



## **An Approach For Congestion Control In Infiniband**

Yoshio Turner, Jose Renato Santos, G. (John) Janakiraman  
Internet Systems and Storage Laboratory  
HP Laboratories Palo Alto  
HPL-2001-277 (R.1)  
May 14<sup>th</sup>, 2002\*

congestion  
control,  
Infiniband,  
system area  
network,  
SAN, ECN,  
I/O  
interconnect

Infiniband switches cannot drop packets to deal with congestion. As a result, switch buffers can fill up, block upstream switches and even choke flows that are not contending for the congested link. The Infiniband standards body is exploring mechanisms to control congestion. This report describes the key elements of a congestion control approach that we have advocated. Our approach combines a low level mechanism that reacts quickly to short term demand variations with a centralized mechanism that deals with long term congestion and non-compliant devices. A simple switch-based mechanism detects flows that are causing congestion or are propagating the congestion. Switches mark the packet headers of these flows to inform the destination device of the congestion which in turn notifies the source device. In response, source devices adjust their packet injection based on a novel hybrid mechanism that combines the advantages of rate and window control. Adjustment policies are designed to enable high throughput while preventing starvation.

---

# AN APPROACH FOR CONGESTION CONTROL IN INFINIBAND

YOSHIO TURNER, JOSE RENATO SANTOS, G. (JOHN) JANAKIRAMAN

## ABSTRACT

Infiniband switches cannot drop packets to deal with congestion. As a result, switch buffers can fill up, block upstream switches and even choke flows that are not contending for the congested link. The Infiniband standards body is exploring mechanisms to control congestion. This report describes the key elements of a congestion control approach that we have advocated. Our approach combines a low level mechanism that reacts quickly to short term demand variations with a centralized mechanism that deals with long term congestion and non-compliant devices. A simple switch-based mechanism detects flows that are causing congestion or are propagating the congestion. Switches mark the packet headers of these flows to inform the destination device of the congestion which in turn notifies the source device. In response, source devices adjust their packet injection based on a novel hybrid mechanism that combines the advantages of rate and window control. Adjustment policies are designed to enable high throughput while preventing starvation.

## 1 INTRODUCTION

The Infiniband local link flow control mechanism prevents the network from dropping packets due to congestion. However, because of this mechanism a congested network can experience an undesired effect known as congestion spreading.

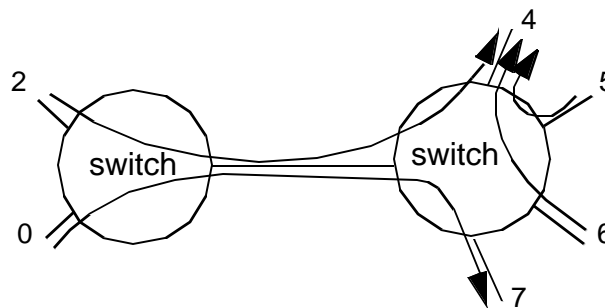
### 1.1 CONGESTION SPREADING

*Congestion spreading*, also known as *tree saturation*, occurs when a switch buffer fills up and blocks the buffers upstream. Congestion spreading originates at a link for which the traffic demand exceeds the link's capacity. Because of the link flow control mechanism, when buffers fill they simply block the arrival of subsequent packets — packets are not dropped. The blocking can spread further upstream until buffers fill all the way back to the source hosts of the affected traffic flows.

The primary bad result from congestion spreading is its impact on flows that do not exert load on the oversubscribed link resource. In [Figure 1](#), three traffic flows are directed to destination 4, and one flow goes from 0 to 7. The flow 0-7 shares the inter-switch link with the flow 2-4. Ideally, the sum of the throughputs of flows 0-7 and 2-4 should equal the capacity of the inter-switch link. However, if destination 4 is oversubscribed, then the switch buffers fill with packets and block flows on the inter-switch link. Therefore, the inter-switch link goes idle, wasting bandwidth that should be used for transmitting packets from 0 to 7. In this example, assuming a fair switch scheduling mechanism, each of the 3 flows directed to destination 4 will use approximately one third of the bandwidth of the bottleneck link. Assuming again a fair switch scheduling policy, the left switch will alternate packets

---

from flow 2-4 and 0-7 when scheduling packets on the inter-switch link. Therefore flow 0-7 will be transmitted at the same rate as flow 2-4, i.e. each will consume one third of the link bandwidth. Therefore one third of the link bandwidth which could be used to send extra packet from flow 0-7 is wasted. If the source node 2 were informed that it could not transmit at the full link bandwidth and reduced its rate to the rate determined by the bottleneck link, i.e. one third of the link bandwidth in this example, the buffers at the switch would not fill up and the bandwidth at the inter-switch link could be more efficiently utilized by the traffic generated by flow 0-7.



**Figure 1 Congestion spreading**

## 1.2 CONGESTION CONTROL FOR INFINIBAND

The current Infiniband specification (version 1.0) does not include a mechanism for controlling congestion. With an intent to support congestion control in a future release of the Infiniband standard, the standards body is exploring congestion control mechanisms appropriate for Infiniband. While congestion control mechanisms have been widely studied in the literature, the Infiniband environment presents several new characteristics which necessitate fresh exploration. The first of these is that, unlike TCP/IP routers, Infiniband switches cannot drop packets in response to congestion. Thus, packet losses are not available as indicators of network congestion. Second, Infiniband switches have relatively small buffers that can accommodate only tens of packets. Hence, Infiniband flows can have only a few packets in transit at any time. Third, round-trip latencies as well as switch and end-device processing latencies will be very low, in tens or hundreds of nanoseconds. Mechanisms for congestion control, therefore, must be capable of reacting very quickly and must be easily implementable in hardware. Fourth, Infiniband subnets will be much smaller in scale (number of devices) than IP networks and each subnet is likely to be controlled and managed by a single business entity. As a result, congestion control approaches using a central manager are reasonable to consider for Infiniband.

## 1.3 DESIRED CHARACTERISTICS OF CONGESTION CONTROL MECHANISMS

The following characteristics are desirable in an efficient congestion control mechanism:

---

- 
- Efficient utilization of network bandwidth

While the congestion control mechanism should limit the traffic injection rate to avoid congestion spreading, it should not be too conservative and reduce the traffic injection beyond what is necessary. If that happens, links will become idle even though traffic demand for them exists and the network bandwidth will be underutilized.

- Fairness

It is important that the bandwidth available at the network be fairly distributed among competing flows. The network congestion mechanism should allocate approximately the same bandwidth to equivalent flows. It is not acceptable to have a mechanism that achieves high network utilization at the cost of starving some flows.

- Fast reaction

The congestion control mechanism should react quickly to changes in the traffic demand. When traffic demand (e.g., number of competing flows) increases, the congestion control mechanism should react as quickly as possible to reduce the injection rates of current flows. If the response is too slow, congestion spreading will take place. When traffic demand decreases, the injection rate of remaining flows should be increased quickly, to avoid underutilization of the network bandwidth.

- Low oscillation

The congestion control mechanism should avoid large oscillations in the controlled flows' injection rates. If large oscillations occur, the state of network links may oscillate between being congested and being underutilized, reducing the overall network throughput. This property is in conflict with the previous one, fast reaction. The faster the mechanism reacts to traffic changes, the more likely it is to have large oscillations. The challenge for the congestion control mechanism is to strike a balance between these two conflicting properties and achieve a reasonable response with low oscillation on the injection rate of flows, maximizing the network bandwidth utilization.

## 1.4 SCOPE OF REPORT

We describe our proposed approach for controlling congestion in Infiniband in this report. Our approach uses a low level mechanism that permits fast reaction in combination with a central management mechanism that can control devices which are not compliant with the congestion control specification. This report focuses on the description of the low level mechanism. The key characteristics of the low level mechanism are described in [Section 2](#). The details of the individual mechanisms for congestion notification, congestion detection and congestion response are described in [Section 3](#), [Section 4](#) and [Section 5](#) respectively. We briefly discuss how a central manager mechanism is used to deal with non-compliant devices in [Section 6](#) but leave a full discussion of the central manager mechanism to a future report. This report assumes reasonable familiarity with the Infiniband architecture and some of its details, which are described in the Infiniband specification available at [\[1\]](#).

---

While we have evaluated several implementations of this congestion control approach and their performance characteristics, we defer discussion of these as well as comparison with related congestion control work to a future report.

## 2 OVERVIEW OF THE PROPOSED CONGESTION CONTROL MECHANISM

We propose a congestion control approach that combines a fast low level mechanism with a centrally managed mechanism.

In a low level mechanism the source CAs<sup>1</sup> are responsible for controlling their traffic injection rates based on congestion notifications received from the network. In this approach, a malicious or non-compliant source node may take most of the bandwidth of a congested link, adversely affecting flows that comply with the mechanism.

In a centrally managed approach, a central controller (e.g. the Subnet Manager) collects traffic statistics from the network and uses them to compute the optimal target data rates for the current active flows. The central controller is responsible for informing the source nodes of each flow the data rates at which they should operate. The current Infiniband specification (version 1.0) provides a static rate control mechanism (IPD<sup>2</sup> mechanism) which allows the Subnet Manager to program a source CA with the maximum data rate for a particular flow. Therefore a centrally managed congestion control mechanism can be deployed in an environment with a mix of spec-1.0-compliant and post-spec-1.0-compliant CAs. However, because the flow rates are computed at a central location, this approach has slow reaction time, when compared to a low level approach such as the one described in this document. Because of its slow response, a centrally managed congestion control approach is not adequate for an environment with a highly dynamic traffic pattern.

By using a low level mechanism combined with a centrally managed one, we are able to provide fast reaction to short term demand variations at the same time we provide long term rate control for non-compliant CAs.

The low level mechanism described in this document has the following characteristics:

- Congestion is controlled independently for each VL<sup>3</sup>  
Link flow control credits are controlled on a VL basis and thus congestion spreading trees are in fact generated by a congested VL, as opposed to a congested physical link. Also, congestion spreading trees are propagated to neighbor switches by VLs that run out of credits. Thus each congestion spreading tree is composed of specific VLs

---

1. Channel Adapters, which refer to the devices at the end-points of the Infiniband network
2. Inter Packet Delay, which refers to the minimum interval between successive packets
3. Virtual Lane, which refers to the (up to sixteen) logical links that can concurrently share a physical link

---

at the set of affected links (possibly only one or a few VLs for each affected link). Therefore congestion control should be done independently for each VL, since they may be experiencing different levels of congestion.

Congestion spreading originates at a VL for which traffic demand exceeds its effective bandwidth. The VL effective bandwidth is the link bandwidth that can be currently used by that VL. This effective bandwidth varies with time and is a function of the VL arbitration mechanism, as well as the traffic generated by other VLs sharing the same physical link. The root of a congestion spreading tree is a VL that is using all its effective bandwidth, has available credits and causes a previous input VL to run out of credit.

The proposed congestion control mechanism controls the rate of traffic injection for flows contributing to the congestion for each VL in a particular link independently of the other VLs.

- Targets both flows that are causing congestion and flows that are helping to propagate a congestion spreading tree

Flows that are using the root of a congestion tree are helping to generate the tree. Other flows which are not using the root, but which are contributing to fill switch buffers, are helping to propagate a congestion spreading tree to neighbor switches.

The proposed congestion control mechanism detects and throttles the rates of both flows that are helping to generate and flows that are helping to propagate congestion spreading trees.

- Explicit Congestion Notification (ECN)

The proposed mechanism is based on an explicit congestion notification (ECN) mechanism, in which the end nodes are explicitly notified of the congestion state of the network on the path traversed by data packets.

- Hybrid rate-window control

The injection of traffic into the network is controlled by an hybrid mechanism which controls both the flow rate and a window which limits the amount of data that a flow can inject into the network before receiving feedback from the destination.

The hybrid mechanism combines the advantages of rate and window control. A window mechanism is self-clocked and limits the amount of buffer that a flow can consume in the network. As soon as congestion delays data packets, ACKs stop arriving which prevents the flow from injecting new packets into the network. However, even if all flows contending for a congested link have a window of only one packet, congestion spreading can still occur if the number of contending flows exceeds the amount of buffer slots in the switch. This is quite feasible in an Infiniband network since, as discussed earlier, switch buffer sizes will be small. A rate control mechanism allows each flow to have an average buffer utilization less than one and therefore is more appropriate when the number of contending flows is large.

---

- Simple Mechanism

Our proposal is a simple mechanism which can be implemented at a low cost, both at switches and at CAs. It is also robust and easy to deploy because it does not require configuration of any special parameter such as thresholds.

In the next sections we describe in more detail the three components of the proposed congestion control mechanism:

- 1) Congestion Notification Mechanism

This is the mechanism used to notify the source CAs of the congestion state of the network detected at the switches.

- 2) Congestion Detection Mechanism

This component is implemented at each switch and is responsible for detecting the congestion state of the network.

- 3) Congestion Response Mechanism

This is the mechanism used by the source CAs to adjust their traffic injection rates in response to the congestion state notifications received from the network.

### 3 CONGESTION NOTIFICATION

The congestion notification mechanism is responsible for signaling the congestion state of the network detected at the switches to the CAs involved in the communication. We adopt an explicit congestion notification (ECN) mechanism, where switches mark packet headers with an one-bit ECN field, to inform the destination CA of the congestion state of the network on the path traversed by the packets.

Switches will only mark SEND, RDMA WRITE requests, and RDMA READ response packets, since these are the packets that have payload data and are expected to be the ones using most of the subnet bandwidth. We refer to these packets in general as **data packets**.

The ECN bit is initially set to “0” when a data packet is first injected in the network by the source CA. When a switch detects network congestion the ECN bit is set to “1”. Any switch on a data packet path can set the ECN bit according to the detection mechanism described later in [Section 4](#). No switch can reset a ECN bit set by a previous switch in the packet path. Thus, a data packet will have its ECN bit set, if and only if one or more switches in its path detects congestion.

Once the destination CA receives a data packet, it needs to inform the source CA of the network congestion state, since traffic injection has to be controlled at the source CA. The mechanism used by the destination CA to inform the source CA of the network congestion

---

state depends on the type of transport service and on the operation associated with the packet as described below:

1) SEND and RDMA Write packets for reliable service

Reliable transport services use acknowledgement packets (ACK/NAK) for confirming successful delivery of SEND and RDMA WRITE packets. For these type of packets the acknowledgement packet is used for propagating the network congestion state from the destination CA to the source CA. The destination CA updates the ECN bit of each acknowledgement packet it sends back to the source CA with the value of the ECN bit of the last received packet. Switches are not allowed to change the ECN bit of acknowledgement packets.

**ACK coalescing:** ACK coalescing is allowed but an acknowledgement packet must be generated for every request packet with the ECN bit set to "1". When the source CA receives an acknowledgement packet that acknowledges  $N$  outstanding packets it should react to the ECN bit of the acknowledgement packet as follows:

a) ECN = 0

The reaction of the source CA should be the same as if  $N$  independent acknowledgement packets with ECN = 0 were received.

b) ECN = 1

The reaction of the source CA should be the same as if  $N-1$  independent acknowledgement packets with ECN=0 followed by an acknowledgement packet with ECN=1 were received.

2) Unreliable service

Unreliable transport services do not use acknowledgement packets to confirm delivery of request packets. In order to propagate the network congestion state back to the source CA for unreliable service we use a new type of packet: the CN (congestion notification) packet. The destination CA generates a CN packet for each marked data packet, i.e. a packet with ECN=1. No CN packets are generated for data packets with ECN=0.

3) RDMA READ response

Since RDMA READ response packets are not acknowledged, the destination CA should use CN packets to propagate the network congestion state back to the source CA in this case too.

## 4 CONGESTION DETECTION

The goal of the congestion detection mechanism is to detect the onset of congestion and mark the data packets that are contributing to congestion spreading, such that the source

---

nodes originating these packets can be notified and throttle their traffic injection. There are two types of data packets that contribute to congestion spreading:

- 1) **Generating Packets:** data packets that are transmitted in the root of a congestion tree.

These packets are responsible for generating congestion.

- 2) **Propagating Packets:** data packets that are queued in a full input queue.

Even though these packets may not be causing congestion themselves, they are helping to propagate a congestion spreading tree to a neighbor switch. Note that packets can be both generating and propagating a congestion tree.

The congestion detection mechanism should mark both types of packets. Marking packets of type 1 is necessary since these are packets associated with traffic that is exceeding the network capacity. Even though marking packets of type 1 will reduce congestion on root links we still expect that some congestion spreading will occur, due to oscillations caused by traffic burstiness and latency in source reaction. Therefore, it is important to also mark packets of type 2 to contain the spreading of congestion when it starts, reducing the size of the congestion tree and thus reducing the number of flows impacted by congestion spreading.

We have designed a switch-based congestion detection mechanism that detects and mark these two types of data packets as described below:

#### a) **Marking packets propagating congestion**

The detection of congestion spreading at a particular switch is triggered when buffers associated with an input VL become full. This indicates that congestion spreading is occurring and all data packets in the full input buffer are marked, i.e. their ECN bit are set to "1". These are packets of type 2 described above.

We chose to use a full input buffer as opposed to a threshold on the buffer size to detect congestion spreading for two reasons:

- i) Input buffers are expected to be small

Due to high cost of high speed memory technology we expect that VL input buffers will be able to store only a few data packets. In order to absorb traffic fluctuations and minimize the probability of link underutilization we should allow the full buffer to be used during normal operation, since there are only a few buffers available per input VL.

- ii) Simplicity

Following our guidelines described before, we preferred a simpler scheme that does not require a threshold parameter to be tuned.

---

However, if technology evolves to a point where switches have large VL input buffers, a detection mechanism based on threshold on the size of the input queue may be more appropriate.

### **b) Marking packets generating congestion**

When an output VL becomes the root of a congestion tree, it will cause input VL buffers to grow and eventually become full. Thus, we also use a full input buffer condition to trigger the mechanism that detects and mark packets contributing to the generation of a congestion tree. This is done as follows:

#### **i) Identifying candidate roots of congestion trees**

Each output VL which is the destination for at least one data packet in the full input queue is considered a candidate root of congestion.

#### **ii) Identifying roots of congestion tree**

The state of each candidate VL identified as described above is examined to determine if it is the root of a congestion tree.

If the output VL is blocked, i.e. it does not have credits for sending data packets, then the VL is not considered the root of a congestion tree. The lack of credits indicates that the input buffer in the next switch is full and thus the root of this congestion spreading tree must be somewhere downstream. Data packets generating the congestion tree should be marked at the root link downstream and not in the current switch.

If the candidate output VL is not blocked then this VL is considered a root of a congestion tree and the subsequent data packets transmitted on this VL should be marked. If at any time in the future the VL runs out of credits and blocks, then the decision is revoked and the VL is not considered a root of a congestion tree anymore.

#### **iii) Marking packets generating a congestion tree**

The data packets to be marked are identified at the time a VL is determined to be the root of a congestion tree as described above. All data packets present in the switch and destined to this output VL at this particular instant (the instant when an input buffer became full) should be marked at the time of their transmission, assuming the VL does not run out of credits until then. If at any time the output VL runs out of credit, the subset of the initially identified data packets that are still in the switch are not marked.

The packet marking mechanism described in this section was designed to be simple and easy to implement. All design choices were made with the goal of having a simple mechanism and avoiding sophisticated traffic instrumentation. For example, we preferred to use a full buffer condition to trigger congestion detection, as opposed to defining arbitrary thresholds on queue sizes. We also preferred to use the instantaneous credit state of an output VL to determine if the VL is the root of a congestion tree, as opposed to computing

---

averages, time intervals, etc. The final result is a robust and simple algorithm which effectively detects congestion spreading and marks the right set of packets and at the same time minimizes complexity and cost of implementation.

## 5 CONGESTION RESPONSE

A source CA receives information about congestion from the fabric, and in response it adjusts packet injection for the corresponding “flow” (defined below). Two parameters for a flow may be increased or decreased by the source CA: rate limit, and window size. However, both parameters are supported only for reliable transport services. ACK packets that correspond to data packets are used to advance sliding congestion windows and to inform the source CA of the congestion status of the network. For unreliable transport services and for RDMA read responses, there are no ACK packets. Hence only the rate limit is adjustable for these traffic flows, and there is no window size limitation.

This section is organized as follows. Hybrid rate/window congestion control for flows that use reliable transport services is described in [Section 5.1](#). Rate control for flows that use unreliable transport services and RDMA read responses is described in [Section 5.2](#).

### 5.1 HYBRID RATE/WINDOW CONTROL: RELIABLE TRANSPORT

A source CA may transmit a data packet of a flow that uses a reliable transport service only if permitted both by the flow’s rate limit and by its window limit. For congestion control, the definition of “flow” depends on which reliable transport service is used:

- **Reliable connections:** a flow is identified by source QP<sup>1</sup>.
- **Reliable datagram:** a flow is identified by source EEcontext.<sup>2</sup>

A source CA sets and enforces per-flow rate and window limits on data packet injection. The rate limit is specified by an integer IPD (inter-packet delay) value, which yields a rate limit of  $\frac{R}{(IPD + 1)}$ , where R is the bandwidth of the source CA link. The initial IPD setting equals the static IPD value determined in path discovery. The IPD value may be adjusted as a result of receiving marked or unmarked ACK packets, but it cannot be set to a value lower than the static IPD.

In addition to the rate limit, a flow’s window limit specifies how many data packets of a flow may be injected without having received the corresponding ACK packets. The window size is initialized to its minimum value: one PMTU<sup>3</sup>-sized data packet. It subsequently may

---

1. Queue Pair, refers to a pair of send and receive queues

2. End-to-End Context, a construct like QP, is an end point of a reliable datagram channel (associated with another remote end point)

3. Path Maximum Transmission Unit, refers to the maximum payload size supported by the ports on the path

---

be adjusted in response to the congestion information in ACK packets, but it cannot fall below its minimum value of one PMTU. In the case of ACK coalescing, if an ACK is not received before a source CA exhausts its congestion window, the flow will be forced to block and lead to a time-out. To avoid this, a source CA has to request an explicit acknowledgement for at least one data packet in a set of data packets that exhaust the congestion window, by setting its AckReq bit in the Base Transport Header.

Reception of an ACK packet may indicate the need to adjust packet injection for a flow. For a decrease adjustment, both the window limit and the rate limit are decreased. In contrast, an increase adjustment depends on what factor is limiting packet injection for the flow. On each packet transmission, the factor that most hindered its injection is sampled and recorded. The result determines how a subsequent increase adjustment is handled. There are three cases:

- 1) Rate limited: In this case, the rate limit is increased.
- 2) Window limited: In this case, the window limit is increased.
- 3) Neither rate limited nor window limited: In this case, some other factor, such as slow generation of the packet by an application, was the limiting factor. Thus neither the window limit nor the rate limit is increased.

### **5.1.1 SOURCE RESPONSE FUNCTION: PROPERTIES**

We have developed source adjustment policies that are designed to enable high throughput and prevent the starvation of any flow. High throughput is enabled by aggressively stamping out congestion spreading and by quickly ramping up flows on lightly congested paths. Starvation is prevented by flow rate increase policies that do not excessively favor the higher rate flows that have the potential to force the lower rate ones into starvation.

To limit congestion from spreading widely, a source CA decreases a flow's rate/window in response to just a single marked ACK packet. Several alternatives for selecting the magnitude of decrease are possible, including multiplicative decrease of rate, and linear increase of IPD. Given a particular decrease function, a flow rate/window increase function must be derived that prevents starvation and allows quick recovery of rate/window on lightly congested paths. The derivation of rate/window increase functions is based on the following design principles:

- 1) *For any two rates (windows)  $r_1$  and  $r_2$  ( $w_1$  and  $w_2$ ), where  $r_1 > r_2$  ( $w_1 > w_2$ ), the number of marked ACK packets it takes to reduce a flow's rate (window) from  $r_1$  to  $r_2$  ( $w_1$  to  $w_2$ ) is less than or equal to the number of unmarked ACK packets it takes to increase the flow's rate (window) from  $r_2$  to  $r_1$  ( $w_2$  to  $w_1$ ).* The primary reason for this rule is to establish as a stable operating condition the absence of congestion spreading. A second reason is 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 de-

---

sirable, or it can mean that the current injection limit is ideal. In contrast, a packet is marked if and only if there is at least some degree of congestion spreading.

- 2) *If two competing flows with different rate/window limits simultaneously decrease their rates/windows, the time for the flow with a lower rate/window limit to recover to its prior setting is less than or equal to the time for the second, higher rate/window flow to recover to its prior setting.* We enforce this rule by requiring the increase function to return each flow to its rate/window setting prior to a single decrease adjustment in a time duration that is given by a monotonically non-decreasing function of the flow's current rate/window setting. The intuition behind this rule is based on a scenario in which multiple flows operate at different rates and share a bottleneck link. Each time the link becomes overutilized, the flows are marked and back off, reducing the link utilization below 100%. By ensuring that the first flows to recover to their prior settings are those operating at the lower rates, the next time the link reaches 100% utilization, the flows will have more similar rates than at the previous occasion of packet marking. Hence the flow rates tend not to diverge and not to lead to the starvation of any flow.
- 3) *Any flow that is decreased to the lowest possible flow rate (i.e. the rate specified by the maximum IPD setting) is to recover to its prior higher rate after only a single unmarked ACK packet has been received.* The intention is to ensure that a flow that remains after the departure of competing flows quickly reclaims the newly freed bandwidth. Recovering the flow from the lowest possible rate with only a single unmarked ACK packet reception results in the minimum incremental recovery time that satisfies rule 1 above. By rule 2 above, the incremental recovery time for the flow at the lowest possible rate is a lower bound on the recovery time of the same flow once it reaches a higher rate. Therefore, rule 3 minimizes the total time required for a flow to reclaim all the link bandwidth, thereby avoiding long periods of link underutilization.

A consequence of the rules above is that the increase functions derived from a large class of possible decrease functions do not always increase the flow rate/window limit each time an unmarked ACK packet is received. Instead, accumulation of multiple unmarked ACK packet receptions is often required before increasing the rate or window limit. This is especially likely to be the case for flows set to a high rate/window limit in order to satisfy the non-decreasing incremental recovery time requirement (rule 2).

For a multiplicative decrease function, the rules above are satisfied by an additive increase function, resulting in the well-known AIMD adjustment policy [2]. For general decrease functions, the rules above can be used to derive starvation-free increase functions with bounded recovery time. For example, an increase function can be derived for a decrease function that decreases the rate by increasing IPD linearly (say, by adding one each time).

---

### 5.1.2 PER-FLOW CONGESTION CONTROL STATE

Congestion control state is associated with each flow. For a flow that uses the reliable connection transport service, the state is associated with the source QP. For a flow that uses the reliable datagram transport service, the state is associated with the source EE-context. This section lists the state variables and their definitions. Their use is described in [Section 5.1.3](#).

Each flow is associated with the following window control state variables:

- **CWND:** congestion window size (number of data packets)
- **OUTS:** number of data packets that have been transmitted but for which no ACK packet has been received.

The congestion window restricts additional packet injection for a flow whenever OUTS equals CWND. OUTS is incremented each time a data packet of the flow is injected. OUTS is decremented each time an ACK packet is received that allows the source to advance its notion of the data packets that have been successfully delivered to the destination. OUTS is decremented by the number of data packets covered by the ACK packet.

Each flow is also associated with the following rate control state variable:

- **CIPD:** congestion control IPD rate limit

Finally, each flow is also associated with the following general state variables:

- **RPI:** the remaining number of data packets for which unmarked ACK packet reception is needed before increasing the flow's rate/window limit.
- **APSN:** adjustment packet sequence number. Identifies the PSN at the last rate/window decrease for the flow.
- **RWL:** 3-state rate/window/other flag which records what factor hindered the injection of the most recent packet injected for the flow: the rate limit, the window limit, or some other factor such as slow generation of packet by an application.

### 5.1.3 FLOW RATE/WINDOW INCREASE/DECREASE

The increase and decrease functions are common to all flows and specify when and by how much to adjust the rate/window limits. Each flow's CWND is initialized to one packet, and its CIPD rate limit is initialized to the static IPD (SIPD). RPI is initialized with a value that corresponds to a CWND of one packet (derived from a conceptual table described below).

---

### 5.1.3.1 RATE/WINDOW INCREASE

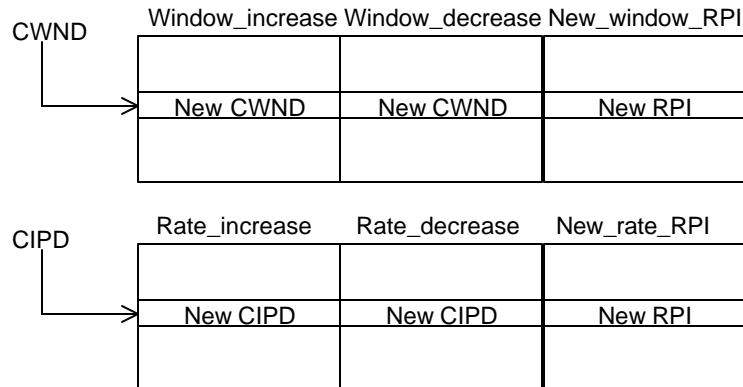
The value of RPI for a flow determines when to increase its rate/window limit. Each time an unmarked ACK packet is received by the source CA, and the value of RWL indicates the flow is rate limited or window limited, then RPI is decremented (without crossing zero) by the number of data packets covered by the notification. Note that simply subtracting PSNs is insufficient for this purpose because the PSN may take a big jump upon transmission of an RDMA read request.

Once RPI reaches zero, then an increase occurs in three steps:

- 1) The limiting factor of the flow (rate or window, as determined by RWL) is increased. The new value for CWND or CIPD is determined by functions that are specified by global tables (the tables, [Figure 2](#), are merely conceptual — a table-based implementation is not required). One table, indexed by CWND, specifies the next value to be assigned to CWND for a window increase. The other, indexed by CIPD, specifies the next value to be assigned to CIPD for a rate increase.
- 2) RPI is assigned a new value that specifies the number of data packets to be acknowledged via unmarked ACK packets before the next increase of the rate/window.

The new value depends on the revised CIPD setting if RWL identifies rate as the limiting factor, or else it depends on the revised CWND. Furthermore, for the case of rate limiting, the value must take into account differences between flows in PMTU and source CA link speed. Otherwise, competing flows that have identical rate limits but differ in their PMTU or source CA link speed could have large differences in the delays required before the next rate/window increase is allowed. The new value for RPI may be obtained from a conceptual, global table that uses normalized units in which the value 1 equals the time required to transmit one maximum-size packet (MTU=4096) on a 1x link. In that case, to convert to the appropriate setting for RPI, the table value is multiplied by the ratio of MTU (4096) to the flow's PMTU, and also by the ratio of the actual CA link speed to the 1x link speed. The resulting product is assigned to RPI.

- 3) APSN is set to the PSN of the next packet transmitted for the flow.



**Figure 2 Source Response Function Implementation Template**

### 5.1.3.2 RATE/WINDOW DECREASE

As stated in [Section 5.1.1](#), the reception of just a single marked ACK packet will cause a flow rate/window decrease. Accumulation of marked ACK packets is not required. However, not all marked ACK packets need to cause decrease adjustments. In particular, a decrease adjustment will be caused only by the first marked ACK packet that corresponds to a data packet injected after the most recent adjustment of the flow's rate/window. That is, only marked ACK packets for data packets with higher PSN than that identified by the APSN variable are used to cause a decrease. This idea was introduced by TCP Vegas [\[3\]](#) to avoid reacting to congestion notification that reflects the network state that resulted from an old setting of the flow's injection limit. In addition, the policy avoids discriminating against flows with small PMTU, for which a large number of packets may be marked in a burst upon congestion detection in the fabric. Only the first packet will cause the source to decrease the flow's window/rate, and the rest of the burst will be ignored.

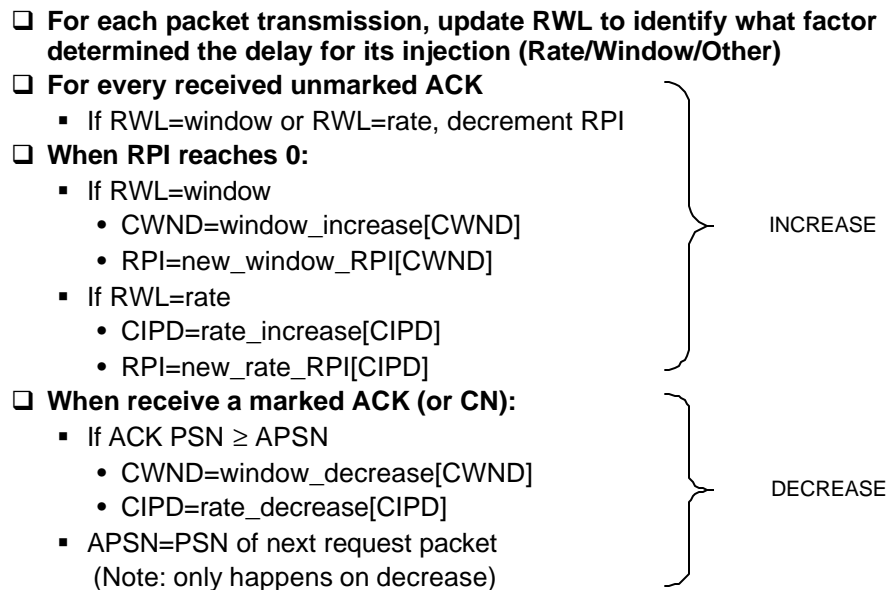
Alternatives to ignoring the subsequent marks could be envisioned. One possibility would be to make small adjustments for each mark received, where the size of an adjustment increases with the packet size or PMTU. Another possibility would be to accumulate marks before making an adjustment, where the number to accumulate decreases with the packet size or PMTU. These alternatives would require an amount of state that is similar to the APSN-based approach specified here.

A decrease adjustment consists of two steps:

- 1) Both the CIPD rate limit and the CWND window limit are assigned new values obtained from conceptual tables that are similar to those for the increase function.
- 2) APSN is assigned the current PSN, just as for flow rate/window increase.

We chose not to re-initialize RPI when the rate or window limit is decreased. This is in order to distinguish between flows with the same rate or window limits that have accumulated different amounts of unmarked ACK packet receptions or data packet transmissions. Flows that stayed longer at a given rate/window than other flows before receiving a mark should return to this rate/window in a shorter time, because the former had been closer to increasing their rates/windows before receiving the mark.

[Figure 3](#) summarizes the general source response function described.



**Figure 3 General Source Response Function**

## 5.2 RATE CONTROL: UNRELIABLE TRANSPORT AND RDMA READ RESPONSE

For unreliable transport services and RDMA read responses, a flow is identified by SLID/DLID/SL<sup>1</sup>, because the packets transmitted between one SLID/DLID pair on one SL use the same sequence of links and VLs in the subnet fabric. Each flow that uses an unreliable transport service or RDMA read response is purely rate controlled. Its congestion control state is associated with the flow's SLID/DLID/SL triple. The only state variables required for congestion control for these flows are CIPD, RPI, APSN, and RWL.

The CIPD rate limit of unreliable flows could be initialized to the static IPD (SIPD) determined during path discovery as was the case for reliable flows. This would start the flow's

1. Source Local Identifier (SLID) and Destination Local Identifier (DLID) are the source and destination addresses; Service Level (SL) identifies the class of service of the flow

---

packet injection at the maximum rate that can be supported by the path, since there is no additional window mechanism limiting the packet injection. This may be acceptable if unreliable flows are expected to be small with a very few packets. Otherwise, this could be quite harmful. It would be better to support a slow start in such cases by initializing the CIPD to an IPD value higher (lower rate) than the SIPD. This initial IPD value could be an offset from the SIPD value that is determined statically based on a knowledge of the specific subnet and its application characteristics. Alternatively, the Subnet Manager can assign an initial IPD value based on its knowledge of the fabric and its current flow assignments.

Since ACK packets are absent with unreliable transport services and RDMA read responses, the description of rate control for reliable transport services must be additionally modified for flows that use unreliable transport services and RDMA read responses as follows:

- RWL identifies whether the flow is currently rate limited or limited by other factors such as generation of packets by the application
- Rate is decreased in response to a single CN packet
- RPI is the remaining number of data packets to transmit before increasing the flow's rate limit
- RPI is decremented by one each time a data packet is transmitted if the flow is rate limited.

## **6 NON-COMPLIANT DEVICES**

In this section we briefly discuss how this proposal can be used on a heterogeneous environment composed by a mix of spec-1.0-compliant and post-spec-1.0-compliant devices.

### **6.1 NON-COMPLIANT SWITCHES**

Switches that are non-compliant will not be able to mark packets. Thus congestion spreading affecting a non-compliant switch, either because it is generated at the switch or propagated from neighbor switches, will likely to be spread to neighbor switches. However as soon as congestion spreading reaches compliant switches these switches will start marking packets that are propagating the congestion tree and congestion spreading will be contained. Therefore congestion spreading will be contained in regions of the subnet composed by non-compliant switches. Therefore the proposed mechanism can be used with non-compliant switches without any modification.

A possible optimization would be to make compliant switches to mark packets destined to blocked output VLs, when the next switch on the link is a non-compliant switch. Thus packets that causes congestion spreading to reach a compliant switch from a non-com-

---

pliant switch will be marked as soon as the output VL runs out of credit, instead of having to wait until congestion spreading reaches an input VL of the compliant switch, i.e until an input buffer of the compliant switch becomes full.

## 6.2 NON-COMPLIANT CAs

In order to deal with non-compliant CAs we propose to use a centrally managed congestion control mechanism to complement the low-level mechanism described in the previous sections.

The Subnet Manager should inform the switches and compliant CAs of the LIDs of non-compliant CAs. Every time a switch detects congestion and sets the ECN bit of a data packet for which the DLID corresponds to a non compliant CA, the switch should generate a CN packet to the Subnet Manager in addition to marking the packet. If a packet had the ECN bit already set to “1”, i.e. it was marked on a previous switch, the current switch should not generate a new CN packet, since the previous switch would have already done so. The implementation of these functions need not be in the critical path of switch operation.

When a destination CA receives a marked data packet from a non-compliant source CA, it should send a CN packet to the subnet manager.

When the Subnet Manager receives CN packets for a flow, it should adjust the rate of the flow at its source CA. This can be done by requesting the source CA to change its static IPD for the affected flow.

## 7 REFERENCES

- [1] <http://www.infinibandta.org>
- [2] D. Chiu and R. Jain, “Analysis of the increase and decrease algorithms for congestion avoidance in computer networks,” *Computer Networks and ISDN Systems*, 17(1), June 1989, pp. 1-14
- [3] 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*, 13(8), October 1995, pp. 1465-1480.