Congestion Control for High Bandwidth-delay Product Networks

Dina Katabi, Mark Handley, Charlie Rohrs

Presented by Chi-Yao Hong

Adapted from slides by Dina Katabi

CS598pbg Sep. 10, 2009

Trends in the Future Internet

- **High Bandwidth**
  - Gigabit Links
- **High Latency**
  - Satellite
  - Wireless

- As we will find out…these spell bad news for TCP!

What's Wrong With TCP?
As delay x bandwidth ↑

- *Oscillatory and prone to instability*
- Inefficiency due to *additive* increase
- Link capacity does not improve the transfer delay of short flows (majority)
- TCP has undesirable bias against long RTT flows (satellite links)

Efficiency and Fairness

- Efficiency of a link involves only the aggregate traffic's behavior
- Fairness is the relative throughput of flows sharing a link.
- Coupled in TCP since the same control low is used for both, uses AIMD (additive increase multiplicative decrease).

What If We Could Do It Over?

- If you could build a new congestion control architecture, what would it look like?
- Points of Observation
  - Packet loss is a poor signal of congestion
    - Dropping packets should be a congestion signal of last resort
    - Congestion is not a binary variable!
  - Aggressiveness of sources should adjust accordiing to the delay
    - As delay increases, rate change should be slower

Points of Observations (Cont'd)

- Needs to be independent of number of flows
  - Number of flows at AQM is not constant therefore it cannot be fast enough to adapt to changes
- De-coupling of efficiency and fairness
  - Done with both an efficiency controller and a fairness controller
  - Simplifies design and provides framework for differential bandwidth allocations

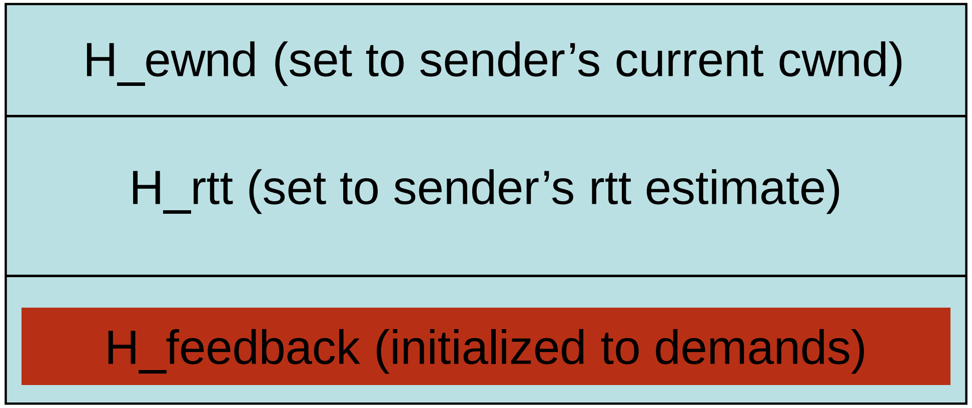
XCP

e**X**plicit **C**ontrol **P**rotocol

- Like TCP, window-based congestion control protocol intended for best effort

- Based on active congestion control and feedback as we have previously discussed

XCP Header



- H_cwnd – sender's current cong. Window
- H_rtt – sender's current RTT estimate
- H_feedback – Modified by routers along path to **directly control the congestion windows**

XCP Sender

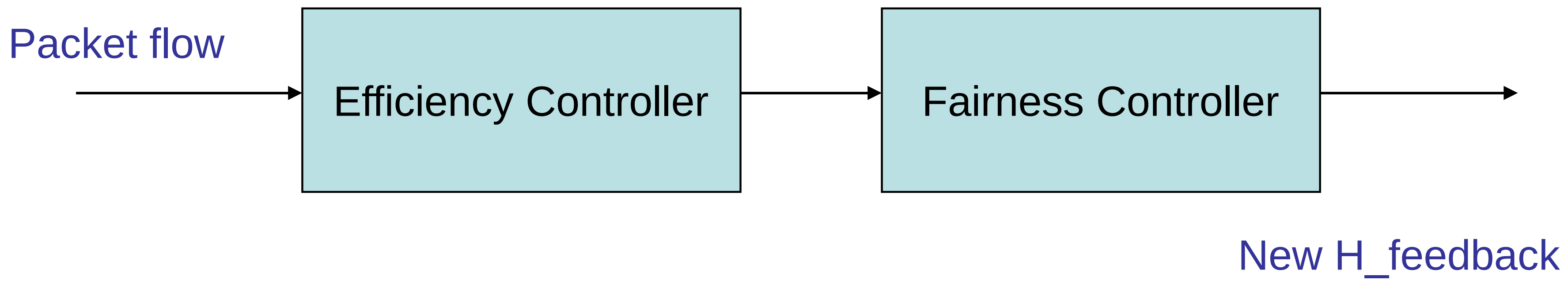
Initialization steps:

1. In first packet of flow, H_rtt is set to zero
2. H_feedback is set to the desired window increase
   - E.g. For desired rate r:
     - H_feedback = (r * rtt – cwnd) / # packets in window
3. When Acks arrive:
   - *cwnd = max(cwnd + H_feedback, s)*

XCP Receiver

- Same as TCP
- *Except* when ack'ing a packet, copies the congestion header into the ACK.

XCP Router



- Key is the use of both an *efficiency controller* (EC) and a *fairness controller (IC)*
- Both compute estimates of the RTT of the flows on each link
- Controller makes a single control decision every control interval
- Current RTT average = *d*

The Efficiency Controller

Aggregate feedback

Persistent queue size

$$\Phi = \alpha * d * S - \beta * Q$$

.226 based on stability analysis

.4 based on stability analysis

Spare BW (input traffic rate– link cap.)

From the previous page (RTT)

- Purpose – to maximize link util. while minimizing drop rate and persistent queues
- Important – Does not care about fairness
- Φ is then used as feedback to add or subtract bytes that the aggregate traffic transmits.
- Q = minimum queue seen by the arriving packet during last propagation delay (avg. RTT – local queuing delay)

The Fairness Controller

- Uses AIMD just like TCP to promote fairness
- When Φ > 0, allocate so the increase in throughput of all flows is the same
  - $\Delta throughput_i \propto \text{constant}$
- When Φ < 0, allocate so the decrease is *proportional* to its current throughput
  - $\Delta throughput_i \propto throughput_i$
- When Φ = 0, use *bandwidth shuffling*, where every average RTT, at least 10% of the traffic is redistributed according to AIMD

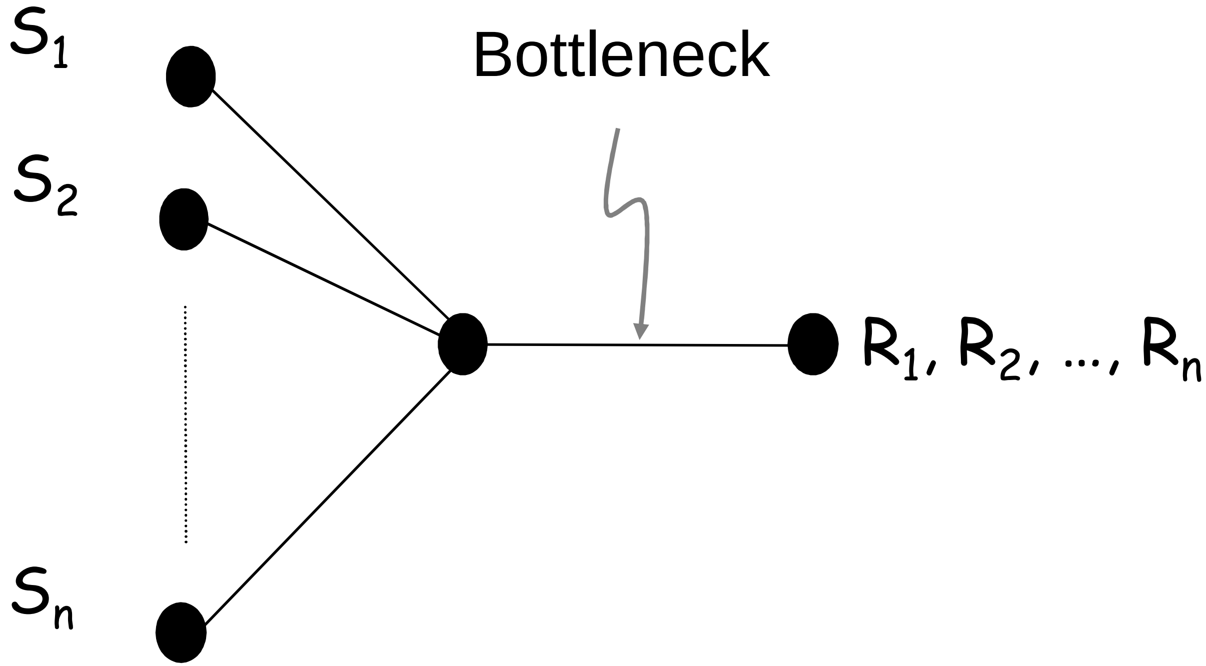
Does It Work?

- Ns-2 simulations of XCP

  vs.

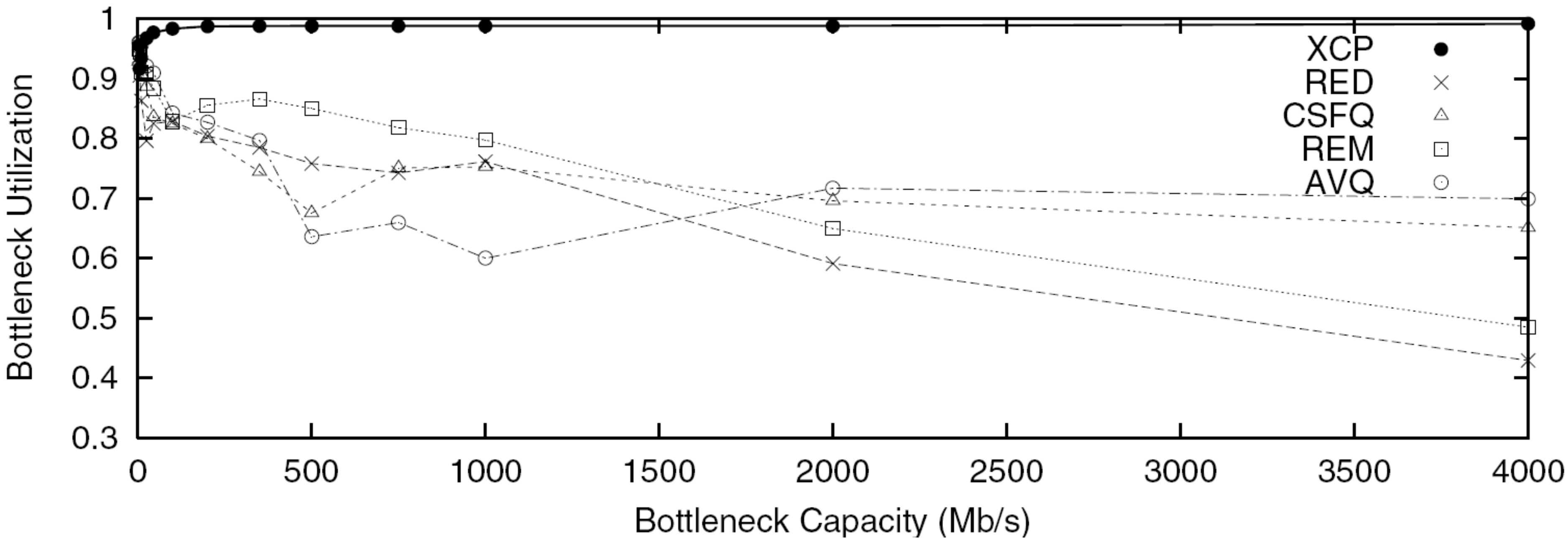
- TCP Reno +
  - **Random Early Discard (RED)**
  - **Random Early Marking (REM)**
  - **Adaptive Virtual Queue (AVQ)**
  - **Core Stateless Fair Queuing (CSFQ)**

Simulation Network



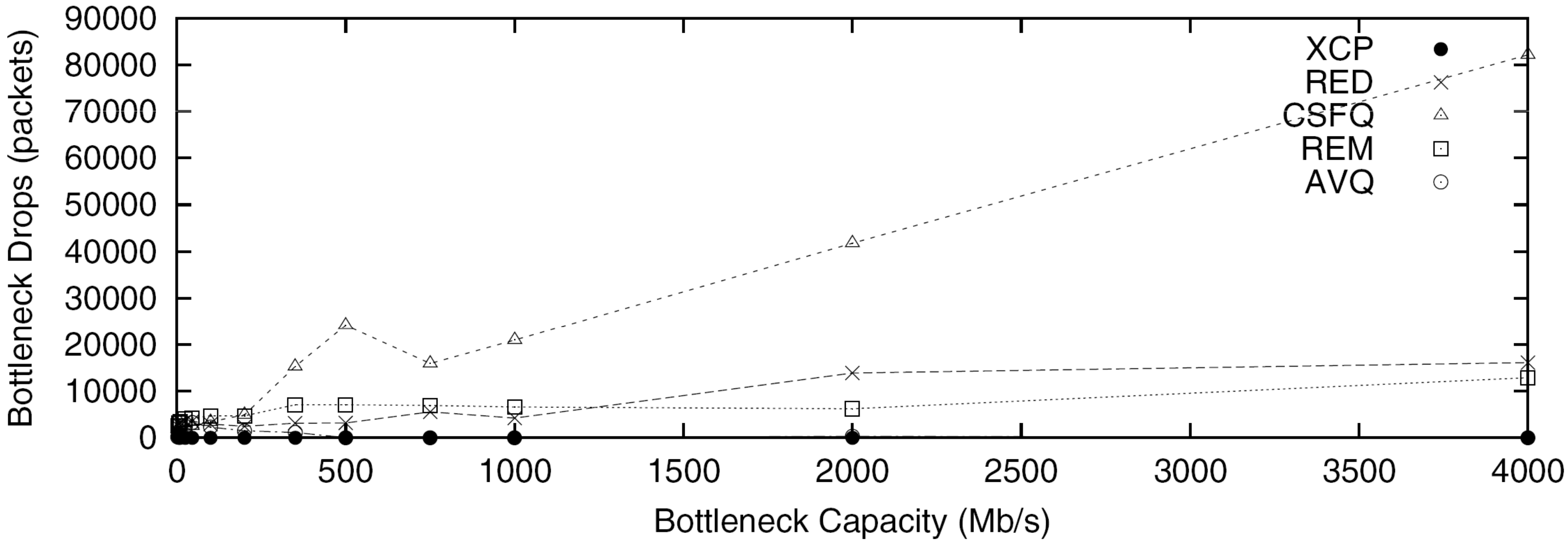
Utilization Vs. Bandwidth

- 50 long-lived TCP flows
- 80ms Prop. Delay
- 50 flows in reverse direction to create 2-way traffic
- *XCP is near optimal!*



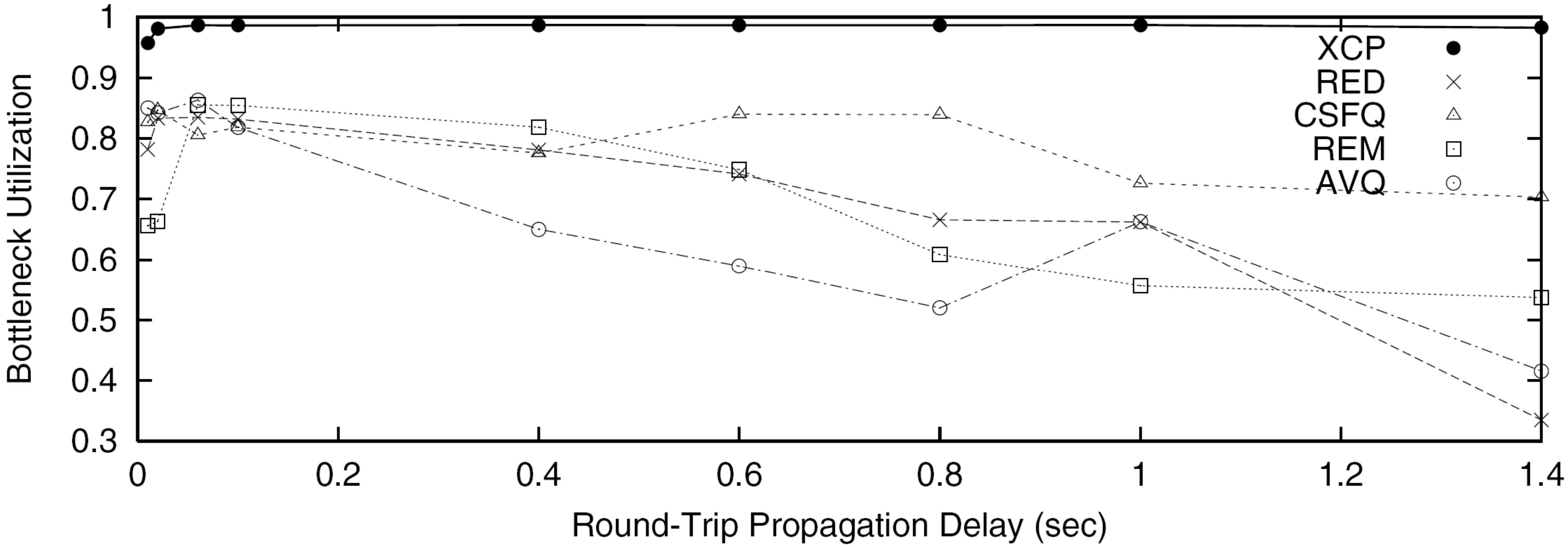
Utilization Vs. Bandwidth

- *XCP have no bottleneck drops!*



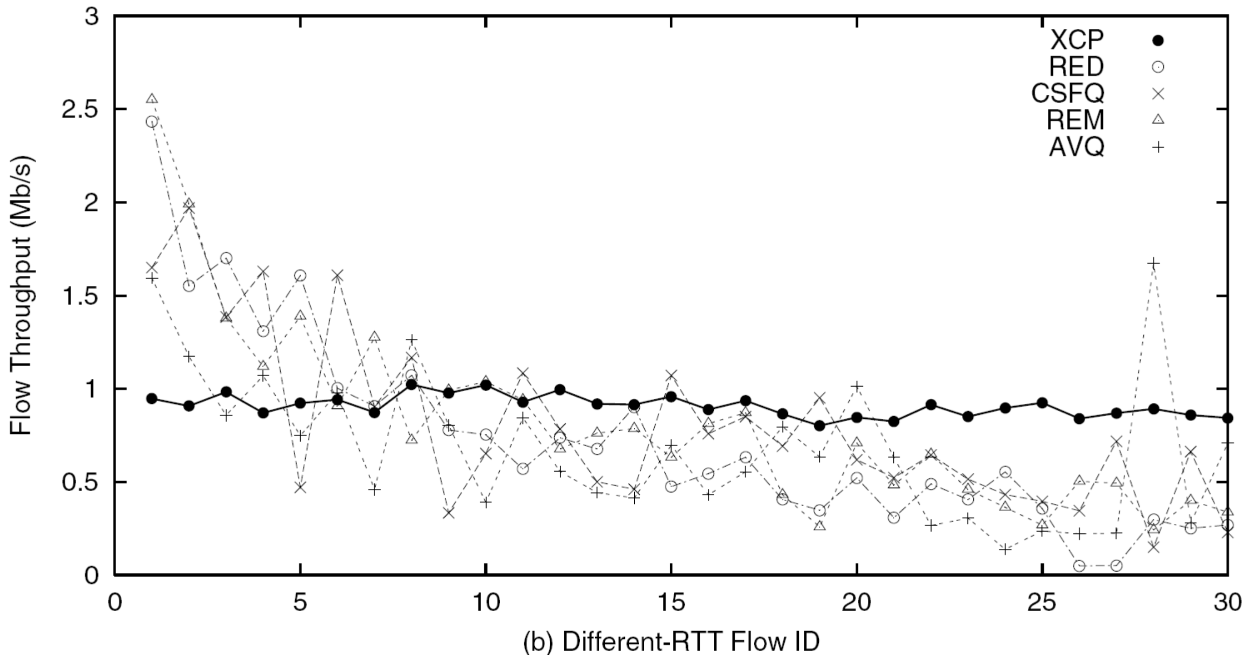
Utilization Vs. Delay

- 50 long-lived TCP flows
- 150 Mb/s Capacity
- 50 flows in reverse direction to create 2-way traffic
- XCP wins again by adjusting it's aggressiveness to round trip delay

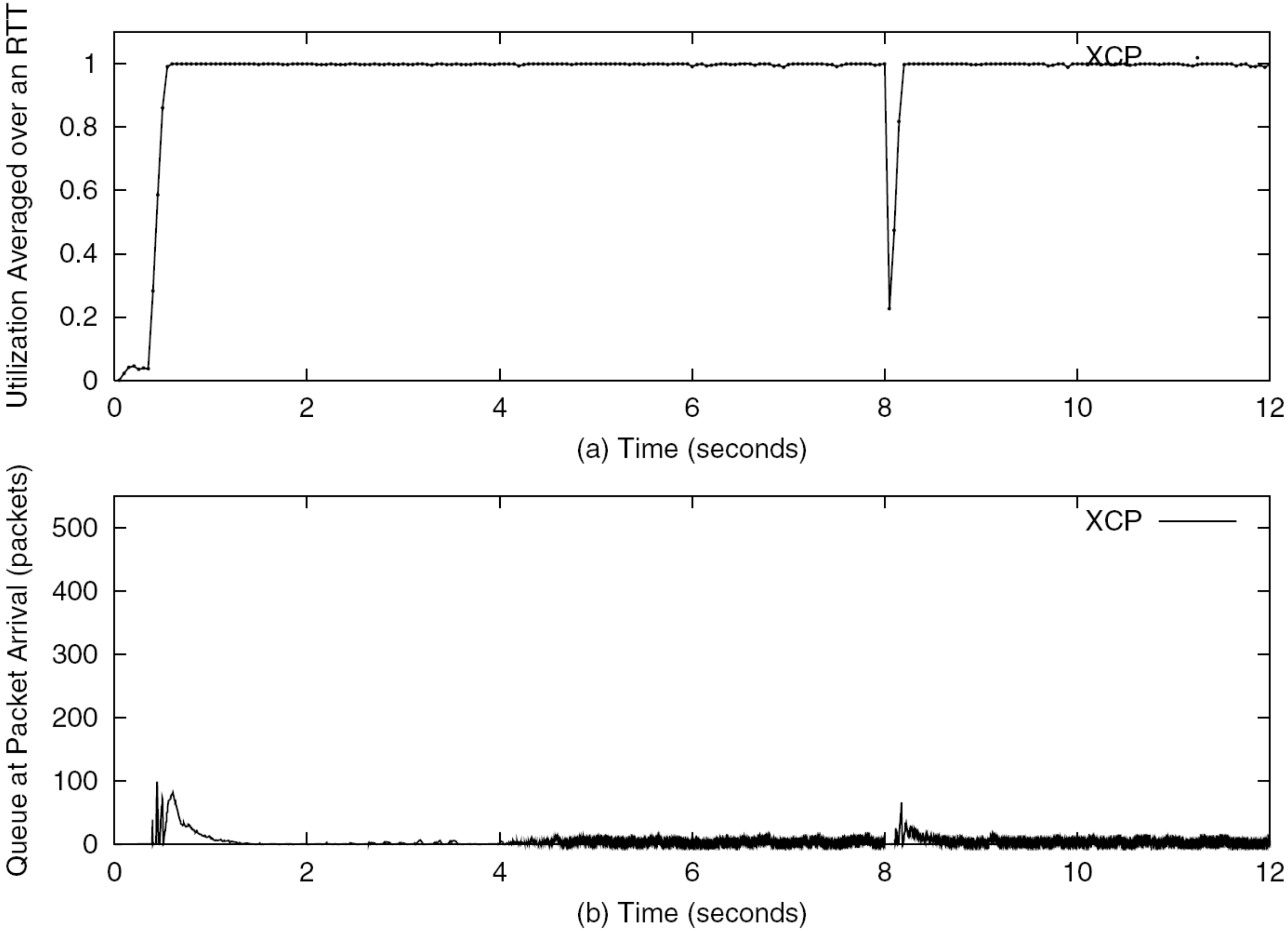


XCP Fairness

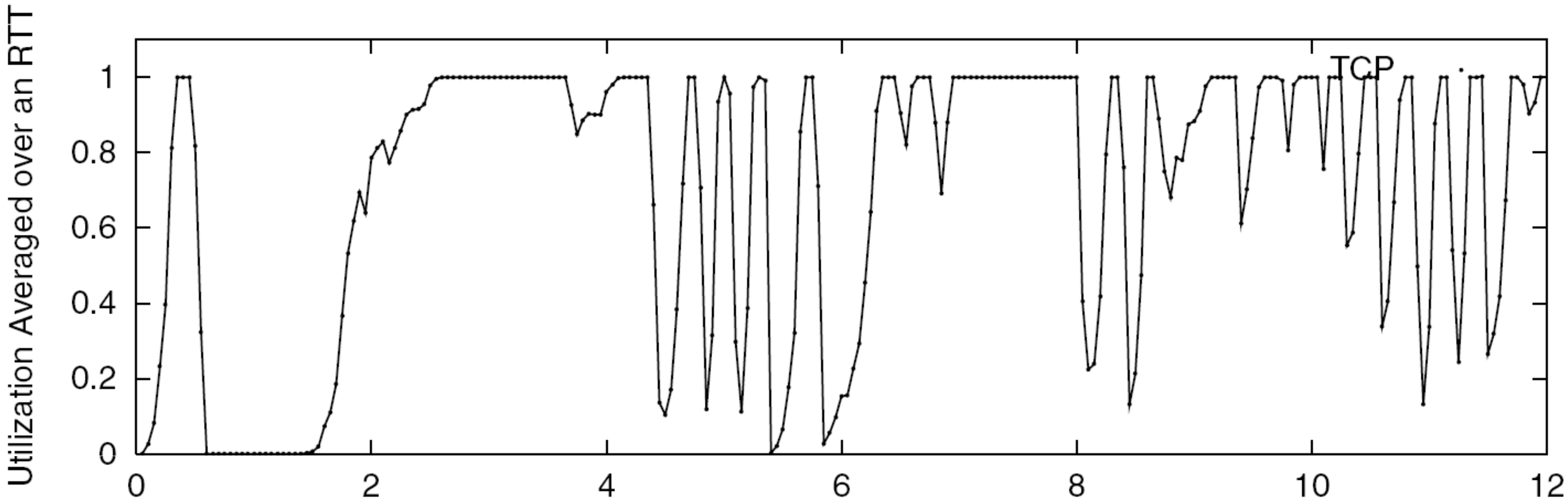
- 30 long-lived FTP flows
- Single 30 Mb/s bottleneck
- Flows are increasing in RTT from 40-330 ms
- *XCP is very fair!*



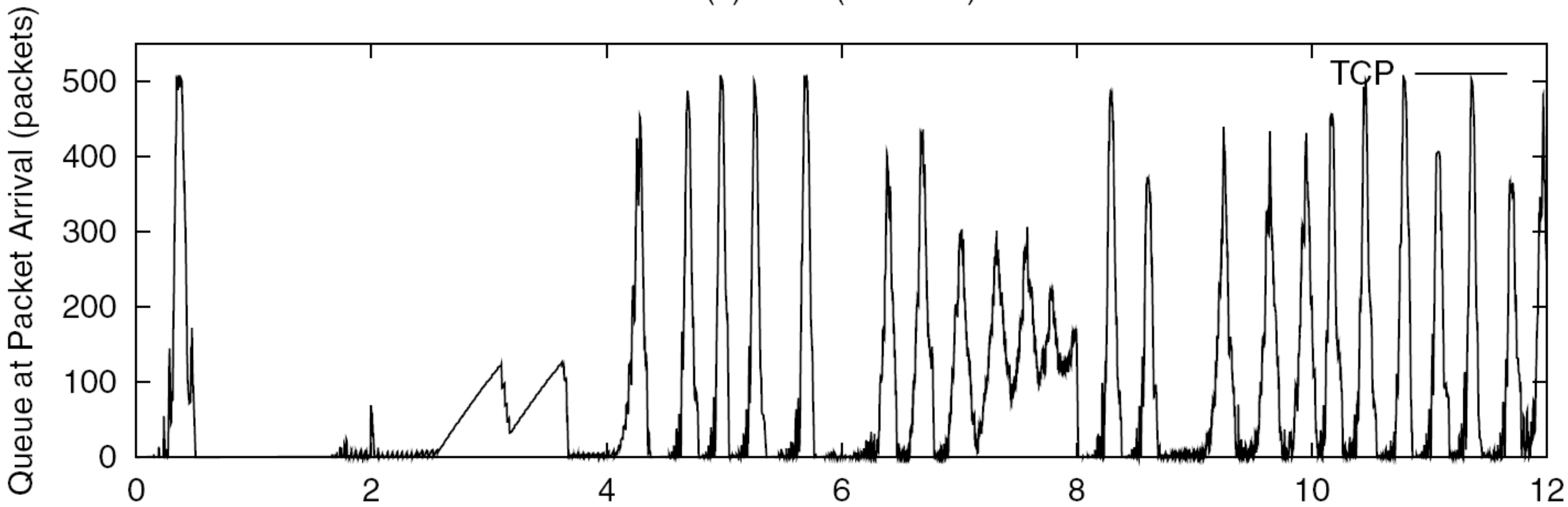
Sudden Traffic Demands?



TCP…



(c) Time (seconds)



(d) Time (seconds)

Security

- Like TCP, need an additional mechanism that polices flows
- Unlike TCP, the agent can leverage the explicit feedback to test a source
- Can test a flow by sending a test feedback requiring it to decrease it's window
- *If the flow does not react in a single RTT then it is unresponsive!*

Deployment of XCP

- Can use XCP-based CSFQ by mapping TCP or UDP into XCP flow across a network cloud
- Or can make a TCP-friendly mechanism that will allow weighing of the protocols to compete for fairness

Conclusions

- Decoupling *congestion control* from *fairness control*
- XCP can handle the high-bandwidth and delay of the future Internet
- Because of it's almost instantaneous feedback, it is a protocol that provides virtually zero drops