InfiniBand Congestion Control

HP Proposal

Overview

- ❑ Propose combination of a fast, low-level mechanism with a centrally managed mechanism (e.g. Compaq's)
  - ★ Low-level: Fast reaction to short-term demand variations
  - ★ Centrally managed: Long-term rate control (e.g. non-compliant CAs)
- ❑ Focus on low-level mechanism:
  - ★ Congestion control independent for each VL
  - ★ Targets both generation and propagation of congestion spreading
  - ★ Uses Explicit Congestion Notification (ECN)
  - ★ Hybrid rate-window control
  - ★ Simple

Approach

❑ Window Mechanism:

- **Advantage:** self-clocked (respond to bandwidth changes quickly**)**
- **Disadvantage:** buffer utilization >= 1 packet

❑ Rate Control Mechanism:

- **Advantage:** mean buffer utilization can be less than 1 packet (reduced congestion spreading)
- **Disadvantage**: Not self-clocked (May inject too many packets before notified of congestion)

❑ Our Proposal : Hybrid

- Limit packet injection by both a rate limit and a maximum window
  - Maximum window provides self-clocking
  - Rate limit allows low buffer utilization per flow
- Dynamically adjust rate or window based on current conditions

Explicit Congestion Notification

- **Switch detects congestion and signals end-node**
- **Forward Binary Notification (Marking packets)**
  - ⋆ Assume: each **data** packet (SEND, RDMA WRITE request, RDMA READ response packets) has one ECN Bit
  - ⋆ Any switch can set the ECN bit (i.e. “mark” the packet)
  - ⋆ No switch can reset a ECN bit set by other switch
  - ⋆ Destination copies the ECN bit from data packets to ACK packets
    - Assume special CN packets are generated when there is no ACK associated with data packet (i.e. RDMA read response and unreliable transport)
    - ACK coalescing:
      - Destination returns an ACK upon reception of marked data packet
      - At least one packet in a congestion window must have its AckReq bit set in BTH

Packet Marking Policy

- **Goal:** Mark only those packets that contribute to congestion spreading
- **Contributing Packets:**
  - Any packet using the root† of a congestion spreading tree is ***generating*** the tree
  - Any packet in a full buffer is likely ***propagating*** a tree
- **Approach:**
  - Mark packets in full buffers (propagation)
  - Use a heuristic to identify root VLs (generation)
    - Output VL that is destination for packet in full buffer is candidate root VL
    - Assume to be root until it runs out of credit or until it transmits all packets currently at the switch for this VL
  - Mark packets currently at the switch for this VL which are not in full buffers

- Policy should be practical to implement for common switch buffer organizations (input-, output-, centrally-queued)

† root VL uses all its effective bandwidth, has available credits and causes a previous input VL to block

Effective Source Response: Properties

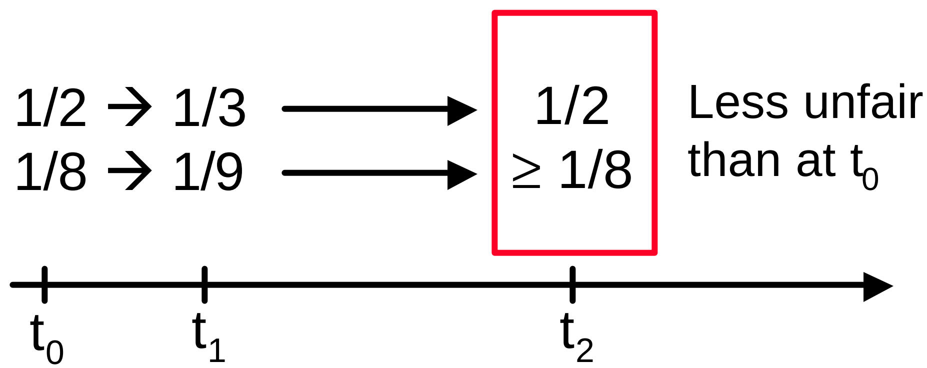
1. Larger/same response to marked packets than to unmarked packets

   Why: Marked packet means congestion spreading,
   Unmarked packet does *not* mean idle

2. Lower rate flows should recover at least as quickly as higher rate flows

   Why: Promotes fairness

   Example:



3. There exists some minimum flow rate, and rate increase from the minimum rate requires only one unmarked packet arrival

   Why: fastest recovery from minimum rate that satisfies #1

Source Response Functions

- Given a decrease response function, a suitable increase function can be derived from the three properties
- Evaluated several source response functions that satisfy the properties, including:
  - AIMD – Additive Increase Multiplicative Decrease
  - LIPDI – Linear IPD Increase (to decrease the rate)

Source Response Implementation – State

Window State	CWND: congestion window size
	OUTS: outstanding packets
IPD Rate State	CIPD: congestion control IPD rate limit
General State	RPI: remaining packets before increase
	APSN: PSN of most recent decrease
	RWL: rate/window limit flag

Implementation for Different Transports

Transport Service	Congestion Control State	State Associated With	Congestion Notification
Reliable Connection	Window/Rate/General	QP	ACK
Reliable Datagram	Window/Rate/General	EEcontext	ACK
Unreliable and RDMA Read Response	Rate/General	SLID/DLID/SL	CN packet

Source Response Implementation – Function Template

- Function common to all flows
- Function can be represented as tables

CWND →

Window_increase	Window_decrease	New_window_RPI
New CWND	New CWND	New RPI

CIPD →

Rate_increase	Rate_decrease	New_rate_RPI
New CIPD	New CIPD	New RPI

General Response Function

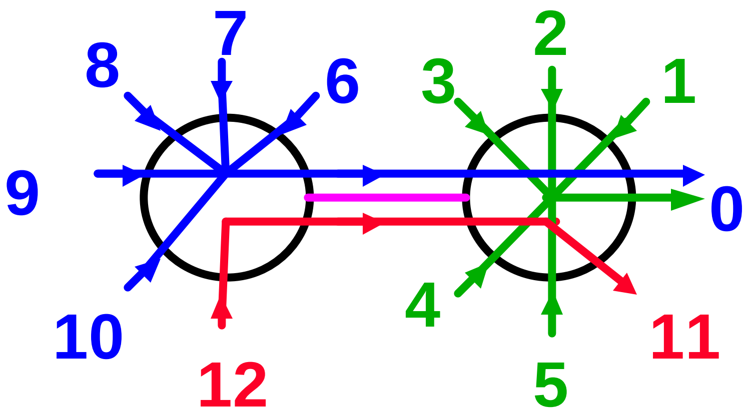
- For each packet transmission, update RWL to identify what factor determined the delay for its injection (Rate/Window/Other)
- For every received unmarked ACK
  - ⋆ If RWL=window or RWL=rate, decrement RPI
- When RPI reaches 0:
  - ⋆ If RWL=window
    - CWND=window_increase[CWND]
    - RPI=new_window_RPI[CWND]
  - ⋆ If RWL=rate
    - CIPD=rate_increase[CIPD]
    - RPI=new_rate_RPI[CIPD]

(INCREASE)

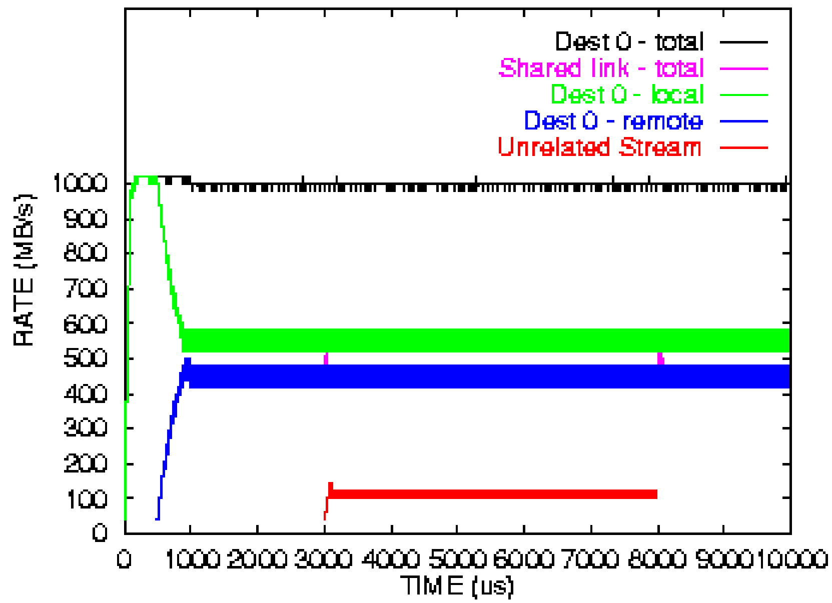
- When receive a marked ACK (or CN):
  - ⋆ If ACK PSN ≥ APSN
    - CWND=window_decrease[CWND]
    - CIPD=rate_decrease[CIPD]
  - ⋆ APSN=PSN of next packet
    (Note: only happens on decrease)

(DECREASE)

Simulation Results



window = 1,
no IPD rate control



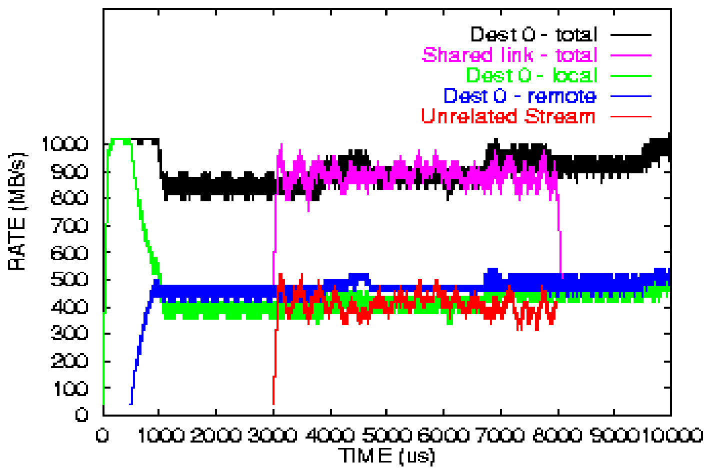
Window is not enough!

Basic Problem:

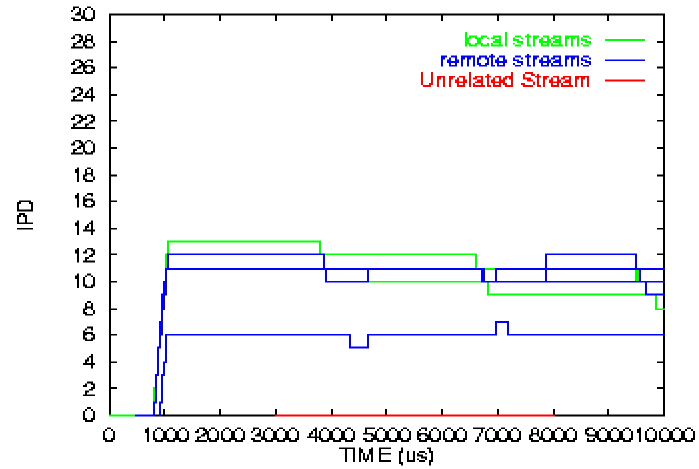
- **Small buffers in switches**
- **Several flows can exhaust buffers**

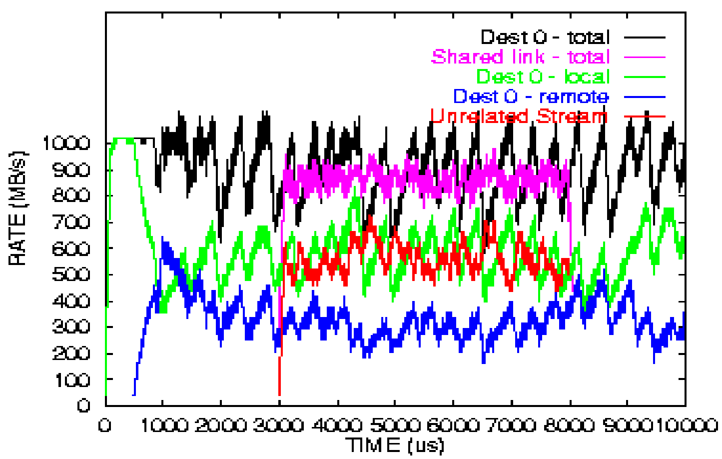
Packet size: 2 KB **Input Port Buffer: 8 KB** **Link BW: 1 GB/s**

Results with IPD Rate Control (Window=1)



LIPDI





AIMD:
decrease by ½