# End-to-End Congestion Control for System Area Networks

Jose Renato Santos, Yoshio Turner, G. (John) Janakiraman
Internet Systems and Storage Laboratory
HP Laboratories Palo Alto
HPL-2002-4 (R.1)
May 13th , 2002*

congestion control, system area networks, SAN, InfiniBand, window control, rate control, ECN, I/O Interconnect

System Area Networks (SANs) using link-level flow control experience congestion spreading, where one bottleneck link causes traffic to block throughout the network. In this paper, an end-to-end congestion control scheme is proposed 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 hybrid window and rate control at end-devices. 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.

* Internal Accession Date Only
Approved for External Publication

# End-to-End Congestion Control for System Area Networks

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

## 1 Introduction

System area networks (SANs) interconnect processors and I/O devices [15, 14, 1, 3]. They can connect hundreds to thousands of devices over distances from a few meters to hundreds of meters. To meet application I/O requirements, SANs can support end-to-end latencies (as seen by the application) under 10 microseconds and bandwidths of several Gb/s. The use of SANs is likely to become more widespread with the emergence of industry standards such as InfiniBand [15].

Congestion is an important issue in the design of SANs, as in any computer network [16]. Typical SAN switches do not allow packet dropping, in order to avoid long latencies associated with packet retransmission and the high cost of packet reordering logic. Switches use link level flow control [8, 17], 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 [16], it may cause an undesired effect known as congestion spreading or tree saturation [9]. 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.

Congestion control has been widely studied in the literature, especially in the context of LANs and WANs [16]. However, this problem needs to be re-evaluated in the context of SAN environments, due to their unique characteristics. First, switches cannot drop packets in response to congestion. Thus congestion spreading is possible. Also, packet losses are not available as indicators of congestion. Second, switches have small forwarding delays ($\sim$ 100ns) due to cut-through switching, and the short links contribute negligible propagation delay. Hence, the bandwidth-delay product is relatively small ($\sim$ 1KB) and a flow can consume all the link bandwidth on its path with few packets (even less than one) in transit at any time. Third, SAN switches are typically single-chip devices [1, 19] with small

packet buffers (typical InfiniBand switches may hold 4 packets of 2KB per port). Therefore, congestion can occur even when the number of flows contending for a single link is small. With small buffers, queueing delays during congestion can be similar to queueing delays in normal operation, making it difficult to use latency to detect incipient network congestion. Fourth, since SAN switches operate at very high speeds, switches are usually configured with buffers at the input ports [18][1]. Input-buffered switches require different techniques to identify packets causing congestion than traditional techniques [22, 12] which are aimed at output-buffered switches. Finally, due to the low latency and high bandwidth characteristics of SANs, congestion control mechanisms must be practical to implement in hardware.

In this paper we address the problem of congestion control in a SAN environment. While our approach is applicable to SANs in general, our investigation was conducted in the context of the InfiniBand architecture [15]. However, details describing how our proposal is combined with specific features of InfiniBand such as virtual lanes, ACK coalescing, remote DMA, unreliable transport, etc., are beyond the scope of this paper. The interested reader is referred to [23] for InfiniBand-specific details of our proposal.

The main contributions of this paper can be summarized as follows:

- We propose a congestion control scheme for a SAN environment that avoids congestion spreading while achieving high network utilization. Our congestion control scheme is based on a judicious combination of an explicit congestion notification (ECN) mechanism at network switches with an efficient source response mechanism at the end devices. We propose a simple ECN based congestion detection mechanism where switches identify the packets that are causing congestion and mark them using a traditional ECN bit in the packet header to notify the end devices of congestion. For input-buffered switches, our ECN mechanism supports better fairness properties than the traditional approach of marking packets in a full buffer. The mechanism can be easily implemented in low cost, high speed hardware. Our source response mechanism is based on a combination of a window-based [16] and a rate-based approach [13]. The window component limits the amount of buffer that a flow can consume in the network and prevents packets from being injected when acknowledgments stop arriving. Compared to window control, rate control allows flows to have a lower buffer utilization at switches, which is appropriate when the switch buffer sizes are small, as is the case in SAN environments. By using a hybrid scheme, we combine the advantages of both approaches.

- We specify requirements for source response functions that allow flows to converge to fair and efficient operating points. Our requirements use more relaxed constraints than the ones proposed by Chiu and Jain [6] which (the latter) are satisfied by the traditional Additive Increase Multiplicative Decrease (AIMD) response function used in TCP [16]. The less constrained properties allow the use of response functions that can react faster

[1]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, making it challenging to design for very high link speeds.

to changes in traffic demand and reach an efficient operating point faster than response functions derived using Chiu and Jain properties, leading to higher network bandwidth utilization, especially in dynamic environments in which flows come and go.

- We propose two specific response functions based on our properties and evaluate their performance through simulation both in a static demand scenario and in a dynamic one. We also compare them with the traditional AIMD response function and show their superior performance both for static and dynamic environments.

The rest of this paper is organized as follows. Related approaches to congestion control are briefly summarized in Section 2. Section 3 motivates the need for congestion control in a SAN environment, by showing the harmful effect of congestion spreading in a simple scenario. The key characteristics of our proposed approach are described in Section 4. Section 5 presents the congestion detection mechanism used by switches to mark packets causing congestion. Section 6 presents our source response function design principles and the specific response functions proposed. Finally, Section 7 presents our conclusion.

# 2 Related Work

For networks that use link-level flow control, *hop-by-hop* congestion control has been proposed [5, 9]. Those schemes limit the number of packets at a switch that share a common output link or final destination. To enforce the limits, switches must implement a substantially enhanced link flow control mechanism. We take a different approach that aims to keep switch design simple by assigning most of the responsibility for traffic control to the flow end-devices (network adapters at the flow source and destination). Possible advantages of our approach include easier incremental deployment of improved policies and coordination with higher-level QoS objectives.

For traditional networks, *end-to-end* control has been widely studied and is exemplified best by TCP [16]. For the predominant versions of TCP, flow sources use endpoint detection of packet dropping as an implied signal of congestion. Other related work have proposed TCP variants which use changes in network latencies to detect congestion before it is severe enough to cause packet loss [4, 20].

An alternative approach to such implicit detection of congestion 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 [22, 12]. Enabling switches to mark packets slightly increases switch implementation complexity. ECN is used in ATM networks [13], and it has been proposed for use with TCP [21, 10]. 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.

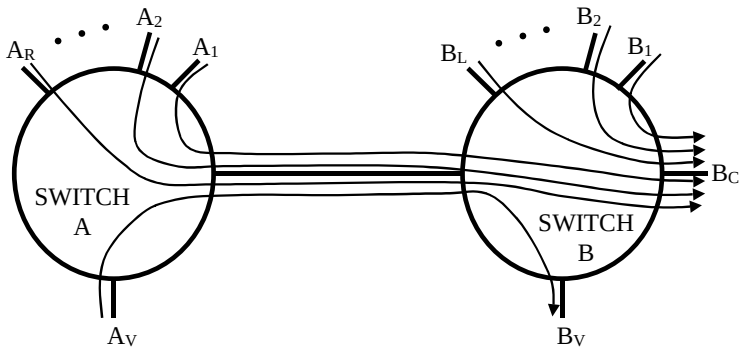



Figure 1: Simulation scenario

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 [6]. AIMD has been used for both window control [16] and rate control [7]. Recently, other response functions aimed largely at multimedia streaming applications have been investigated that attempt to be compatible with TCP without suffering the large fluctuations in injection rate that can arise from the multiplicative decrease of AIMD [2, 11]. We propose more aggressive source response functions, based on more relaxed fairness convergence requirements, that allow flows to rapidly reclaim available bandwidth, increasing the effective network throughput in a environment in which flows come and go dynamically.

# 3 Congestion Spreading

In this section, as a motivation for our congestion control scheme, we show the harmful effect of congestion spreading on the performance of the network. 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 Figure 1 and which we use for all results presented in this paper. Table 1 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 that originate at $L$ sources attached to switch $B$, a set of $R$ *remote* flows that originate at $R$ sources attached to switch A, and a *victim* flow that originates at source $A_V$ also attached to switch A. All remote and local flows are destined to end device $B_C$ through a congested output link on switch B. The victim flow is destined to a non congested end device $B_V$ and suffers from congestion spreading. Congestion spreading originates at the oversubscribed link connecting switch $B$ to end device $B_C$ which we refer to as the root link of the congestion spreading tree or simply *root*. The number of remote and local flows varies over the experiments presented in this paper. We always assume that the

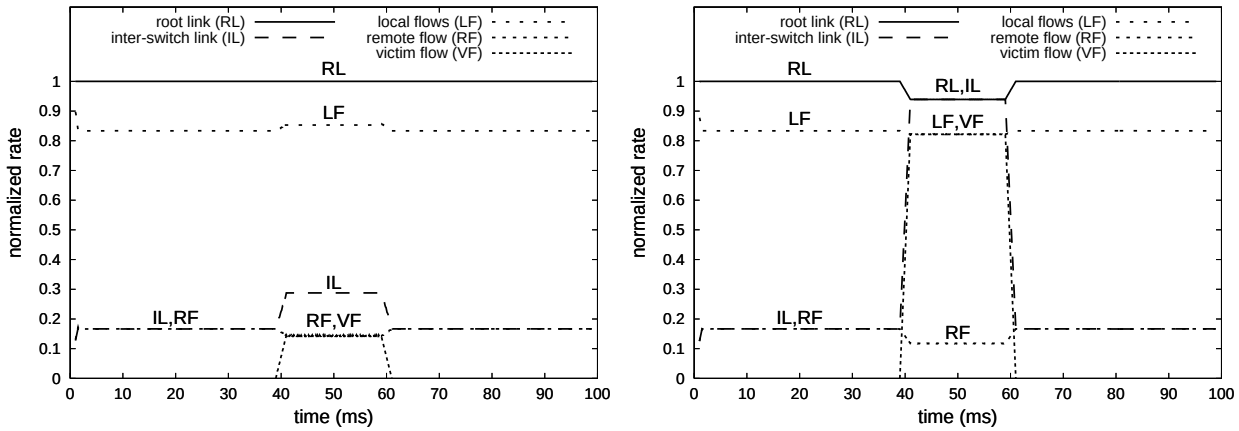
| parameter | default value (unless otherwise specified) |
|---|---|
| link bandwidth | 1 GB/sec (Infiniband 4X links) |
| packet header | 20 bytes (Infiniband Local Header) |
| data packet size | 20 + 2048 = 2068 bytes |
| data packet tx time | 2.068 $\mu s$ |
| ACK packet | 20 bytes |
| switch minimum 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) |

Table 1: Simulation Parameters

flow rates are only limited by the network, i.e. flows try to use all the network bandwidth that they can. We adopted a switch scheduling that implements a modified FIFO discipline for each output port. In order to improve throughput, packets at the head of the FIFO for a particular output port can be bypassed by newer packets when their input buffer is busy transmitting a packet to another output port. The number of times the packet at the head of the FIFO may be bypassed is limited to prevent starvation. In our simulations, this limit is set to four. Although we adopted this particular switch scheduling in our simulation, the congestion scheme we propose in this paper is orthogonal to the particular switch scheduling adopted.

To illustrate the problem caused by congestion spreading, we consider the scenario shown in Figure 1 with 5 local flows and 1 remote flow ($L = 5$, $R = 1$). Figure 2(a) 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 begining 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 rate of local flows (aggregate rate), remote flow, and victim flow. The 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 about 15% of the bandwidth on the inter-switch link, even though the inter-switch link is only about 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. Since there are 6 flows contending for the root link, each flow consumes approximately $\frac{1}{6}$ of the link bandwidth. Switch $A$ alternates transmitting packets for each of the two flows, remote and victim, whenever the inter-switch link is not blocked by the link level flow control mechanism. Thus both the remote and victim flows are transmitted at the same rate, i.e. $\frac{1}{6}$ of the link bandwidth, leaving $\frac{2}{3}$ of the inter-switch link unused. If the remote flow did




(a) No Congestion Control

(b) Window=1

Figure 2: Congestion Spreading Example ($L = 5$, $R = 1$): buffer capacity = 4 packets

not attempt to transmit at the full link bandwidth and 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.

The goal of our congestion control scheme is to contain congestion spreading such that network bandwidth can be used more efficiently. The challenge is to design a scheme that enables victim flows to increase their throughput without compromising the throughput at a root link.

# 4 Overview of the Proposed Congestion Control Approach

Our approach for congestion control in SAN environments combines two key elements. The first is an ECN mechanism where network switches selectively detect and mark packets belonging to flows that are *generating* congestion spreading trees. The second is a source response mechanism that is a hybrid of traditional window-based and rate-based response mechanisms.

## 4.1 Switch Congestion Detection and Notification Mechanism

When switches are not permitted to discard packets during congestion, end-devices cannot identify network congestion by observing packet losses. End-devices can attempt to infer congestion by observing variations in round-trip latencies. This technique is unlikely to be

accurate since latency variations due to congestion are not easily distinguished from latency variations due to random traffic bursts, especially when switch buffers are small. We adopt an approach where switches explicitly detect and signal congestion, since such an approach can be more accurate.

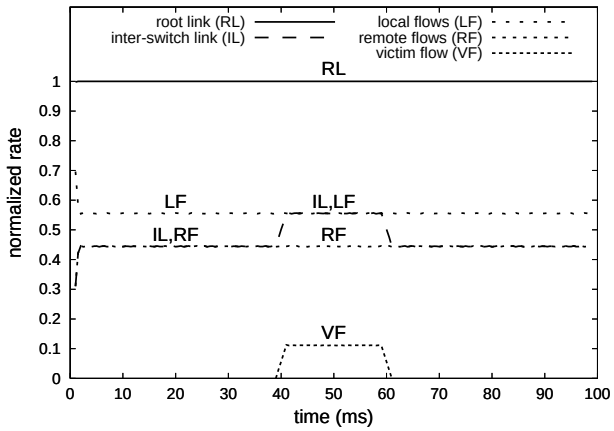
When traffic demand exceeds a link capacity, the link becomes the root of a congestion spreading tree. We call packets that are transmitted on a root link, *generating* packets. When a switch identifies a packet as a generating packet, it marks a single bit ECN field in the packet's header to indicate 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.

## 4.2 Hybrid Window-Rate Source Response Mechanism

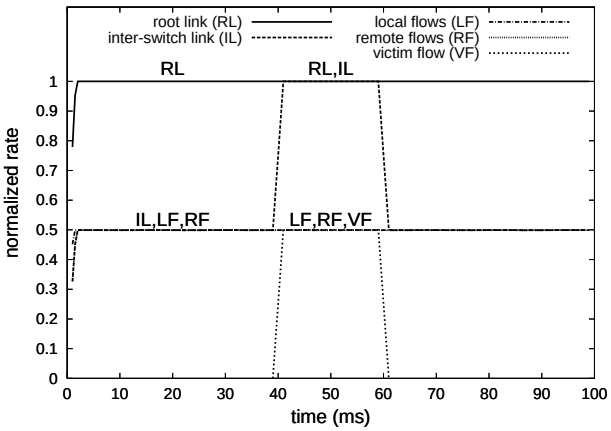
As described earlier, a key characteristic of a SAN environment is its small bandwidth-delay product, which is the result of low cut-through switching delays and the use of short links. Consequently, it is reasonable to consider a response mechanism that limits each flow to a fixed window size of just one packet. A window-based mechanism offers the benefit that packet injection is self-clocked [16] and it limits the amount of buffer space that a flow can consume in the network (in this case one). Figure 2(b) shows the result of a simulation of the same scenario used in Figure 2(a), except that each flow is limited to a window size of one packet. Figure 2(b) shows that congestion spreading can be eliminated by the window limit. Since only two flows share the inter-switch link, switch B's input buffer for the inter-switch link never becomes full 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). However, congestion spreading can occur even when flows are limited to a congestion window of one packet when the number of contending flows exceeds the number of buffer slots. That condition can easily occur in SAN networks where switch buffer sizes are small. Figure 3(a) 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 Figure 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. However, a rate control mechanism can satisfy this requirement. Figure 3(b) shows simulation results that illustrate the potential for rate control to improve performance over the results in Figure 3(a). In the experiment, the rate limit for each flow is set manually to the optimal value. The graph shows that all flows can achieve their ideal throughputs when their injection rates are set appropriately.

Our congestion control approach aims to integrate window and rate control in a single hybrid




(a) Congestion Spreading:Window=1




(b) Fixed Optimal Rates with Window = 1. Rates are set to 1/10 of the link bandwidth for remote and local flows and to 1/2 for the victim flow

Figure 3: Congestion Spreading ($L = 5$, $R = 5$): fixed window vs. rate control (buffer capacity = 4 packets)

mechanism that can provide the benefits of both approaches. A hybrid mechanism would control both the flow window size and the flow rate. Although a small fixed window of one packet is satisfactory in most cases, an adjustable window size may be required to achieve high throughput if the network diameter is large or if flows experience significant queueing delays in the reverse path used to return ACKs and congestion notifications to flow sources. However, in this paper we only discuss scenarios in which a window of size 1 is sufficient, i.e. the network diameter is small and the reverse path is not congested. Therefore, we focus our discussion on the rate control mode of operation of our response mechanism and use a fixed window of size 1 in all results presented in this paper. A discussion of the operation of our response mechanism when the window needs to be set to larger values is left as future work.

# 5 Congestion Detection and Notification: Packet Marking

Switches detect congestion when a buffer becomes full[2], 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

[2]With the 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.

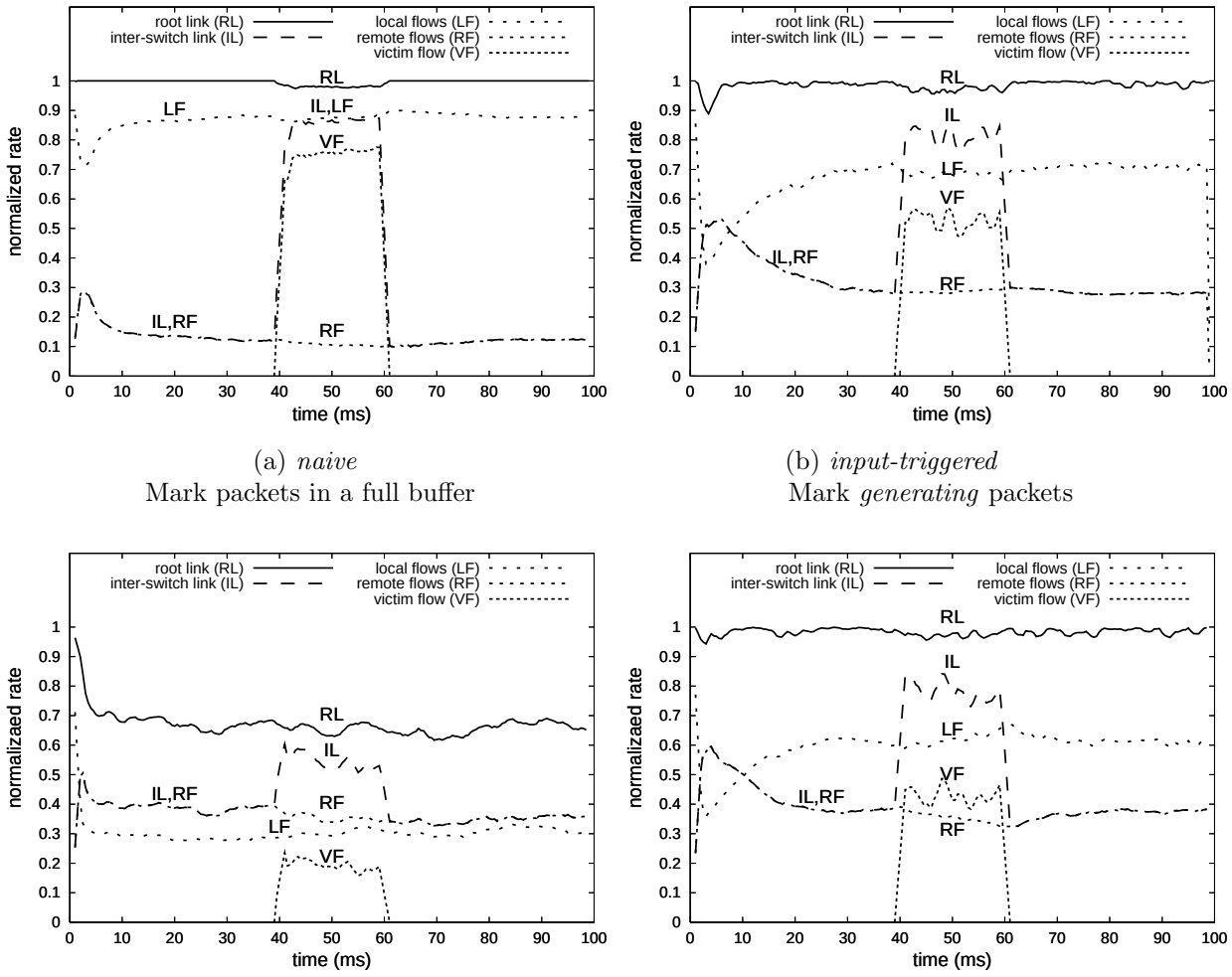
may be *generating* packets destined for a busy root link or they may be *victim* packets that are waiting for blocked links that are upstream from a root link. With congestion control, our simulation results, presented later, show that congestion spreading is reduced to very low levels, and the time 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[3].

In a switch with output buffer configuration, the packets in the full buffer are the only generating packets at the switch. However, in input buffered switches (typical for SANs) packets in the full buffer are not the *only* generating packets. Other input buffers at the switch which are not full may contain generating packets as well. Thus a way to identify all generating packets is required. In this section we describe and evaluate such marking policies. The simulation experiments discussed in this section assume the scenario presented in Figure 1, with $L = 10$ and $R = 10$, and use the LIPD response function that we propose in section 6.

Figure 4(a) presents simulation results for a naive packet marking mechanism that simply mark the packets in a full buffer (we treat a buffer as "full" when the receipt of the last byte of a packet leaves it with insufficient space to receive one additional maximum sized packet, since a buffer in this state can block the upstream switch if it needs to transmit a maximum size packet). The results show that this packet marking mechanism can avoid congestion spreading and allow the victim flow to receive high throughput. However, this mechanism 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 the Figure), the local flows utilize 90% of the available root link bandwidth. The unfairness observed in the results 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 packets are marked since each local flow uses a different input buffer and the window limit prevents it from filling the buffer. This penalizes the remote flows, which have their rate reduced while the local flows take a disproportionate share of the congested link bandwidth. In general, this scheme penalizes flows that arrive at a switch competing for an oversubscribed link through an input port shared with many competing flows.

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, packet marking is triggered when one of the switch input buffers becomes full. Second, any output link that is the destination for at least one packet in the full buffer is classified as a congested link. Third, all packets that are resident (in any buffer) at the switch and and are destined to an output link that was classified as congested

[3]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.




(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

Figure 4: Comparison of Packet Marking Policies: $L = 10$, $R = 10$, buffer capacity = 4 packets.
(LIPD source response – defined in Section 6)

in the second step are classified as generating packets and marked[4]. 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 when only one becomes full. We specify an efficient implementation that does not require this scan. This 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

[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.

determine the number of packets that should be marked, we use two counters for each output link. The first counter *cnt1* contains 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* contains the number of next packets that need to be marked when transmitted on that output link. Most of the time $cnt2 = 0$. When a buffer becomes full, counter $cnt2$ is updated with the value of counter $cnt1$. Then, the output port starts marking the next transmitted packets, decrementing $cnt2$ at each transmission, until it reaches zero again. 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 in fact turns out to be an advantage, since our implementation will mark the first packets to leave the switch and provide faster feedback to end devices.

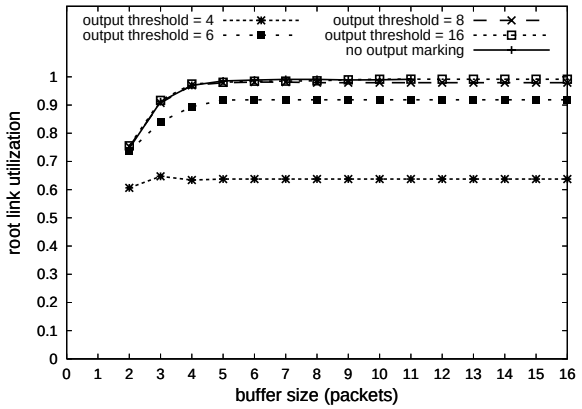
Figure 4(b) shows the simulation results obtained for the input-triggered marking policy. The results show that this marking policy also avoids congestion spreading and keeps the inter-switch link at high utilization. Moreover, fairness between the remote and local flows is improved when compared to 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[5]. 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.

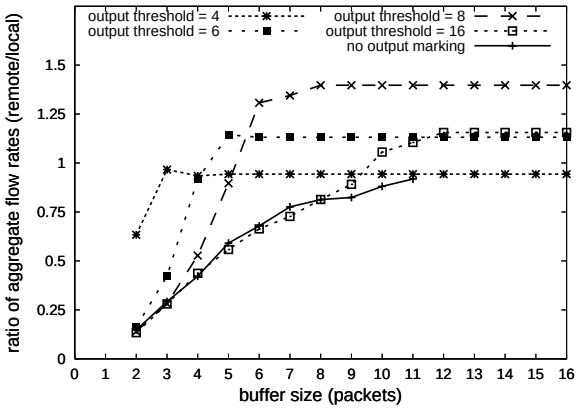
For improved fairness among the flows contending for the congested link, the input-triggered marking mechanism can be combined with an additional *output-triggered* mechanism that is triggered when the total number of packets that are waiting for an output link exceeds a threshold. We refer to this combined marking mechanism as *input-output-triggered*.

Output-triggered marking events tends 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.

[5]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

Figure 5: 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 Figure 1 ($L = 10$, $R = 10$). LIPD source response – defined in Section 6.

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. Figures 4(c) and 4(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 (Figure 4(c)), packets are too frequently marked before an input buffer fills, resulting in low utilization of the congested link. With a threshold of eight packets (Figure 4(d)), the root link has high utilization without congestion spreading, and the local and remote flows receive approximately equal rates.

Figure 5 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*. Figure 5(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. Figure 5(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 dominates, which preferentially mark 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[6] 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[7]. Hence the fairness ratio in Figure 5(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, when buffer size is large, 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 with $L = R$ (10 to 40 total flows) and obtained results similar to the ones shown in Figure 5.

# 6 Source Response

The source response mechanism controls the injection of packets into the network based on ECN information delivered to the source via ACKs. While we propose a source response mechanism that controls both the flow rate and the flow window, the discussion in this paper is focused on the rate control. Upon receipt of an unmarked ACK, the source response must increase the flow's rate limit based on an *increase function*, $r_{new} = f_{inc}(r_{curr})$. Similarly, upon receipt of a marked ACK, the source response must reduce the rate limit based on a *decrease function*, $r_{new} = f_{dec}(r_{curr})$. The rate increase and decrease functions should be designed to operate together to enable high throughput while preventing the starvation of any flow.

[6]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

[7]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.

## 6.1 Source Response Function Design Principles

AIMD (*Additive Increase Multiplicative Decrease*) [6] is a well-known source response function, which has been shown to converge to a fair operating point that utilizes all available bandwidth when all flows receive feedback and adjust their rates synchronously. Under the synchronous assumption, an additive increase function increases fairness, while a multiplicative decrease function does not change fairness [6]. Thus every rate decrease/increase cycle increases fairness. Source response functions that assume synchronous feedback can be overly conservative (i.e slow). 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 allowing the use of response functions that do not improve fairness in a synchronous scenario. These response functions can increase rate more aggressively and reclaim available network bandwidth faster than response functions that satisfy the stronger requirement.

Our source response functions are based on three design principles. The description of our design principles assumes that a source can set a flow's rate limit to any value between some minimum setting, $R_{min}$, and some maximum setting, $R_{max}$[8].

1. **Convergence Principle:** *If two competing flows with different rate limits simultaneously decrease their rates, the time for the flow with a lower rate limit to recover, subsequently, to its prior setting is less than or equal to the time for the second, higher rate flow to recover, subsequently, to its prior setting.*
   This principle is our weaker requirement for convergence to fairness, which only requires that source response functions *not decrease* fairness when operating in the synchronous environment (we intend to explore weaker requirements that even allow response functions to decrease fairness in the synchronous environment, in the future).
2. **Congestion Avoidance Principle:** *The response to a marked ACK packet must be at least as significant as the response to an unmarked ACK packet.*
   This principle is based on the desire to operate most of the time in a non congested state. It is motivated by the fact that the lack of a mark is not a clear signal to increase the rate: it 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. In contrast, a packet is marked if and only if there is at least some degree of congestion spreading. This principle is enforced by ensuring that for any two rates $r1$ and $r2$, where $r1 > r2$, the number of marked ACK packets it takes to reduce a flow's rate from $r1$ to $r2$ is less

[8] 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.

than or equal to the number of unmarked ACK packets it takes to increase the flow's rate from $r2$ to $r1$.

3. **Responsiveness Principle:** *Any flow that is decreased to the minimum rate limit setting $R_{min}$ in one mark is to recover to its prior higher rate after only a single unmarked ACK packet has been received.*
   This principle minimizes the time required to reclaim all available link bandwidth, thereby avoiding link underutilization. Recovering a flow from the lowest possible rate on the receipt of only a single unmarked ACK packet allows the minimum incremental recovery time that satisfies principle 2 above. By principle 1, this also minimizes the lower bound on the recovery time of the same flow once it reaches a higher rate.

Many source response functions can satisfy our design principles. We construct two new source response functions using these design principles in Section 6.2. Our evaluation of these new source response functions shows that the time to reclaim bandwidth improves dramatically with our increase functions compared to a classic AIMD approach. As a consequence, the new increase functions improve link utilization, particularly in dynamic scenarios in which flows come and go.

## 6.2 Source Response Functions: FIMD, LIPD and AIMD

When defining response functions we made two design choices consistent with our design principles. First, to ensure quick reaction to congestion, a flow's rate is decreased on every single mark, unless it is already at the minimum rate $R_{min}$. Second, in order to derive aggressive increase functions we use the weakest convergence constraint that satisfies our principle 1: a flow's rate, which is decreased as a result of a mark, returns to the previous higher setting in constant time $T$. From principle 3, $T = \frac{1}{R_{min}}$.

Assuming a decrease function $f_{dec}(r)$ is defined, we derive an increase function $f_{inc}(r)$ that satisfies our principles. In the absence of marks, we would like the rate to gradually increase over time. Suppose $F^r_{inc}(t)$, 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 $F^r_{inc}(0) = r$ ($R_{min} \leq r \leq R_{max}$). 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 $R_0$ and $R_1$, ($R_{min} \leq R_0 < R_1 \leq R_{max}$), should be identical for rates $r > R_1$, i.e.:

$$F^{R_1}_{inc}(t) = F^{R_0}_{inc}(t + t') \qquad \text{for } t \geq 0, \text{ and } t' \text{ such that } \quad F^{R_0}_{inc}(t') = R_1 \tag{1}$$

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

Furthermore, from our two design choices described above, this function $F_{inc}(t)$ should satisfy the following property:

$$F_{inc}(t) = f_{dec}(F_{inc}(t+T)) \qquad 0 \ \leq \ t \ \leq \ T_{rec} - T, \quad \text{where} \quad T_{rec} = F_{inc}^{-1}(R_{max}) \qquad (2)$$

This property ensures that the flow recovers from any mark in constant time $T$.

In practice we cannot adjust the flow rate continuously with time, but only at discrete times. 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[9] $f_{inc}(r) = min(F_{inc}^{r}(1/r), R_{max})$.

We show how we obtained $f_{inc}(r)$ for two specific response functions. A discussion of how this can be accomplished for general response functions is out of the scope of this paper. Next, we describe the two proposed source response functions.

- *Fast Increase Multiplicative Decrease (FIMD).*
  The FIMD source response uses a multiplicative rate decrease function:

  $$f_{dec}^{fimd}(r) = max\left(\frac{r}{m}, R_{min}\right) \qquad \text{where } m > 1 \text{ is constant}$$

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

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

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

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

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

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

  and

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

- *Linear Inter-Packet Delay (LIPD).*
  The LIPD response function is designed to leverage the Inter-Packet Delay (IPD) design feature in InfiniBand [15]. 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

[9] 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.

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:

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

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

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

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

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

and

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

We also describe the traditional AIMD source response function [6]:

- *Additive Increase Multiplicative Decrease (AIMD).*
  As with FIMD, the AIMD source response uses a multiplicative rate decrease function:

$$f_{dec}^{aimd}(r) = min\left(\frac{r}{m}, R_{min}\right) \qquad \text{where } m > 1 \text{ is constant}$$

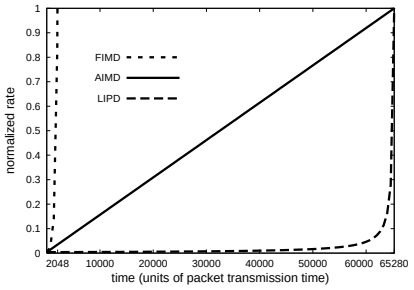



Figure 6: $F_{inc}(t)$ with $R_{min} = R_{max}/256$. Rate recovery over time from the minimum rate limit to the maximum rate limit for the three response functions: FIMD ($m = 2$), AIMD ($m = 2$), and LIPD. (Example: with 1 GByte/sec links and 2048 byte packets, recovery time from minimum to maximum rate limit is 4.2 ms for FIMD and 133.7 ms for LIPD and AIMD)

For unmarked ACKs, the rate limit is increased linearly with time[10]. The *slope* of the linear rate increase is limited by the recovery from the minimum rate $R_{min}$ to the value prior to a mark, $m \cdot R_{min}$. The minimal time interval to change the rate $R_{min}$ to $m \cdot R_{min}$ is given by the period between consecutive ACK receptions, $\frac{1}{R_{min}}$. Thus $slope = \frac{(m \cdot R_{min} - R_{min})}{1/R_{min}} = (m-1) * R_{min}^2$. At any rate $r$ the interval between ACKs is $1/r$, and the rate limit should increase by $(m-1) * R_{min}^2 \cdot (1/r)$. Hence,

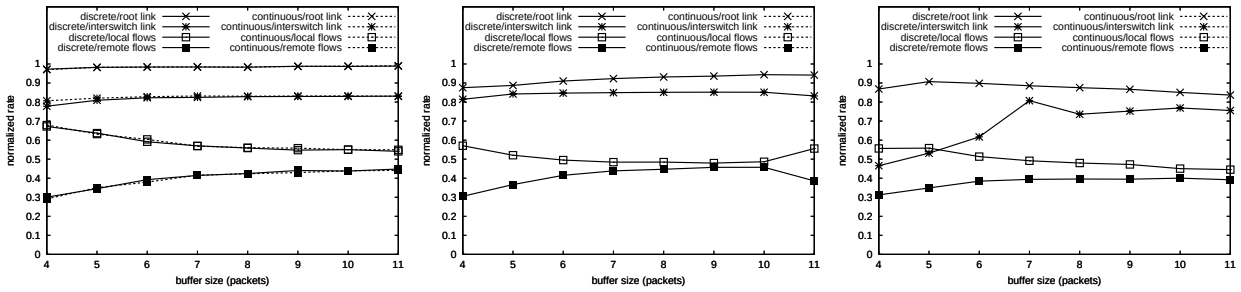
$$f_{inc}^{aimd}(r) = min\left(r + \frac{(m-1) * R_{min}^2}{r}, R_{max}\right) \qquad \text{(In our experiments, } m = 2\text{)}$$

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 in Section 6.3. To compare the increase behavior of the three response functions, Figure 6 plots $F_{inc}(t)$ normalized by $R_{max}$, which shows how the flow rate increases over time starting at rate $R_{min} = R_{max}/256$. Figure 6 shows that AIMD recovers from minimum rate to maximum rate in 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.

## 6.3 Evaluation of Source Response Functions

We next present simulation results that compare the performance of LIPD, FIMD, and traditional AIMD. For FIMD and AIMD we use a decrease factor $m = 2$. 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. The experimental

[10]This is analogous to TCP's window-based AIMD, which increases the window size by one maximum segment size each round trip time [16].




(a) LIPD (b) FIMD (c) AIMD

Figure 7: Performance of source response functions with static traffic pattern. Scenario in Figure 1 ($L = 10$, $R = 10$), input-triggered marking. (Dashed lines are mostly invisible because they are hidden by the solid lines)

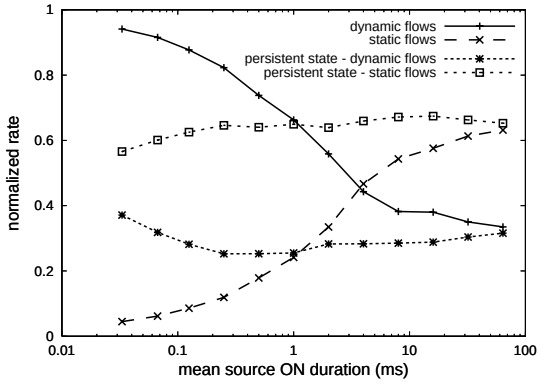
results use the scenario in Figure 1. Each simulation is run for *500 ms* simulated time, and the reported results average the rates over the last *400 ms*. Two traffic environments are investigated: the first uses long-lived static flows, and the second dynamic flows that come and go.

### 6.3.1 Static Traffic Pattern

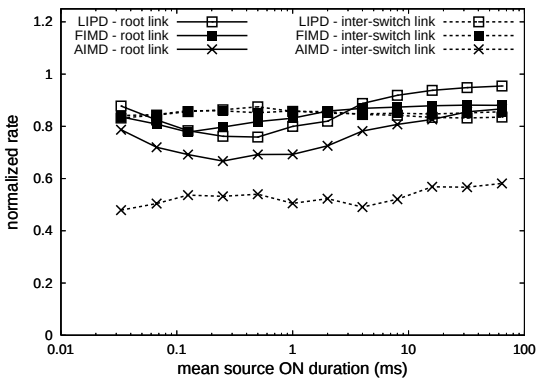
Simulation results in Figure 7 show the impact of response function on link utilization when all flows are static (long-lived). For each response function, the results are plotted for a design that employs a discrete set of rate limits and for a design with continuous rate limits. For the discrete case we use 256 discrete rates, as supported by the InfiniBand [15] 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 7 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 5.

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




(a) Mixture of static flows (5 local + 5 remote) and dynamic flows (5 local + 5 remote). With and without preserving congestion control state across dynamic flows having the same (source, destination) pair. LIPD source response (FIMD and AIMD have similar results).




(b) Utilization of root link and inter-switch link when all flows except the victim are dynamic. Congestion control state persists across flows.

Figure 8: Dynamic traffic patterns. Scenario in Figure 1 ($L = 10$, $R = 10$).
For dynamic flows, mean ON time = mean OFF time.
Discrete IPD rate limits.

occasional victim packet marking.

### 6.3.2 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.

Our first experiment examines a mixed environment in which half the local and remote flows in Figure 1 ($L = 10$, $R = 10$) are dynamic, and half are static. Figure 8(a) shows the aggregate flow rates of dynamic and static flows. The rates are 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. Results for this approach are also presented in Figure 8(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 than 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 8(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 6.3.1), 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 behaves as static flows as mentioned before. The intermediate range of ON durations (from approximately 0.2 ms[11] to 2 ms), corresponds to a dynamic traffic behavior and thus more adequate for FIMD which can adapt faster to changes in traffic demand. The results shows that FIMD can achieve higher root link utilization in this range.

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

# 7 Conclusions

In this paper, a new congestion control scheme for System Area Networks was developed and evaluated. The scheme eliminates congestion spreading, a consequence of SAN link level flow control in which congestion that originates at one oversubscribed link may drastically

[11]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

reduce the throughput of seemingly unrelated traffic throughout the network, and at the same time enables high network throughput. Key properties of SANs 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 source response functions that are a hybrid of window control and rate control, which can rapidly reclaim unused bandwidth.

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, although some unfairness remains. To further improve fairness we explore an approach which combines input-triggered marking with output-triggered marking. This combination is evaluated across a range of buffer sizes and output threshold values to understand their impact on fairness and utilization. For future work we plan to investigate approaches by which switches could tune the output triggering threshold to improve fairness without sacrificing utilization.

We proposed new principles for developing source response functions which specify more relaxed constraints for fairness convergence than the ones satisfied by traditional AIMD response functions. These relaxed constraints are realized by forgoing the classic assumption of synchronized congestion feedback and instead exploiting the higher probability of marking of packets from higher rate flows. The relaxed constraints allow the use of more aggressive increase functions that can rapidly reach efficient operating points. Two specific response functions, LIPD and FIMD, are derived and experimental results are presented that demonstrate they outperform conventional AIMD in both static and dynamic traffic scenarios.

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 ideal for SANs in which ACKs experience significant queueing delays or in large diameter networks with long empty network delays. We intend to build on the work presented here by investigating how rate control and window control can best be incorporated into a single cohesive 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 for richer traffic patterns and larger and more general network topologies.

# References


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

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

[3] Boden, N. J., Cohen, D., Felderman, R. E., Kulawik, A. E., Seitz, C., Seizovic, J. N., and Su, W.-K. Myrinet: a gigabit-per-second local area network. *IEEE Micro 15*, 1 (February 1995), 29–36.

[4] Brakmo, L. S., and Peterson, L. L. TCP Vegas: End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications 13*, 8 (October 1995), 1465–1480.

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

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

[7] Committee, T. A. F. T. *Traffic Management Specification Version 4.1.* No. AF-TM-0121.000. www.atmforum.com, March 1999.

[8] Dally, W. J. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems 3*, 2 (Mar. 1992), 194–205.

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

[10] Floyd, S. TCP and explicit congestion notification. *Computer Communication Review 24*, 5 (October 1994), 8–23.

[11] Floyd, S., Handley, M., Padhye, J., and Widmer, J. Equation-based congestion control for unicast applications. In *SIGCOMM* (August 2000).

[12] Floyd, S., and Jacobson, V. Random Early Detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking 1*, 4 (August 1993), 397–413.

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

[14] Horst, R. W. TNet: a reliable system area network. *IEEE Micro 15*, 1 (February 1995), 37–45.

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

[16] Jacobson, V. Congestion avoidance and control. In *SIGCOMM* (August 1988), Lawrence Berkeley Laboratory, ACM, pp. 314–329.

[17] Kung, H. T., Blackwell, T., and Chapman, A. Credit-based flow control for ATM networks: Credit update protocol, adaptive credit allocation and statistical multiplexing. *SIGCOMM '94 24*, 4 (Aug. 1994), 101–114.

[18] McKeown, N., Izzard, M., Mekkittikul, A., Ellersick, W., and Horowitz, M. Tiny tera: A packet switch core. *IEEE Micro 17*, 1 (Jan. 1997), 26–33.

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

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

[21] Ramakrishnan, K. K., Floyd, S., and Black, D. The addition of Explicit Congestion Notification (ECN) to IP. Tech. Rep. RFC 3168, IETF, September 2001.

[22] Ramakrishnan, K. K., and Jain, R. A binary feedback scheme for congestion avoidance in computer networks. *ACM Transactions on Computer Systems 8*, 2 (May 1990), 158–181.

[23] Turner, Y., Santos, J. R., and Janakiraman, G. J. An approach for congestion control in InfiniBand. Tech. Rep. HPL-2001-277, HP Laboratories, October 2001.