

---

# EFFICIENT, SCALABLE CONGESTION MANAGEMENT FOR INTERCONNECTION NETWORKS

---

COMPARED TO THE OVERDIMENSIONED DESIGNS OF THE PAST, CURRENT INTERCONNECTION NETWORKS OPERATE CLOSER TO THE POINT OF SATURATION AND RUN A HIGHER RISK OF CONGESTION. AMONG PROPOSED STRATEGIES FOR CONGESTION MANAGEMENT, ONLY THE REGIONAL EXPLICIT CONGESTION NOTIFICATION (REC�) MECHANISM ACHIEVES BOTH THE REQUIRED EFFICIENCY AND THE SCALABILITY THAT EMERGING SYSTEMS DEMAND.

**Pedro J. García**

**Francisco J. Quiles**

University of Castilla-  
La Mancha

**José Flich Cardo**

**José Duato**

Technical University  
of Valencia

**Ian Johnson**

Xyratex

**Finbar Naven**

VirtenSys

..... Congestion in interconnection networks is a widely known problem. When different packets request the same resource, typically an output port, contention occurs. The network grants access to only one packet, while the rest wait. When contention persists, the buffers containing the blocked packets fill, and in lossless networks, which do not allow dropped packets, flow control prevents other switches from sending packets to the congested ports. Although flow control is essential to avoid dropping packets, it rapidly propagates congestion to other switches, because it will also block packets stored at some of their ports. Through these aggregation effects, congestion can spread progressively through the network, forming *congestion trees*, whose branches match the paths followed by data flows contributing to the congestion (hot flows).

Although congestion trees can evolve in different ways (see the “Dynamics of congestion trees” sidebar), all of them severely affect network performance. Packets belonging to hot

flows prevent other packets from advancing, even those belonging to noncongested flows (cold flows). This phenomenon, known as head-of-line (HOL) blocking, occurs when a packet at the head of a first-in, first-out (FIFO) queue becomes blocked, preventing the remaining packets in the same queue from advancing. Of course, the wider the congestion tree is, the greater the probability of congested packets causing HOL blocking.

The impact of HOL blocking on network performance can be very serious: Network throughput can degrade and packet latency can increase dramatically. For instance, Figure 1a shows the generated traffic and network throughput in a bidirectional multistage interconnection network with 64 sending and receiving end nodes (a  $64 \times 64$  BMIN). In this simulated scenario, we have generated congestion by injecting traffic (at the maximum injection rate) from eight sources to the same destination (hot-spot traffic) during a short time interval (300  $\mu$ s). The remaining 56 end nodes send traffic to random destina-

## Dynamics of congestion trees

The traditional view of congestion trees was based on assumptions about their dynamics that have become obsolete for current commercial switch architectures. For instance, one popular belief was that congestion always appears first at switch output ports. However, this idea is reminiscent of the days when switches had queues only at output ports (OQ switches); it is not valid for the switches commonly used today. Current switches might have queues at input ports (IQ switches), or at both input and output ports (CIOQ switches). In fact, in modern switches, congestion can appear on input or output ports, depending on the number of flows causing congestion and on the internal switch speedup (the maximum speed at which packets can be forwarded to the outputs ports, relative to link speed).<sup>1</sup>

Another classic assumption was that congestion trees always grow from root to leaves. Again, recent research has refuted this idea, showing that, in modern networks, depending on traffic patterns and switch architecture, congestion trees can evolve in several ways, exhibiting very complex dynamics.<sup>1</sup> For example, a tree can grow from leaves to root, or vice versa. Several congestion trees can grow independently and later merge, and it is even possible for trees to overlap completely but remain independent. Any congestion management technique that seeks to handle congestion situations efficiently must carefully consider congestion trees' complex and varied, dynamic evolution.

## Reference

1. P.J. Garcia et al., "Dynamic Evolution of Congestion Trees: Analysis and Impact on Switch Architecture," *Proc. Int'l Conf. High Performance Embedded Architectures & Compilers (HiPEAC 2005)*, LCNS 3793, Springer, 2005, pp. 266-285.

tions. Thus, a congestion tree, whose root is at the hot-spot destination, forms. Network throughput not only diminishes severely (by approximately 70 percent) when congestion appears, it also remains low well beyond the end of the hot-spot traffic generation. This degradation is the result of HOL blocking produced by the congestion tree affecting other (noncongested) packets.

Figure 1b shows the average packet latency

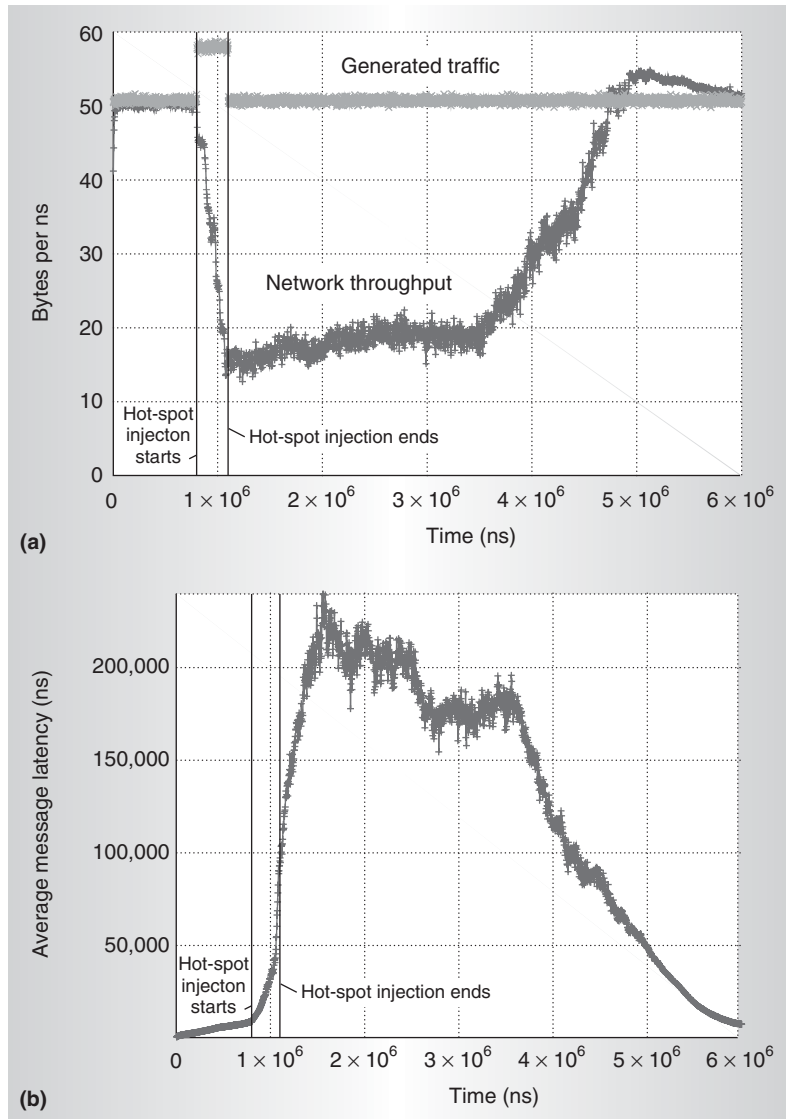


Figure 1. Network performance degrades after congestion appears: Throughput and generated traffic (a), and average latency (b).

during the same congested situation. Latency increases from a few hundred nanoseconds (before congestion) to several hundred microseconds, an increase of three orders of magnitude. Packet latencies eventually return to the low values, but only after several milliseconds. This data also confirms that packets experience massive HOL blocking during and after the hot-spot injection. Once this kind of blocking has occurred, even under relatively benign traffic patterns, congestion trees can remain in the network and keep it operating at reduced efficiency for extended periods of time.

Although the negative consequences of such congestion scenarios have always been clear, designers have not considered network congestion to be a worrying problem until very recently. For many years, designers avoided the problem by overdimensioning interconnection networks, using many more network components than the minimum required for connecting all the system end nodes. This approach let networks operate well below the saturation point, and the lower level of link use reduced the probability of congestion.

So, then, why should we care about congestion now? Unfortunately, cost and power consumption constraints are making overdimensioned networks inappropriate. Current interconnect components are very expensive when compared to processors, so the network represents a high percentage of total system cost. In addition, as VLSI technology advances and link speed increases, interconnects are consuming an increasing fraction of the total system power.<sup>1</sup> Making the situation even more critical, current high-speed links require continual pulse transmission to keep both ends synchronized, even when no data is being transmitted. Consequently, link power consumption is almost independent of link use. Although researchers have proposed frequency- and voltage-scaling techniques to reduce power consumption,<sup>1</sup> these techniques respond slowly and don't solve the cost problem.

Taking all this into account, most designers today choose to reduce system cost and power consumption by reducing the number of network components. If we are to maintain a system's computational power, there are only two ways to reduce the number of network components: increasing the number of end nodes attached to each switch, or using a more suitable fabric topology. However, each of these solutions leads to a higher link-use level, thereby driving the network closer to its saturation point and increasing the likelihood of congestion.

Thus, congestion will become more common in future networks, and designers will have to implement some specific congestion management mechanism or network performance will suffer the corresponding degradation. As we will show in this article, there are various strategies proposed for congestion management, but only the Regional Explicit

Congestion Notification (RECN) mechanism achieves both the required efficiency and the scalability that emerging systems demand.

### Approaches to congestion management

The problems associated with congestion are more difficult to solve in lossless networks than in lossy networks such as the Internet, because lossless networks cannot drop packets when congestion occurs. However, researchers have proposed many techniques, representing two basic approaches. The first approach, described in the "Classical congestion elimination or reduction" sidebar, tries to eliminate the congestion either by preventing its appearance (proactive techniques) or by acting immediately upon detection of congestion (reactive techniques). However, in general, neither of these techniques can effectively eliminate congestion. Proactive congestion management requires knowledge of the network status and application requirements that is not always available. The larger the network, the greater the difficulty in knowing network status, and the more conservative the behavior of the mechanism would have to be. Reactive techniques, on the other hand, do not scale well: The delay between congestion detection at hot spots and reaction at the traffic sources increases with network size, thereby increasing the mechanism's response time. This delay in the feedback chain of a closed-loop control system can lead to oscillations at sources (by the time actions are taken, the congestion has vanished), thus reducing network throughput. Therefore, reactive techniques do not scale—either with network size or with link bandwidth. Because most current networks use high-speed links, this last drawback is an important one. Ultimately, both proactive and reactive techniques present scalability problems.

The second, and more recent, congestion management approach takes the view that congestion itself is not a problem. Techniques that follow this approach try to eliminate congestion's negative effects—mainly HOL blocking—rather than eliminating the congestion trees. The idea is that if the technique can completely eliminate HOL blocking, congestion can exist harmlessly. The elimination of HOL blocking implies that congested flows will not affect the forwarding of noncongested packets. If this were possible, the only pack-

ets delayed by congestion would be the ones contributing to congestion. Because this delay is unavoidable, congested situations would minimally degrade network performance. So the questions now become: Is it possible to effectively eliminate HOL blocking? Will this approach react faster than reactive approaches? Let's turn to those questions next.

### Strategies for eliminating HOL blocking

Historically, the complete elimination of HOL blocking has been possible, but neither easy nor inexpensive. Over many years, researchers have proposed several techniques with this aim, but until recently, none has been fully satisfactory.

In general, the most effective HOL-blocking elimination techniques use separate queues to store packets belonging to different flows. This idea has been implemented in various ways. Some techniques distinguish data flows on the basis of their final destination (HOL blocking elimination at the network level) and use this criterion for mapping packets to queues.<sup>2,3</sup> For instance, if a network uses virtual output queues (VOQs) at the network level (VOQnet), each port has as many queues as end points in the network, and every incoming packet is stored in the queue assigned to its destination. Most techniques for eliminating network-level HOL blocking are very efficient, but they require heavy use of resources (queues) for networks with many end points. Thus, these techniques don't scale well; their implementation on large networks is very expensive in terms of silicon area.

Other techniques try to solve this problem by eliminating HOL blocking at the switch level.<sup>4-7</sup> In this case, both resource allocation and packet storing in a switch port depend on the switch output port that the packet requests. For example, if a network uses VOQs at the switch level (VOQsw), each switch port has as many queues as output ports in the switch, and a packet is 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 the logarithmic scale) required for implementing VOQnet and VOQsw on networks of different sizes (64, 512, and 2,048

---

## Classical congestion elimination or reduction

Although researchers have proposed several taxonomies,<sup>1,2</sup> 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 such that congestion never occurs. Some techniques achieve this goal by reserving network resources in advance (avoidance-based techniques);<sup>3,4</sup> others limit the routes that packets can follow (prevention-based techniques).<sup>5,6</sup> Both types generally require the implementation of admission control and transmission scheduling that need information about the connection requirements and occupancy of network resources (buffers, links, and so on). Because of their conservative behavior, proactive techniques are suitable for providing quality of service support (QoS), but not for congestion management.

*Reactive techniques* detect the formation of hot spots and activate various mechanisms to eliminate the congestion. Most of these mechanisms involve notifying the end nodes of the congestion so that they throttle the injection of packets.<sup>7-10</sup> These notifications must travel from the point where congestion is detected to the traffic sources. The detection itself is usually based on locally measuring the occupancy of network resources.<sup>6,11</sup> The main drawback of these techniques is their lack of scalability: Their reaction time is directly proportional to the distance from the congestion point to the traffic sources.

---

## References

1. S.P. Dandamudi, "Reducing Hot-Spot Contention in Shared-Memory Multiprocessor Systems," *IEEE Concurrency*, vol. 7, no. 1, Jan. 1999, pp. 48-59.
2. C.Q. Yang and A.V.S. Reddy, "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks," *IEEE Network*, vol. 9, no. 4, July-Aug. 1995, pp. 34-45.
3. P. Yew, N. Tzeng, and D.H. Lawrie, "Distributing Hot-Spot Addressing in Large-Scale Multiprocessors," *IEEE Trans. Computers*, vol. 36, no. 4, Apr. 1987, pp. 388-395.
4. R. Bianchini et al., *Alleviating Memory Contention in Matrix Computations on Large-Scale Shared-Memory Multiprocessors*, tech. report 449, Dept. of Computer Science, Rochester University, 1993.
5. W.S. Ho and D.L. Eager, "A Novel Strategy for Controlling Hot Spot Contention," *Proc. Int'l Conf. Parallel Processing*, IEEE Press, 1989, pp. 14-18.
6. S.L. Scott and G.S. Sohi, "The Use of Feedback in Multiprocessors and Its Application to Tree Saturation Control," *IEEE Trans. Parallel Distributed Systems*, vol. 1, no. 4, Oct. 1990, pp. 385-398.
7. M. Thottethodi, A.R. Lebeck, and S.S. Mukherjee, "Self-Tuned Congestion Control for Multiprocessor Networks," *Proc. Int'l Symp. High-Performance Computer Architecture (HPCA 01)*, IEEE CS Press, 2001, pp. 107-120.
8. J.H. Kim, Z. Liu, and A.A. Chien, "Compressionless Routing: A Framework for Adaptive and Fault-Tolerant Routing," *IEEE Trans. Parallel and Distributed Systems*, vol. 8, no. 3, Mar. 1997, pp. 229-244.
9. W.J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 4, Apr. 1993, pp. 466-475.
10. E. Baydal and P. Lopez, "A Robust Mechanism for Congestion Control: INC," *Proc. 9th Int'l Euro-Par Conf.*, LNCS 2790, Springer, 2003, pp. 958-968.
11. W. Vogels et al, "Tree-Saturation Control in the AC<sup>3</sup> Velocity Cluster Interconnect," *Proc. IEEE Symp. High-Performance Interconnects*, 2000; <http://www.hoti.org/archive/hoti8papers/008.pdf>.

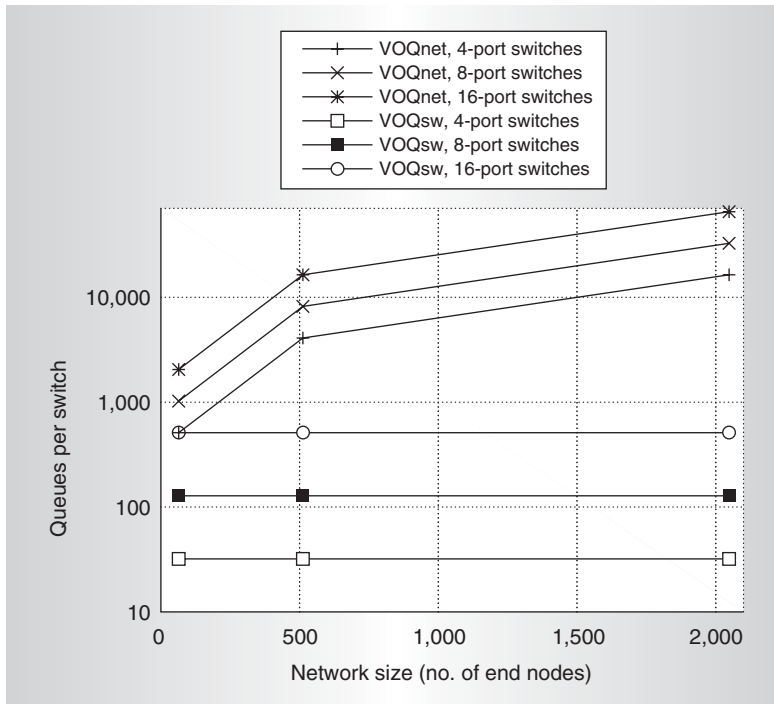


Figure 2. Number of queues required per switch, assuming virtual output queues at the network level (VOQnet) and virtual output queues at the switch level (VOQsw).

end nodes) built with switches with different radices (4, 8, and 16 ports per switch). The VOQnet queue requirements grow with the number of ports and especially with network size, but VOQsw requirements grow only with the number of ports. Queue requirements for the two techniques differ by approximately two orders of magnitude for medium and large networks, and this difference tends to grow as network size increases. In fact, the excessive number of queues that VOQnet requires for medium or large networks makes the scheme practically infeasible.

Although VOQsw requirements are constant with network size, they do not completely eliminate HOL blocking, because congested and noncongested packets might be stored in the same queue. Figure 3 shows the network throughput obtained when using VOQnet, VOQsw, and only one queue per input port, for the same traffic and network scenario assumed in Figure 1. Although VOQsw alleviates congestion by reducing HOL blocking, it still has some network throughput degradation (approximately 20 percent) and exhibits a considerable packet latency (around 50 μs).

This confirms that VOQsw only partially eliminates HOL blocking. On the other hand, VOQnet maximizes network throughput and minimizes packet latency before, during, and after the hot-spot injection.

To sum up, previous techniques for eliminating HOL blocking were either efficient or scalable, but not both. In fact, for many years, it seemed that it wouldn't be possible to obtain a technique with both characteristics. This situation changed recently, with the proposal of RECN. This technique for eliminating HOL blocking is fully efficient and scalable.<sup>8</sup>

Figure 4 shows the network throughput obtained when using VOQnet, VOQsw, and RECN, this time for two BMINs of different sizes (64 × 64 and 512 × 512) with different traffic generation rates. Again, we generated congestion in each simulated point by temporarily injecting packets from 25 percent of the sources to hot-spot destinations (one hot-spot destination for the small network, four hot spots for the larger one). In both cases, results for RECN and VOQnet are very similar—and far better than VOQsw results. This means that RECN almost completely eliminates HOL blocking, regardless of the traffic rate. Indeed, both RECN and VOQnet allow maximum injection rates without throughput degradation. However, RECN achieves these results for both networks using the same number of queues per switch port, whereas the number of queues VOQnet requires per switch port (64 and 512) increases with network size. Thus, RECN efficiently eliminates HOL blocking, with affordable and constant resource use.

So, how does RECN differ from other HOL-blocking elimination techniques? The first key difference is that, to store congested flows separately, the previous techniques allocated queues statically, whereas RECN allocates and deallocates queues dynamically. Thus, RECN uses queues only as necessary. Another difference is that RECN can address congestion at any network point, not only endpoints. This lets RECN accurately identify congested flows and noncongested ones, so it can separate them. Finally, RECN recognizes that packets belonging to noncongested flows can be stored in the same queue without producing significant HOL blocking, even if they follow different routes. This allows a great

reduction in the number of queues that the mechanism requires.

### RECN implementation

RECN is implemented both at input and output ports. Figures 5 and 6 show, for the two sides, the main structures and events involved in the basic RECN procedure.

### Routing requirements

Like other techniques for eliminating HOL blocking, RECN is based on storing the packets belonging to congestion trees in separate queues. More precisely, 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, the packet belongs to a congestion tree and must be stored at this port in the corresponding separate queue. Therefore, RECN requires the ability to inspect packet routes in advance at any port. In addition, to achieve maximum accuracy and efficiency, RECN must also detect congestion within the network, not only at the endpoints. Therefore, it requires a method of identifying any network port.

The use of source deterministic routing helps RECN meet both requirements. Specifically, RECN assumes the routing properties of Advanced Switching, an open standard for fabric-interconnection technologies developed by the ASI Special Interest Group and supported by many leading enterprises. AS is based on PCI Express technology, extending it to include other features. (RECN could be easily adapted to any other interconnection technology that allows source routing, such as Myrinet; [http://www.cspi.com/multicomputer/products/2000\\_series\\_networking/2000\\_networking.htm](http://www.cspi.com/multicomputer/products/2000_series_networking/2000_networking.htm).) Packet headers in Advanced Switching contain a 31-bit field that encodes all the turns that a packet must take along its route (a turn is a shift from a switch input port to an output port). This field is known as the *turnpool*. The header also includes a pointer (turn pointer) that indicates the next turn. This routing mechanism lets each of the different network ports address the root of a congestion tree using a specific sequence of turns. Moreover, the routing mechanism also lets RECN determine

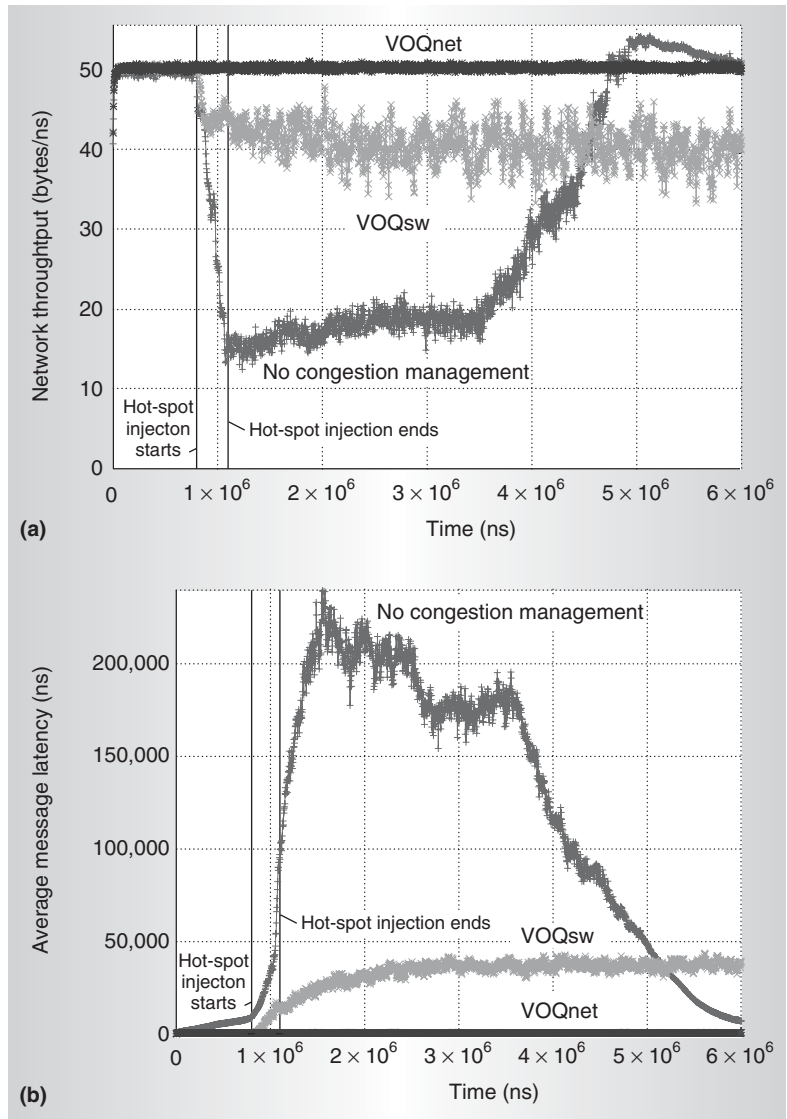


Figure 3. Effectiveness of VOQnet and VOQsw compared to a network without congestion management: throughput and generated traffic (a), and average latency (b). Network performance degrades in a congestion situation in a network using VOQsw, but not in a network using VOQnet.

whether a packet in a port will pass through a congestion tree root simply by inspecting a sequence of turns from its turnpool.

Figure 7 (page ??) shows an example in which several packets travel across a network where a congestion tree exists (the root is at output port P5 of switch 2). The root can be identified at any other port by a relative route (some consecutive turns) from that port, allowing advance detection of which packets will pass through the root (packets A, B, and D) and which packets will not (packet C).

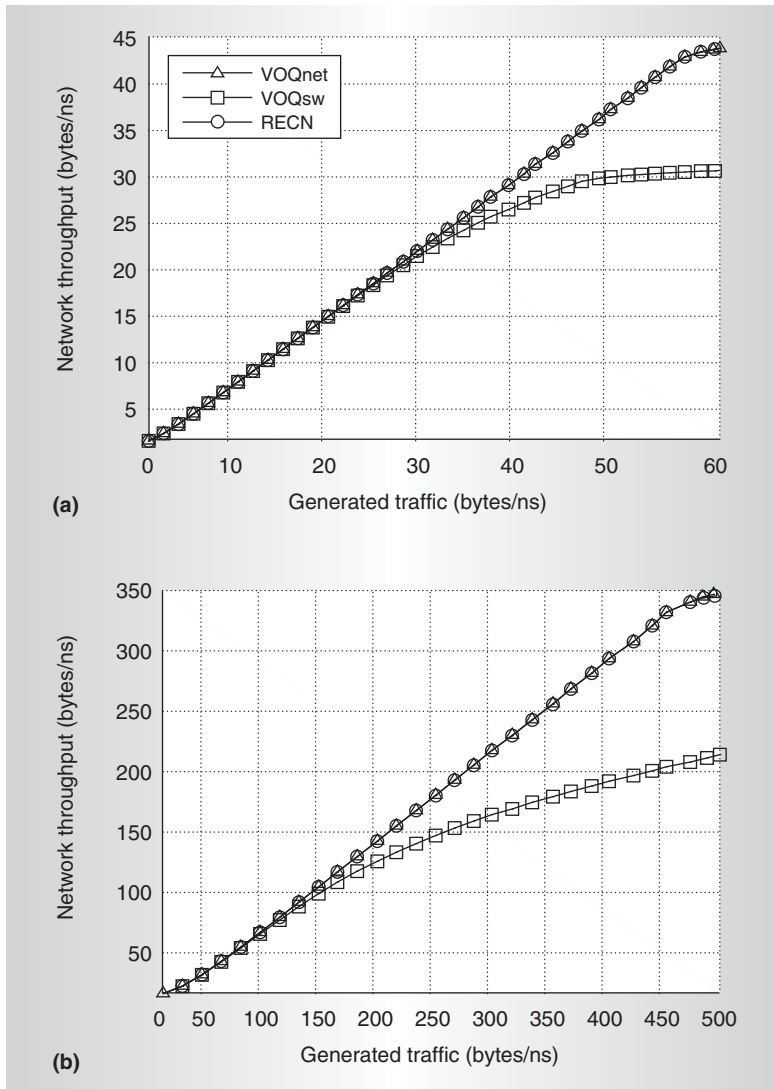


Figure 4. Comparative network throughput using VOQnet, VOQsw, and RECN in 64 × 64 BMIN network (a) and 512 × 512 BMIN network (b).

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

#### Required resources

RECN requires additional resources both at input (Figure 5) and output (Figure 6) switch ports. In particular, besides the stan-

dard queue, at each port RECN uses a set of additional queues, called set-aside queues (SAQs). A port dynamically allocates a SAQ to store packets belonging to a specific congestion tree once the port detects, or is notified of, the tree's existence. When congestion clears, the port deallocates the corresponding SAQ, making it available to store packets belonging to other congestion trees. Each switch port stores packets belonging to non-congested flows in its standard queue. Thus, the normal queue and the set of SAQs share the data RAM at each port.

Although the number of SAQs per port could be varied, the dynamic allocation and deallocation arrangement means that fewer SAQs can efficiently handle congested situations. Moreover, the number of SAQs is independent of network size. For instance, Figure 4 shows that eight SAQs per port can completely eliminate HOL blocking on 64 × 64 and 512 × 512 BMINs. In fact, RECN can eliminate HOL blocking in a 2,048 × 2,048 BMIN network using just eight SAQs per port.<sup>9</sup> In short, the RECN mechanism is fully scalable.

Figure 8 compares the number of queues per switch required for implementing VOQnet, VOQsw, and RECN. RECN and VOQsw require very similar numbers of queues per switch, further demonstrating RECN's scalability. Together, Figures 8 and 4 show that RECN combines the scalability of VOQsw and the efficiency of VOQnet.

RECN uses a content addressable memory (CAM) to manage the SAQs. 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  $X_{on}/X_{off}$  flow control (where  $X_{on}$  and  $X_{off}$  are the thresholds for resuming and stopping the incoming flow); and a ready bit (R), which indicates whether the SAQ may send packets to the link controller. Only 65 bits of control information are required per SAQ: 62 bits for the turnpool and bit mask, and the 3 control bits.

Whenever a new packet arrives at an input port (Figure 5), the port compares the packet's turnpool to the routes (turnpool and bit mask) contained in all the CAM lines associ-

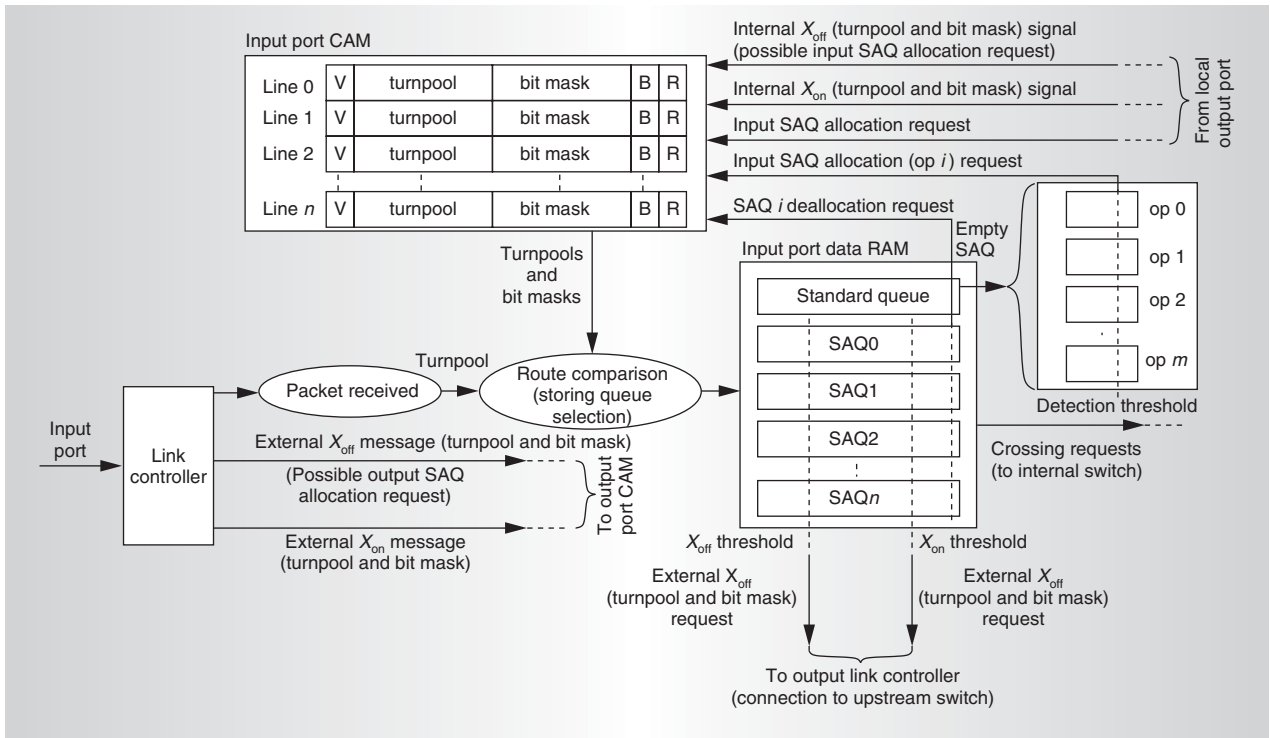


Figure 5. RECN mechanism at a switch input port.

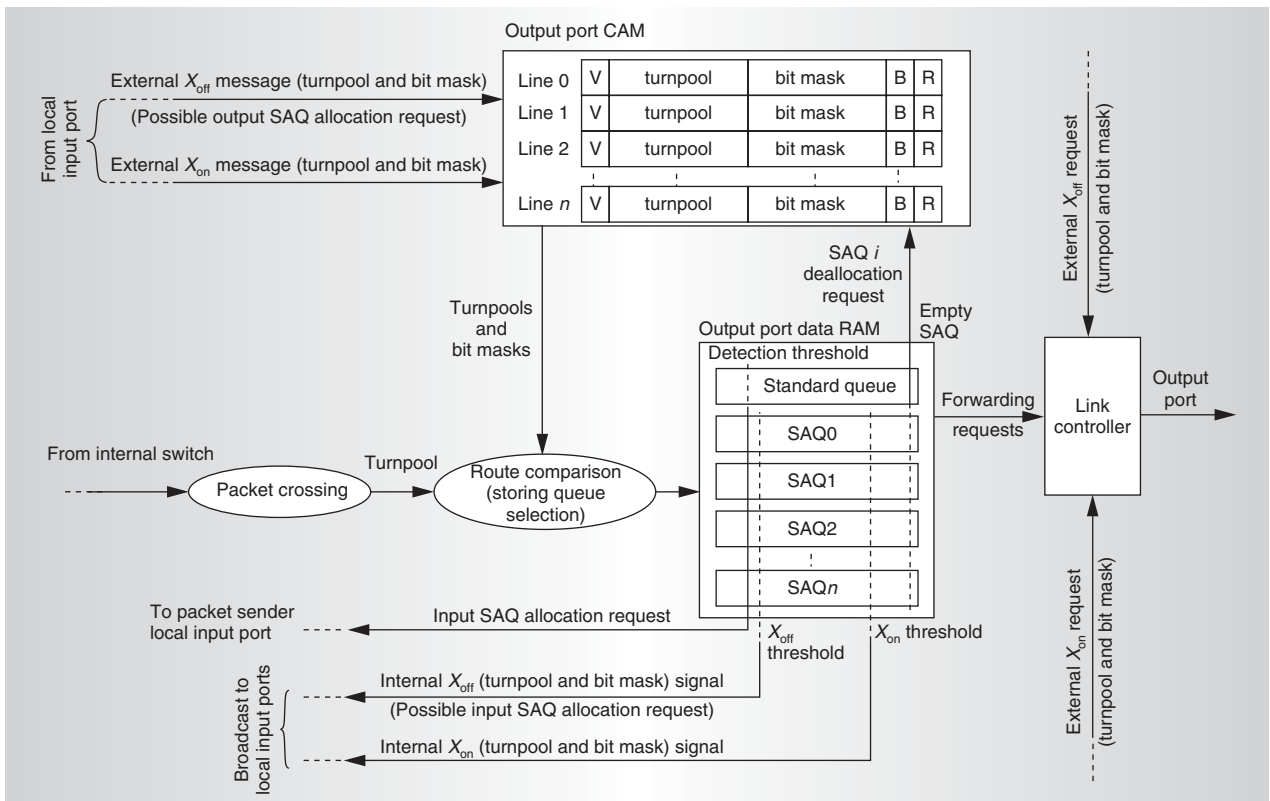


Figure 6. RECN mechanism at a switch output port.

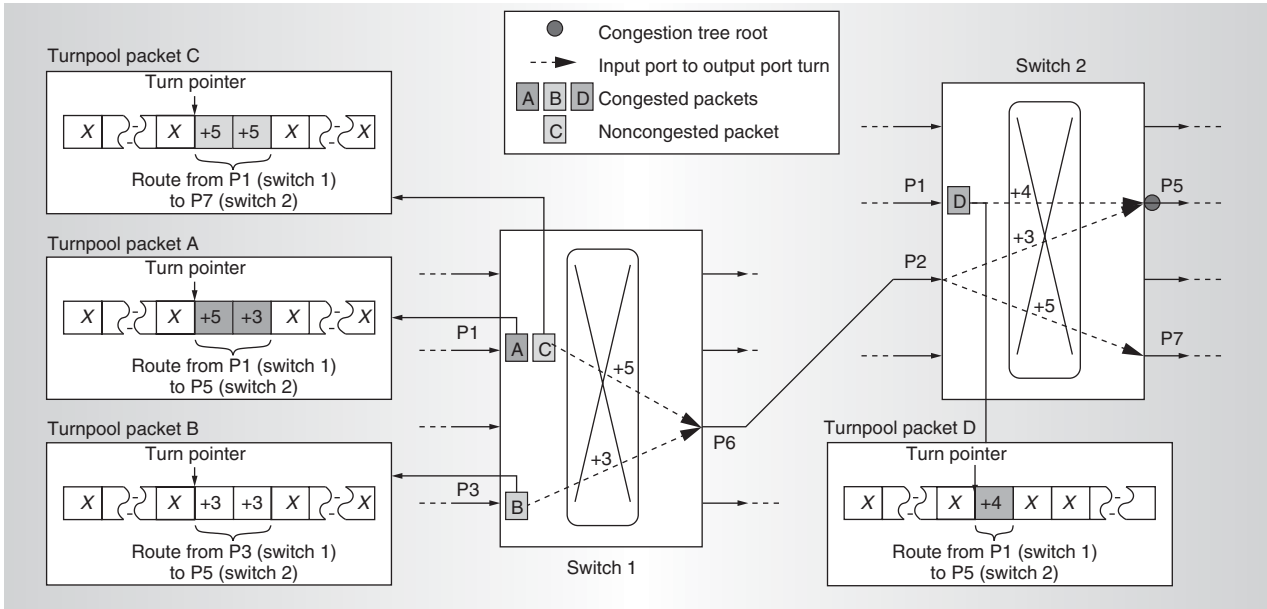


Figure 7. Turnpool inspection allows advance detection of congested packets.

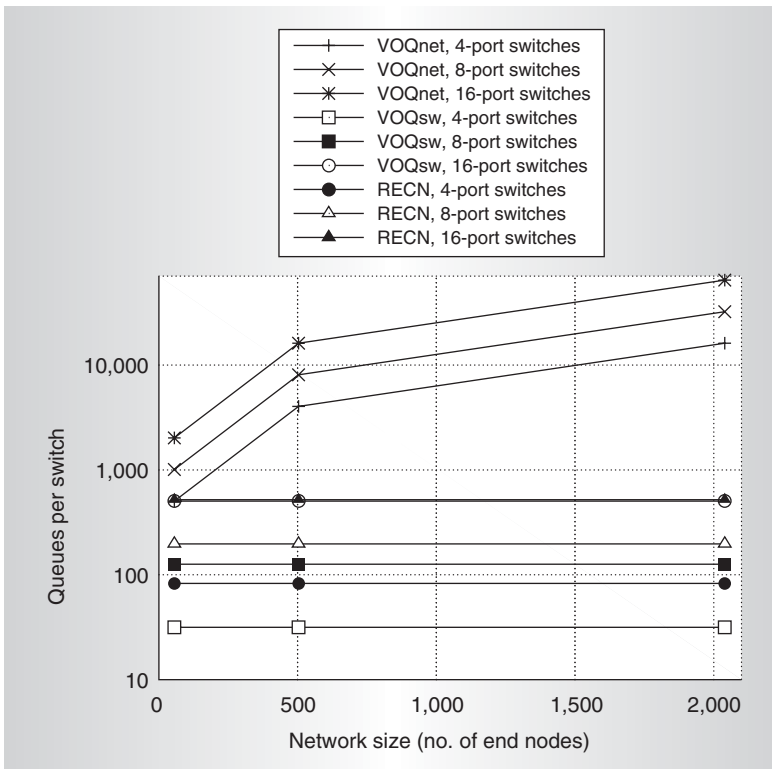


Figure 8. Number of queues per switch for VOQnet, VOQsw, and RECN.

ated with active SAQs (those with bit V active). When a SAQ matches, it becomes the queue selected for storing the packet. If the packet turnpool matches several routes, the longest

matching one is selected. When the packet's turnpool fails to match, the packet goes to the standard queue (actually, a detection queue). This turnpool comparison takes place in the same way at output ports (Figure 6).

### Congestion detection and notification

At input ports (Figure 5), the standard queue in the RECN mechanism is divided into several detection queues, each assigned to a specific output port. Each noncongested packet is stored in the detection queue associated with its requested output port. (Without the SAQs, the memory arrangement at the input ports would follow a virtual output queuing scheme at the switch level. However, RECN does not use the detection queues to eliminate HOL blocking introduced by congested packets. Instead, the SAQs perform this task at each port.)

Whenever the occupancy of a detection queue reaches a certain input detection threshold, that input port triggers a new SAQ allocation. Each such allocation involves the valid bit being set in the corresponding CAM line, and the turnpool and bit mask fields being filled to address the output port associated with the detection queue. Additionally, SAQ allocation resets the blocked bit (B).

At the output port (Figure 6), whenever a new packet arrives and the standard queue

## RECN control packet format for Advanced Switching

Advanced Switching allows the use of various packet formats for the different protocol interfaces (PIs) contained in the AS specifications. Figure A shows the Regional Explicit Congestion Notification (RECN) PI-7 format currently undergoing standardization by the Advanced Switching Interconnect Special Interest Group (ASI-SIG; <http://www.asi-sig.org>). PI-7 packets serve as reliable, highest-priority AS transaction layer packets (TLPs) and do not consume any credits for being transmitted. The AS packet format proposal for PI-7 consists of two AS dwords (a total of 8 bytes or 64 bits), organized so that they can easily contain all the information that must be sent inside each RECN notification. Thus, the PI-7 packet format can be used for building RECN control packets.

Figure A shows the field organization for a generic RECN control packet that follows the PI-7 packet format scheme. (Figure 9 shows an example of a real control packet.) This format includes some fields required only for identifying and transmitting the packet, such as PI, packet type, and cyclic redundancy check (CRC). Other fields, such as turnpool and number of bits set to one at bit mask, contain the useful congestion information. This congestion information comes from a CAM line at the noti-

fying (downstream) port.

In the packet type field, a 0 value identifies the packet as an RECN control packet. (This value is subject to final agreement with ASI-SIG.) The CRC, ordered-only (OO), and traffic class fields are required by AS features not related to RECN notifications. The flow-control-type field (FC) consists of two bits encoding four possible values corresponding to the four different types of notifications or notification acknowledgments:  $X_{\text{off}}$  (0),  $X_{\text{on}}$  (1), positive ack (2), and negative ack (3).

PI-7 packets are not routed, as they communicate only between link partners. Therefore, the five bits usually assigned to the turn pointer field (included in all AS headers) are available for other purposes. Specifically, RECN control packets use these bits to codify the number of bits set to 1 in the bit mask associated with the notified turnpool. Because the bits set to 1 in any bit mask in a CAM line are always consecutive and right aligned (turnpools at CAM lines always consist of right-aligned, consecutive turns), no more information is necessary for defining the bit mask. However, the control packet includes the entire turnpool, taken directly from the corresponding CAM line.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header CRC				Number of set bits in bit mask				Packet type (RECN=0)		OO		Traffic class		FC type		PI (7)															
Turnpool																															

Figure A. Advanced Switching PI-7 packet format for RECN.

occupancy is beyond a given output detection threshold, that port sends an input SAQ allocation request to the input port that sent the arriving packet. This internal notification includes the turnpool and bit mask that identifies the output port from the notified input port. This information goes into the CAM line associated with the input SAQ that the input port allocates upon receiving the notification.

Whenever a SAQ at the input port (Figure 5) reaches the  $X_{\text{off}}$  threshold, it raises the sending of an  $X_{\text{off}}$  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 as the input port), and it contains the turnpool and the bit mask associated with the input SAQ. Similarly, once the  $X_{\text{on}}$  threshold is reached in a SAQ, an  $X_{\text{on}}$  flow control packet—also containing the corresponding turnpool and bit mask—is sent through the adjacent output port. Upon receiving the  $X_{\text{off}}$  control packet, the output port at the upstream switch compares the

received turnpool and bit mask to all the valid turnpools from the CAM. When the output port finds a match, it blocks the corresponding SAQ (by setting the B bit). When it does not find a match, it considers the  $X_{\text{off}}$  control packet to be a congestion notification, and thus allocates a new SAQ for the network point indicated by the received turnpool and bit mask.

Because RECN uses flow control messages to propagate congestion information between switches, the mechanism requires only one type of control message. Moreover, the information that must be included in this type of message fits into a certain control packet format defined in the Advanced Switching specifications (see the “RECN control packet format for Advanced Switching” sidebar). In terms of control packets, RECN is fully compatible with Advanced Switching. Figure 9 shows an example in which a  $X_{\text{off}}$  notification packet is sent between two switches. Along with other control information, the format of the notification packet includes all the

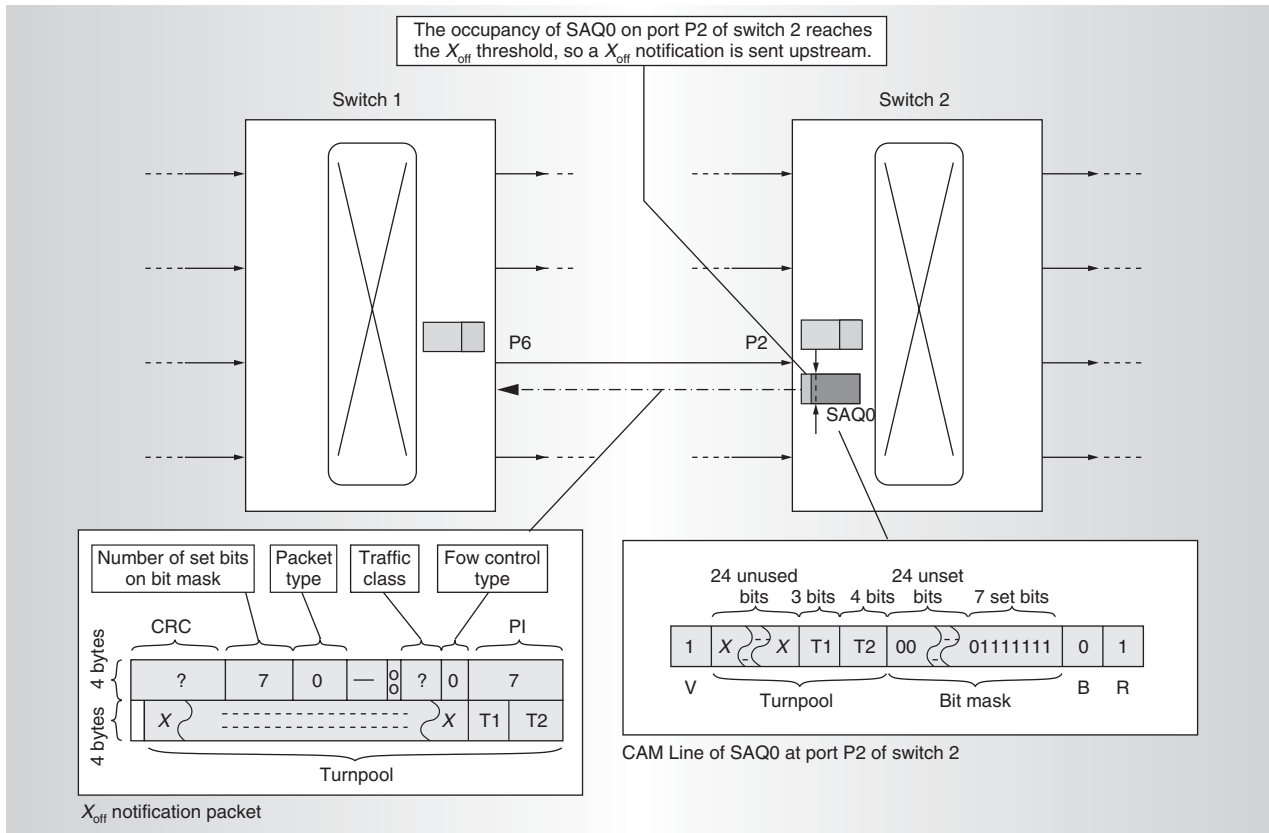


Figure 9. RECN  $X_{off}$  notification between ports at different switches.

information required for identifying the congested root associated with the SAQ whose occupancy has reached the  $X_{off}$  threshold (taken from that SAQ's CAM line).

Similarly, whenever a SAQ at the output port reaches the  $X_{off}$  or  $X_{on}$  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. Each input port updates this routing information 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. An input port that finds a match handles the internal notification as a flow control notification, and sets the B bit of the corresponding CAM line accordingly. If the notification is an  $X_{off}$  type and there is no match, 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.

### In-order delivery

As SAQs are dynamically allocated, out-of-order delivery is a concern. Whenever a SAQ is allocated to store packets passing through a congested root, there might already be packets following the same route in the standard queue. Thus, a packet addressed to a certain destination could overtake the packet preceding it through a newly allocated SAQ. To prevent this problem, RECN 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, preventing the SAQ from sending packets to the scheduler and the link controller. At the same time, the standard queue attaches a link pointer (pointing to the SAQ) 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), which enables the SAQ to send packets to the scheduler and the link controller.

This mechanism could introduce a certain degree of HOL blocking while the link pointer is traveling up the standard queue. However, this is not a problem because the HOL blocking affects packets that are in a SAQ, and so they are already congested.

### Resource deallocation

When a congestion tree vanishes, the SAQ associated with it must be deallocated. Of course, it is important to avoid early deallocation, which could introduce HOL blocking. RECN requires certain conditions before it detects that congestion is no longer present in a port, and only then does it deallocate the corresponding SAQ. To allow distributed SAQ deallocation, RECN requires the following conditions, which can be ascertained only from local information:

- The SAQ is empty.
- The SAQ is not blocked by the ready (R) bit at the CAM line. Thus, 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 by  $X_{off}$  flow control. The SAQ must be in the  $X_{on}$  state to be deallocated. This condition avoids SAQ deallocation while the congestion tree is near (the corresponding downstream SAQ occupancy is over the  $X_{off}$  threshold).

A SAQ's deallocation does not depend on other SAQs. As soon as a SAQ is not necessary (because the congestion tree associated with it has disappeared), RECN deallocates it. Immediately after its deallocation, the SAQ can be allocated for another congestion tree. Thus, distributed SAQ deallocation allows efficient use of network resources. The deallocation mechanism does not require any specific control message between SAQs, maintaining RECN's control packet compatibility with Advanced Switching.

### RECN robustness

RECN is a dynamic mechanism, and its effectiveness depends largely on the different parameters it uses: queue detection threshold,  $X_{on}$  and  $X_{off}$  flow control thresholds, and the number of SAQs per port. These para-

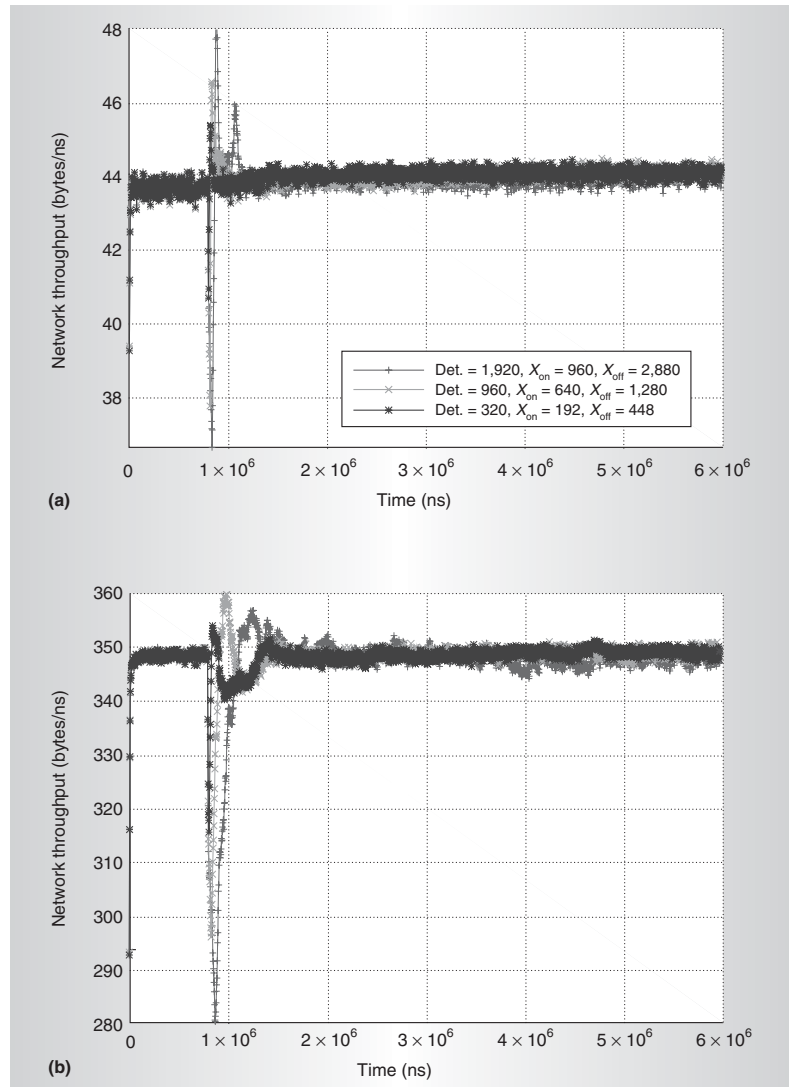


Figure 10. Network throughput for different RECN thresholds on BMINs of different sizes.

eters also determine the memory requirements at switches.

Figure 10 shows the performance achieved in networks of different sizes using RECN with different detection and flow control thresholds. The traffic load is the same used in the earlier figures, and the networks again use eight SAQs per port. Figure 10 indicates the different thresholds used by three consecutive numbers: the first corresponds to the queue detection threshold; the second and third correspond to the  $X_{on}$  and  $X_{off}$  flow control thresholds. All the thresholds are expressed in bytes.

In all cases, RECN filters the congestion. The only difference, clearly visible in large net-

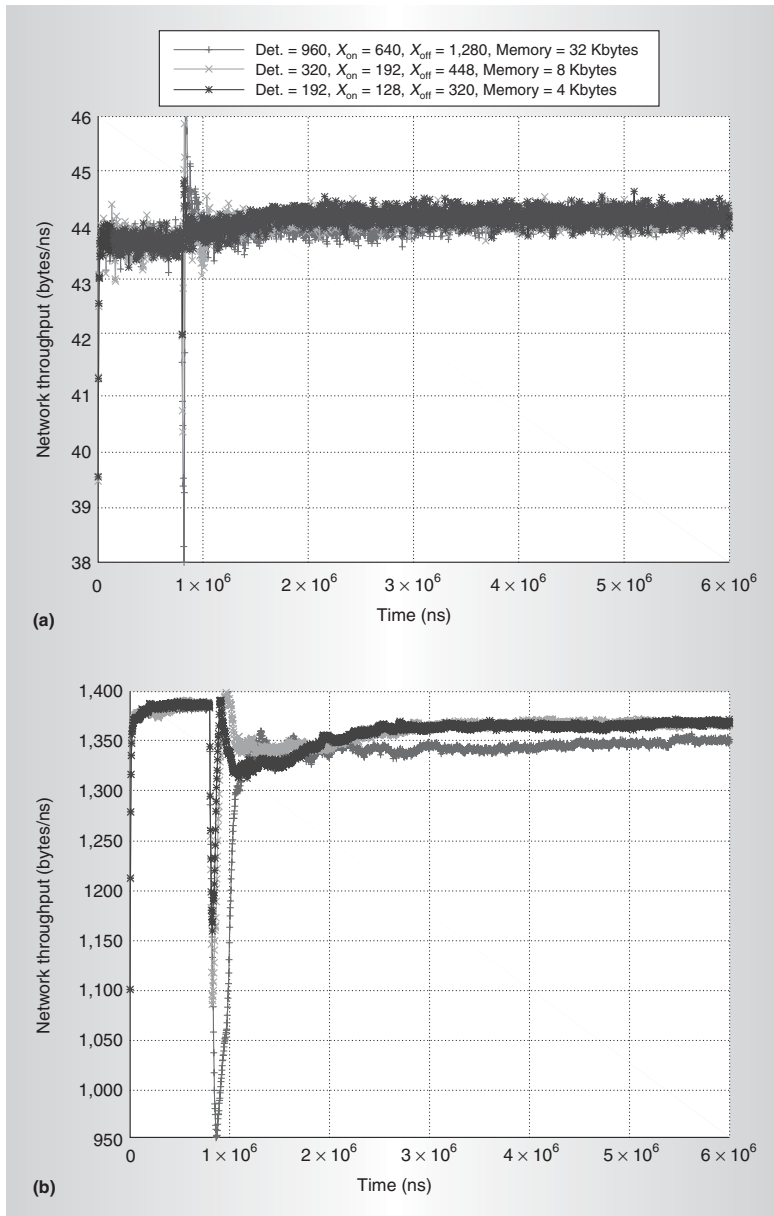


Figure 11. Network throughput for RECN on BMINs of different sizes, different port memory sizes, and different thresholds:  $64 \times 64$  BMIN (a) and  $2,048 \times 2,048$  BMIN (b).

works, is the transitory state while RECN is detecting and notifying the switches about the congested situation. Low thresholds lead to quicker response times to congestion and therefore minimize the transient HOL blocking introduced by RECN. On the other hand, 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 also reduces the memory requirements for implementing RECN. The required amount of memory at an input port depends on the detection and the  $X_{off}$  flow control thresholds, as well as the number of detection queues and SAQs. In particular, for the analyzed cases with eight-port switches, the required memory for RECN (with eight SAQs, detection threshold set to 320 bytes, and  $X_{off}$  flow control threshold set to 448 bytes) is only 6 Kbytes per input port. (At the output port, which has no detection queues, memory requirements are even lower.) Thus, using low thresholds also simplifies switch design.

Figure 11 shows the performance achieved in networks of different sizes 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. Under these conditions, performance is practically at its maximum in the presence of congestion, regardless of memory size and thresholds. Figure 11b shows the performance achieved on a very large BMIN, with 2,048 end nodes.

Finally, Figure 12 shows RECN, VOQnet, and VOQsw results for storage area network (SAN) traces on a  $64 \times 64$  BMIN. In all cases, the memory at each port was 8 Kbytes. The RECN configuration had eight SAQs per port and low detection and flow control thresholds. The I/O traces used in our evaluations were provided by Hewlett-Packard Labs (<http://ginger.hpl.hp.com/research/itc/csl/ssp>). They include all the I/O activity generated from 14 January 1999 to 28 February 1999 at the disk interface of the cello system. Because these traces are seven years old, we applied time compression factors of 20 and 40 to them. Regardless of the compression factor, RECN achieved the maximum throughput for real traffic, matching VOQnet but using fewer resources.

**R**ECN, the first congestion management technique for lossless interconnection networks to guarantee both scalability and efficiency, is currently being incorporated into

Advanced Switching specifications. Therefore, RECN will likely be implemented in future AS products. RECN achieves efficiency by quickly and accurately detecting and isolating congestion trees within the network. It thus eliminates HOL blocking introduced by congested packets and makes congestion harmless. RECN is a scalable technique because its response time is not affected by network size, it requires roughly the same resources regardless of network size, and it achieves maximum efficiency in all cases. Thus, RECN lets interconnection networks work near the saturation point without risking performance degradation. This makes it possible, for instance, for designers to reduce network connectivity by using fewer switches and links, while maintaining computational power by using the same number of attached end nodes.

#### References

1. L. Shang, L.S. Peh, and N.K. Jha, "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," *Proc. Int'l Symp. High-Performance Computer Architecture (HPCA 03)*, IEEE CS Press, 2003, pp. 91-102.
2. W.J. Dally, P. Carvey, and L. Dennison, "The Avici Terabit Switch/Router," *Proc. IEEE Symp. High-Performance Interconnects (Hot Interconnects)*, IEEE CS Press, 1998, pp. 41-50.
3. M. Katevenis, D. Serpanos, and E. Spyridakis, "Credit-Flow-Controlled ATM for MP Interconnection: the ATLAS I Single-Chip ATM Switch," *Proc. Int'l Symp. High-Performance Computer Architecture*, IEEE CS Press, 1998, pp. 47-56.
4. Y. Tamir and G.L. Frazier, "Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches," *IEEE Trans. Computers*, vol. 41, no. 6, June 1992, pp. 725-737.
5. A. Smai and L. Thorelli, "Global Reactive Congestion Control in Multicomputer Networks," *Proc. Int'l Conf. High-Performance Computing*, IEEE CS Press, 1998, pp. 179-186.
6. W.J. Dally, "Virtual-Channel Flow Control," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 2, Mar. 1992, pp. 194-205.
7. V. Krishnan and D. Mayhew, "A Localized Congestion Control Mechanism for PCI Express Advanced Switching Fabrics," *Proc.*

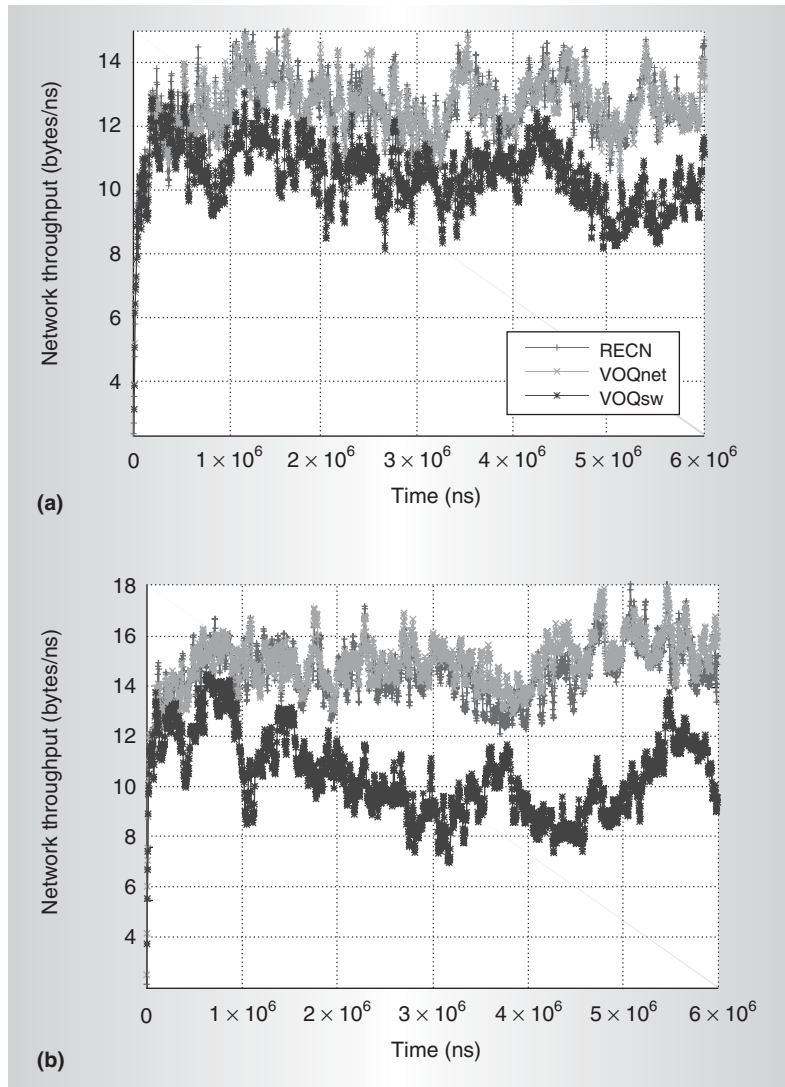


Figure 12. Network throughput, storage area network (SAN) traces on a 64 x 64 BMIN: compression factor 20 (a) and compression factor 40 (b).

8. *IEEE Symp. High-Performance Interconnects (Hot Interconnects)*, 2004; <http://www.hoti.org/hoti12/program/papers/2004/paper1.1.pdf>.
9. J. Duato et al., "A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks," *Proc. Int'l Symp. High-Performance Computer Architecture (HPCA 05)*, IEEE CS Press, pp. 108-119.
10. P.J. Garcia et al., "Dynamic Evolution of Congestion Trees: Analysis and Impact on Switch Architecture," *Proc. Int'l Conf. High Performance Embedded Architectures & Compilers (HiPEAC 2005)*, LNCS 3793, Springer, 2005, pp. 266-285.

**Pedro J. García** is an assistant professor of computer architecture and technology and a PhD candidate in the Computer Systems Department (DSI) at the University of Castilla-La Mancha, Spain. His research interests are high-performance interconnection networks—mainly congestion management and deadlock-avoidance techniques, but also reconfiguration and routing algorithms. He has a degree in communication engineering from the Technical University of Valencia, Spain.

**Francisco J. Quiles** is a professor of computer architecture and technology and Vice-Rector of Research at the University of Castilla-La Mancha. His research interests include high-performance networks, parallel algorithms for video compression, and video transmission. He has a degree in physics (electronics and computer science) and a PhD in computer science from the Technical University of Valencia, Spain.

**José Flich Cardo** is an associate professor of computer architecture and technology in the Department of Computer Engineering at Technical University of Valencia. His research interests include high-performance interconnection networks for multiprocessor systems and clusters of workstations. He has an MS and a PhD in computer science from the Technical University of Valencia, Spain. He is a member of the IEEE Computer Society.

**José Duato** is a professor in the Department of Computer Engineering at Universidad Politécnica de Valencia, Spain. His research interests include interconnection networks and multiprocessor architectures. His research results contributed to the design of the Alpha 21364 microprocessor and the Cray T3E and IBM BlueGene/L supercomputers. He has a PhD in electrical engineering from the Technical University of Valencia. He is a member of the IEEE and has served as associate editor of *IEEE Transactions on Parallel and Distributed Systems* and *IEEE Transactions on Computers*.

**Ian Johnson** is Chief Scientist at Xyratex. Previously, he worked in the Mainframe Systems Technology Development Division of ICL/Fujitsu, and he went on to found Power X, a UK company that operated in the communications sector of the IT industry. His

research interests range widely across many aspects of computing, communications, and storage systems architectures; IC design; and silicon manufacturing processes.

**Finbar Naven** is a chief architect of switch architectures at VirtenSys. His research interests include switch architectures and congestion management strategies, and he has many years experience in the architectural development and implementation of silicon devices for computing and switching. He has a BSc in electrical engineering from the University of Salford, UK.

Direct questions and comments about this article to José Flich Cardo, Universidad Politécnica de Valencia, DISCA, Escuela Técnica de Informática Aplicada, Camino de Vera s/n, 46022 Valencia, Spain; [jflich@disca.upv.es](mailto:jflich@disca.upv.es).

For further information on this or any other computing topic, visit our Digital Library at <http://www.computer.org/publications/dlib>.