University of Castilla-La Mancha

A publication of the
Computing Systems Department

Efficient Congestion Management for High-Performance Interconnection Networks with Distributed Routing

by

J. Escudero-Sahuquillo, P.J. Garcia, F.J. Quiles, J. Flich, J. Duato

e-mail: jescudero@dsi.uclm.es

(J. Flich and J. Duato belong to Computer Engineering Department.
Technical University of Valencia)

Technical Report **#DIAB-10-04-1** April 2010

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. It was also partly supported by JCCM and the European Commission FSE 2007-2013 fund, under project PCC08-0078 (PhD. grant A08/048).

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS
ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34 967599200, Fax +34 967599224

Abstract

The Interconnection networks are essential elements in current computing systems, from PC Clusters to Massive Parallel Processors. For this reason, achieving the best network performance, even in congestion situations, has been a primary goal in recent years. In that sense, there exist several techniques focused on eliminating the main negative effect of congestion: the Head of Line (HOL) blocking. One of the most successful HOL blocking elimination techniques is RECN, which can be applied in source routing networks. FBICM follows the same approach as RECN, but it has been developed for distributed deterministic routing networks.

Although FBICM effectively eliminates HOL blocking, it requires too much resources to be implemented. In this paper we present a new FBICM version, based on a new organization of switch memory resources, that significantly reduces the required silicon area, complexity and cost. Moreover, we present new results about FBICM, in network topologies no yet analyzed. From the experiment results we can conclude that a far less complex and feasible FBICM implementation can be achieved by using the proposed improvements, while not losing efficiency.

Contents

Chapter 1

Introduction

The interconnection network is a key element in current parallel computing systems. In order to reduce its cost there exist some research works which try to design cost-effective networks using a limited number of resources while keeping the required performance level. In that sense, current network designs reduce network size in order to reduce power consumption and resource requirements, according to dropping processor prices. However, as network size decreases, so does network offered bandwidth, thus increasing the probability of congestion, as the saturation point of the network is reached with lower traffic loads. Therefore, congestion is a problem that must be solved in current cost-effective designs, since its consequences can be disastrous for network performance.

Specifically, congestion appears when several packet flows persistently request the same output port inside a switch, then most packets have to stop and wait for long. Assuming lossless networks (without packet discarding), buffers containing these blocked packets finally collapse. Moreover, flow control will propagate congestion to upstream switches, forming "congestion trees" [5]. When the "leaves" of these congestion trees reach many network points, the immediate consequence is a severe network performance degradation.

The specific cause of this degradation is that congested flows may share queues with non-congested flows, thereby the former slowing the advance of the latter. In detail, in a queue storing packets belonging to congested and non-congested flows, a "congested packet" reaching the head of the queue will usually have to wait for a long period before being forwarded, and consequently all the other packets in the queue (both congested and non-congested) will suffer this delay. This effect is known as Head-Of-Line (HOL) blocking, and it may limit the throughput of the switch up to about 58% of its peak value [9].

In current high-speed interconnection networks, the use of some technique for solving the problems related to congestion has become mandatory. In that sense, many proposed techniques can help to reduce the negative effects of congestion (see section 2). Among them, one of the most successful is Regional Explicit Congestion Notification (RECN)[3, 6, 10]. In contrast with other techniques that try to avoid or prevent congestion, RECN completely eliminates the HOL blocking, requiring a limited, reduced set of additional queues per switch port. However, RECN was proposed for networks which use source routing, so not being applicable in networks where distributed deterministic routing is present, like Infiniband [8]. In order to cover this type of networks, Flow-Based Implicit Congestion Management (FBICM) [4] has been recently put forward. FBICM effectively eliminates HOL blocking following the same RECN philosophy, but in networks using distributed deterministic routing.

However, FBICM presents a drawback that must be solved, since it needs too much resources to store and propagate congestion information, thus making more difficult its real implementation. In order to solve this problem, we propose in this paper a new version of FBICM, which significantly reduces the required silicon area, complexity and cost, without losing efficiency. Specifically, a new, optimized memory organization model together with a new procedure for congestion propagation (more appropriated for reduced resource requirements) are proposed and evaluated. Moreover, we present new results about FBICM efficiency in networks topologies no yet analyzed. From these evaluation results, we conclude that FBICM achieves the desirable results for an efficient and feasible congestion control technique, applicable to interconnection networks with distributed deterministic routing.

The remainder of this paper is organized as follows. Section 2 shows an overview of the existing related work in the congestion management field. Next, in section 3 we summarize the basics of the FBICM mechanism, pointing out the issues that make difficult its implementation. The new version of FBICM is described in section 4. In Section 5, FBICM is compared in terms of performance and resource requirements to other previously proposed HOL blocking elimination techniques. Finally, in Section 6 some conclusions are drawn.

Chapter 2

Related Work

In recent decades, many strategies for controlling the congestion have been proposed, motivated by the seriousness of the congestion problem. The simplest ones are overdimensioning the network and/or dropping packets when congestion arises. However, none of them are appropriated for modern interconnection networks designs due, respectively, to the high cost and consumption of current network components and to the lossless nature of these networks.

Therefore, other more thorough techniques have been especially proposed for avoiding or eliminating congestion. For instance, proactive strategies [15] reserve network resources for each data transmission, thus requiring a traffic scheduling based on network status information which is not always available. On the other hand, reactive congestion management [14] notifies congestion to the sources contributing to its apparition, in order to cease or reduce the traffic injection from those sources. Regrettably, this approach may be not efficient due to the delay between congestion detection and notification.

Other strategies directly attack the main negative effect of congestion: the HOL blocking. In that sense, the most common approach to deal with HOL blocking is to have different queues at each switch port, in order to separately store packets belonging to different flows. This is the basic approach followed by several techniques like Virtual Output Queues (VOQs) [2], Dynamically Allocated Multiqueues (DAMQs) [13], congestion buffers [12], Destination-Based HOL Blocking Elimination (DBBM) [11], etc. Among others, the main differences between all the mentioned techniques are the required number of queues and the policy for mapping packets to these queues.

In general, traditional HOL blocking elimination techniques are either feasible or effective, but not feasible and effective at the same time. For instance, the use of VOQs at network level requires as many queues at each port as end-points in the network, being so an effective but not scalable tech-

nique. A variation of VOQ uses as many queues at each port as output ports in a switch [1](eliminating switch-wide HOL blocking). So, this technique is feasible, but it does not eliminate completely network-wide HOL blocking.

By contrast, RECN dynamically eliminates HOL blocking in an efficient and scalable way, separating congested and non-congested flows in different queues. Specifically, RECN adds a set of additional queues (set aside queues, SAQs) to the standard queues at switch ports. While standard queues will store non-congested packets, SAQs are dynamically allocated for storing packets passing through a specific congested point. These SAQs can be deallocated when congestion vanishes, so RECN uses these resources efficiently. RECN assumes the use of source deterministic routing, thereby addressing congested network points by means of explicit routes toward these points (the entire route is placed at packet headers before injection). Although the RECN mechanism has proved to be very efficient, it presents the obvious limitation of requiring the use of source deterministic routing.

In that sense, FBICM has been recently proposed for networks which use distributed deterministic routing, achieving the same performance level as RECN. In particular, FBICM detects whether some packet is addressed to some congested point only by means of its destination, which is placed into packet header. Then, as RECN, FBICM stores congested packets in special queues, in order to separate congested packets from non-congested ones, thus eliminating the HOL blocking. As we mentioned above, FBICM presents difficulties, regarding its application in real systems, since it requires too many control memory resources, thus its implementation being too complex and expensive.

In the next section we describe the basics of FBICM: congestion detection philosophy, congested flows isolation, the way of storing congestion information, and the drawback which should be solved by the new version. Later, in section 4 we present the new, cost-effective FBICM version.

Chapter 3

FBICM Basics

The first version of FBICM [4], is an early approach to a congestion management technique that efficiently eliminates HOL blocking in distributed deterministic routing networks.

Specifically, FBICM has been proposed for networks using table-based routing, thus the routing logic at each switch is based on a routing table, indicating the output port for each incoming packet. We assume routing tables are filled during network setup according to some routing algorithm. Furthermore, packets are forwarded inside a switch depending on the packet destination placed at their header.

On the other hand, FBICM has been developed for Input Queued (IQ) switches, where queues are only present at input ports. This kind of switches are very popular, and they are cheaper than Combined Input and Output Queued (CIOQ) switches, the former requiring less memory and resources for operating than the latter[1].

Next, we focus on how the first version of FBICM detects the congestion, separates congested and non-congested flows, stores congestion information and deallocates resources.

3.1 Congested Points Identification

In Fig.3.1 we can see an example to show how FBICM identifies congested points. We assume a 8-port IQ switch, storing congested packets (in red) and non-congested packets (in green). In this figure, a contention situation is created due to packets at the head of the queues in input port P1 and input port P2, requesting the same output port (P5). Note that there are also packets that will request P5, in the queues at P3 and P4, thus their will

[1]Note that the latest RECN version [10] was also designed for IQ switches.

contribute to contention sooner or later. If "red" flows persist in time, input buffers may collapse, appearing congestion and its main negative effect: the HOL-blocking (red packets will slow the advance of green ones).

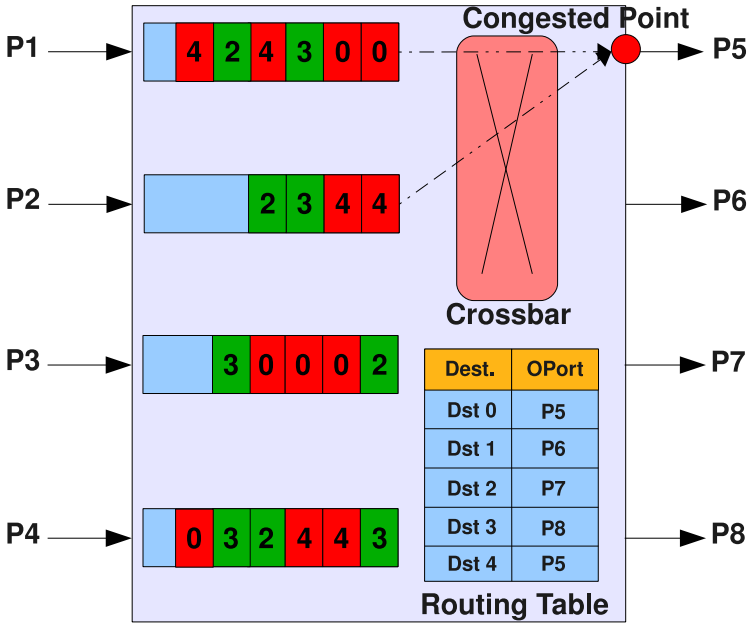


Figure 3.1: FBICM Detection Philosophy Diagram.

In order to deal with congestion FBICM requires both a mechanism to detect congested points and an identification (addressing) criterion, in order to keep track of all the congested points. In that sense, FBICM detects congestion at switch input queues using a detection threshold. When the number of packets stored in some input queue exceeds this threshold, a new congested point is located (in Fig.3.1, we may consider the congestion threshold is exceeded in P1 and P4). Moreover, we assume it is likely the packet placed at the top of the queue where detection occurs, is part of a flow which contributes to the congestion situation. The detection method infers this packet is delaying the normal progress of packets addressed to their different requested output port. Thus, it is very probably the requested output port of the top packet will be congested, being the "root" of a growing "congestion tree". On the other hand, it is possible that the packet at top of the queue where detection occurs is not the responsible of this congestion situation, but this failure is solved by means of post-processing mechanism, which is afterwards explained.

Regarding identification, a detected congested point could be easily iden-

tified inside its switch by its port number. However, any congested point must be communicated to other switches, in order to deal with all the congested packets along their paths. As the only routing information available is packet destinations, FBICM addresses a congested point by means of the destinations of packets crossing that point. In most cases, there will be more than one destination addressed to the same congested point, so FBICM includes all "congested destinations" in a list. By means of this list, FBICM uniquely identifies that congested point. For instance, congested point in Fig.3.1 would be identified as the port crossed by packets with destinations 0 and 4.

Summing up, once a congestion point a is detected, FBICM builds a list of congested destinations of packets which will cross that congested point. This list requires some memory structures to be stored (see section 3.4), and it will be used for detecting whether some packet belongs to any "congested flow", then separating it from non-congested ones, as we explain in the next section.

3.2 Congested Flows Isolation

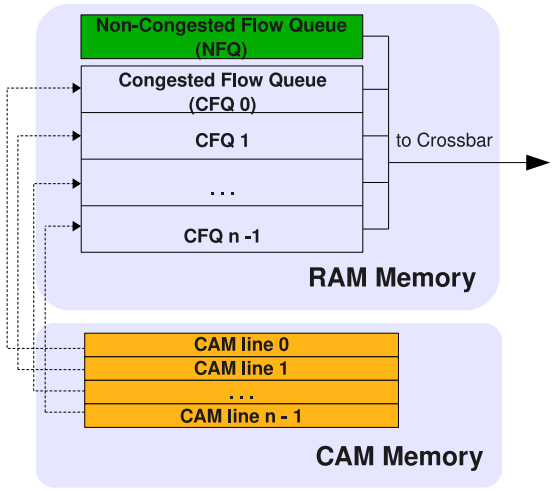


Figure 3.2: Input Port Memory Organization.

When a congestion situation is detected, it begins the process of network congested packets isolation, in order to guarantee the elimination of HOL blocking. Fig.3.2 shows a switch input port memory organization where two basic elements are present: RAM Memory and Content Addressable

Memory (CAM). The RAM memory is divided in two types of queues: one Non-Congested Flow Queue (NFQ), for storing non-congested packets, and a set of Congested Flow Queues (CFQs), for storing congested packets. On the other hand, a CAM memory is present in each input (and output) port, being used for storing congestion and CFQ status information (one CAM line per CFQ is needed). The CAM organization and functionality are described in next section.

If a congestion detection happens, FBICM dynamically allocates a new CFQ, together with a CAM line including information about packets which are contributing to create a congestion situation. In particular, the CAM line contains a list with all the destinations of packets which will cross a congested point. Then, all the packets belonging to a specific congested flow will be stored in the assigned CFQ, not delaying the advance of non-congested ones. As we explain later, if congestion situation persists in time CFQs may become filled and congestion information will have to be propagated to upstream switches, where it is necessary information for isolate the congested flows.

Once congestion has been detected and a CFQ+CAM have been allocated, any packet responsible of the congestion situation must be moved from the NFQ to the corresponding CFQ. This action is performed by the packet processing mechanism (post-process) which detects if some packet belongs to some flow involved in a congestion situation. Basically, when a packet reaches the top of the NFQ, its destination is checked, in order to compare it to all the destinations included in the lists that identify congested points. In the case of matching, the packet belongs to a congested flow. If congested, the packet is moved to the CFQ and, otherwise, the packet will cross to the requested output port. Note, the post-process mechanism leaves in the top of the NFQ only non-congested packets, thus avoiding the HOL blocking problem. Moreover, this mechanism decides what input port queue (NFQ or active CFQs) can request an arbitration, which will forward its top packet to the requested output port. This decision is took based on a first-come first-served policy, ever giving absolute crossing priority to NFQs. Finally, another important post-process feature, is the assurance of packet-in-order delivery.

3.3 Congestion Information Propagation and Flow Control

In the same way that FBICM sets a congestion detection threshold in the NFQ, a "Stop" threshold is used, when congestion persists for long, both for

avoiding CFQ overflow and for propagating congestion information. Specifically, when the occupancy of a CFQ exceeds the Stop threshold, a Stop notification containing the information of the associated CAM line is sent to the upstream switch output port. When an output port receives an external Stop notification, it checks if there is already an active CAM line containing the same received information. If not, it will allocate a new CAM line, filling it with that information (so, active output CAM lines will be exact copies of downstream CAM lines). On the contrary, if there were already an allocated CAM for the received information, the Stop notification is considered just as a flow control message. In both cases (new CAM line allocation or not), the involved CAM line will be set to Stop state, and as a consequence this output port will propagate the congestion information to any input port, sending congested packets to that output port. Therefore, a similar congestion propagation process is followed by FBICM inside a switch. If some input port receives a Stop notification, it will allocate a new CAM line (if there were not yet an allocated one), containing the received congestion information, together with a CFQ for storing packets belonging to the congested flow. From this input port, congestion propagation process is repeated, by using the Stop threshold, in the way detailed above.

Note that, in both cases (output and input ports) the information congestion is all transmitted in every Stop notification (because it is necessary in order to check upstream allocated CAM lines) and, even in flow control ones, thus consuming a high fraction of link bandwidth. As we explain later, the new version of FBICM optimizes the link use.

On the other hand, when the occupancy of a CFQ that sent a Stop notification decreases enough (until a given "Go" threshold), a Go notification is sent upstream, with the opposite effect than Stop ones. Thus, upon reception of a Go notification, an output port CAM line will be set in the Go state, thus unblocking the flow of packets. In the same way, inside of the switch Go notifications unblock all the input port CAM lines associated with output port ones. Therefore, by means of Stop/Go notifications, FBICM performs the flow control between CFQs.

Summing up, FBICM separates congested and non-congested flows in separate queues along any path followed by congested packets, thus isolating the congestion and reducing the HOL blocking apparition at maximum.

3.4 Congestion Information Storage

As we have described above, the congestion and CFQ status information is stored in a CAM memory, divided into as many number of CAM lines as

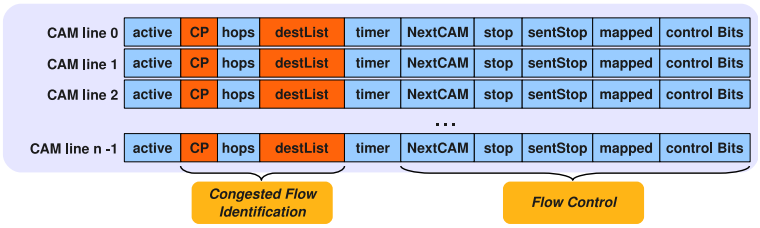


Figure 3.3: CAM organization Diagram.

CFQs in the input port memory. A detailed schema of the CAM organization is shown in Fig.3.3. We can distinguish two important blocks of fields: Congestion Flow Identification and Flow Control fields. The later fields are used by the flow control mechanism between CFQs, while the former are used in order to identify a congested point. Moreover, there are other fields used for saving the CAM status, as the "active" field, which indicates whether some CAM is active or not, and the "timer" field, used in the resource deallocation process, explained later.

Specifically, in the Congested flow Identification block, "CP" stores the congested output port number (Px), while "hops" indicates the distance to that congested point, and "destList" field stores all the destinations that will cross the congested point. Note that, theoretically, this field should be dimensioned to include all the possible destinations assigned to any output port by the routing algorithm. By using this four fields FBICM is able to identify any congested point inside the network. "CP" and "destList" are both colored in red because the former is removed in the new version of FBICM, while the latter is reduced in order to make less complex the CAM memory.

Regarding Flow Control fields, the field "NextCAM" stores the downstream CAM line which the current CAM line is linked to. The remainder fields are used in order to implement the Stop/Go flow control mechanism between CFQs.

This CAM organization presents a problem related with the great amount of memory needed for storing the destination list. Specially, in high-sized networks, the required space for storing all the destinations contributing to congestion can be prohibitive, consequently making the real implementation of FBICM unfeasible. In that sense, we presented in [4] an idea in order to solve this problem. This solution removes the destination list from CAMs, moving it to the routing tables. In each row of the routing table some bits are added, to store what destination is congested and what CAM line or lines it belongs to (e.g. 44 bits per routing table entry are used for a 4x4 switch

and FBICM using 8-CFQs per port). However, this approach is, in fact, unfeasible because when network size increases, the table complexity increases as well. As an example, the routing tables in Infiniband [8] can address 48K destinations, then this method would require a vast amount of additional bits (e.g. 48K entries each of one containing 44 extra bits). Although we have thoroughly considered these two options, we have discarded both, since they increase the switch complexity.

Another important issue of FBICM that must be analyzed, is related to the number of destinations added to a CAM line at the moment of its allocation. When a congestion detection happens, FBICM stores in a CAM line all the destinations that will cross the detected congested point, extracting this information form the routing table. Although packets addressed to all those destinations could be received in the input port, it is very likely that only some of them arrive in the near future. Therefore, there exists a splurge of the resources dedicated to store destinations, which can be excessive in big networks.

Summing up, it is necessary using a non-speculative criterion to add destinations to the CAM lines, thus reducing the size of CAM memories. In section 4, we present a solution for this FBICM flaw, based on limiting the space used for storing destinations.

3.5 Resource Deallocation

Another important issue of FBICM is dynamic, distributed resource deallocation. "Dynamic" because a CFQ can be deallocated during network operation (Go status), and "distributed" because the CFQs can be deallocated based on local information, without external orders.

Specifically, if a CFQ becomes empty and CFQ status is non-blocked (Go), it will be deallocated releasing the resources used by CFQ and CAM. The upstream linked CFQ, will be informed by means of deallocation notifications and it will update the "NextCAM" field to null, "unhooking" this branch of the congestion tree.

The problem of resource deallocation is related with the "speculative" approach of adding destinations to the CAM lines, described above. In that sense, the deallocation of a CAM line containing too much destinations in its list will be more difficult, since it is very likely that its associated CFQ contains always some packet. This problem is bigger in larger networks where a CAM line could contain a lot of unnecessary destinations.

In the next section, we propose a new version of FBICM that solves the problem of adding destinations in an speculative way. Moreover, this new

proposal reduce the complexity of CAM memory, by reducing the number of destinations stored in any CAM line. Additionally, our new technique optimizes the way congestion information is propagated, thus achieving an efficient link utilization. As a consequence of these enhancements, we reduce switch complexity, keeping the excellent network performance when congestion arises.

Chapter 4

FBICM Enhancements

As we have pointed out in the previous section, FBICM requires a lot of resources in order to store and propagate the congestion information, since it uses a "speculative" approach for filling destination lists in the case of congestion detection. Moreover, this way of adding destinations to lists slows down the resource deallocation process, and delays the advance of packets not actually congested; this problem is bigger as greater is the size of the network.

In this section, we present a new version of FBICM that solves the problems related to this congestion information storage and propagation. Specifically, we propose a new, less-complex CAM organization that efficiently stores congestion information. Besides, we define a new congestion notification procedure which requires a few bits for transmitting the congestion information, thus reducing link use.

As a consequence of this enhancements, we achieve three important benefits:

- A cost-effective CAM organization, which reduces its size requirements.
- An optimal link use, as notifications are redefined in order to reduce their transmission time.
- A fast resource deallocation, since CAM destination lists contains less destinations, thus being easier to deallocate.

Our new proposal follows the same philosophy as the previous version for detecting congestion, since it maintains the congestion detection mechanism, as well as the post-process mechanism and Stop/Go thresholds; as explained in section 3. In the remainder of this section we describe the important changes we have introduced in FBICM, together with a reduced number of small improvements that contribute to make better our proposal.

4.1 Cost-Effective CAM organization

The new proposed CAM organization differs from the previous one basically in two fields: "CP" and "destList" (see Fig. 3.3). The former is removed because we only need two fields for identify a congested flow: "hops" to reach the congested point and "destList" only including destinations really contributing with congestion. Regarding destination list, we have fixed a low maximum size for that list in each CAM line. This important change, a priori insignificant, obligates to redefine the way destinations are added to the list, as only destinations contributing to congestion should be added. In this way, the required resources for storing congestion information are significantly reduced. Note that with this restriction, it would be possible that a CAM line does not have enough space to include all the destinations involved in a congestion situation. However, in these cases, packets addressed to destinations not included in the list will not be stored in CFQs, and will continue contributing to congestion, thus a new detection for those destinations will happen, allocating a new CFQ for them. In this way, congested flows may be "splitted" in different CFQs, but always controlled.

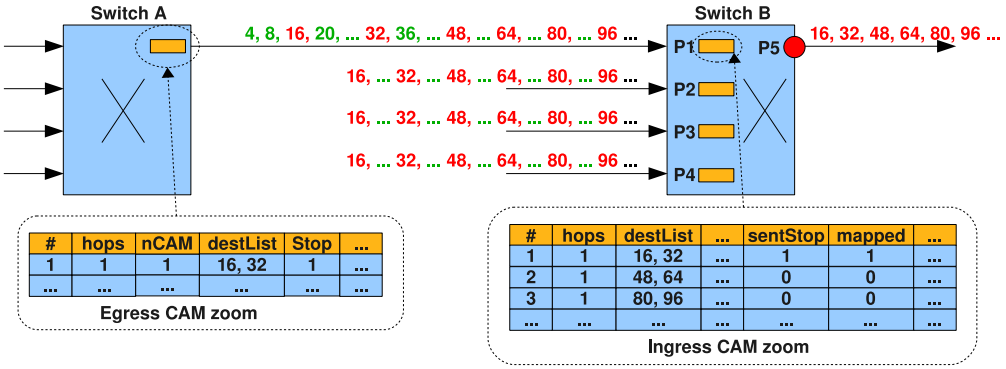


Figure 4.1: Example of the new FBICM congestion information storage.

In Fig. 4.1 we can see an example of the new FBICM operation. The Switch B receives packets from four switches which address their packets to the destinations colored in green and red. Packets addressed to red destinations are contributing with the congestion situation at P5, thus delaying the packets addressed to green colored destinations. We assume that the maximum size of "destList" is set to two destinations, therefore, we need more than one CAM line in order to store all the destinations of packets contributing to congestion. It is important to notice, that destinations lists are dynamically filled, thus storing only destinations of packets which are

actually creating congestion. In that sense, the post-process mechanism is in charge of, besides its described task, either add new destinations to a existing destination list or, if the destination list becomes full, allocating new CAM lines (together with its corresponding CFQs) for storing other congested packet destinations.

On the other side, along with this new CAM organization, we have included another simple improvement to FBICM. When none CAM line is allocated, it is not necessary to postprocess any packet, so it does not make sense that the corresponding CAM line silicon area are wasting power. In that sense, CAM memories are provided with a bit in order to activate/deactivate the memory, when congestion arises/vanishes in the network. Therefore, with this simple "CAM activation bit" FBICM reduces the power consumption and operates in a simpler and more efficient way.

Summing up, due to these features, only the information really related to the congestion are added into the CAM lines, thus improving the previous speculative way of storing congestion information. Moreover, CAM lines will be deactivated when congestion are not present in the network, thus reducing FBICM power consumption. As a result, we consider that new CAM organization is cost-effective, since it uses few resources for storing congestion information, thus improving the previous FBICM version.

4.2 Congestion Propagation Optimization

In the new FBICM version we propose a new notifications scheme which reduces link bandwidth use, since it fits the new CAM organization which "splits" the entire congestion tree branches into some parts, each of them stored independently.

Specifically, the new version of FBICM defines this types of notifications:

- Allocation: it is sent when a CFQ exceeds the Stop threshold in order to allocate a new CAM line in the upstream switch. It contains information to identify the congested point and it is sent when congestion information must be propagated.

- Update: At the moment of sending an allocation it is possible that not all the packet destinations contributing to congestion are present in the CAM line. Therefore, update notifications are sent to propagate new destinations participating in the congestion.

- Stop/Go: They are only used for flow control between CFQs, only including information about the "NextCAM" field (see Fig.3.3) which is enough for indicating the CAM line which must be blocked/unblocked.

- Deallocation: As flow control notifications, it only includes the "NextCAM" field. This notifications are used for resource deallocation, as we explain later.

In Fig.4.1, some ingress CAM lines (and their corresponding CFQs) have been allocated in P1 in order to identify a single congested point. Therefore, only packets addressed to current destinations-in-CAM are stored in the corresponding CFQs. When the number of packets of some CFQ exceeds the Stop threshold an "Allocate" notification, containing the information for identifying the congested point ("hops" and "destList") along with "NextCAM" field, is sent to the upstream switch (Switch "A" in the Fig.4.1). The congested point identification information, is completely sent only in "Allocate" notifications (thus never in other notifications), thus saving link bandwidth occupancy. When some "Allocate" is sent and accepted at the upstream switch, the "sentStop" and "mapped" bits are set to true.

If a new destination is added to the CAM line #1 of the P1 when the "mapped" bit is activated, an "Update" notification, containing this new destination together with "NextCAM" field value, is sent to the the upstream CAM line, since congestion information must be updated. The "Update" notifications automatically propagates the new congested destination throughout the congestion tree, so adding such destination to all the linked CAM lines.

On the other hand, the flow control between CFQs is performed by the Stop/Go notifications, in the same way as in the first FBICM version, but using less space for storing the required information. Specifically, Stop/Go notifications only needs the "NextCAM" field value, for detecting which CAM line are going to be blocked/unblocked. Finally, the "deallocation" notifications are sent when some CFQ becomes empty, and whether its associated CAM line fulfils the deallocation conditions; this deallocation process is described in next section.

In conclusion, with the new notifications scheme FBICM uses smaller notifications, thus reducing the use of link bandwidth.

4.3 Faster Resource Deallocation

As in the previous version of FBICM, the dynamic and distributed resource deallocation process begins when a CFQ satisfy some conditions: do not contain any packet, and its associated CAM line is in Go status. If some CAM line is deallocated, and it has the "mapped" bit set to true, a "Deallocation" notification is sent to upstream switch (or input port linked CAM line) in-

forming about the new situation. The process continues as CFQs become empty, in a distributed and dynamic way.

Notice that, in the new version of FBICM, if congestion are vanishing, it is more likely a CFQ becomes empty faster than in previous version of FBICM. In particular, the destination lists of our new proposal contains less destinations than the ones in the first FBICM version. Therefore, the number of stored packets is smaller, easing the CFQ+CAM deallocation. Moreover, in the new FBICM version the arbiter gives more priority for crossing toward the output ports to packets belonging to "unlinked" CFQs ("NextCAM" bit set to null), because we assume this type of CFQs have higher probability of being deallocated, due to congestion is vanishing.

Summing up, a faster resource deallocation process is performed in the new version of FBICM, since less destinations are included in the CAM line and, therefore, it is less likely to find packets in the CFQ when congestion is vanishing.

Chapter 5

Evaluation

In this section we evaluate the new version of FBICM by comparing it to other HOL blocking elimination techniques, as Virtual Output Queues at network level (VOQNet), Virtual Output Queues at switch level (VOQSw) and Destination-Based HOL Blocking Elimination (DBBM). First, we describe the simulation tool and the different traffic patterns used in our experiments, as well as the network configurations where that experiments have taken place. Next, we present and analyze the obtained results in terms of normalized network throughput. Finally, we present a study of CAM resources required by both FBICM versions.

5.1 Simulation Model

The simulation tool used in our experiments is an ad-hoc, event-driven simulator modeling interconnection networks at cycle level. The simulator models different types of network topologies (e.g. *perfect shuffle*, *butterfly*, ...), by defining the number of switches, endnodes and links. In our experiments, we model fat-trees (k-ary n-trees) with different network sizes. In particular, we use the network configurations shown in Table 5.1.

#	# Network Size	Interconnection Pattern	#Switch radix	#Switches (total)	#Stages
#1	64×64	*4*-ary *3*-tree	8	48	3
#2	256×256	*4*-ary *4*-tree	8	256	4

Table 5.1: Evaluated network configurations

Regarding link model, we use the same link model for all network configurations. In particular, we assume serial full-duplex pipelined links with 1GByte/s of bandwidth and 1 nanoseconds for link delay, both for switch-to-switch links as node-to-switch links.

We assume table-based, distributed routing has been modeled, thus in these cases the routing algorithm has been used for filling the routing tables. For our experiments we have used the FIR-based deterministic routing algorithm described in [7].

Regarding message switching policy, we assume Virtual Cut-Through. Moreover, in all the switches, the flow control policy is a credit-based mechanism at the queue level. Packets are forwarded from input memories to output ports through a multiplexed crossbar, modeled with a speedup of 1 (link bandwidth is equal to crossbar bandwidth).

The modeled switch architecture follows the IQ scheme, thus RAM memories have been modeled at each input port of the switch, with different sizes depending on the simulated HOL blocking elimination technique. Specifically, the simulator models the following HOL blocking elimination techniques:

- FBICM. We use a memory of 8 KB per input port, which is divided in a 8 CFQs and 1 NFQ. Moreover, there are CAMs both at input and output ports. We have fixed the maximum number of destinations per destination list at 8.

- Single Queue. This is the simplest case, with only one queue at each input port for storing all the incoming packets. Hence, there is no HOL-blocking reduction policy at all, thus this scheme allows to evaluate the performance achieved by the "alone" FIR-based deterministic routing algorithm. 8 KB memories are used for this case.

- DBBM. As in the previous scheme, a memory of 8 KB per input port is assumed, statically and equally divided among all configured DBBM queues in the port. Each queue is assigned to a set of destinations according to the modulo-mapping function $assigned\_queue\_number = destination\ MOD\ number\_of\_queues$. We consider 8 queues per port for DBBM.

- VOQSw. 8 KB memories per input port are used, statically and equally divided into as many queues as switch output ports, in order to store each incoming packet in the queue corresponding to its requested output port. As the number of queues equals switch radix, 8 queues per input port are used.

- VOQNet. This scheme, although the most effective one, requires a greater memory size per port, because the memory must be splitted into as many queues as endnodes in the network, and each queue requires

a minimum size. Considering flow control restrictions, packet size, link bandwidth and link delay, we fix the minimum queue size to 256 bytes, which implies input port memories of 16 KB for the 64×64 networks and 64 KB for the 256×256 networks. This scheme is actually almost unfeasible, thus it is considered only for showing theoretical maximum efficiency in HOL-blocking elimination.

Notice that we have used the same amount of memory for FBICM, DBBM, VOQSw and Single Queue, but not for VOQNet which needs larger memories.

The endnodes are connected to switches through Input Adapters (IAs). Every IA is modeled with a fixed number of admittance queues (as many as destinations in the network, in order to avoid HOL-blocking before packet injection), and a variable number of injection queues, which follows the same selected scheme as queues at input port memories (both for FBICM technique and for DBBM, VOQSw, VOQNet and Single Queue schemes). A generated message is completely stored in a particular admittance queue assigned to its destination. Then, the stored message is packetized before being transferred to an injection queue. We use 64-byte packets.

Regarding traffic loads, we use both synthetic traffic patterns (in order to simulate ideal traffic scenarios) and storage area network (SAN) traces. The considered synthetic traffic patterns are shown in table 5.2. Three traffic patterns have been modeled for each network configuration:

- Completely uniform (random) destination distribution (cases #1 and #4 from table 5.2).

- Single end-node hot-spot (cases #2 and #5). A percentage of sources (25%) generate traffic addressed to a unique, hot-spot destination, thereby creating congestion.

- Multiple end-node hot-spot (cases #3 and #6). Similarly, a percentage of sources (25%) generate traffic addressed to a set of multiple destinations, thus creating congestion at internal points of the network. Particularly, this traffic pattern creates heavy congestion situations in intermediate points of networks (multiple destinations crossing a single hot-spot), being a corner-case for the new version of FBICM, but not for other techniques.

In all these patterns, packet generation rate is indicated as a relative percentage of link bandwidth. Note that, in case in cases #1 and #4, random generation rate is incremental, thus increasing the traffic rate from 0% up to

100% of link bandwidth. However, traffic patterns #2, #3, #5 and #6 have been used to obtain performance results as a function of time, in order to show the impact of a sudden congestion situation (which arises between the time instants 1000 μs and 1300 μs).

		Random Traffic			Hot-Spot Traffic				
Traffic case	Network (BMIN)	# Srcs	Dest	Generation rate	# Srcs	Dest	Generation rate	Start time	End time
# 1	64 × 64	100%	random	incremental	0%	-	-	-	-
# 2	64 × 64	75%	random	100%	25%	32	100%	1000 μs	1300 μs
# 3	64 × 64	75%	random	100%	25%	Multiple	100%	1000 μs	1300 μs
# 4	256 × 256	100%	random	incremental	0%	-	-	-	-
# 5	256 × 256	75%	random	100%	25%	123	100%	1000 μs	1300 μs
# 6	256 × 256	75%	random	100%	25%	Multiple	100%	1000 μs	1300 μs

Table 5.2: Synthetic traffic patterns used in the evaluation

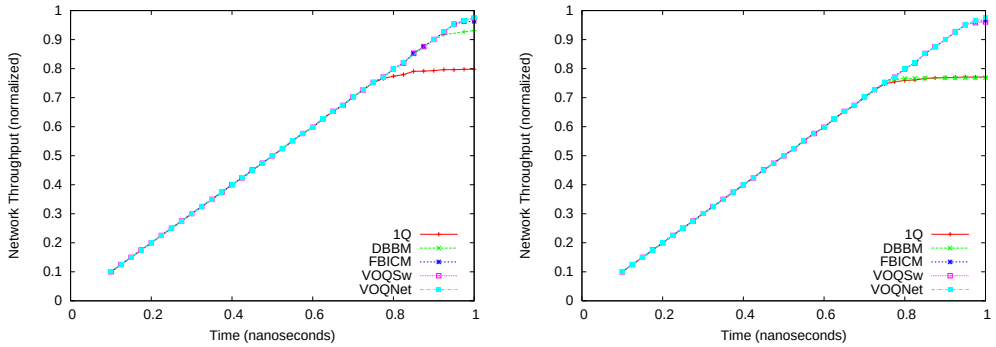
Regarding SAN traces, we use the ones provided by Hewlett-Packard Labs, and they include all the I/O activity generated from 1/14/1999 to 2/28/1999 at the disk interface of the *cello* system. As these traces are eleven years old, we apply several time compression factors to the traces. Of course, only results as a function of time are shown in this case. Traces are used for network configuration #1 (see table 5.1).

In all the experiments, the simulator offers different metrics but, for our evaluation, we have only considered network throughput (as a function of time or traffic load) as the main metric for measuring the performance of the networks when the different HOL blocking elimination techniques are used. Consequently, in the following subsections we analyze, by means of these metrics, the obtained network performance.

5.2 Results for Uniform Traffic

Figures 5.1a and 5.1b respectively show the normalized network throughput simulation results as a function of traffic load, for network configurations #1 and #2 (traffic patterns #1 and #4). As we can see, in both figures FBICM achieves the best performance, at the level of VOQNet and VOQSw. Due to the uniform traffic properties, along with routing algorithm benefits, the congestion situations are very short in time and they occur in many different network points, thus none of these techniques faces great difficulties for keeping network performance at a good level. DBBM achieves worse results as higher are both the network size and the traffic loads, because the number of destinations that must be mapped to its queues is greater. The Single Queue scheme achieves the poorest results in both sizes of networks since it

is affected by the apparition of the HOL blocking. Therefore, FBICM and VOQSw achieve similar results to VOQNet, but using less memory resources (VOQNet needs 16KB memories for 64×64 networks and 64KB memories for 256×256).



(a) Network Configuration #1. Traffic Pattern #1.

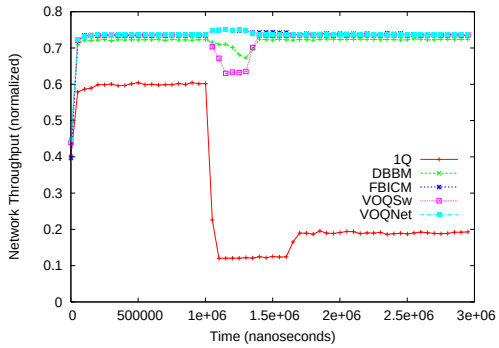
(b) Network Configuration #2. Traffic Pattern #4.

Figure 5.1: Network Efficiency versus Generated traffic. Uniform distribution of packet destinations.

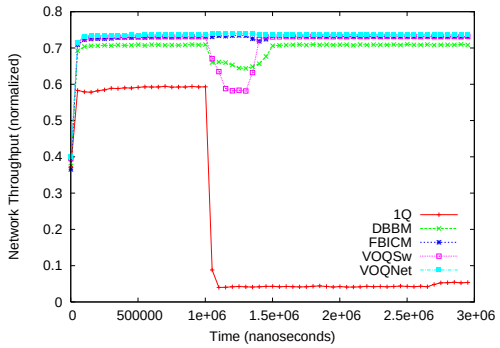
5.3 Results for Hot-Spot Traffics

When we use Hot-Spot traffic scenarios, where congestion suddenly arises, the obtained simulation results are quite different. The Figures 5.2a and 5.2b show throughput as a function of time, when the traffic pattern #2 and #5 (Single End-Node Hot-Spot) are generated in the network. In both cases, FBICM newly achieves the best performance, at level of VOQNet. By contrast, the throughput of VOQSw and DBBM (and of course Single Queue) decreases in the moment congestion arises. DBBM falls half (to 65% in both figures) with respect to VOQSw, since the former, with low traffic loads and equal number of queues, reacts better to congestion situations. The Single Queue (1Q) throughput is rather poor, and its recovery due to congestion is very slow.

The Figures 5.3a and 5.3b depict the simulation results when traffic patterns #3 and #6 (Multiple End-Node Hot-Spot) are used. Notice that, this traffic pattern has been used to generate hot spots which are crossed by multiple destinations, in order to check the new version of FBICM in the worst case for that technique (we have defined the maximum number of destinations per CAM line in 8). For that reason, other techniques as DBBM



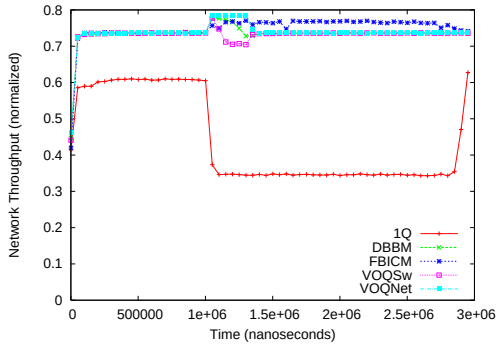
(a) Network Configuration #1. Traffic Pattern #2.



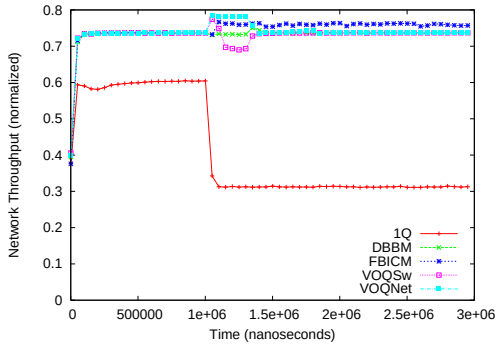
(b) Network Configuration #2. Traffic Pattern #5.

Figure 5.2: Network Efficiency versus Time. Single End-Node Hot-Spot.

and VOQSw are able to manage this traffic better than the previous traffic pattern. As we can see, FBICM achieves excellent results, even using this "corner-case" traffic, in all the network configurations, reaching the level of VOQNet, but using less resources.



(a) Network Configuration #1. Traffic Pattern #3.



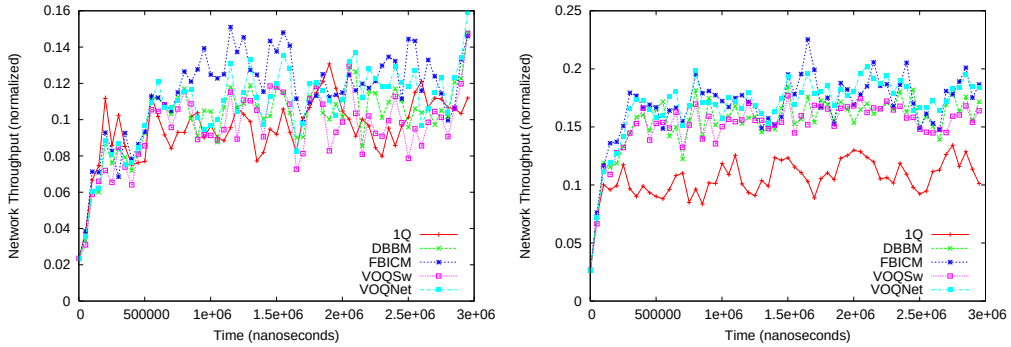
(b) Network Configuration #2. Traffic Pattern #6.

Figure 5.3: Network Efficiency versus Time. Multiple End-Node Hot-Spot.

5.4 Results for Real Traffic

Finally, we evaluate network performance when real traffic (I/O SAN traces) are used as traffic load, in the network configuration #1 of table 5.1. As can be seen in figures 5.4a and 5.4a FBICM achieves similar results as VOQNet, and slightly better than DBBM and VOQSw, regardless of the traces

compression factor. Therefore, FBICM also achieves more than acceptable results for real traffic.



(a) Network Configuration #1 (CF=20). (b) Network Configuration #1 (CF=40).

Figure 5.4: Network Efficiency versus Time. Real Traffic.

5.5 CAM Memory Requirements

#	Network Size	Previous FBICM version CAM size (bytes)	New FBICM version CAM size (bytes)
1	64×64	1.072	176
2	256×256	4.144	176
3	1024×1024	16.432	176

Table 5.3: Comparative between CAM size in previous and current FBICM versions.

Finally, in this section we present a comparison of the CAM memory requirements for both the first and new versions of FBICM. Table 5.3 shows the CAM space required by both versions for different network sizes. We assume that one CAM line without "destList" field needs 6 bytes (4 bytes for "timer" field plus 2 bytes for the other fields) of memory. Moreover, we consider 8 lines per CAM memory, thus 48 bytes are required in order to store all the information except the destinations lists, and 2 bytes are required for each destination, thus the first version of FBICM stores as many destinations per CAM line, as the end-nodes in the network. For instance, 64 destinations should be stored per CAM line in a 64×64 BMIN; therefore, 128 bytes per CAM line are required. As we assume 8 lines per CAM, 1024 bytes are required for storing all the destination lists, and 1072 bytes are

required for the entire CAM memory. For the new version of FBICM we assume 8 destinations per CAM line are stored regardless network size, thus reducing the space for storing destinations to 176 bytes.

As can be seen in table 5.3 only a few of bytes are required for storing the congestion information when the new FBICM version of FBICM is used.

Chapter 6

Conclusions

In the current interconnection networks designs solving the negative effects of congestion has became of vital importance. In that sense, many strategies for dealing with congestion have been proposed. Among the most successful techniques are RECN (used in source routing networks) and FBICM (used in deterministic distributed routing networks). Both techniques are based on dynamically allocating queues to separate congested and non-congested packets, thus eliminating HOL blocking.

The way the first version of FBICM identifies congested points triggers the need for a huge number of resources, in order to store and propagate the congestion information. In this paper, we have presented a new version of FBICM which reduces the number of required resources, without losing efficiency. We achieve it by assigning resources only when strictly required, in a non-speculative way.

From the evaluation results, we can conclude that the new version of FBICM exhibits excellent performance for distributed routing networks, reaching the level of VOQNet, but using less resources and performing other techniques which require a similar number of resources. FBICM achieves these performance independently on the traffic type (synthetic or real). In conclusion, the new version of FBICM is a cost-effective technique that efficiently eliminates HOL blocking, in distributed routing interconnection networks.

Bibliography

[1] T. Anderson, S. Owicki, J. Saxe, and C. Thacker. High-speed switch scheduling for local-area networks. *ACM Transactions on Computer Systems*, 11(4):319–352, November 1993.

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

[3] 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. In *Proceedings of the 11th Symposium on High Performance Computer Architecture (HPCA)*, 2005.

[4] J. Escudero-Sahuquillo, P. J. García, F. J. Quiles, J. Flich, and J. Duato. FBICM: Efficient congestion management for high-performance networks using distributed deterministic routing. In *HiPC*, pages 503–517, 2008.

[5] 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. *Lecture Notes in Computer Science (HiPEAC 2005)*, 3793:266–285, November 2005.

[6] P. J. García, J. Flich, J. Duato, I. Johnson, F. J. Quiles, and F. Naven. Efficient, scalable congestion management for interconnection networks. *IEEE Micro*, 26(5):52–66, September 2006.

[7] C. Gomez, F. Gilabert, M. Gomez, P. Lopez, and J. Duato. Deterministic versus adaptive routing in fat-trees. In *Workshop on Communication Architecture on Clusters, as a part of IPDPS'07*, page 235, March 2007.

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

[9] M. J. Karol, M. G. Hluchyj, and S. P. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Trans. on Commun.*, COM-35:1347–1356, 1987.

[10] G. Mora, P. J. García, J. Flich, and J. Duato. RECN-IQ: A cost-effective input-queued switch architecture with congestion management. In *Proceedings of 36th International Conference on Parallel Processing (ICPP'07)*, 2007.

[11] T. Nachiondo, J. Flich, and J. Duato. Destination-based hol blocking elimination. In *12th International Conference on Parallel and Distributed Systems (ICPADS 2006)*, pages 213–222, July.

[12] A. Smai and L. Thorelli. Global reactive congestion control in multicomputer networks. In *Proc. 5th Int. Conference on High Performance Computing*, 1998.

[13] Y. Tamir and G. Frazier. Dynamically-allocated multi-queue buffers for vlsi communication switches. *IEEE Transactions on Computers*, 41(6), June 1992.

[14] M. Thottetodi, A. Lebeck, and S. Mukherjee. Self-tuned congestion control for multiprocessor networks. In *Proc. of 7th. Int. Symp. on High Performance Computer Architecture*, February 2001.

[15] M. Wang, H. J. Siegel, M. A. Nichols, and S. Abraham. Using a multipath network for reducing the effects of hot spots. *IEEE Transactions on Parallel and Distributed Systems*, 6(3):252–268, March 1995.