Congestion Control for High Bandwidth-Delay Product Networks

Dina Katabi, Mark Handley and Charlie Rohrs

SIGCOMM2002 Pittsburgh

Presenter – Bob Kinicki

Outline

- Reasons for a NEW protocol
  - Internet Trends
  - TCP Problems
- Previous Related Research
- Design Rationale
- eXplicit Control Protocol (XCP)
  - Sender, Receiver, Router
  - Stability Analysis
- XCP Performance Study Using ns-2 Simulations
- XCP Issues
- Conclusions and Critique

Internet Trends

Internet "High Speed" of 10 to 100 Mbps upgraded to current "High Speed" of 10 to 100 Gbps.

\+ Potential end-to-end delays increased due to satellite transmissions and last hop wireless retransmissions (the spread of modern RTTs has increased).

→ BDP (Bandwidth Delay Product) increased dramatically!!

Since packet drops occur over wireless links, dropping is NOT an unambiguous implicit indicator of congestion.

Problems with TCP

- TCP becomes oscillatory and prone to instability as BDP increases.
- TCP is inherently biased against flows with high RTTs (satellite links).
- AIMD in TCP responds very slowly to available high capacities.
- With majority of short web flows (TCP mice) and over-provisioned router buffers, higher available link capacity does not necessarily improve the transfer delay of mice flows.

Previous Related Work

- "Round up the usual suspects" of AQM schemes
  - 1993 RED {including ECN}
  - 1998 CSFQ*
  - 1999 SRED
  - 2001 ARED
  - 2001 REM*
  - 2001 PI Controller*
  - 2001 AVQ*
- Good performance involves parameter tuning for these schemes.

* utilize control theory with fluid flow models and feedback loops.

Design Rationale

- Packet loss is a poor signal of congestion.
  - A binary signal of ONLY presence or absence of congestion.
- Congestion signaling should indicate the degree of congestion.
- The dynamics of congestion control is abstracted as a control loop with feedback delay.

Design Rationale

- These control systems become unstable for large feedback delays (i.e., large flow RTTs).
- **How exactly should feedback depend on delay to establish system stability?**
- Robustness to congestion needs to be independent of number of flows.
- Efficient link utilization needs expressive feedback.
- Expressive feedback in ‘coupled systems’ led to per flow state (Unscalable!!).
- Solution – uncouple efficiency from fairness.

eXplicit Control Protocol (XCP)

- XCP involves a joint design of XCP end-system Hosts and XCP routers.
- XCP is a window-based congestion control protocol intended for best effort traffic (namely, it does not involve different QoS metrics).
- Sources use cwnd, congestion window, similar to TCP.
- Routers interact with flows and provide **explicit** feedback to source hosts.

XCP Congestion Header

Sending Host fills

H_cwnd
H_rtt
H_feedback

Routers Update

H_cwnd :: sender's current congestion window (cwnd)

H_rtt :: sender's current rtt estimate

H_feedback:: Initialized to desired increase in cwnd.

Modified by routers along path to directly control senders' congestion windows.

[Dion 03]

XCP Sender

- Maintains a congestion window of outstanding packets **(cwnd)** and its own estimate of round trip time **(rtt)*.**

Initialization steps:

1. In first packet of flow, H_rtt set to zero.
2. H_feedback is set to the desired window increase.

   For a desired rate r:

   H_feedback = (r * rtt – cwnd) / # packets in current congestion window

- When ACKs arrive, positive feedback increases cwnd and negative feedback reduces cwnd:

cwnd = max(cwnd + H_feedback, s)

where s is packet size.

XCP must also respond to packet losses {although they are rare}.

* Note – rtt and RTT are different in Katabi notation!!

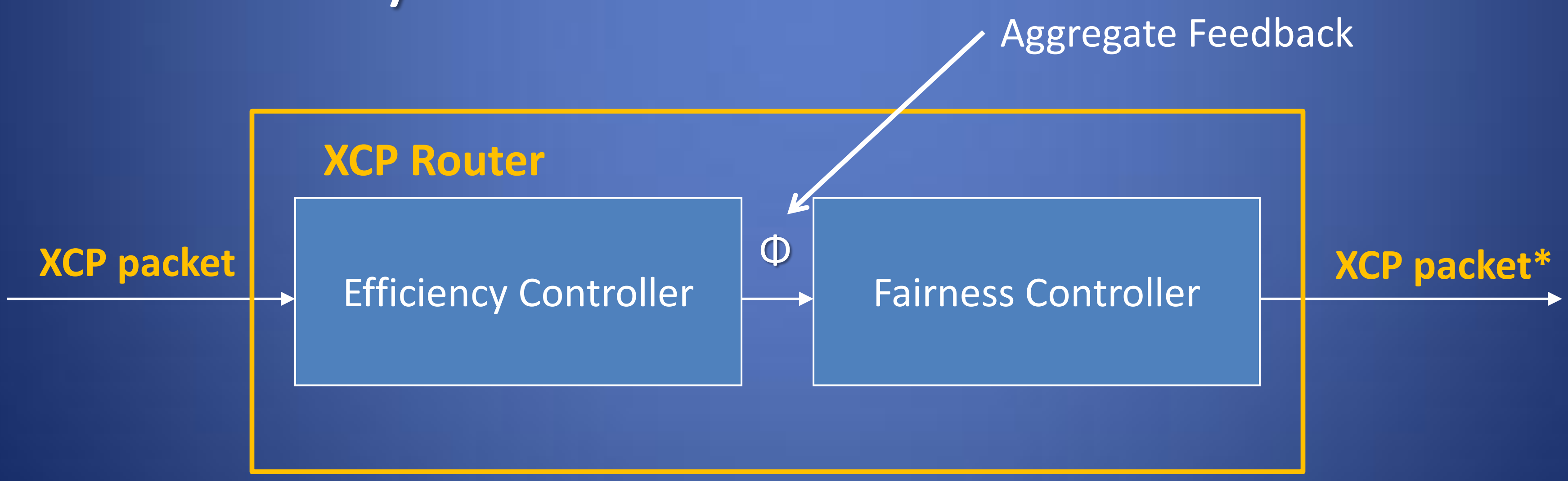
[Dion 03]

XCP Receiver

- XCP Receiver is similar to a TCP Receiver.
- When XCP Receiver ACKs a packet, it copies received congestion header from data packet into the ACK packet.

XCP Router

- XCP router operates on top of dropping policy (e.g., DropTail or RED) and computes feedback such that system converges to optimal efficiency and min-max fairness.



* modified H_feedback

[Dion 03]

XCP Router

- Both XCP controllers make a single control decision per control interval.
- **d** (the average RTT) :: the XCP control interval is computed using information in the congestion header.
- XCP router maintains a per link estimation-control timer that is set to d.
- Upon timeout, router updates its estimates and control decisions.

The Efficiency Controller (EC)

persistent queue size

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

0.226 based on stability analysis

0.4 based on stability analysis

spare capacity (input traffic rate – link capacity)

average RTT (feedback delay)

- EC maximizes link utilization while minimizing drop rate and persistent queues. This MIMD algorithm increases the traffic rate proportionally to the spare capacity.
- EC does not care about fairness (does not need flow id).
- Φ :: aggregate feedback computed once each control interval 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).

[Dion 03]

The Fairness Controller (FC)

- FC apportions the aggregate feedback to individual packets to achieve fairness.
- Uses AIMD algorithm to promote fairness.
- When $\Phi > 0$, allocate so the increase in throughput of all flows is the same.
- When $\Phi < 0$, allocate so the decrease in a flow's throughput is *proportional* to its current throughput.
- When $\Phi = 0$, uses ***bandwidth shuffling*** to prevent convergence stalling.

Bandwidth Shuffling

- Bandwidth Shuffling :: simultaneous allocation and deallocation of flow sending rate such that the total traffic rate does not change, yet the throughput of each individual flow gradually approaches its fair share.
- The shuffled traffic is computed as:

$$h = \max(0, \gamma * y - |\Phi|)$$

where y is the input traffic during **d** and **γ** is set to 0.1 {This implies that 10% of the traffic is redistributed according to AIMD.}

Per-Packet Feedback

- FC computes per-packet feedback:

$$H\_feedback_i = p_i - n_i \qquad (3)$$

Basic Idea

- p_i (the per-packet positive feedback (when $\Phi > 0$)) is proportional to the square of the i^{th} flow's rtt and inversely proportional to its congestion window divided by its packet size.
- n_i (the per-packet negative feedback (when $\Phi < 0$)) should be proportional to its packet size (s_i) and the i^{th} flow's rtt .

Proportional constants ξ_p and ξ_n are estimated every d and used during the following control interval.

Stability Analysis

Theorem 1. *Suppose the round trip delay is d. If the parameters α and β satisfy:*

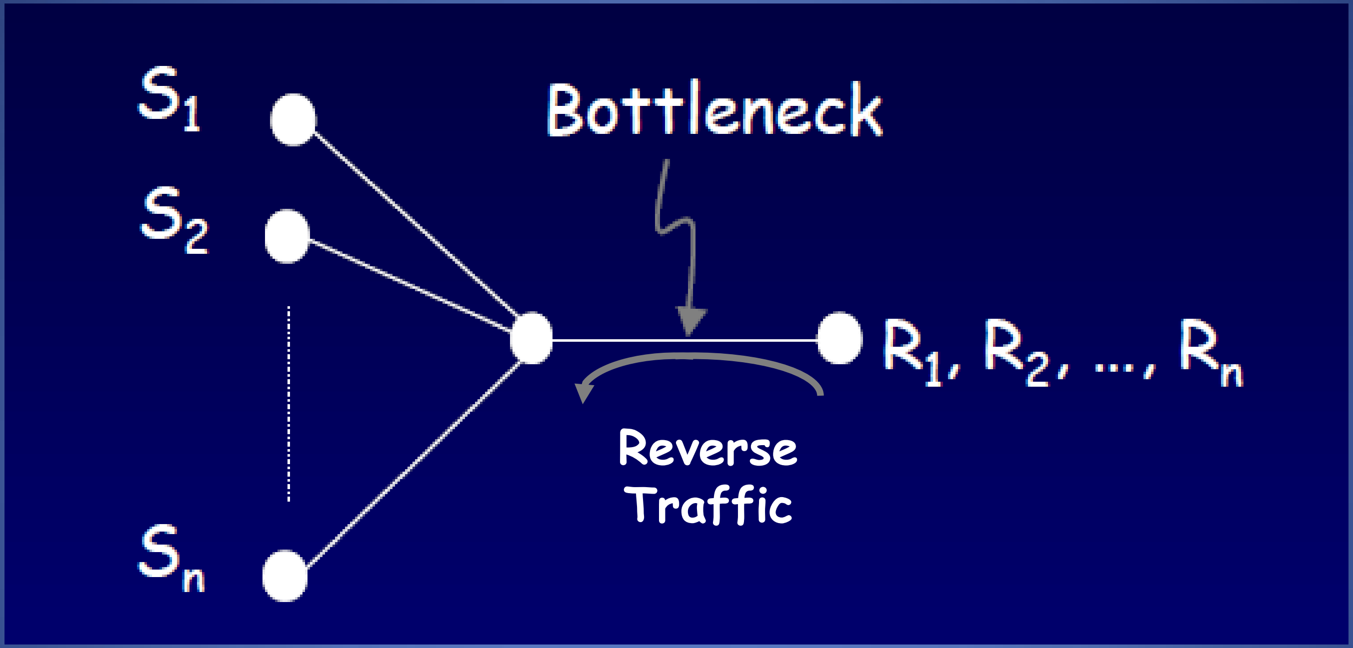
$$0 < \alpha < \frac{\pi}{4\sqrt{2}} \quad \text{and} \quad \beta = \alpha^2\sqrt{2}$$

Then the system is stable (independent of delay, capacity and number of flows).

XCP Performance

- Authors study XCP performance via an extensive series of ns-2 simulations.
- They compare XCP against the 'usual AQM suspects'.
- Simulation results substantiate the stability analysis claims of independence of XCP with respect to capacity, feedback delay and number of flows.

Single Bottleneck Topology



ns-2 simulation details

[Katabi 02]

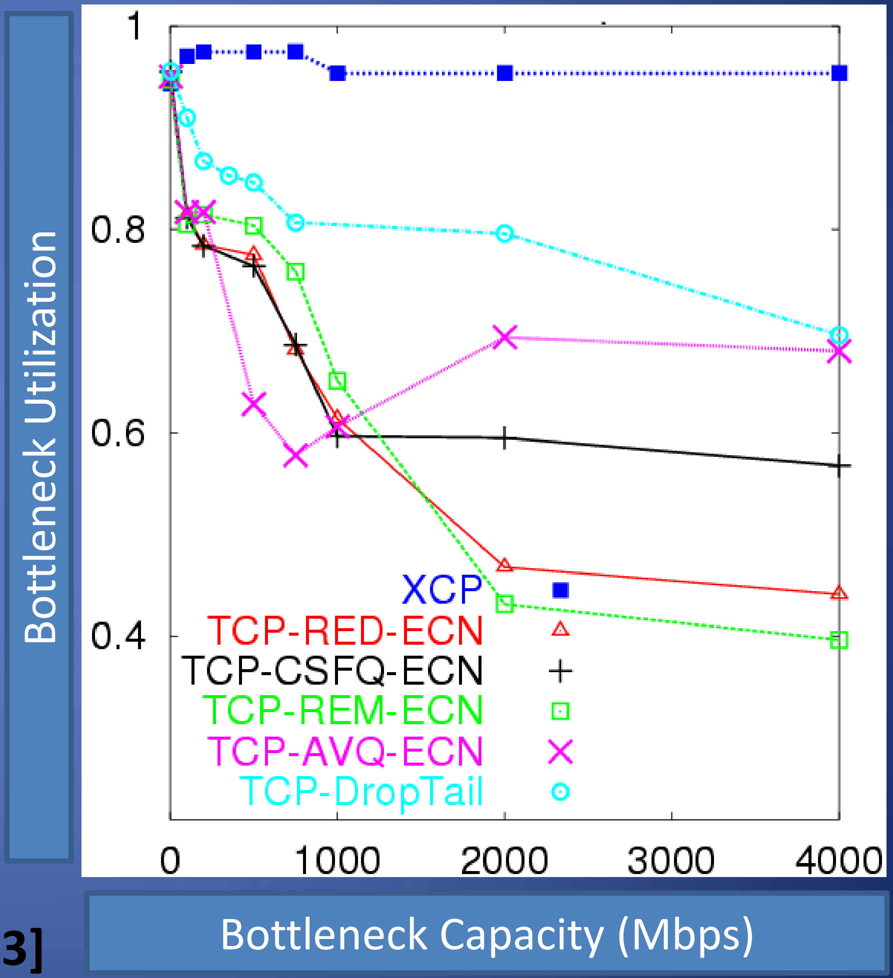
Packet size = 1000 bytes; buffer = BDP;

FTP flows are homogeneous with equivalent RTTs.

Simulation running times always longer than 300 RTTs.

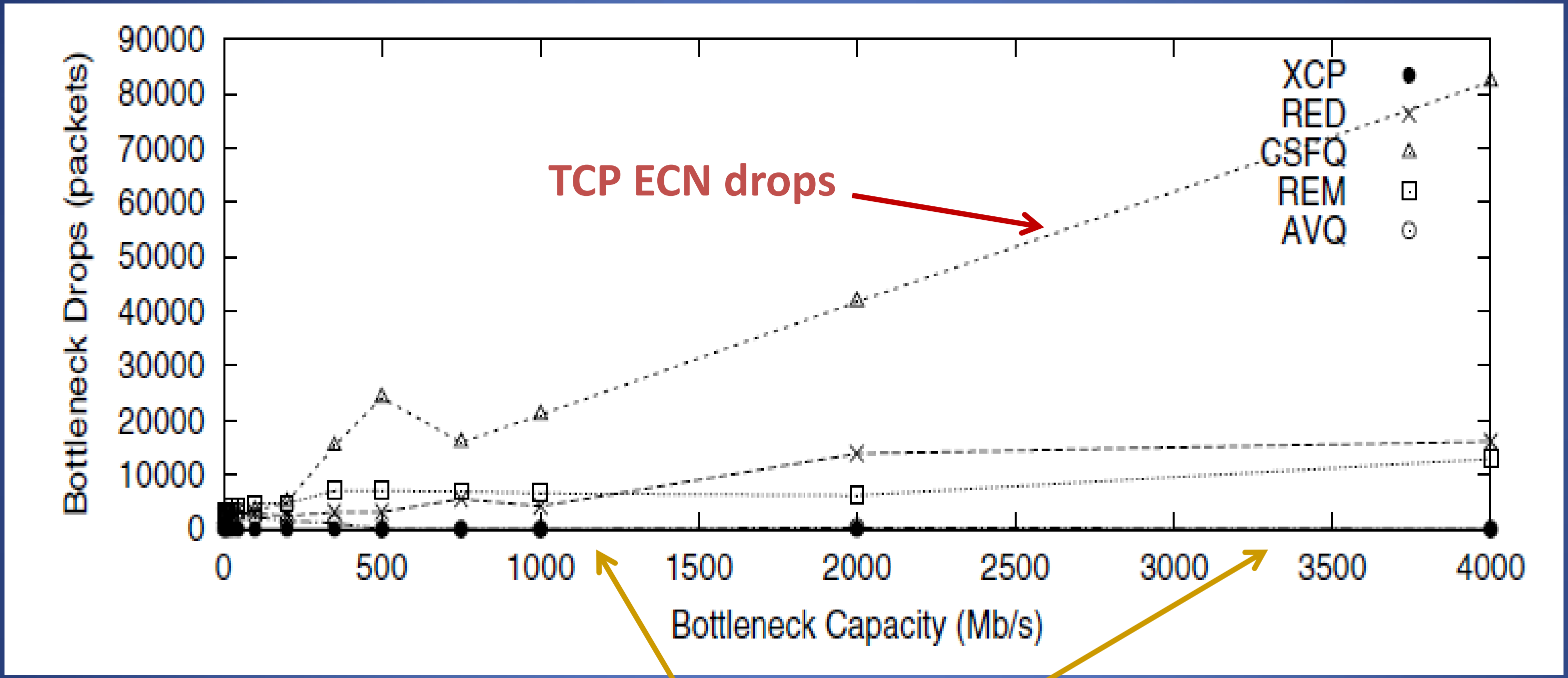
Figure 4 (top): Utilization vs Bottleneck Capacity

- 50 long-lived TCP flows
- 50 flows in reverse direction (two -way traffic)
- 80 ms. round-trip propagation delay
- Regardless of AQM scheme, bottleneck utilization for TCP degrades as capacity increases
- ***XCP is near optimal!***



[Dion 03]

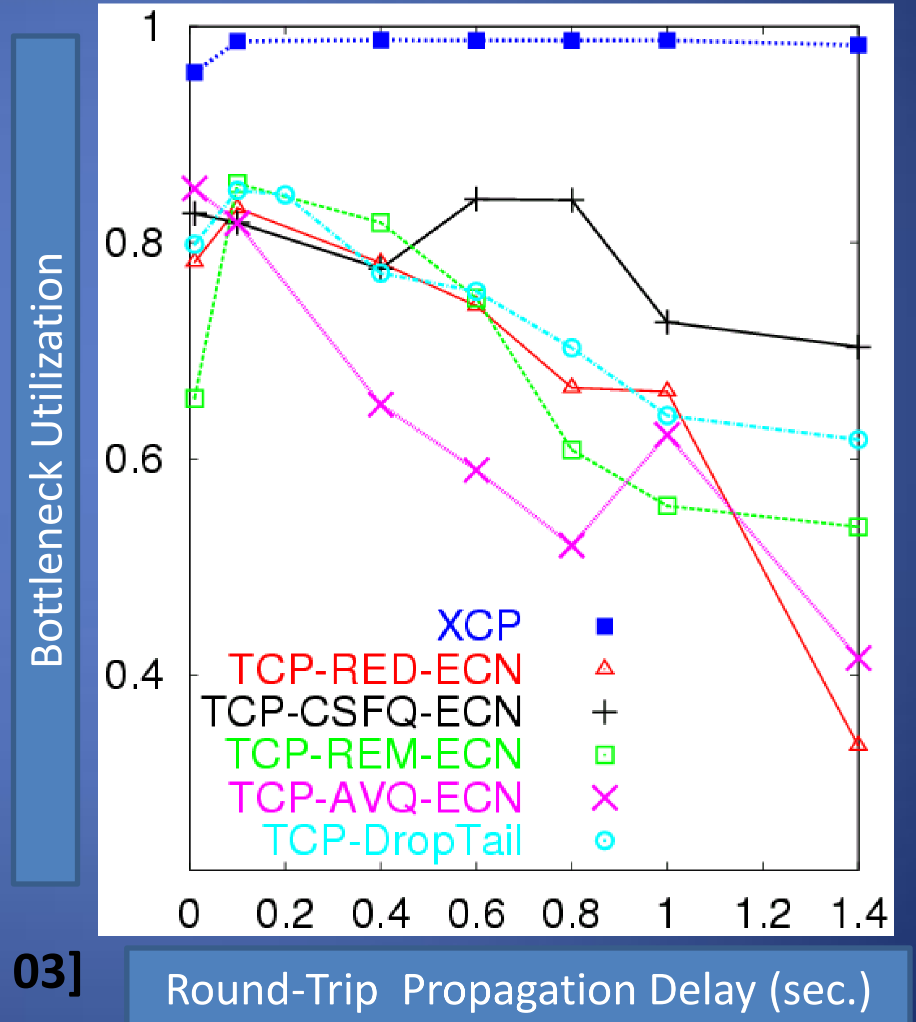
Figure 4 (bottom): Drops vs Bottleneck Capacity



XCP never drops packets

Figure 5: Utilization vs. Delay

- Bottleneck capacity fixed at 150 Mbps.
- All other parameters and flow characteristics are the same as in Figure 4.
- XCP keeps utilization high while TCP degrades with increased propagation delay (regardless of AQM scheme).



[Dion 03]

Impact of Number of Flows

- 50 long-lived TCP flows
- 50 flows in reverse direction
- 80 ms. round-trip propagation delay
- 150 Mbps capacity
- Claim: XCP increased queue size as number of flows increase is due to its high fairness!

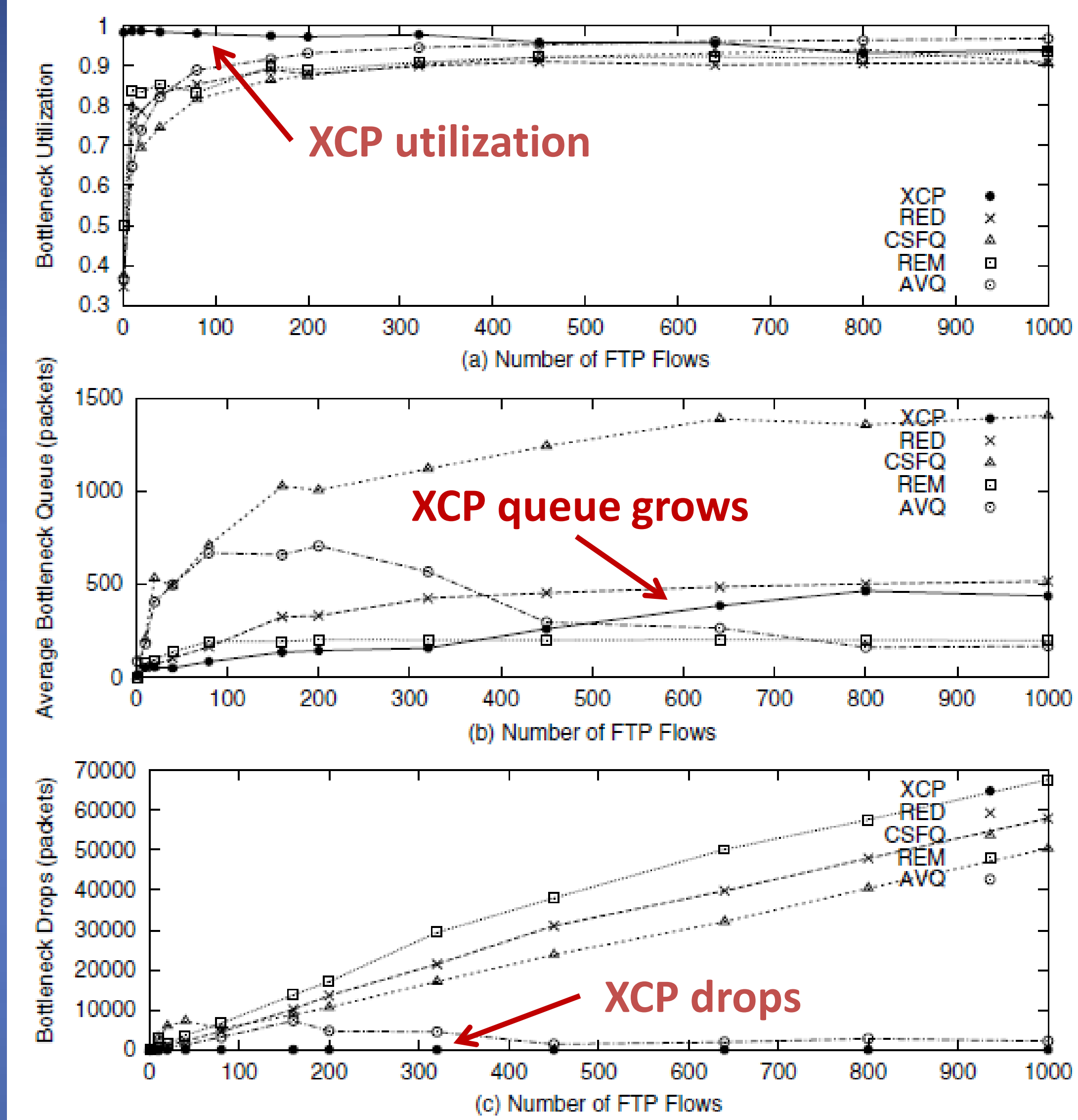


Figure 6: XCP is efficient with any number of flows. The graphs compare the efficiency of XCP and TCP with various queuing schemes as a function of the number of flows.

Impact of Short Web-Like Traffic

- 50 long-lived TCP flows
- 50 flows in reverse direction
- 80 ms. round-trip propagation delay
- 150 Mbps capacity

Short flows:

Poisson process arrivals

Transfer size – Pareto distribution with 30 packet mean and shape = 1.35

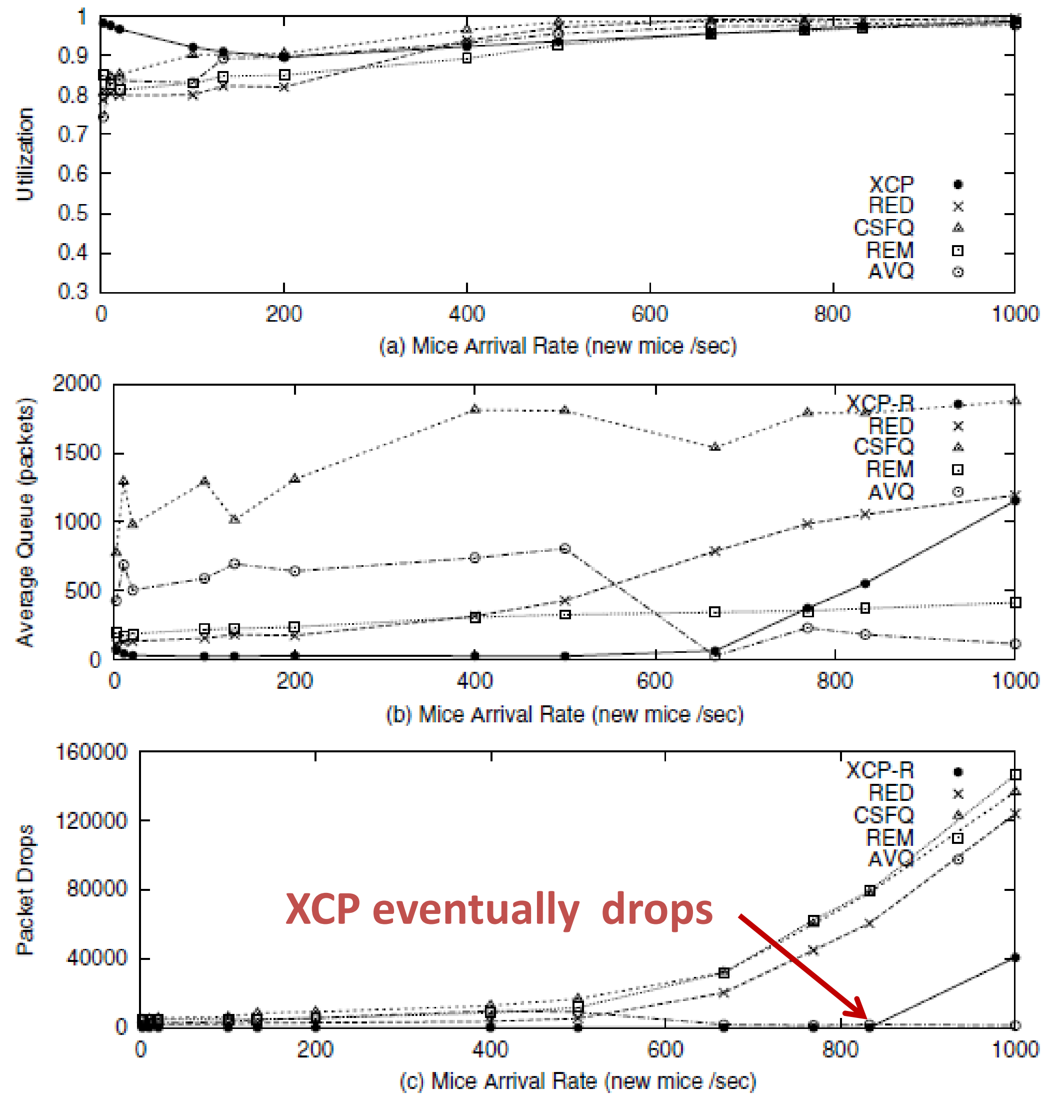
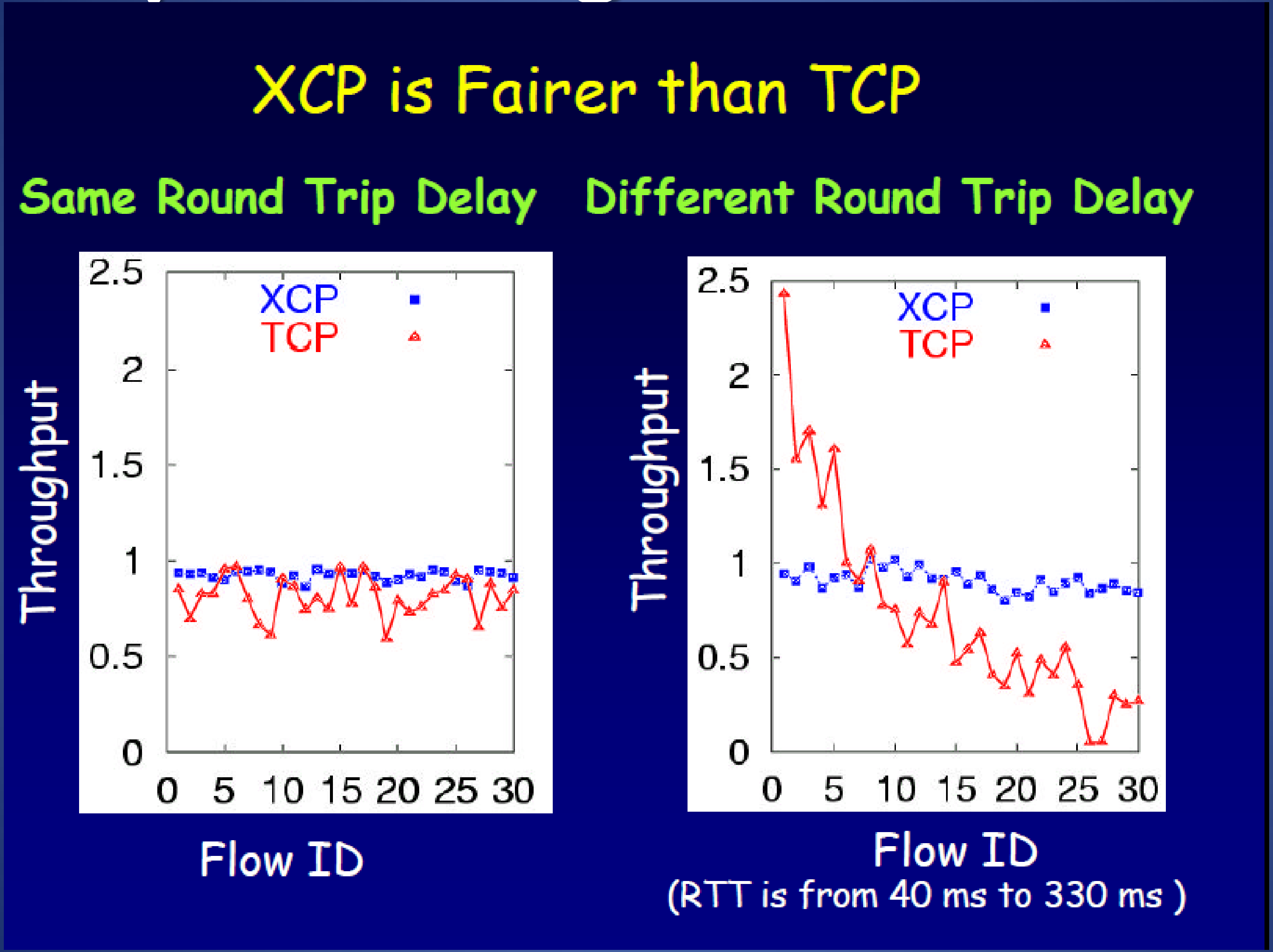


Figure 7: XCP is robust and efficient in environments with arrivals and departures of short web-like flows. The graphs compare the efficiency of XCP to that of TCP over various queuing schemes as a function of the arrival rate of web-like flows.

Simplified Figure 8 [TCP == RED]



[Katabi 02]

XCP Convergence Dynamics

- 5 long-lived flows with 2-sec staggered start times.
- 45 Mbps capacity
- Common 40 ms RTT

XCP maintains min-max fairness without harming utilization.

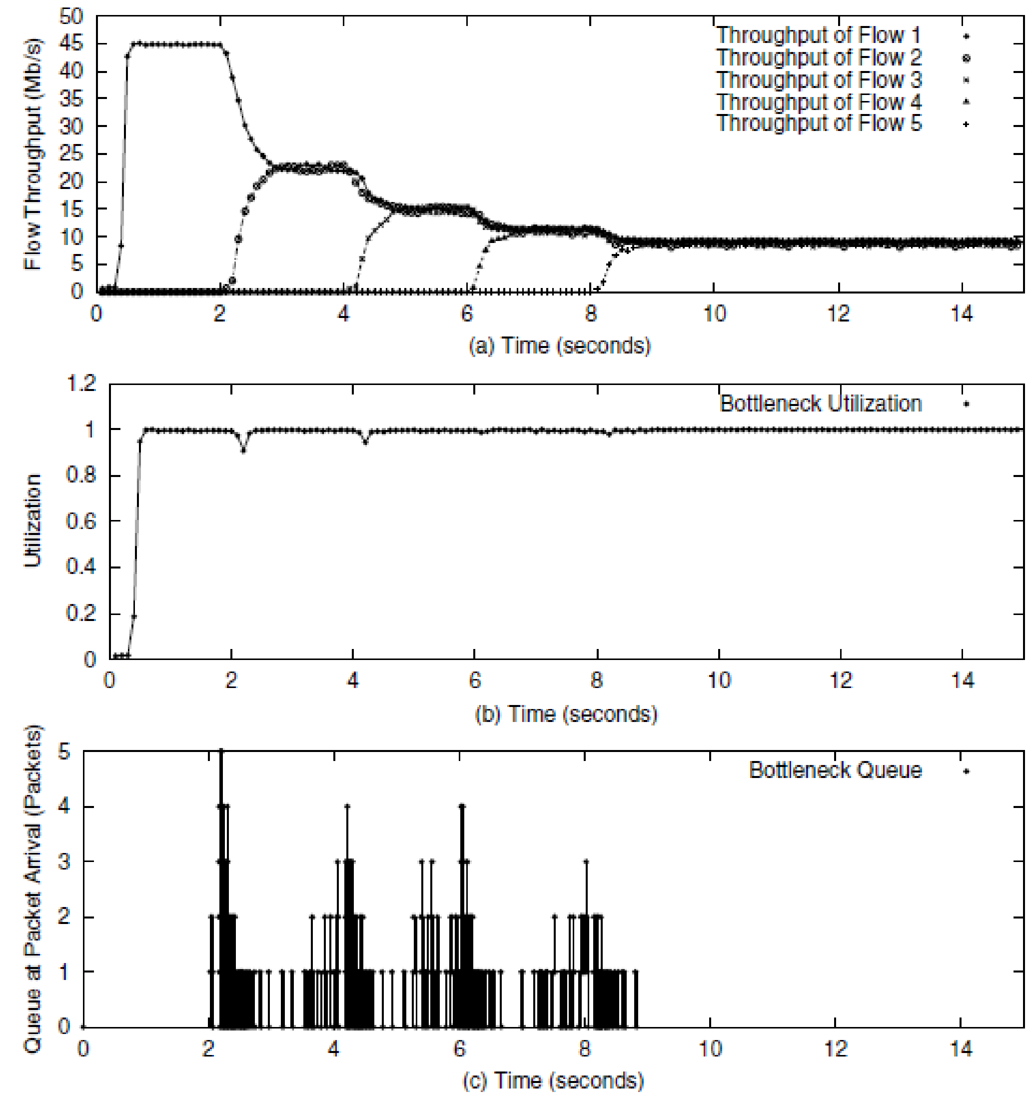


Figure 10: XCP's smooth convergence to high fairness, good utilization, and small queue size. Five XCP flows share a 45 Mb/s bottleneck. They start their transfers at times 0, 2, 4, 6, and 8 seconds.

Figure 11 Robustness to Sudden Changes in Traffic Demand

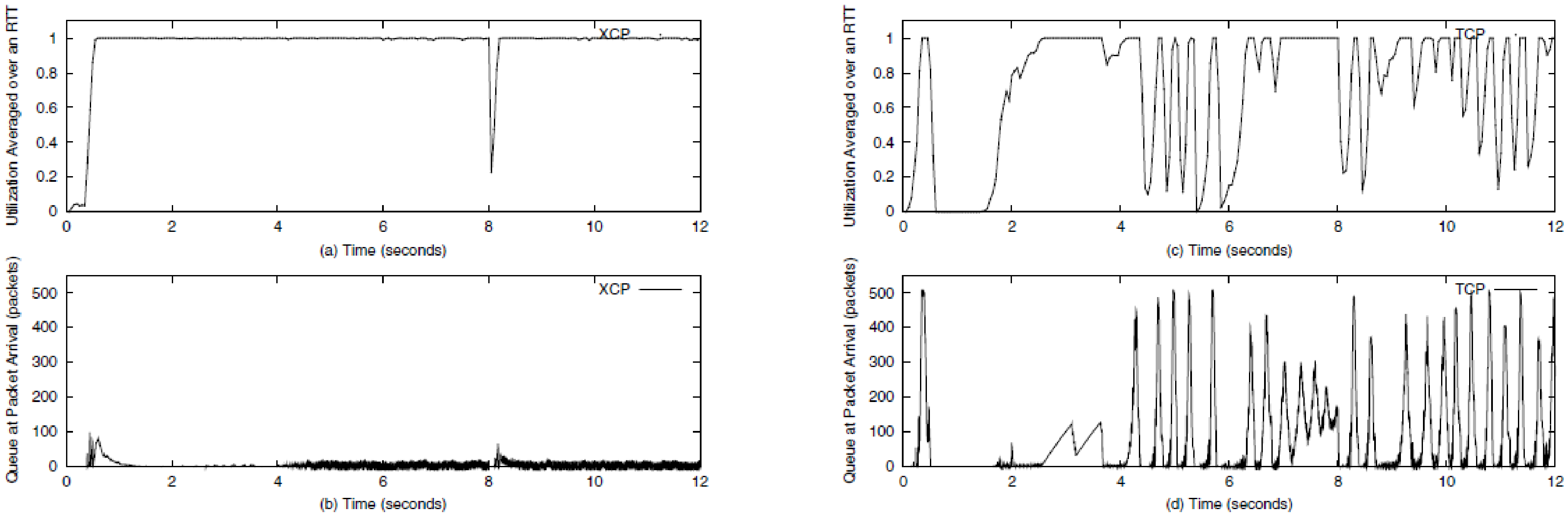


Figure 11: XCP is more robust against sudden increase or decrease in traffic demands than TCP. Ten FTP flows share a bottleneck. At time $t = 4$ seconds, we start 100 additional flows. At $t = 8$ seconds, these 100 flows are suddenly stopped and the original 10 flows are left to stabilize again.

Flow Characteristics

10 long-lived FTP flows share 100 Mbps bottleneck capacity.

All flows have 40 ms. RTTs. **TCP flows traverse RED router.**

High RTT Variance

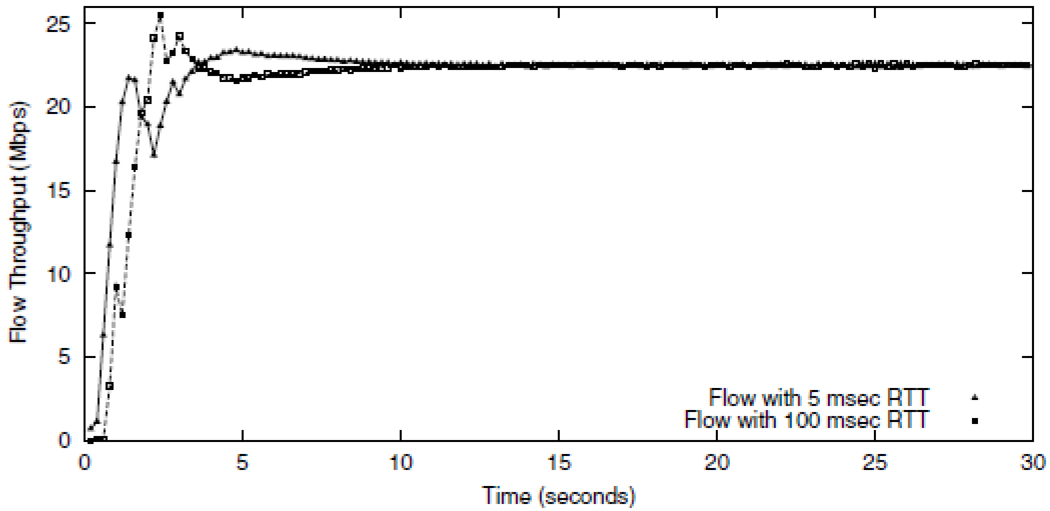


Figure 16: XCP robustness to high RTT variance. Two XCP flows each transferring a 10 Mbytes file over a shared 45 Mb/s bottleneck. Although the first flow has an RTT of 20 ms and the second flow has an RTT of 200 ms both flows converge to the same throughput. Throughput is averaged over 200 ms intervals.

XCP Issues

1. Source 'cheating'
   - How to handle misbehaving XCP sources that lie about RTT and do not use correct sending rate?
   - XCP needs 'policying agent' in edge XCP router.
2. How to deploy XCP?
   - Use island concept (called cloud-based) similar to CSFQ.
3. How to deal with UDP?
   - Encapsulate TCP and UDP into an XCP flow at ingress to island and use egress router as XCP receiver.

   → Ingress router must retain XCP state info for each flow.

XCP Issues

4. How to be TCP-friendly?
   – For XCP to co-exist on deployment with TCP RED at router, authors offer WFQ scheme for T-queue and X-queue.

- **Problem :: WFQ is stateful and does not scale!**
- This means XCP valuable only if its deployment eliminates TCP flows which dominate the current Internet (~90%).

Conclusions

- New high speed links in Internet cause flow BDPs to grow.
- Usual AQM suspects, even with control theory, have trouble with stability when feedback delay gets high.
- XCP decouples efficiency from fairness with two controllers in the XCP router.
- XCP fairness mechanism with bandwidth shuffler converges faster than TCP to fair allocation.

XCP Critique

- Paper includes no simulations with UDP. (Remember – this was the strength of the CSFQ scheme.)
- XCP forgets about **advertised window** in TCP (i.e., how does XCP adjust if receiver buffering is limited?).
- Later researchers (Low 2005) worry about restricted XCP utilizations (~80%) when all flows do not share the same bottleneck link. Additionally, with bad parameter choices a flow may only receive a small fraction of its min-max fairness (see Yang 2010 for proposed iXCP improvement).

XCP Critique (cont.)

- The implicit XCP trust of the Sender host enables denial-of-service attacks from malicious hosts.
- How does XCP perform if packets are dropped downstream (especially last-hop wireless LANS)?
- Other recent researchers point out that the inability to effectively determine available capacity in WLANs (with dynamic rate adaptation) cause XCP to over-allocate link capacity among the flows.

Acknowledgements

- [Dion 03] Used a few figures and modified a few slides from Chris Dion's student presentation in CS577 (Spring 2003).
- [Katabi 02] Used a couple of figures/slides from Dina Katabi's SIGCOMM02 presentation.

Thanks!

Questions ??