Evaluation of Congestion Detection Mechanisms for InfiniBand Switches

Jose Renato Santos, Yoshio Turner, G. (John) Janakiraman
Internet Systems and Storage Laboratory
HP Laboratories Palo Alto

HPL-2002-224
August 13th , 2002*

congestion control, InfiniBand, SAN, networks, switches, ECN

InfiniBand System Area Networks using link-level flow control can experience congestion spreading, where one bottleneck link causes traffic to block throughout the network. End-to-end congestion control using Explicit Congestion Notification (ECN) packet marking has the potential to solve this problem. In this paper, we develop and evaluate ECN mechanisms for switches with input-buffered configuration (typical for InfiniBand), as opposed to traditional techniques that target output-buffered switches. Our mechanisms vary in two dimensions: the condition that triggers a marking event, and the set of packets marked at such an event. The experimental results show that our approaches avoid congestion spreading while preserving high network throughput. In addition, selective marking of packets at all input buffers on each marking event provides better fairness than a naive mechanism that marks only packets in a full input buffer. Finally, schemes that use state at both input and output ports to trigger marking events can improve fairness over purely input-triggered marking schemes.

* Internal Accession Date Only   Approved for External Publication
©Copyright IEEE
To be published in and presented at IEEE Globecom 2002, 17-21 November 2002, Taipei, Taiwan

Evaluation of Congestion Detection Mechanisms for InfiniBand Switches

Jose Renato Santos, Yoshio Turner, G. (John) Janakiraman
Hewlett Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304

***Abstract*— InfiniBand System Area Networks using link-level flow control can experience congestion spreading, where one bottleneck link causes traffic to block throughout the network. End-to-end congestion control using Explicit Congestion Notification (ECN) packet marking has the potential to solve this problem. In this paper, we develop and evaluate ECN mechanisms for switches with input-buffered configuration (typical for InfiniBand), as opposed to traditional techniques that target output-buffered switches. Our mechanisms vary in two dimensions: the condition that triggers a marking event, and the set of packets marked at such an event. The experimental results show that our approaches avoid congestion spreading while preserving high network throughput. In addition, selective marking of packets at *all* input buffers on each marking event provides better fairness than a naive mechanism that marks only packets in a *full* input buffer. Finally, schemes that use state at both input and output ports to trigger marking events can improve fairness over purely input-triggered marking schemes.**

I. Introduction

InfiniBand [1] is a new industry standard System Area Network (SAN) for interconnecting processors and I/O devices [1], [2], [3], [4]. InfiniBand networks can connect hundreds to thousands of devices over distances from a few meters to hundreds of meters. To meet application I/O requirements, InfiniBand can support end-to-end latencies (as seen by the application) under 10 microseconds and bandwidths of several Gb/s.

Congestion is an important issue in the design of InfiniBand SANs, as in any computer network [5]. InfiniBand switches do not allow packet dropping, in order to avoid long latencies for packet retransmission and the cost of packet reordering logic. Switches use link level flow control [6], [7], which prevents a switch from transmitting a packet when the downstream switch lacks sufficient buffering to receive it. Although this property prevents the well-known congestion collapse scenario [5], it may cause an undesired effect known as congestion spreading or tree saturation [8]. Congestion spreading starts at a switch when traffic demand for one of its links exceeds its capacity causing a packet buffer to fill up. This causes the link level flow control to block upstream traffic, which in turn may cause buffers in upstream switches to fill up as well. The blocking can spread further upstream and eventually fill buffers all the way back to the traffic sources. This reduces throughput on all switches in which buffers fill up. Thus, congestion spreading has the harmful effect that it can reduce the throughput of even those flows that do not exert load on the oversubscribed link.

The InfiniBand standards body [1] has formed a working group to define a congestion control mechanism for future versions of the standard. We have submitted to the working group a proposal [9] for an end-to-end congestion control scheme that consists of Explicit Congestion Notification (ECN) at switches and a window-rate control response mechanism at flow sources[1]. We believe an ECN approach is suited to the InfiniBand environment for several reasons. Packet losses are not available as indicators of congestion due to the link level flow control. Using latency as an implicit signal of congestion is also problematic, as typical InfiniBand switches with small packet buffers (e.g. 4 packets of 2KB per input port) lead to queueing delays that are similar during congestion and normal operation. It is possible to devise ECN mechanisms that have simple implementation and thus match the requirements of InfiniBand switches, which are typically single-chip devices with low cut-through switching delays [3], [11]. Finally, the use of fast switches and short links with low propagation delay allows congestion feedback to be returned quickly to flow source endpoints, which in response adjust their packet injection.

In this paper we use simulation to evaluate key design choices for the ECN-based congestion detection and notification component of end-to-end congestion control for InfiniBand. With the proposed mechanisms, InfiniBand switches identify the packets that cause congestion and mark them using a traditional ECN bit in the packet header. When a destination endpoint (i.e., network interface) receives a packet, it piggybacks the ECN value onto a short ACK packet that is returned to the received packet's source endpoint as congestion feedback. Three variations of the ECN mechanism are devised that we call *naive*, *input-triggered*, and *input-output-triggered*. These mechanisms vary in two design dimensions: the condition that triggers packet marking, and the set of packets marked at such a marking event.

Our primary goal for congestion control is to avoid congestion spreading while delivering high throughput. A second goal is to provide approximate throughput fairness to flows that compete for a bottleneck link. We evaluate how

[1] Our proposal [9] suggests a hybrid window and rate control mechanism using the well-known AIMD [10] function or a more efficient variation for this environment that we term LIPD. The simulation results in this paper all use LIPD rate control and a fixed window size of one packet, which is appropriate for typical InfiniBand networks because one packet is approximately equal to the network bandwidth-delay product. Our proposal describes how to handle details of InfiniBand such as heterogeneous links, ACK coalescing, variable packet size, unreliable transport, etc., that are out of the scope of this paper.

well the mechanisms meet these goals in the context of input-buffered switches. Since InfiniBand switches operate at very high speeds, they are usually configured with buffers at the input ports[2] [12]. To identify packets causing congestion, input-buffered switches can benefit from approaches that differ from traditional techniques [13], [14] which are aimed at output-buffered switches. For input-buffered switches, our *input-triggered* and *input-output-triggered* ECN mechanisms support better fairness properties than the traditional *naive* approach of marking packets in a full buffer.

II. Related Work

For networks that use link-level flow control, *hop-by-hop* congestion control has been proposed which limits the number of packets at a switch that share a common output link or final destination [15], [8]. To enforce the limits, switches must implement a substantially enhanced link flow control mechanism. In contrast, our approach aims to keep switch design simple and empower flow endpoints to control traffic injection rates.

For traditional networks, such *end-to-end* control is exemplified best by TCP, in which flow sources use endpoint detection of packet dropping [5] or changes in network latencies [16], [17] as an implicit signal of congestion. An alternative to implicit notification is Explicit Congestion Notification (ECN), in which switches detect incipient congestion and notify flow endpoints, for example by marking packets when the occupancy of a switch buffer exceeds a desired operating point [13], [14]. Enabling switches to mark packets slightly increases switch implementation complexity. ECN is used in ATM networks [18], and it has been proposed for use with TCP [19], [20]. These approaches assume switches with output buffer configurations while we consider switches with input buffer configurations. The different buffer organization of switches requires different approaches for identifying packets contributing to congestion.

III. Congestion Spreading

In this section, to motivate the need for congestion control, we show the harmful effect of congestion spreading. In order to illustrate this effect and to evaluate the performance of our ECN mechanisms, we conducted a series of simulation experiments using an example scenario that is shown in Fig. 1 and which we use for all results presented in this paper. Table I shows the parameters used in the simulations. Our simulation topology consists of two switches A and B connected by a single link. The traffic is generated by a set of 10 *local* flows generated at endpoints B_1 through B_{10}, 10 *remote* flows generated at endpoints A_1 through A_{10}, and a *victim* flow generated at endpoint A_V. All remote and local flows are destined to endpoint B_C through a congested output link on switch B. The victim flow is destined to a non-congested endpoint B_V and suffers from congestion spreading. All flows are greedy, i.e. flows try to use all the network bandwidth that they can. Congestion spreading originates at the oversubscribed link connecting switch B

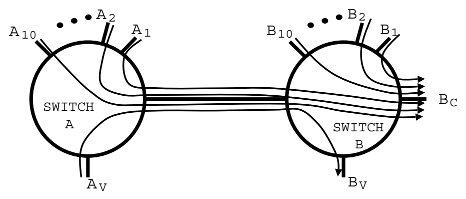


Fig. 1. Simulation scenario

TABLE I
Simulation Parameters

parameter	default value (unless otherwise specified)
link bandwidth	1 GB/sec (InfiniBand 4X links)
packet header	20 bytes (InfiniBand Local Header)
data packet size	20 + 2048 = 2068 bytes
data packet tx time	2.068 μs
ACK packet	20 bytes
switch minimum forwarding delay	40 ns (header delay)
buffer configuration	input port
buffer size	4 packets/port
switch scheduling	FIFO with possible bypass of older packets when crossbar is busy (max bypass: 4)

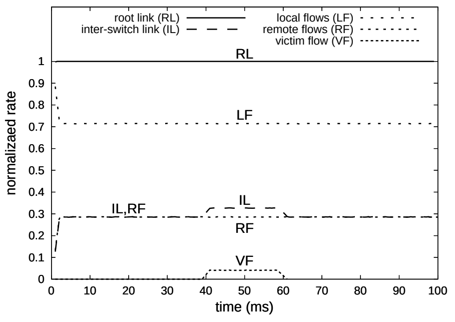


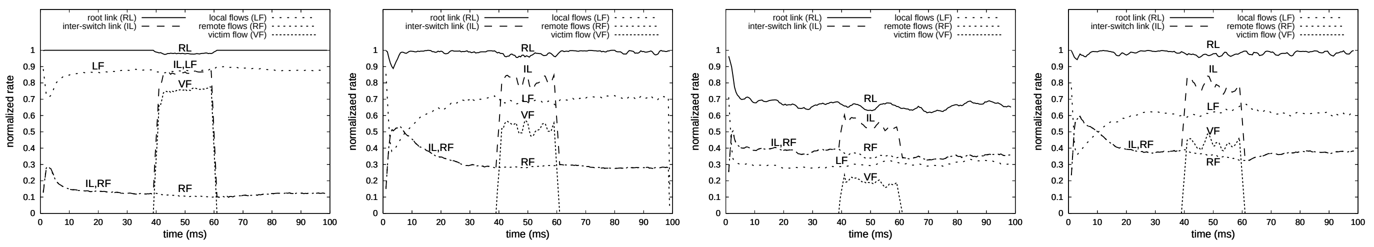
Fig. 2. Congestion Spreading: fixed window = 1 (buffer capacity = 4 packets)

to endpoint B_C which we refer to as the *root* link of the congestion spreading tree.

Fig. 2 shows the results of a simulation for the scenario shown in Fig. 1, when no congestion control is used. The experiment simulates the example scenario for a period of $100ms$. At the begining of the simulation, local and remote flows start sequentially every $100\mu s$, with the local flows starting before the remote flows. The local and remote flows remain active until the end of the simulation, while the victim flow is active only in the time interval [$40ms$,$60ms$]. The graph shows the traffic rate on the root link and on the inter-switch link, as well as the (aggregate) rates of local flows, remote flows, and the victim flow. Rates are computed considering the number of packets transmitted in a sliding time window of duration $2ms$ centered on the corresponding time point.

The results reveal that the victim flow uses only 4% of the bandwidth on the inter-switch link, even though the inter-switch link is only 32.5% utilized. Since the link to destination B_C is oversubscribed, the buffers at switch B (at the input port for the inter-switch link) fill with packets and block incoming flows, causing the inter-switch link to go idle. If each remote flow did not attempt to transmit at the full link bandwidth and instead

[2]Other buffer configurations, such as central or output buffer, require internal switch data transfer rates higher than the link speed to service multiple packets that can arrive simultaneously from different input ports, increasing the challenge of designing for very high link speeds.



(a) *naive*
Mark packets in a full buffer

(b) *input-triggered*
Mark *generating* packets

(c) *input-output-triggered*
Mark *generating* packets
output threshold = 4

(d) *input-output-triggered*
Mark *generating* packets
output threshold = 8

Fig. 3. Comparison of Packet Marking Policies: buffer capacity $= 4$ packets.

proactively reduced its rate to the rate determined by the bottleneck link, i.e. $\frac{1}{20}$ of the link bandwidth, the buffers at switch B would not fill up and the victim flow would be able to utilize the available bandwidth at the inter-switch link, improving the network throughput.

IV. Congestion Detection and Notification: Packet Marking

This section describes our three ECN packet marking mechanisms: *naive*, *input-triggered*, and *input-output-triggered* marking.

A. Naive Packet Marking Mechanism

In the *naive* scheme, switches detect congestion when a buffer becomes full[3], since a full buffer propagates congestion by blocking the upstream switch from transmitting. After detecting congestion the switch must identify the packets that are *generating* congestion, i.e. packets that are transmitted on an oversubscribed link (root). Without congestion control, packets in full buffers may be *generating* packets destined for a busy root link or they may be *victim* packets that are waiting for links blocked by a downstream root link. With congestion control, our simulation results, presented later, show that congestion spreading is reduced to very low levels, and the time a link spends in a blocked state is insignificant. Since blocked links are rare, so are victim packets that wait for them. Thus we simply treat *any* packet in a full buffer as a *generating* packet, regardless if the link for which the packet is waiting is blocked or not[4].

In a switch with output buffer configuration, the packets in full buffers are the only generating packets at the switch. Thus a naive mechanism, which simply marks all packets in a buffer whenever an arriving packet fills it up, successfully marks all generating packets. In switches with input buffers (typical for SANs), other packets at the switch besides those in a full buffer may be generating packets, because they all contend for the same root link. In this case, the naive mechanism fails to mark some generating packets. Simulation results for the naive mechanism using input-buffered switches are shown in Fig. 3(a). The results show that although the naive mechanism can avoid congestion spreading and allow the victim flow to receive high throughput, it results in an unfair allocation of rates between remote and local flows. While the average throughput is approximately the same among flows of the same type, local or remote (this is not shown in Fig. 3(a)), the local flows utilize 90% of the available root link bandwidth. This unfairness is a consequence of the selection of packets to be marked. Packets of remote flows are marked when they collectively fill the input buffer at switch B that receives packets from the inter-switch link. In contrast, none of the packets of the local flows is marked since each local flow uses a different input buffer and the window limit prevents it from filling the buffer. That penalizes the remote flows, which have their rate reduced while the local flows take a disproportionate share of the congested link bandwidth. In general, the naive mechanism penalizes flows that arrive at a switch competing for an oversubscribed link through an input port shared with many competing flows.

B. Input-Triggered Packet Marking Mechanism

We propose a marking mechanism for input-buffered switches that promotes fairness by marking all generating packets at the switch. Our marking approach operates in three steps. First, as in the naive approach, a switch input buffer triggers packet marking each time it becomes full. Second, any output link that is the destination for at least one packet in such a full buffer is classified as a congested link. Third, all packets that are resident (in any buffer) at the switch and are destined to an output link that was classified as congested in the second step are classified as generating packets and marked[5]. We refer to this marking mechanism as *input-triggered*. The third step seems to require an expensive scan of all input buffers in a switch even when only one becomes full. We specify an efficient implementation that does not require this scan. The implementation does not mark packets immediately after an input buffer becomes full, but waits to mark them at the time of their transmission, avoiding the scan. In order to determine the

[3]With the current use of small buffers in SAN switches, a lower buffer size threshold is likely to only reduce link utilization by causing the buffer to empty more frequently. If switch buffers become larger, using a buffer size threshold below the maximum capacity might be beneficial by preventing congestion spreading before its occurrence while preserving high utilization.

[4]In fact, we simulated various scenarios using a marking policy that does not mark packets that are transmitted on a link that was recently blocked and confirmed that the results were identical to those obtained with the mechanism presented in this paper.

[5]Our design choices favor simple mechanisms that can be easily implemented in low cost fast switches and avoid solutions that require complex instrumentation and parameter tuning, such as for example congestion detection based on a time averaged buffer occupancy threshold or time averaged link utilization.

number of packets that should be marked, we use two counters for each output link. The first counter *cnt1* records the current number of packets in the switch that are waiting for that output link; *cnt1* is incremented and decremented as packets enter and leave the switch. The second counter *cnt2* records the number of next packets that need to be marked when transmitted on that output link. Counter *cnt2* is initialized to zero. Whenever a buffer becomes full, the value of counter *cnt1* is copied to counter *cnt2*. Then, the output port starts marking the next transmitted packets, decrementing *cnt2* at each transmission, until it reaches zero again. Note that different input buffers can trigger marking on the same output port in a short time interval, causing counter *cnt2* to be updated with the current value of counter *cnt1* a second time before it reaches zero. In fact, this can also happen if the same input buffer becomes full a second time triggering the marking of all new packets that arrived since the last marking event, in addition to the ones already marked and still in the input buffer.

Note that this counter implementation may mark a different set of packets than a direct packet scanning approach, since packets can be transmitted out of order. This turns out to be an advantage, since our implementation will mark the first packets to leave the switch and provide faster feedback to network endpoints.

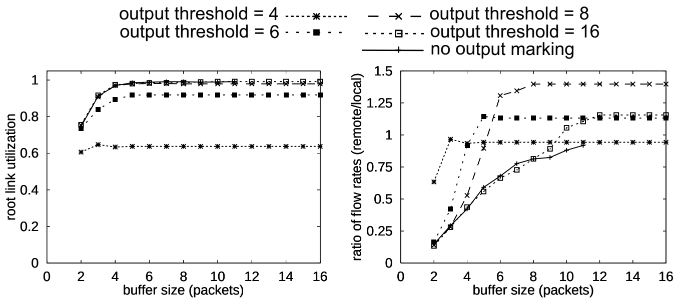
Fig. 3(b) shows the simulation results obtained for input-triggered marking. The results show that this marking policy also avoids congestion spreading and keeps the inter-switch link at high utilization. Moreover, fairness between the remote and local flows is improved when compared to the naive scheme. This is expected since the mechanism marks all generating packets, both from remote and local flows.

Unfairness is not entirely eliminated with this marking policy because the event that triggers packet marking (a full input buffer) is biased to preferentially mark remote flows. Marking is triggered at times that sample the *peak* buffer usage for the remote flows and only the *average* buffer usage for the local flows. In input-triggered marking, the number of packets of remote flows that are marked is approximately equal to the number of input buffer slots[6]. In contrast, for the local flows the marking scheme samples a distribution of buffer usage over the whole range from zero usage to the peak usage. A fair state, in which local and remote flows have the same rate limits, is not stable because in that state the packet marking bias tends to mark more packets of remote flows than of local flows, reducing the rate limits of each remote flow more frequently than for each local flow.

C. *Input-Output-Triggered Packet Marking Mechanism*

For improved fairness among the flows contending for the congested link, the input-triggered marking mechanism can be combined with an additional *output-triggered* mechanism that is triggered when the total number of packets that are waiting

[6]It is not exactly the number of buffer slots, because sometimes a victim flow may be using one of the buffer slots or a packet in the buffer is being transmitted and cannot be marked anymore. However the probability of having a victim packet in a full buffer is very small, since most of the time the victim can cut through and start being transmitted to its output port just after its header is received, occupying the buffer just for a short period of time.



(a) Root link utilization (b) Fairness

Fig. 4. Root link utilization and fairness between remote and local flows as a function of buffer size, for various output thresholds (including none – no output marking). Scenario in Fig. 1.

for an output link exceeds a threshold. We refer to this combined marking mechanism as *input-output-triggered*. We do not adopt a pure output-triggered mechanism, because it would allow congestion spreading caused by a full input buffer to occur without being detected.

Output-triggered marking events tend to preferentially mark local flows. Marking is triggered by a burst of arriving packets that cannot be served immediately by the single output link. The serialization of remote packets in the shared inter-switch link reduces the burstiness in the arrival of remote packets, which reduces their probability of participating in a burst that causes an output-triggered marking event. Bursts of packets from the local flows are more likely since local flows arrive in parallel on independent input links.

The performance of the combined approach, which attempts to balance two opposing biases, varies with the output threshold, the buffer capacity, and the traffic pattern (distribution of the flows among the input buffers). We present preliminary results that show the impact of output threshold and buffer capacity on the link utilization and fairness. Fig. 3(c) and Fig. 3(d) show the performance of the combined approach using thresholds of four and eight packets for each output link. The results show that the flow rates for the remote and local flows are closer with the combined approach than with solely input-triggered marking. With a small output threshold of 4 packets (Fig. 3(c)), packets are too frequently marked before an input buffer fills, resulting in low utilization of the congested link. With a higher threshold of eight packets (Fig. 3(d)), the root link has high utilization without congestion spreading. Although fairness is degraded with the higher threshold, it is still better than that achieved by the input-triggered marking mechanism.

Fig. 4 shows the results of several simulations that vary the buffer size and the output threshold. Each simulation is run for *500 ms* simulated time, and the reported results average the rates over the last *400 ms*. Fig. 4(a) shows the root link is underutilized with a low output threshold (4) or small input buffer size (2 to 4). Under-utilization results from overly severe packet marking. Using a slightly higher output threshold (6) enables high root link utilization ($> 90\%$), except at the smallest buffer sizes. Fig. 4(b) shows the fairness of flows contending for the oversubscribed root link as the ratio of aggregate rates of remote flows to local flows. The fairness results are discussed below:

- At buffer sizes much lower than the output threshold, input-triggered marking events dominate, which preferentially marks remote flows, resulting in a reduced ratio of *remote/local* flow throughput.

- When input-triggered marking dominates, fairness is improved as the buffer size increases. This can be explained as follows. If the arrival process and mean rate of remote packets did not change with buffer size, a larger buffer would be less likely to become full. This would reduce the frequency of marking events, but each marking event would mark a larger number of packets. The net effect would be a reduction in the mean rate of marks, since the first component should decrease faster than linearly[7] while the second component should increase linearly with the buffer size. This would cause all flows to receive a lower frequency of marks and reach higher rates. But the mean flow rate cannot exceed the link bandwidth. Thus the new operating point should correspond to a higher mean arrival rate for remote packets, which increases the fraction of root link bandwidth that is used by the remote flows.
- At buffer sizes larger than the output threshold, marking is always output-triggered[8]. Hence the fairness ratio in Fig. 4(b) does not change with buffer size.
- Local flows receive a lower share of the throughput as the output threshold increases from 4 to 8, for a fixed and large buffer size, because a higher threshold requires larger bursts which can only be generated by local flows. Thus the number of local packets marked in an output-triggered marking event is increased for larger thresholds while the number of remote packets marked is not, decreasing the rate of local flows more strongly and reducing fairness.
- When the output threshold is increased to even larger values (e.g. 16), the number of local flows (10) cannot themselves trigger the marking. Output-triggered marking events are triggered only when there are sufficient remote packets as well. This reduces the difference between the number of marked packets from local and remote flows, increasing fairness.

We also conducted experiments varying the number of remote and local flows from 5 to 20 (equal number of each type) and obtained results similar to the ones shown in Fig. 4.

V. Conclusions

We proposed and evaluated three mechanisms for detecting congestion at input-buffered switches. The mechanisms are designed to be simple and inexpensive to implement in high speed switches. All three schemes were shown to eliminate congestion spreading, a consequence of link level flow control, and at the same time enable high network throughput. The performance results show that, with input-triggered marking, selectively marking at all input buffers packets that are destined to a congested link achieves better fairness than a naive scheme that marks only packets in a full input buffer. To further improve fairness we investigated an approach that combines input-triggered marking with output-triggered marking. The results show that high link utilization is achieved even with small input buffers, and fairness can be improved by balancing the opposing impacts of input-triggered and output-triggered packet marking.

The proposed congestion detection mechanisms described here were used as part of a complete end-to-end congestion control mechanism proposed for InfiniBand networks that is described in [9], and evaluated with various alternative source response functions in [21]. For future work, we plan to further study and explain how network performance is impacted by complex interactions between the ECN detection mechanisms and the various alternative source response functions.

[7]This is equivalent to the blocking probability of queueing systems with finite queue capacity, which is known to decrease faster than linearly with the queue capacity.

[8]For buffer sizes 12 and greater, all marking is output-triggered regardless of output threshold, because the input buffer never becomes full, since there are only 10 remote flows and one victim flow sharing the inter-switch link.

References

[1] InfiniBand^{SM} Trade Association, *InfiniBand^{TM} Architecture Specification Volume 1,* Release 1.0. (www.infinibandta.org), Oct. 2000.

[2] R. W. Horst, "TNet: a reliable system area network," *IEEE Micro*, vol. 15, no. 1, pp. 37–45, Feb. 1995.

[3] W. Baker, R. Horst, D. Sonnier, and W. Watson, "A flexible ServerNet-based fault-tolerant architecture," in *25th Intl. Symp. Fault-Tolerant Computing*, June 1995, pp. 2–11.

[4] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. Seitz, J. N. Seizovic, and W.-K. Su, "Myrinet: a gigabit-per-second local area network," *IEEE Micro*, vol. 15, no. 1, pp. 29–36, Feb. 1995.

[5] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM*, Aug. 1988, pp. 314–329.

[6] W. J. Dally, "Virtual-channel flow control," *IEEE Tr. Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, Mar. 1992.

[7] H. T. Kung, T. Blackwell, and A. Chapman, "Credit-based flow control for ATM networks," *ACM SIGCOMM*, vol. 24, no. 4, pp. 101–114, Aug. 1994.

[8] D. M. Dias and M. Kumar, "Preventing congestion in multistage networks in the presence of hotspots," in *International Conference on Parallel Processing*, Aug. 1989, pp. 1.9–1.13.

[9] Y. Turner, J. R. Santos, and G. Janakiraman, "An approach for congestion control in InfiniBand," Tech. Rep. HPL-2001-277, HP Labs, Oct. 2001.

[10] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, June 1989.

[11] Mellanox Technologies Inc., *InfiniScale^{TM}, Mellanox's 2nd Generation Switch,* (www.mellanox.com/news/articles/intro.pdf), Oct. 2001.

[12] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz, "Tiny Tera: A packet switch core," *IEEE Micro*, vol. 17, no. 1, pp. 26–33, Jan. 1997.

[13] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," *ACM Tr. Computer Systems*, vol. 8, no. 2, pp. 158–181, May 1990.

[14] S. Floyd and V. Jacobson, "Random Early Detection gateways for congestion avoidance," *IEEE/ACM Tr. Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[15] L. Cherkasova, A. Davis, R. Hodgson, V. Kotov, I. Robinson, and T. Rokicki, "Components of congestion control," in *ACM Symposium on Parallel Algorithms and Architectures*, 1996, pp. 208–210.

[16] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[17] C. Parsa and J. J. Garcia-Luna-Aceves, "Improving TCP congestion control over internets with heterogeneous transmission media," in *7th International Conference on Network Protocols (ICNP'99)*. oct-November 1999, pp. 213–221, IEEE Computer Society.

[18] N. Golmie, Y. Saintillan, and D. Su, "ABR switch mechanisms: design issues and performance evaluation," *Computer Networks and ISDN Systems*, vol. 30, pp. 1749–1761, 1998.

[19] K. K. Ramakrishnan, S. Floyd, and D. Black, "The addition of Explicit Congestion Notification (ECN) to IP," in *RFC 3168*, Sept. 2001.

[20] S. Floyd, "TCP and explicit congestion notification," *Computer Communication Review*, vol. 24, no. 5, pp. 8–23, Oct. 1994.

[21] J. R. Santos, Y. Turner, and G. Janakiraman, "End-to-end congestion control for system area networks," Tech. Rep. HPL-2002-4, HP Labs, Jan. 2002.