End-to-End Congestion Control for System Area Networks

Renato Santos, Yoshio Turner, John Janakiraman

Linux Systems, Networks Department
Internet Systems and Storage Laboratory

Problem: Network Congestion

- Cause: Network congestion arises when injected traffic exceeds network capacity

- Effect: Performance degradation to levels below what could be achieved in the absence of congestion
  - Need a congestion control mechanism

- Our focus: Cong. Control for System Area Networks (SAN)
  - Previous work: focused on traditional TCP networks
  - SAN has several unique characteristics that make the congestion control problem unique in this environment

Outline

- Motivation
- Part 1: Congestion Detection and Notification
- Part 2: Source Response
- Conclusion

System Area Networks (SAN)

- High speed and low latency interconnect for high performance I/O and cluster communication
  - Data rates: 10s of Gb/s
  - Latency: 100s of ns to a few ms, end to end delay
- Examples of proprietary SANs
  - Myrinet, Quadrics, Memory Channel (HP), ServerNet (HP)
- InfiniBand: Industry Standard for SAN

SAN Characteristics and Congestion Control Implications

- No packet dropping (link level flow control)

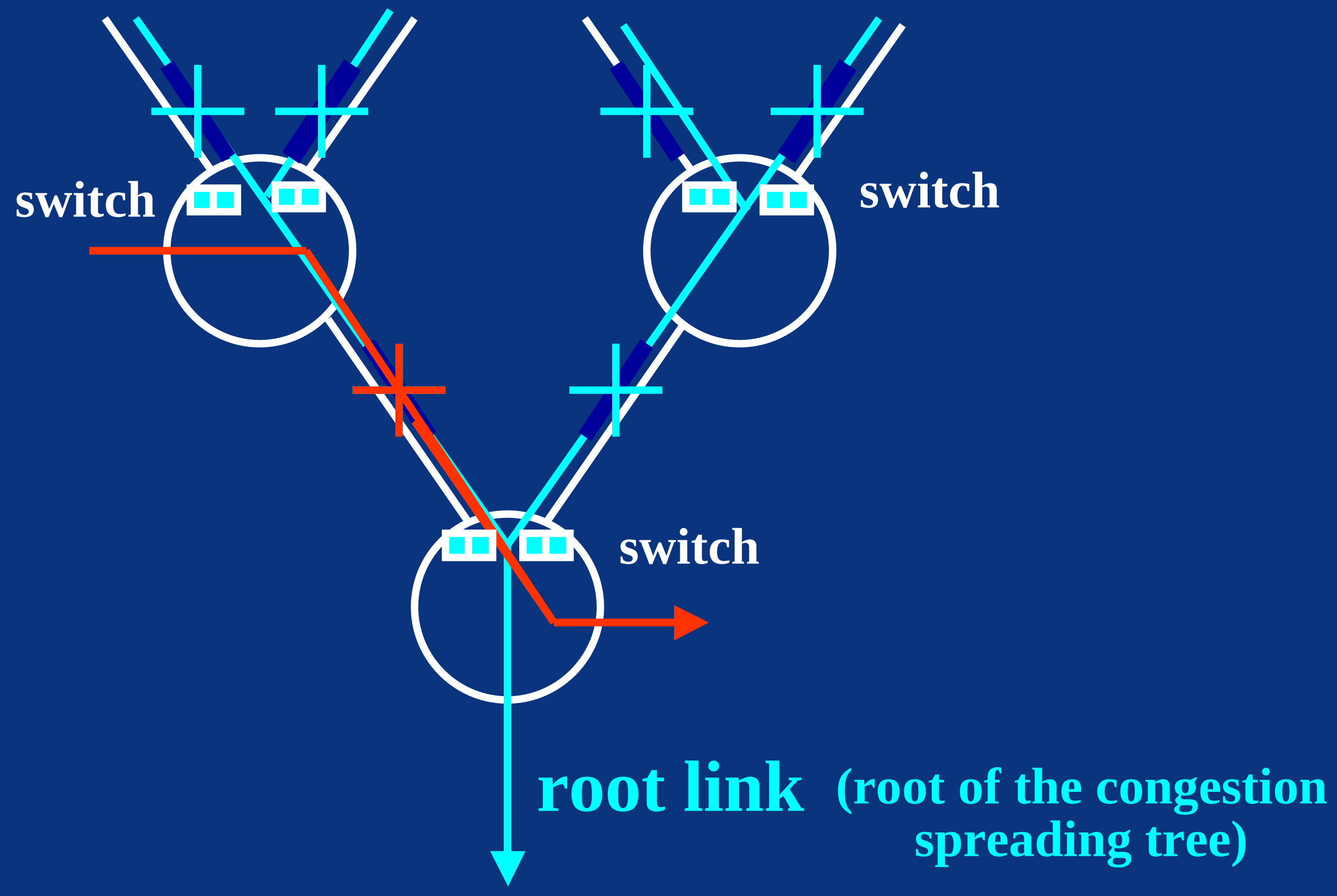
  à Need network support for detecting congestion
- Low network latency (tens of ns cut-through switching)

  à Simple logic for hardware implementation
- Low buffer capacity at switches (e.g., 8KB buffer per port can store only 4 packets of 2KB each)

  à TCP window mechanism inadequate
  (narrow operational range)
- Input-buffered switches

  à Alternative congestion detection mechanisms

Problem: Congestion Spreading



Avoiding Congestion Spreading

- **Congestion Control Mechanism (feedback-control loop)**
  - **Congestion detection mechanism (feedback)**
    - Detect when congestion is forming
  - **Source response (control)**
    - Adjust flows injection rate based on feedback to avoid congestion
    - Discussed in the 2nd part of this talk

Congestion Detection and Notification

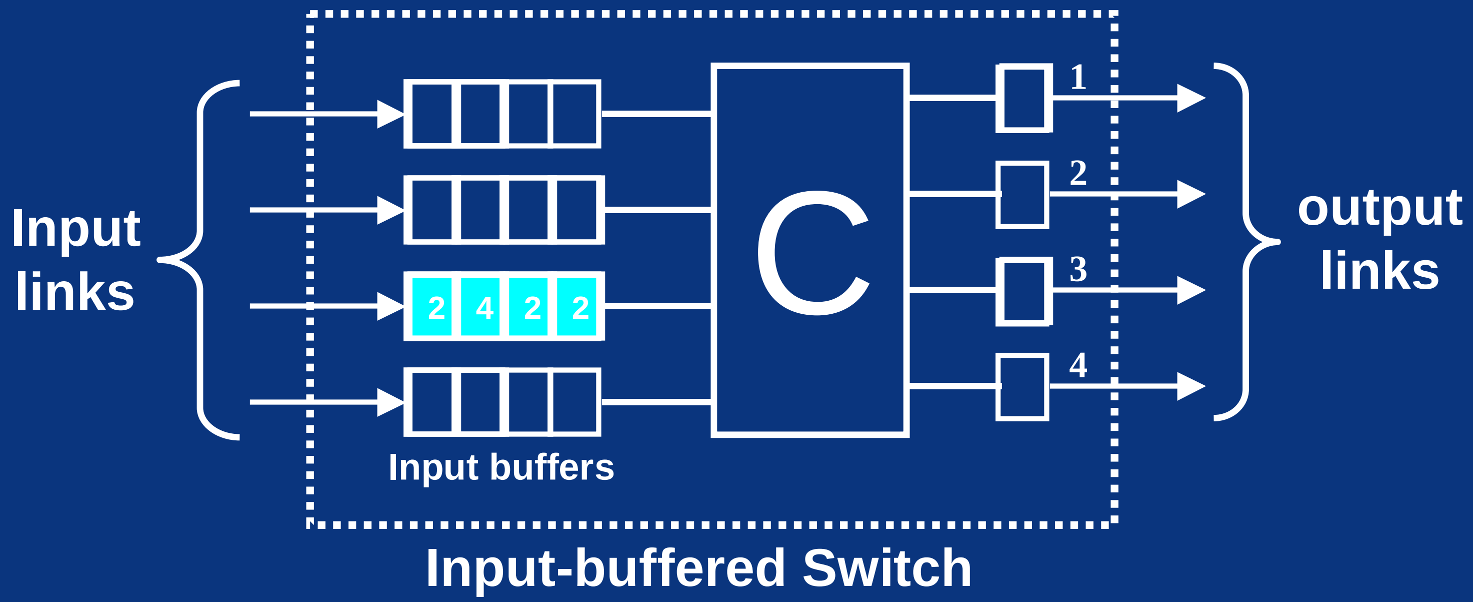
- **Need network support for detecting congestion**
  - **Cannot use packet loss at end nodes to detect congestion**

- **ECN (Explicit Congestion Notification) approach**
  - **Switch detects congestion when switch buffer becomes full**
  - **Switch sets a *congestion bit* on headers of packets in full buffer (packet marking)**
  - **Destination node copy *congestion bit* (mark) into ACK packet**
  - **Source adjusts flow rate according to the value of the *congestion bit* (mark) received in ACK packet.**

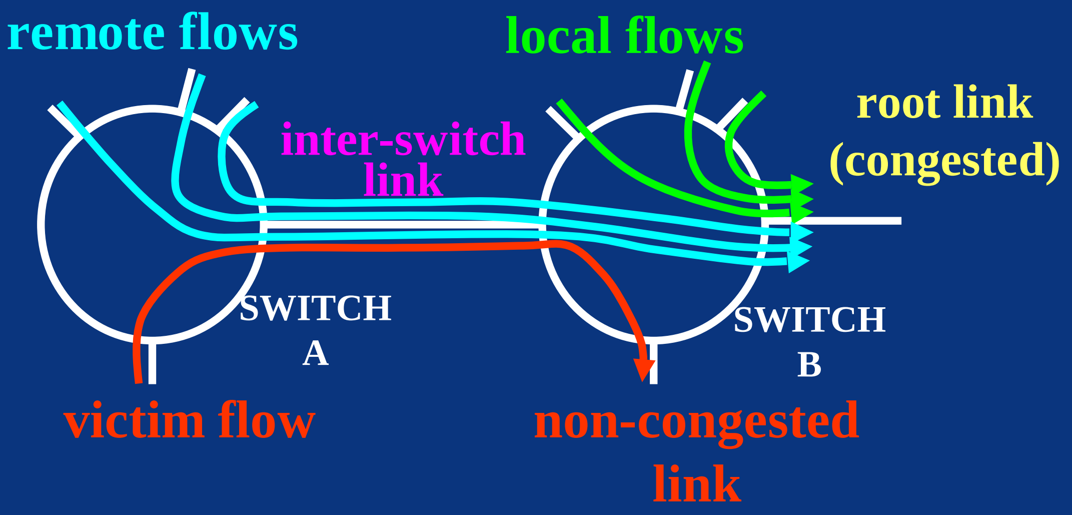
- **What is unique in our ECN mechanism?**
  - **Packet marking appropriate for input-buffered switches**
  - **Simple to implement in hardware**

Naive Approach: Marking Packets in Full Buffers

- When an input buffer becomes full:
  Mark all packets in input buffer



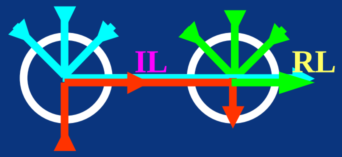
Simulation Scenario



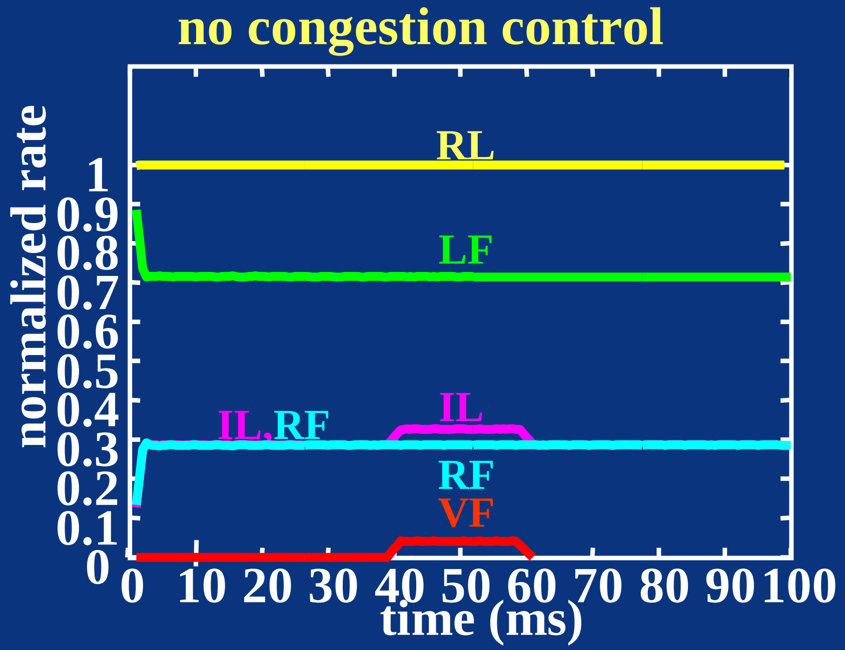
congestion spreading in this scenario:

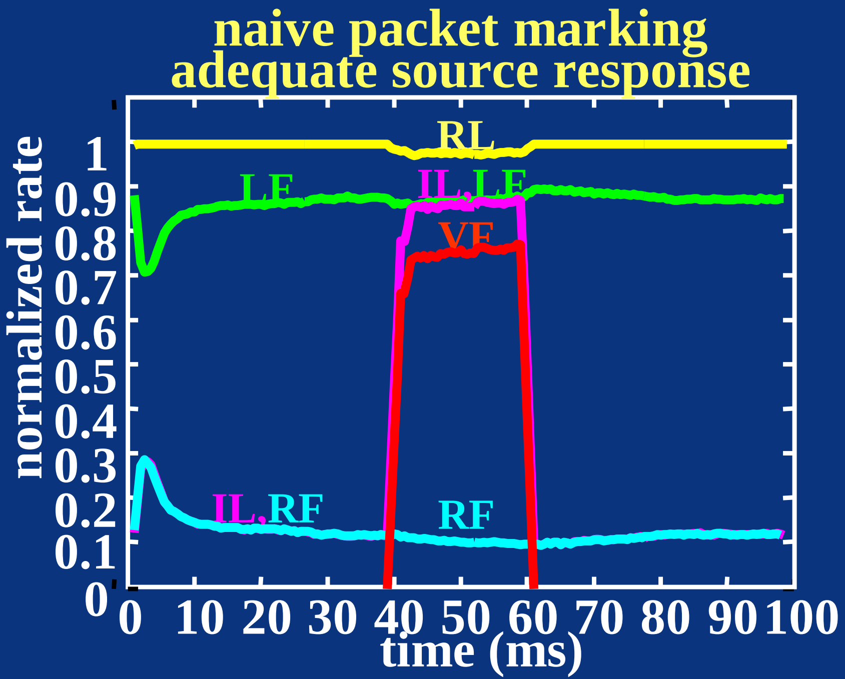
- contention for root link
- buffer used by remote flows fills up
- inter-switch link blocks
- victim flow cannot use available inter-switch link bandwidth

- Assumptions:
  - 10 local flows + 10 remote flows + 1 victim flow
  - All flows are greedy (try to use all BW available)
  - Buffer Size: 4 packets/input_port
  - Sources react to packet marking using an adequate response function (discussed later)



Simulation Results





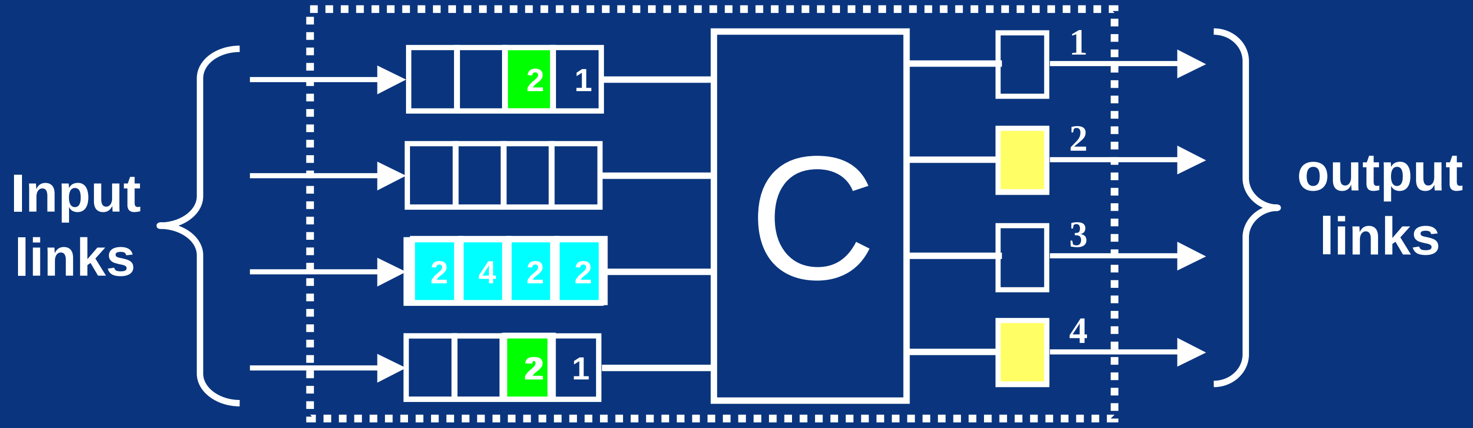
local flows (LF) 🟩 remote flows (RF) 🟦 victim flow (VF) 🟥 root link (RL) 🟨 inter-switch link (IL) 🟪

- Effectively avoiding cong. spreading
- Unfairness (remote vs. local flows)
  – shared full buffer causes remote packets to be marked more frequently than local packets
  – local flows get higher share of BW

Input-Triggered Packet Marking

- **Goal: Improve fairness**
  - **Mark all packets using congested link**
  - **Not only packets in full buffer**

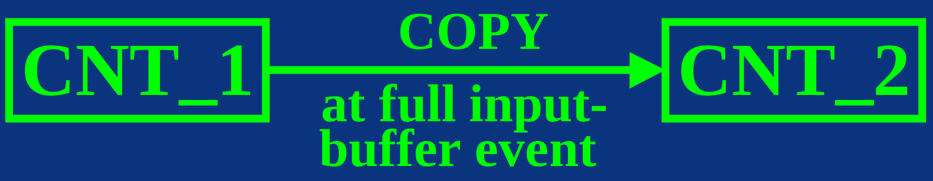
- Marking triggered by a full input buffer
- Mark all packets in input buffer
- Identify root (congested) links:
  - Destination of packets at full buffer
- Mark any packet destined to root links



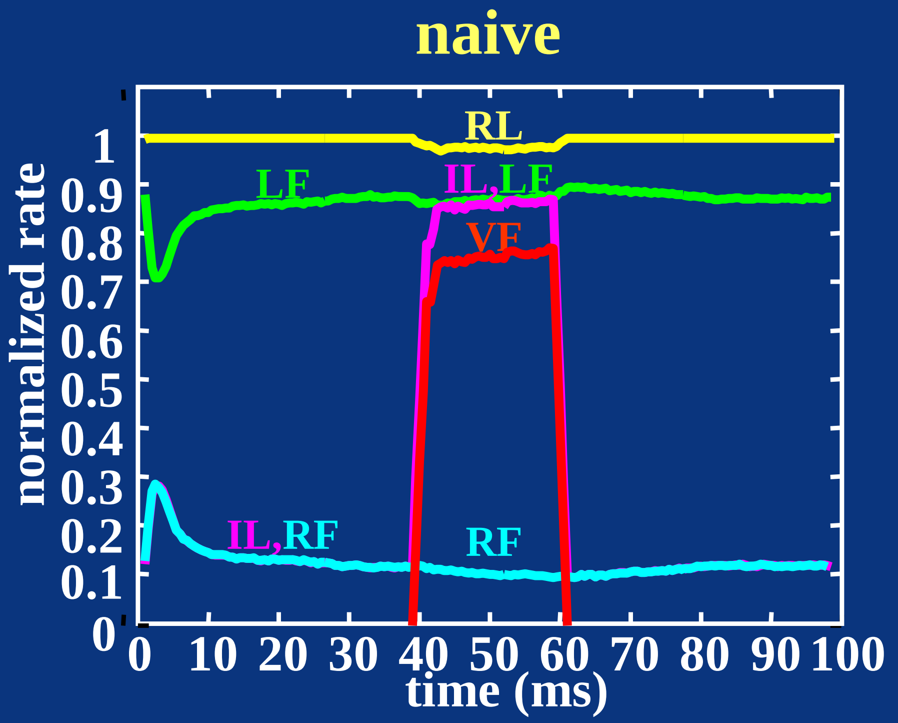
Efficient Implementation

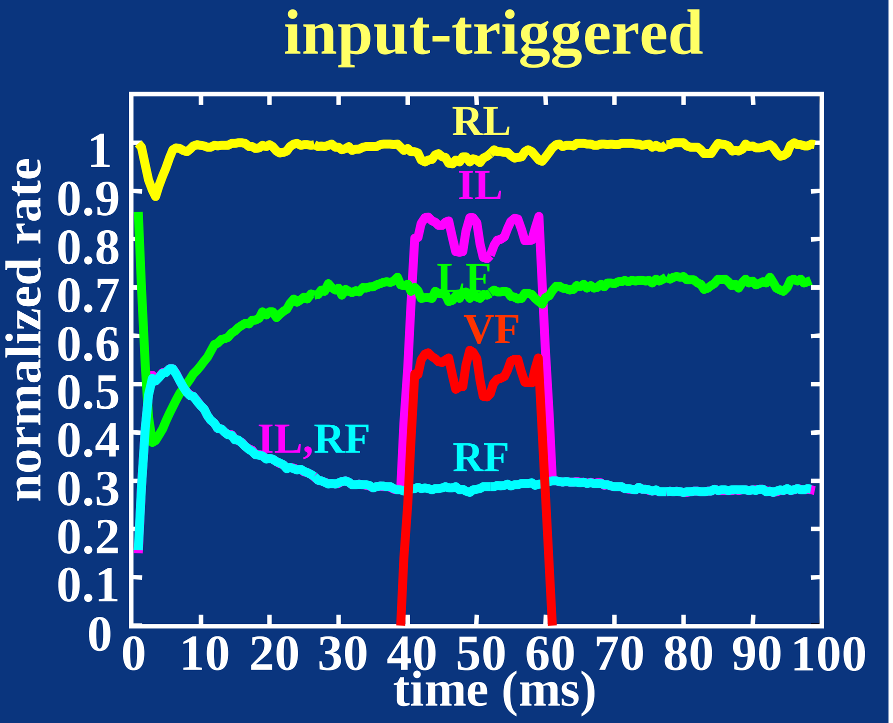
- Use counters to avoid expensive scan of all switch packets (when searching for packets destined to a congested link)

- 2 counters per output port
  - CNT_1: Total number of packets in the switch that are destined to this output port.
  - CNT_2: Total number of packets destined to this output port that need to be marked

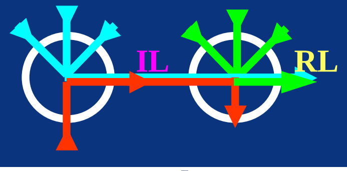


Input-Triggered Packet Marking





local flows (LF) 🟩
remote flows (RF) 🟦
victim flow (VF) 🟥
root link (RL) 🟨
inter-switch link (IL) 🟪



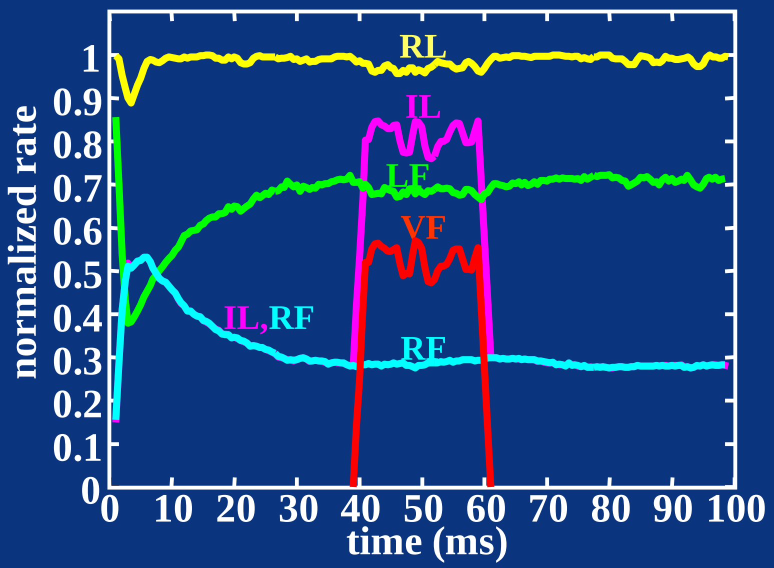
- Fairness Improved (still some unfairness)
- Marking still triggered by remote packets (bias marking towards remote packets)

Input-Output-Triggered Packet Marking

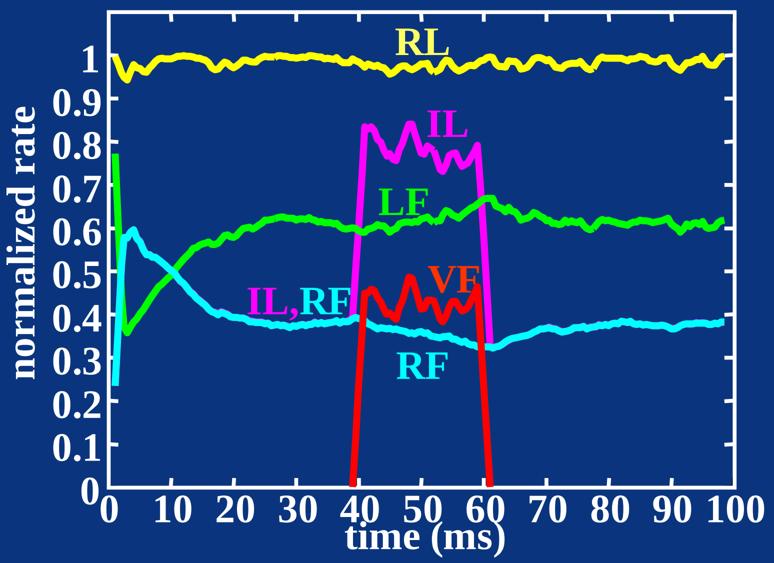
- Additional output triggered mechanism
  - Mark packets when total number of packets destined to an output port exceeds a threshold
- Still mark packets when input buffer is full (input triggered)
  - To avoid link blocking and congestion spreading

Input-Output-Triggered Packet Marking

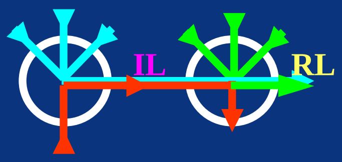
Input-Triggered



Input-Output-Triggered (threshold: 8 packets)



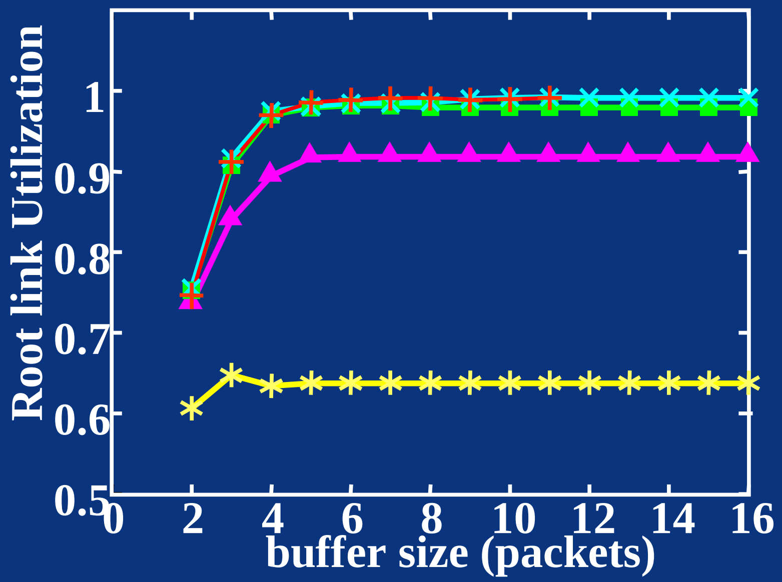
local flows (LF) 🟩
remote flows (RF) 🟦
victim flow (VF) 🟥
root link (RL) 🟨
inter-switch link (IL) 🟪



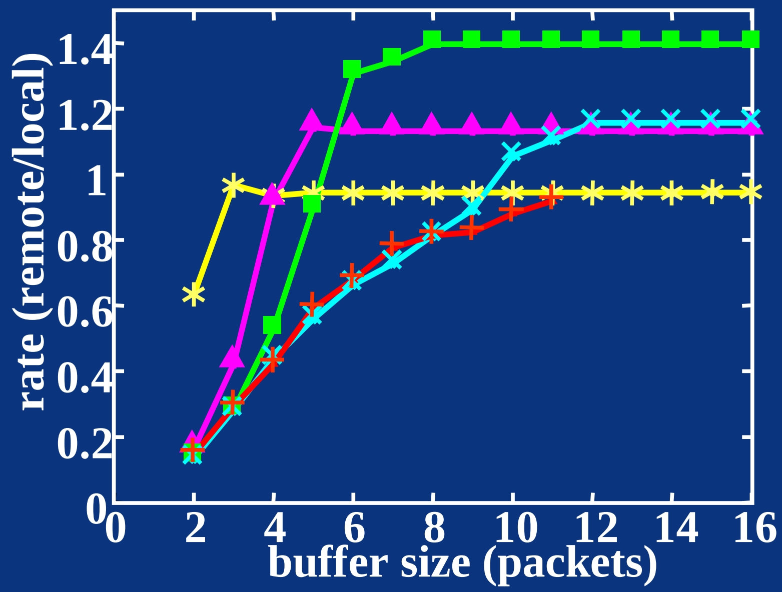
- High Bandwidth Utilization
- Better fairness than input-triggered

Input-Output-Triggered Packet Marking

efficiency



fairness



- Right threshold value need to be tuned (function of buffer size and traffic pattern)

Proposed Packet Marking Mechanism

- Input-triggered packet marking
  - Improve fairness over naive approach
  - Simple to implement
  - Does not require tuning

Part 2: Source Response

Source Response: Window or Rate Control

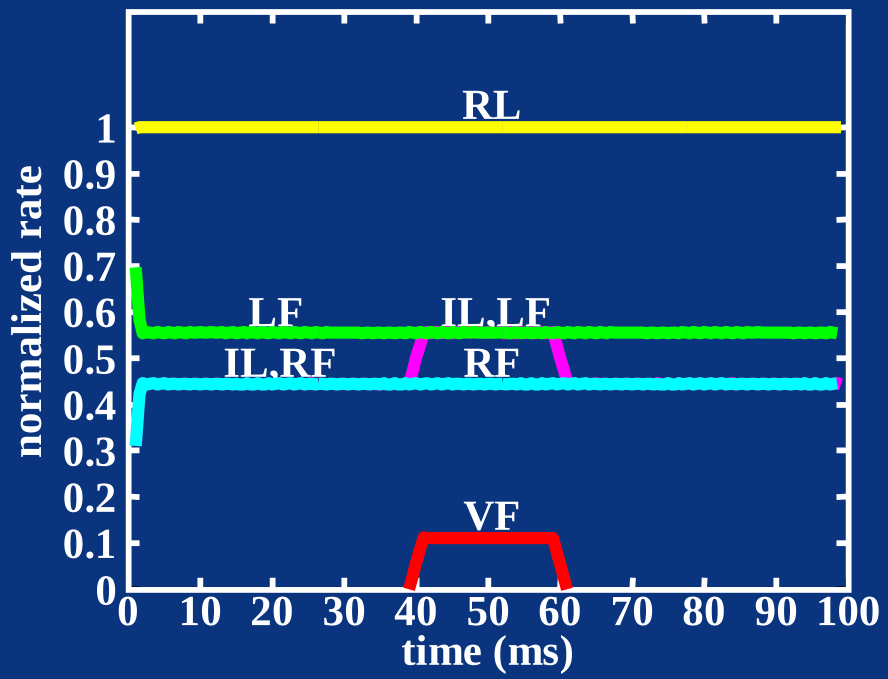
- Flow source adjusts injection in response to ECN
- Rate Control
  - Flow source adjusts rate limit explicitly

    (e.g., Enforce by adjusting delay between packet injections)
- Window Control (e.g., TCP)
  - Flow source adjusts window = # of outstanding packets
    Corresponds to rate = window/RTT (round trip time)

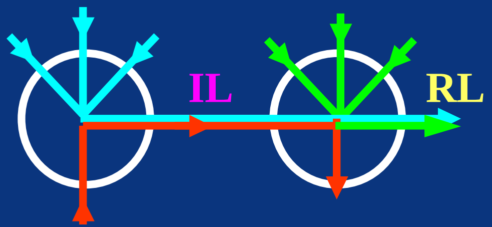
Window Control

- Advantages
  - Self-clocked: congestion à - RTT à instant ¯ rate (rate = window/RTT)
  - Window size bounds switch buffer utilization
- Disadvantage: Narrow operational range for SANs
  - Window=2 uses all bandwidth on path in idle network
    - Cut-through switching à packet header reaches destination before source can transmit last byte
  - Window=1 fails to prevent congestion spreading if # flows > # buffer slots at bottleneck

Congestion Spreading (Window=1)



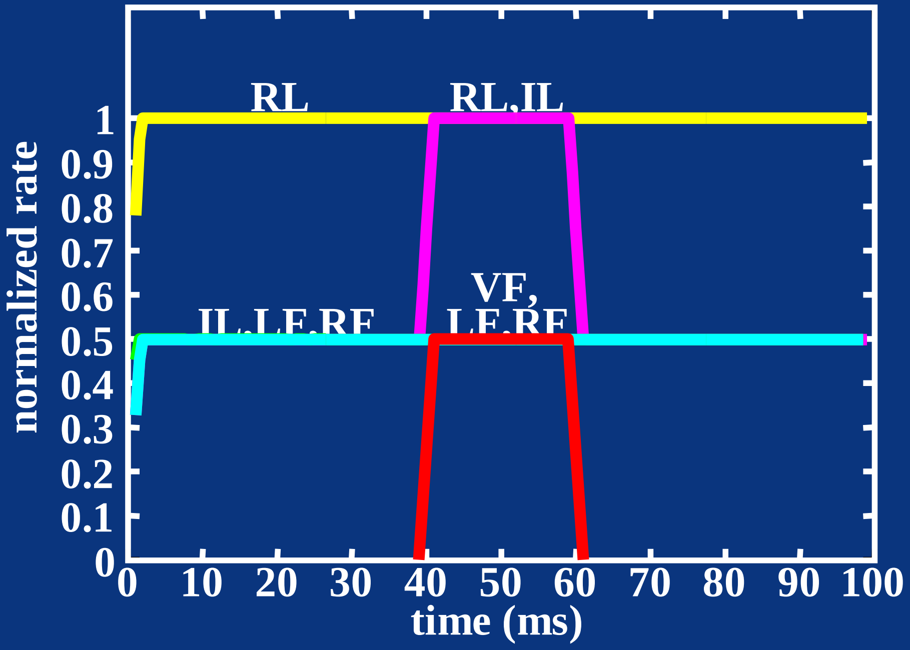
5 local flows, 5 remote flows, 4 buffer slots



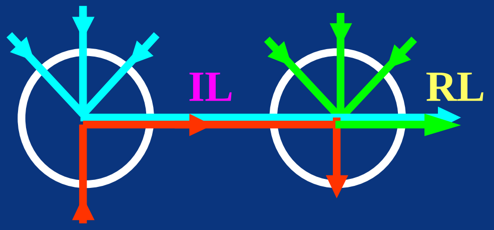
Rate Control

- Advantages:
  - Low buffer utilization possible (< 1 packet per flow)
  - Wide operational range
- Disadvantage: Not self-clocked

Fixed Optimal Rates



5 local flows, 5 remote flows, 4 buffer slots



Proposed Source Response Mechanism

- Rate control with a fixed window limit (window=1 packet)
  - Wide dynamic range of rate control
  - Self-clocking provided by the window (window=1 nearly saturates path bandwidth in low latency SAN)
- Focus on design of rate control functions

Designing Rate Control Functions

- Definition: When source receives ACK

  Decrease rate on marked ACK: $r_{new} = f_{dec}(r)$
  Increase rate on unmarked ACK: $r_{new} = f_{inc}(r)$

- $f_{dec}(r)$ and $f_{inc}(r)$ should provide :
  - Congestion avoidance
  - High network bandwidth utilization
  - Fair allocation of bandwidth among flows
- Develop new sufficient conditions for $f_{dec}(r)$ & $f_{inc}(r)$
  - Exploit differences in packet marking rates across flows to relax conditions
    - Requires novel time-based formulation

Avoiding Congested State

- Steady state: flow rate oscillates around optimal value in alternating phases of rate decrease and increase
- Want to avoid time in congested state

Congestion Avoidance Condition:

$f_{inc}(f_{dec}(r))$ £ r

- Magnitude of response to marked ACK is larger or equal to magnitude of response to unmarked ACK

Fairness Convergence

- [Chiu/Jain 1989][Bansal/Balakrishnan 2001] developed convergence conditions assuming all flows receive feedback and adjust rates synchronously
  - Each increase/decrease cycle must improve fairness
- Observation: In congested state, the mean number of marked packets for a flow is proportional to the flow rate.
  - bias promotes flow rate fairness

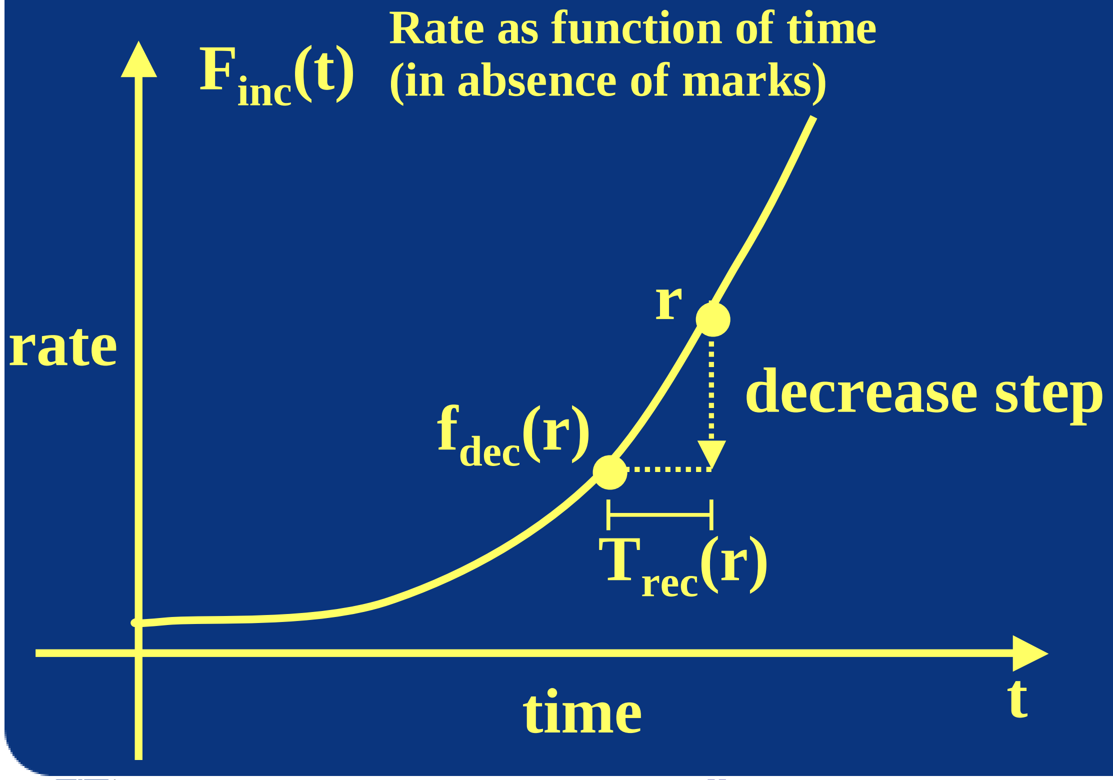
à Enables weaker fairness convergence condition

à Benefit: fairness with faster rate recovery

Fairness Convergence

Relax condition: rate decrease-increase cycles need only <u>maintain</u> fairness in the synchronous case

– If two flows receive marks, lower rate flow should recover earlier than <u>or in the same time</u> as higher rate flow



Fairness Convergence Condition:

$T_{rec}(r1)$ £ $T_{rec}(r2)$
for r1 < r2

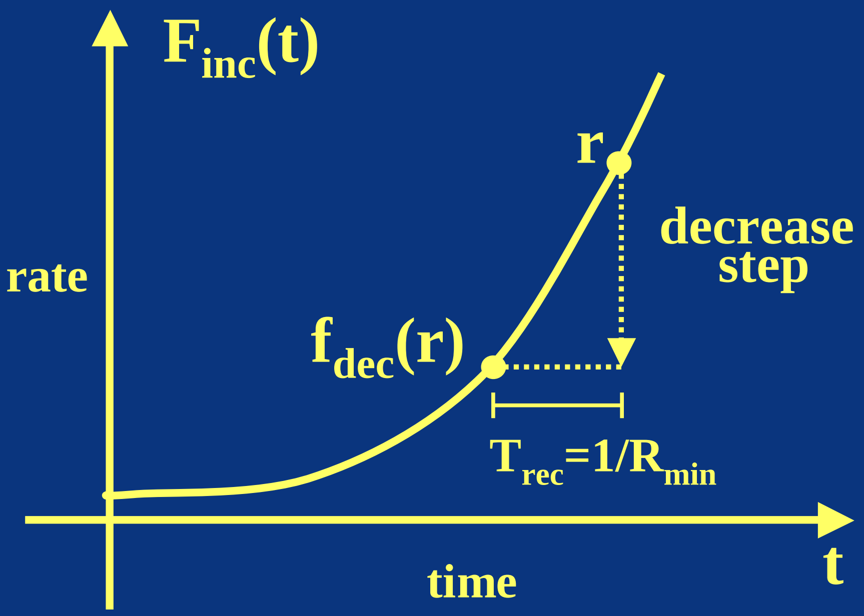
Maximizing Bandwidth Utilization

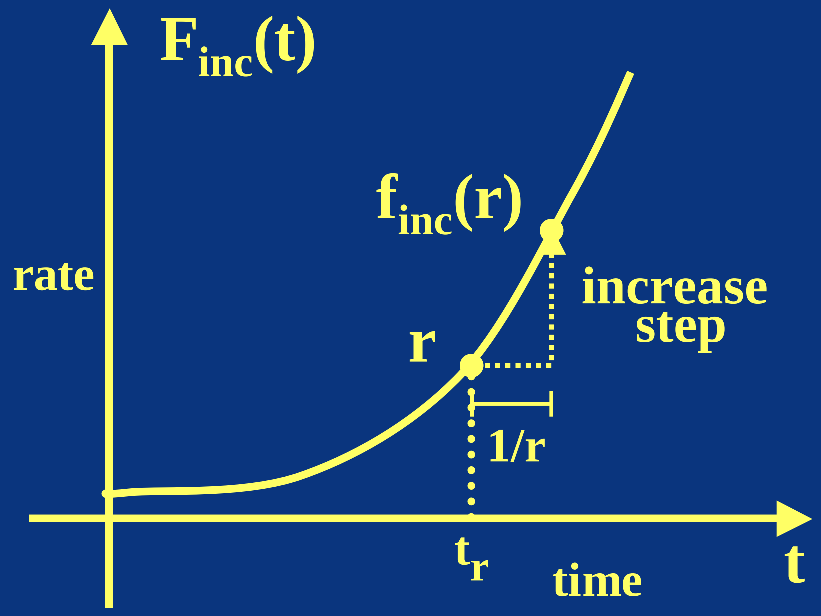
- Goal: as flows depart, remaining flows should recover rate quickly to maximize utilization
- Fastest recovery: use limiting cases of conditions
  - Congestion Avoidance Condition $f_{inc}(f_{dec}(r))$ £ r
    Use $f_{inc}(f_{dec}(r)) = r$ for minimum rate R_{min}
    - Recovery from decrease event requires only one unmarked ACK at rate R_{min} (time = $1/R_{min}$)
  - Fairness Convergence Condition $T_{rec}(r1)$ £ $T_{rec}(r2)$
    Use $T_{rec}(r1) = T_{rec}(r2)$ for higher rates

Maximum Bandwidth Utilization Condition:
$T_{rec}(r) = 1/ R_{min}$ for all r

Design Methodology:

Choose $f_{dec}(r)$, find $f_{inc}(r)$ satisfying conditions





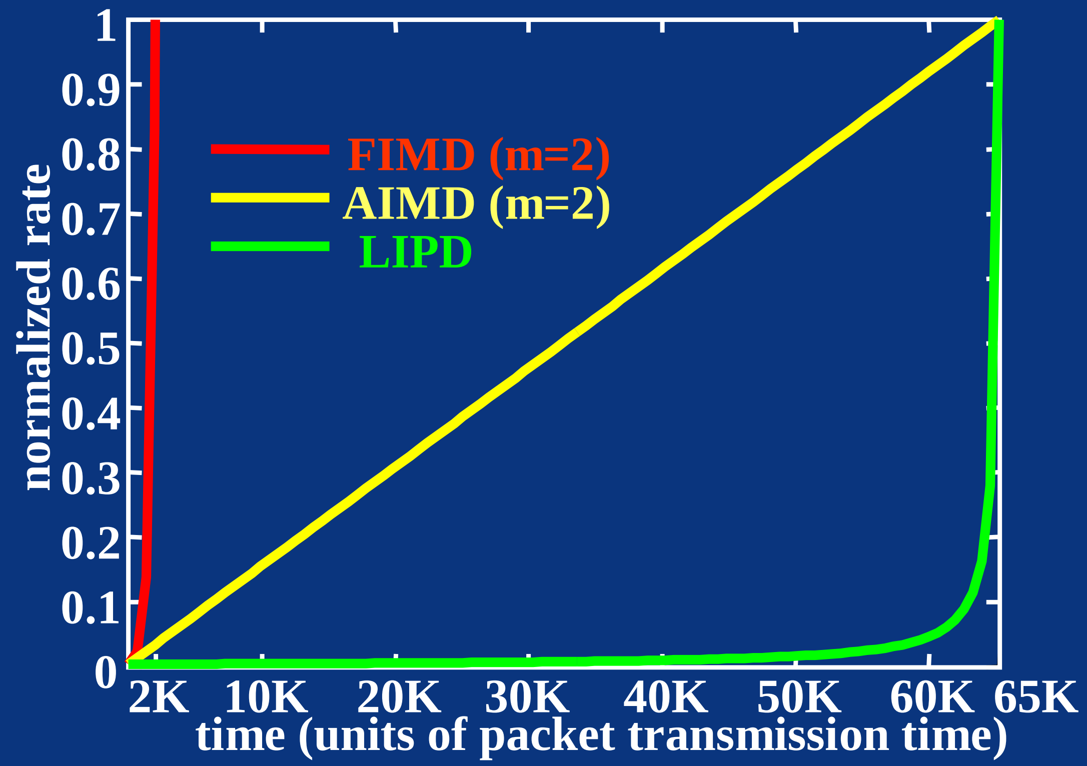
Use $f_{dec}(r)$ to derive $F_{inc}(t)$:
$F_{inc}(t) = f_{dec}(F_{inc}(t + T_{rec}))$,
$T_{rec}=1/R_{min}$

Use $F_{inc}(t)$ to find $f_{inc}(r)$:
$f_{inc}(r) = F_{inc}(t_r+1/r)$
where $F_{inc}(t_r) = r$

New Response Functions

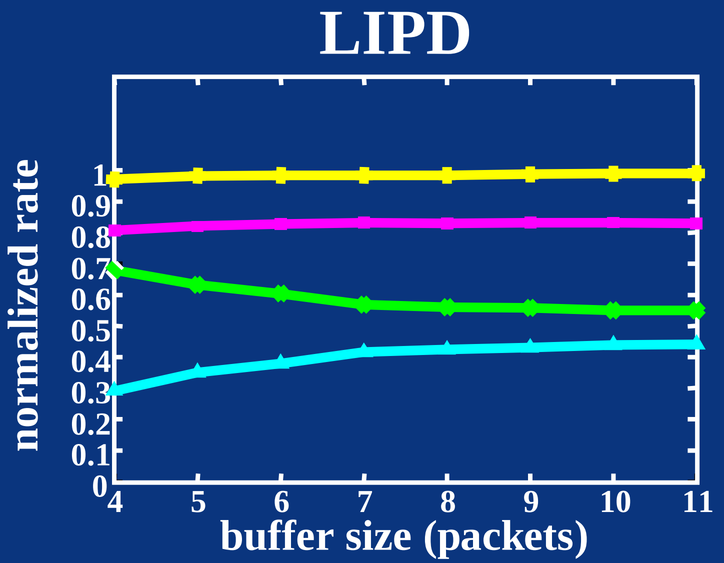
- Fast Increase Multiplicative Decrease (FIMD):
  - Decrease function: $f_{dec}^{fimd}(r) = r/m$, constant $m>1$ (same as AIMD)
  - Increase function: $f_{inc}^{fimd}(r) = r \cdot m^{Rmin/r}$
  - Much faster rate recovery than AIMD
- Linear Inter-Packet Delay (LIPD):
  - Decrease function: increases inter-packet delay (ipd) by 1 packet transmission time
    $r = R_{max}/(ipd+1)$
  - Increase function: $f_{inc}^{lipd}(r) = r/(1- R_{min}/R_{max})$
  - Large decreases at high rate, small decreases at low rate
- Simple Implementation: e.g., table lookup

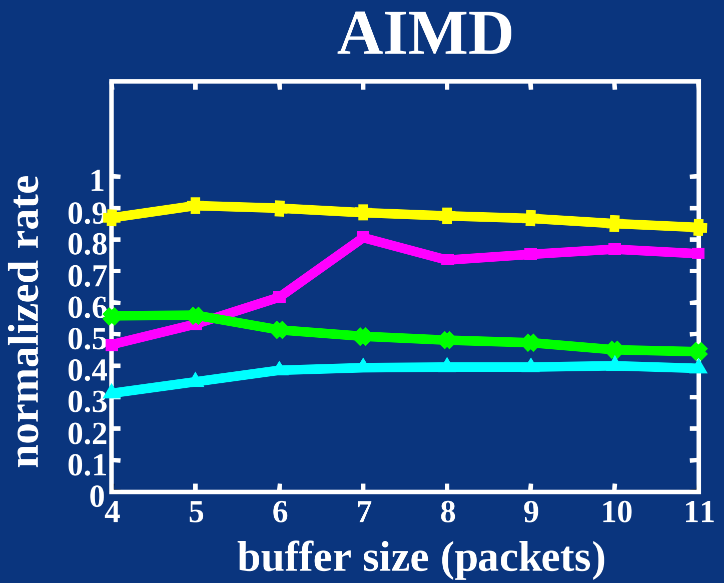
Increase Behavior Over Time : FIMD, AIMD, LIPD

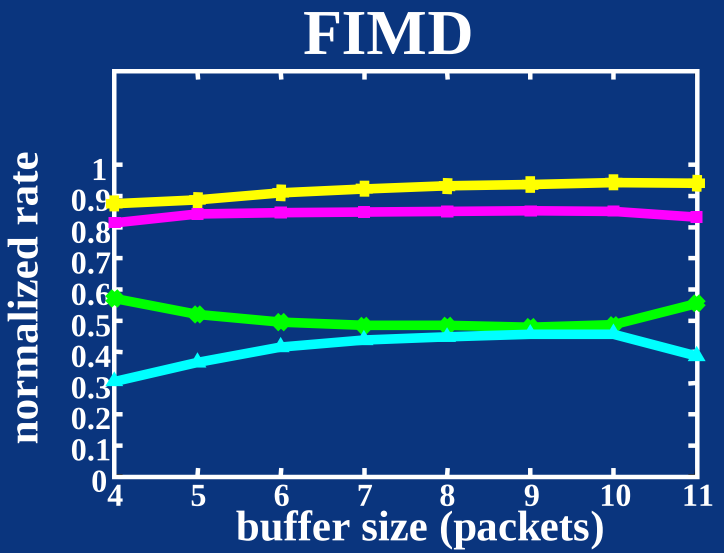


$F_{inc}(t)$

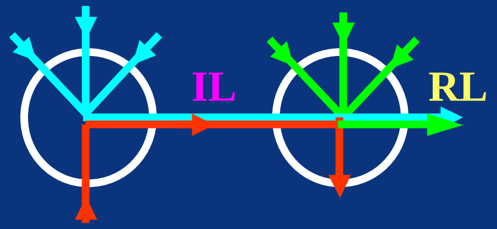
Performance: Source Response Functions



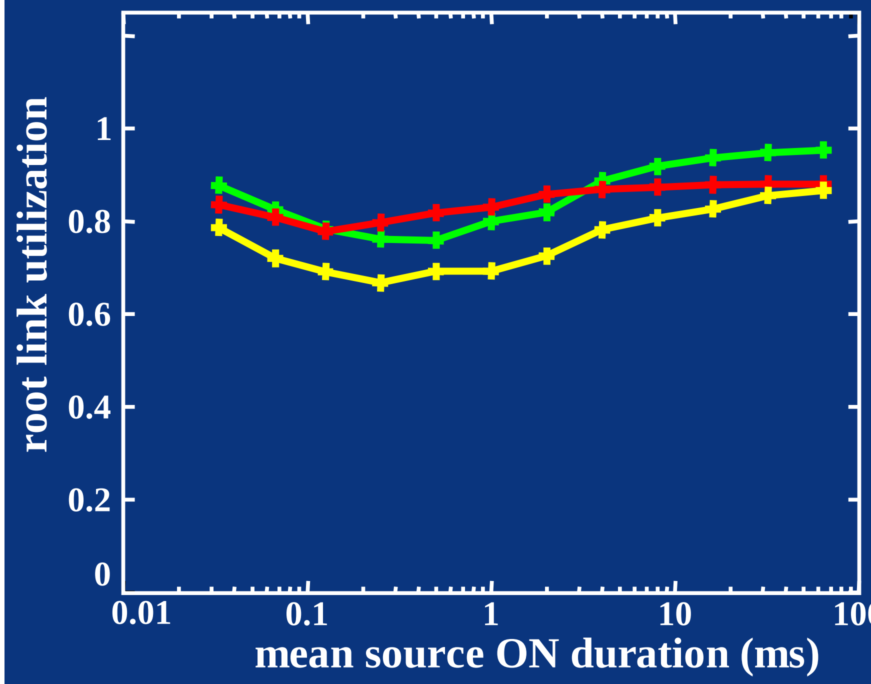


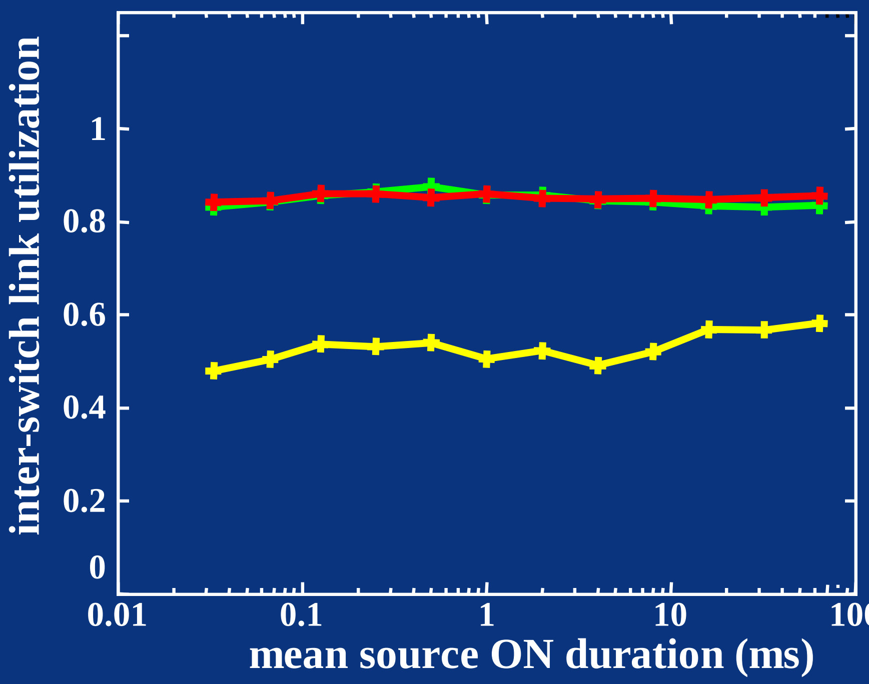


root link (RL) ▬ local flows (LF) ▬
inter-switch link (IL) ▬ remote flows (RF) ▬



Performance: Bursty Traffic





LIPD 🟩 FIMD 🟥 AIMD 🟨

Each flow: ON/OFF periods exponentially distributed with equal mean

Summary

- Proposed/Evaluated congestion control approach appropriate for unique characteristics of SANs such as InfiniBand
  - ECN applicable to modern input-queued switches
  - Source response: rate control w/ window limit
- Derived new relaxed conditions for source response function convergence à functions with fast bandwidth reclamation
  - Based on observation of packet marking bias
  - Two examples: FIMD/LIPD outperform AIMD
- Submitted our proposal to the InfiniBand Trade Organization congestion control working group

For Additional Information

- "End-to-end congestion control for InfiniBand", IEEE INFOCOM 2003.
- "Evaluation of congestion detection mechanisms for InfiniBand switches", IEEE GLOBECOM 2002.
- "An approach for congestion control in InfiniBand", HPL-2001-277R1, May 2002.

i n v e n t