RCP: Congestion Control to Make Flows Complete Quickly

Nandita Dukkipati

High Performance Networking Group
Computer Systems Laboratory
Stanford University

Ph.D. Oral Seminar
5 October, 2006

Talk Outline

- Congestion Control Wish List
- What high-speed TCPs and XCP achieve
- Rate Control Protocol (RCP)
  - Goals
  - Short Flow Completion Times
  - Stability
  - RCP under worst-case traffic patterns
  - RCP-AC: Rate Control Protocol with Acceleration Control
- Conclusion

TCP does not work well

1. Slow additive increase means flows take a long time to acquire spare capacity

2. Unsustainable large equilibrium window; requires extremely small loss $p = 3/(2w^2)$

3. Confused by lossy links -- low throughput in wireless links

4. Unfair bandwidth sharing: Flow throughput $\propto \frac{1}{RTT}$

5. Inefficient Slow Start
   - Flows made to last multiple round trip times
   - Instability -- exponential increase in aggregate traffic

6. Large queueing delay

Wish List

1. Emulate Processor Sharing
   - Performance is invariant of flow size distribution
   - Mix of flows: Short flow-completion Times -- close to the minimum achievable
   - Long flows: Results in 100% link utilization -- even under high bandwidth-delay, lossy links...
   - All flows get fair share of bottleneck bandwidth
2. Want stability -- convergence to equilibrium operating point
3. Close to zero buffer occupancy, Loss-free network
4. Makes efficient use of high bandwidth-delay links, e.g. fiber links
5. Allows proportional bandwidth sharing among flows
6. Want all the above under any network conditions (mix of RTTs, capacities, topologies, lossy and high-delay [wireless] links).
7. Any traffic mix: short flows, mix of flow-sizes, sudden flash-crowds
8. Can police flows to enforce congestion control
9. No per-flow state, per-flow queue
10. No per-packet computation in routers

High-speed TCPs - Pros and Cons

1. Emulate Processor Sharing
   - Performance is invariant of flow size distribution
   - Mix of flows: Results in flows finishing quickly -- close to the minimum achievable
   - Long flows: Results in 100% link utilization -- even under high bandwidth-delay, lossy links...
   - All flows get fair share of bottleneck bandwidth
2. Want stability -- convergence to equilibrium operating point
3. Close to zero buffer occupancy, Loss-free network
4. Makes efficient use of high bandwidth-delay links, e.g. fiber links
5. Allows proportional bandwidth sharing among flows
6. Want all the above under any network conditions (mix of RTTs, capacities, topologies, lossy and high-delay [wireless] links).
7. Any Traffic mix: long flows, mix of flow-sizes, sudden flash-crowds
8. Can police flows to enforce congestion control
9. No per-flow state, per-flow queue
10. No per-packet computation in routers

Explicit Control Protocol (XCP)

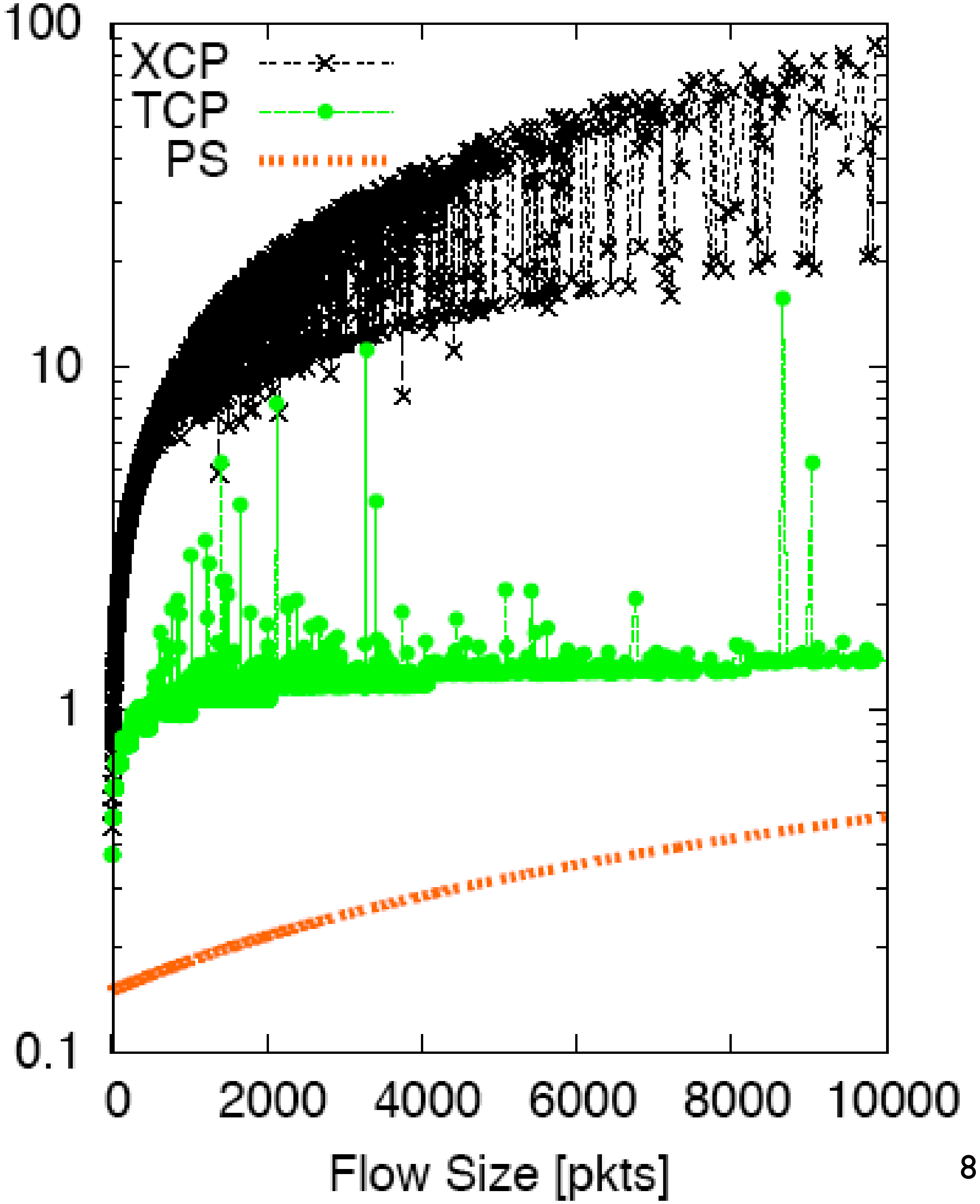
- Proposed by Katabi et. al Sigcomm 2002; part of NewArch project
- Explicit feedback on congestion from the network
- Flows receive precise feedback on window increment/decrement
- Convergence of fair share rates could take many round trip times
- Routers do detailed per-packet calculations

XCP - Pros and Cons

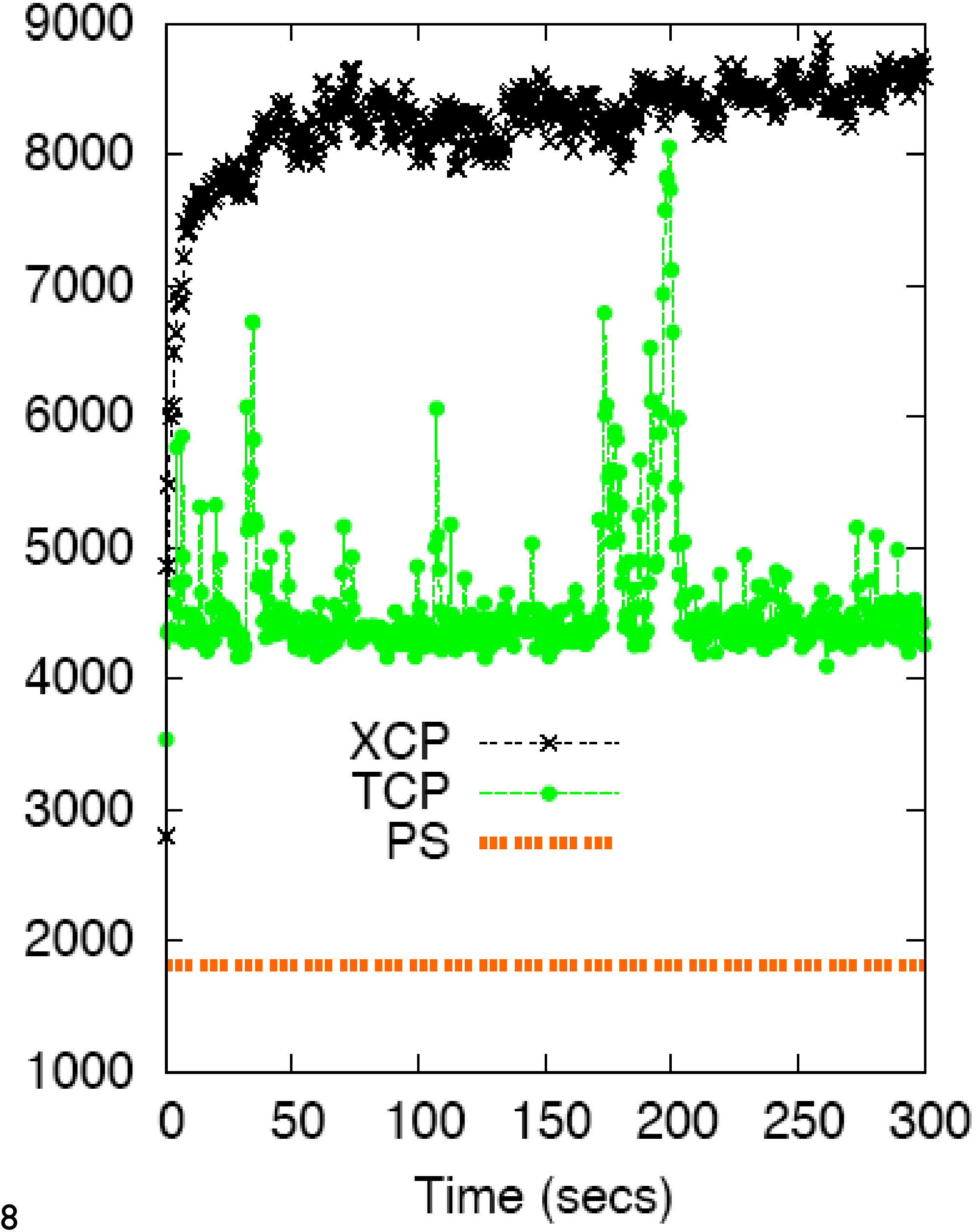
1. Emulate Processor Sharing
   - Performance is invariant of flow size distribution
   - Mix of flows: Results in flows finishing quickly -- close to the minimum achievable
   - Long flows: Results in 100% link utilization -- even under high bandwidth-delay, lossy links...
   - All flows get fair share of bottleneck bandwidth
2. Want stability -- convergence to equilibrium operating point
3. Close to zero buffer occupancy, Loss-free network
4. Makes efficient use of high bandwidth-delay links, e.g. fiber links
5. Allows proportional bandwidth sharing among flows
6. Want all the above under any network conditions (mix of RTTs, capacities, topologies, lossy and high-delay [wireless] links).
7. Any Traffic mix: long flows, mix of flow-sizes, sudden flash-crowds
8. Can police flows to enforce congestion control
9. No per-flow state, per-flow queue
10. No per-packet computation in the routers

Example: XCP vs. TCP vs. PS

Flow Duration (secs) vs. Flow Size



Active Flows vs. time



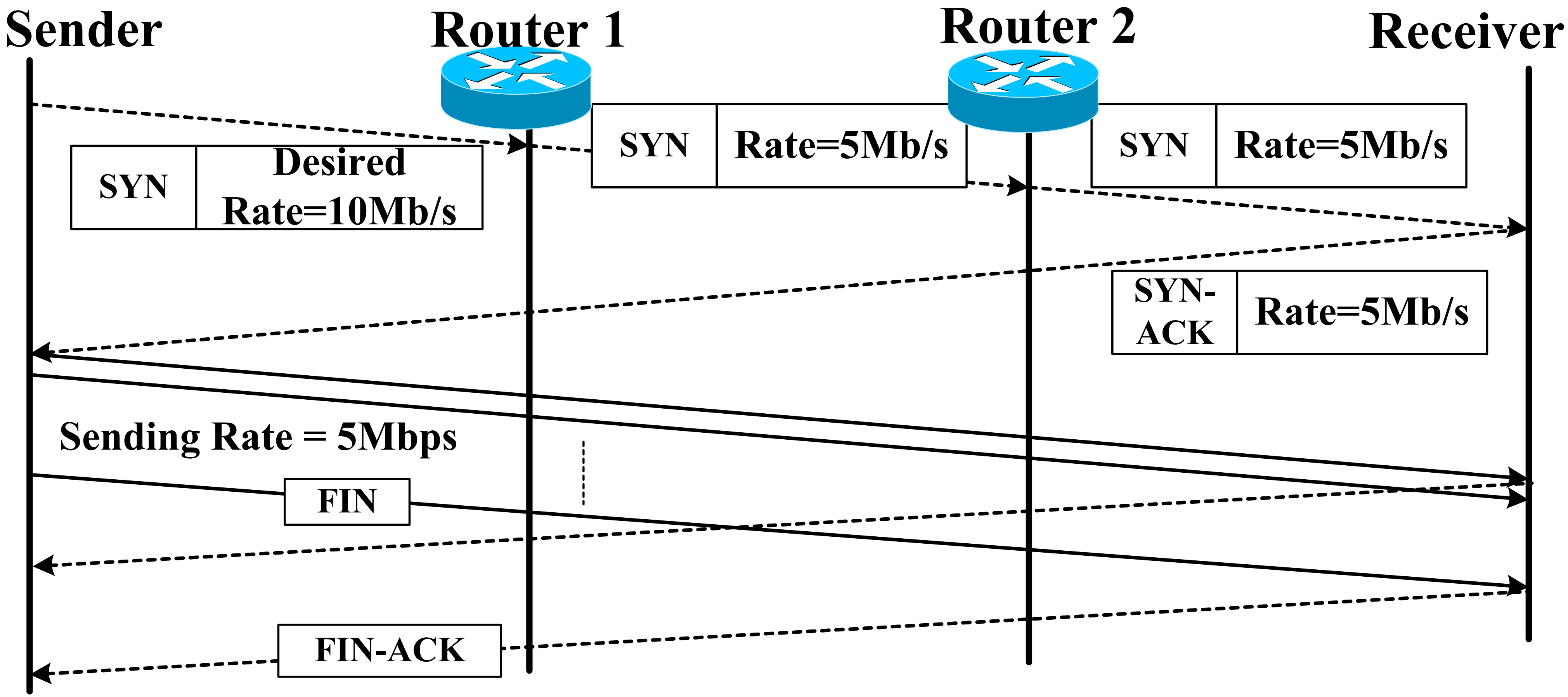
RCP - Pros and Cons

1. Emulate Processor Sharing
   - Performance is invariant of flow size distribution
   - Mix of flows: Results in flows finishing quickly -- close to the minimum achievable
   - Long flows: Results in 100% link utilization -- even under high bandwidth-delay, lossy links...
   - All flows get fair share of bottleneck bandwidth
2. Want stability -- convergence to equilibrium operating point
3. Close to zero buffer occupancy, Loss-free network
4. Makes efficient use of high bandwidth-delay links, e.g. fiber links
5. Allows proportional bandwidth sharing among flows
6. Want all the above under any network conditions (mix of RTTs, capacities, topologies, lossy and high-delay [wireless] links).
7. Any Traffic mix: long flows, mix of flow-sizes, sudden flash-crowds
8. Can police flows to enforce congestion control
9. No per-flow state, per-flow queue
10. No per-packet computation in the routers

Rate Control Protocol (RCP): Picking the Flow Rate

- Is there one rate a router can give out to all the flows so as to emulate Processor Sharing ?
- Rate $R(t) = C/N(t)$
- RCP is an adaptive algorithm to emulate PS:
  - $R(t)$ picked by the routers based on queue size and aggregate traffic
  - Router assigns a single rate to all flows
  - Requires no per-flow state or per-flow queue

RCP: The Basic Mechanism



RCP: The Algorithm

Link Capacity

Aggregate Traffic

queue

Average RTT

$$R(t) = R(t - d_0) + \frac{\alpha(C - y(t)) - \beta \frac{q(t)}{d_0}}{\widehat{N(t)}}$$

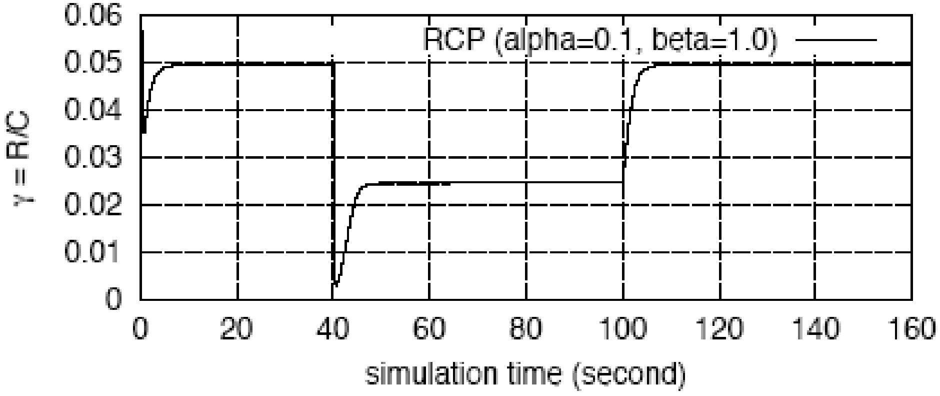
Estimate of # flows

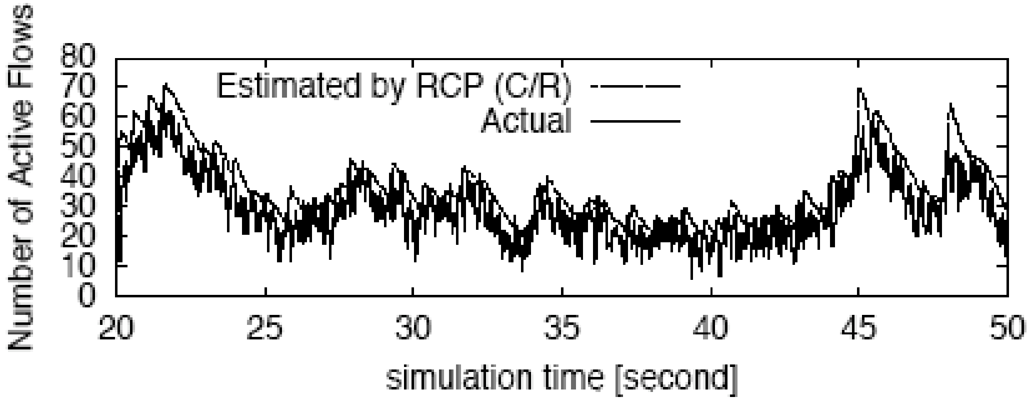
$$\widehat{N(t)} = \frac{C}{R(t - d_0)}$$

$$R(t) = R(t - T)[1 + \frac{\frac{T}{d_0}(\alpha(\gamma C - y(t)) - \beta \frac{(q(t) - q_0)}{d_0})}{\gamma C}]$$

Understanding RCP: C/R

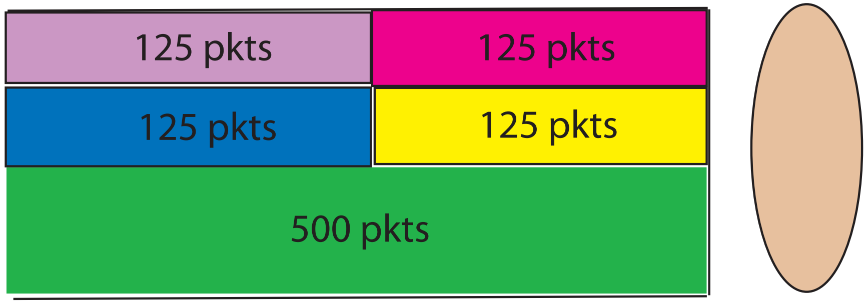
Many large flows

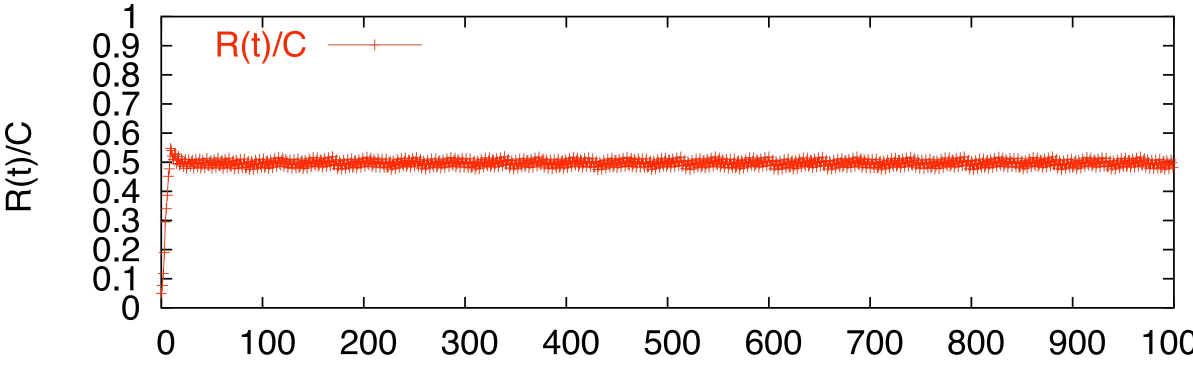




Many small flows: $$R = \frac{C.RTT - \sum \text{traffic in short flows}}{n_{\text{large flows}} \cdot RTT}$$

C. RTT = 1000 pkts





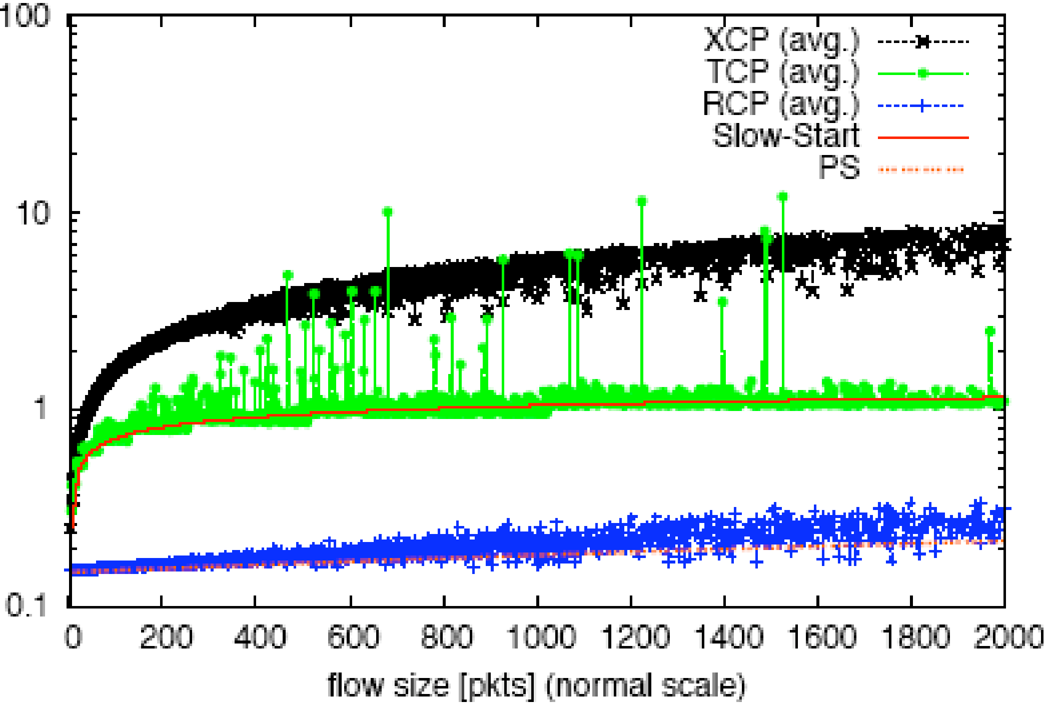
Understanding RCP

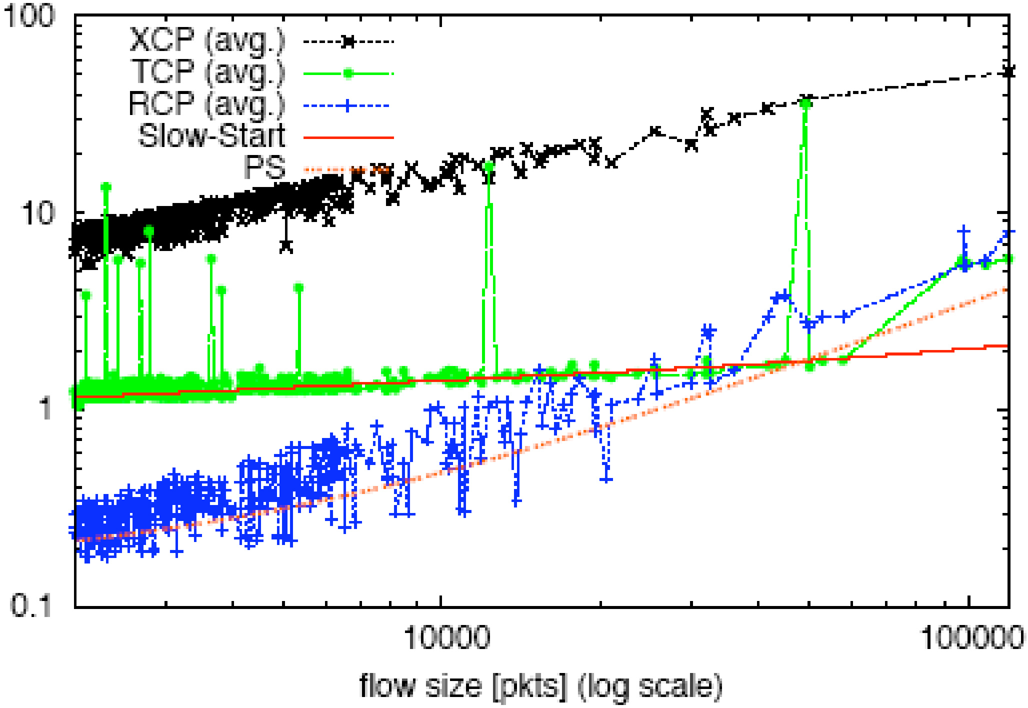
- Short Flow Completion Times
- Stability
  - Homogeneous delays
  - Heterogeneous delays
- Choosing alpha, beta for good performance
- Proportional bandwidth sharing
- Buffer-sizing under RCP
- Implementing RCP: Routers and End-hosts
- RCP's Achilles Heel
- RCP-AC: Rate Control Protocol with Acceleration Control

RCP Performance

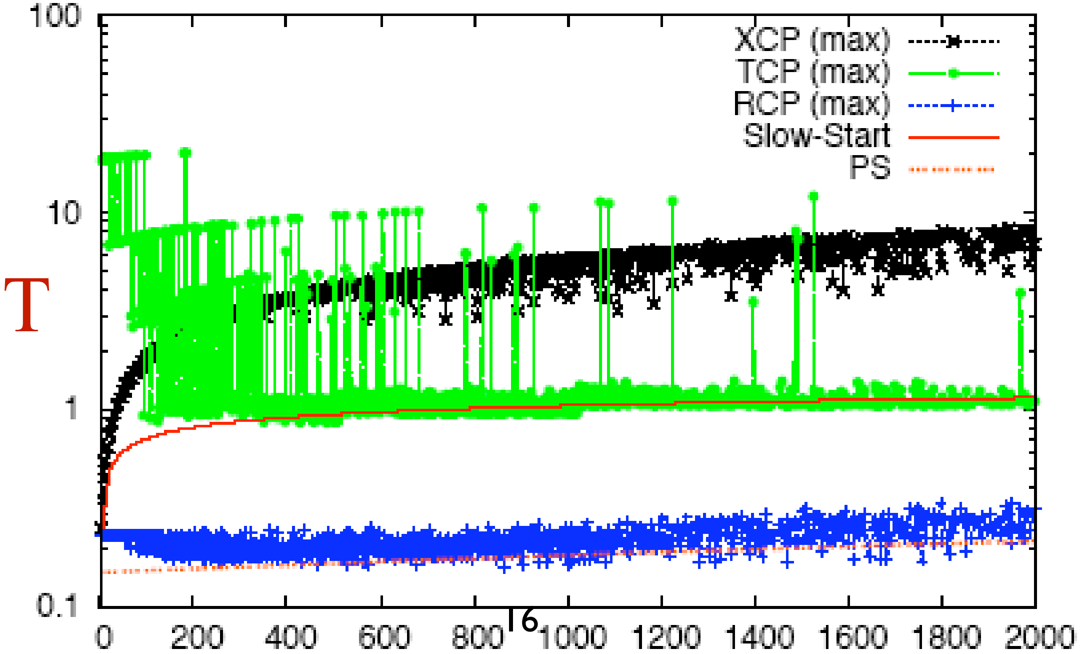
- When traffic characteristics vary
  - Different flow sizes
  - As mean flow size increases
  - Different flow size distributions
  - Non Poisson arrivals of flows
  - As load increases
- When Network Conditions vary
  - As link capacity increases
  - As RTT increases
  - Flows with different RTTs
  - Multiple bottlenecks
  - Congested reverse links
- Compared with: $AFCT \geq 1.5RTT + \frac{E[L]}{C}; FCT_{PS} = 1.5RTT + \frac{L}{C(1-\rho)}$

Example 1: Achieves PS for different Flow Sizes

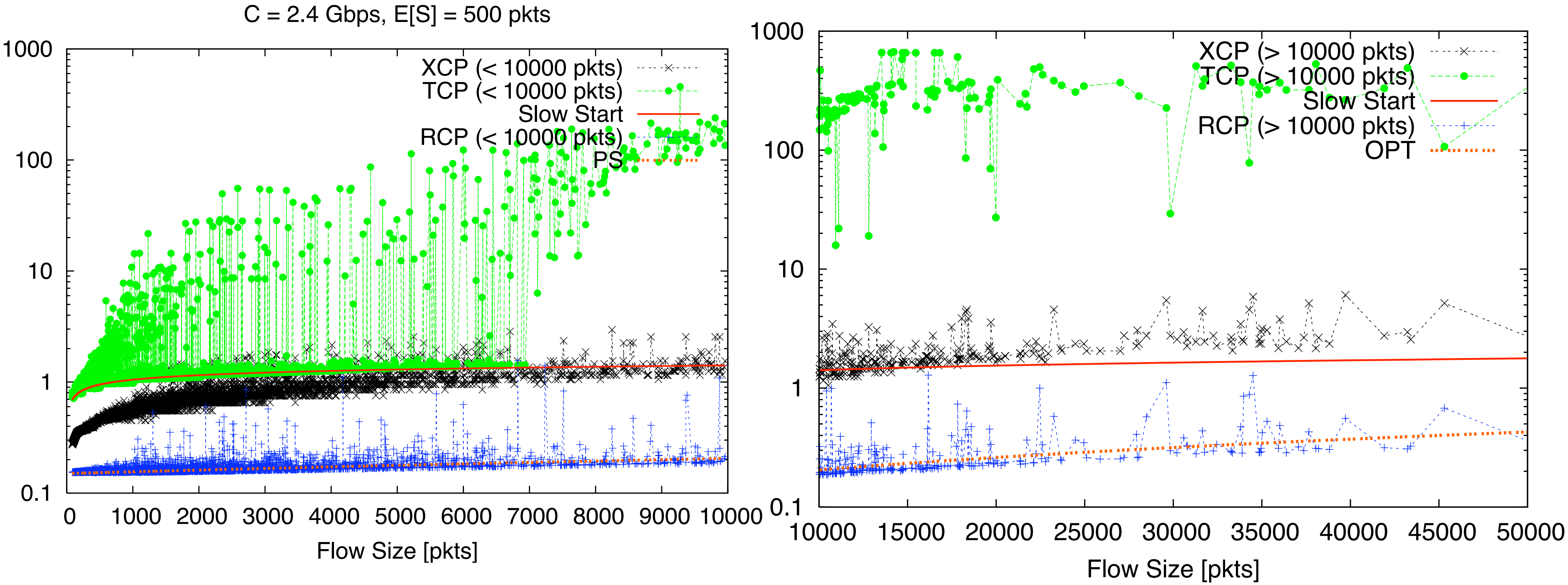




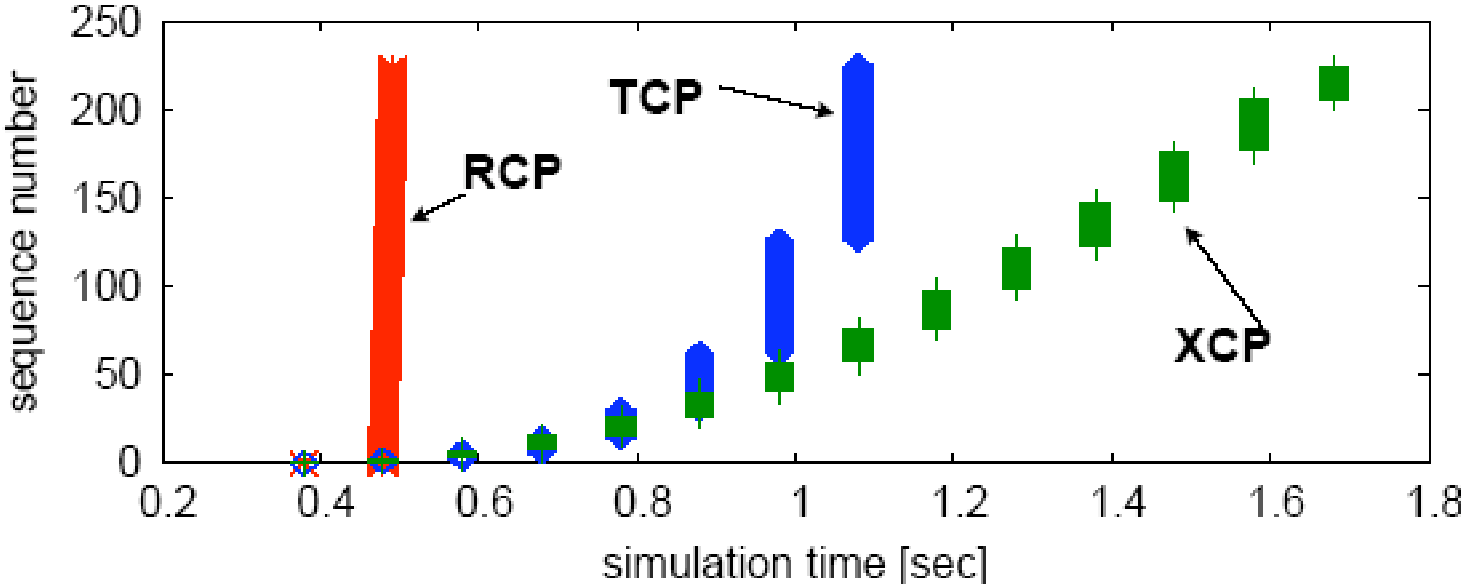
Max. FCT

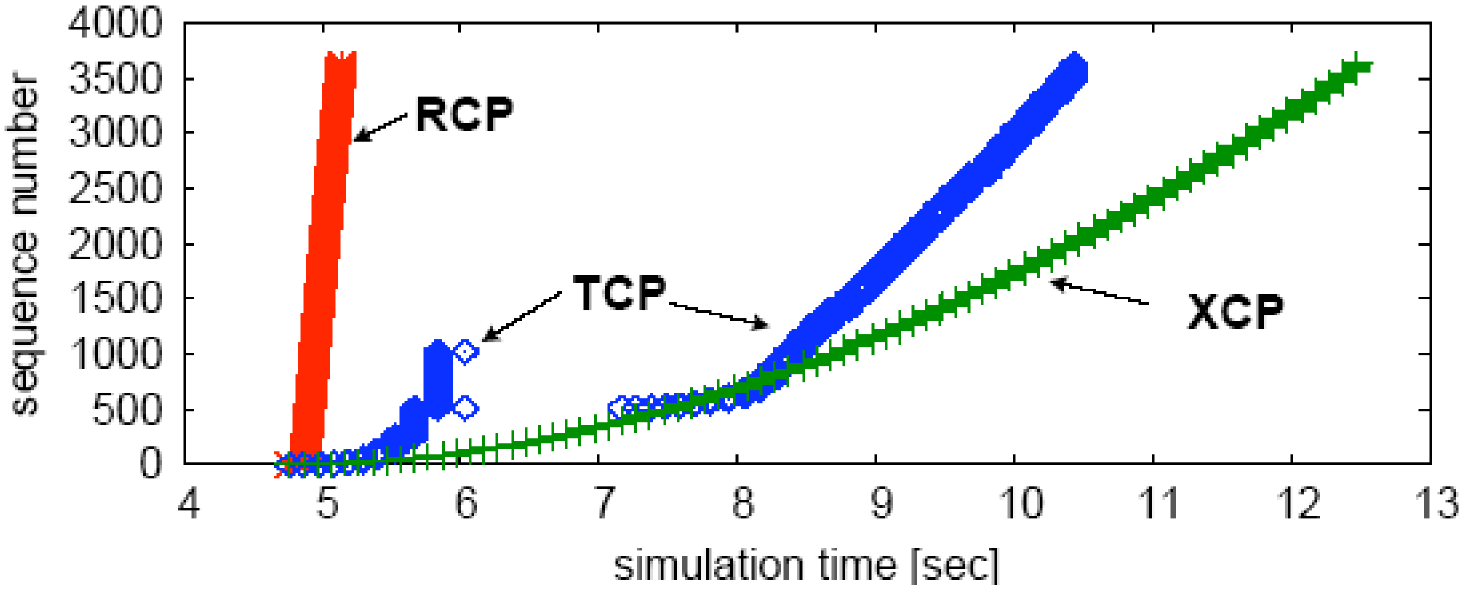


Example 2: Achieves PS for different Flow Sizes

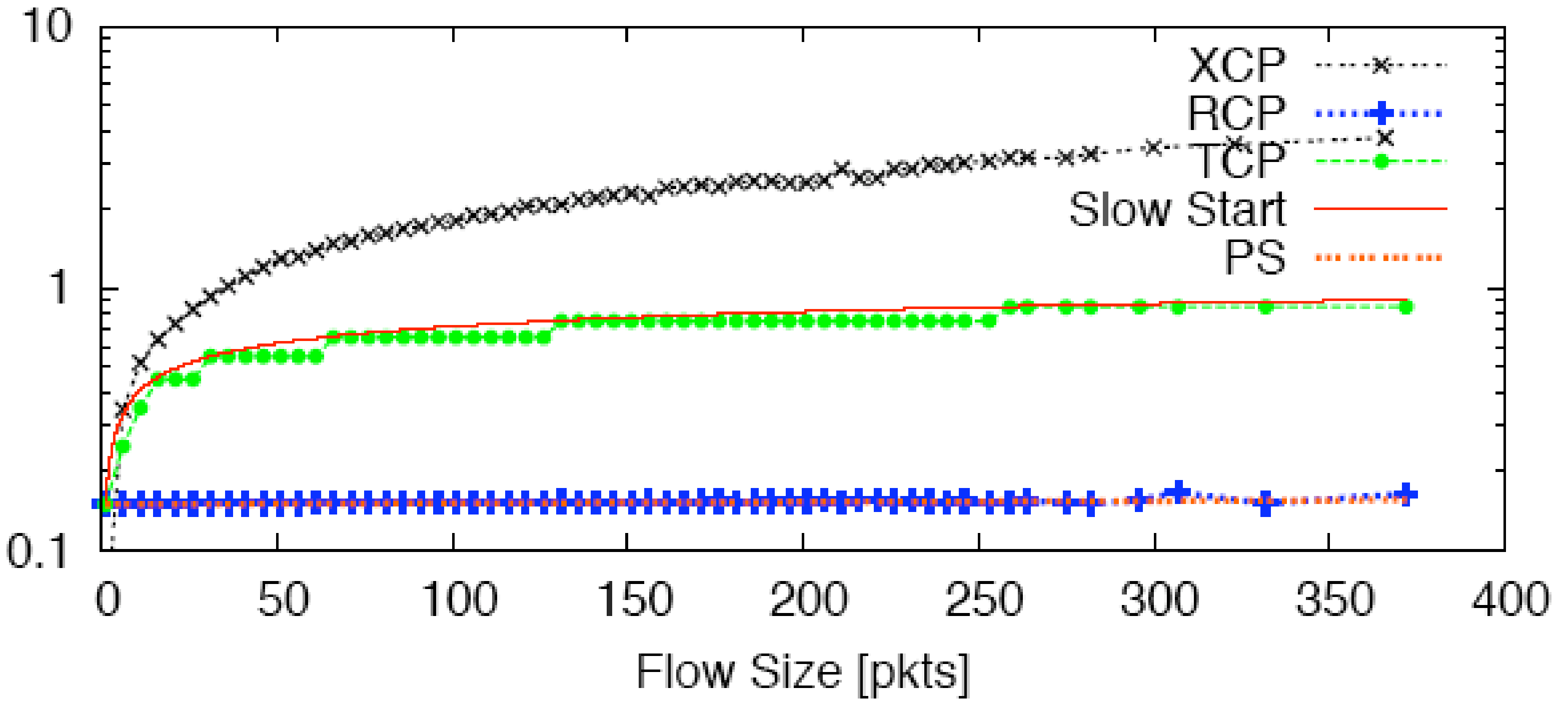


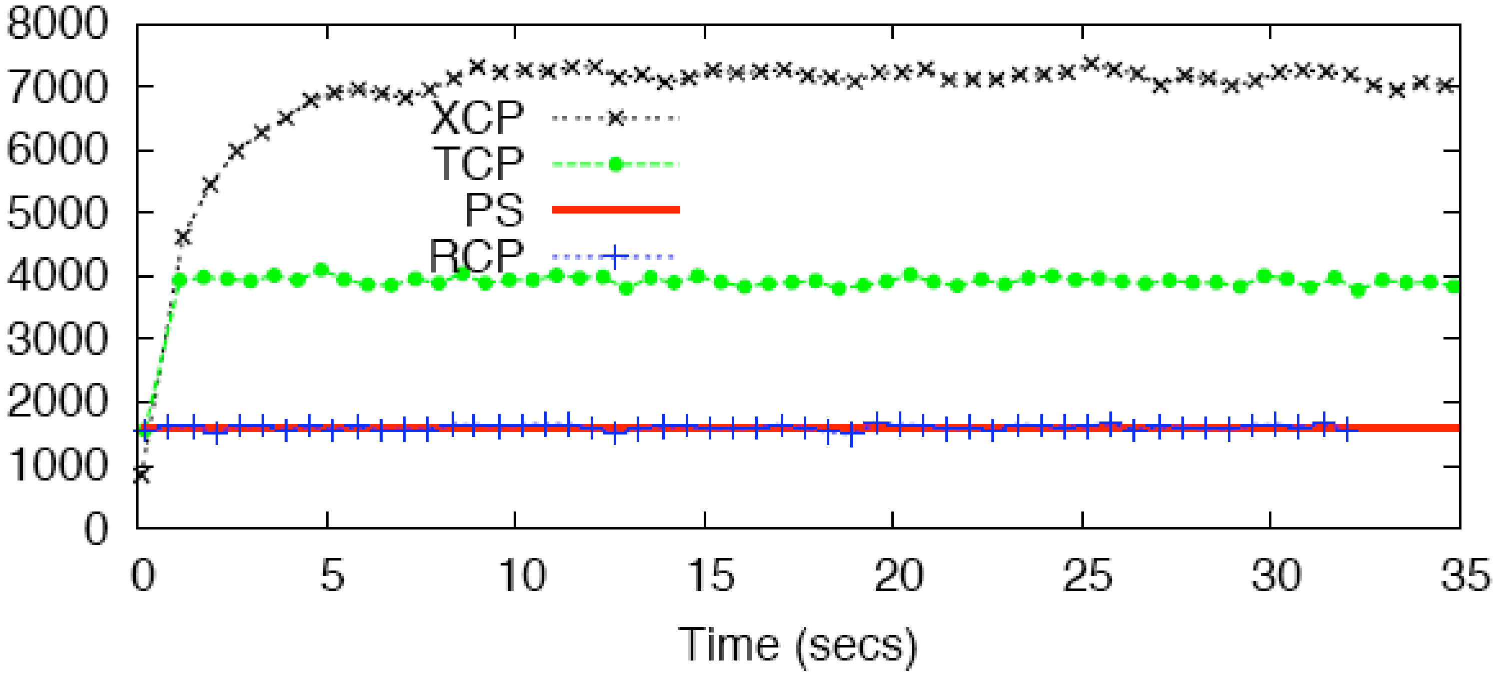
RCP vs. TCP vs. XCP





Example 3: Achieves PS for any flow-size distribution

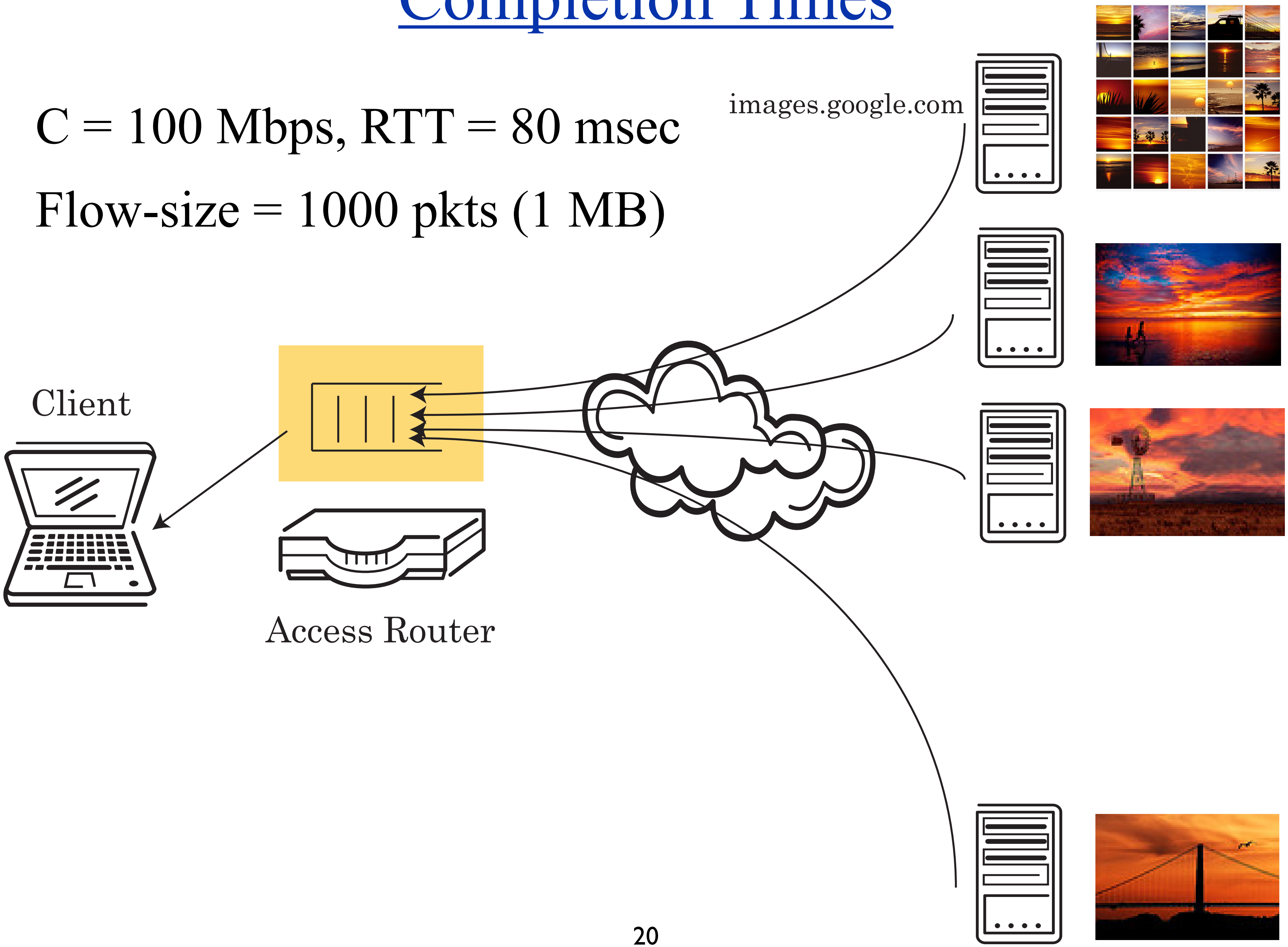




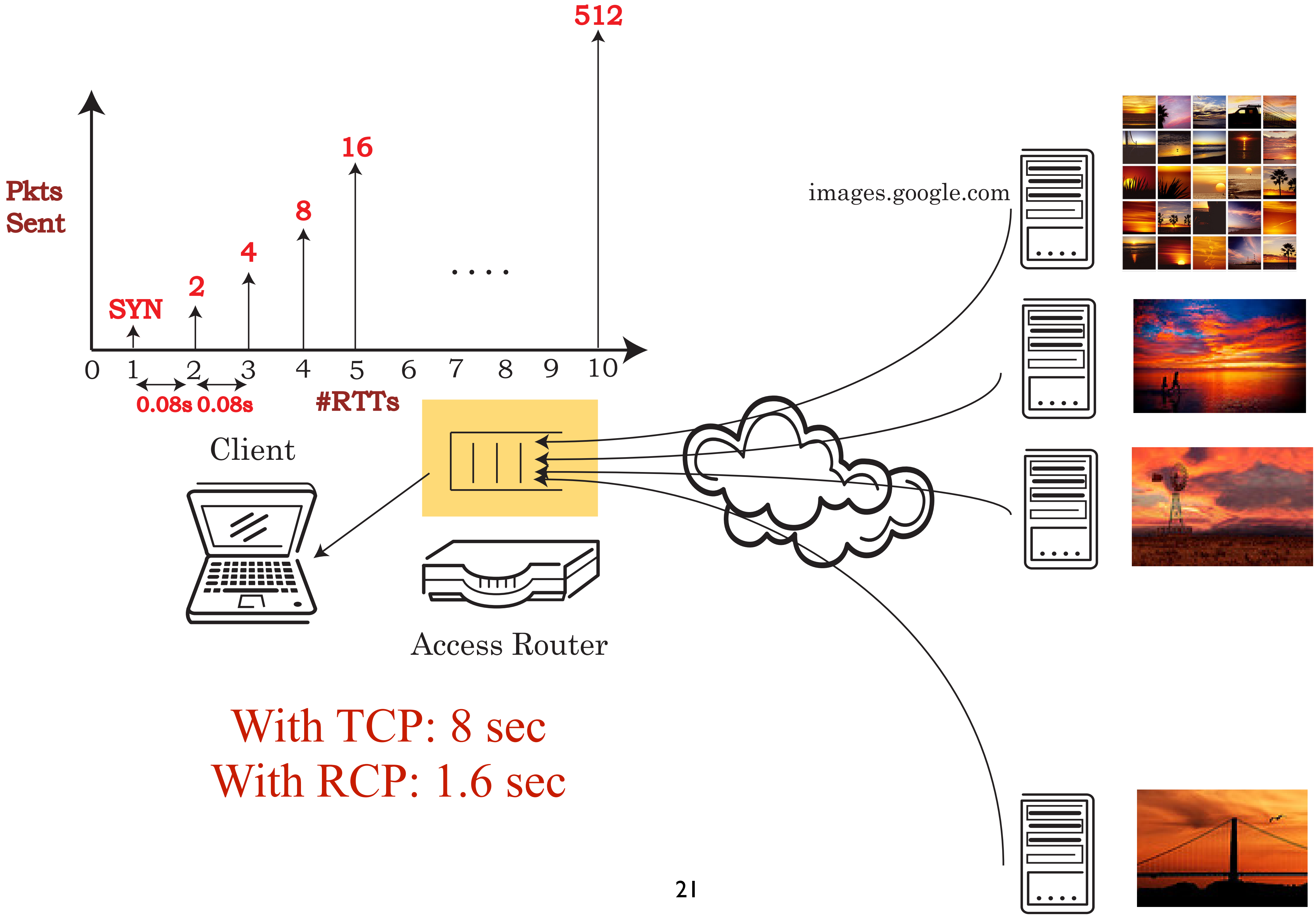
Example 1: Impact of RCP's Short Flow-Completion Times

C = 100 Mbps, RTT = 80 msec

Flow-size = 1000 pkts (1 MB)



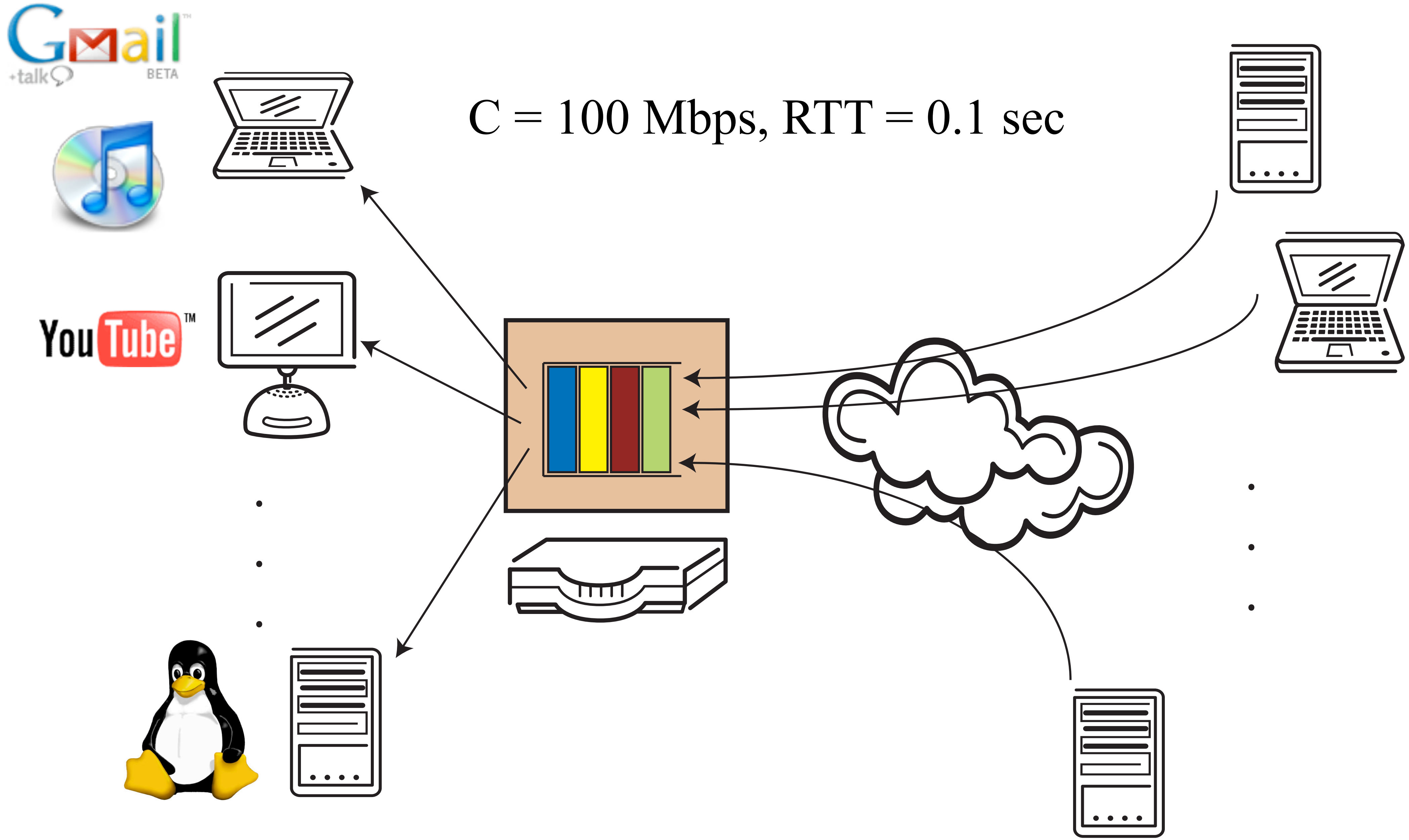
Example 1: TCP takes 5 times longer than need be



With TCP: 8 sec
With RCP: 1.6 sec

Example 2: Sudden Changes in Available Bandwidth

C = 100 Mbps, RTT = 0.1 sec

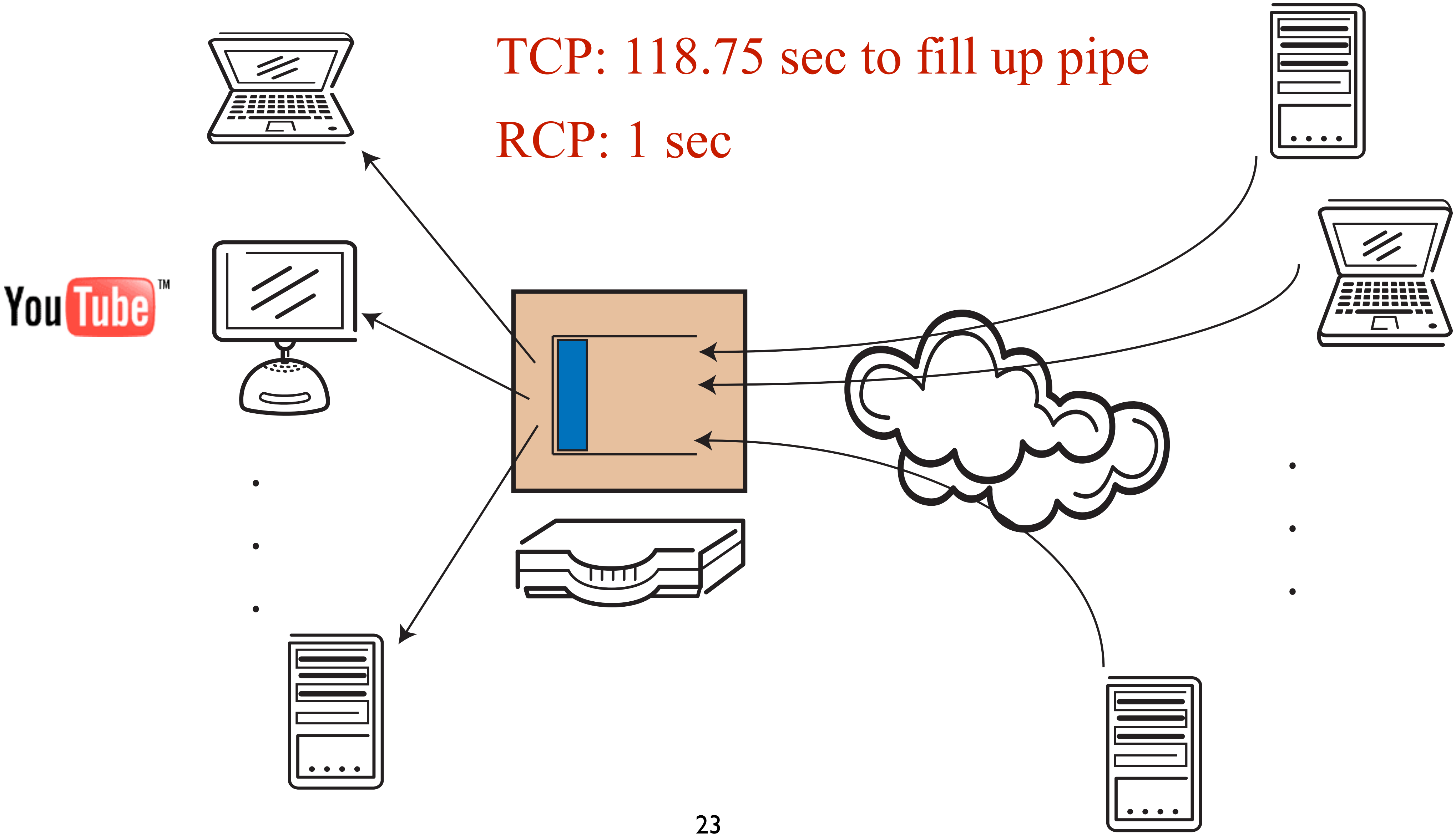


Example 2: Sudden Changes in Available Bandwidth

C = 100 Mbps, RTT = 0.1 sec

TCP: 118.75 sec to fill up pipe

RCP: 1 sec



Is RCP Stable?

RCP System:

$$\dot{R}(t) = R(t)\left[\frac{\alpha(C - y(t)) - \beta\frac{q(t)}{d(t)}}{Cd(t)}\right]$$

$$d(t) = d_0 + \frac{q(t)}{C}$$

$$\dot{q}(t) = \begin{array}{l} [y(t) - C] \text{ if } q(t) > 0 \\ [y(t) - C]^+ \text{ if } q(t) = 0 \end{array}$$

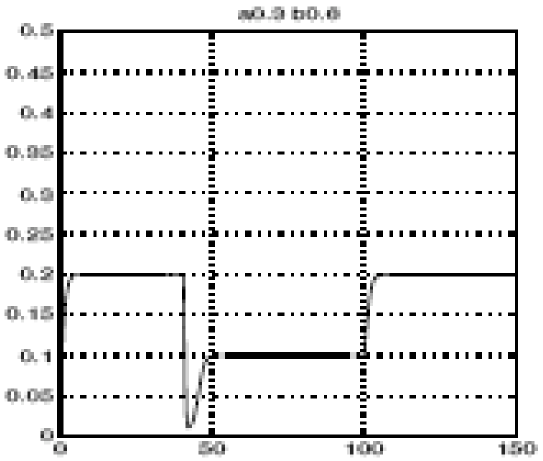
$$y(t) = N \times R(t - d_0)$$

Equilibrium:

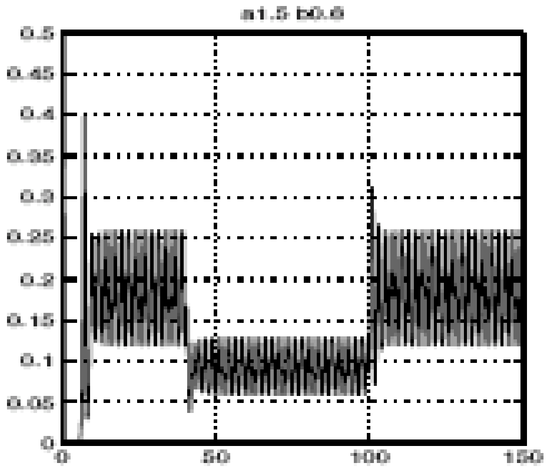
$$\dot{R}(t) = 0;\ \dot{q}(t) = 0$$

$$(R^*, q^*) = (\frac{C}{N}, 0)$$

$\alpha = 0.9, \beta = 0.6$

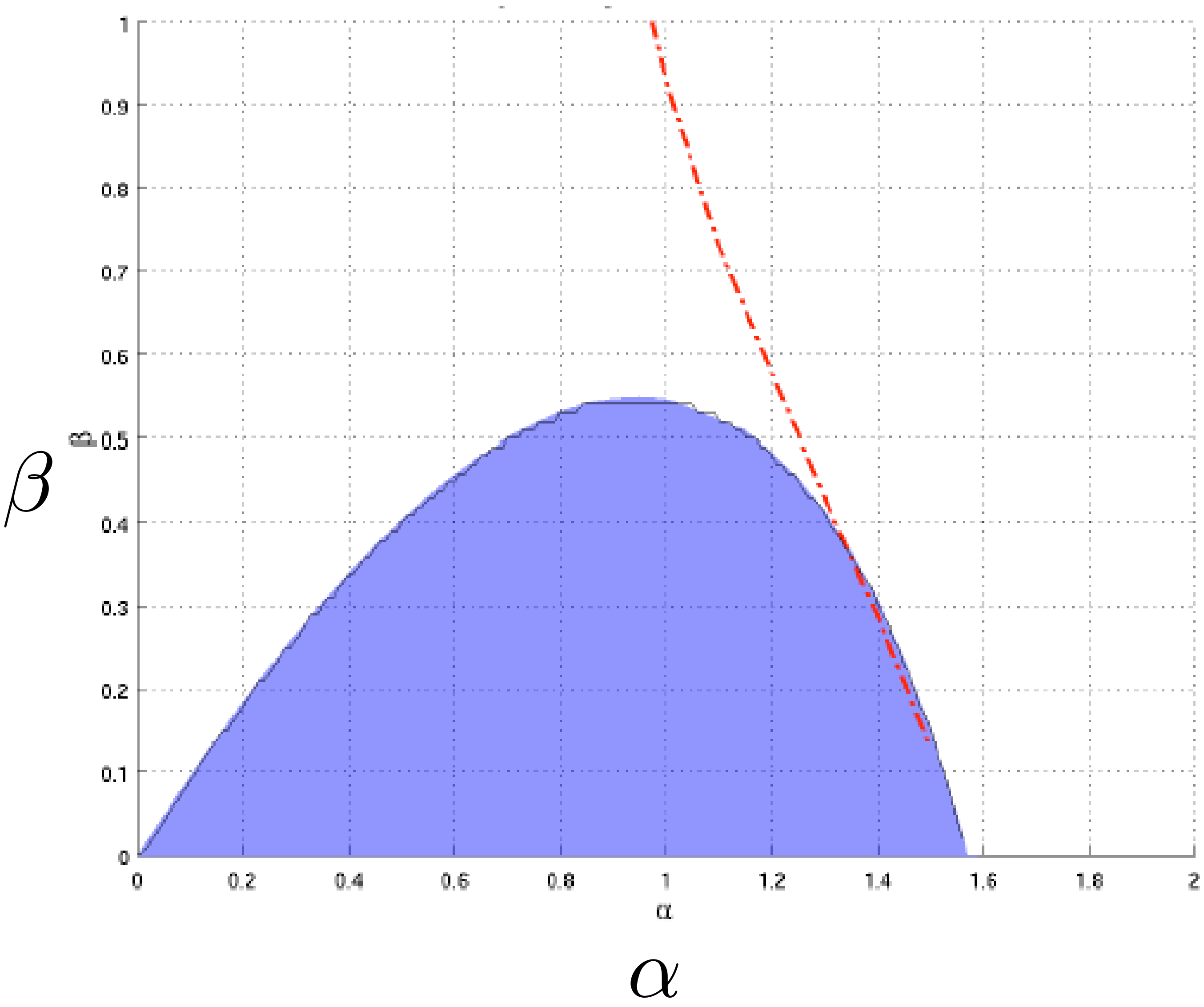


$\alpha = 1.2, \beta = 0.6$

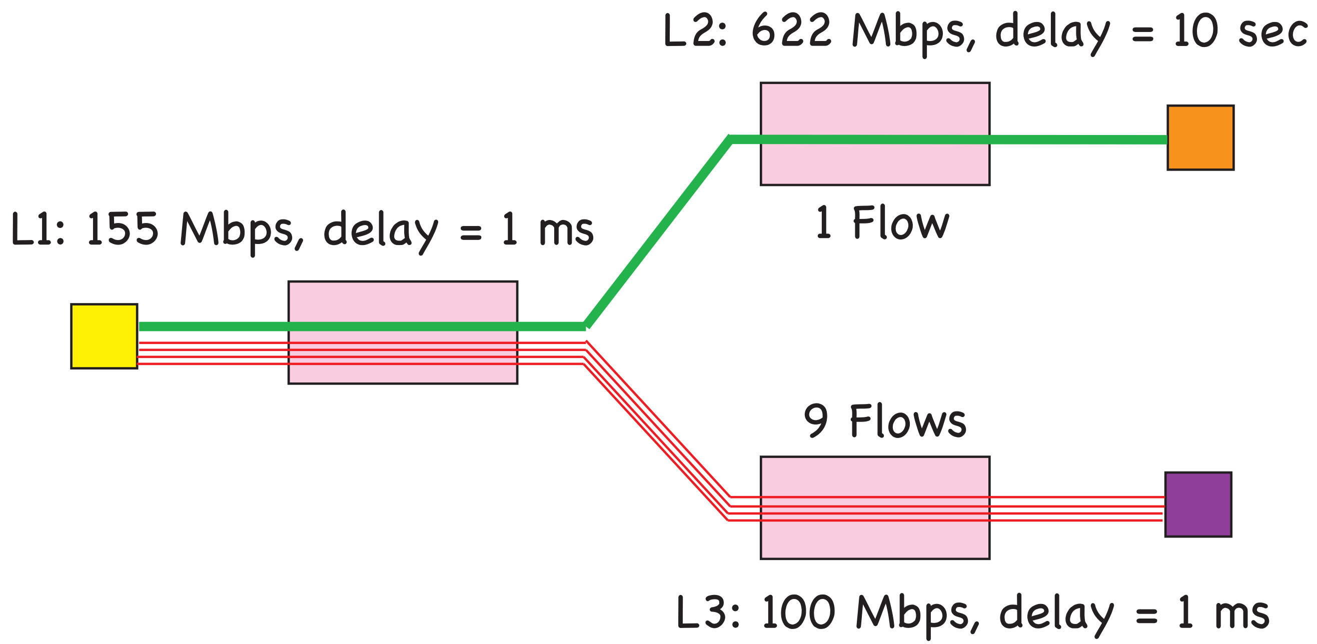


RCP is Stable

Stable Independent of C, RTT and # Flows



RCP Stability: Heterogeneous RTTs



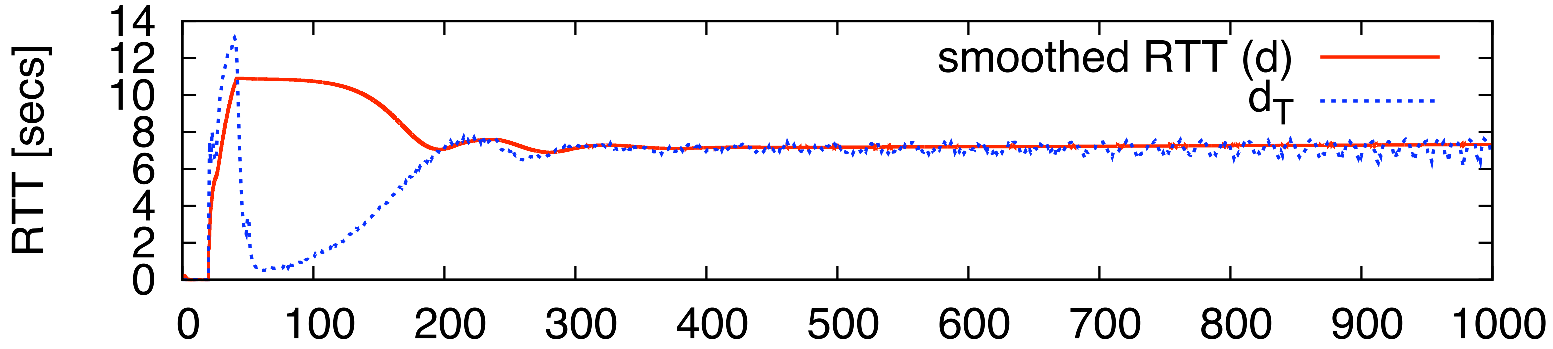
L2: 622 Mbps, delay = 10 sec

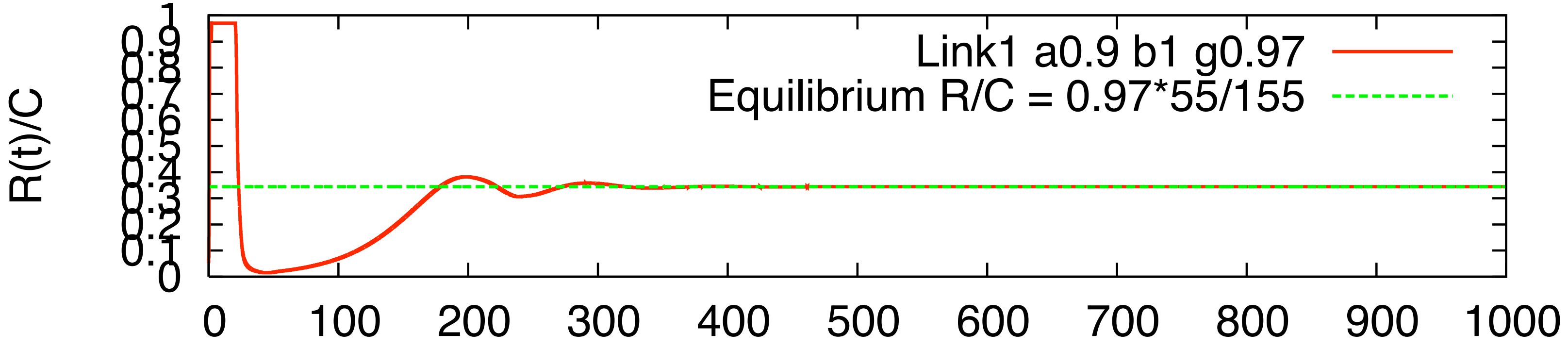
1 Flow

L1: 155 Mbps, delay = 1 ms

9 Flows

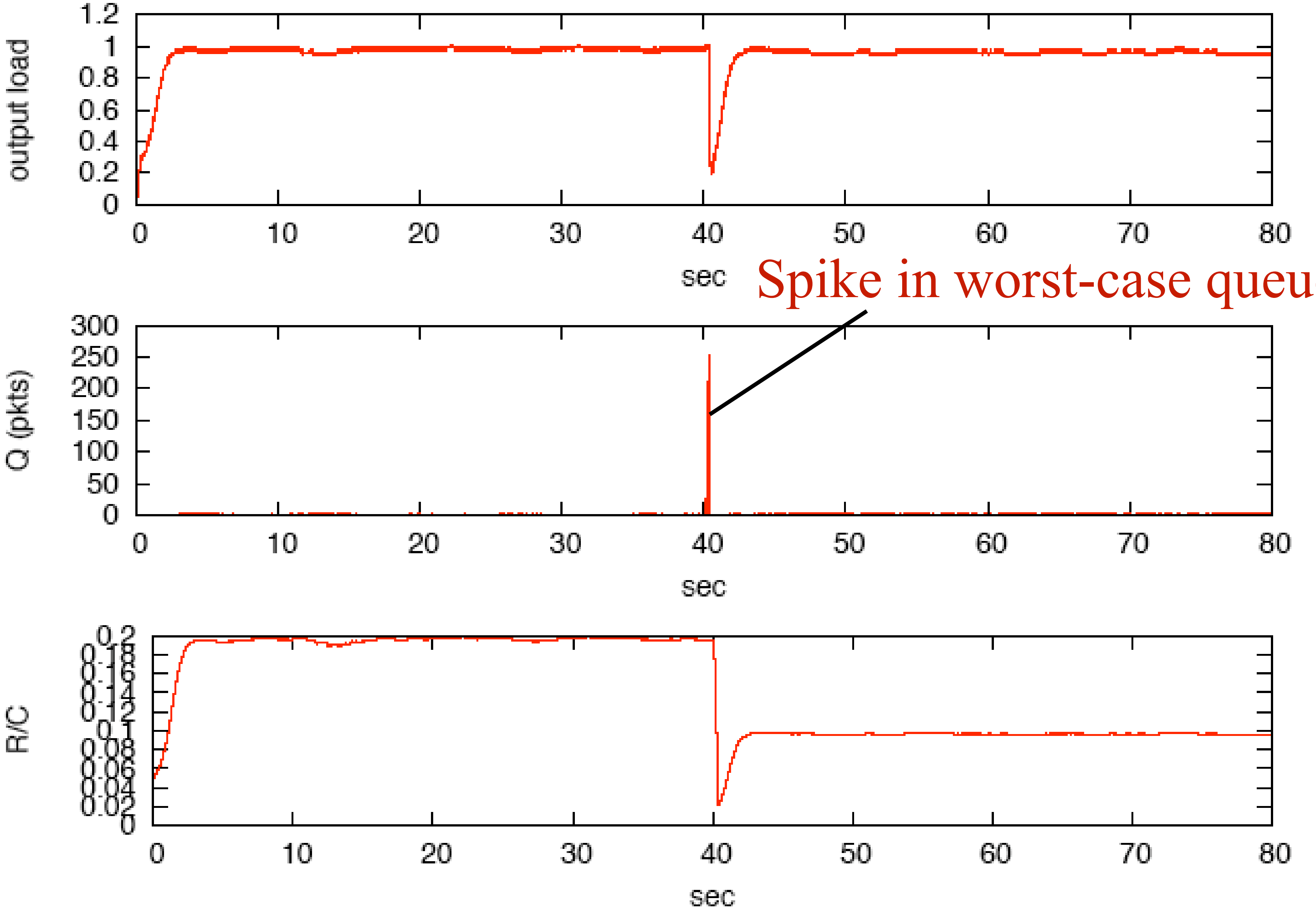
L3: 100 Mbps, delay = 1 ms





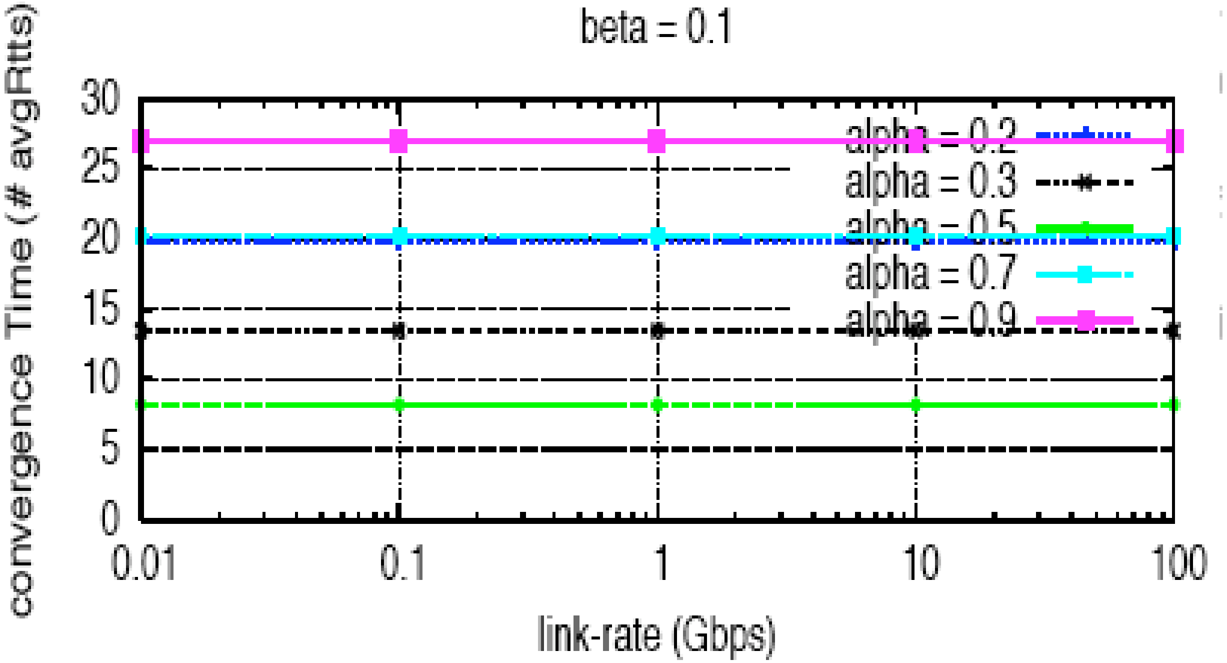
RCP’s Weakness: Sudden Traffic Changes

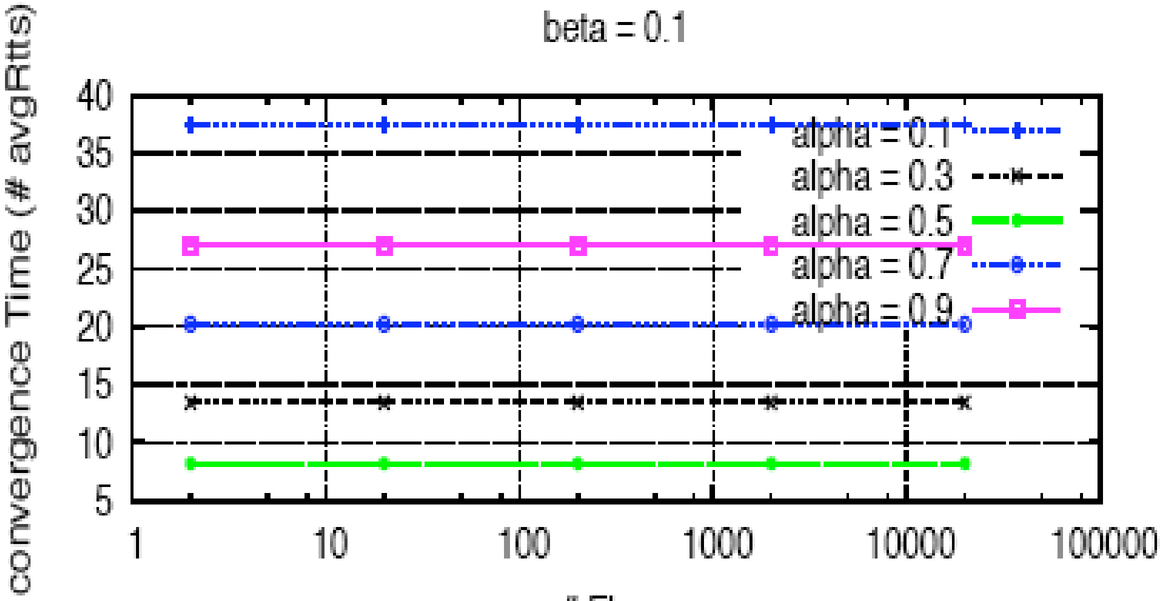
Example: Offered load doubles within a round-trip time



Spike in worst-case queue size

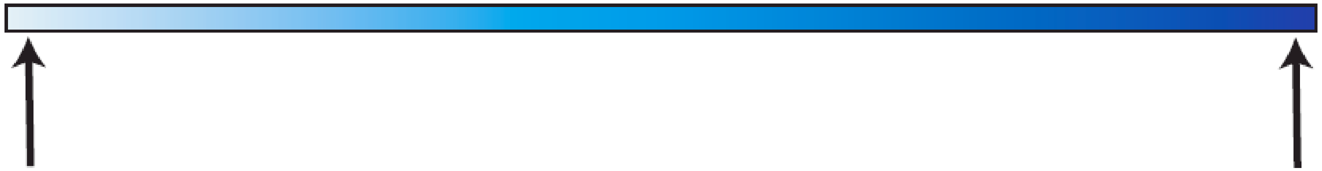
Example: RCP's Convergence Time





RCP-AC: Rate Control Protocol with Acceleration Control

- Enhanced RCP:
  - Rate Control
  - Acceleration Control
  - Feedback Control



- acceleration: small

- Sudden changes: works well, small queues, near-zero losses, XCP-like

-Common case: long flow completion times

- acceleration: large

- Sudden changes: too aggressive, large queues, packet-losses

- Common case: finishes flows fast

Conclusion

- TCP is unsuitable for high bandwidth-delay network
- XCP is a bold attempt but hasn't achieved what it set out to do
- Making network faster doesn't help; Flow duration is constrained by protocols
- Consequences of RCP:
  - For Internet-sized flows: 10 times faster download times than TCP; 30 times faster than XCP.
  - Inherently fair among flows
  - Provably stable independent of link-capacities, RTT, #flows
  - Easy to police flows
  - Easy to provide differential service for flows - GPS
  - Efficient use of high bandwidth-delay networks

Acknowledgments

- Nick
- Committee members: Balaji Prabhakar, Scott Shenker, Mendel Rosenblum
- People I collaborated with when working on RCP:
  - Rui Zhang-Shen (Stanford)
  - Masayoshi Kobayashi (NEC Labs, Japan)
  - Hamsa Balakrishnan and Prof. Claire Tomlin (Stanford and UC Berkeley)
  - Sandy Fraser (Fraser Research)
  - Ashvin Lakshmikantha and Prof. R. Srikant (UIUC)
  - Glen Gibb and Greg Watson (Stanford)
- HPNG Group: Rui, Neda, Martin, Greg, Yashar, Sundar, Isaac, Da, Pablo, Gireesh, Guido
- Judy Polenta
- Friends
- Family
- Subhachandra Chandra
- Coffee shops in Stanford and Palo Alto: Bytes, Peets (Clark center), Coupa, Cafe Del Doge...

PCP: Probe Control Protocol

- Sends a sequence of probe packets at a specific rate to detect if network can support the rate; else end-point tries a new probe at a slower rate.
- Works best: Lightly loaded links with good history information (solves the problem of slow-start in lightly loaded regime with history info.)
- No Network support
- Congestion Control: Efficiency, fairness and stability. No new benefits or any different from TCP. Has to rely on Loss and/or Delay.
- Anti Congestion Control:
  - - Has TCP's O(log n) RTTs at start-up. Uses history for improvement.
  - - Heuristic emulation of explicit feedback. Achieves a subset of properties. Will not have strong properties of explicit feedback. Example (from Sally Floyd): New flow wants to blast at 1 Gbps for 2 RTTs and stop.
  - - Accept/Reject probe test: Delay-based measurements. Need accurate timers and small jitter at end-hosts for correctness.

Why not just increase TCP's initial cwnd

1. Does not solve all problems:
   - Long FCTs due to slow AIMD
   - Unfairness
   - Filling up buffers
2. Sunny day: O(log n/w_init) RTTs
3. Rainy Day:
   - Increases packet-losses. Early exit to CA mode. Worse than conservative SS.
   - Increase in unfairness with DropTail Buffers. Different flows in different stages of SS. Those in later stages benefit at the expense of flows in earlier stages.
   - Small buffers. Flash-crowds. Above problems are magnified
4. Summary: Increasing the initial window size and/or the rate of increase in slow-start while having only loss as feedback from the network could be either be too aggressive or conservative, depending on the network conditions.

But PS (GPS) is not the best for minimizing FCTs...

1. From Theoretical point of view:
   - Needs some additional information on flow-sizes: Either complete information on flow-sizes (SRPT), or age of a flow for Feedback policy.
   - Feedback policy: Does not always beat PS (depends on flow-size distribution).
   - What is the optimal discipline in the network-case ?
2. From the practical point of view:
   - Job-size based disciplines give higher priority to the shorter flows, but because these flows do not have many packets in them there is only so much better one can do as compared to PS. Approximating PS already gets down the number of RTTs these flows are made to last to a small number (O(1) RTTs), beyond which we find SRPT/SJF have diminishing returns.
   - PS is fair :) Easier to explain what the system is doing.

More on RCP’s Weakness

- Bound it
  - Hard with the q(t) term
  - Alternative: Rate reduces and queue drains within 2-3 RTTs, now work without the q(t) term and bound convergence time
- Prevent it
  - RCP-AC
- Make it harder to enter this regime
  - Stamp in *Acceleration* along with *Rate*
  - Small *Acceleration:* Lightly loaded link, relatively high R(t) (compared to C), heavily congested link.

Proportional bandwidth-sharing in RCP

- Compute R(t) just as in RCP
- Stamp the rate differently for different flows - for example, stamping R in packets of flow 1 and w*R (w is between 0 and 1) in packets of flow 2 shares bandwidth in the ratio 1:w
- R has to be allowed to increase to C/w as opposed to C - ensuring that even if there are just flows of the lower priority, they can make use of the entire bandwidth
- The rate stamped is capped at C - so even if R equals C/w and a high priority flow comes by, it gets at most C
- This can be extended to arbitrary number of priority levels

Description of RCP-AC

- Rate Controller: Just like in RCP
- Acceleration Controller:

Controls acceleration

$$\phi = \alpha(C - y(t)) \cdot d + (\gamma B - \beta q(t))$$

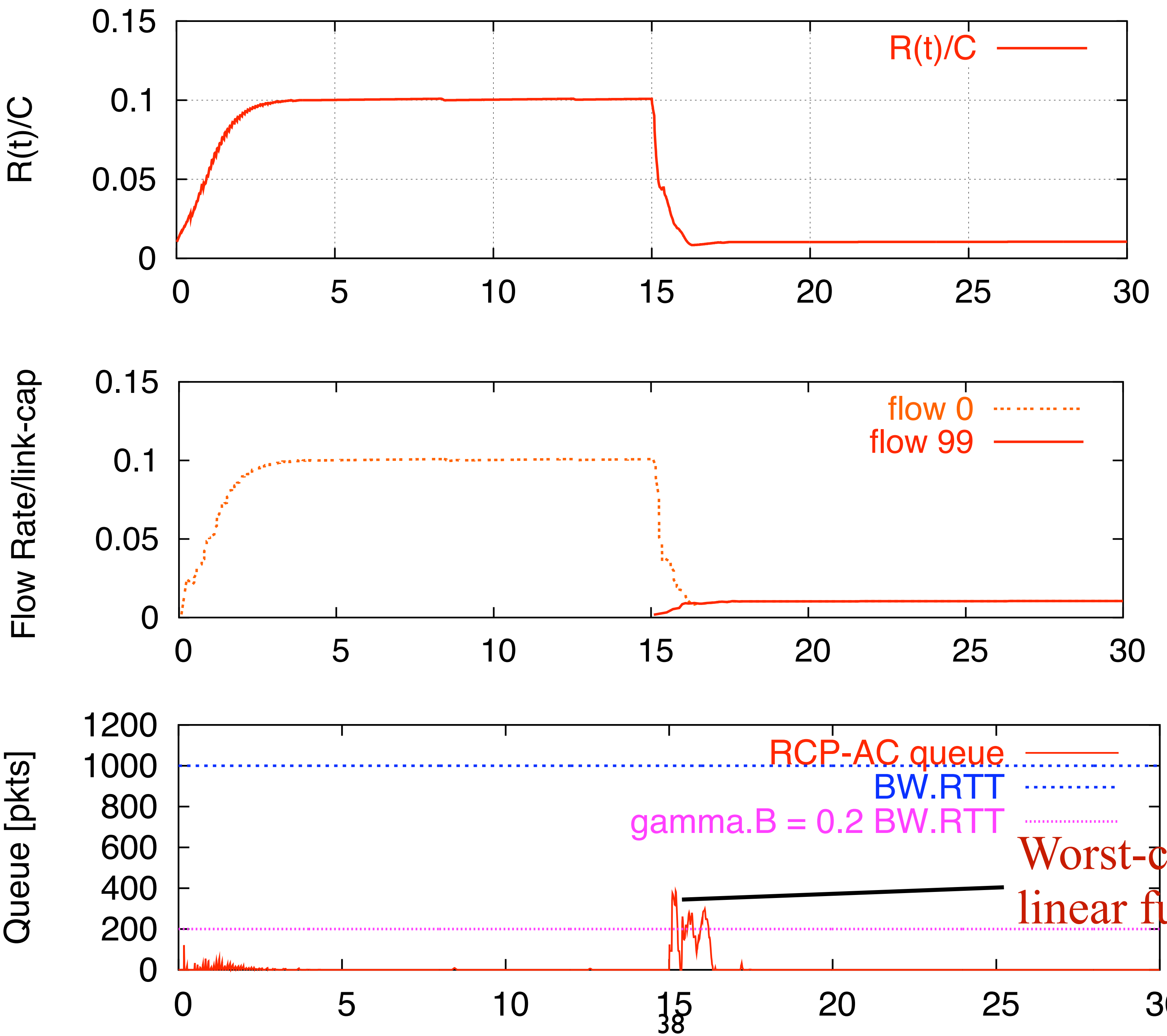
- Feedback Controller:

$$feedback_i^- = -\frac{\max(0, \frac{cwnd_i}{rtt_i} - R(t)) \cdot rtt_i}{\frac{cwnd_i}{rtt_i} \cdot d}$$

$$feedback_i^+ = \min[\frac{\max(0, R(t) - \frac{cwnd_i}{rtt_i}) \cdot rtt_i}{\frac{cwnd_i}{rtt_i} \cdot d}, p_i]$$

$$p_i = \max(\phi, 0) \cdot \frac{\max(0, R(t) - \frac{cwnd_i}{rtt_i}) \cdot \frac{rtt_i}{cwnd_i}}{\sum^K \frac{rtt_i}{cwnd_i} \max(0, R(t) - \frac{cwnd_i}{rtt_i})} \cdot \frac{rtt_i}{d}$$

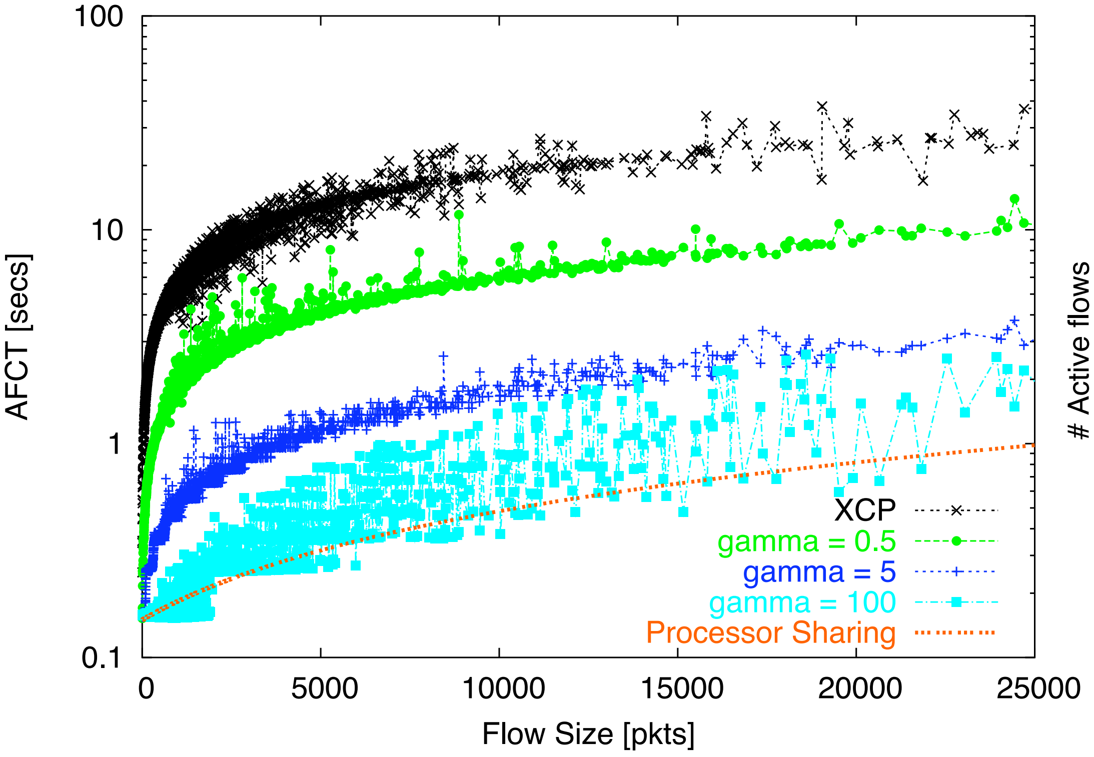
RCP-AC under Sudden Traffic Changes

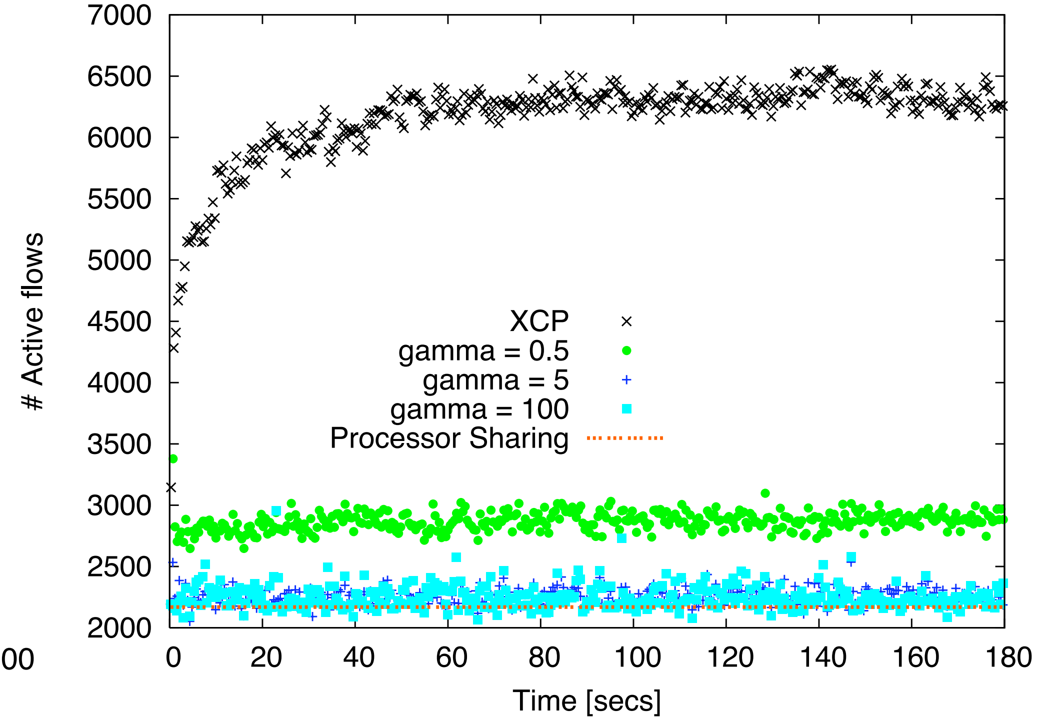


Worst-case queue:
linear function of γB

RCP-AC under Aggregated Traffic

Flow-completion times get closer to Processor Sharing as γ gets larger



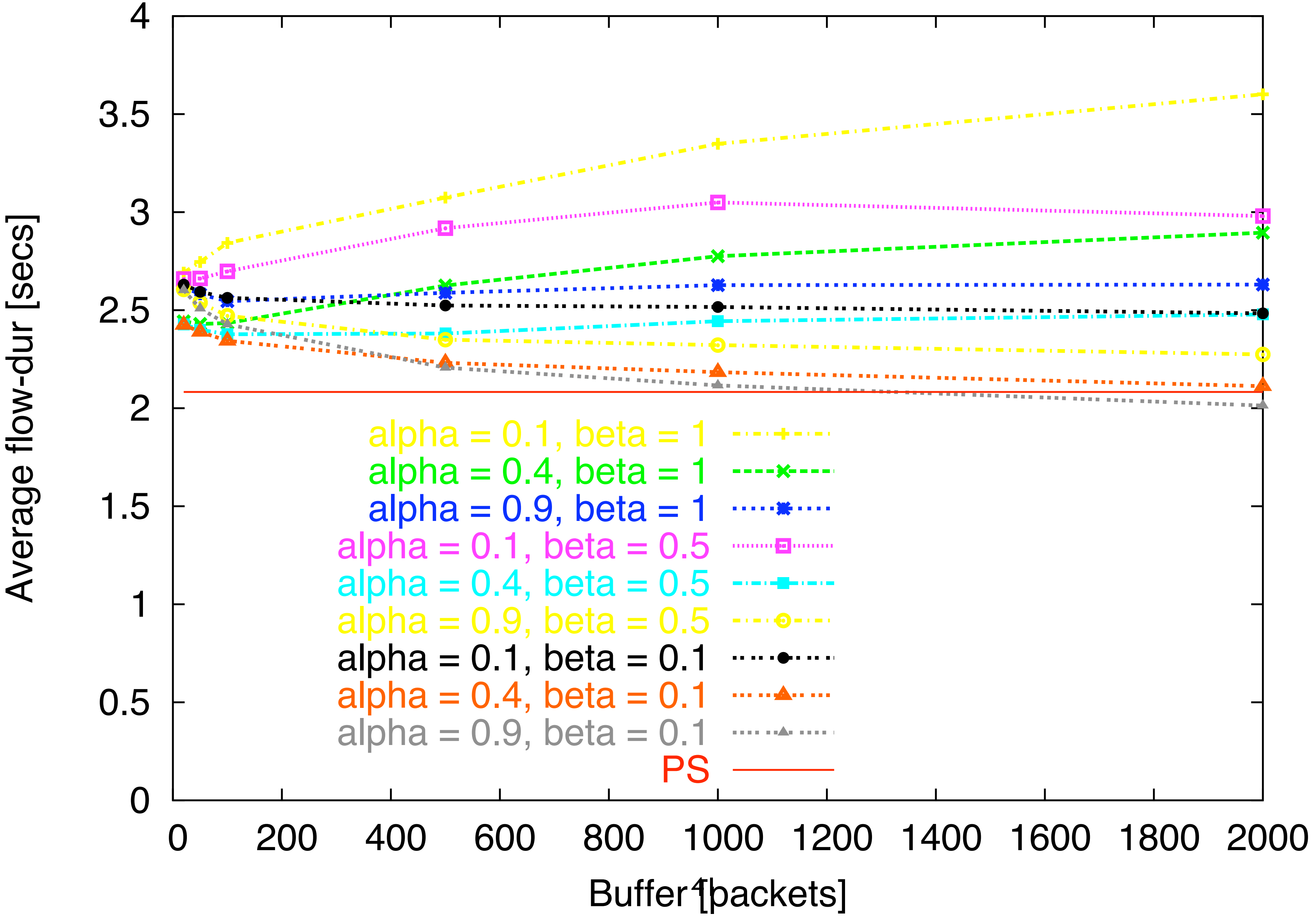


Loss Detection and Retransmission Mechanisms

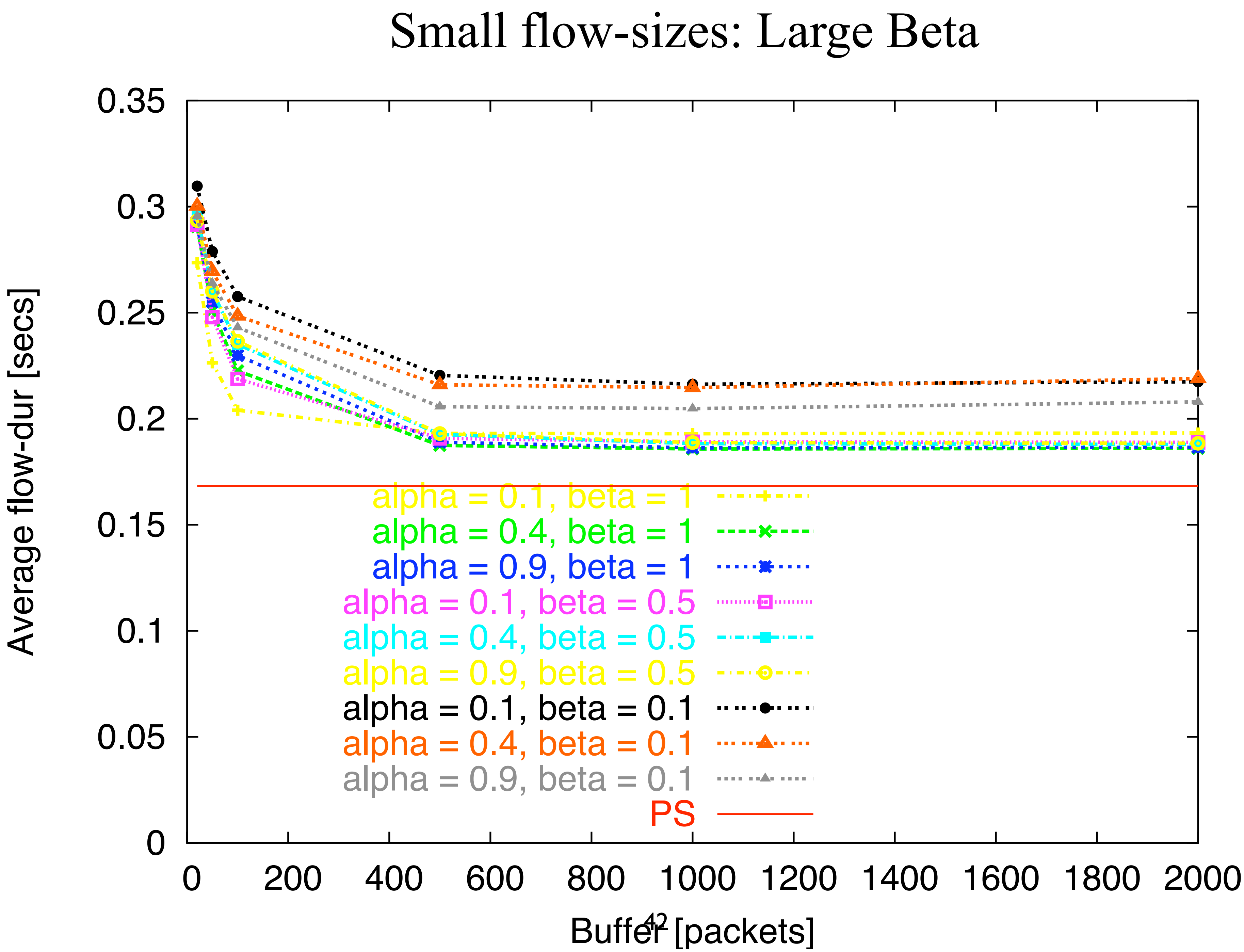
- Simulations:
  - Simulations done with bandwidth-delay buffering
  - Observed losses in flash-crowd scenarios
  - Receiver informs sender of the number of packets received. At the end, the sender times out and retransmits the difference. Continues until entire flow is transmitted.
  - Ignores the exact mechanisms to determine (at the receiver) which packets are lost, and how is this information notified to the sender.
- Implementation in Linux:
  - SACK Mechanism which informs the sender of the exact lost packets in every RTT.

Choosing RCP Parameters for short FCTs

Large flow-sizes: Small beta

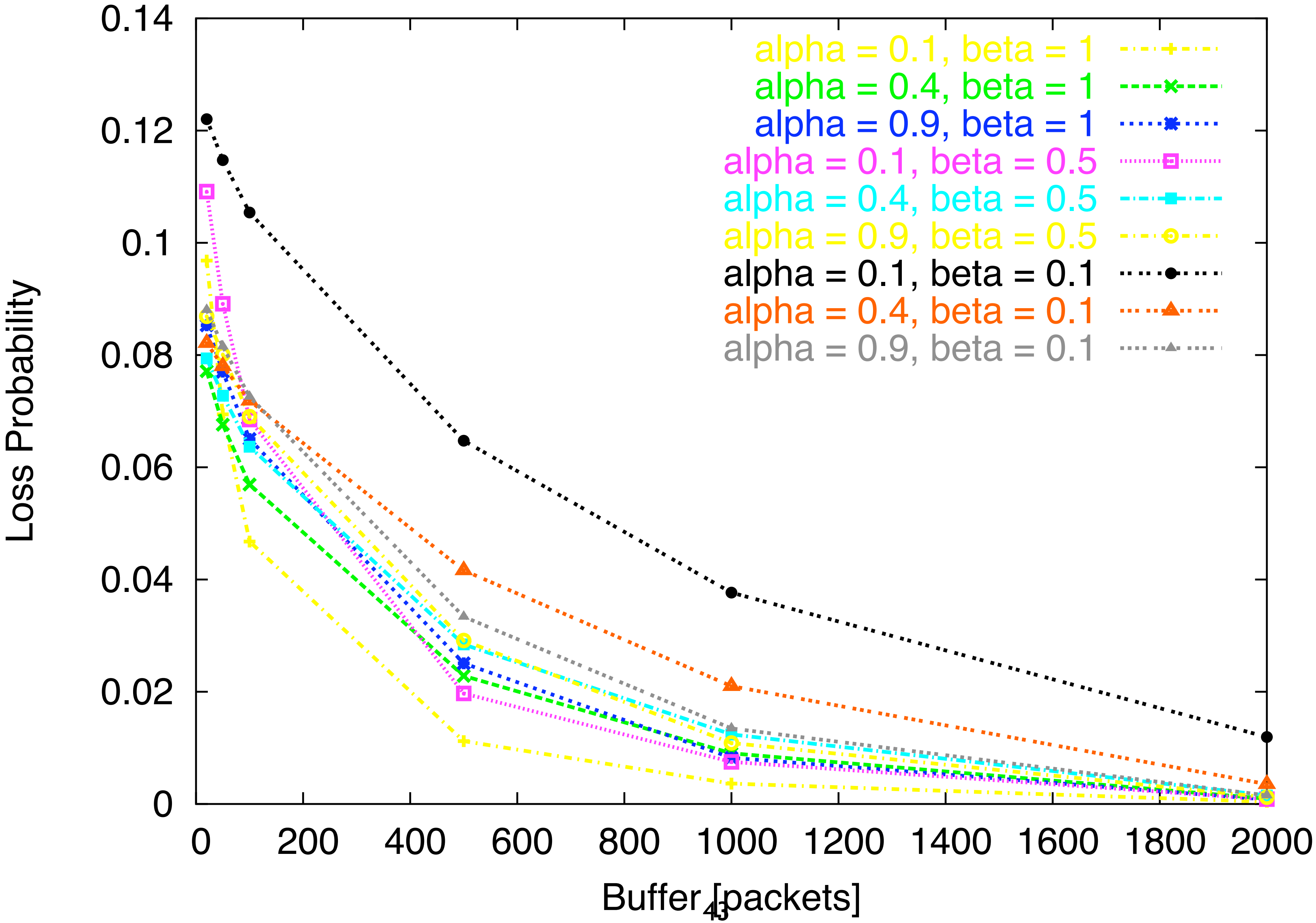


Choosing RCP Parameters for short FCTs

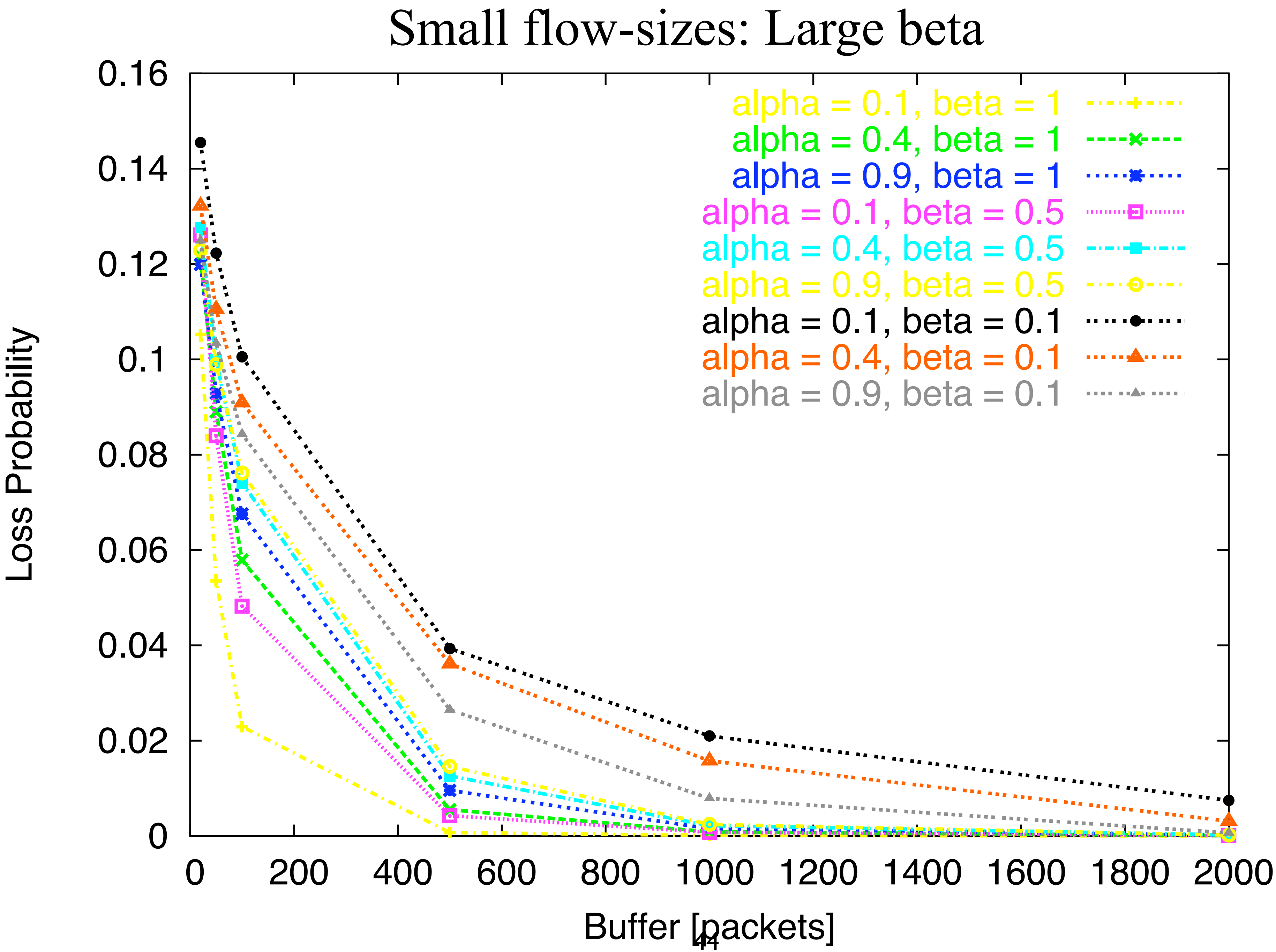


Choosing RCP Parameters for small queues

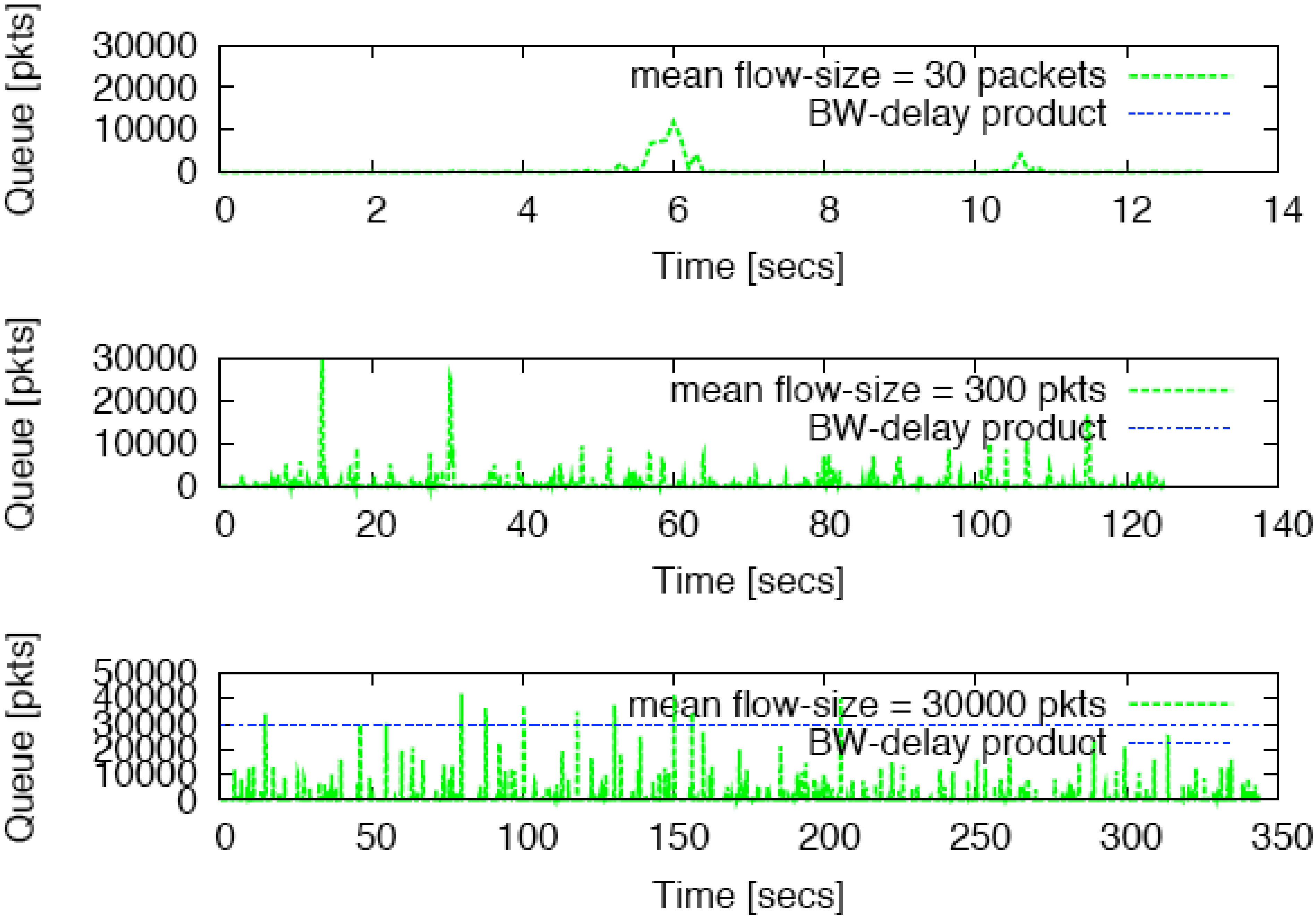
Large flow-sizes: Large beta



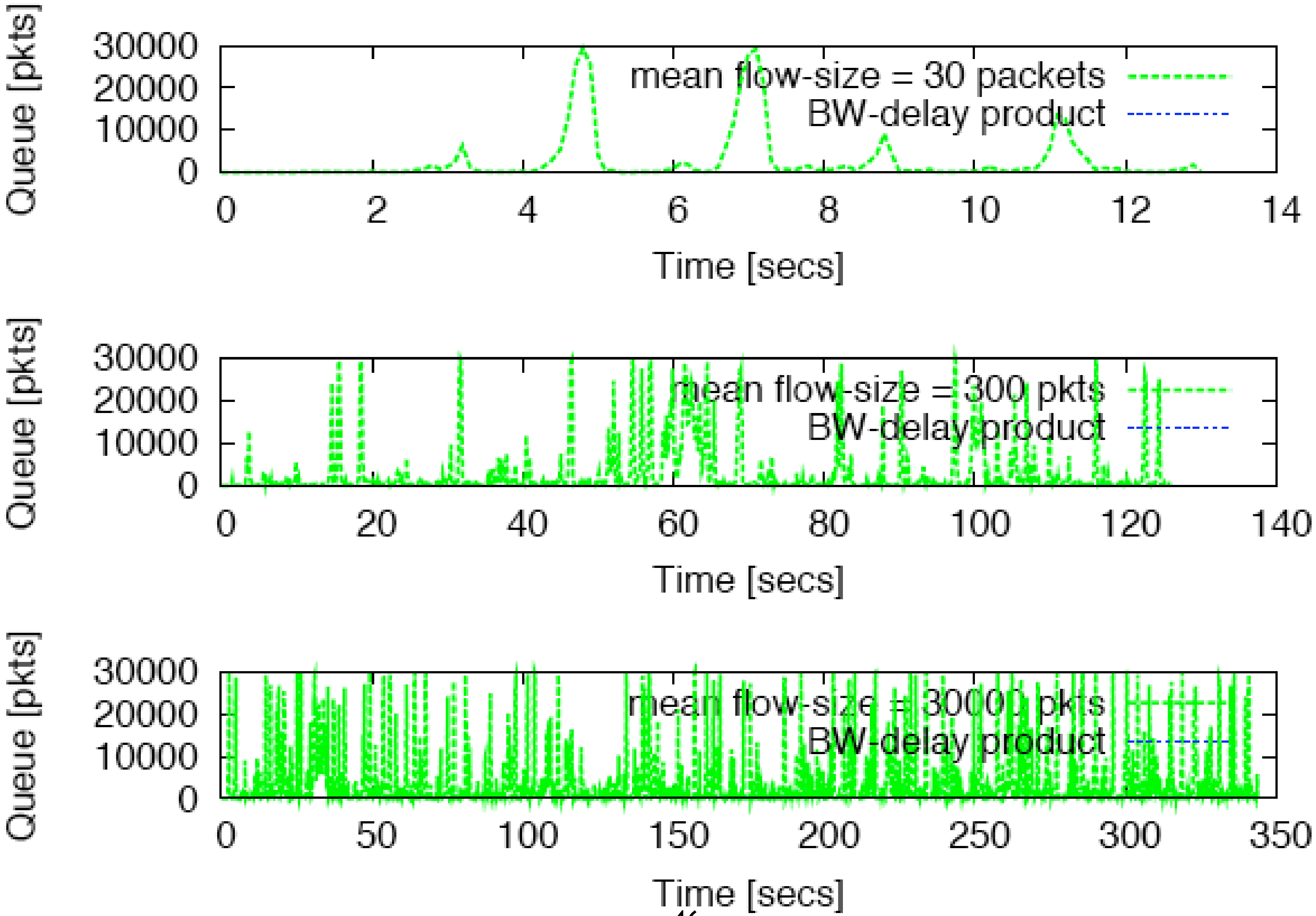
Choosing RCP Parameters for small queues



Queue-sizes under RCP



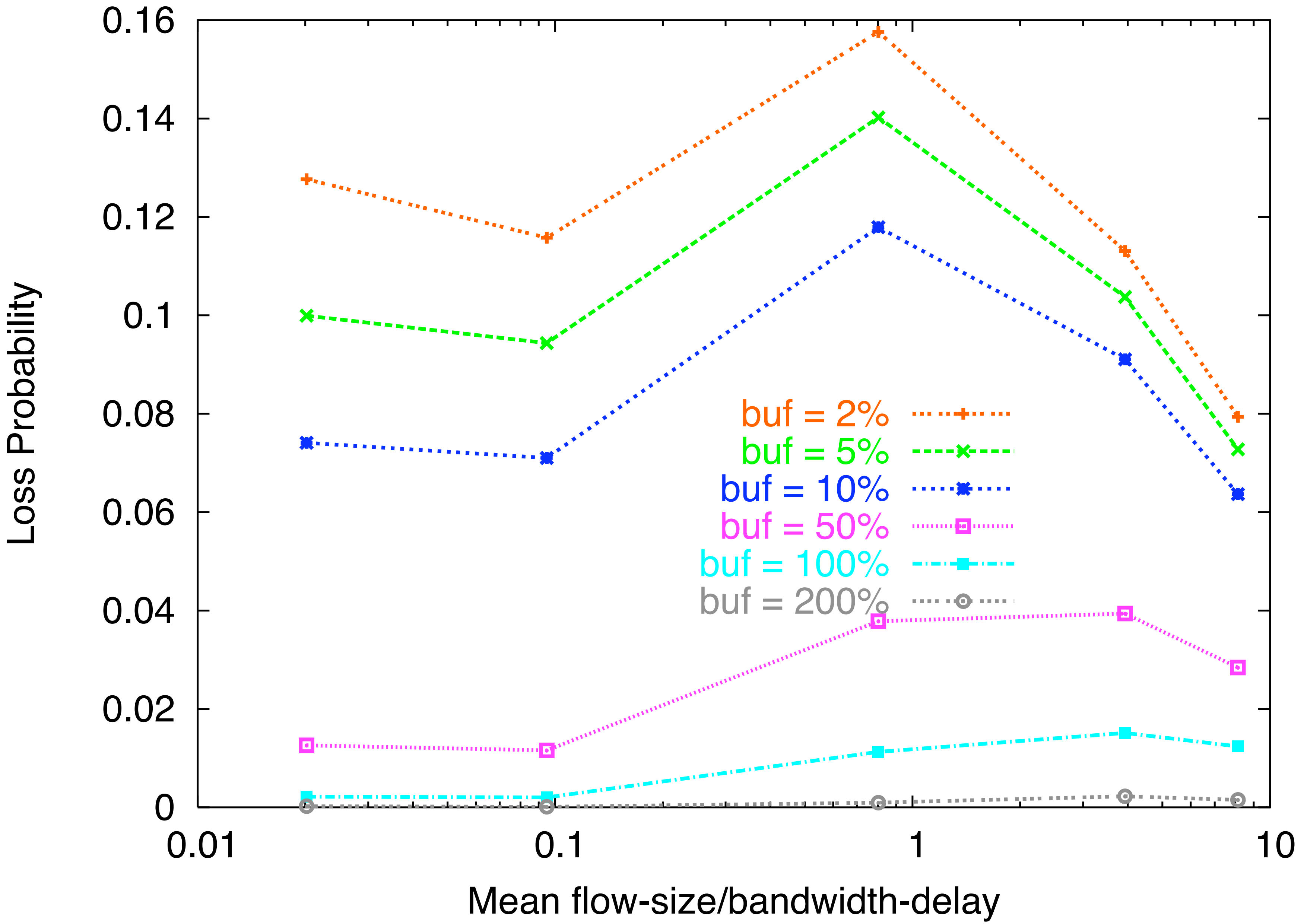
Queue-sizes under TCP (no AQM)



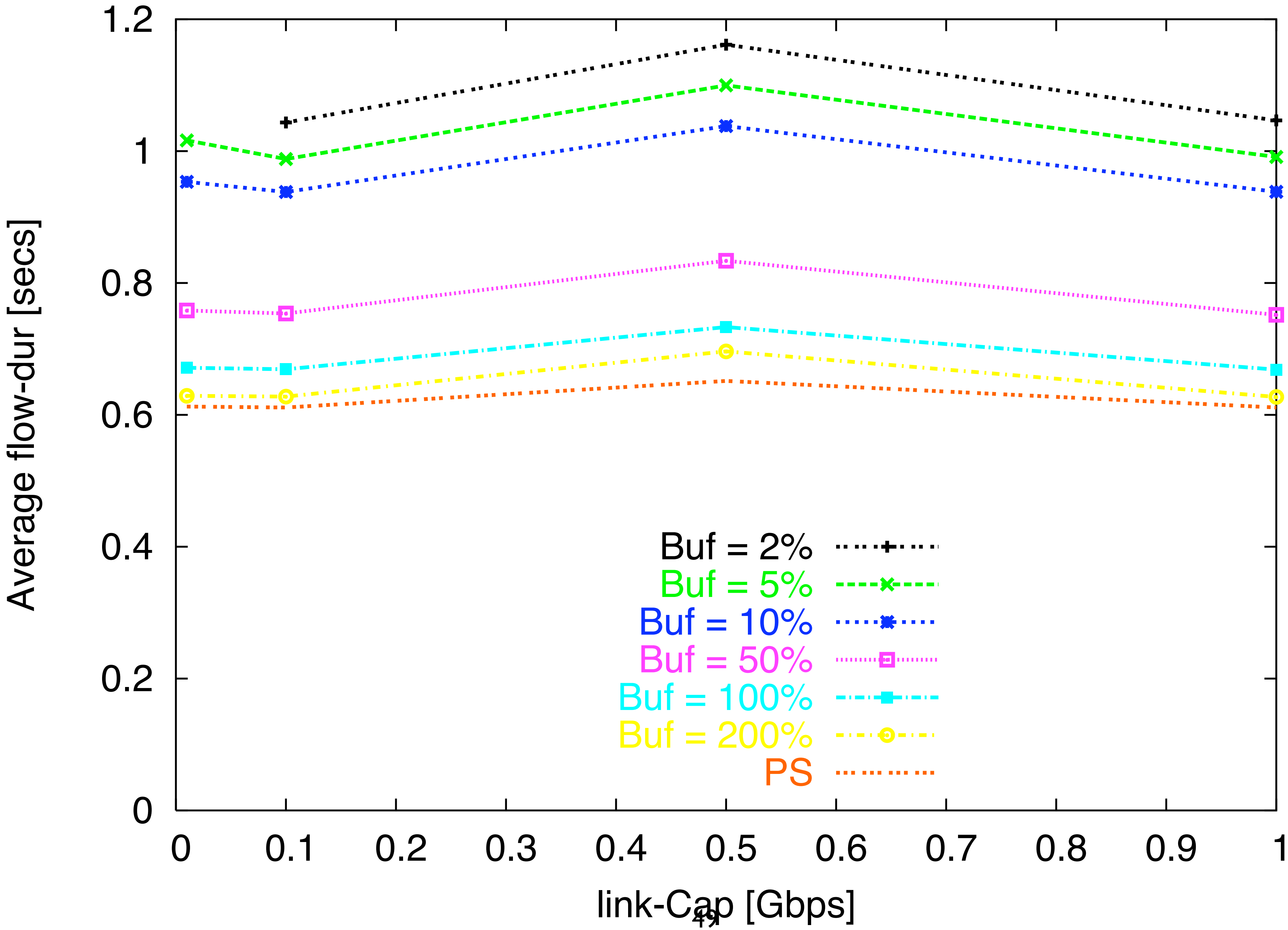
RCP's Buffer-size Requirements

- Static long flows: Zero equilibrium queue
- Relative Mean flow-size to bandwidth-delay product
    - Large mean flow-size - less new arrivals, smaller buffers for any given loss-probability
    - Small mean flow-size - many new arrivals, larger buffer for any given loss probability
- As the Link-rate (C) increases: Buffer = O(C.RTT) for constant loss-probability

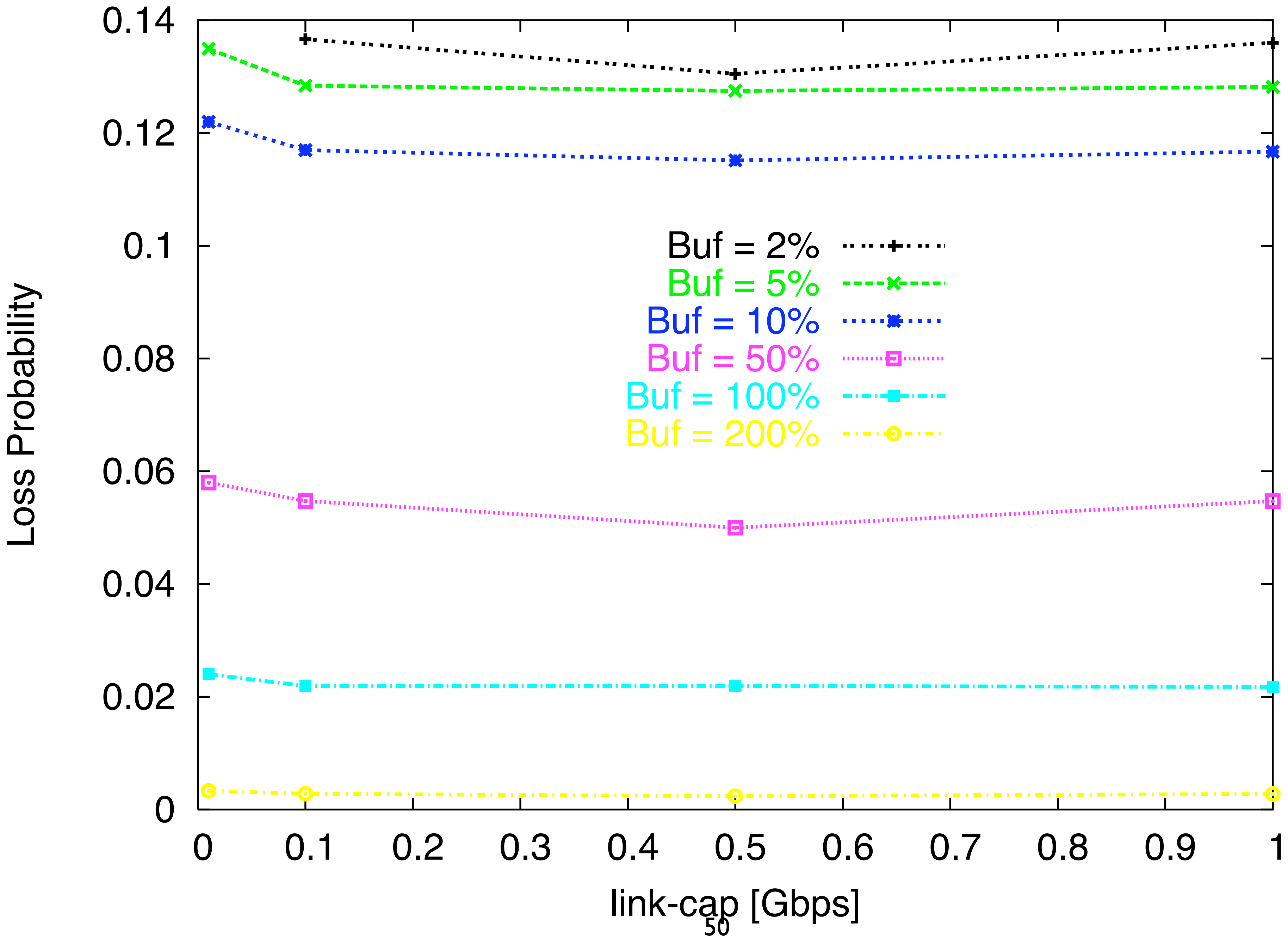
RCP’s Buffer-size Requirements: As Mean flow-size Increases



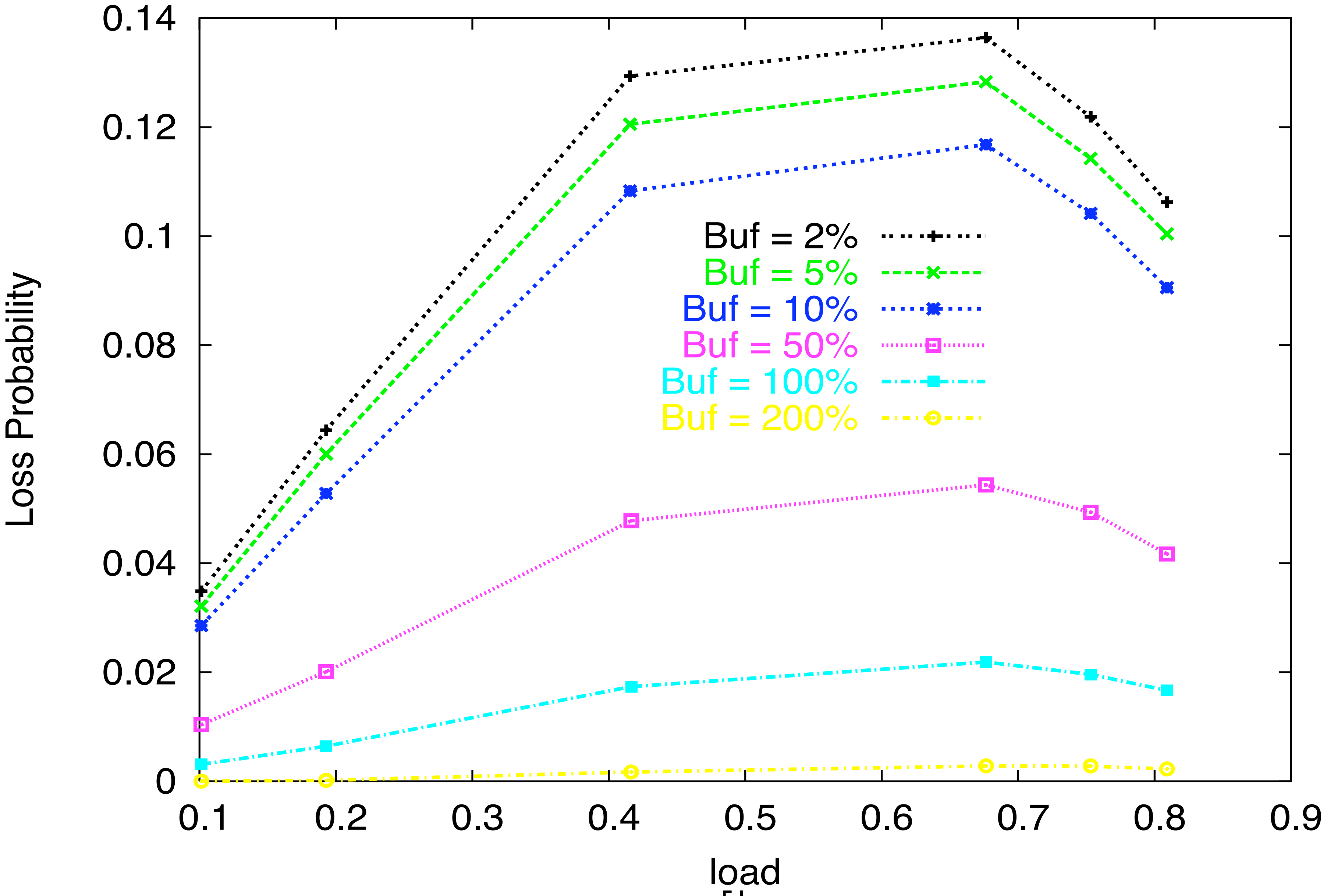
RCP's Buffer-size Requirements: As Link-rate Increases



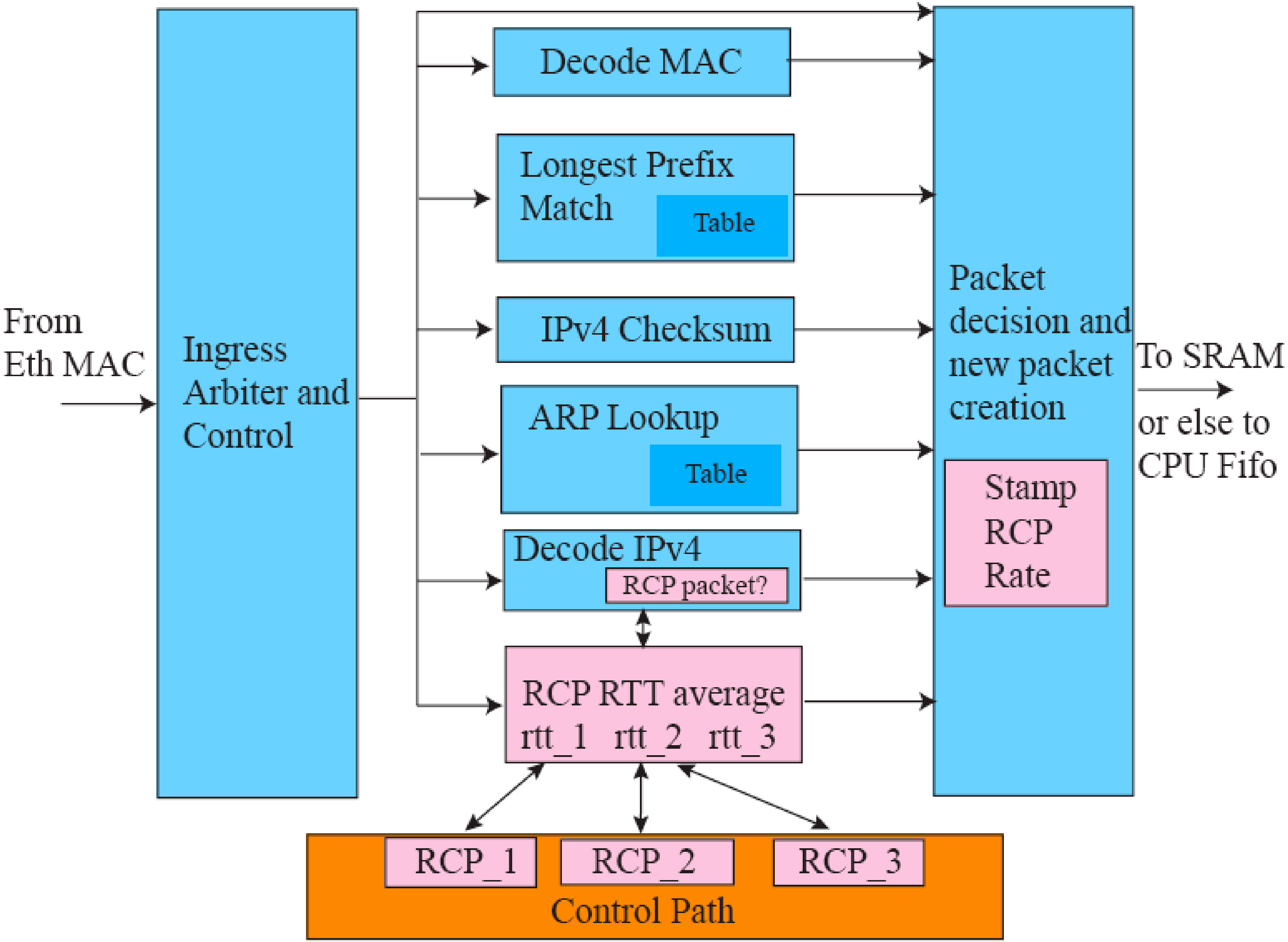
RCP's Buffer-size Requirements: As Link-rate Increases



RCP's Buffer-size Requirements: As Offered Load Increases



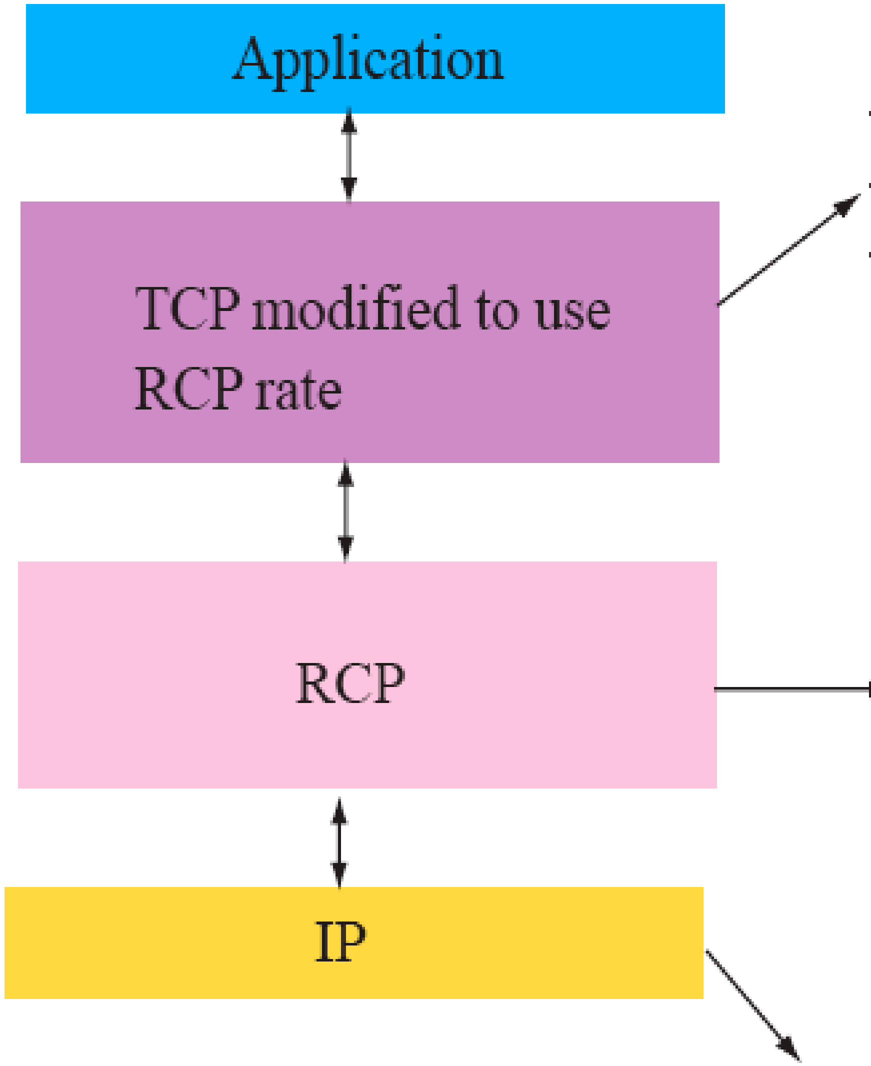
Implementation: RCP Algorithm on NetFPGA



Prototyping RCP: Router Algorithm on NetFPGA

- Periodic Computations:
  - Done in S/W once every T = min(10 ms, avg-rtt)
- Per-packet Computations
  - 3 Additions:
    1. Add pkt-rtt to running sum
    2. Add pkt-size to input-traffic
    3. Increment #pkts seen
  - Two Comparisons:
    1. Is pkt-rtt valid?
    2. Is RCP-rate in pkt higher than what router can offer?

Implementation: RCP End-host



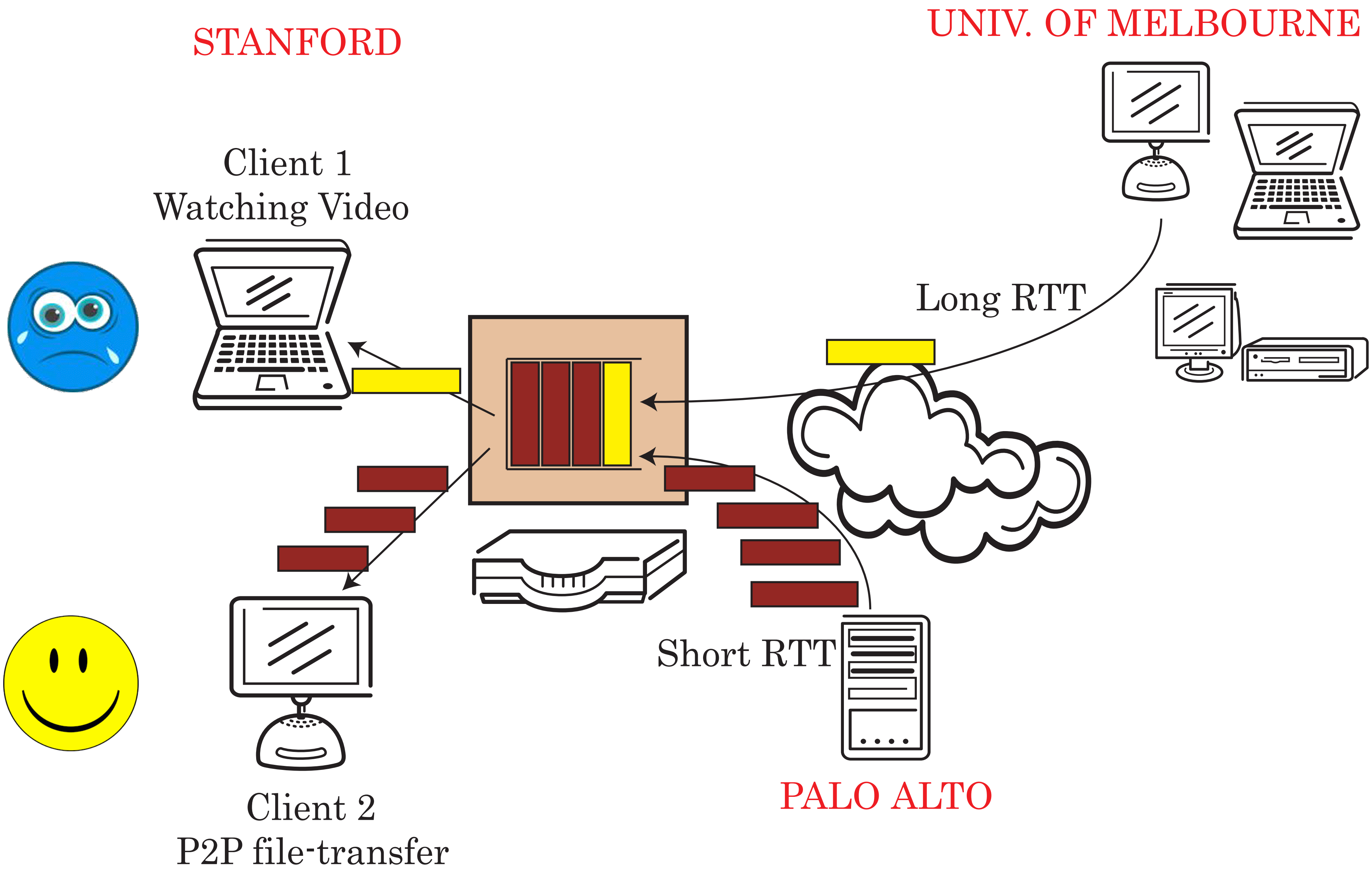
- Uses RCP rate information to modulate cwnd
- Paces packets
- Other TCP functionalities (retransmission mechanisms, state machine...) as before

0 1 2 3 ... ..14 15	16.. ..23	24.. ...30 31
rcp_bottleneck_rate		
rcp_reverse_bottleneck_rate		
rcp_rtt	rcp_p	unused

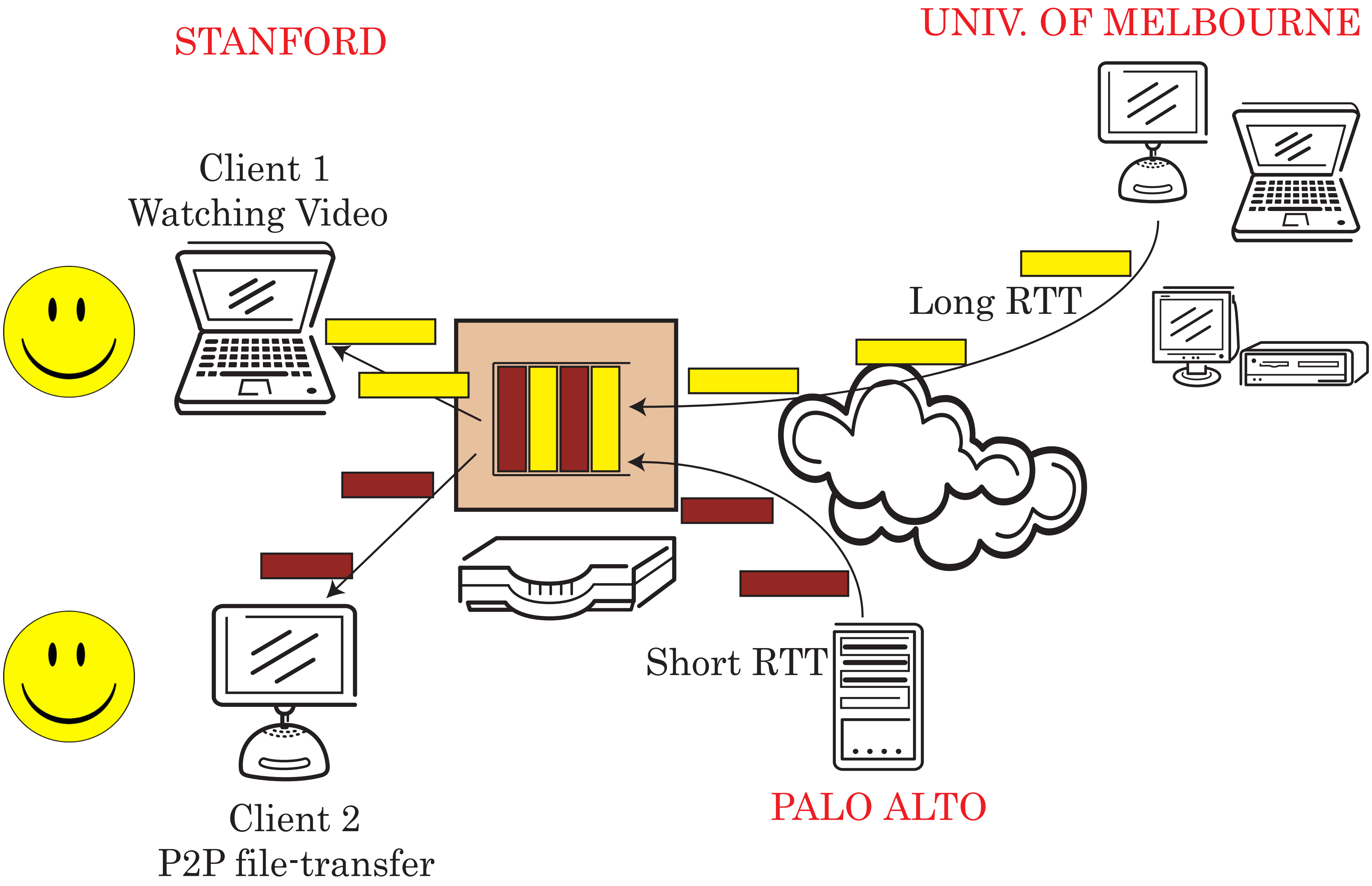
- RCP Header = 12 B
- Routers modify rcp_bottleneck_rate

IP header protocol field
carries RCP protocol number

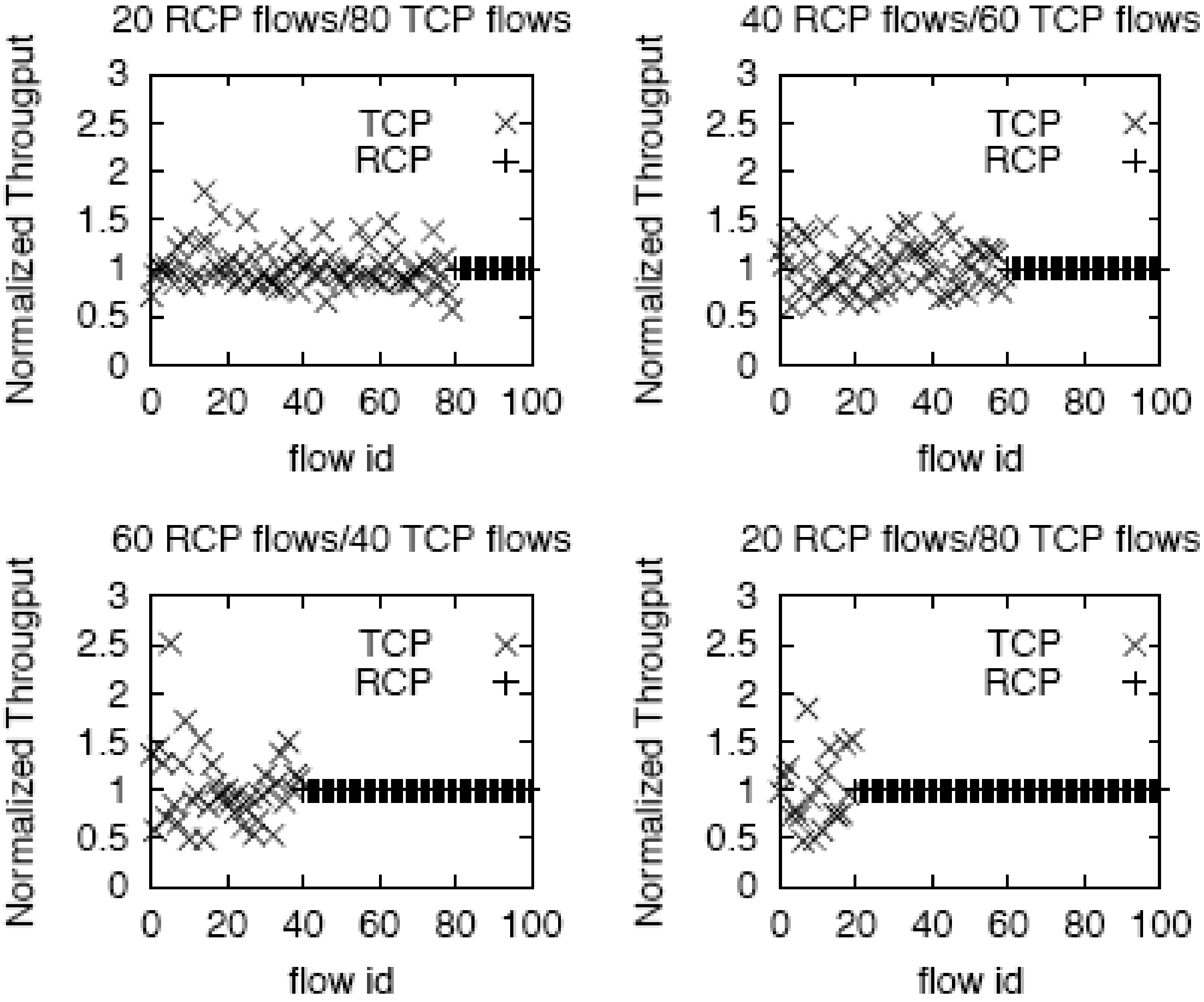
Example 3: TCP Penalizes Long RTT Flows



Example 3: Fair Resource Allocation in RCP



Bandwidth Sharing Between x% TCP Flows and (100 - x)% RCP Flows


20 RCP flows/80 TCP flows
40 RCP flows/60 TCP flows
60 RCP flows/40 TCP flows
20 RCP flows/80 TCP flows
Normalized Througput
flow id
TCP ×
RCP +