Congestion Control for High Bandwidth-Delay Product Networks

Diana Katabi	**Mark Handley**	**Charlie Rohrs**
MIT-LCS	ICSI	Tellabs
dk@mit.edu	mhj@icsi.berkeley.edu	crohrs@mit.edu

Presented by **Matthew Packard**

June 24, 2003

About the Authors - Dina Katabi

Dina Katabi → http://ana.lcs.mit.edu/dina

- Assistant Professor at MIT Laboratory for Computer Science
- Academic history:
  - BS in Electrical Engineering, Damascus University - School of Engineering
  - MS in Computer Science, MIT
  - PhD in Computer Science, MIT
- Research interests:
  - congestion control
  - differentiated services
  - routing
  - wireless networking
  - network security
  - control and coding theory

About the Authors (Continued) - Mark Handley

Mark Handley → http://www.icir.org/mjh

- Professor of Networked Systems at the University College, London

- Academic history:
  - BS in Computer Science/Electrical Engineering, University College, London
  - PhD in Computer Science (?), University College, London

- Research interests:
  - XORP (eXtensible Open Router Platform)
  - routing protocols
  - congestion control

- Professional activities:
  - Internet Architecture Board (IAB) member
  - IETF Routing/Transport Area Directorate member

About the Authors (Continued) - Charlie Rohrs

Charlie Rohrs

- Fellow at Tellabs Research Center and Visiting Associate Professor at MIT

- Academic history:
  - BS in Computer Science (?) from Notre Dame
  - MS in Computer Science (?) from MIT
  - PhD in Computer Science (?) from MIT

- Research interests:
  - adaptive control
  - signal processing
  - communication theory
  - linear control theory

Introduction - Where Does TCP Fail?

- Internet is rapidly growing with many high bandwidth links
- High latency links will still exist (satellite, wireless)
- TCP becomes oscillatory and unstable as bandwidth-delay product increases
- It has been shown that *no* AQM solution can provide stability for TCP:
  - when the delay or bandwidth becomes too great
  - encompasses RED, REM, PIC, and AVQ
- Additive increase policy in TCP is too conservative for most high capacity links:
  - too many RTTs to acquire proper bandwidth - wasted time and bandwidth
- Short flows suffer the limitations of slow start - wasted RTTs in ramp up
- Unfairness results when high delay packets compete with low delay packets

Introduction (Continued) - What Does XCP Gain Us?

- XCP (eXplicit Control Protocol) - TCP replacement utilizing extended ECN
  - congestion no longer a binary notification - XCP allows for congestion degrees
  - decoupled utilization and fairness controllers
    - aggressiveness modified based on spare bandwidth and end-to-end delay
    - prevents oscillations, ensures throughput stability, and ensures efficiency
    - fairness controller reclaims from bandwidth hogs and redistributes it

- XCP requires no individual flow state information
  - scalable to any number of flows
  - minimal CPU overhead protocol

- XCP will be shown to exhibit:
  - high utilization (near 100%)
  - small queues
  - nearly zero drops

Introduction (Continued) - Additional XCP Benefits

- Decoupling fairness and efficiency controllers allow for service differentiation
- XCP distinguishes error losses from congestion losses - (congestion uncommon)

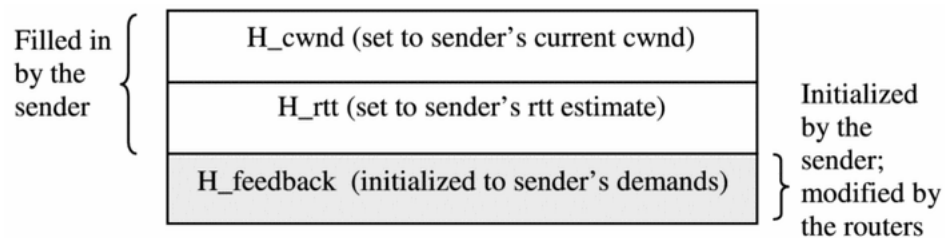
Design Rationale - Why Build XCP?

- What to avoid when building a congestion control algorithm from the ground up:
  - packet loss is not a useful congestion metric - congestion drop a last resort
  - implicit signalling using drops is not useful - other loss types exist
  - packet loss is a binary signal - hard to quickly find choke point
  - AIMD (additive increase, multiplicative decrease) needed when probing congestion
- XCP network nodes inform sender of congestion state - reduced reaction time
  - senders rapidly reduce window sizes during congestion
  - senders slowly reduce window sizes when utilization near maximum
  - overall effect is faster response with less oscillation
- XCP forces senders to react slowly to delay so as not to incur destabilization
- XCP should isolate congestion reaction from other network metrics (flows)

Design Rationale (Continued) - EC/FC Decoupling

- XCP decouples efficiency and fairness:
  - fair, per-flow bandwidth allocated independently of aggregate manipulations
- TCP uses AIMD for both fairness and efficiency
- Separating EC and FC allows for the independent updating of either one

Protocol - Framework and Congestion Header



XCP Congestion Header

- Senders maintain the congestion window (cwnd) and the round trip time (rtt)
  - communicated to routers in every packet
- Routers compare headers to available bandwidth and ask senders to adjust
  - notification sent via the H_feedback field in congestion header
  - other routers may overwrite this header with a higher restriction
- Sender receives updated congestion header, acknowledges it, and updates cwnd
- H_cwnd and H_rtt are never modified in line

Protocol (Continued) - XCP Sender, Receiver, and Router

- Sender requests up front bandwidth (r) in the $\texttt{H\_feedback}$ header section
  - $\texttt{H\_feedback} = \frac{r \cdot \texttt{rtt} - \texttt{cwnd}}{\texttt{cwnd} \cdot s}$, s is the packet size
  - This allows for one RTT desired bandwidth acquisition
- Upon header acknowledgement, `cwnd` increases (pos) or decreases (neg)
  - $\texttt{cwnd} = max(\texttt{cwnd} + \texttt{H\_feedback}, s)$
- The receiver copies the congestion header as is and sends it back to the sender
- XCP works on top of an existing drop policy (RED, Drop Tail, or AVQ)
- Feedback is monitored by the efficiency and fairness controllers
  - EC/FC updates information over the average RTT to prevent sluggishness
  - controllers act upon data every average RTT - verify previous action
- Each router interface has a separate average RTT timer, d

Protocol (Continued) - Efficiency Controller

- EC utilized to maximize link utilization - 100% goal
  - useful if EC prevents packet drops and maintains minimal queues
  - aggregate traffic interest only - no concern for per-flow fairness

- EC determines modifications to aggregate window size over an average RTT:
  - feedback function modeled by: $\phi = \alpha \cdot d \cdot S - \beta \cdot Q$
  - α and β are stability constants, 0.4 and 0.226 respectively
  - S is the spare bandwidth (link capacity - input traffic) - can be negative
  - Q is the persistent queue size (non single RTT drained)

- ϕ is positive when $S \geq 0$ - link is underutilized (request more)
  - ϕ is negative when $S < 0$ - link is saturated (back off)

- ϕ incorporates persistent queue issue, when $S = 0$ - queue steadily 'filled'

- ϕ returned to sender via `H_feedback`

Protocol (Continued) - Fairness Controller

- FC takes ϕ from EC and distributes it to even out all flows

- FC uses TCP's AIMD for fairness convergence - compute per packet feedback:
  - $\phi > 0$, allocate ϕ across all flows evenly
  - $\phi \leq 0$, deallocate a flow's throughput proportionally

- FC ensures continuous fairness convergence while $\phi \neq 0$
  - $\phi \approx 0$, perform bandwidth shuffling to prevent stalling
    - steal bandwidth from one and add simultaneously to another
  - shuffled traffic computed as: $h = \max(0, \gamma \cdot y - \mid \phi \mid)$
  - y is the average input traffic over an RTT
  - γ is a constant set to 0.1 - 10% traffic shuffling per RTT

- Compute individual packet's (i) feedback (pos - neg), maintaining AIMD:
  - $\texttt{H\_feedback}_i = p_i - n_i$

Protocol (Continued) - Fairness Controller Effects

- $\phi > 0$, increase flow i `cwnd` proportional to its RTT
  - Per packet feedback increase determined by:
  - $p_i = \xi_p \frac{\mathtt{rtt}_i^2 \cdot s_i}{\mathtt{cwnd}_i}$ where $\xi_p = \frac{h + \max(\phi, 0)}{d \cdot \sum \frac{\mathtt{rtt}_i \cdot s_i}{\mathtt{cwnd}_i}}$
- $\phi < 0$, decrease flow i `cwnd` proportional to its RTT
  - Per packet feedback decrease determined by:
  - $n_i = \xi_n \cdot \mathtt{mbox}_i \cdot s_i$ where $\xi_n = \frac{h + \max(-\phi, 0)}{d \cdot \sum s_i}$

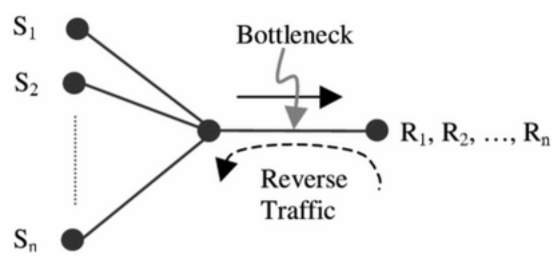
Protocol (Continued) - Efficiency/Fairness Controller Notes

- EC is MIMD based for fast acquisition and release of bandwidth
- FC is AIMD based for slow acquisition and fast release of bandwidth
- XCP's FC converges toward fairness faster than TCP
  - XCP AIMD allows all flows to increase equally, with rapid decrease (fair part)
  - TCP MD tied to packet drops, XCP MD decoupled and occurs every average RTT

Performance - Simulation Setup

- Simulations run with the following inputs:
  - link capacities from 1.5 Mb/s to 4 Gb/s
  - propagation delays from 10 ms to 1.4 seconds
  - number of sources from 1 to 1000
  - two-way traffic with ACK compression (burst queued ACKs)
  - short, web-like traffic

- Simulations utilize the topology in the following diagram:



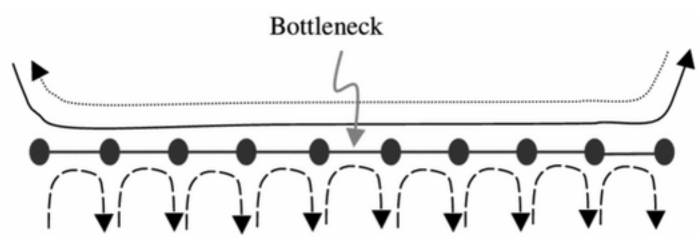
Single Bottleneck Topology

Performance (Continued) - Extended Simulation Setup

- Simulations run with the NS-2 simulator with an XCP module versus TCP Reno
- XCP compared with TCP Reno over:
  - gentle RED - $q_{min} = \frac{1}{3}$ and $q_{max} = \frac{2}{3}$
  - REM - $\phi = 1.001$, $\gamma = 0.001$, update interval $=$ 10 packets
  - AVQ - $\gamma = 0.98$ and $\alpha = 0.15$
  - CSFQ - set via CSFQ paper (chosen to show CSFQ can be made fairer)
- XCP settings, α set to 0.4, and β set to 0.226
  - XCP used RED and TD, but did not make much difference (few drops)
- Default packet size set at 1000 bytes (jumbo frames for GigE?)
- Buffer size set to the delay-bandwidth product
- All flows are long lived FTP sessions

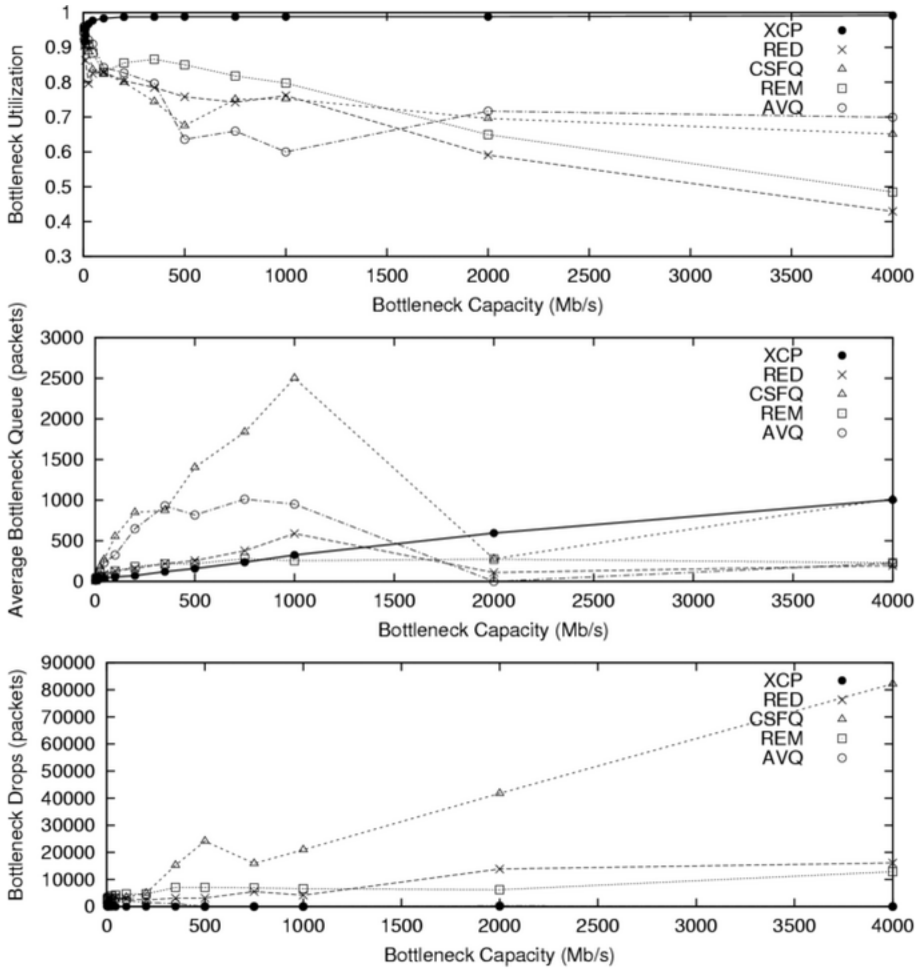
Performance (Continued) - Extended Simulation Setup (Cont)

❖ Simulations can be extended to show that more complex topologies can be extracted:

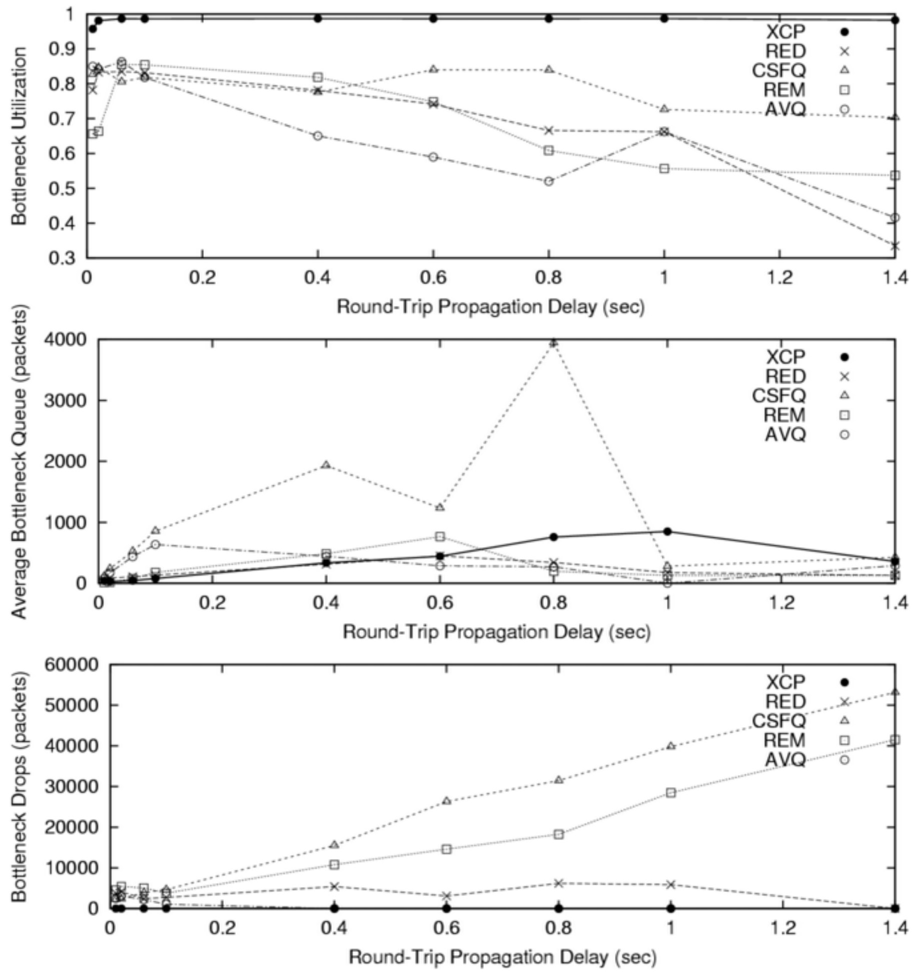


Parking Lot Topology

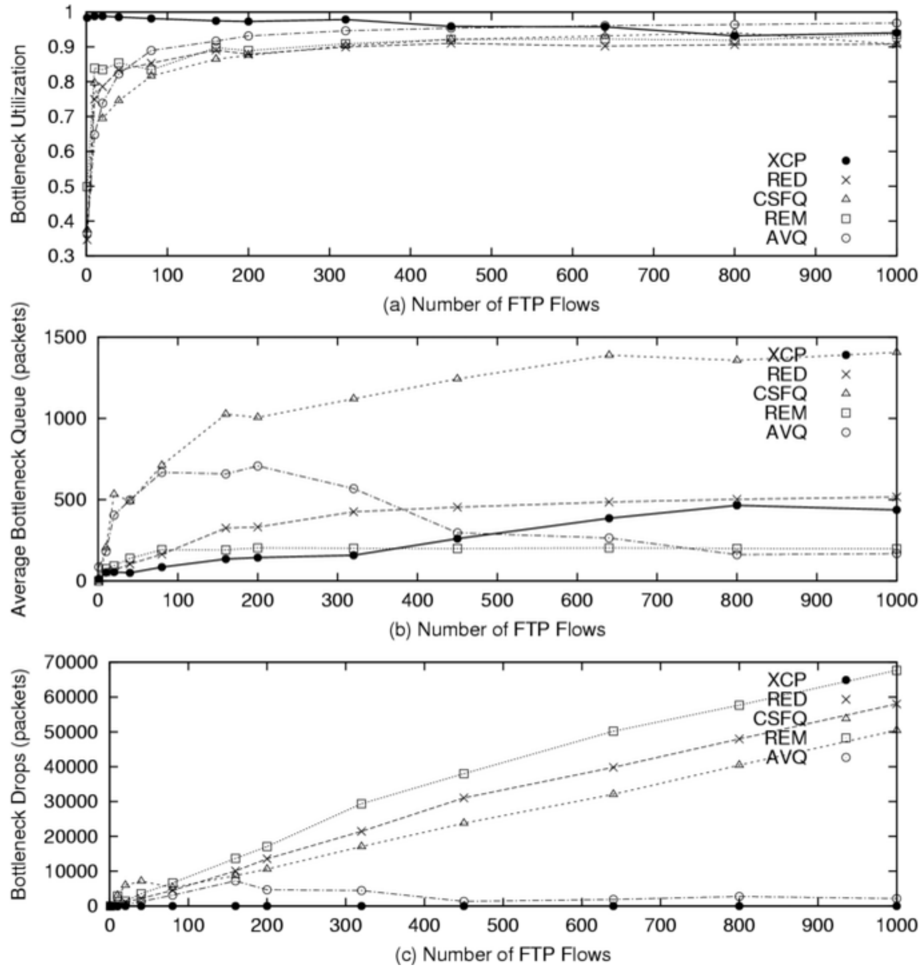
XCP Efficiency as a Function of Capacity



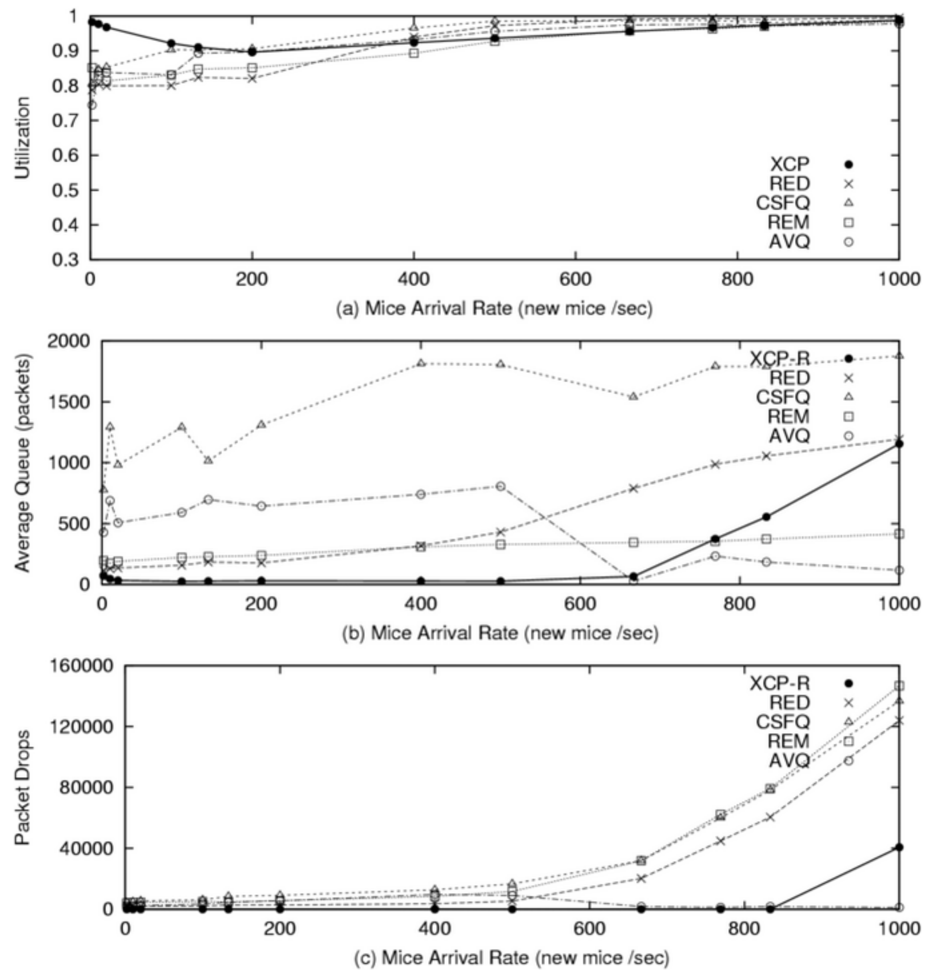
XCP Efficiency as a Function of RTT Delay



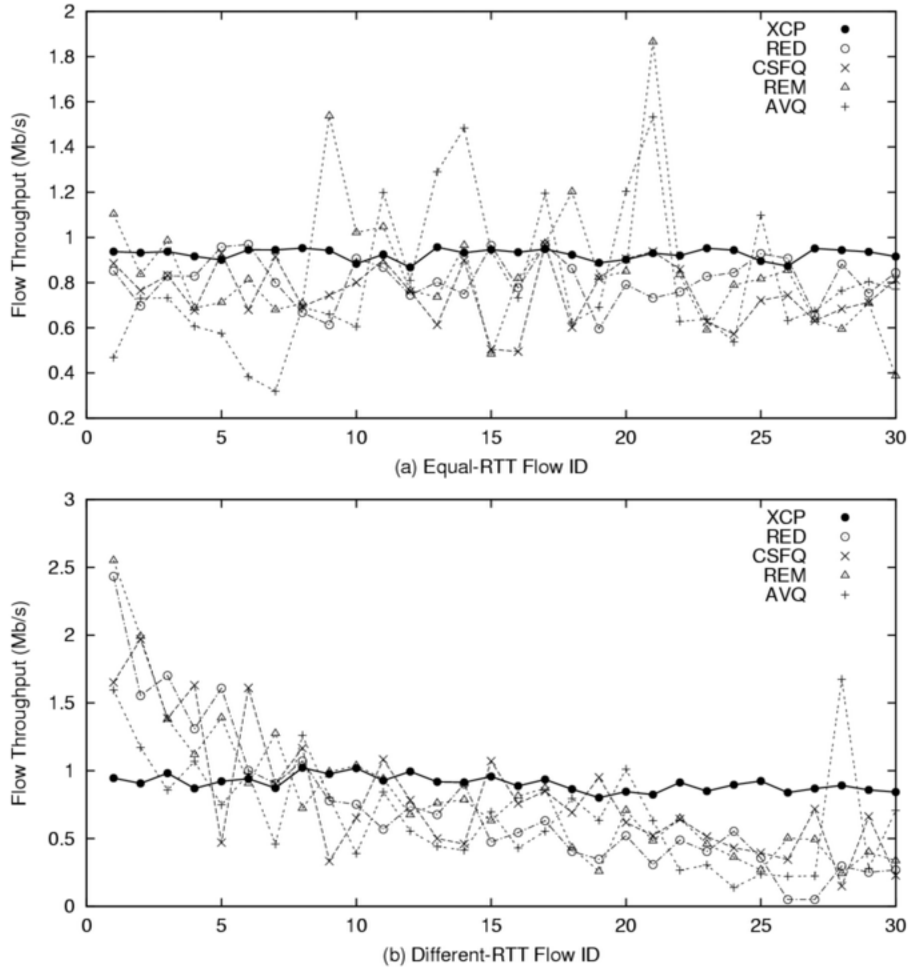
XCP Efficiency as a Function of FTP Flows



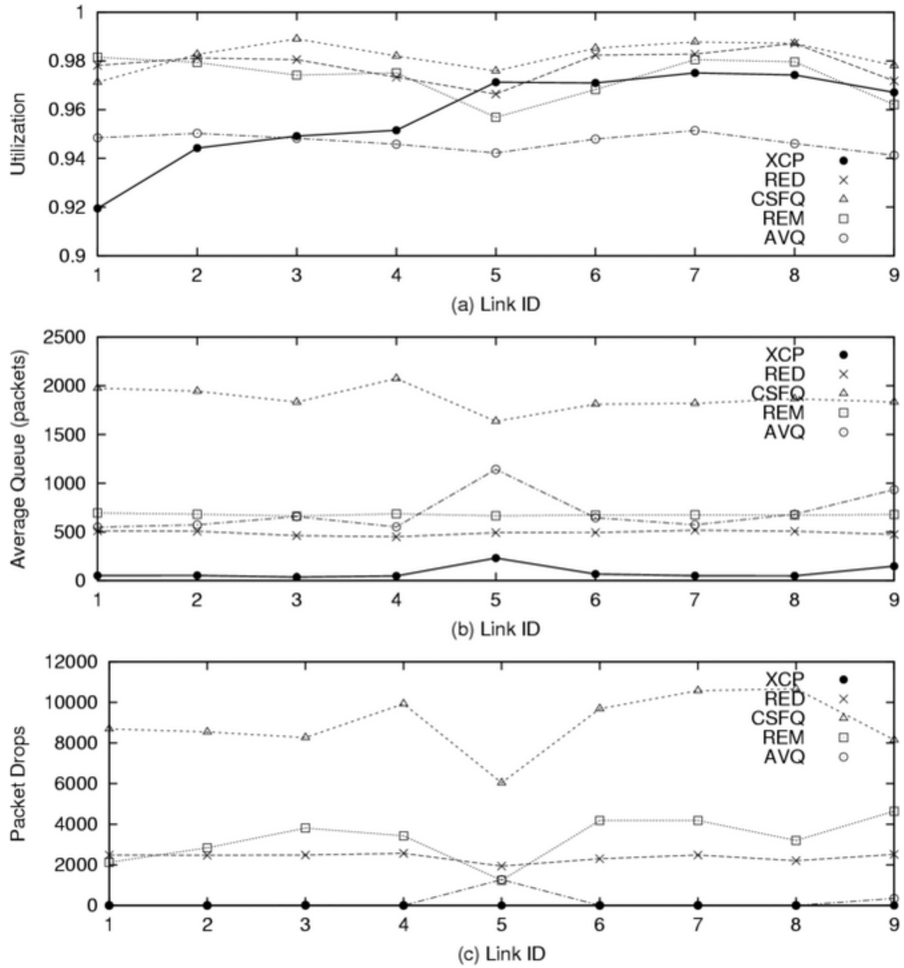
XCP Efficiency as a Function of Mice Arrivals



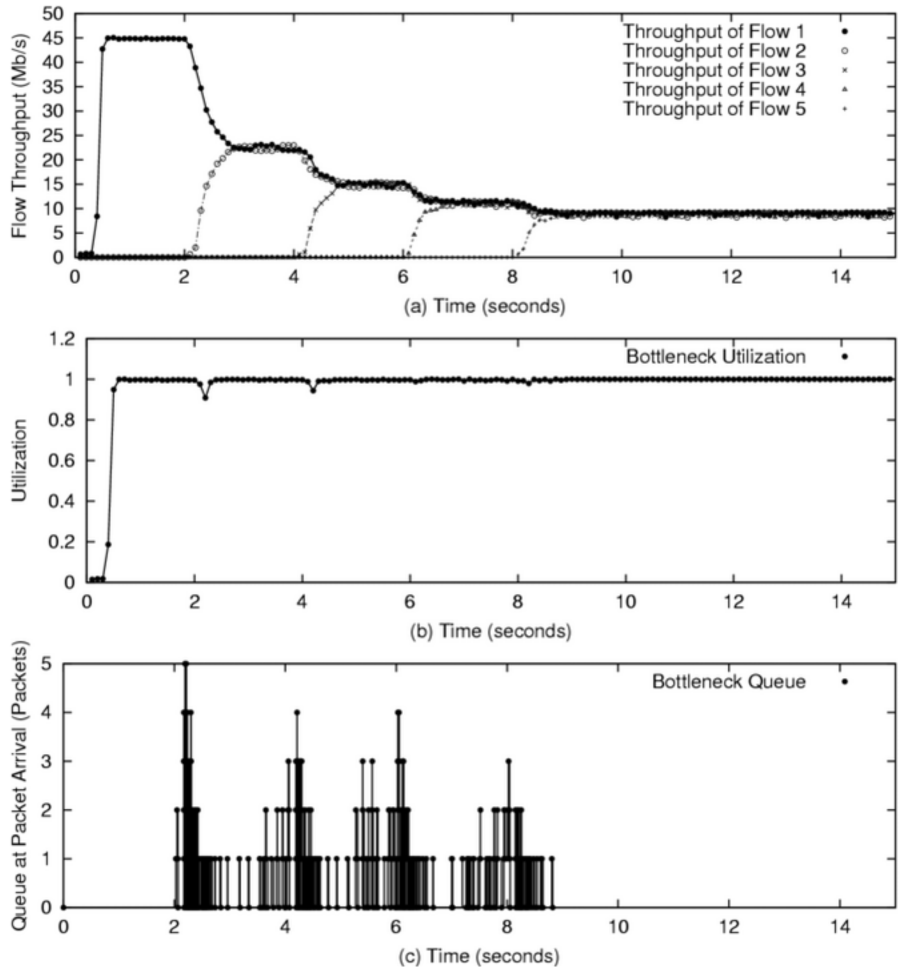
XCP Throughput as a Function of Mixed RTT



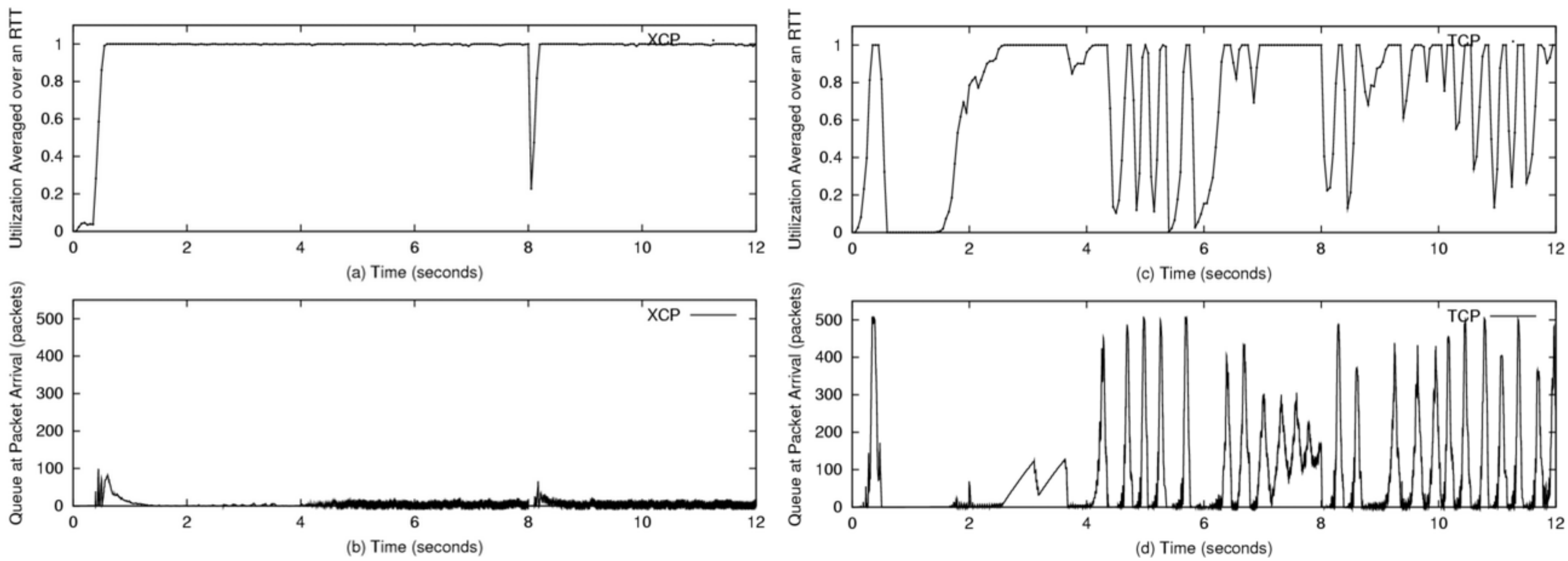
XCP Efficiency as a Function of Congested Queues



XCP Smoothness as a Function of Time



XCP Flexibility as a Function of Flows



Security - Detecting Misbehaving Flows

- XCP allows for detection of unresponsive or misbehaving flows
  - Use of explicit feedback to test for unresponsiveness in one RTT
- TCP does not maintain RTT and must keep track of long-interval average

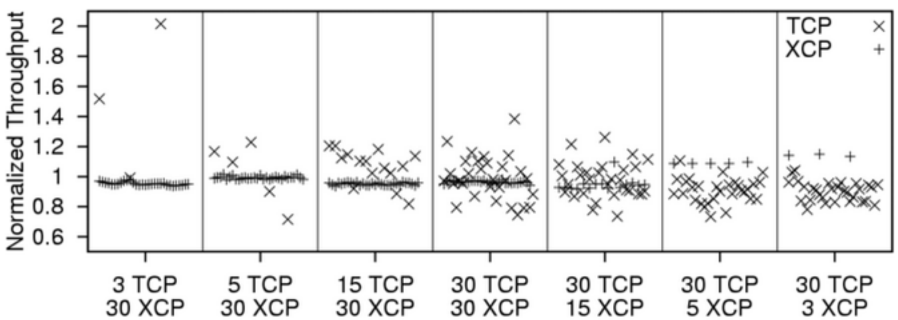
Gradual Deployment - TCP/UDP Mapping and Coexistence

- Deployment akin to CSFQ - core of XCP with edges of FIFO/TD, RED, etc

- Map TCP/UDP flows onto XCP flows between source/destination edge routers
  - XCP flow associated with queue on inbound router - sets dispatch frequency

- Or, use no congestion header - use control packet from edge routers
  - updated every RTT - one XCP flow per same in/out router pairs

- XCP can coexist with TCP - sender checks for XCP compatibility at start
  - router treats TCP flows with RED, and XCP normally (equal service)



XCP is TCP-friendly

Conclusion - Quick Recap

- TCP falters under higher delay-bandwidth product
- XCP decouples fairness and efficiency
- XCP congestion header - one RTT bandwidth modifications (explicit)
- XCP is:
  - highly efficient (100% link utilization)
  - low cost to router CPUs
  - prevents packet drops (very low percentage)
  - maintains low queues

Discussion

❖ Questions?

Slide Generation Utilities

- The GIMP → http://www.gimp.org
  - PNG cropping/chopping
- ImageMagick → http://www.imagemagick.org
  - `convert` utility for PDF image extraction and PNG conversion
- LaTeX → http://www.tug.org
  - `pdflatex` utility for PDF slide output
- Slide Generation Process:
  - scale original PDF to at least 4 times normal size:
    - `convert -enhance -antialias -density 300 xcp.pdf xcp.png`
  - open each PNG with `display` and cut out the enlarged picture
  - crop/chop the image with `display` or The GIMP
  - generate the LaTeX source and create the PDF with `pdflatex`