Towards Experimental Evaluation of Explicit Congestion Control

Presented by A. L. Narasimha Reddy

Saurabh Jain, Yueping Zhang, and Dmitri Loguinov

Texas A&M University
College Station, TX 77843

June 3rd, 2008

Agenda

- Introduction
- Implementation
- Experimental results
- Conclusions

Explicit Congestion Control

- Explicit congestion control utilizes multi-byte congestion notification from network devices to adjust sending rates of end-users
- Analysis and simulations indicate that explicit congestion control achieves scalability in terms of bandwidth and user population, stability under delay, and inter-flow fairness
- Examples include XCP, MaxNet, EMKC, RCP, JetMax, and PIQI-RCP

Existing Explicit Congestion Controls

- We study four explicit congestion control methods
- XCP (eXplicit Control Protocol)
  - Window-based
  - Each router implements a fairness and efficiency controller
- JetMax
  - Achieves provable delay-independent stability, capacity-independent scalability, max-min fairness, and zero loss
- RCP (Rate Control Protocol)
  - Fast transient dynamics and max-min fairness
- PIQI-RCP (Proportional Integral Queue Indep. RCP)
  - Better transient dynamics than RCP and delay-independent

Existing Experimental Studies

- As to the best of our knowledge, existing experimental studies of explicit congestion control are limited to the following three papers
  - Zhang *et al.*: XCP over a 10-mb/s bottleneck link
  - Suchara *et al.*: MaxNet over a 10-mb/s bottleneck link
  - Zhang *et al.*: JetMax in 3 scenarios with multiple links
- Here, we seek to advance research in this area by
  - Implementing a unified Linux framework of explicit congestion control for both window- and rate-based methods
  - Conducting simple, yet informative, comparison of existing explicit congestion control and identify their key problems

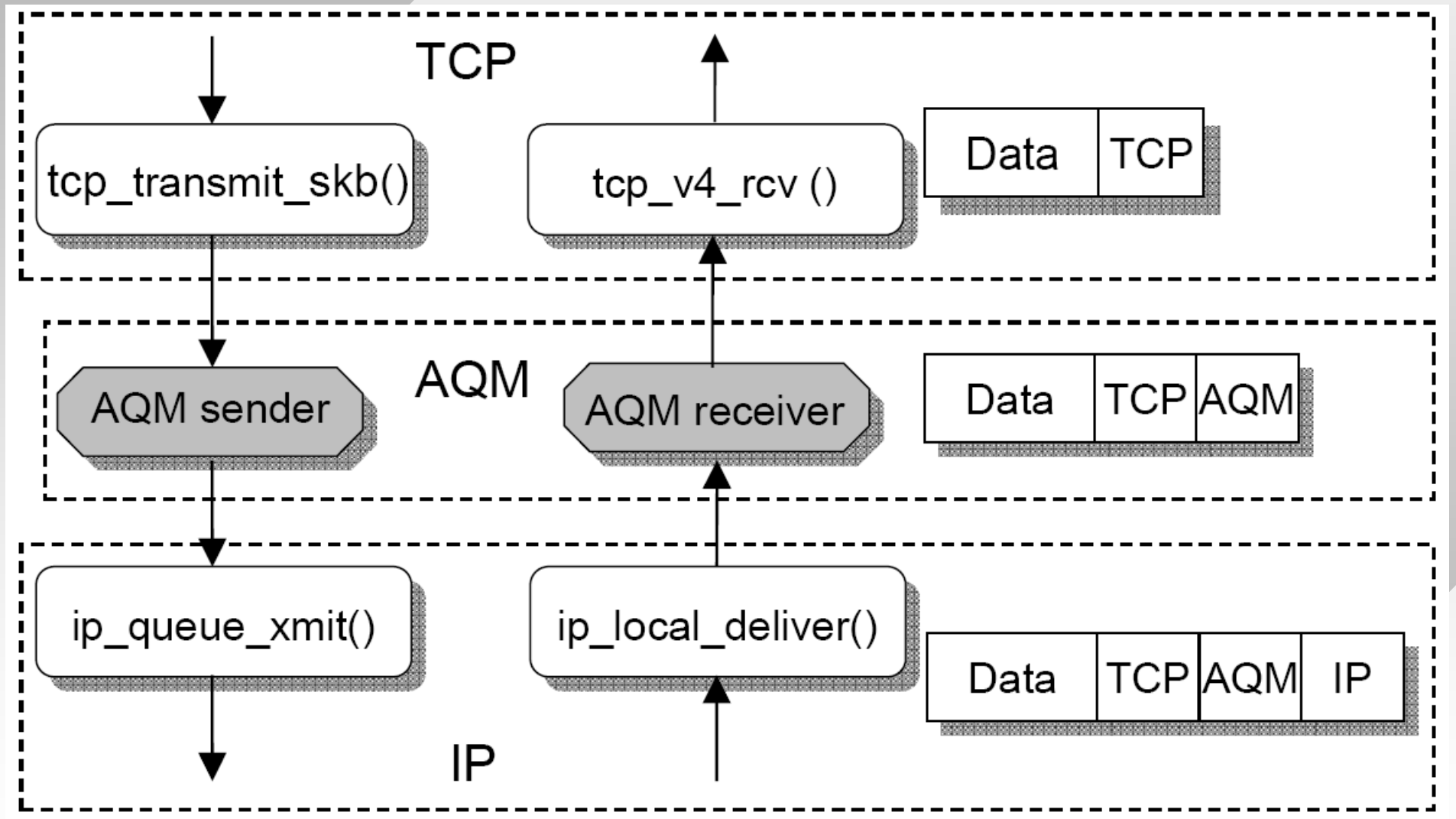
Agenda

- Introduction
- Implementation
- Experimental results
- Conclusions

End-User I

- We place between each packet's transport and network headers a new congestion (AQM) layer, which is used to communicate congestion-related information and network feedback between end-hosts and routers
- Congestion headers can be conveniently inserted into (or removed from) every outgoing (or incoming) packet by the end-host module without interfering with the operation in the TCP/IP layer
- Thus, this methodology is versatile enough to be easily implemented in different operating systems without modifying their TCP/IP stacks

End-User II


TCP
tcp_transmit_skb()
tcp_v4_rcv ()
Data
TCP
AQM
AQM sender
AQM receiver
Data
TCP
AQM
ip_queue_xmit()
ip_local_deliver()
Data
TCP
AQM
IP
IP

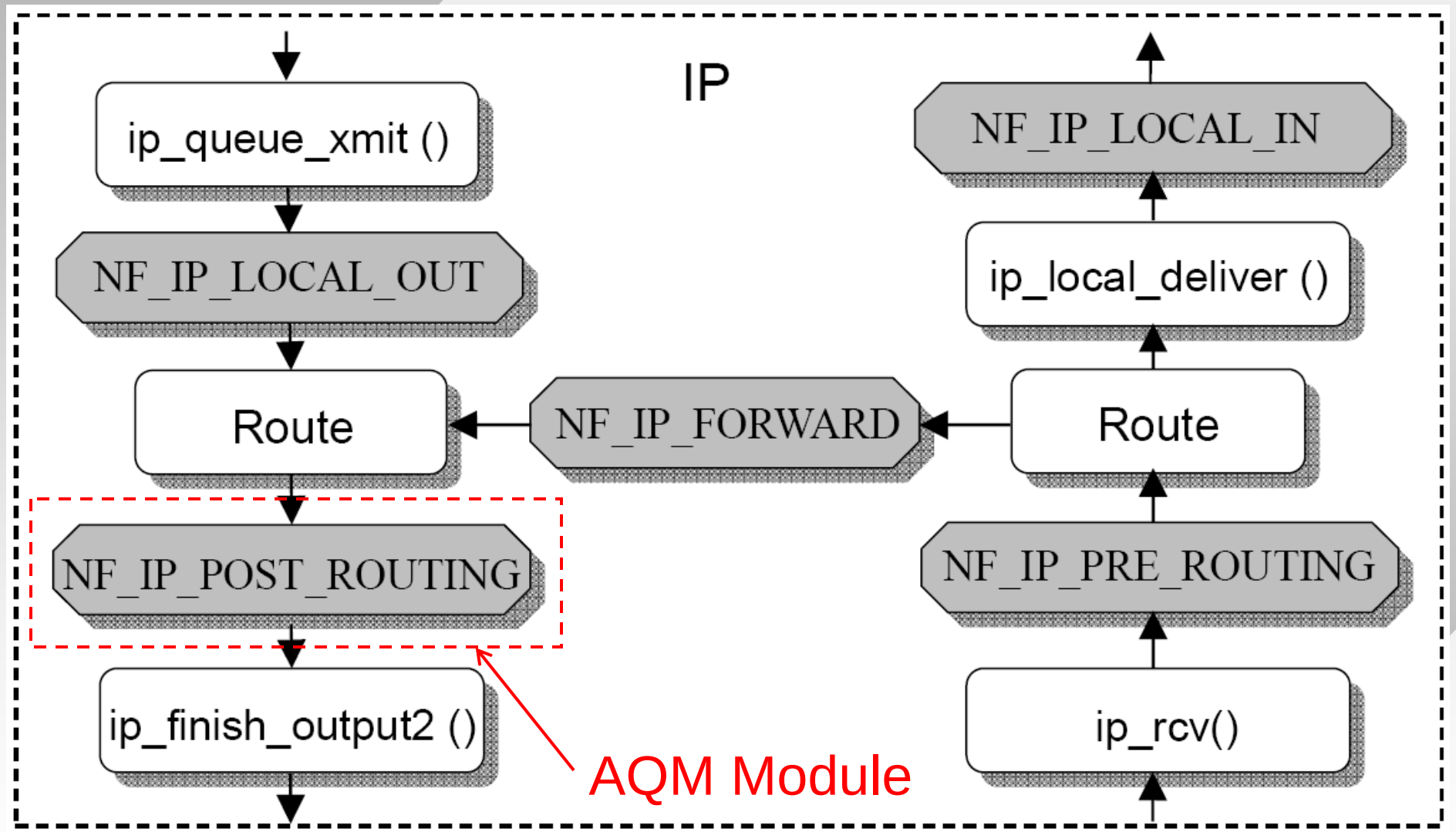
End-User III

- In addition, we successfully implement rate-based methods (i.e., JetMax, RCP, PIQI-RCP) in the existing window-based TCP/IP stack
- The idea is simple – to pace data transmission and maintain a smooth rate calculated by the controller
  - Disable bursty packet transmission behavior of TCP
  - Implement a control timer to periodically process data held by the queue
- This way, without changing the underlying network stack, all the methods (both window- and rate-based) can be conveniently and fairly tested in the same system by simply loading/unloading their corresponding kernel driver module

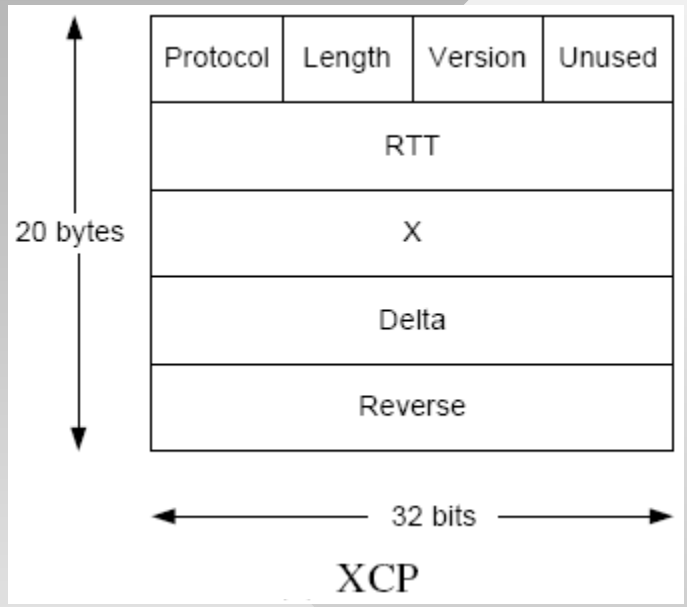
Router I

- Routers in explicit congestion control algorithms need to provide end-hosts with accurate feedback of the actual congestion level
- This information is injected into every packet's congestion header and further fed back to the sender
- In this paper, we take advantage of `netfilter` and develop AQM modules of RCP, PIQI-RCP, XCP, and JetMax in the hook function `NF_IP_POST_ROUTING`, at which point all outgoing packets from the local machine or incoming packets from other network interfaces that need to be forwarded are processed

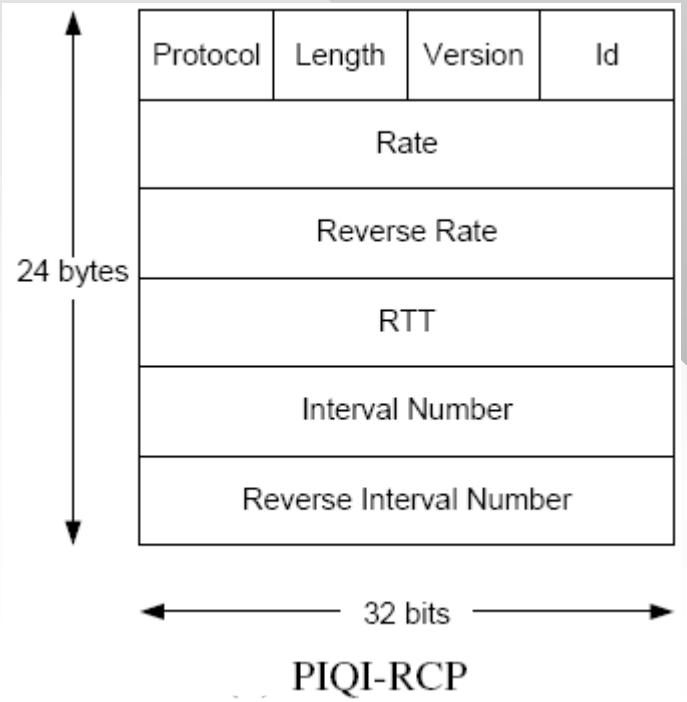
Router II


IP
ip_queue_xmit ()
NF_IP_LOCAL_OUT
Route
NF_IP_POST_ROUTING
ip_finish_output2 ()
NF_IP_FORWARD
NF_IP_LOCAL_IN
ip_local_deliver ()
Route
NF_IP_PRE_ROUTING
ip_rcv()
AQM Module

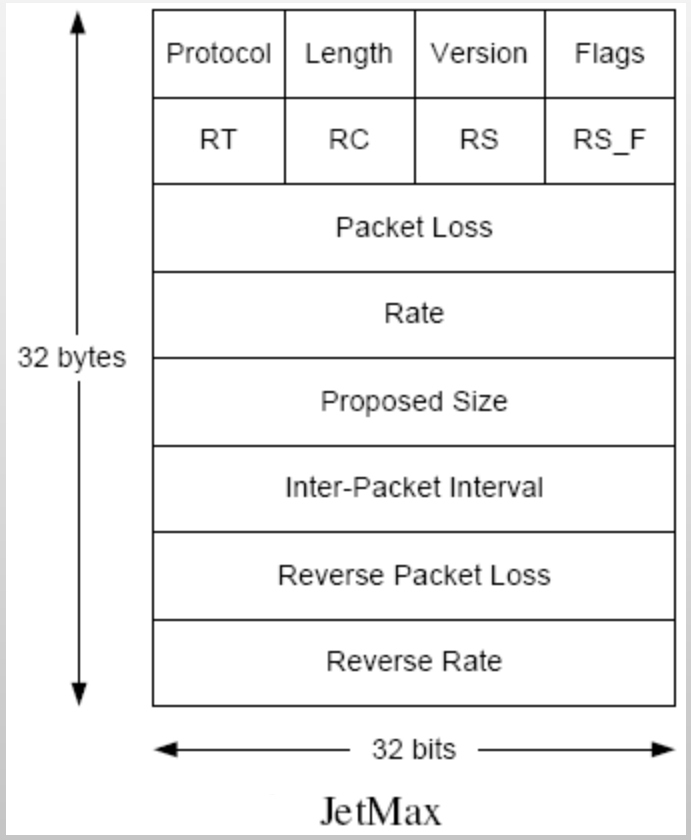
Congestion Header Format



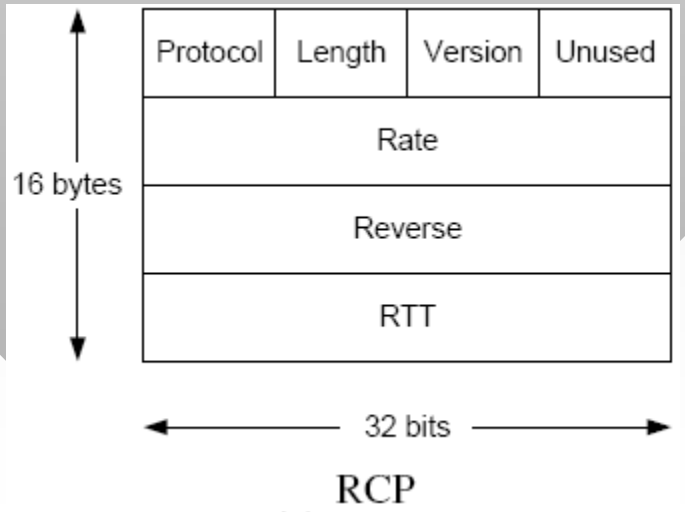
XCP



PIQI-RCP



JetMax



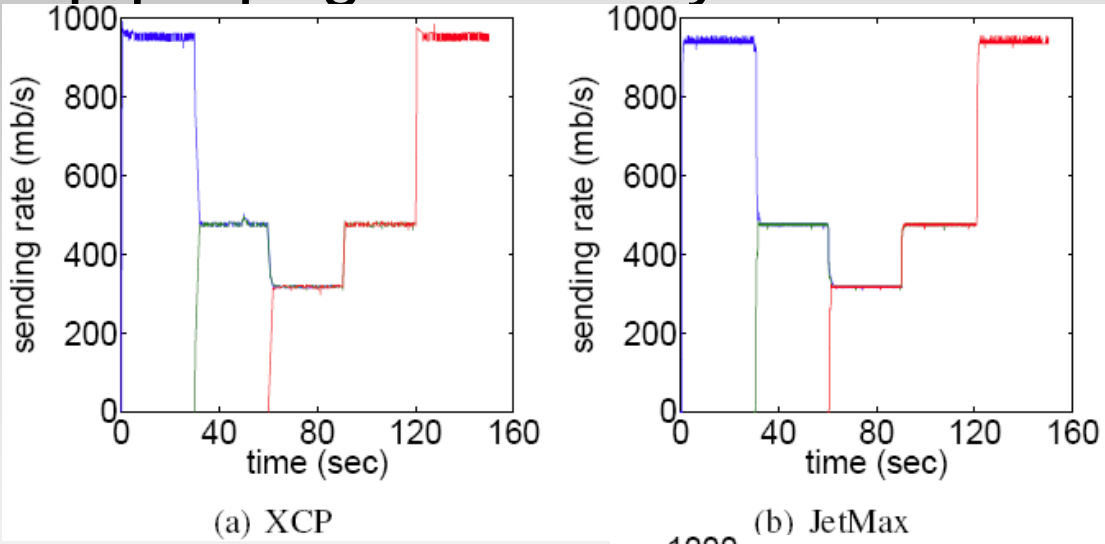
RCP

Agenda

- Introduction
- Implementation
- Experimental results
- Conclusions

Single-Bottleneck Topology

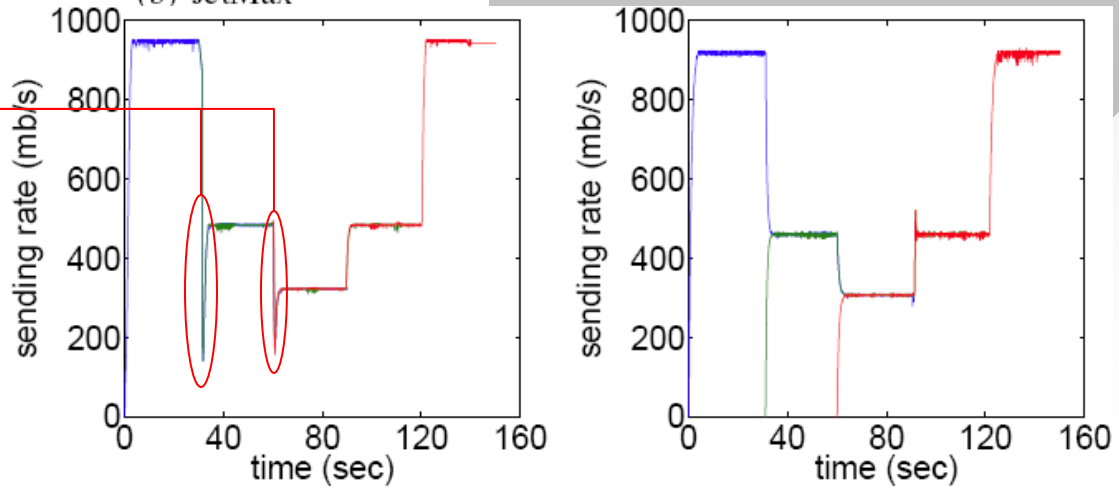
- A dumb-bell topology where three flows pass through a single bottleneck link of capacity 1 gb/s and round-trip propagation delay 50 ms



(a) XCP (b) JetMax

All methods are able to maintain high sending rates and do not experience any packet loss

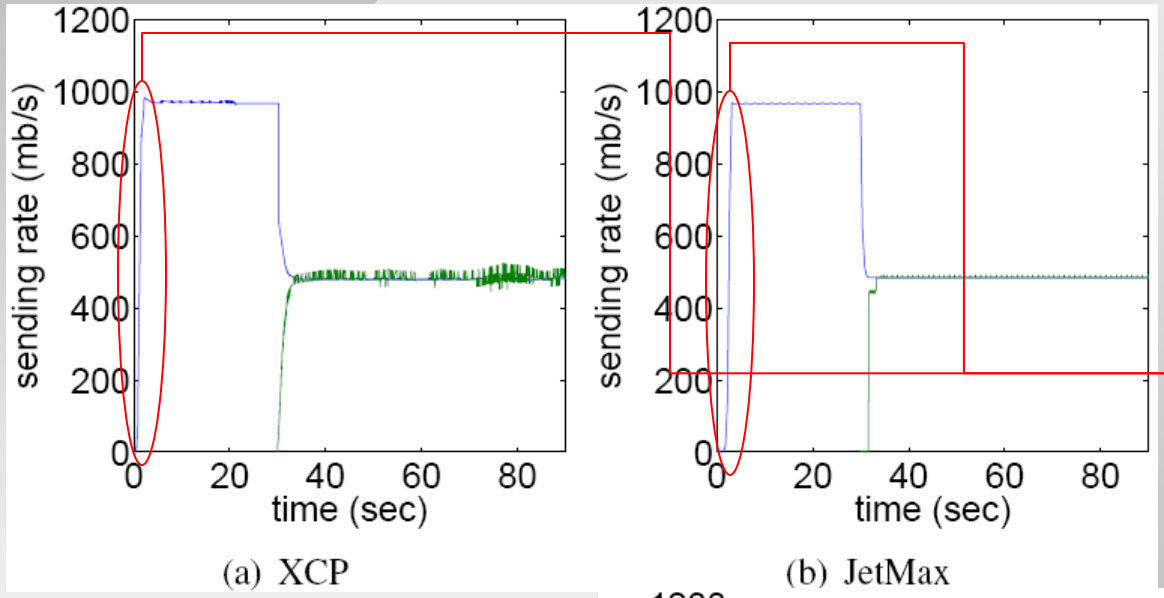
RCP experiences significant queue buildup (up to 9415 packets) whenever a new flow joins the system



(c) RCP (d) PIQI-RCP

RTT Unfairness

- A dumb-bell topology where a 1-gb/s link shared by two flows with RTT 30 and 200 ms, respectively

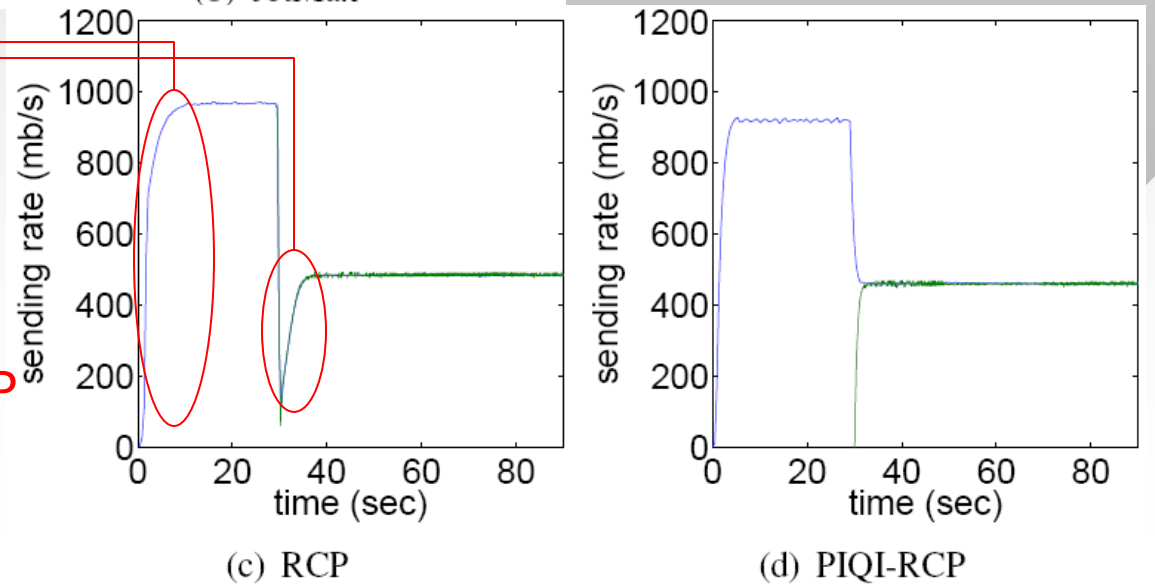


All methods achieve RTT fairness in the steady state

XCP and JetMax achieve full link utilization almost instantaneously

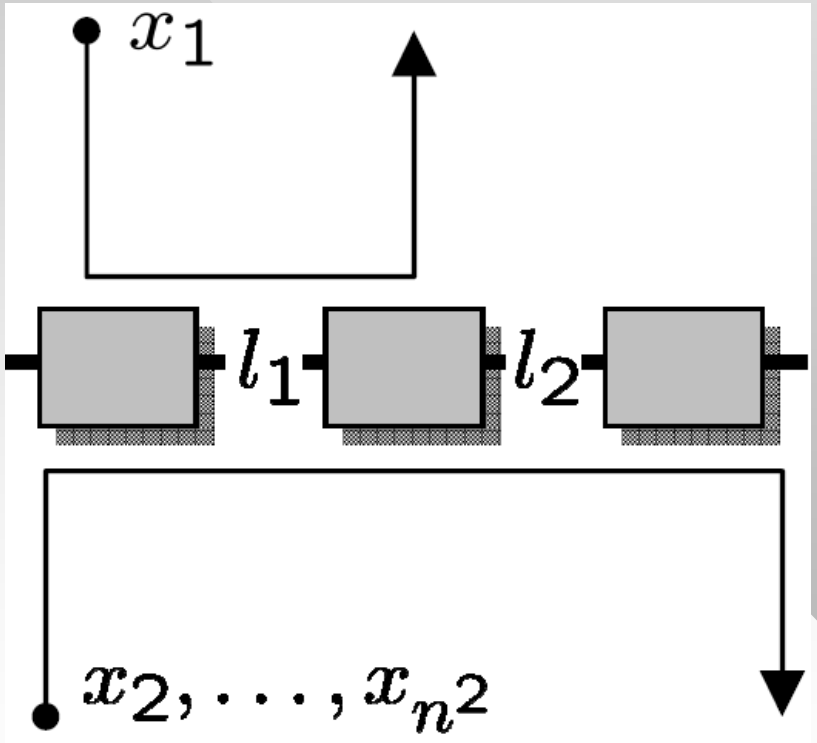
To saturate the link, it takes RCP nearly 10 sec initially and 5 sec when a flow joins

Compared to RCP, PIQI-RCP has better performance in transient phase

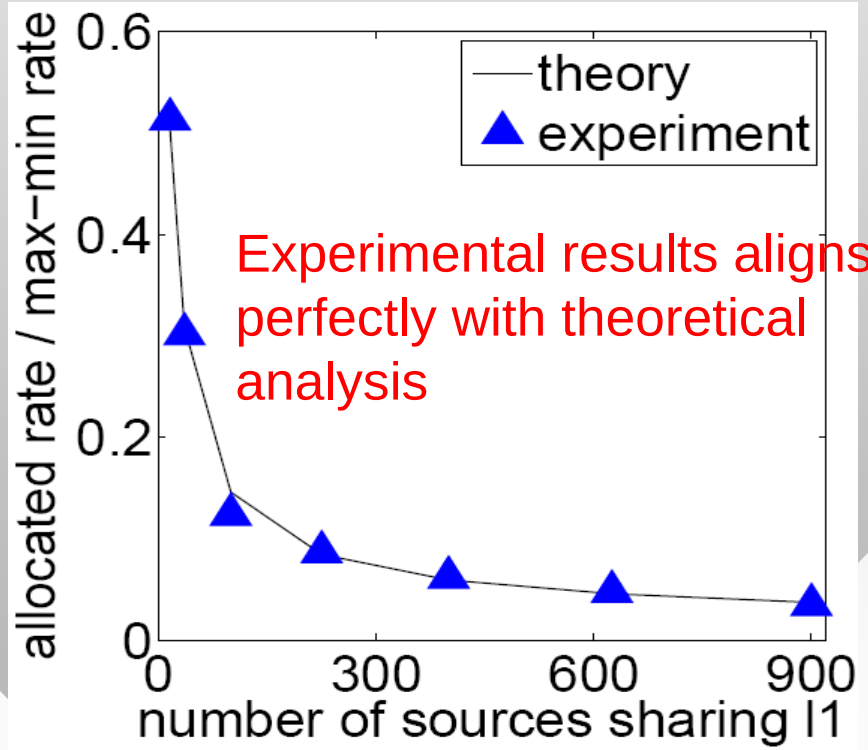


Max-Min Fairness in XCP

- We next verify the unfairness problem identified by Steven Low by considering the following topology
  - n^2 flows and two links l_1 and l_2, whose capacity C_1 and C_2 are respectively $C_1 = 155$ and $C_2 = C_1 \,/\, [n\,(n+1)]$ mb/s



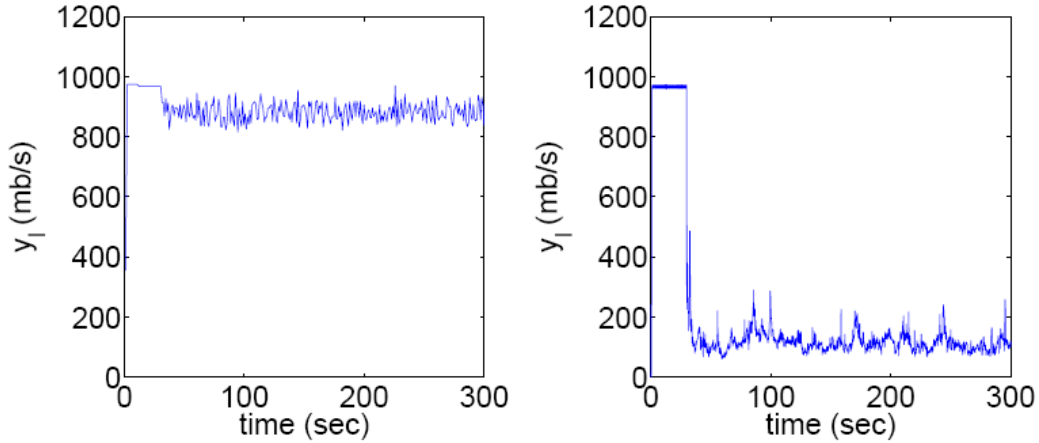
(a) topology



(b) result

Performance with Mice Traffic

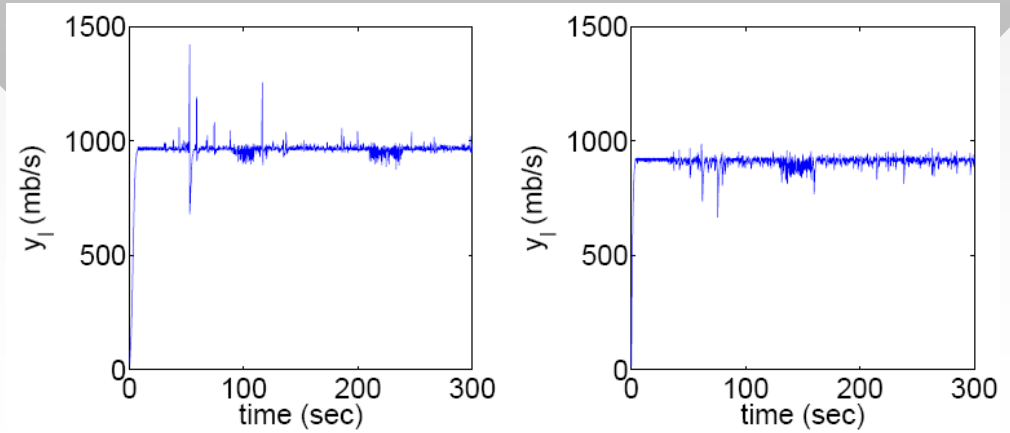
- The pattern of mice traffic follows Poisson arrivals with mean inter-arrival time of 0.2 sec and Pareto distributed traffic size with shape parameter 1.4 and mean of 100 packets



(a) XCP (b) JetMax

XCP, RCP, and PIQI-RCP all perform reasonably well

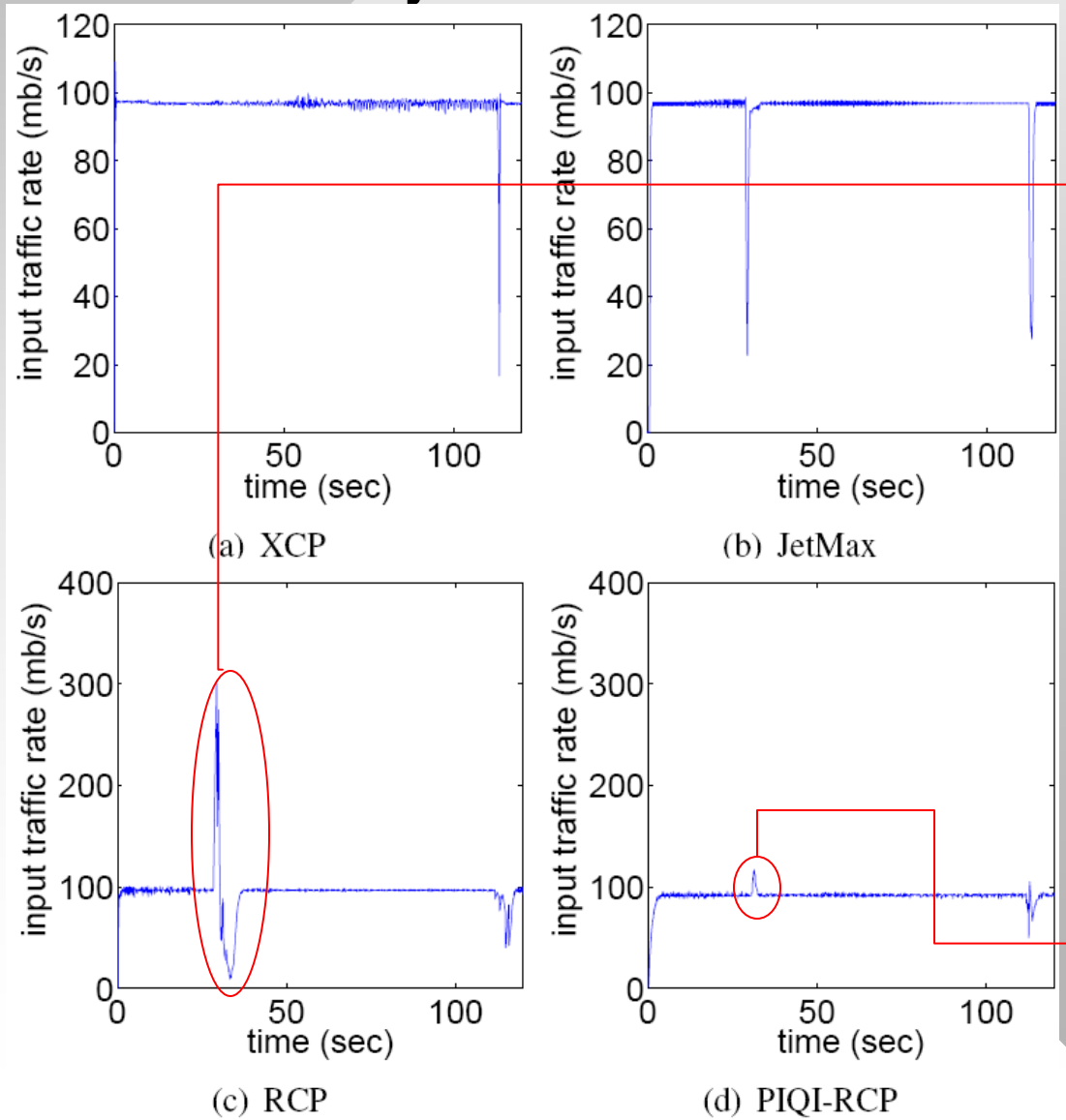
JetMax suffers low utilization since it allocates an even rate to both long and short flows, the latter of which may not be able to fully utilize the allocated bandwidth during their lifetimes



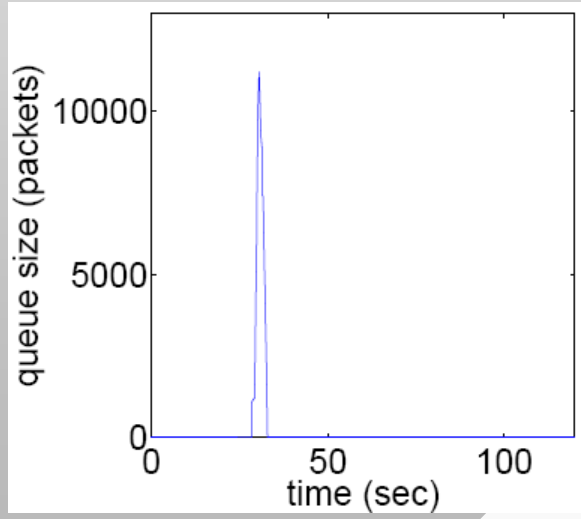
(c) RCP (d) PIQI-RCP

Abrupt Change in Traffic Demand

- One flow starts at $t=0$ and lasts for 120 secs and another 10 join at $t=30$ sec and exist at $t=113$ sec



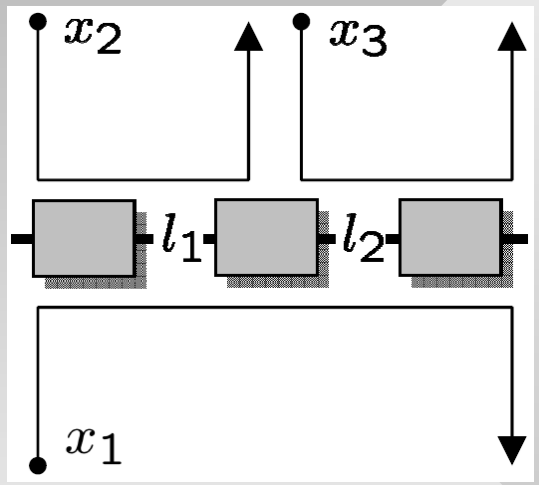
When 10 flows join, RCP overshoots the link capacity by 300% and its queue size jumps to 11000 packets

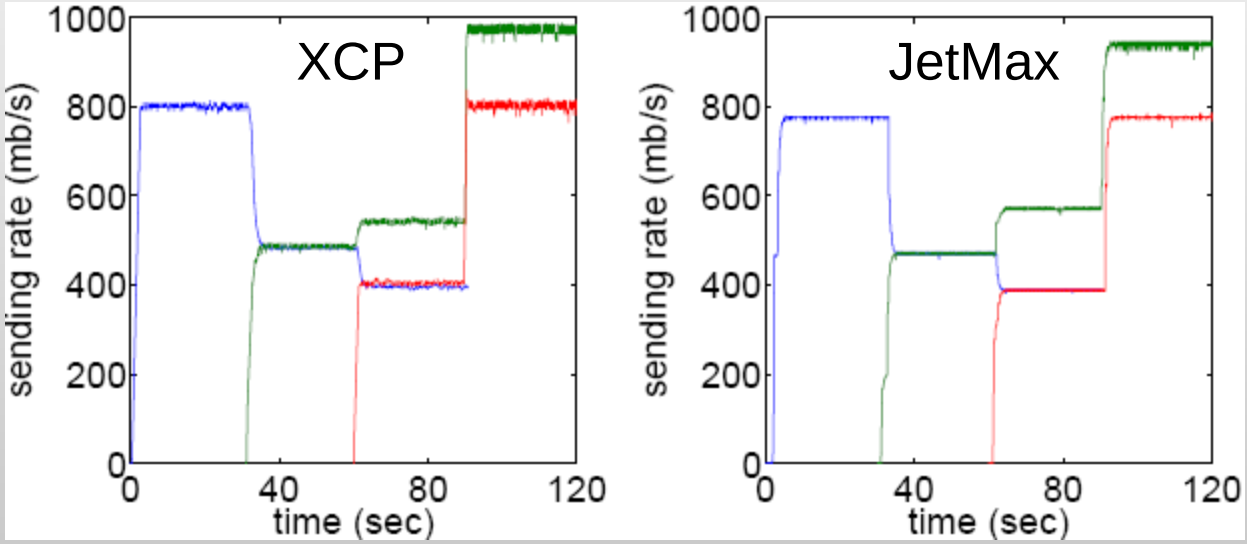


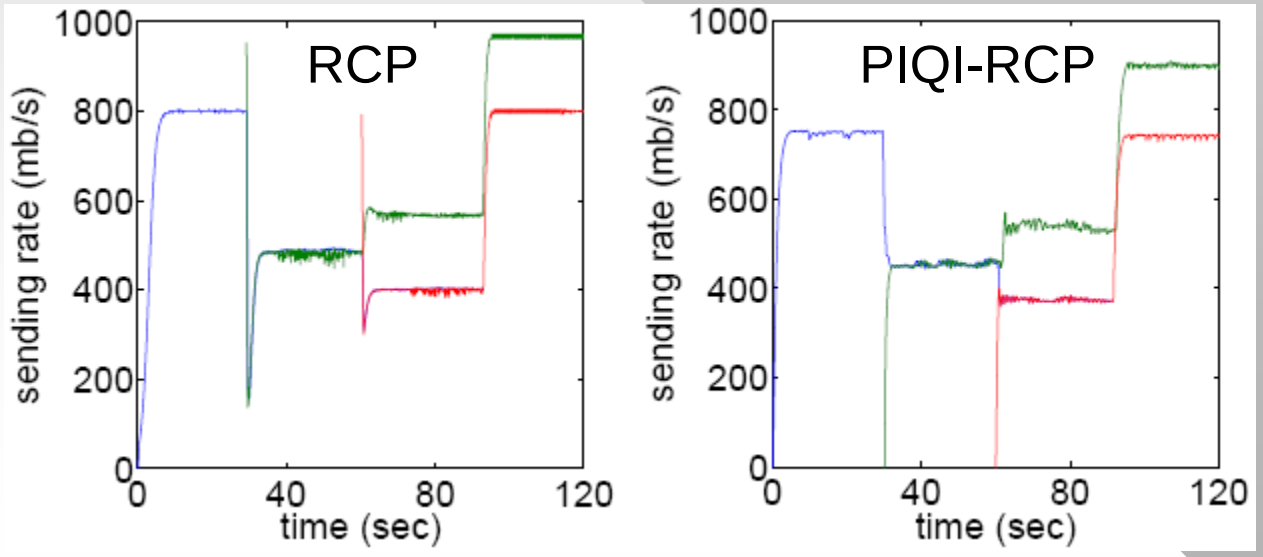
Compared to RCP, PIQI RCP performs much better

Multi-Link Topology

- $C_1 = 970$ and $C_2 = 800$ mb/s







All methods are stable and max-min fair in this case with their dynamics following the theoretical understanding

Summary of Experimental Results I

- All methods are scalable with increase in link capacity and round-trip delay, and admit low queuing delay and almost zero packet loss in the steady state
- XCP can maintain high link utilization and has low buffering requirement, but it does not achieve max-min fairness in certain topologies and has the largest number of per-packet computations (6 additions and 3 multiplications)
- JetMax has the least buffering requirement and the smallest number of per-packet overhead (3 additions), but loses link utilization in the presence of mice traffic

Summary of Experimental Results II

- RCP has a reasonable amount of per-packet calculations (2 additions and 2 multiplications), but has a very high buffering requirement to avoid packet loss and exhibits significant overshoot under abrupt traffic changes
- Compared to RCP, PIQI-RCP retains most of the strengths of RCP, has significantly lower buffering needs, and is more robust to abrupt surge in traffic demand

Conclusions

- In this work, we implemented a unified platform for developing both window- and rate-based explicit congestion control and evaluated several existing methods in gigabit networks
- Using our implementation, we, besides verifying previously reported results of these methods based on ns2 simulations, also discover several new limitations of RCP (i.e., high buffering requirement and significant transient overshoot) and JetMax (i.e, poor utilization under mice traffic)
- Our results lead us to conclude that PIQI-RCP performs the best among the studied methods given scenarios considered in this work