A Clean Slate Design of Internet's Congestion Control Algorithm

Nandita Dukkipati

(Collaborators: Nick McKeown, Masayoshi Kobayashi, Rui Zhang-Shen)

High Performance Networking Group
Stanford University

Hamilton Institute Workshop on Congestion Control
27 September, 2005

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. Puzzled 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

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
- Routers do detailed per-packet calculations

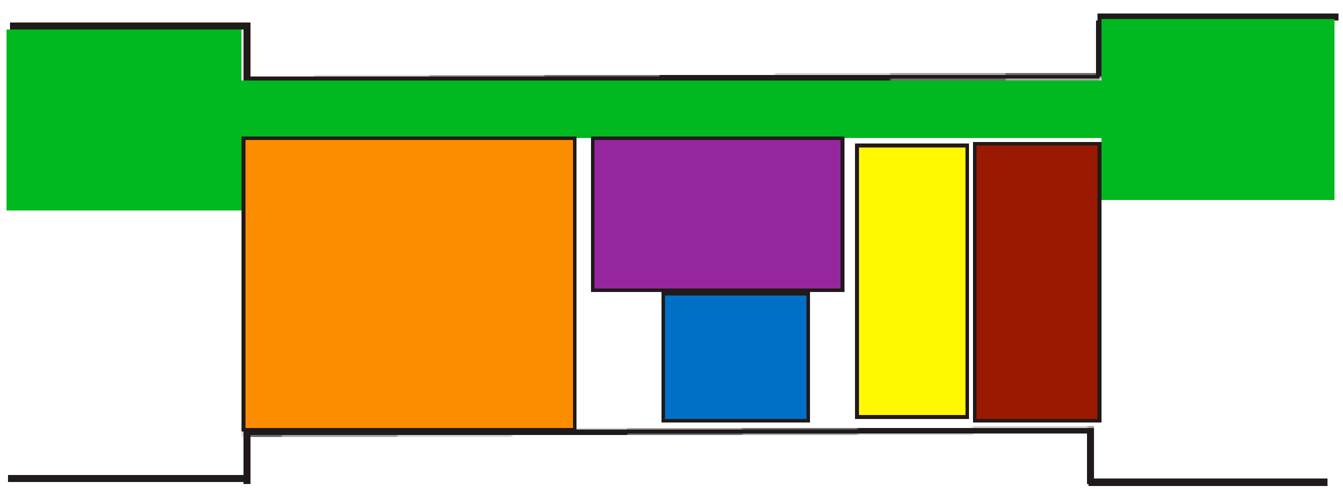
XCP -- Pros and Cons

- Pros:
  - Long-lived flows: Works very well -- convergence to fair-share rates, high link utilization, small queue occupancy, low loss.
- Cons:
  - With a mix of flow lengths: Deviates far from Processor Sharing. Unfair and inefficient.
  - Flow durations: Makes the flows last two orders of magnitude higher than necessary. Worse than TCP.
  - Complexity: Requires detailed per-packet computations

Bandwidth-limited vs. Latency-limited

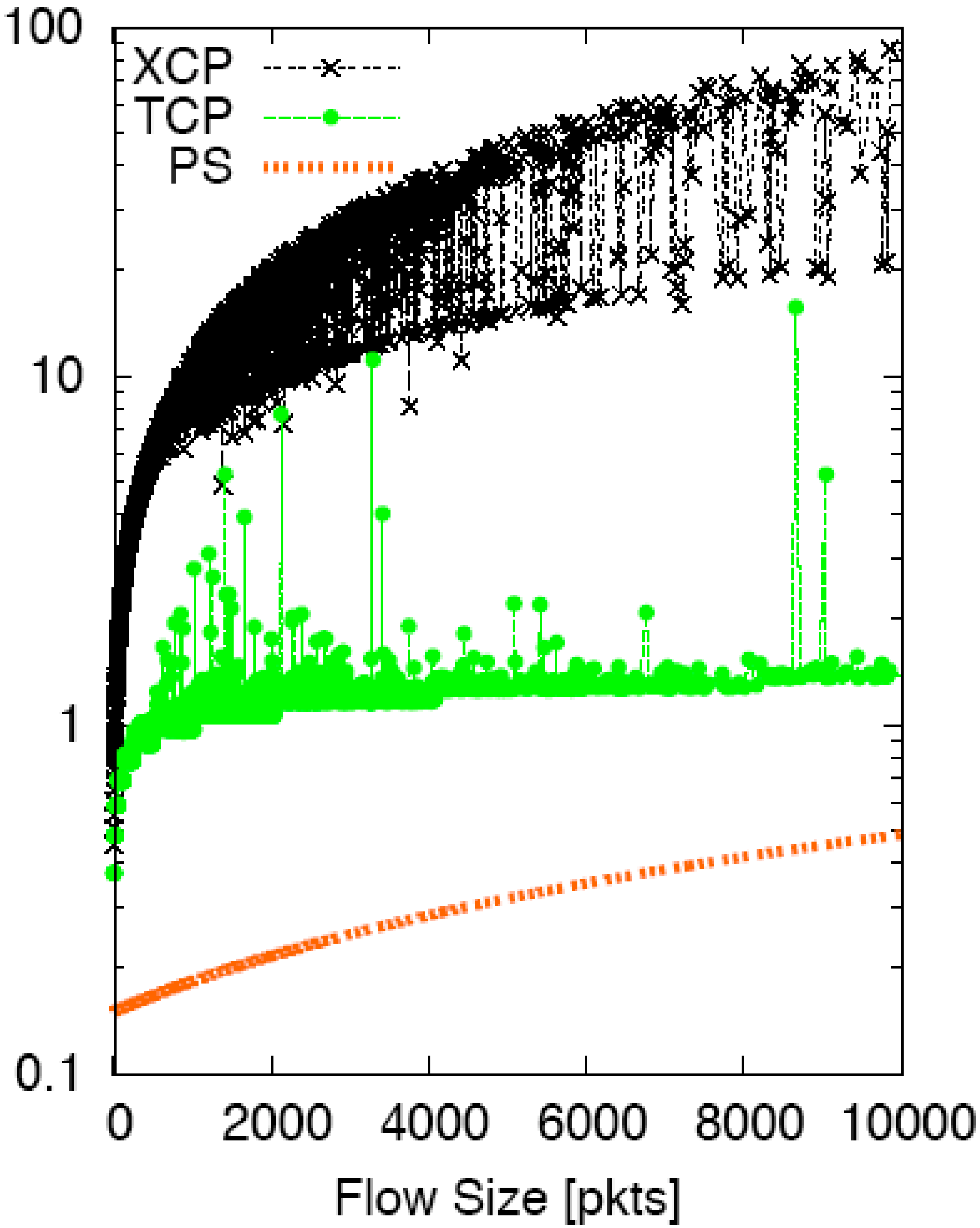
mean flow size >~ “pipe” size

mean flow size << “pipe” size

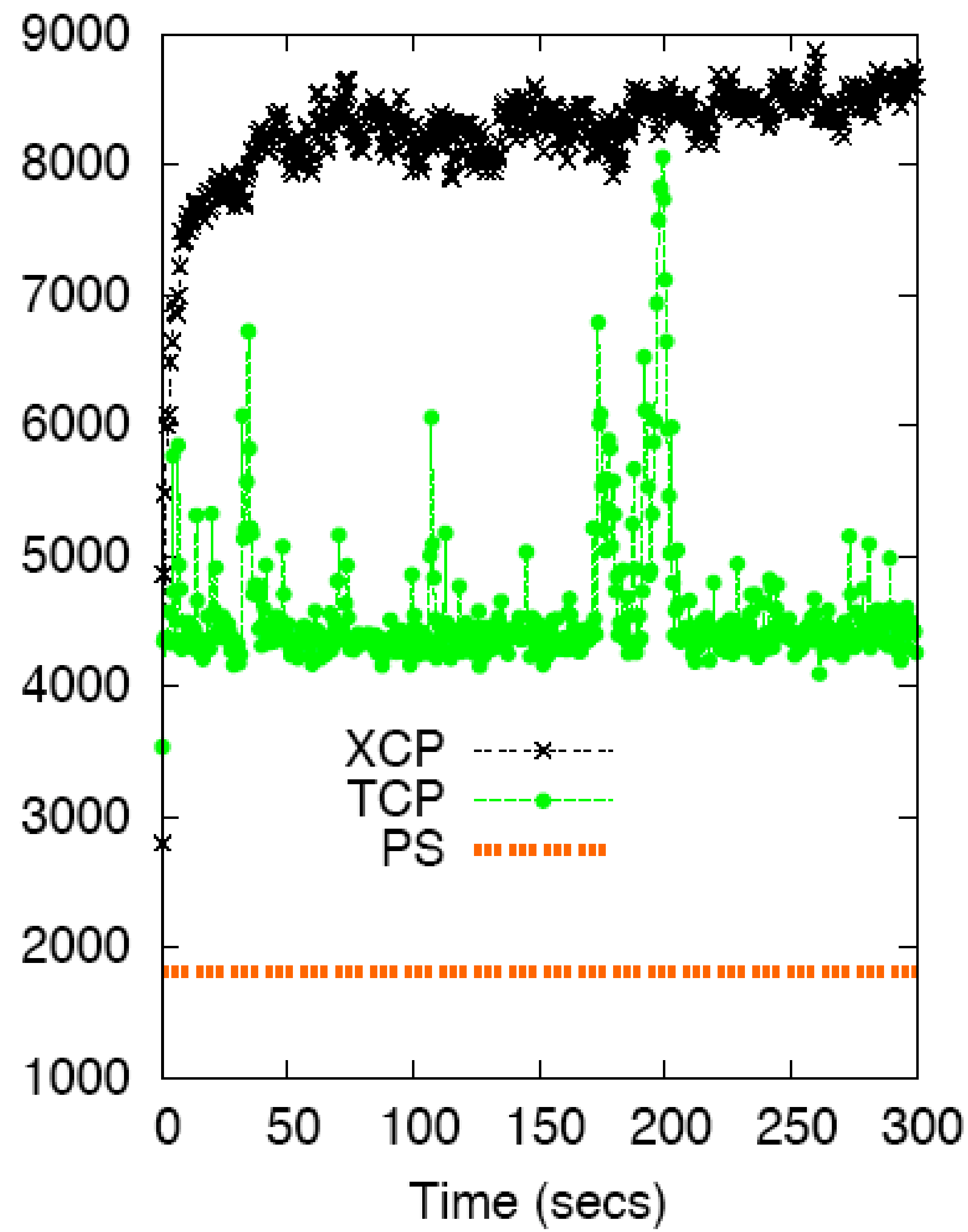


Example: XCP vs. TCP vs. PS

Flow Duration (secs) vs. Flow Size



Active Flows vs. time



Wish List

I. Emulate Processor Sharing
   1. Performance is invariant of flow size distribution
   2. Mix of flows: Results in flows finishing quickly -- close to the minimum achievable
   3. Long flows: Results in 100% link utilization -- even under high bandwidth-delay, lossy links...
   4. All flows get fair share of bottleneck bandwidth

II. Want stability -- convergence to equilibrium operating point

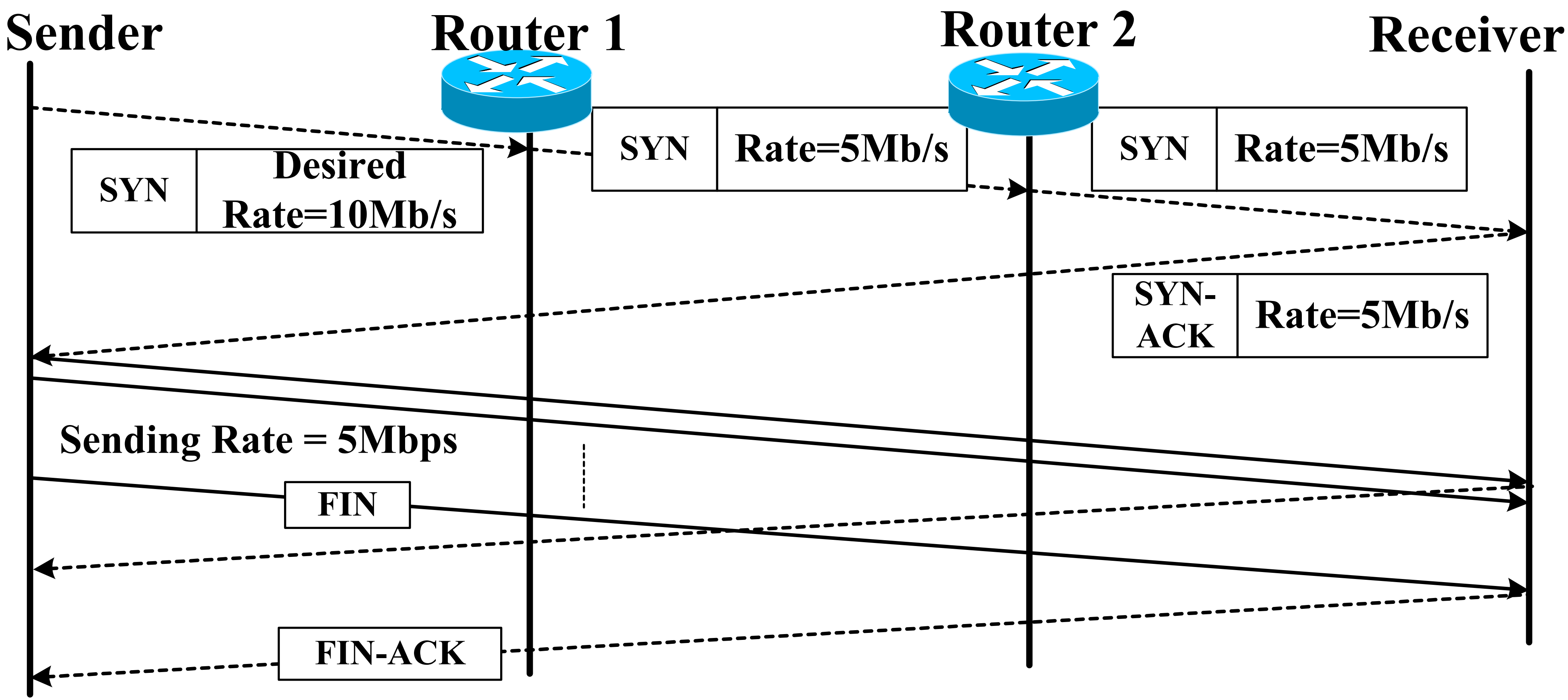
III. Want all the above under any network conditions (mix of RTTs, capacities, topologies) and flow mixes

IV. Without any per-flow state, per-flow queue or per-packet computation in the routers

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-packet calculation

RCP: The Basic Mechanism



RCP: The Algorithm

Link Capacity

Aggregate Traffic

queue

Average RTT

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

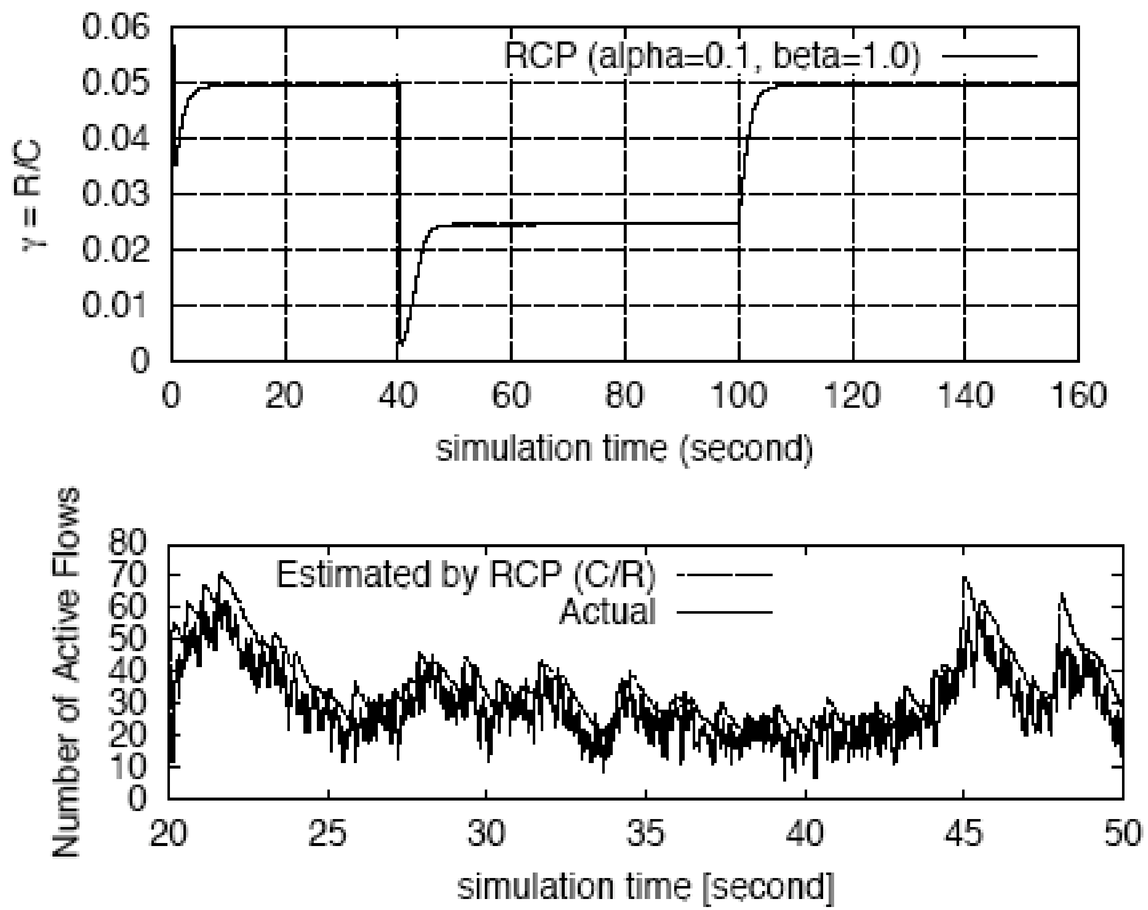
Estimate of # flows

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

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

Understanding RCP

- How good is the estimate, $C/R(t)$?



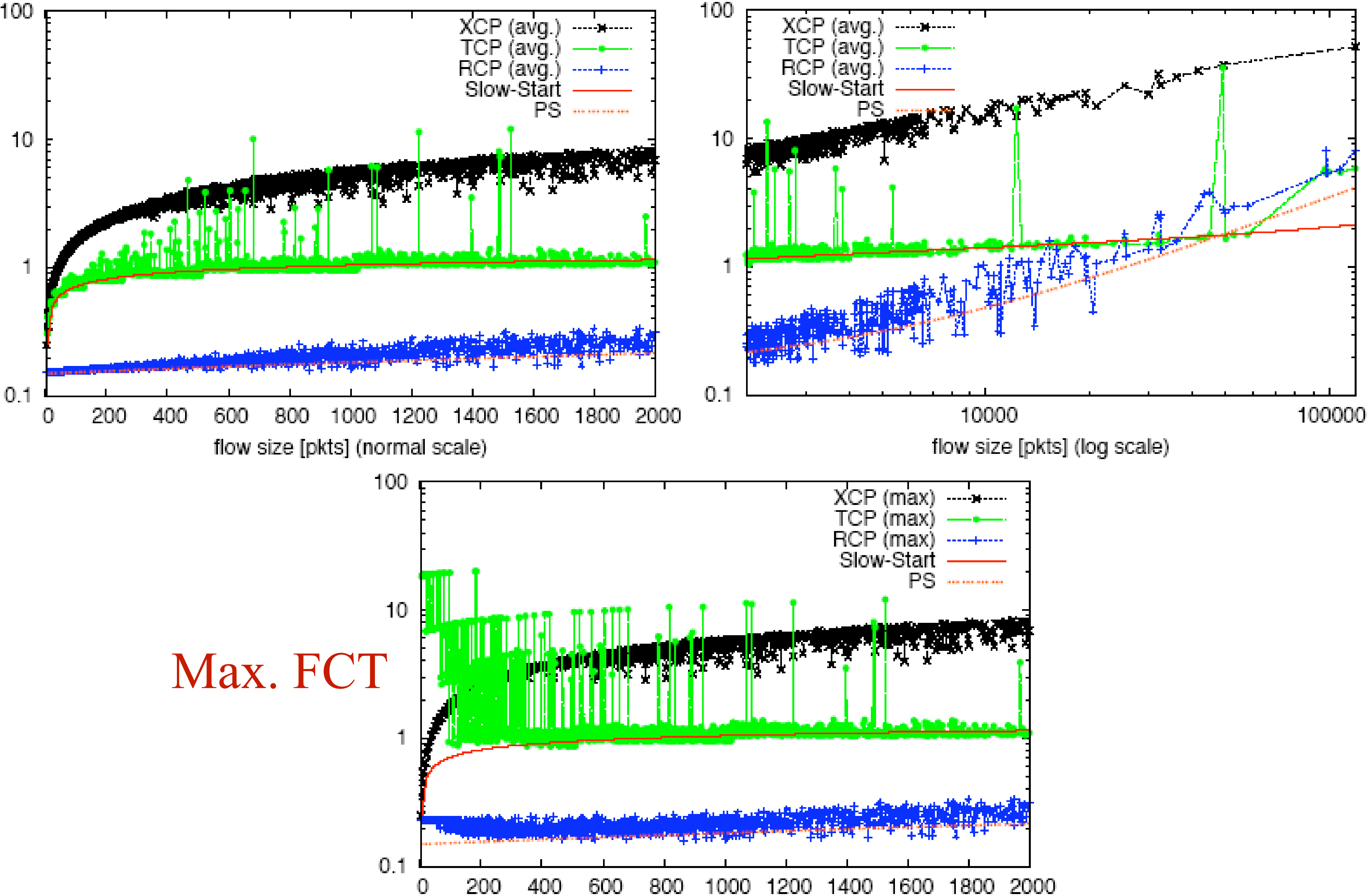
-

- RCP performs well and is stable for a broad range of it's parameters α and β

RCP Performance

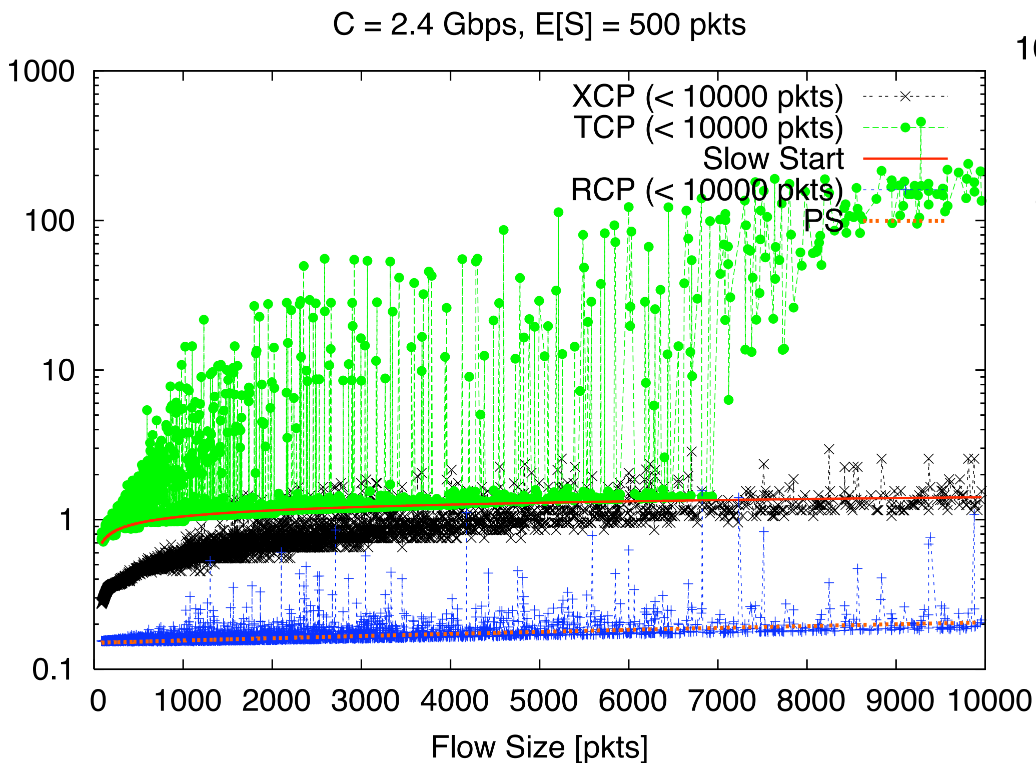
- When traffic characteristics vary
  - Different flow sizes
  - As mean flow size increases
  - Different flow size distributions
  - Non Poisson arrivals of flows
  - As load increases
- When Network Conditions vary
  - As link capacity increases
  - As RTT increases
  - Flows with different RTTs
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

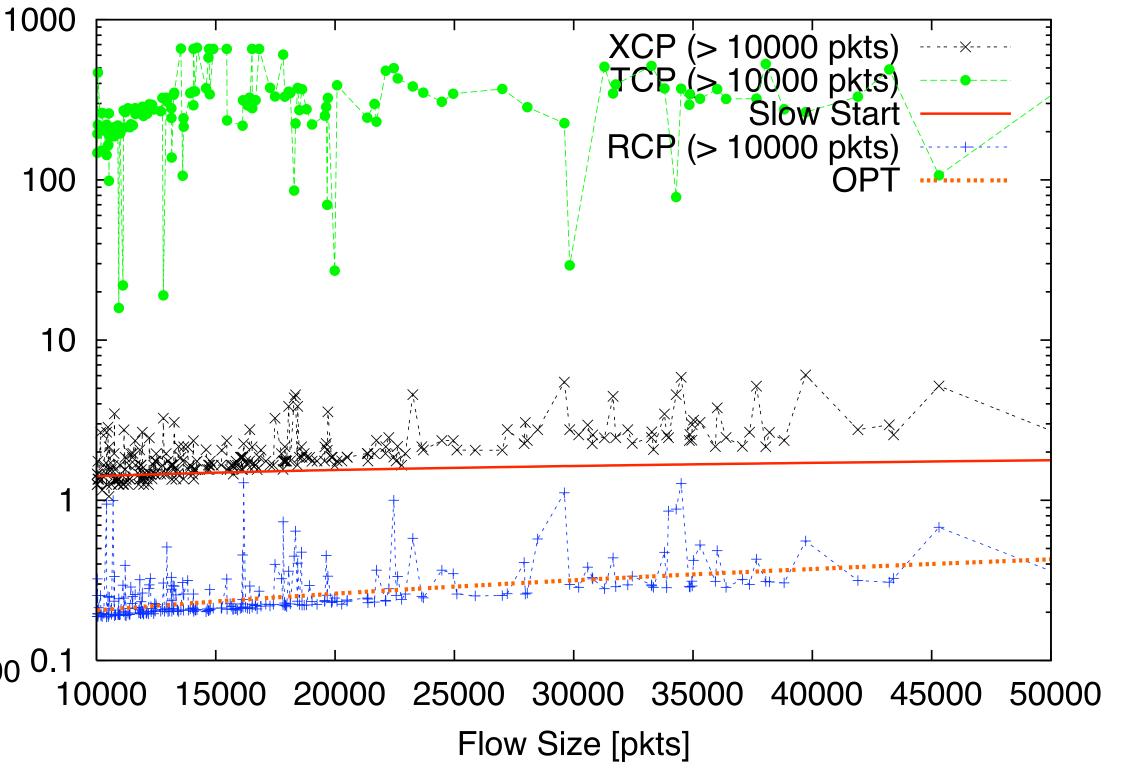
Example 1: Achieves PS for different Flow Sizes



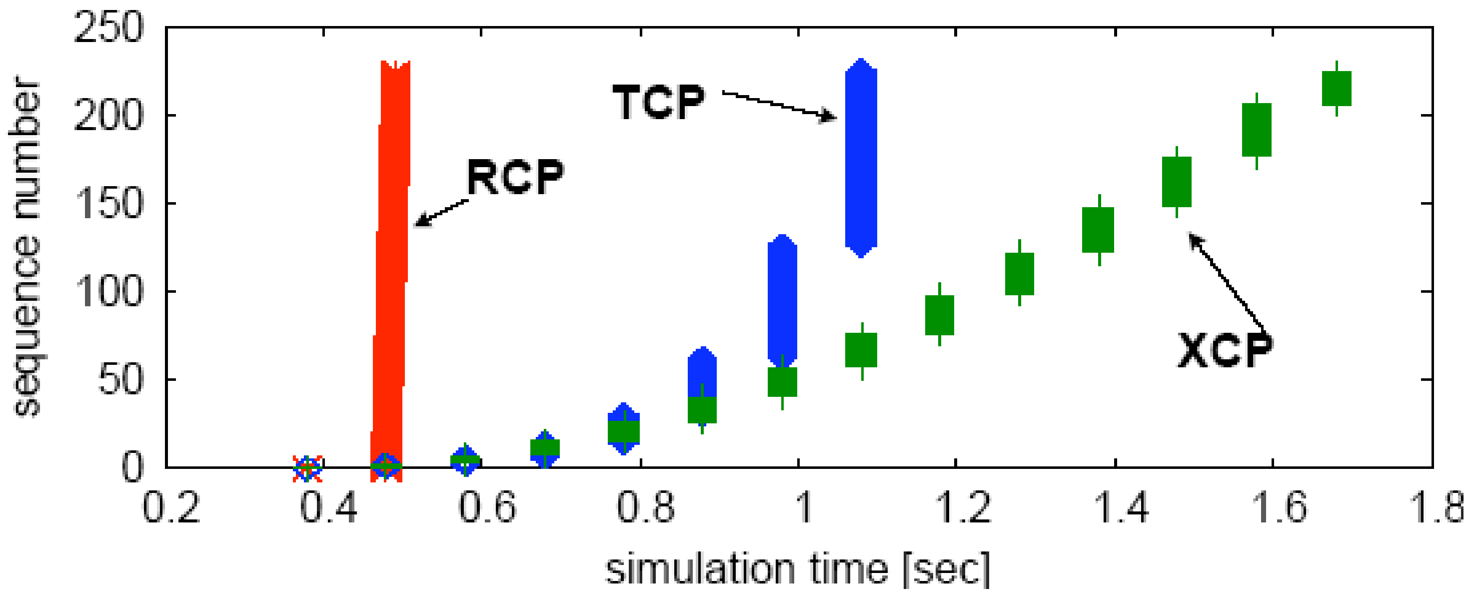
Max. FCT

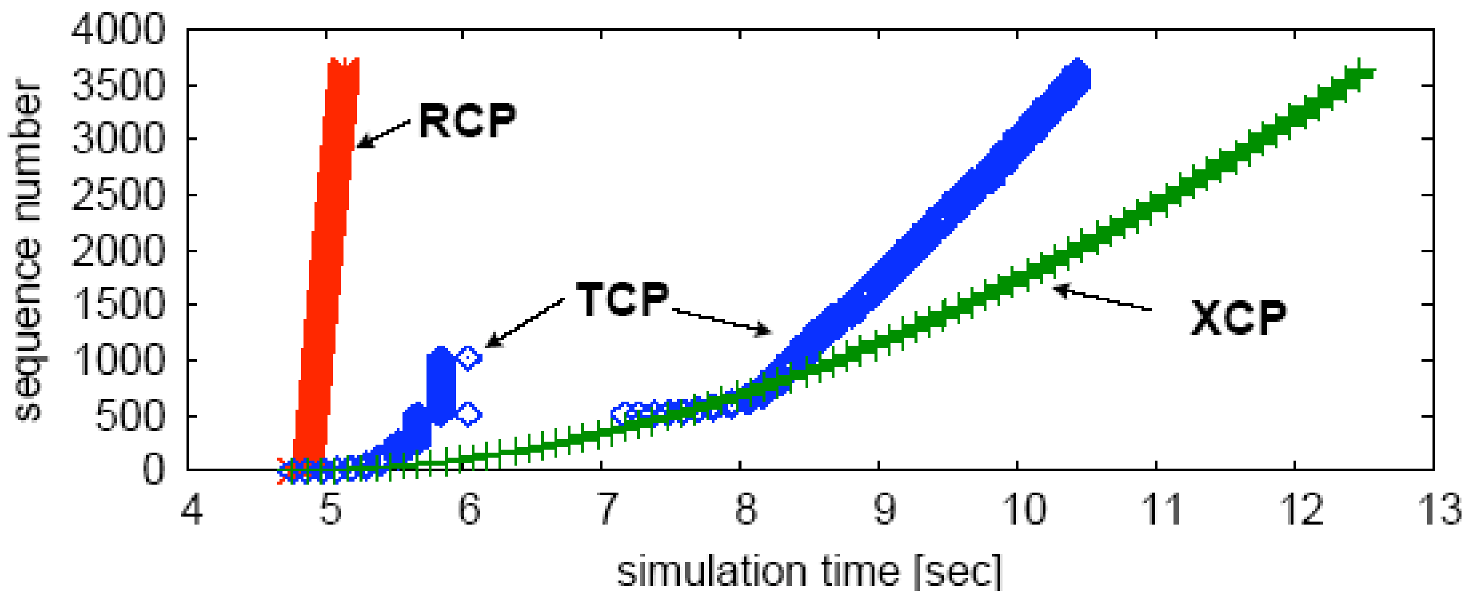
Example 2: Achieves PS for different Flow Sizes



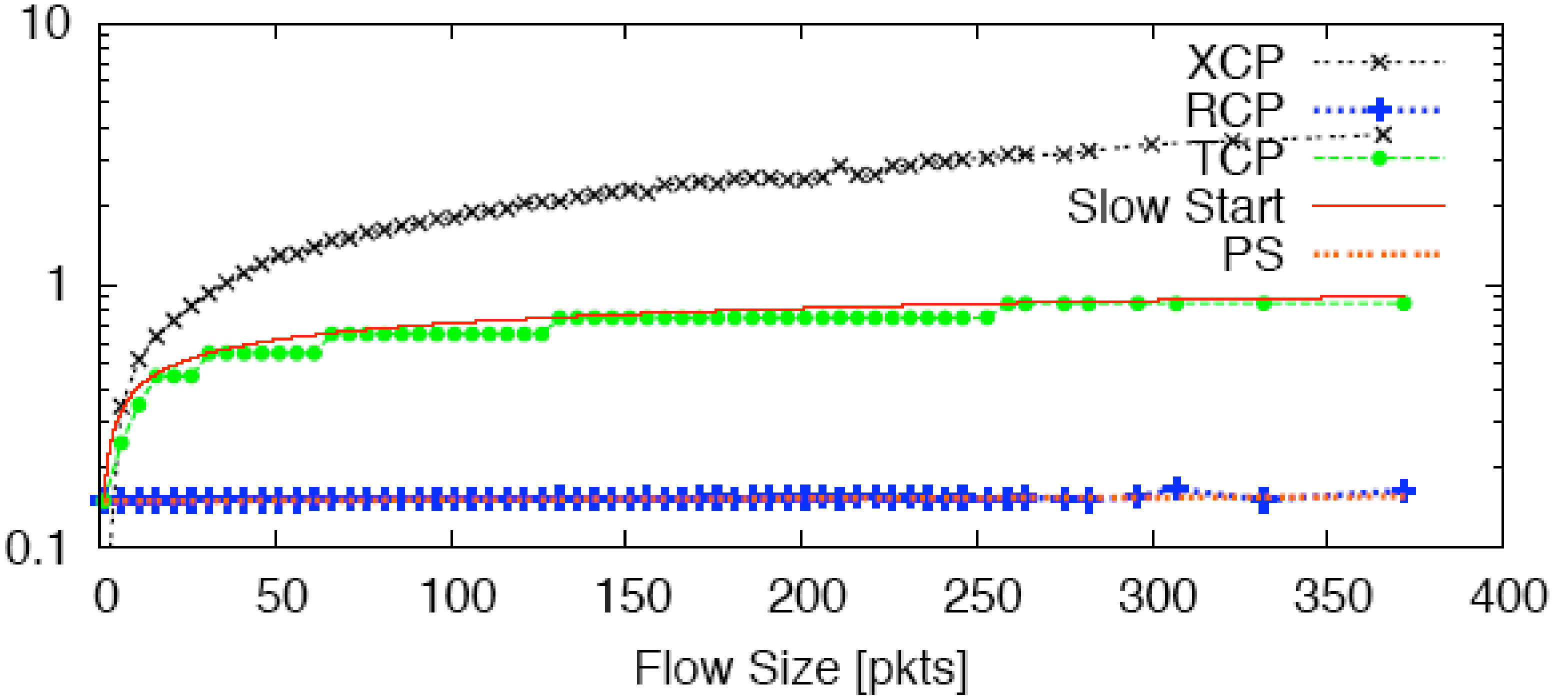


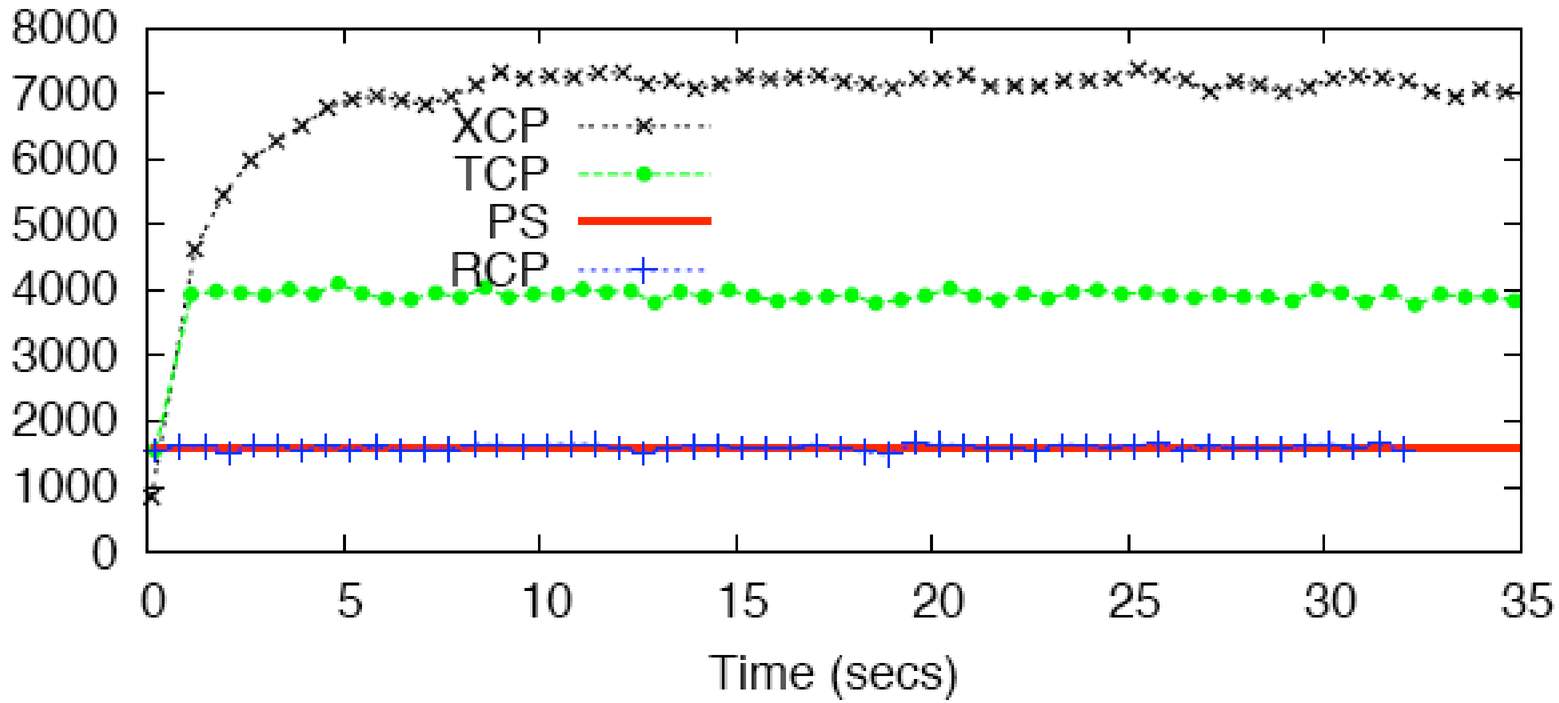
RCP vs. TCP vs. XCP





Example 3: Achieves PS for any flow size distribution





RCP Stability

RCP System:

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

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

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

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

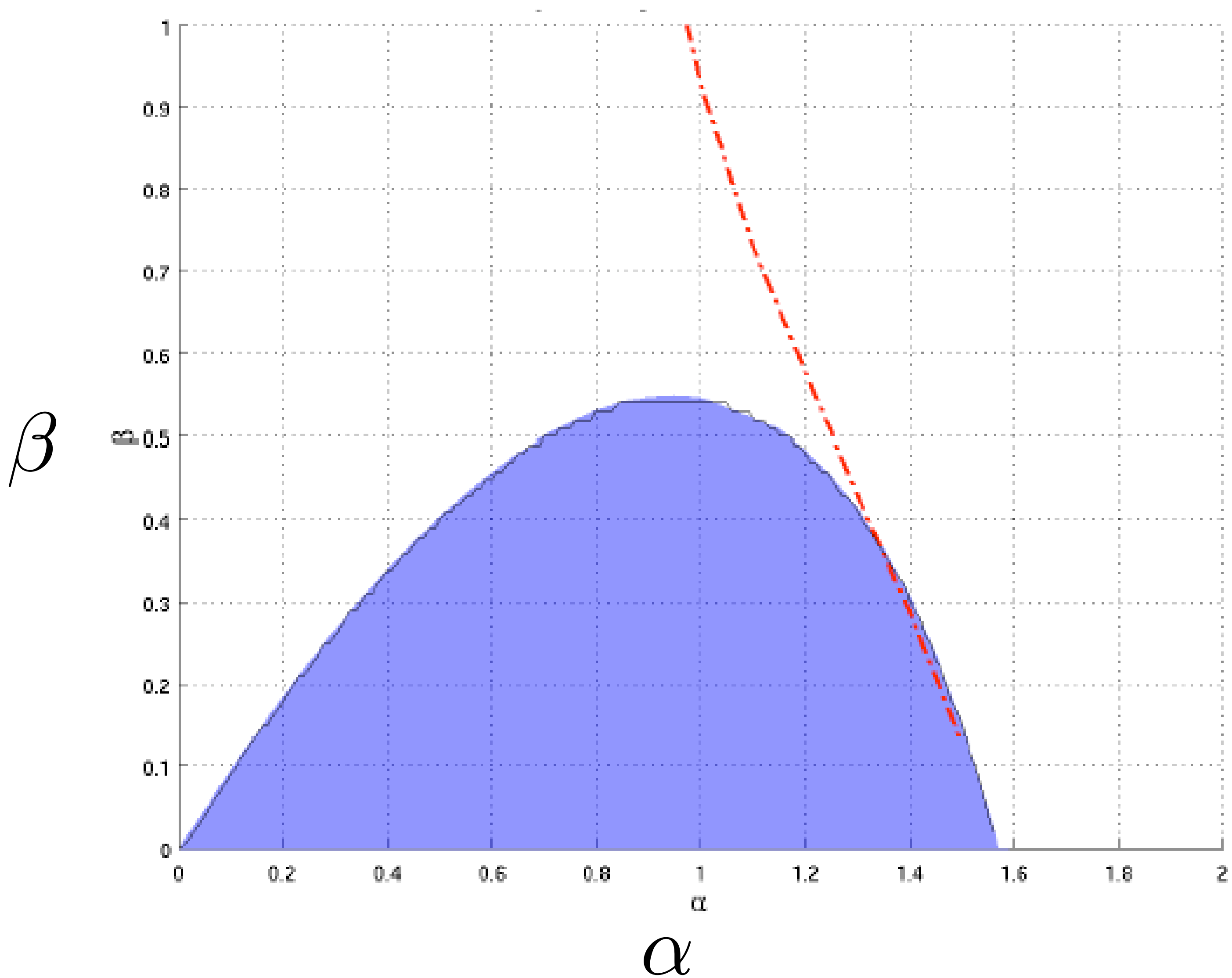
Equilibrium:

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

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

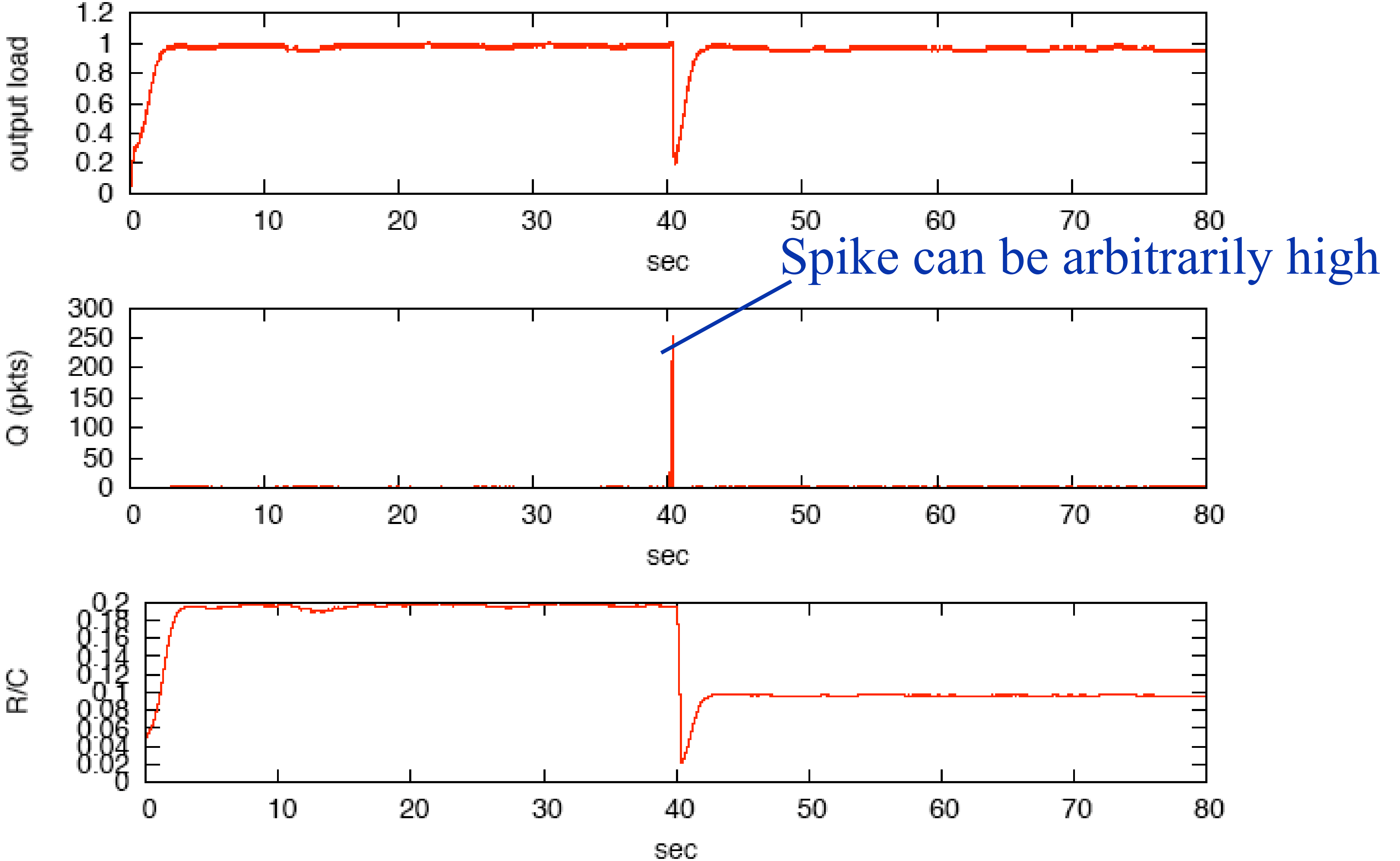
RCP is Stable

Stable Independent of C, RTT and # Flows



RCP's weakness

A lot of flows starting at once: $N \times R(t) >> C$



Intuition: Spectrum of Protocols

- RCP is aggressive --- incoming traffic could be unbounded
- Acceleration: Control how aggressively flow-rates converge to R(t)
- Protocol Spectrum:

- acceleration: small

- bandwidth-limited: works well, small queues, near-zero losses, XCP-like

- Latency-limited: long flow completion times

- acceleration: large

- bandwidth-limited: aggressive

- Latency-limited: finishes flows fast

Best of both: Adaptive Algorithm?

Conclusion

- Making network faster doesn't help; Flow durations and performance is constrained by protocols
- XCP: bold attempt in clean-slate design but there is more to do
- Network bandwidth increases => more flows capable of completing in fewer RTTs
- Metrics: Flow completion time vs. link utilization
- RCP: a simple algorithm that completes flows quickly