

Combining Efficiency and Scalability on Congestion Management for Interconnection Networks

P. J. Garcia, J. Flich, J. Duato, I. Johnson, F. J. Quiles, and F. Naven

Abstract

Congestion has been an open issue in lossless interconnection networks for the last twenty years. In this environment, as packets can not be dropped, congestion may lead to a dramatic degradation of network performance. Although congestion was traditionally avoided by overdimensioning the network, this solution is no longer valid due to cost and power consumption concerns, that tend to reduce the size of the network in current systems. The restriction on network size makes the system working closer to the network saturation point, thus increasing congestion probability. Therefore, an effective congestion management mechanism is essential in order to avoid network performance degradation.

However, except the recently proposed RECN mechanism, none of the congestion management strategies proposed in the literature had been able to achieve, at the same time, the required efficiency and scalability levels demanded by emerging systems. In this paper we analyze the difficulties in handling congestion on lossless interconnection networks, identifying the key points for the design of an efficient and scalable congestion management technique. We also describe how these ideas have been implemented in the RECN design. Moreover, we also show that RECN is fully compatible with the Advanced Switching standard, allowing efficient and scalable congestion management in real systems.

Keywords:

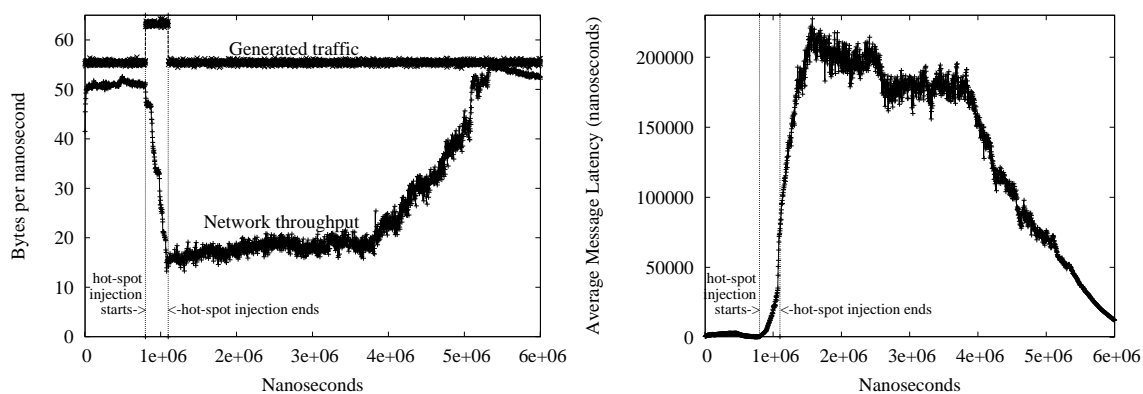
Interconnection Networks, Switch Architecture, Congestion, HOL Blocking.

1 Congestion in Lossless Interconnection Networks

Congestion in interconnection networks is a widely known phenomenon. The origin of congestion is network contention, that happens when different packets request the same resource (typically the output port).

In these cases, the access will be granted to only one packet, while the rest must wait. When contention persists over time, the buffers containing the blocked packets will be filled and the flow control, in lossless networks, will prevent other switches from sending packets to the congested ports. Although flow control is essential, it will rapidly propagate congestion to other switches, as packets stored on some of their ports will be also blocked. In this way, congestion may progressively spread through the network, forming “congestion trees” whose “branches” will match the paths followed by data flows contributing to congestion (“hot flows”).

Although congestion trees may actually evolve in different ways (see sidebar “Dynamics of Congestion Trees”), all of them degrade severely network performance. Packets belonging to hot flows prevent other packets from advancing, even if they belong to non-congested flows (“cold flows”). This phenomenon is known as Head-Of-Line (HOL) blocking, and it happens when a packet ahead of a FIFO queue blocks, preventing the rest of packets in the same queue from advancing. Of course, the wider the congestion tree, the greater the probability of HOL blocking caused by congested packets.



(a) Throughput and Generated Traffic

(b) Average Latency

Figure 1: Network performance degrades after congestion appearance

The impact of HOL blocking on network performance can be very serious: specifically, network throughput may degrade and packet latency may increase. For instance, Figure 1(a) shows the generated traffic and network throughput in a 64×64 bidirectional multistage interconnection network (BMIN). Congestion has been generated by the injection of traffic (at the maximum injection rate) from 8 sources to the same des-

mination (hot-spot traffic) during a small time interval (300 microseconds). The rest of endnodes (56) send, all the time, traffic addressed to random destinations. Thus, a congestion tree is created whose root is at the hot-spot destination. Note that network throughput not only severely degrades (by a 70%, approximately) when congestion appears, but it also keeps low beyond the end of the hot-spot traffic generation. Of course, this network throughput degradation is due to the HOL blocking produced by the congestion tree to other (non-congested) packets.

On the other hand, Figure 1(b) shows the average packet latency during the same congested situation. As it can be observed, this latency increases from few nanoseconds (before congestion) to several hundreds of microseconds (five orders of magnitude). Note also that this latency returns to the low values, but only after several milliseconds. This also confirms that packets experience massive HOL blocking during and after the hot-spot injection.

Although the negative consequences of congested situations have been always clear, network congestion has not been considered a problem until very recently. This is because, for many years, it was usual to overdimension the interconnection network, using much more network components than the minimum required for connecting all the system endnodes. By doing this, the network is working far below the network saturation point due to the low link utilization, and so the probability of congestion is also low.

Thus, the question is: Why should we care for congestion now? The answer is that, unfortunately, overdimensioned networks are becoming unappropriate due to cost and power consumption constraints. Current interconnect components are very expensive when compared to processors. Thus, the network represents a high percentage of the total system cost. On the other hand, as VLSI technology advances and link speed increases, interconnects are consuming an increasing fraction of the total system power [21]. To make this even more critical, current high-speed links require continuous pulse transmission in order to keep both ends synchronized, even when no data are being transmitted. As a consequence, link power consumption is almost independent of link utilization.

Although frequency/voltage scaling techniques [21] have been proposed in order to reduce power consumption, the efficiency of these techniques is not satisfactory at all due to their slow response, and anyway they do not solve the cost problem. Taking all this into account, it is nowadays advisable to reduce network size for reducing system cost and power consumption. Obviously, if the system computational power

Dynamics of Congestion Trees

The traditional view of congestion trees was based on some assumptions about their dynamics that have become obsolete for current commercial switch architectures. For instance, it was a popular belief that congestion appears always at switch output ports. However, this idea is a reminiscence of the days when switches had only queues at output ports (OQ switches), and it is not valid for the switches commonly used nowadays, that have queues at input ports (IQ switches) or both at input and output ports (CIOQ switches). In fact, in [11] it is shown that, in modern switches, congestion may appear on input or output ports, depending on the number of flows causing congestion and on the internal switch speedup (maximum speed at which packets can be forwarded to the outputs ports, relative to link speed).

Another “classic” assumption was that congestion trees always grow from root to leaves. Again, in [11] this idea is refuted, showing that, in modern networks, and depending on traffic patterns and switch architecture, congestion trees may evolve in several ways, exhibiting very complex dynamics. For example, a tree may grow from leaves to root and vice versa. Also, several congestion trees may grow independently and later merge, and it is even possible that some trees completely overlap while being independent. This complex and varied evolution of congestion trees should be considered on the design of any congestion management technique in order to handle congestion situations in an efficient way.

must be kept, the only way for reducing network size is to increase the number of endnodes attached to each switch or using a more suitable topology. However, any of these solutions will lead to a higher link utilization, and so the network working close to the saturation point, thus increasing congestion probability. To sum up, congestion situations will be more common in current networks. Therefore, some specific mechanism must be implemented for managing congestion, otherwise network performance will suffer the corresponding degradation.

2 Different Approaches to Congestion Management

The problems associated to congestion are more difficult to solve in lossless networks than in lossy networks (like Internet), as packets can not be dropped when congestion occurs. However, many techniques have been proposed. Basically, there are two approaches to address congestion in lossless networks. The first approach (see sidebar “Classical Congestion Elimination/Reduction Techniques”) tries to eliminate the congestion, either by acting as soon as congestion is detected (reactive techniques) or by preventing its appearance (proactive techniques). However, in general, those techniques are not able to guarantee an effective elimination of congestion. On the one hand, proactive congestion management requires a knowledge of the network status that is not always available. Moreover, the larger the network, the greater the difficulty in knowing network status and the more conservative the mechanism would behave. On the other hand, reactive techniques do not scale because the delay between congestion detection and reaction increases with network size, increasing the response time of the mechanism. Moreover, congested packets injected during that time can make congestion worse, even leading to oscillations at sources (by the time actions are taken the congestion has vanished or vice versa). Therefore, those techniques do not scale either with network size nor with link bandwidth. As most current networks use high speed links, this last drawback can not be ignored. Note that both proactive and reactive techniques present scalability problems.

On the contrary, the second (and more recent) congestion management approach considers that congestion is not a problem by itself. Instead, techniques that follow this approach try to eliminate the negative effects of congestion, mainly HOL blocking, rather than eliminating the congestion trees. The key idea is that, if HOL blocking is completely eliminated, congestion may exist while being harmless. Note that HOL blocking elimination implies that congested flows would not affect the forwarding of non-congested packets, that would cross the network normally. If this is possible, the only packets delayed by congestion would be the ones contributing to congestion. As this delay is unavoidable, network performance would experiment a minimum degradation in congested situations. So, the question now is: Is it possible to effectively eliminate HOL blocking? The answer is in the following Section.

Classical Congestion Elimination Techniques

Although several taxonomies have been proposed [9, 27], there are basically two classes of strategies that focus on eliminating congestion: proactive and reactive congestion management.

- **Proactive techniques** are based on controlling each data transmission in such a way that congestion should never happen. This is achieved either by reserving in advance network resources (avoidance-based techniques) [28, 5] or by limiting the routes followed by packets (prevention-based techniques) [12, 1, 20]. In general, the use of both techniques implies a planification of transmissions that requires information about the occupancy of network resources (buffers, links, etc.). Actually, due to their conservative behavior, proactive techniques are suitable for providing quality of service support (QoS), but not for congestion management.
- **Reactive techniques** are based on detecting the formation of a congested situation, activating later some mechanisms for eliminating the congestion. Most of these mechanisms [25, 14, 8, 17, 3, 4] consist on notifying congestion to the endnodes in order to cease or reduce the injection of packets. Note that these notifications must travel from the point where congestion is detected to the sources. Regarding congestion detection, it is usually based on locally measuring the occupancy of network resources [26, 16, 20]. The main drawback of these techniques is their lack of scalability since the reaction time is directly proportional to the distance of the congestion to the sources.

3 HOL Blocking Elimination Strategies

A complete HOL blocking elimination is possible, but it is not easy to achieve. In fact, along many years, several techniques have been proposed with this aim without being fully satisfactory until very recently.

In general, the most effective HOL blocking elimination techniques are based on storing in separate

queues packets belonging to different flows. This basic idea has been implemented in very different ways. So, some techniques [6, 13] distinguish data flows depending on their final destination (HOL blocking elimination at network level), and use this criterion for mapping packets to queues. For instance, if Virtual Output Queues (VOQs) are used at network level (VOQnet), there will be at each port as many queues as endpoints in the network, and every incoming packet will be stored in the queue assigned to its destination. Network-level HOL blocking elimination techniques are in general very efficient, but they require a lot of resources (queues) for networks with many endpoints. So, they are not scalable (their implementation on large networks is very expensive in terms of silicon area).

Other techniques [2, 24, 22, 7, 15] try to solve this problem by eliminating HOL blocking at switch level. In this case, both resource allocation and packet storing in a switch port depend on the switch output port requested by the packet. For instance, if VOQs at switch level (VOQsw) are used, there will be at each switch port as many queues as output ports in the switch, and a packet will be stored in the queue assigned to its switch output port. Therefore, the number of queues at each port depends on the number of switch ports, but not on the number of network endpoints. Figure 2 shows the number of queues per switch (note logarithmic scale) required for implementing VOQnet and VOQsw on networks with different size (64, 512 and 2048 endnodes) built with switches with different radices (4, 8 and 16). As it can be seen, the VOQnet queue requirements grow with the number of ports and specially with network size, but VOQsw ones just grow with the number of ports. Note that queue requirements for both techniques differ by approximately two orders of magnitude for medium or large networks, and this difference tends to grow. In fact, the excessive number of queues required by VOQnet for medium or large networks makes the VOQnet scheme practically unfeasible.

Although VOQsw requirements are constant with network size, they do not eliminate completely HOL blocking (as it is possible that congested and non-congested packets are stored in the same queue). Figure 3 shows the network throughput obtained when using respectively VOQnet, VOQsw and just one queue per input port, for the same traffic and network scenario assumed in Figure 1. Note that, although VOQsw alleviates congestion by reducing HOL blocking, it still has some network throughput degradation (by a 20%, approximately) and exhibits a considerable packet latency (around 50 microseconds). This confirms that VOQsw eliminates HOL blocking just partially. On the contrary, VOQnet is able to keep network

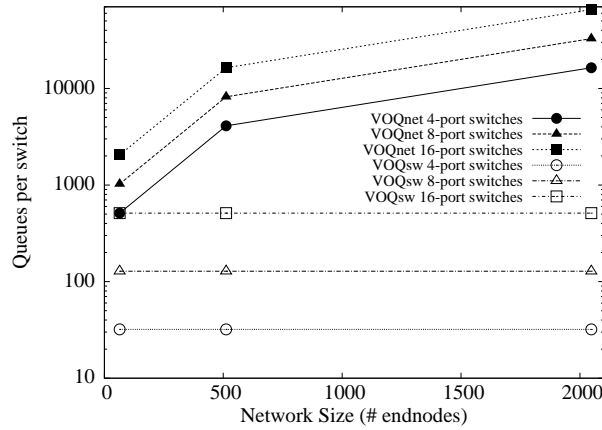


Figure 2: Number of queues required per switch assuming VOQnet and VOQsw

throughput at maximum and packet latency at minimum before, during, and after the hot-spot injection.

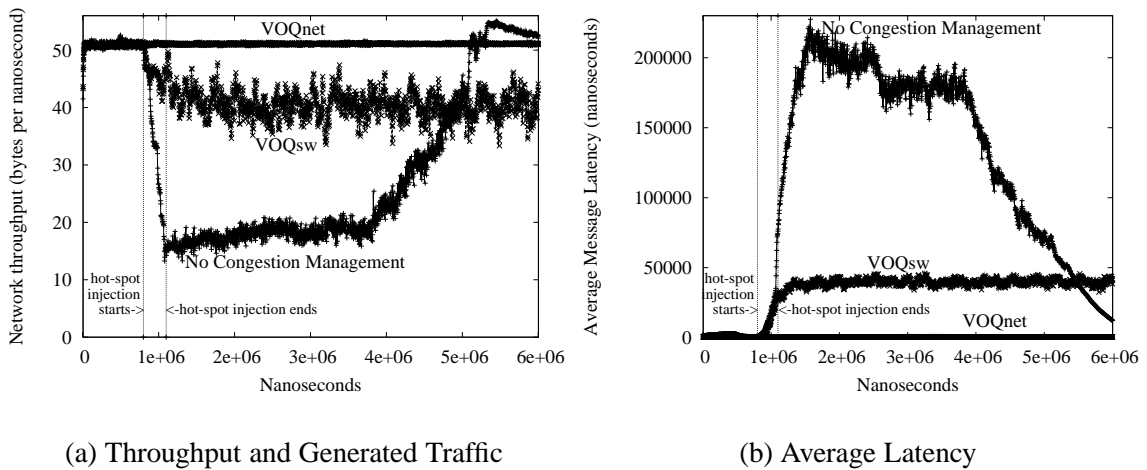


Figure 3: Network performance degrades on a congestion situation if Virtual Output Queues at switch level are used, but not if they are used at network level

To sum up, the formerly explained techniques are efficient or scalable (in resources), but none of them is efficient and scalable at the same time. In fact, for many years, it seemed that it was not possible to obtain a technique with such characteristics. This situation changed recently, when a fully efficient and scalable HOL blocking elimination technique (REC�, Regional Explicit Congestion Notification [10]) was proposed.

In order to analyze RECN performance, Figure 4 shows the network throughput obtained when using VOQnet, VOQsw and RECN, this time for two BMINs of different size (64×64 and 512×512) for different traffic generation rates. Again, congestion is generated in each simulated point by the temporary injection of packets from a percentage (12.5 %) of sources to hot-spot destinations (1 hot-spot destination for the small network, 4 hot-spots for the larger one). Note that, in both cases, results for RECN and VOQnet are very similar (and far better than VOQsw results). This means that HOL blocking is almost completely eliminated by RECN regardless of the traffic rate. Indeed, both RECN and VOQnet allow maximum injection rates without throughput degradation. However, RECN results for both networks have been obtained by using the same number of queues per switch port while the number of queues required by VOQnet (64 and 512 queues per switch port, respectively) increases with network size. Therefore, RECN eliminates efficiently HOL blocking while requiring an affordable and constant number of resources.

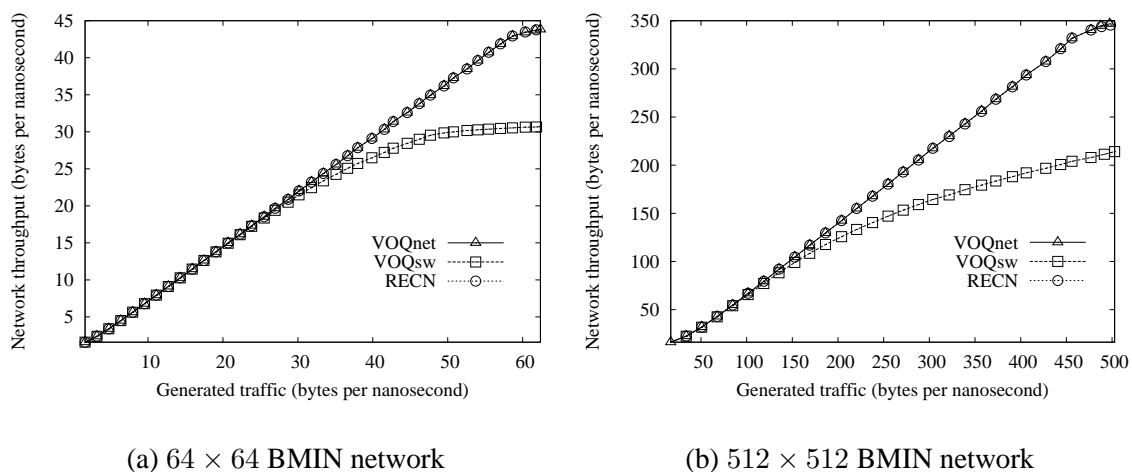


Figure 4: Network throughput when VOQnet, VOQsw and RECN are used on BMINs of different size

So, the next question to answer is: What makes the difference between RECN and other HOL blocking elimination techniques?. The first key difference is that, in order to store separately congested flows, RECN allocates and deallocates queues dynamically, on the contrary to former techniques, that just allocate queues in a static way. This allows to use queues only if it is really necessary. Another great difference is that RECN can address congestion to any network point, not just endpoints. This allows to accurately identify congested flows and non-congested ones, so they can be separated. The last main difference is that RECN

considers that packets belonging to non-congested flows can be stored in the same queue without producing significant HOL blocking, even if they follow different routes. This allows a great reduction on the number of queues required by the mechanism.

Of course, the implementation of these basic ideas of RECN is not obvious. The following section explains the implementation details of RECN.

4 RECN Implementation

RECN is implemented both at input and output ports. Figures 6 and 7 show, for both sides, the main structures and events involved in the basic RECN procedure. They all will be mentioned on the following sections, where the RECN mechanism is described.

4.1 Routing Requirements

Like other HOL blocking elimination techniques, RECN is based on storing in separate queues the packets belonging to congestion trees. In detail, RECN detects and notifies the position of the roots of the different congestion trees present in the network, and later checks at any notified port whether an incoming packet will pass through one of these roots. If so, then the packet belongs to a congestion tree and it must be stored at this port on the corresponding separate queue. Therefore, RECN requires that packet routes could be inspected in advance at any port.

On the other hand, in order to achieve the maximum accuracy and efficiency, RECN must detect congestion also within the network (not only at endpoints). Therefore, a method is required for identifying any network port.

The use of source deterministic routing helps in meeting both requirements. Specifically, RECN has been designed assuming the routing properties of Advanced Switching (AS) [19]¹. Packet headers in AS contain a 31-bit field that encodes all the turns (shift from a switch input port to an output port) that a packet must take along its route. This field is known as turnpool. The header also includes a pointer (turn pointer) that indicates the next turn. This routing mechanism allows to address the root of a congestion tree from different ports within the network. Simply, different sequences of turns will address the root from each

¹Anyway, RECN could be easily adapted to any other interconnection technology that allows source routing, like Myrinet [18].

specific port. Moreover, the routing mechanism also allows to know in advance if a packet in a port will pass through a congestion tree root simply by inspecting a sequence of turns from its turnpool.

Figure 5 shows an example where several packets travel across a network where a congestion tree exist (the root is placed at output port *P5* of switch 2). As it can be seen, the root can be identified at any other port by a relative route (some consecutive turns) from the port, and this allows to detect in advance which packets will (packets A,B and D) and which packets will not (packet C).

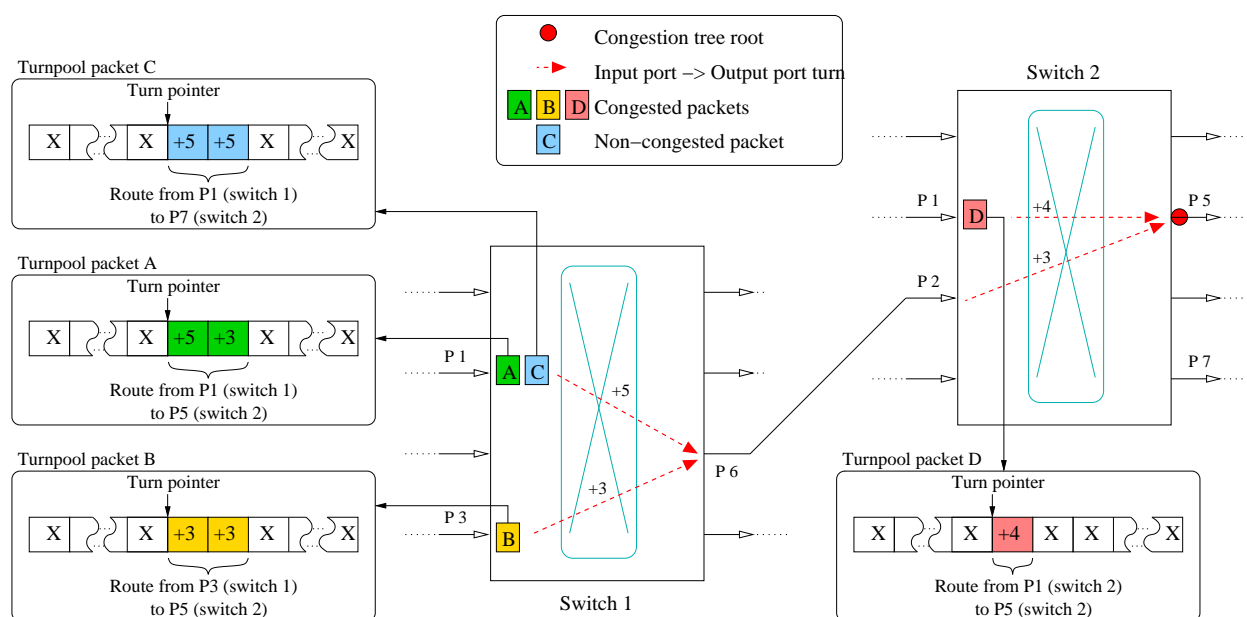


Figure 5: Turnpool inspection allows to detect in advance congested packets

As Advanced Switching specifications allow turnpools with turns of different lengths (in order to route packets through switches with different number of ports), it is necessary to exactly know how many bits of the packet turnpool must be checked at each port. RECN solves this problem by encoding the route to each root by means of a complete turnpool and a bit mask that selects which bits must be compared.

4.2 Required Resources

RECN requires some additional resources both at input (Figure 6) and output (Figure 7) switch ports. In particular, besides the standard queue, RECN uses at each port a set of additional queues, referred to as SAQs

(Set Aside Queues). A SAQ will be dynamically allocated to store packets belonging to a specific congestion tree once the existence of this tree is detected at (or notified to) the port. Upon congestion vanishment, the corresponding SAQ will be deallocated, and thus it will be available to store packets belonging to other congestion trees. On the other hand, packets belonging to non-congested flows will be stored in the standard queue at each switch port. Thus, the data RAM at each port is shared by the normal queue and by the set of SAQs.

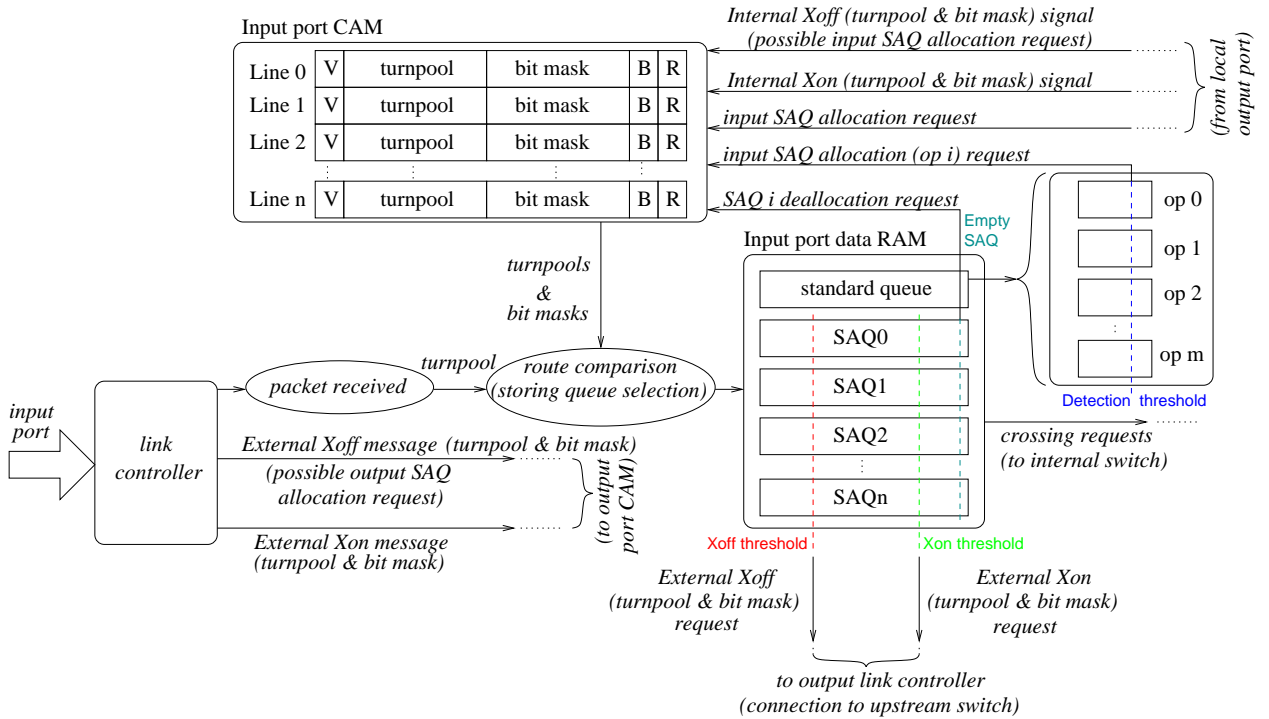


Figure 6: RECN mechanism at the input port of a switch

Although the number of SAQs per port could be varied, the dynamic allocation and deallocation allows that a small number of SAQs is enough for handling efficiently congested situations. Moreover, this number is independent of network size. For instance, it has been shown in figure 4 that 8 SAQs per port are enough for completely eliminating HOL blocking on 64×64 and 512×512 BMINs. In fact, in [11] it is shown that RECN is able to eliminate HOL blocking in a 2048×2048 BMIN network network by just using 8 SAQs per port. Therefore, the RECN mechanism is fully scalable.

A comparison among the number of queues per switch required for implementing VOQnet, VOQsw and

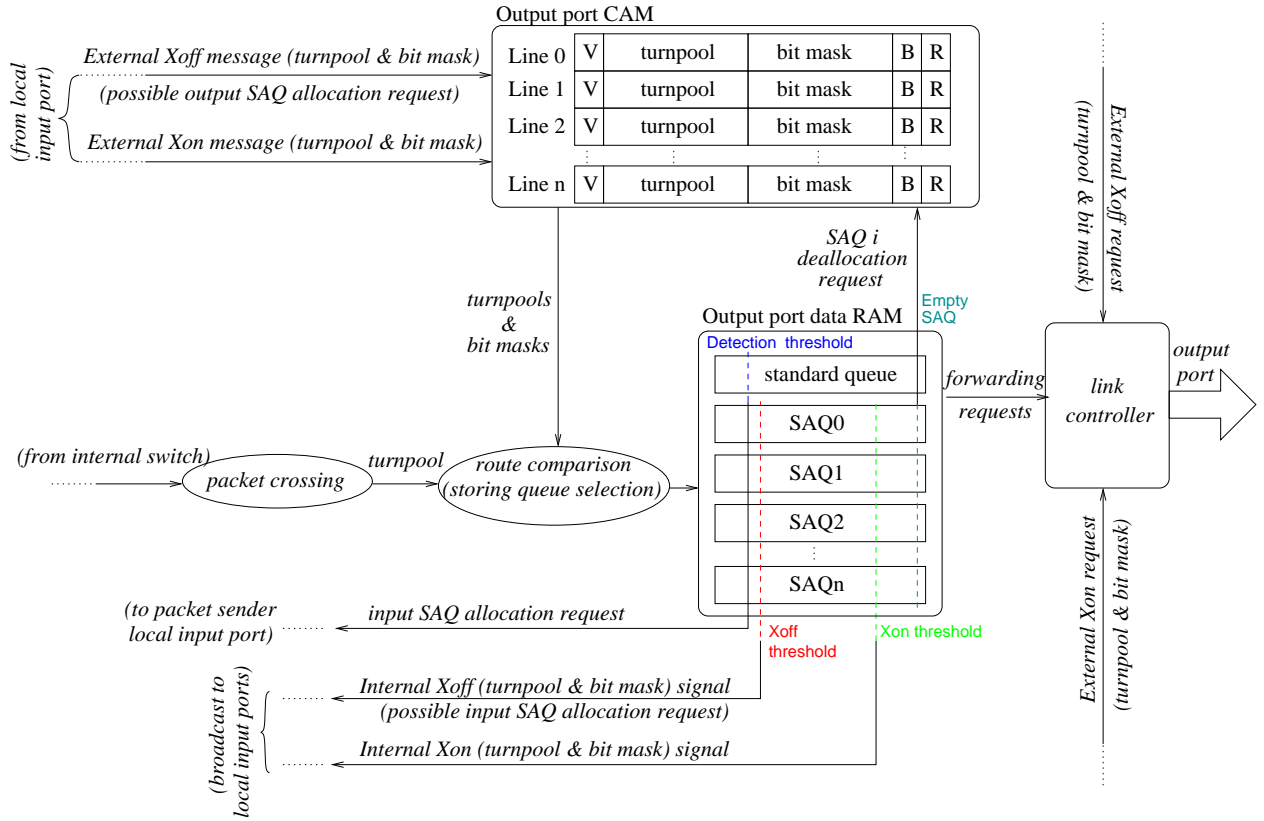


Figure 7: RECN mechanism at the output port of a switch

RECN can be seen in Figure 8. For RECN, 8 SAQs per port are used. Note that the number of queues per switch required by RECN is very similar to the VOQsw mechanism, and therefore RECN can be considered a scalable technique. In fact, taking into account results of figures 8 and 4, we can conclude that RECN combines the scalability of VOQsw and the efficiency of VOQnet.

SAQs are managed through a CAM memory. Each CAM line contains the control information associated with a specific SAQ. This information consists of: a valid bit (V) indicating whether the SAQ is active, a turnpool and a bit mask identifying the root of the congestion tree for which this SAQ is allocated, a blocked bit (B) for the Xon/Xoff flow control (see next section), and a ready bit (R) that indicates whether the SAQ is allowed for sending packets to the arbiter/link controller (see section 4.4). Note that only 65 bits of control information are required per SAQ: 62 bits for the turnpool and bit mask plus other three control bits.

Whenever a new packet arrives to an input port (Figure 6), the turnpool of the packet is compared to the

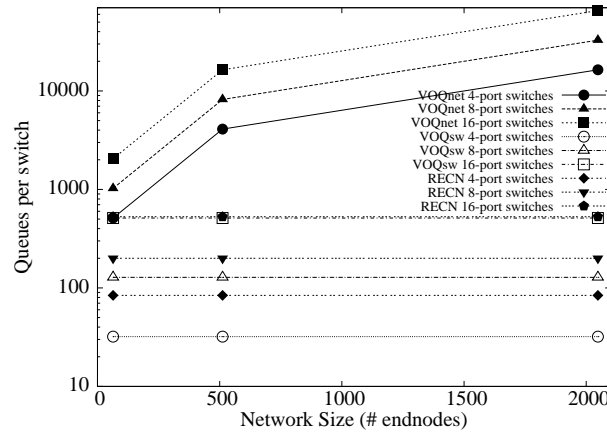


Figure 8: # of queues per switch Virtual Output Queues at network level, at switch level and RECN are used

routes (turnpool+bit mask) contained in all the CAM lines associated with active SAQs (bit V active). In the case of matching, the queue selected for storing the packet is the corresponding SAQ. If the packet turnpool matches several routes, the longest matching is selected. On the opposite, if there is no matching, the packet will be stored in the standard queue (actually, a detection queue, as it will be explained later). This turnpool comparison is performed in the same way at output ports (Figure 7) .

4.3 Congestion Detection and Notification

RECN detects congestion both at input and output ports. At input ports (Figure 6), the standard queue is handled as several detection queues, each one assigned to an specific output port. Non-congested packets are stored in the detection queue associated with their requested output port². Whenever the occupancy of a detection queue reaches a certain *input detection threshold*, a new SAQ allocation is triggered at the same input port. This SAQ allocation implies that the valid bit in the corresponding CAM line is set, and the turnpool and bit mask fields are filled in order to address the output port associated with the detection queue. Additionally, the blocked bit (B) is reset.

At the output port (Figure 7), if a new packet arrives and the standard queue occupancy is beyond a

²Note that if there were no SAQs, the memory at input ports would follow a Virtual Output Queuing scheme at switch level. However, detection queues are not used for eliminating the HOL blocking introduced by congested packets. Instead, the SAQs perform this task at each port.

given *output detection threshold*, an *input SAQ allocation request* is notified to the input port that sent the arriving packet to the output port. This internal notification includes the turnpool and bit mask that identifies the output port from the notified input port. Of course, this information will be stored in the CAM line associated with the input SAQ that will be allocated upon reception of the notification.

Whenever a SAQ at the input port (Figure 6) reaches the Xoff threshold, it raises the sending of an Xoff flow control packet to the upstream switch. This control packet is sent through the adjacent output port (the output port connected to the same bidirectional link than the input port), and it contains the turnpool and the bit mask associated with the input SAQ. Similarly, once the Xon threshold is reached down in a SAQ, an Xon flow control packet (also containing the corresponding turnpool and bit mask) is sent through the adjacent output port. Upon reception of the Xoff control packet, the output port at the upstream switch compares the received turnpool and bit mask to all the valid turnpools from the CAM. If there is a matching, then the corresponding SAQ is blocked (B bit is set). If there is no matching, then the Xoff control packet is considered also as a congestion notification and thus, a new SAQ is allocated for the network point indicated by the received turnpool and bit mask.

Similarly, whenever a SAQ at the output port (Figure 7) reaches the Xoff or Xon threshold, it broadcasts an internal flow control notification to all the input ports of its switch. This notification includes the turnpool and bit mask associated with the output SAQ. At each input port, this routing information is updated to include a new turn: the one required for reaching the notifying output port from the notified input port. Each input port compares the modified turnpool and bit mask to all the valid turnpools in its CAM. On matching, it handles the internal notification as a flow control notification (the B bit of the corresponding CAM line is set accordingly). If the notification is Xoff type and there is no match, then the input port that sent the last packet to the notifying output port allocates a new SAQ for the network point encoded by the modified turnpool and bit mask. This newly allocated SAQ is blocked by setting its B bit.

Note that due to the use of flow control messages for propagating congestion information, RECN does not require specific control messages. Moreover, the Xon/Xoff flow control is included in the Advanced Switching specifications. Therefore, regarding control packets, RECN is fully compatible with Advanced Switching.

4.4 In Order Delivery

As SAQs are dynamically allocated, out of order issues may arise. Note that whenever a SAQ is allocated to store packets passing through a congested root, there may be packets following the same route in the standard queue. Thus, a packet addressed to a certain destination could take over its preceding packet through a newly allocated SAQ. In order to prevent this problem, the RECN mechanism incorporates a blocking mechanism that prevents the sending of packets from a SAQ until in order delivery is guaranteed. In particular, whenever a SAQ is allocated, its R bit is set. This bit prevents the SAQ from sending packets to the scheduler/link controller. At the same time, in the standard queue, a link pointer (pointing to the SAQ) is attached to the last packet. Once a link pointer reaches the head of the standard queue, the corresponding SAQ is unblocked (its R bit is reset), and thus, this SAQ is enabled for sending packets to the scheduler/link controller.

Notice that this mechanism may introduce a certain degree of HOL blocking during the time the link pointer is in the normal queue. However, this is not a problem as HOL blocking is introduced from the cold packets to the congested ones.

4.5 Resource Deallocation

A SAQ must be deallocated when its associated congestion tree vanishes. Of course, it is very important to avoid early deallocations, that could introduce HOL blocking. RECN detects that congestion is no longer present in a port when certain conditions are met, and only then the corresponding SAQ will be deallocated. In order to allow a distributed SAQ deallocation, these conditions must be checked from local information. In detail, they are the following:

- The SAQ is empty.
- The SAQ is not blocked due to the ready (R) bit at the CAM line. So, there are no link pointers to this SAQ on any other queue. This condition avoids an empty SAQ being deallocated just after its allocation while the congestion tree is present.
- The SAQ is not blocked due to the Xoff flow control. So, the SAQ must be in the Xon state to be deallocated. This condition avoids a SAQ to be deallocated while the congestion tree is near (the

corresponding downstream SAQ occupancy is over the Xoff threshold).

Note that a SAQ does not depend on other SAQs for being deallocated. Therefore, as soon as a SAQ is not necessary (its associated congestion tree has disappeared), it will be deallocated. Of course, immediately after its deallocation, the SAQ can be allocated for other congestion tree. This means that the distributed SAQ deallocation allows a very efficient use of network resources. Note also that the deallocation mechanism does not require any specific control message between SAQs, guaranteeing the mentioned RECN compatibility with Advanced Switching.

5 RECN Robustness

RECN is a dynamic mechanism. As a dynamic process, its effectiveness largely depends on the different parameters it uses. As mentioned in the description, RECN depends exclusively on the following parameters: queue detection threshold, Xon and Xoff flow control thresholds, and finally, the number of SAQs per port. All these parameters will also determine the memory requirements at switches.

Figure 9 shows the performance achieved in networks of different size when RECN is used with different detection and flow control thresholds. In all the cases, the traffic load is the same used in previous Figures and 8 SAQs per port have been used. The different thresholds used are indicated in the Figures by three consecutive numbers: the first one corresponds to the queue detection threshold while the second and third correspond to the Xon and Xoff flow control thresholds, respectively. All the thresholds are expressed in bytes.

As it can be observed, RECN filters in all the cases the congestion. The only difference, clearly visible in large networks, is the transitory state while the congested situation is being detected and notified by RECN. Notice that low thresholds lead to quicker response times to congestion and therefore, the transient HOL blocking introduced by RECN is minimized. Notice also that the probability of detecting false congested situations is higher with low thresholds. However, the distributed deallocation of SAQs alleviates the possible negative effects, thus enabling the use of low thresholds. From these results we can conclude that RECN effectiveness is largely insensitive to detection and flow control thresholds.

The use of low thresholds reduces also the memory requirements to implement RECN. The required

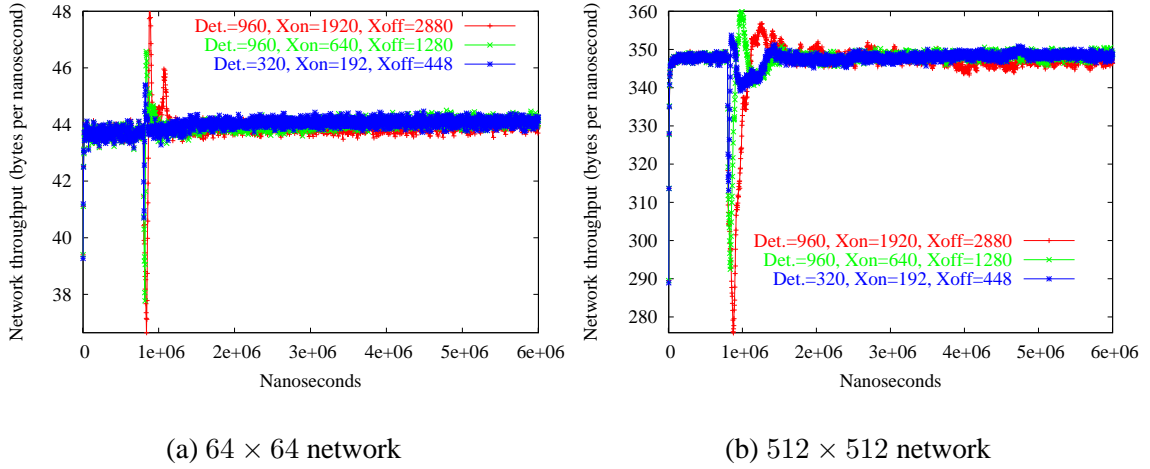


Figure 9: Network Throughput for different RECN thresholds on BMINs of different size

amount of memory at an input port depends on the detection and Xoff flow control thresholds, and also on the number of detection queues and SAQs. In particular, for the analyzed cases with 8-port switches, the required memory for RECN with 8 SAQs, detection threshold set to 320 bytes and Xoff flow control threshold set to 448 bytes, the memory requirements raise to only 6 KB per input port (at the output port, since there are no detection queues memory requirements are even lower). Thus, using low thresholds also simplifies switch design.

Figure 10 shows the performance achieved in networks of different size when RECN is used with different thresholds, but this time adjusting the memory at input ports to match the amount required by each set of thresholds. As it can be observed, performance is practically maximum in the presence of congestion regardless of memory size and thresholds. Note that figure 10(b) shows the performance achieved on a very large BMIN (2408 endnodes).

Finally, Figure 11 shows RECN, VOQnet and VOQsw results for SAN traces on a 64×64 BMIN. In all the cases, the memory at each port is 8 KB. For RECN, 8 SAQ per port have been used, with low detection and flow control thresholds. The I/O traces used in our evaluations were provided by Hewlett-Packard Labs [23]. 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 almost seven years old, we have applied a time compression factor to the traces. It can be observed that, regardless of the compression factor, RECN achieves the maximum

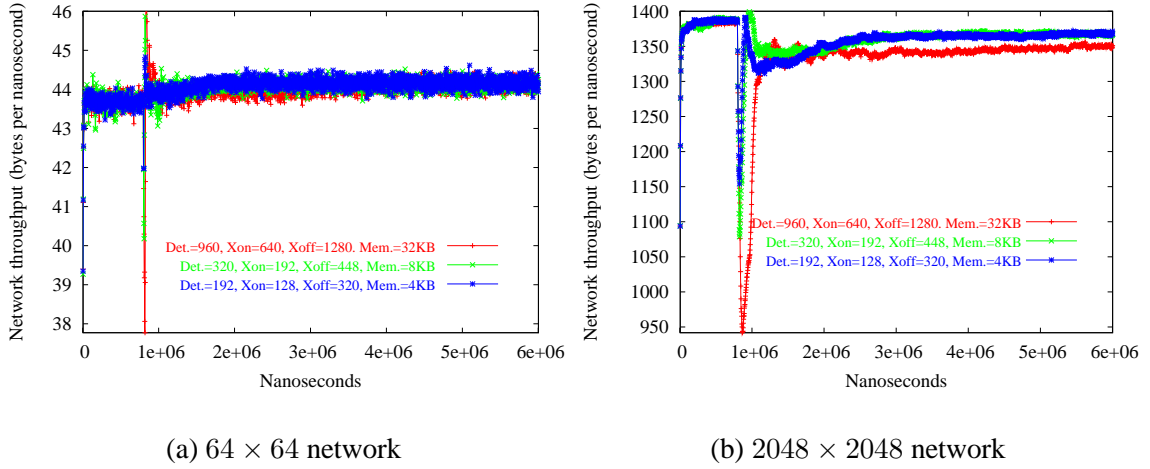


Figure 10: Network Throughput for RECN on BMINs of different size, different port memory sizes and different thresholds

throughput also for real traffic, matching the VOQnet results while requiring reduced resources.

6 Conclusions

In this paper we have described the basis for designing congestion management techniques for lossless interconnection networks that guarantee at the same time scalability and efficiency. Furthermore, we have presented a detailed description of RECN, the first technique of such characteristics. RECN achieves efficiency by accurately detecting and isolating congestion trees within the network. By doing this, HOL blocking introduced by congested packets is eliminated and therefore, congestion is harmless. In the same way, RECN is a scalable technique in the sense that it requires the same resources regardless of network size and it achieves in all the cases maximum efficiency. The paper has proven that RECN is highly insensitive to its main parameters and thresholds. Indeed, thanks to its distributed deallocation mechanism, RECN can use low detection thresholds that lead to very low memory requirements at switches.

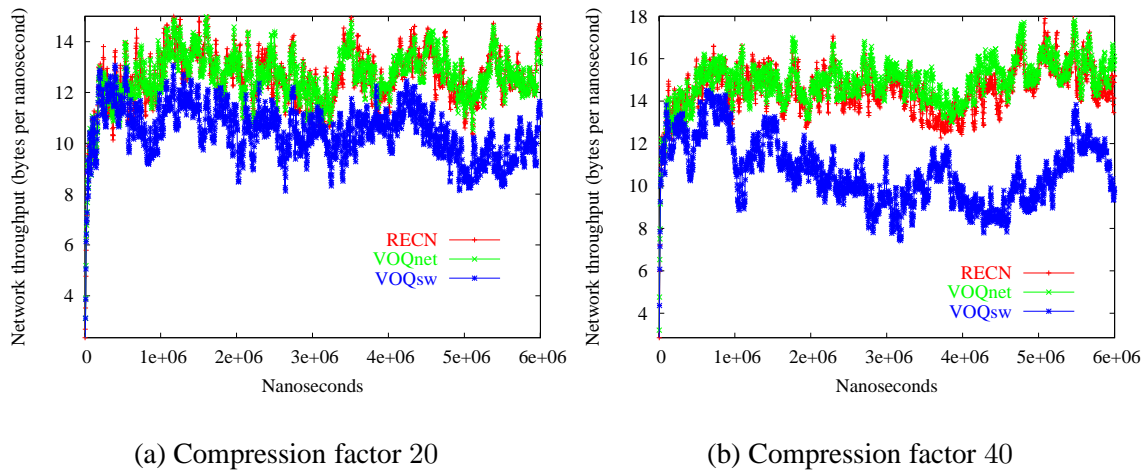


Figure 11: Network Throughput, SAN traces on a 64×64 BMIN

References

- [1] G. S. Almasi, A. Gottlieb, “Highly parallel computing”, *Ed. Benjamin-Cummings Publishing Co., Inc.*, 1994.
- [2] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, “High-Speed Switch Scheduling for Local-Area Networks”, *ACM Trans. on Computer Systems*, vol. 11, no. 4, pp. 319–352, Nov. 1993.
- [3] E. Baydal, P. Lopez and J. Duato, “A Congestion Control Mechanism for Wormhole Networks”, in *Proc. 9th. Euromicro Workshop Parallel & Distributed Processing*, pp. 19–26, Feb. 2001.
- [4] E. Baydal and P. Lopez, “A Robust Mechanism for Congestion Control: INC”, in *Proc. 9th International Euro-Par Conference*, pp. 958–968, Aug. 2003.
- [5] R. Bianchini, T. J. LeBlanc, L. I. Kontothanassis, M. E. Crovella “Alleviating Memory Contention in Matrix Computations on Large-Scale Shared-Memory Multiprocessors”, Technical report 449, Dept. of Computer Science, Rochester University, April 1993.
- [6] W. J. Dally, P. Carvey, and L. Dennison, “The Avici Terabit Switch/Router”, in *Proc. Hot Interconnects 6*, Aug. 1998.
- [7] W. J. Dally, “Virtual-channel flow control,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, March 1992.
- [8] W. J. Dally and H. Aoki, “Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels”, *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 4, pp. 466–475, April 1993.

- [9] S. P. Dandamudi, “Reducing Hot-Spot Contention in Shared-Memory Multiprocessor Systems”, in *IEEE Concurrency*, vol. 7, no 1, pp. 48–59, January 1999.
- [10] J. Duato, I. Johnson, J. Flich, F. Naven, P.J. Garcia, T. Nachiondo, “A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks”, in *Proc. 11th International Symposium on High-Performance Computer Architecture (HPCA05)*, pp. 108–119, Feb. 2005.
- [11] P.J. Garcia, J. Flich, J. Duato, I. Johnson, F.J. Quiles, F. Naven, “Dynamic Evolution of Congestion Trees: Analysis and Impact on Switch Architecture”, *Lecture Notes in Computer Science (HiPEAC-2005)*, vol. 3793, pp. 266–285, November 2005.
- [12] W.S.Ho, D.L.Eager, “A Novel Strategy for Controlling Hot Spot Contention”, in *Proc. Int. Conf. Parallel Processing*, vol. I, pp. 14–18, 1989.
- [13] M. Katevenis, D. Serpanos, E. Spyridakis, “Credit-Flow-Controlled ATM for MP Interconnection: the ATLAS I Single-Chip ATM Switch”, in *Proc. 4th Int. Symp. on High-Performance Computer Architecture*, pp. 47–56, Feb. 1998.
- [14] J. H. Kim, Z. Liu, and A. A. Chien, “Compressionless Routing: A Framework for Adaptive and Fault-Tolerant Routing”, *IEEE Trans. on Parallel and Distributed Systems*, vol. 8, no. 3, 1997.
- [15] V. Krishnan and D. Mayhew, “A Localized Congestion Control Mechanism for PCI Express Advanced Switching Fabrics”, in *Proc. 12th IEEE Symp. on Hot Interconnects*, Aug. 2004.
- [16] J. Liu, K. G. Shin, C. C. Chang, “Prevention of Congestion in Packet-Switched Multistage Interconnection Networks”, *IEEE Transactions on Parallel Distributed Systems*, vol. 6, no. 5, pp. 535–541, May 1995.
- [17] P. Lopez and J. Duato, “Deadlock-Free Adaptive Routing Algorithms for the 3D-Torus: Limitations and Solutions”, in *Proc. Parallel Architectures and Languages Europe 93*, June 1993.
- [18] Myrinet 2000 Series Networking. Available at http://www.cspi.com/multicomputer/products/2000_series_networking
- [19] “Advanced Switching Core Architecture Specification”. Available at <http://www.asi-sig.org/specifications> for ASI SIG.
- [20] S. L. Scott, G. S. Sohi, “The Use of Feedback in Multiprocessors and Its Application to Tree Saturation Control”, *IEEE Transactions on Parallel Distributed Systems*, vol. 1, no. 4, pp. 385–398, Oct. 1990.
- [21] L. Shang, L. S. Peh, and N. K. Jha, “Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks”, in *Proc. Int. Symp. on High-Performance Computer Architecture*, pp. 91–102, Feb. 2003.
- [22] A. Smal and L. Thorelli, “Global Reactive Congestion Control in Multicomputer Networks”, in *Proc. 5th Int. Conf. on High Performance Computing*, 1998.
- [23] SSP homepage, <http://ginger.hpl.hp.com/research/itc/csl/ssp/>

- [24] Y. Tamir and G. L. Frazier, “Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches”, *IEEE Trans. on Computers*, vol. 41, no. 6, June 1992.
- [25] M. Thottethodi, A. R. Lebeck, S. S. Mukherjee, “Self-Tuned Congestion Control for Multiprocessor Networks”, in *Proc. Int. Symp. High-Performance Computer Architecture*, Feb. 2001.
- [26] W. Vogels et al, “Tree-Saturation Control in the AC3 Velocity Cluster Interconnect”, in *Proc. 8th Conference on Hot Interconnects*, Aug. 2000.
- [27] C. Q. Yang and A. V. S. Reddy, “A Taxonomy for Congestion Control Algorithms in Packet Switching Networks”, *IEEE Network*, pp. 34–45, July/Aug. 1995.
- [28] P. Yew, N. Tzeng, D. H. Lawrie“ Distributing Hot-Spot Addressing in Large-Scale Multiprocessors”,*IEEE Transactions on Computers*, vol. 36, no. 4, pp. 388–395, April 1987.