different window sizes for a 4-ary 5-fly network configuration.

$$\omega = \frac{K + (L_{head} + L_{payload})}{(L_{head} + L_{payload})} = \frac{76 + (22 + 256)}{22 + 256} = \frac{354}{278} \approx 1.28.$$

Therefore, a fixed value of one, as Renato's strategy defines, would restrict too much the injection rate as the given network configuration may accept the injection of more packets.

In order to confirm by simulation this theoretical result, let us assume a uniform traffic destination distribution and an injection rate near the saturation point. In this scenario, we have obtained a window size for the network load parameters assumed above of $\omega \approx 1.4$ packets. This value has been obtained by measuring the average number of outstanding packets per flow during all the simulation time. As the calculated value is 1.28 packets, and the one achieved by simulation is 1.4 packets (larger than 1.28 because a slighter congestion appears into the network), the window size to be chosen will be $\omega = 2$, which is the nearest integer value which does not penalize network throughput as a smaller value would do. Larger values for the window size will produce a delay on reducing the injection rate when congestion appears.

In order to check by simulation the initial value selected for the window size, we show network performance results when applying a static window mechanism to palliate the effects of a congestion situation. No other corrective strategy has been used, only a fixed window size with different values during all the simulation. Fig. 4 shows the influence of the window size for a 4-ary 5-fly network configuration when applying synthetic traffic (Pattern I, see Section 5) with a high injection rate.

The shaded area in Fig. 4 indicates the period during which the origin hosts are injecting hot-spot traffic to create a congestion situation. We can observe that using a window size equal to 1, lowest latency is obtained when congestion appears. However, once the hot-spot disappears, latency would never recover its initial values because a window size of 1 does not allow to drain all the packets accumulated at origin hosts during the congestion period. As we can see, a window size of 2 provides the best behavior. Notice that when the network is not saturated, a larger window size does not provide advantages, since the ACKs arrive to the origin before more than two packets can be injected into the network. However, as the network is getting saturated, packet contention arises and packets stay longer at the switch queues, increasing the RTT and delaying ACKs. In this situation, a larger window size allows to have more packets in transit through the network, thus leading to a even more congested situation, as can be observed for a window size equal to 5.

Finally, in environments with large line speeds, wide data paths, or short data packets, a greater window value must be used, but the mechanism continues to work correctly. For example, in a 4-ary 5-fly network with data packets $= 256$ bytes and a link rate of 16 bytes/cycle, a value of $\omega \approx 4.18$ is obtained, so $\omega = 5$ is used.

Results showed that MVCM also correctly reacted against congestion by reducing the DW until the minimum value of 1. However in this situation, this window strategy was almost enough to stop congestion. Only a few flows needed to additionally insert WS. The conclusion is that when a network configuration has a high DW value, the correction mechanism continues to function properly.

3. **Waiting Interval Calculation.** Additionally to the dynamic window technique, the injection rate reduction is carried out by adding WS between consecutive injected packets. A wrong election of both the duration of a Waiting Slot (WS) and the total number of WS (*number_WS*) inserted between consecutive packets, could cause undesirable effects, as oscillations, unstable performance, etc. Previous proposals define both parameters depending on either the *Hotspot Degree* as Pfister's implementation does or a constant value that is used to regulate the injection rate as Renato's proposal does. However, both procedures could not be useful under other network configurations or traffic loads. So, in order to define a general strategy able to be applied in every network and traffic conditions, it is recommended to define both parameters according to the network configuration parameters. Therefore, it is interesting to find a relationship with those parameters.

In particular, the switch radix value (k) could be used to calculate the *number_WS* value because it defines the maximum number of hosts connected to a switch. So, in a congested situation this value could be used to evenly reduce the injection rate in a fair proportion. Additionally, the duration of the WS could be related to the RTT_{min} as it is the minimum time needed to send a packet plus to receive its corresponding ACK.

As a result, the waiting interval is calculated as follows: in absence of congestion, the *number_WS* value is equal to 0. When the first ACK packet with its MB and VB set is received, and DW is equal to one, then *number_WS* is set to the value of one. If more validated ACK packets are received, then the *number_WS* value will be increased by a constant factor equal to the switch radix (k). Thus, the *number_WS* value will be multiplied by k each time a new ACK packet is received as it is shown in (1). Assuming that WS is equal to RTTmin, the waiting interval is calculated as the product of the number of WS by RTTmin (2)

$$number\_WS_{current} := k * number\_WS_{previous}, \quad (1)$$

$$WaitingInterval := number\_WS_{current} * RTTmin. \quad (2)$$

Combining both functions (1) and (2) with the reception of some validated packets produces the results shown in Table 2.

On the other hand, the maximum number of WS inserted between consecutive packets has to be bounded in order to prevent packets from being stopped too much at origin hosts while network throughput is being reduced. To this

TABLE 2
Reduction Procedure for the k-Ary n-Fly Network

MB=VB=1	number_WS	WaitingInterval
0	0	0
1	$k^0 = 1$	RTT
2	k^1	$k^1 * RTT$
3	k^2	$k^2 * RTT$
...	...	...

end, the network parameter n, that identifies the number of stages of the network, could be used to limit the maximum number of times the *number_WS* value can be increased.

To justify the use of both k and n as parameters for calculating the waiting interval, a theoretical analysis is presented in *Appendix B*, available in the online supplemental material.

5 Performance Evaluation

5.1 Network Configurations

The MVCM mechanism has been evaluated and compared with other CMM proposals in the same scenario by using an interconnection network simulator. A generic switch-based cut-through network with point-to-point links and buffered credit-based flow control with a transmission link rate of 1 byte/cycle was assumed. Packets will be transmitted over the link if there is enough buffer space (measured in credits of 64 bytes) to store the entire packet. Switches requires buffers associated at both their input and output ports (CIOQ switches). Following the results presented in [8], a switch speedup of two has been assumed for all the simulations in this study.

A deterministic routing algorithm is used to forward packets in the network. We have evaluated several network configurations with different values of switch radix (k) and number of network stages (n): perfect Shuffle with bidirectional links (4-ary 3-fly, 4-ary 4-fly, 4-ary 5-fly, 8-ary 3-fly) and some configurations for Butterfly MINs with unidirectional links (2-ary 5-fly, 2-ary 6-fly, 2-ary 7-fly, 4-ary 3-fly, 4-ary 4-fly). These scenarios have been simulated with low, intermediate, and high injection rates. In order to eliminate the HOL blocking phenomenon at origins, we assume a Full Virtual Output Queuing (VOQ) at the NICs of the source hosts. This implementation defines a number of queues equal to the number of destination hosts. Each origin host generates packets, classifying and storing them in a queue depending on their destination host. The number of packets that can be stored in a queue is equal to the DW size, because it limits the outstanding packets per flow. Each time an ACK packet arrives at the origin host, the first buffer in the corresponding queue is released allowing to inject another data packet into the network. Notice that the queues are under the FIFO policy.

For simulation purposes, in order to know the time when packets have been generated and therefore, to calculate the latency from generation time, the origin hosts do not stop generating packets although the queues at NICs are full. In an actual implementation, origins would stop generating packets till a buffer is released at the corresponding queue.

Simulations have been carried out with different data packet sizes. We have considered both fixed payloads of 256, 512, and 1,024 bytes and variable payloads of 256 up to 1,024 bytes. In all the configurations, 22 bytes of header have been assumed. Moreover, the ACK packet size is 22 bytes for all cases. The sizes of the buffers used in the simulations are 1, 2, and 4 Kb for the 256, 512, and 1,024 bytes of fixed payload, respectively, and 3 Kb for the variable payload. In addition, different traffic patterns based on traces and synthetic traffic have been also evaluated. These patterns provoke network congestion with different intensity levels and are intended to check the analyzed proposals under diverse traffic conditions. Although in all simulation scenarios we have obtained very good results, we only show some results for some bidirectional network configurations (4-ary 5-fly, 4-ary 4-fly, 4-ary 3-fly, and 8-ary 3-fly) with a fixed packet size of 256 bytes (payload) plus 22 bytes (header), and with two different traffic patterns, that is, synthetic traffic and traces.

TABLE 3
Synthetic Traffic Pattern

4-ary 5-fly	4-ary 4-fly	4-ary 3-fly	8-ary 3-fly	Traffic type
#Sources	#Sources	#Sources	#Sources	Destination
448	224	56	448	Uniform
64	32	8	64	Stop-HS-Stop

(HS stands for hot-spot).

For synthetic traffic, two types of flows have been applied, that is, flows injecting uniform traffic and flows injecting hot-spot traffic. Table 3 shows the traffic pattern applied to all network configurations. We first generate packets according to a uniform distribution of destinations. Then, we create a hot-spot in the network and analyze what happens with both cold and hot-flows. Hosts that send uniform traffic remain injecting packets during the whole simulation, and hosts that generate hot-spot traffic remain inactive until the first 50,000 packets have been received. Then, they start injecting 1,000 packets from each origin host with the same injection rate as that of the other hosts, but addressed to a single destination host (the hot-spot). Later, when each one has completed the injection of the packets, they stop generating packets.

A medium network load has been applied for all the configurations whose simulation results are displayed in the next section. In particular, for the 4-ary 5-fly network, an injection rate of 0.45 bytes/cycle/sw[6] has been applied. That is, a total amount of 58 bytes/cycle due to the fact that this configuration network has 128 switches at the first stage. Qualitatively, similar results are obtained for other injection rates (0.26 and 0.62 bytes/cycle/sw for the low and high injection rate, respectively) and for the rest of configuration scenarios. Additionally, a traffic load based on traces has been also applied. In particular, the traces used in our evaluations were provided by the applications DL_POLY [10] and CPMD [9], and they have been also run simultaneously in order to stress even more the traffic network.

5.2 Evaluation Results

First, we show an analysis of the MVCM proposal when it is applied in different network configurations and with two types of traffic load, synthetic, and traces. The aim of this

6. sw refers to the switches with source hosts connected to them (i.e., the switches of the first stage).

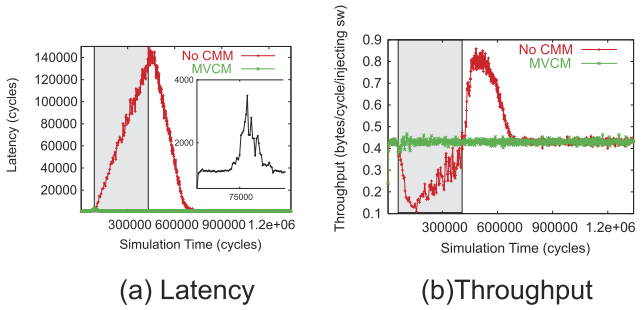

Fig. 5. Latency and throughput for cold-flows in a 4-ary 5-fly network with No CMM, and MVCM with $DW_{max} = 2$.

analysis is to verify if the proposed mechanism is able to reduce congestion regardless of the network configuration and the traffic conditions.

Next, a comparison study between the current CMMs and the proposed MVCM, under the same network configurations and traffic loads, is presented in order to determine if MVCM outperforms previous proposals. Notice that, in all the graphs, the latencies shown are computed since generation time.[7]

5.2.1 Evaluating the MVCM Proposal

In order to check if the MVCM proposal is able to reduce the undesirable effects caused by a congestion situation, first, we analyze how the mechanism reacts against the congestion created in a 4-ary 5-fly network when applying the synthetic traffic shown in Table 3 and with a medium traffic load. Fig. 5 shows how the performance of the cold-flows are affected when the corrective actions are applied to the flows responsible for the congestion. In this figure, the latency and throughput are represented when applying no CMM and the proposed MVCM.

In particular, Fig. 5a shows latency while Fig. 5b shows throughput. Notice that the shaded area in graphs indicates the period during which the origin hosts are injecting hot-spot traffic to create the congestion situation. Additionally, the small graph inside Fig. 5a represents a zoomed version of the MVCM graph from the main figure around the point where the hot-spot traffic begins. Along it, latency increases up to more than 140,000 cycles when no CMM is applied as it can be seen. However, when applying the MVCM proposal, the maximum value for latency is reduced till 3,000 cycles, approximately. In the same way, a throughput drop is produced during the congestion situation, as it is shown in Fig. 5b. Again, MVCM reacts avoiding this sharp drop and providing a sustained performance level thanks to minimize the throughput degradation.

Next, in order to validate the fairness of the strategy applied to reduce the congestion, it is mandatory to verify that MVCM does not excessively reduce the injection rate of flows injecting packets toward the hot-spot. To this end, Fig. 6 shows the percentage of utilization of the link connected to the hot-spot for the analyzed 4-ary 5-fly network under the synthetic traffic pattern.

Figs. 6a and 6b present results when no actions are taken and when the corrective actions are applied over the

7. Time required to deliver a packet including the time spent waiting at the origin host.

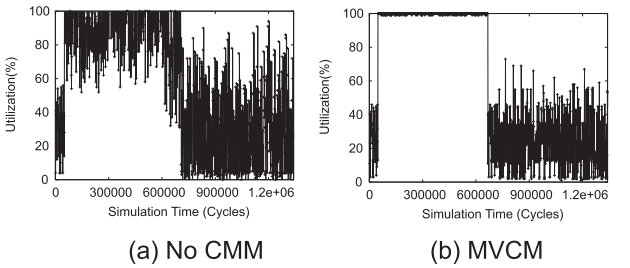

Fig. 6. Utilization of the link connected to the hot-spot for a 4-ary 5-fly network.

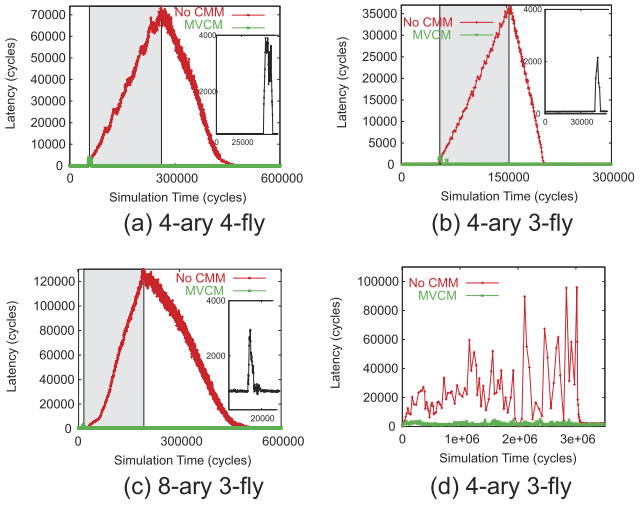

Fig. 7. (a), (b), and (c) latency for cold-flows with $DW_{max} = 2$, and (d) latency for traffic based on traces.

hot-flows, respectively. Notice that during the congestion period, the performance of the analyzed link shows that it is busy at 100 percent when the MVCM proposal is applied, graph 6b. Moreover, when congestion disappears, the utilization values present less oscillations than when no congestion management is applied. So, although corrective actions are stopping packets belonging to hot-flows at origins to provide enough bandwidth to the cold-flows, those hot-flows continue to inject enough packets into the network to keep busy the link connected to the host-spot. Therefore, the MVCM proposal also benefits the hot-flows. This improvement is due to the application of a DW strategy in the first phase of the set of corrective actions. A higher injection of packets belonging to hot-flows would create HOL blocking into the network and, therefore, the latency for cold-flows would increase.

In order to confirm that the MVCM mechanism achieves good performance regardless of the network size, Fig. 7 presents the results of latency for cold-flows with different network configurations. In particular, Figs. 7a, 7b, and 7c show results when No CMM and MVCM are applied in a 4-ary 4-fly, 4-ary 3-fly, and 8-ary 3-fly, respectively. For these network configurations, the traffic pattern applied is the one shown in Table 3. The shaded area also indicates the period the origin hosts are injecting hot-spot traffic to create a congestion situation. Again, the small graphs inside the main figure are zoomed versions of the main figure. As can be observed, in all the network configurations, the MVCM mechanism reduces the negative effects on cold-flows during

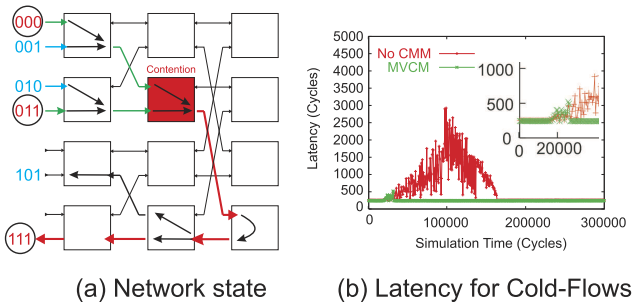

Fig. 8. Congestion situation created by two flows.

the congestion situation. For the three network configurations, the maximum values of latency are significantly reduced when the corrective actions have been applied.

Finally, it is interesting to check the behavior of the MVCM proposal when a real traffic load based on traces is applied. The used traces are the ones mentioned in Section 5.1. For the analysis, they have been applied in a 4-ary 3-fly network. Fig. 7d shows the latency when no corrective actions are taken and when the MVCM proposal is applied. As it can be seen, different congestion points appear and disappear during the simulation time, increasing the value of the latency. As a result of applying the MVCM proposal, the mechanism reacts palliating the congestion effects. Again, the MVCM mechanism significantly reduces the negative effects of the congestion by reducing the latency for cold-flows.

To better illustrate that our mechanism also works in specific situations, next we show results where a couple of flows congest one link. Fig. 8a presents a 2-ary, 3-fly network where two flows, (000-111 and 011-111), create congestion in a switch at the second stage. To evaluate how the MVCM mechanism is able to palliate the negative effects of congestion over the cold-flows crossing this switch, we have traced the flows from hosts 001 and 010, and destined to any node different from the hot-spot (111). In particular, only the flows targeted to the destination host 101 has been traced. Fig. 8b shows the performance evaluation results for cold-flows when no congestion management mechanism and MVCM are applied. As can be seen, although only two hot-flows create congestion, they affect a few cold-flows and MVCM can correctly detect the beginning of the congestion in time, also applying corrective actions.

5.2.2 Comparing MVCM with Other CMMs

A comparison between the proposed mechanism (MVCM) and the current CMMs (Renato and Pfister), is shown in Fig. 9a. Graphs in this figure represent latency versus accepted traffic for the three analyzed mechanisms in a 4-ary 5-fly network when applying the traffic pattern shown in Table 3. In particular, each represented value in the graphs identifies the average packet latency for all the generated packets during the simulation, that is, packets belonging to either cold or hot-flows. Graphs RP, PI, and MVCM in Fig. 9a, represent the Renato's Proposal, the Pfister's Implementation and the Marking and Validation Congestion Management mechanisms, respectively. Notice that to simulate the Renato's proposal, the LIPD response function has been applied in all cases to make a fair comparison.

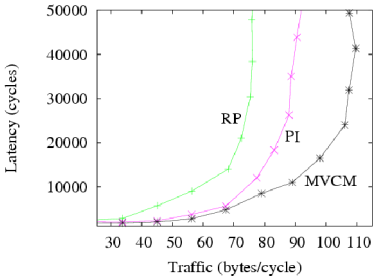

Fig. 9. Analysis of CMMs in a 4-ary 5-fly network.

As it can be observed, Renato's proposal shows the worst results. The achieved results are mainly due to the application of the static window technique with a fixed size equal to one at any situation. In absence of congestion, the mechanism cannot take advantage of the available bandwidth because this strategy restricts too much the injection rate due to the packet limitation imposed by the static window. Moreover, its IPM strategy is not enough to correctly identify the flows truly responsible for congestion.

In the same way, although the Pfister's implementation achieves better results than Renato's proposal, it suffers from several weaknesses, as described below. First, since it applies a OPM strategy, it does not guarantee that corrective actions will be timely applied. Second, as this proposal does not use any window control, it is unable to limit the number of outstanding packets. This technique works well with a low injection rate, but when traffic load approaches to medium or high injection rates, it cannot react quick enough. Moreover, when marked ACK packets begin to arrive to the origin hosts, it directly inserts WSs that could limit the injected traffic rate in excess. Third, it does not introduce any limitation in the application of the WS insertion technique, as MVCM does. Finally, not applying an *immediate recovery* action, as MVCM does, could produce a delay in recovering the injection rate if congestion vanishes suddenly.

On the contrary, the MVCM mechanism obtains the best results as a consequence of combining a more thorough and refined packet marking strategy with a fairer and effective corrective action scheme. Later on, when congestion is reduced, the applied strategy allows to recover the initial injection rate in a *progressive way*, as long as nonmarked ACK packets are received, or quickly, setting all the parameters in the origin host to their initial values if the host finds an empty injection queue when it goes to inject a new packet into the network.

In order to analyze the impact of the more efficient packet marking strategy applied by MVCM, Fig. 10 shows the percentages of marked packets carried out by the three compared mechanisms, separately analyzing cold

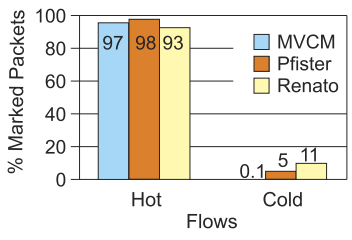

Fig. 10. Percentage of marked packets.

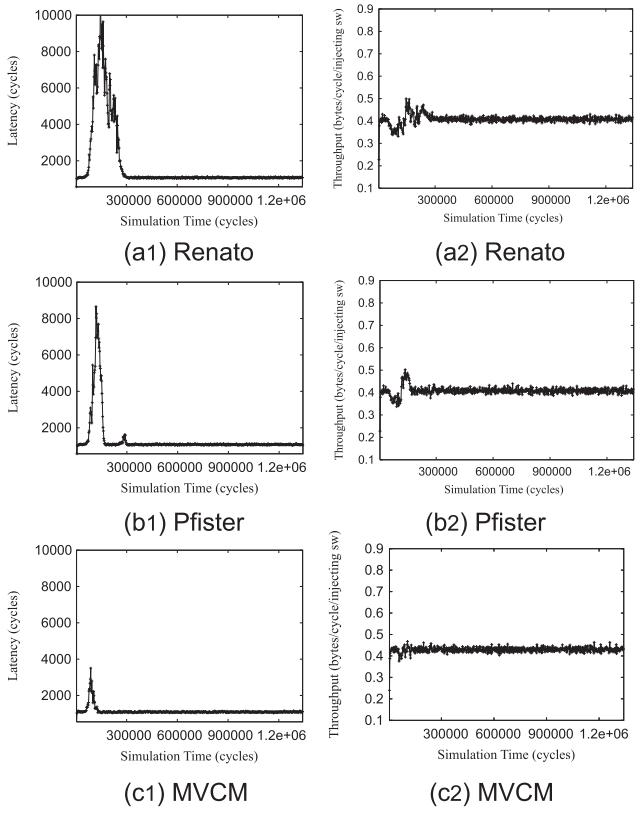

Fig. 11. Latency and throughput for cold-flows in a 4-ary 5-fly network.

and hot-flows. Values represent the percentage of marked packets of each type. As can be seen, although the values of marked hot-packets show small differences among them, Pfister's marking strategy reaches the highest value, whereas Renato's strategy achieves the lowest one. The MVCM strategy achieves a value slightly lower than the Pfister's one because MVCM defines a dependency between marking and validation actions. In contrast, marking values for cold-packets show important differences. Renato's and Pfister's strategies mark about 11 and 5 percent of the packets belonging to flows nonresponsible for congestion, respectively, whereas MVCM strategy reaches the minimum value (i.e., about 0.1 percent). This low value is due to the application of a packet marking strategy based on 2 bits, which is able to correctly identify the responsible flows for congestion.

Notice that as the number of cold-packets is greater than that of hot-packets, a small difference in the percentage of marked cold-packets represents a great amount of wrongly marked packets, which could penalize the applications that are sending or waiting for those packets.

Although the aim of a well-structured packet marking strategy is to detect and mark packets belonging to flows responsible for congestion, the key to reach good results is to avoid marking packets belonging to nonresponsible flows. The consequence of this negative effect is that, hosts could be wrongly warned about congestion not created by them, and therefore apply some corrective actions that are not necessary.

Next, it is interesting to compare and analyze the behavior of cold-flows when the current CMMs (Renato and Pfister) are applied, comparing their results to the ones achieved by the MVCM proposal. Fig. 11 shows latency

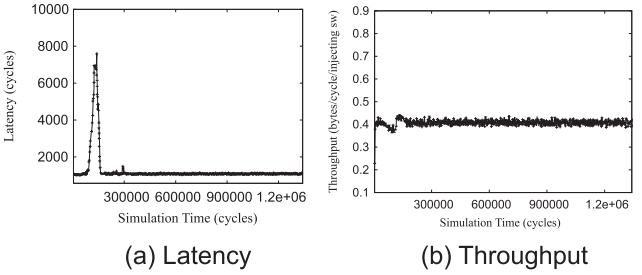

Fig. 12. Latency and throughput for cold-flows in a 4-ary 5-fly network when applying the Pfister's corrective strategy only to validated packets.

and throughput for a 4-ary 5-fly network with the traffic pattern shown in Table 3. The following mechanisms have been applied, (a_i) the Renato's proposal, (b_i) the Pfister's implementation, and (c_i) the MVCM mechanism. In particular Figs. 11a1, 11b1, and 11c1 represent latency while Figs. 11a2, 11b2, and 11c2 represent throughput. Comparing these results, we can observe that both Renato's proposal and Pfister's implementation improves cold-flows performance over the results obtained with no CMM (see Figs. 5a and 5b). In graph *No CMM* in Fig. 5a, latency reaches values greater than 140,000 cycles while Renato's proposal reduces latency up to 10,000 cycles and Pfister's implementation up to more than 8,000 cycles, as shown in Figs. 11a1 and 11b1, respectively. However, comparing their results with the ones achieved by the MVCM proposal in Fig. 11c1, it can be observed that MVCM achieves the best latency values, because latency is reduced till 3,000 cycles. In the same way, as shown in graphs 11a2 and 11b2, both proposals react palliating the throughput drop with regard to the absence of CMM (see graph 5b). However, again, MVCM achieves the best performance by reducing even more the throughput drop, and the undesirable oscillations, as it is shown in Fig. 11c2 .

Following, we illustrate the effectiveness of the marking and validating packets with respect to pure OPM and IPM marking. As described in Section 4.1, the MVCM packet marking strategy does not validate packets if they have not been previously marked at input buffers. In order to verify the positive effects of this strategy, let us analyze what would happen if the Pfister's packet marking strategy (pure OPM mechanism) marked only packets truly responsible for congestion (i.e, packets marked and validated as MVCM).

Figs. 12a and 12b show latency and throughput, respectively, when applying the Pfister's corrective actions only over the packets truly responsible for congestion (validated packets). As can be seen, both the latency and most of the oscillation in throughput appeared in Figs. 11b1 and 11b2, have been reduced in Fig. 12. This is because marking only packets validated at output buffers does not penalize other flows nonresponsible for congestion, thus increasing network performance.

Figs. 13a and 13b show latency and throughput, respectively, when applying the MVCM corrective actions only over the marked packets at input buffers (pure IPM mechanism). These results can be compared to the ones in Figs. 11c1 and 11c2. As can be seen, the IPM strategy causes an increase in latency and slightly introduces oscillations in throughput. These effects are caused by marking cold-packets crossing a congested input buffer, which are wrongly identified as hot-packets.

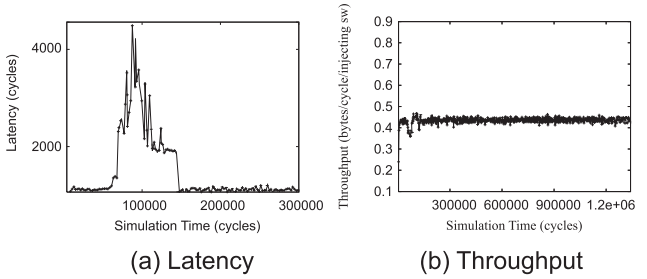

Fig. 13. Latency and throughput for cold-flows in a 4-ary 5-fly network when applying the MVCM corrective strategy only to marked packets (not validated).

Finally, we analyze the benefits of identifying warm-packets as MVCM does. Fig. 14b shows latency for cold-flows for the MVCM congestion management mechanism but assuming that warm-packets are not marked. In its turn, Fig. 14a presents the same results as Fig. 11c1, but with different scales in both axes. As can be seen in Fig. 14b, latency of cold-packets is penalized when warm-packets are not marked. This is because not applying corrective actions on them causes congestion to quickly spread to previous switches, thus affecting other cold-flows that were not involved in the initial congestion. Notice that, although this increment in latency does not seem very relevant, the affected packets could belong to a critical application, for which small delays could negatively impact its performance.

Next, Figs. 15b, 15c, and 15d show the evaluation results of MVCM, Renato's proposal, and Pfister's implementation, respectively, for the traffic pattern based on traces (Section 5.1). As it can be observed, the achieved results are better than the ones shown in Fig. 15a where no CMM is applied. But again, MVCM achieves the best performance, as it is observed in Fig. 15b, due to the fact that MVCM reduces the maximum values of latency below those obtained by the other two proposals during all the simulation. In particular, MVCM reaches a maximum value of latency lower than 5,000 cycles and a value of 1,300 cycles, on average, during the simulation time. On the other hand, Renato's and Pfister's ones reach a maximum value about 12,000 and 10,000 cycles, respectively, and an average value of 4,500 cycles during all the simulation.

5.2.3 Analyzing the Effect of WS Insertion Limitation

In this section, we show how the WS insertion limitation imposed by MVCM contributes to improve performance of hot-flows. To this end, we analyze what happens if there is not limitation on inserting WSs. As it was shown in Section 4.2.3, MVCM limits the number of inserted WSs according to the number of network stages. Applying the waiting interval

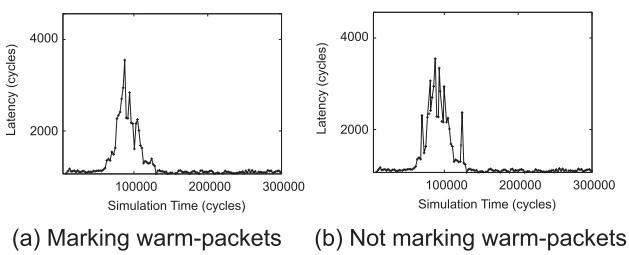

Fig. 14. Latency for cold-flows in a 4-ary 5-fly network for MVCM with/without warm-packets marking.

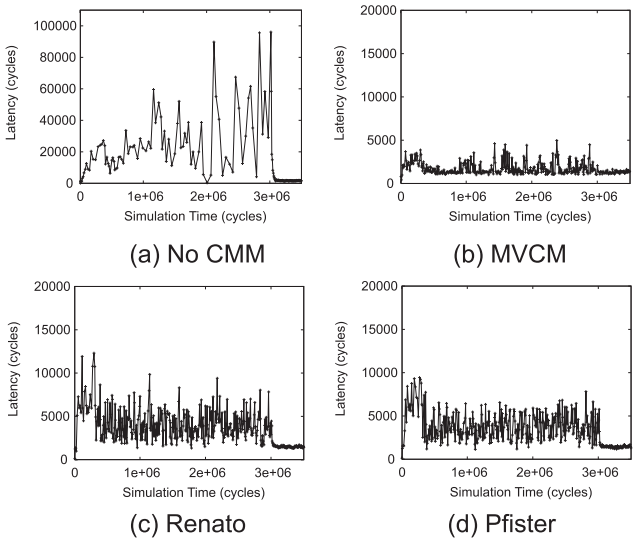

Fig. 15. Latency with traffic based on traces for a 4-ary 3-fly.

calculation without any limit will reduce too much the injection rate of packets, stopping them at origins while network is getting idle. To justify this, let us analyze what happens when there is not any limitation in WS insertion.

Fig. 16a represents the average latency of packets belonging to the hot-flows when the MVCM proposal is applied in a 4-ary 5-fly. For the MVCM mechanism, we assume its native strategy based on limiting the times the number of WSs can be incremented on ACK arrivals. Fig. 16b shows the performance when MVCM is applied without any limitation on the number of WSs. As can be observed, some gaps appear as a consequence of increasing more than n-times the number of WSs. This is because, despite the fact that there is available network bandwidth, several packets are stopped at the origin hosts because the waiting interval applied is too long. Also, as it can be seen in Fig. 16c, Pfister's implementation also suffers from the same problem, because, unlike MVCM, this CMM applies the waiting interval calculation strategy without imposing any limitation as MVCM does. Moreover, the

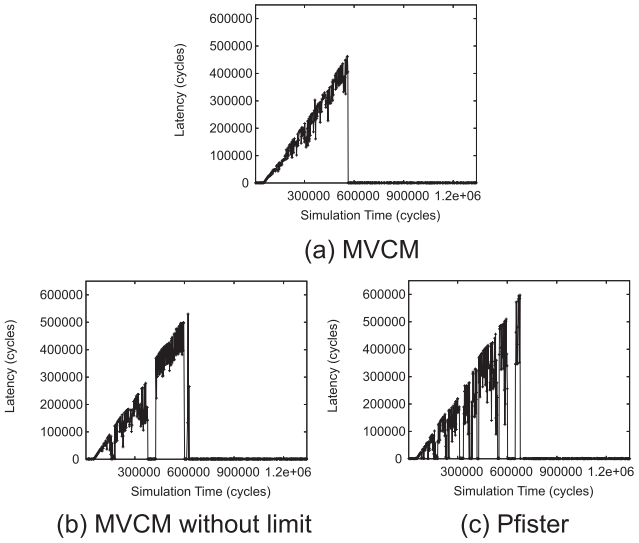

Fig. 16. Latency for hot-flows in a 4-ary 5-fly.

Pfister's implementation presents a worse performance due to the fact that, it does not apply any window-based mechanism, which reduces oscillations.

6 Conclusion

In this paper, we have proposed a new congestion management mechanism for MINs, based on Explicit Congestion Notification and referred to as Marking and Validation Congestion Management (MVCM). This new mechanism contributes with both a new packet marking strategy, that efficiently detects the root of congestion and correctly classifies flows belonging to the tree congestion, and a fair set of corrective actions, that makes packets belonging to the flows responsible of congestion wait at their source hosts, instead of remaining blocked into the network.

As the adjustment of the parameters of the proposed mechanism only depends on the network configuration, the mechanism is simple, robust, and easy to implement.

The MVCM proposal has been globally evaluated, showing that it provides good performance for congestion management regardless of the traffic load. Moreover, the MVCM performance not only reduces latency for cold-flows with regard the current proposals, but also improves the performance of hot-flows by avoiding oscillations in network throughput and keeping their packet injection at the maximum rate.

To conclude, performance evaluation results shows that the MVCM proposal is able to effectively manage congestion in MINs regardless of network configuration, traffic load, and congestion degree.

Acknowledgments

This work was supported by the Spanish MEC and MICINN, as well as European Commission FEDER funds, under Grants CSD2006-00046 and TIN2009-14475-C04-01, and by the European Commission in the context of the SARC integrated project #27648 (FP6).

References

[1] M. Allman, V. Paxsm, and W. Stevens, "TCP Congestion Control," http://www.rfc-editor.org/rfc/rfc2581.txt, 1999.
[2] M. Alonso, S. Coll, J. Martinez, V. Santonja, P. Lopez, and J. Duato, "Dynamic Power Saving in Fat-Tree Interconnection Networks Using On/Off Links," *Proc. 20th Int'l Conf. Parallel and Distributed Processing (IPDPS '06),* 2006.
[3] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High-Speed Switch Scheduling for Local-Area Networks," *ACM Trans. Computer Systems,* vol. 11, pp. 319-352, 1993.
[4] W. Dally, P. Carvey, and L. Dennison, "The Avici Terabit Switch/Router," *Proc. Hot Interconnects,* 1998.
[5] P. Devkota and A. Reddy, "Performance of Quantized Congestion Notification in TCP Incast Scenarios of Data Centers," *Proc. IEEE Int'l Symp. Modeling, Analysis, and Simulation of Computer and Telecomm. Systems (MASCOTS),* 2010.
[6] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo, "A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks," *Proc. 11th Int'l Symp. High-Performance Computer Architecture,* 2005.
[7] J. Ferrer, E. Baydal, A. Robles, P. Lopez, and J. Duato, "Congestion Management in MINs through Marked & Validated Packets," *Proc. 15th Euromicro Int'l Conf. Parallel, Distributed and Network-Based Processing (PDP '07),* 2007.
[8] P. Garcia, J. Flich, J. Duato, I. Johnson, F. Quiles, and F. Naven, "Dynamic Evolution of Congestion Trees: Analysis and Impact on Switch Architecture," *Proc. Int'l Symp. HiPEAC,* 2005.
[9] http://www.cpmd.org/, 2011.
[10] http://www.cse.scitech.ac.uk/ccg/software/DLPOLY/, 2011.
[11] http://www.infinibandta.org, 2011.
[12] M. Katevenis, D. Serpanos, and E. Spyridakis, "Credit-Flow Controlled ATM for MP Interconnection: The ATLAS I Single-Chip ATM Switch," *Proc. Fourth Int'l Symp. High-Performance Computer Architecture (HPCA '98),* 1998.
[13] V. Krishnan and D. Mayhew, "A Localized Congestion Control Mechanism for PCI Express Advanced Switching Fabrics," *Proc. IEEE Symp. Hot Interconnects,* 2004.
[14] G. Pfister and V. Norton, "Hot Spot Contention and Combining in Multistage Interconnection Networks," *IEEE Trans. Computers,* vol. 34, no. 10, pp. 943-948, Oct. 1985.
[15] G. Pfister et al., "Solving Hot Spot Contention Using Infiniband Architecture Congestion Control," *Ion High Performance Interconnects for Distributed Computing,* 2005.
[16] J. Renato Santos, Y. Turner, and G. Janakiraman, "End-to-End Congestion Control for Infiniband," *Proc. IEEE INFOCOM,* 2003.
[17] L. Shang, L. Peh, and N. Jha, "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," *Proc. Ninth Int'l Symp. High-Performance Computer Architecture (HPCA),* 2003.
[18] A. Smai and L. Thorelli, "Global Reactive Congestion Control in Multicomputer Networks," *Proc. Fifth Int'l Conf. High-Performance Computing (HIPC '98),* 1998.
[19] M. Thottetodi, A. Lebeck, and S. Mukherjee, "Self-Tuned Congestion Control for Multiprocessor Networks," *Proc. Seventh Int'l Symp. High-Performance Computer Architecture,* 2001.
[20] W. Vogels et al., "Tree-Saturation Control in the AC3 Velocity Cluster Interconnect," *Proc. Conf. Hot Interconnects,* 2000.

Joan-Lluís Ferrer received the MS degree in computer science from the Universitat Politècnica de València, Spain in 1994. He is currently working toward the PhD degree in the Department of Computer Engineering (DISCA) at the Universitat Politècnica de València. His research interests include the field of congestion management in high-performance interconnection networks.

Elvira Baydal received the MS degree in physics in 1987 from the Universitat de València (Spain), and the PhD degree in computer engineering from the Universidat Politècnica de València (Spain), in 2003. She was with the Department of Computer Science from the Universidad de Castilla-La Mancha (Spain), from 1987 until 1990 when she joined the Department of Computer Engineering (DISCA), where she is currently employed as an assistant professor of computer architecture and technology. She has taught several courses on computer organization, computer networks, and security networks. Her research interests include high-performance interconnection networks for multiprocessor systems and cluster of workstations, specially congestion control.

Antonio Robles received the MS degree in physics (electricity and electronics) from the Universitat de València, Spain, in 1984 and the PhD degree in computer engineering from the Universitat Politècnica de València in 1995. He is currently a full professor in the Department of Computer Engineering (DISCA) at the Universitat Politècnica de València, Spain. He has taught several courses on computer organization and architecture. His research interests include high-performance interconnection networks for multiprocessor systems and clusters and scalable cache coherence protocols for SMP and CMP. He has published more than 70 refereed conference and journal papers. He has served on program committees for several major conferences. He is a member of the IEEE Computer Society.

Pedro López received the BEng degree in electrical engineering and the MS and PhD degrees in computer engineering from the same university in 1984, 1990, and 1995, respectively. He is a full professor in computer architecture and technology at the Department of Computer Engineering (DISCA), Universitat Politècnica de València, Spain. He has taught several courses on computer organization and architecture. His research interests include high performance interconnection networks for multiprocessor systems and clusters, and processor microarchitecture. He has published more than 100 refereed conference and journal papers. He served as a member of the editorial board of *Parallel Computing Journal*. He is a member of the IEEE Computer Society.

José Duato received the MS and PhD degrees in electrical engineering from the Universitat Politècnica de València, Spain, in 1981 and 1985, respectively. He is currently a professor in the Department of Computer Engineering (DISCA) at the Universitat Politècnica de València. He was an adjunct professor in the Department of Computer and Information Science at The Ohio State University, Columbus. His research interests include interconnection networks and multiprocessor architectures. He has published more than 380 refereed papers. He proposed a powerful theory of deadlock-free adaptive routing for wormhole networks. Versions of this theory have been used in the design of the routing algorithms for the MIT Reliable Router, the Cray T3E supercomputer, the internal router of the Alpha 21364 microprocessor, and the IBM BlueGene/L supercomputer. He is the first author of the *Interconnection Networks: An Engineering Approach* (Morgan Kaufmann, 2002). He was a member of the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computers*, and the *IEEE Computer Architecture Letters*. He was a cochair, member of the steering committee, vice chair, or member of the program committee in more than 55 conferences, including the most prestigious conferences in his area of interest: HPCA, ISCA, IPPS/SPDP, IPDPS, ICPP, ICDCS, EuroPar, and HiPC. He has been awarded with the National Research Prize Julio Rey Pastor 2009, in the area of Mathematics and Information and Communications Technology and the Rei Jaume I Award on New Technologies 2006.