End-to-End Congestion Control for InfiniBand

Jose Renato Santos, Yoshio Turner, G. (John) Janakiraman

Hewlett Packard Laboratories

1501 Page Mill Road

Palo Alto, CA 94304

Abstract— **InfiniBand System Area Networks (SANs) which use link-level flow control experience congestion spreading, where one bottleneck link causes traffic to block throughout the network. In this paper, we propose an end-to-end congestion control scheme that avoids congestion spreading, delivers high throughput, and prevents flow starvation. It couples a simple switch-based ECN packet marking mechanism appropriate for typical SAN switches with small input buffers, together with a source response mechanism that uses rate control combined with a window limit. The classic fairness convergence requirement for source response functions assumes network feedback is synchronous. We relax the classic requirement by exploiting the asynchronous behavior of packet marking. Our experimental results demonstrate that compared to conventional approaches, our proposed marking mechanism improves fairness. Moreover, rate increase functions possible under the relaxed requirement reclaim available bandwidth aggressively and improve throughput in both static and dynamic traffic scenarios.**

I. INTRODUCTION

InfiniBand [1] System Area Networks (SANs) [2][3][4], which provide high throughput and low latency for efficient I/O and cluster communication, can have persistent traffic congestion that severely limits performance. They can experience congestion spreading [5], where one bottleneck link causes traffic to block throughout the network. These networks will experience these congestion events frequently given their large network size (number of devices), application diversity (e.g., for storage as well as interprocess communication), and lower host overheads that enable high applied loads on the network fabric.

Congestion control has been widely studied in traditional networks, such as Local Area (LANs) and Wide Area Networks (WANs). The unique characteristics of InfiniBand SANs, however, make the congestion control problem unique in this environment:

1) No packet dropping at switches

   InfiniBand 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. This property prevents packet dropping at switches and avoids the well-known congestion collapse scenario of traditional networks [8], but it may cause an undesired effect known as congestion spreading or tree saturation [5], which is discussed in detail in Section III. A consequence of this characteristic is that packet losses cannot be used as indication of congestion.

2) Low network latencies

   Due to cut-through routing at switches and short link distances, network latencies in empty networks are very small (on the order of 100s of nanoseconds). Thus switch logic, including any support required for congestion control, must be simple enough to be implemented in hardware. Low network latency results in a relatively small bandwidth-delay product (usually less than one Kilobyte) and a flow can use all the available bandwidth on its network path with a small number of bytes in transit at any time, even less than one packet. In this environment, a traditional window control mechanism as used by TCP [8] is inadequate for controlling flow rates.

3) Low buffer capacity at switches

   InfiniBand switches are typically single-chip devices [3][9] with small packet buffers. A typical InfiniBand switch design, that we are aware of, can hold 4 packets of 2KB per port. Therefore, congestion can occur even when the number of flows contending for a single link is small. In addition, with small buffers, queueing delays during congestion can be on the same order of magnitude as queueing delays in normal operation. Thus it is difficult to rely on network latency as an implicit signal of network congestion.

4) Input-buffered switches

   Since InfiniBand switches operate at very high speeds, they are usually configured with buffers at the input ports[1] [10]. To identify packets causing congestion,

---

[1]Other buffer configurations, such as central or output buffer, require internal switch data transfer rates higher than the link speed to service

input-buffered switches may benefit from approaches that differ from traditional techniques [11][12] which are aimed at output-buffered switches.

In this paper we propose an end-to-end congestion control scheme for InfiniBand that consists of an ECN packet marking mechanism at switches and a source response mechanism that combines rate control with a window limit[2]. We also propose source response functions that achieve higher bandwidth utilization than traditional approaches by exploiting the asynchronous behavior of packet marking.

The main contributions of this paper are summarized as follows:

1) Congestion control solution suited to InfiniBand SAN environment

- We propose a novel ECN packet marking mechanism for input-buffered InfiniBand switches. An ECN approach was adopted mainly because packet losses cannot be used as indication of congestion and network latencies cannot be effectively used to distinguish normal traffic conditions from network congestion. For input-buffered switches, our approach has better fairness properties than the traditional approach of simply marking packets in full buffers (appropriate for output-buffered switches). In addition, our ECN mechanism is simple to implement in hardware.

- We propose a source response mechanism that is best suited to a SAN environment with low bandwidth-delay product and low buffer capacity. The mechanism combines rate control with a window limit to provide the wide range of operating points and low buffer utilization associated with a flow rate control mechanism and the self-clocking property of a window limit.

2) Novel rate control source response functions

- We derive a new set of conditions for the design of source response functions. The new conditions exploit a bias of asynchronous packet marking for high rate flows in order to weaken the convergence requirements previously proposed by Chiu and Jain [14]. The new conditions allow the use of source response functions that achieve higher bandwidth utilization than is possible with

the stricter requirements while ensuring congestion avoidance and fairness.

- We propose two new source response functions that satisfy the above properties and demonstrate their advantages over the traditional AIMD (Additive Increase Multiplicative Decrease) response function in a SAN environment, using simulation.

The rest of this paper is organized as follows. Related approaches to congestion control are briefly summarized in Section II. Section III motivates the need for congestion control in a SAN environment, by showing the harmful effect of congestion spreading in a simple scenario. The details of the proposed congestion control mechanism are discussed in Section IV. Section V presents our source response function design methodology and specific response functions. Section VI presents simulation results for our mechanisms, and Section VII presents our conclusion.

II. RELATED WORK

Hop-by-hop congestion control, which limits the number of packets at a switch that share a common output link or final destination, has been proposed for networks that use link-level flow control [15][5]. 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 easy to implement in hardware, and adopts an *end-to-end* mechanism that relies on 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 [8] 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 [11][12]. 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.

A source of traffic should adjust packet injection in response to congestion information. The most widely used response function is Additive Increase Multiplicative Decrease (AIMD), which has been shown to converge to fairness under an assumption of synchronized feedback to flow sources [14]. AIMD has been used for both window control [8] and rate control [21]. Recently, other response functions aimed largely at multimedia streaming applications have been investigated that attempt to be compatible

---

multiple packets that can arrive simultaneously from different input ports, increasing the challenge of designing for very high link speeds.

[2]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 our proposal [13] to the working group. Our proposal addresses additional issues not discussed here, such as, heterogeneous links, ACK coalescing, variable packet size, unreliable transport, etc.
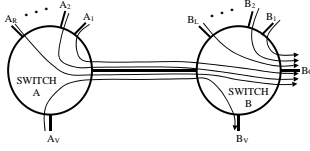
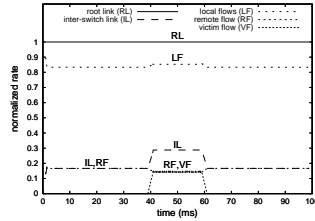Fig. 1. Simulation scenario

Fig. 2. Congestion Spreading

with TCP without suffering the large fluctuations in injection rate that can arise from the multiplicative decrease of AIMD [22][23]. We propose source response functions based on more relaxed fairness convergence requirements, that can reclaim available bandwidth faster than response functions that satisfy the traditional convergence requirement (as for example, the traditional AIMD), increasing the effective network throughput in a dynamic environment in which flows come and go.

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 congestion control scheme, 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 *L local* flows generated at endpoints B_1 through B_L, *R remote* flows generated at endpoints A_1 through A_R, 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 to endpoint B_C which we refer to as the *root* link of the congestion spreading tree.

To illustrate the problem caused by congestion spreading, we consider the scenario shown in Figure 1 with 5 local flows and 1 remote flow ($L = 5$, $R = 1$) for a switch buffer that can hold 4 packets per input port. Figure 2 shows the results of a simulation for this scenario, when no congestion control is used. The experiment simulates the example scenario for a period of $100ms$. At the beginning of the simulation, local and remote flows start sequentially every $100\mu s$, with the local flows starting before the

remote flow. 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 flow, 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 15% of the bandwidth on the inter-switch link, even though the inter-switch link is only 30% 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 the remote flow did not attempt to transmit at the full link bandwidth and instead proactively reduced its rate to the rate determined by the bottleneck link, i.e. $\frac{1}{6}$ 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 CONTROL MECHANISM

This section describes the two components of our proposed congestion control mechanism for InfiniBand: an ECN packet marking mechanism, and a source response mechanism that combines rate control with a window limit.

A switch detects and identifies packets which are contributing to congestion. The switch sets a single bit ECN field in the header of an identified packet to indicate the occurrence of congestion to the destination. The destination returns the ECN value in the acknowledgment packet and the source uses this information to adjust its packet injection rate.

A. Packet Marking

Congestion is propagated by a switch full buffer since it blocks an upstream switch from transmitting. Therefore, a naive but straightforward way for switches to detect and indicate the occurrence of congestion would be to mark all packets in a buffer whenever it becomes full [3]. In a switch with output buffer configuration, this approach successfully marks all packets that are transmitted on the root link of a congestion spreading tree. In a switch with input buffers (typical for InfiniBand), however, other packets at the switch besides those in a full buffer may be generating congestion by contending for the same root link. As we show later in the simulation results of Section VI-A, the failure of the naive approach to mark those additional packets results in unfairness among flows contending for the root link.

We propose a marking mechanism for input-buffered switches that promotes fairness by marking all packets at the switch that are generating congestion by contending for a busy root link. The mechanism 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 marked[4]. 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. Instead, it determines the number of packets that should be marked and marks them at the time of their transmission, avoiding the scan. For this purpose, 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 subsequent 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 marking can be re-triggered on the same output port even before counter *cnt2* reaches zero. The implementation will operate correctly even in such cases by triggering the marking of all new packets that arrived since the last marking event, in addition to the packets previously identified.

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.

Our proposed packet marking mechanism and descriptions of additional schemes are discussed in more detail in [24].

B. Source Response: Rate Control with a Window Limit

The source response mechanism controls the injection of packets into the network in response to ECN information delivered to the source via ACKs.

Window-based congestion control is a common approach which adjusts the number of outstanding packets for a flow based on the congestion feedback. A window-based mechanism offers the benefit that packet injection is self-clocked [8], and it limits the amount of buffer space that a flow can consume in the network. The range of useful window sizes is very small in an InfiniBand SAN environment since its bandwidth-delay product is small. For example, a network with 1 GByte/sec links, 64 ns per-switch forwarding delay, and a diameter of 32 switches has a bandwidth-delay product equal to just one 2048-byte packet. Thus a flow that is limited to a window of size one (packet) is able to use most of the bandwidth on its path in an otherwise empty network. A window of size two completely saturates the bandwidth (the ACK for one packet is returned in parallel with the transmission of a second packet). For example, Fig. 3(b) shows that when each flow in Fig. 2 is limited to a window size of one packet, high link utilization can be sustained while eliminating congestion spreading. Since only two flows share the inter-switch link, the window ensures at most two packets reside at switch B's input buffer for the inter-switch link and the link is never blocked. Hence the inter-switch link is fully utilized and the victim flow consumes all its idle bandwidth (the slight under-utilization of the root link and the inter-switch link is an artifact of the starvation prevention function of switch B's scheduling mechanism).

---

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

[4]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.

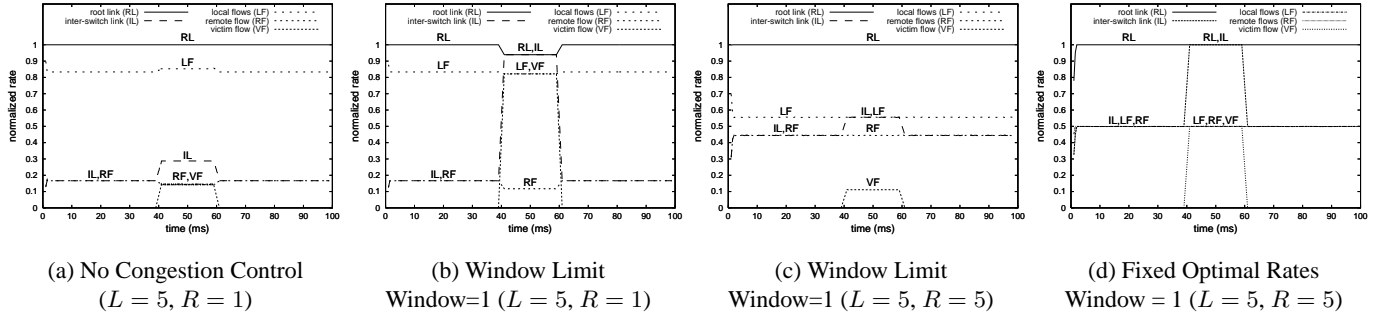| (a) No Congestion Control $(L = 5, R = 1)$ | (b) Window Limit Window=1 $(L = 5, R = 1)$ | (c) Window Limit Window=1 $(L = 5, R = 5)$ | (d) Fixed Optimal Rates Window = 1 $(L = 5, R = 5)$ |

Fig. 3. Congestion Spreading: fixed window vs. rate control (buffer capacity = 4 packets).
(Fig. 3(a) is the same as Fig. 2 and is shown here for convenient comparison)

However, such a window-based mechanism is inadequate for InfiniBand SANs for two reasons. First, the small range (one to two packets) of useful window sizes severely limits the flexibility of the congestion control[5]. Second, even with the use of a minimal per-flow window of size one packet, congestion spreading can occur when the number of contending flows exceeds the number of buffer slots. That condition can occur easily in InfiniBand networks where switch buffer sizes are small. Fig. 3(c) shows that when there are five local flows and five remote flows ($L = 5$, $R = 5$, and a buffer with 4 packet slots) in the scenario of Fig. 1, with each flow limited to a window size of one packet, congestion spreading prevents the victim flow from achieving high throughput and the inter-switch link is under-utilized.

Congestion spreading can be avoided in scenarios where numerous flows contend for a smaller number of buffer slots if the average buffer utilization in the network is maintained at *less* than one packet per flow. This cannot be achieved by a pure window control mechanism, since the minimum window size is one packet. In contrast, a rate control mechanism can satisfy this requirement. Rate control also greatly increases the range of control compared to pure window-based control in the InfiniBand environment. We therefore propose the use of rate control. In order to preserve the self-clocking benefits of a window, we further propose to maintain a fixed window size of one packet in addition to the use of rate control. A variable window limit may be beneficial when ACKs suffer short delays in the reverse path. We plan to investigate this case as future work.

Fig. 3(d) shows simulation results that illustrate the potential for rate control to improve performance over the results in Fig. 3(c). In the experiment, the rate limit for

[5]Larger windows could be required to enable high utilization of a flow's path, but only in two unlikely cases: an unusually large diameter network with long delay in an empty network, or traffic with persistent high delay on the path taken by the flow's ACKs (unlikely because of the use of congestion control).

each flow is set manually to the optimal value (1/10 of the link bandwidth for remote and local flows and to 1/2 for the victim flow). The graph shows that all flows can achieve their ideal throughputs when their injection rates are set appropriately. In the following section, we describe our approach for designing source response functions that can automatically converge to appropriate rate limits.

V. DESIGNING SOURCE RESPONSE FUNCTIONS

The source response function defines how the flow rate is adjusted in response to network congestion feedback. Since congestion feedback is delivered through ACK packets, we assume that the flow's injection rate is adjusted at each time an ACK packet is received. Upon receipt of an unmarked ACK, the source response must increase the flow's rate based on an *increase function*, $r_{new} = f_{inc}(r)$. Similarly, upon receipt of a marked ACK, the source response must reduce the rate limit based on a *decrease function*, $r_{new} = f_{dec}(r)$. Naturally, these functions must maintain the flow rate limits between some minimum setting, R_{min}, and some maximum setting, R_{max}[6]. f_{inc} and f_{dec} should be designed to operate together and have the following desired properties:

- Congestion Avoidance
- High Network Bandwidth Utilization
- Fair Allocation of Bandwidth among Flows

A. Design conditions for Source Response Functions

We now identify source response conditions that we use to design $f_{inc}(r)$ and $f_{dec}(r)$, in order to achieve the above desired properties:

- **Condition 1: Avoiding Congested State**
  In steady state, flow rates will oscillate around an optimal value. Congestion notification will be sent to

[6]In this paper we assume packets have the same size and rate is represented in packets per unit of time. The extension of our results for packets with different sizes and rate represented in bytes per unit of time is straightforward [13].

sources when the aggregate rate of flows sharing a bottleneck link exceeds the link bandwidth, causing the flows to reduce their rate. On the other hand, while the aggregate rate is kept below the link capacity, the absence of congestion notifications will cause the flows to increase their rates with time, until they exceed the link bandwidth again. This will cause flows to operate in cycles of rate decrease steps followed by rate increase steps. In these cycles, it is desirable for the increase steps to recover the rate by less than the magnitude of the decrease so that flows are less likely to experience the same (or higher) degree of congestion after recovery. This is also reasonable because the lack of a mark is not a clear signal to increase the rate whereas a packet is marked if and only if there is at least some degree of congestion spreading. The absence of a mark can mean either that there is spare bandwidth and thus an increase is desirable, or it can mean that the current injection rate is ideal. This leads to our first condition.

Condition 1: *The magnitude of the response to a marked ACK should be larger than or equal to magnitude of the response to an unmarked ACK*

$$f_{inc}(f_{dec}(r)) \leq r$$

- **Condition 2: Fairness Convergence**
  The response function should be able to converge to a fair operating point, starting from any initial distribution of rates among competing flows.
  Chiu and Jain [14] have identified sufficient conditions that ensure linear source response functions converge to fairness assuming all flows receive feedback and adjust their rates synchronously. The conditions require: an increase function that improves fairness combined with a decrease function that either improves or maintains the fairness, or a decrease function that improves fairness combined with an increase function that either improves or maintains the fairness. Chiu and Jain [14] show that the traditional AIMD response function satisfies their convergence requirement.
  A fairness convergence requirement that assumes synchronous feedback and synchronous rate adjustment, as proposed in [14], can be overly conservative. For most networks, and particularly for SANs that have switches with small buffers, packet marking is not synchronous. These networks have the property that only a subset of flows is affected by a marking event, and packets of higher rate flows are more likely to be marked than those of lower rate flows. Source response functions that are designed to increase fairness

in a synchronous environment do not take into account the packet marking bias.

We exploit the packet marking bias and weaken the fairness convergence requirement in two ways. First, it is sufficient to require that each cycle of decrease and increase *not degrade* the level of fairness. Specifically, unlike Chiu and Jain[14], each cycle need not strictly improve fairness; they can maintain the same level of fairness. Due to the marking bias against higher rate flows, these flows will receive a higher rate of congestion notification and experience more decrease steps reducing their rate over time. Second, we relax the requirement that the increase and decrease functions must individually maintain or improve fairness. We allow one of these functions to decrease fairness as long as the other function ensures that a cycle of increase and decrease at least maintains the fairness. The observation that source response functions can converge to fairness even if either (but not both) of its decrease or increase functions decreases fairness has also been made in [22]. This weaker fairness convergence requirement allows the use of response functions that can reclaim available network bandwidth faster than response functions that satisfy the stronger requirement. In our congestion control mechanism, congestion feedback is asynchronous because congestion information is piggybacked on ACKs. Furthermore, the rate of congestion notification is not the same for all flows since higher rate flows receive ACKs more often than lower rate flows. Therefore, we formulate the convergence requirement using a description of flow rate adjustments over time (rather than flow rate adjustments at synchronous events). We define recovery time $T_{rec}(r)$ for a flow at rate r as the time elapsed from the time the flow rate is decreased from r, due to a marked ACK, until the time the flow rate recovers to its original rate r, assuming no other marked ACK is received until rate r is achieved. Consider the case in which flows at different rates each receive a marked ACK due to the same congestion event. If we guarantee the recovery time $T_{rec}(r)$ for lower rate flows does not exceed that of higher rate flows, fairness is not degraded over the decrease and recovery cycle. This meets our weaker convergence requirement. This also allows decrease/increase function designs that individually degrade fairness. Hence, we formulate our condition for fairness convergence as follows:

Condition 2: *For any two competing flows with different rate limits, the recovery time for the lower rate flow should be less than or equal to the recovery time for the higher rate flow*

$$T_{rec}(r_1) \leq T_{rec}(r_2) \quad \text{for } r_1 < r_2$$

. **Condition 3: Maximizing Bandwidth Utilization**

In order to maximize bandwidth utilization the source response function must be able to reclaim available link bandwidth as fast as possible. Minimizing the time to reclaim available bandwidth corresponds to using the limiting case for conditions *1* and *2*.

First, assuming the recovery time after a rate decrease to the minimum rate R_{min} is fixed, the recovery time at higher rates is minimized when condition *2* is set at the limiting case, $T_{rec}(r_1) = T_{rec}(r_2)$. Second, we choose the minimum value of T_{rec} that satisfies condition *1*, for the minimum rate. This corresponds to recovering a flow at the minimum rate R_{min} to the original rate R'_{min} (which is $f_{dec}^{-1}(R_{min})$) after receiving only one unmarked ACK. Since ACK packets are received at the same rate as data packets are transmitted the minimum recovery time is given by the time between two consecutive packet transmissions, i.e. $T_{rec} = \frac{1}{R_{min}}$. Condition *3* is then summarized as follows:

Condition 3: *The recovery time after a rate decrease from an arbitrary rate r is constant and equal to the interval between the transmission of two consecutive packets at the lowest rate R_{min}.*

$$T_{rec}(r) = \frac{1}{R_{min}} \quad \text{for } f_{dec}^{-1}(R_{min}) \leq r \leq R_{max}$$

Note that condition *3* is equivalent to the limiting case of condition *1*, only for the minimum rate. For higher rates ($r > R'_{min}$), the response to a marked ACK is strictly larger than the response to an unmarked ACK, i.e. $f_{inc}(f_{dec}(r)) < r$. Higher rate flows receive ACK packets more frequently and therefore should receive a larger number of ACKs during the constant recovery time T_{rec}.

B. Methodology for Designing Response Functions

In this section we describe the methodology we use to design fair and efficient source response functions. We assume a decrease function $f_{dec}(r)$ is defined, and then derive an increase function $f_{inc}(r)$ using the conditions proposed in the last section.

In the absence of marks, we would like the rate to gradually increase over time. Suppose $F_{inc}^r(t)$, for $t \geq 0$, are

a family of continuous monotonic increasing functions, each of which describes the desired flow rate increase behavior as a function of time since the last rate decrease to an arbitrary rate $F_{inc}^r(0) = r$ ($R_{min} \leq r \leq R_{max}$). Since we define the increase function $f_{inc}(r)$ as a function of the current rate, the time behavior of the rate increase should be independent of the past history of the flow rate, i.e. it should be independent of the elapsed time since the last decrease. Therefore, the time behavior of the rate for two arbitrary initial rates r_1 and r_2, ($R_{min} \leq r_1 < r_2 \leq R_{max}$), should be identical for rates $r > r_2$, i.e.:

$$F_{inc}^{r_2}(t) = F_{inc}^{r_1}(t + t') \quad \text{for } t \geq 0, \text{ and } t' \text{ such that}$$
$$F_{inc}^{r_1}(t') = r_2 \quad (1)$$

It follows that the rate increase behavior can be represented by just one member of the family of functions: $F_{inc}(t) = F_{inc}^{R_{min}}(t)$. All other functions F_{inc}^r, for $R_{min} < r \leq R_{max}$, can be obtained by shifting the time origin of $F_{inc}(t)$ as described in equation 1.

From our condition *3*, the recovery time T_{rec} is constant for any rate r. Thus, after a flow decreases its rate to $r = f_{dec}(r')$, due to a marked ACK, it would take a constant time T_{rec} for recovering to its original rate r', i.e. $F_{inc}^r(T_{rec}) = r'$. Equivalently, $F_{inc}(t + T_{rec}) = r'$, for t such that $F_{inc}(t) = r$. Therefore,

$$f_{dec}(F_{inc}(t + T_{rec})) = f_{dec}(r') = r = F_{inc}(t) \quad \text{or}$$
$$F_{inc}(t) = f_{dec}(F_{inc}(t + T_{rec})) \quad (2)$$

In practice we cannot adjust the flow rate continuously with time, but only at discrete times. As previously stated, we chose to adjust the rate at the reception of each unmarked ACK. After an adjustment to rate r, the next ACK is nominally received in a time interval $1/r$. Thus we define[7] $f_{inc}(r) = min(F_{inc}^r(1/r), R_{max})$.

In summary, to obtain an increase function $f_{inc}(r)$ we need to find a function $F_{inc}(t)$ that satisfies Equation 2. A discussion of how this can be accomplished for general response functions is out of the scope of this paper. In the next Sections, we show how we obtained $f_{inc}(r)$ for two specific response functions.

C. Function 1:

Fast Increase Multiplicative Decrease (FIMD)

For the FIMD source response function we adopt a multiplicative rate decrease function, which is the same de-

---

[7]Our derivations and definitions assume all packets are of the same size and that each ACK acknowledges a single packet of this size. Our analysis can be easily extended to handle variable size packets by defining T in terms of the maximum size of a packet and increasing the rate on an ACK in proportion to the size of the packet that is being acknowledged by the ACK.

crease function used by the traditional AIMD function.

$$f_{dec}^{fimd}(r) = max\left(\frac{r}{m}, R_{min}\right)$$

where $m > 1$ is constant

From Equation 2, $F_{inc}(t)$ must satisfy:

$$F_{inc}(t + T_{rec}) = F_{inc}(t) * m$$

With $F_{inc}(0) = R_{min}$, this is satisfied by the continuous function:

$$F_{inc}(t) = R_{min} * m^{t/T_{rec}}$$

For any rate r, there exists a t' for which $r = F_{inc}(t') = R_{min} * m^{t'/T_{rec}}$. Therefore,

$$F_{inc}^{r}(t) = F_{inc}(t + t') = R_{min} * m^{t'/T_{rec}} * m^{t/T_{rec}}$$
$$= r * m^{t/T_{rec}}$$

and

$$f_{inc}^{fimd}(r) = min(F_{inc}^{r}(1/r), R_{max})$$
$$= min(r * m^{1/rT_{rec}}, R_{max})$$
$$= min(r * m^{R_{min}/r}, R_{max})$$

D. Function 2:

Linear Inter-Packet Delay (LIPD)

The LIPD response function is designed to leverage the Inter-Packet Delay (IPD) design feature in InfiniBand [1]. IPD is the idle period length that is inserted between the injection of consecutive packets of a flow, expressed in units of packets transmission time. A flow operating at an IPD of ipd corresponds to a flow rate of $\frac{R_{max}}{1+ipd}$. We define a flow's rate decrease as an increment by one of the flow's IPD value (which increases the inter-packet delay by one packet transmission time). This rate decrease function is intuitively attractive for the following reason. If n identical flows share a bottleneck link, the optimal rate for each flow is $\frac{R_{max}}{n}$ (IPD of $n - 1$). If n flows are at the optimal rate and a new flow arrives, then upon receiving one mark each of these flows reduces its rate to $\frac{R_{max}}{(n+1)}$ (IPD of n), which is the new optimal rate assignment. Also, at lower rates this function decreases the rate by smaller steps than a multiplicative decrease function (FIMD and AIMD). In typical scenarios where several dynamic flows are sharing a link, the use of smaller decrease steps results in lower amplitude of oscillation and larger overall utilization of the link. This rate decrease function can be derived using the inverse relationship of flow rate to the flow IPD:

$$f_{dec}^{lipd}(r) = max\left(\frac{R_{max}}{\frac{R_{max}}{r} + 1}, R_{min}\right)$$

From Equation 2, $F_{inc}(t)$ must satisfy:

$$F_{inc}(t + T_{rec}) = \frac{R_{max}}{\frac{R_{max}}{F_{inc}(t)} - 1}$$

With $F_{inc}(0) = R_{min}$, this is satisfied by the continuous function:

$$F_{inc}(t) = \frac{R_{max}}{\frac{R_{max}}{R_{min}} - \frac{t}{T_{rec}}}$$

For any rate r, there exists a t' for which $r = F_{inc}(t') = \frac{R_{max}}{\frac{R_{max}}{R_{min}} - \frac{t}{T_{rec}}}$. Therefore,

$$F_{inc}^{r}(t) = F_{inc}(t + t') = \frac{R_{max}}{\frac{R_{max}}{R_{min}} - \frac{t'}{T_{rec}} - \frac{t}{T_{rec}}}$$
$$= \frac{R_{max}}{R_{max}/F_{inc}(t') - t/T_{rec}}$$
$$= \frac{R_{max}}{R_{max}/r - t/T_{rec}}$$

and

$$f_{inc}^{lipd}(r) = min(F_{inc}^{r}(1/r), R_{max})$$
$$= min\left(\frac{R_{max}}{R_{max}/r - 1/rT_{rec}}, R_{max}\right)$$
$$= min\left(\frac{R_{max}}{R_{max}/r - R_{min}/r}, R_{max}\right)$$
$$= min\left(\frac{r}{1 - R_{min}/R_{max}}, R_{max}\right)$$

Although the analytical description of the response functions presented above looks complex, these functions can be easily implemented in hardware. This can be done by choosing a finite set of discrete rates and using lookup tables to implement discrete versions of the functions $f_{inc}(r)$ and $f_{dec}(r)$. The equations presented here, are just needed at design time to compute the values stored in these lookup tables.

VI. EXPERIMENTAL RESULTS

We have evaluated the performance of our packet marking and source response mechanisms through simulation and present the results in this section. The simulations use the example topology of Fig. 1. Source response functions combine the appropriate rate control function with a window limit of one packet.

A. Marking Policy Comparison

In our first set of simulation experiments, shown in Fig. 4, we compare our proposed packet marking mechanism with the naive scheme that only mark packets in
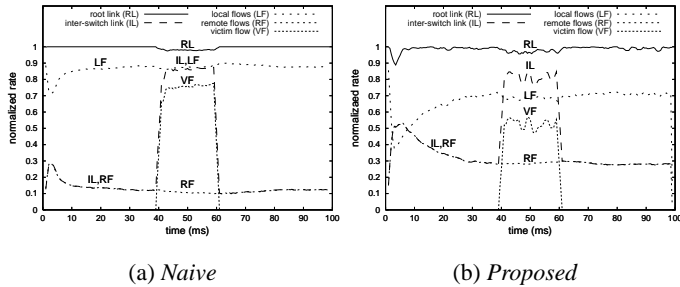
(a) *Naive*　　　　　(b) *Proposed*

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

Fig. 5.　$F_{inc}(t)$ with $R_{min} = R_{max}/256$
for AIMD, FIMD, and LIPD.

a full buffer. In this set of experiments we assumed the scenario shown in Fig. 1 with 10 remote and 10 local flows (R=10, L=10) and used the LIPD source response function. Fig. 4(a) 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. 4(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.

Fig. 4(b) shows the simulation results obtained for our proposed packet marking mechanism. 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 proposed mechanism marks all packets that are generating congestion on a root link, 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 our proposed marking mechanism, the number of packets of remote flows that are marked is approximately equal to the number of input
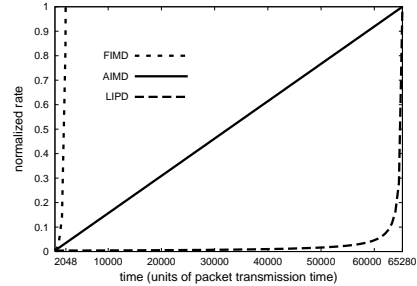
buffer slots[8]. In contrast, for the local flows the marking scheme samples a distribution of buffer usage over the whole range from zero usage to the peak usage. A fair state, in which local and remote flows have the same rate limits, is not stable because in that state each marking event tends to 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.

We have elsewhere proposed and evaluated additional packet marking mechanisms that can further improve fairness if properly tuned [24].

B. Evaluation of Source Response Functions

In this section we compare the performance of LIPD, FIMD, and traditional AIMD. As with FIMD, the AIMD source response uses a multiplicative rate decrease function. For the increase function, the rate limit is increased linearly with time[9]. The maximum *slope* of the linear rate increase is limited by the minimum recovery time T_{rec} at the lowest rate, which corresponds to a recovery with just a single unmarked ACK. In the results presented here we use this maximum *slope* for the AIMD rate increase.

In all our evaluations we set the minimum rate limit R_{min} to $\frac{R_{max}}{256}$, based on the limitation imposed by the InfiniBand IPD mechanism as explained later in this Section. For FIMD and AIMD we use a decrease factor $m = 2$. To compare the increase behavior of the three response functions, Fig. 5 plots $F_{inc}(t)$ normalized by R_{max}, which shows how the flow rate increases over time starting at rate $R_{min} = R_{max}/256$. Fig. 5 shows that AIMD recovers from minimum rate to maximum rate in

---

[8]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.

[9]This is analogous to TCP's window-based AIMD, which increases the window size by one maximum segment size each round trip time [8].
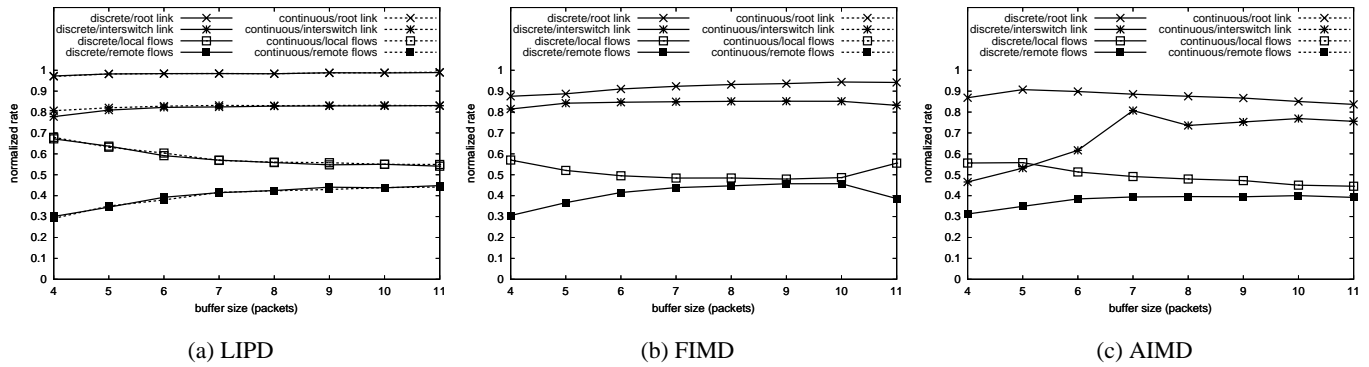
Fig. 6. Performance of source response functions with static traffic pattern.
(Dashed lines are mostly invisible because they are hidden by the solid lines)

the same total time as LIPD and much slower than FIMD, even though AIMD and FIMD reduce their rates identically with the same number of marks. LIPD recovers quickly at high rates and slowly at low rates, which matches its rapid decrease in response to marks at high rate and gradual decrease at low rates. For example, for 1 GByte/sec links and 2048 byte packets, the total time to reach the maximum rate starting from the minimum rate is 4.2 ms for FIMD and 133.7 ms for LIPD and AIMD.

In the following subsections, we compare the dynamic behavior of our response functions with AIMD. For that, we run simulations assuming the scenario shown in Fig. 1 with 10 remote and 10 local flows (R=10, L=10). Each simulation is run for *500 ms* simulated time, and the reported results average the rates over the last *400 ms*. With all three response functions, flows are initialized to the maximum rate limit R_{max}. We expect traffic flows in the SAN environment to be bursty, short-lived and sensitive to latency. For such environments, initializing flows to the maximal rate can allow the flows to attain maximum bandwidth quicker and incur lower latency than an approach based on slow-start. Two traffic environments are investigated: a static traffic pattern that has long-lived static flows, and a dynamic traffic pattern that has flows that come and go.

1) **Static Traffic Pattern:** Simulation results in Figure 6 show the impact of response function on link utilization when all flows are static (long-lived). The graphs show link utilization as a function of the size of the switch buffer in number of packets per input port. For each response function, the results are plotted for a design that employs a discrete set of rates and for a design with continuous rate values. For the discrete case we use 256 discrete rates, as supported by the InfiniBand [1] IPD mechanism. We choose rates corresponding to integer values of *ipd* in the range $[0, 1, \ldots, 255]$, yielding rates $\frac{R_{max}}{1+ipd}$. The discrete and continuous curves in Figure 6 are nearly identical, suggesting that an IPD mechanism that supports

a discrete set of rates, as in InfiniBand, can be leveraged and used for congestion control without sacrificing performance.

Overall, the results show that LIPD performs the best for this static flow scenario, resulting in almost 100% utilization of the root link and high utilization of the inter-switch link. In comparison to FIMD and AIMD (with $m = 2$), LIPD responds to a packet mark with a smaller reduction of the rate limit. Thus at equilibrium the oscillation of the flow rate has lower amplitude with LIPD than with FIMD and AIMD. For all the schemes, fairness between local and remote flows improves with larger buffers, as explained in Section IV-A.

In contrast to LIPD and FIMD, with AIMD the inter-switch link has low utilization in scenarios with small input buffers. Although victim packets rarely receive marks (usually they cut through switch B and avoid an extended stay in the input buffer), victim packets receive more marks with smaller buffer sizes. The slow rate increase function of AIMD causes the victim to recover slowly from the sporadic marks resulting in poor utilization of the inter-switch link. In contrast, LIPD and FIMD exhibit fast recovery that tolerates occasional victim packet marking.

2) **Dynamic Traffic Pattern:** In a real network, traffic flows arrive and depart dynamically. To gain understanding of the performance impact of the source response function with dynamic traffic, we performed experiments in which flows come and go dynamically. In our experiments, an ON-OFF process determines the arrival and departure of dynamic flows for a (source, destination) pair. A new flow arrives at the source at the start of an ON period and departs at the start of the OFF period. The ON and OFF times are exponentially distributed with equal mean duration. Simulation times were set to values large enough to have an average of at least 20 ON cycles per dynamic flow for the experiments with long ON times, and to at least *500 ms* for experiments with short ON times.

(a) 50% static flows
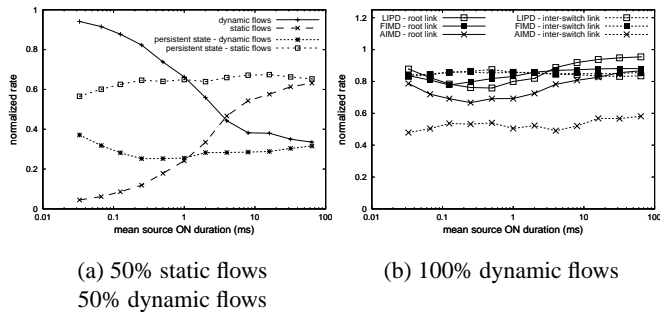50% dynamic flows

(b) 100% dynamic flows

Fig. 7. Dynamic traffic patterns.

Fig. 7(a) shows results for a mixed environment in which half the local and remote flows in the scenario of Fig. 1 are dynamic, and half are static ($L = 5 + 5$, $R = 5 + 5$). The Figure shows the aggregate flow rates of dynamic and static flows, plotted as a function of the mean ON duration. The curves labeled "dynamic flows" and "static flows" illustrate that with frequent arrivals and departures (small mean ON duration), dynamic flows hog the bandwidth, starving the static flows. When a new flow arrives it is initialized to the maximum rate limit, and its contention with the static flows causes both to be marked. Since the dynamic flows are short-lived, the marks have little impact on them. The static flows, however, suffer continually from the frequent arrival of new flows and the consequent marking. As ON duration increases, the dynamic flows arrive less frequently, approaching a static scenario in which the static flows receive twice as much bandwidth than the dynamic flows, since just half of the dynamic flows are active on average at any time.

Since initializing each new flow to have the maximum rate limit results in poor performance for long-lived flows, we propose the use of a scheme in which rate limit persists across consecutive flows that have the same (source, destination) pair, similar to the approach proposed in [25]. Results for this approach are also presented in Figure 7(a), corresponding to the curves labeled "persistent state". We observe that when using this approach static flows are not penalized and receive a fair share of the bandwidth. For the shortest ON durations, the persistent congestion control state makes short-lived flows that arrive frequently behave similarly to a single static flow. In this case the dynamic flows receive the same amount of bandwidth as static flows, since they all behave as static flows, explaining why the persistent curves approach a normalized rate of 0.5 at low ON durations.

Figure 7(b) shows the results of an experiment in which all the flows (except the victim) are dynamic, with persistent congestion control state. The graph shows how the choice of response function affects the utilization of the root link and the inter-switch link. The inter-switch link

has high utilization, except in the case of AIMD (as explained in Section VI-B.1), which confirms that congestion spreading does not affect the victim, when using our proposed response functions.

For the root link, when the ON duration has the lowest and highest values, the source response functions have similar behavior as with static traffic patterns; utilization is maximized by LIPD, then FIMD, then AIMD. For large ON durations, the traffic pattern is nearly static and for short ON durations dynamic flows behave as static flows as mentioned before. The intermediate range of ON durations (from approximately 0.2 ms[10] to 2 ms), corresponds to a dynamic traffic behavior and thus benefits from using FIMD which can adapt faster to changes in traffic demand. The results show that FIMD can achieve higher root link utilization in this range.

AIMD has the worst performance on all ranges achieving approximately 10% lower utilization than the best response function, which is FIMD for more dynamic scenarios and LIPD for more static scenarios.

VII. CONCLUSIONS

In this paper, a new congestion control scheme for InfiniBand networks was developed and evaluated. The scheme eliminates congestion spreading, a consequence of InfiniBand link level flow control in which congestion that originates at one oversubscribed link may drastically reduce the throughput of seemingly unrelated traffic throughout the network. Key properties of InfiniBand such as no packet drops, small bandwidth-delay product, small packet buffers, etc. guided the development of a scheme that has two components: a simple ECN packet marking mechanism applicable to modern input-buffered switches, and a source response mechanism that combines rate control with a window limit, adequate for an InfiniBand environment.

The proposed ECN mechanism is triggered by a full input buffer and differs from conventional approaches by marking all packets that contribute to congestion even if their buffers are lightly utilized. The performance results show improved fairness of this approach over conventional packet marking.

We derived a set of conditions to be satisfied by source response functions in order to achieve convergence to fair and efficient operating points. While fairness convergence requirements have been proposed in previous work [14] for a scenario with synchronous network congestion feedback, we derive convergence requirements for a more realistic asynchronous environment. Our conditions are based

---

[10]Each flow can transmit only a few packets in a ON period of 0.2 ms, 5 to 10 packets assuming there are 10 to 20 active flows

on a more relaxed constraint for fairness convergence than proposed in [14]. The use of more relaxed conditions enables the use of source response functions that can reclaim unused link bandwidth faster and can achieve higher bandwidth utilization than could be achieved by response functions based on the stricter convergence requirement proposed in [14]. We proposed two novel source response functions based on our weaker convergence requirement. We showed through simulation results that these functions outperform the traditional AIMD response function which satisfies the stricter convergence requirent proposed in [14].

This paper focused on the rate control aspects of congestion control, while maintaining a fixed window size of one packet. We envision, however, that a hybrid window and rate control approach may be beneficial for SANs in which ACKs experience queueing delays in the reverse path. We plan to investigate how rate control and window control can be combined into a single mechanism that appropriately adjusts both the window size and the rate limit. In addition, we want to explore our end-to-end congestion control mechanisms with richer traffic patterns and larger and more general network topologies.

REFERENCES

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

[2] Robert W. Horst, "TNet: a reliable system area network," *IEEE Micro*, vol. 15, no. 1, pp. 37–45, February 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 Wen-King Su, "Myrinet: a gigabit-per-second local area network," *IEEE Micro*, vol. 15, no. 1, pp. 29–36, February 1995.

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

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

[7] H. T. Kung, Trevor Blackwell, and Alan Chapman, "Credit-based flow control for ATM networks: Credit update protocol, adaptive credit allocation and statistical multiplexing," *SIGCOMM '94*, vol. 24, no. 4, pp. 101–114, Aug. 1994.

[8] Van Jacobson, "Congestion avoidance and control," in *SIGCOMM*. Lawrence Berkeley Laboratory, August 1988, pp. 314–329, ACM.

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

[10] Nick McKeown, Martin Izzard, Adisak Mekkittikul, William Ellersick, and Mark Horowitz, "Tiny tera: A packet switch core," *IEEE Micro*, vol. 17, no. 1, pp. 26–33, Jan. 1997.

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

[12] Sally Floyd and Van Jacobson, "Random Early Detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[13] Yoshio Turner, Jose Renato Santos, and G. (John) Janakiraman, "An approach for congestion control in InfiniBand," Tech. Rep. HPL-2001-277, HP Laboratories, October 2001.

[14] 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.

[15] Ludmila Cherkasova, Al Davis, Robin Hodgson, Vadim Kotov, Ian Robinson, and Tomas Rokicki, "Components of congestion control," in *ACM Symposium on Parallel Algorithms and Architectures*, 1996, pp. 208–210.

[16] Lawrence S. Brakmo and Larry 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, October 1995.

[17] Christina 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)*. October-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," Tech. Rep. RFC 3168, IETF, September 2001.

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

[21] The ATM Forum Technical Committee, *Traffic Management Specification Version 4.1*, Number AF-TM-0121.000. www.atmforum.com, March 1999.

[22] D. Bansal and H. Balakrishnan, "Binomial congestion control algorithms," in *IEEE INFOCOM*, April 2001, vol. 2, pp. 631–640.

[23] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM*, August 2000.

[24] Jose Renato Santos, Yoshio Turner, and G. (John) Janakiraman, "Evaluation of congestion detection mechanisms for InfiniBand switches," in *IEEE GLOBECOM – High-Speed Networks Symposium (to appear)*, November 2002.

[25] Hari Balakrishnan, Hariharan S. Rahul, and Srinivasan Seshan, "An integrated congestion management architecture for Internet hosts," in *ACM SIGCOMM*, September 1999.