

CS 268: Lecture 7 (Beyond TCP Congestion Control)

Ion Stoica
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720-1776

(Based on slides from R. Stallings, M. Handley and D. Katabi)

Outline

- TCP-Friendly Rate Control (TFRC)
 - ATM Congestion Control
 - eXplicit Control Protocol

TCP-Friendly

- Any alternative congestion control scheme needs to *coexist with TCP in FIFO queues* in the best-effort Internet, or *be protected from TCP* in some manner.
- To co-exist with TCP, it must impose the same long-term load on the network:
 - No greater long-term throughput as a function of packet loss and delay so TCP doesn't suffer
 - Not significantly less long-term throughput or it's not too useful

3

TFRC: General Idea

Use a model of TCP's throughput as a function of the loss rate and RTT directly in a congestion control algorithm.

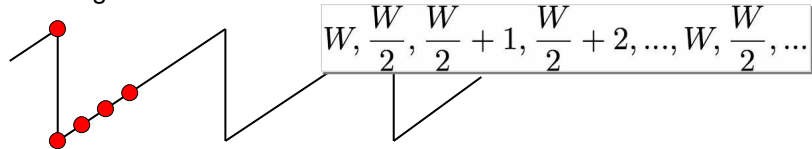
- If transmission rate is higher than that given by the model, reduce the transmission rate to the model's rate.
- Otherwise increase the transmission rate.

4

TCP Modelling: The "Steady State" Model

The model: Packet size B bytes, round-trip time R secs, no queue.

- A packet is dropped each time the window reaches W packets.
- TCP's congestion window:



- The maximum sending rate in packets per roundtrip time: W
- The maximum sending rate in bytes/sec: WB/R
- The average sending rate T : $T = (3/4)WB/R$

- The packet drop rate p :
$$p = \frac{1}{\frac{3}{8}W^2}$$

- **The result:**
$$T = \frac{\sqrt{6}B}{2R\sqrt{p}} = \frac{\sqrt{3/2}B}{R\sqrt{p}}$$

5

An Improved "Steady State" Model

A pretty good improved model of TCP Reno, including timeouts, from Padhye et al, Sigcomm 1998:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1 + 32p^2)}$$

T : sending rate in bytes/second]

R : round trip time

p : fraction of packets lost

t_{RTO} : TCP retransmission timeout

Would be better to have a model of TCP SACK, but the differences aren't critical.

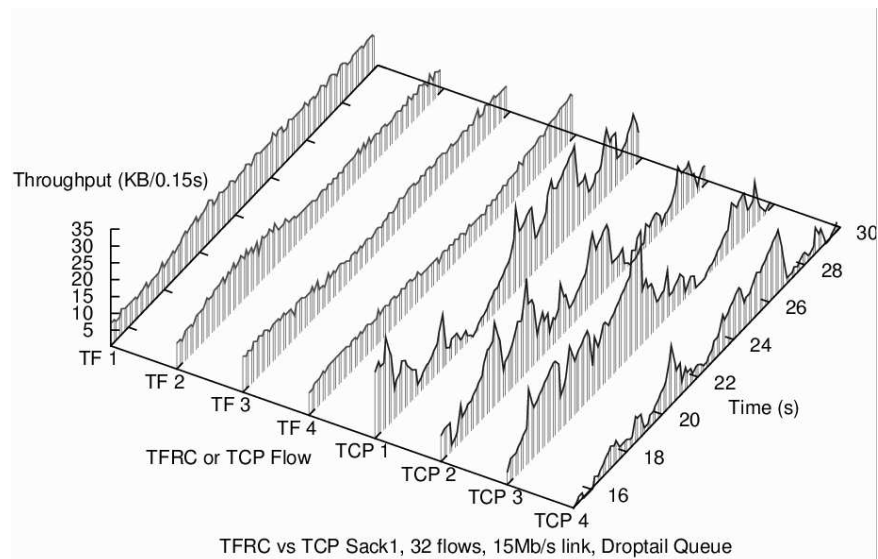
6

TFRC Details

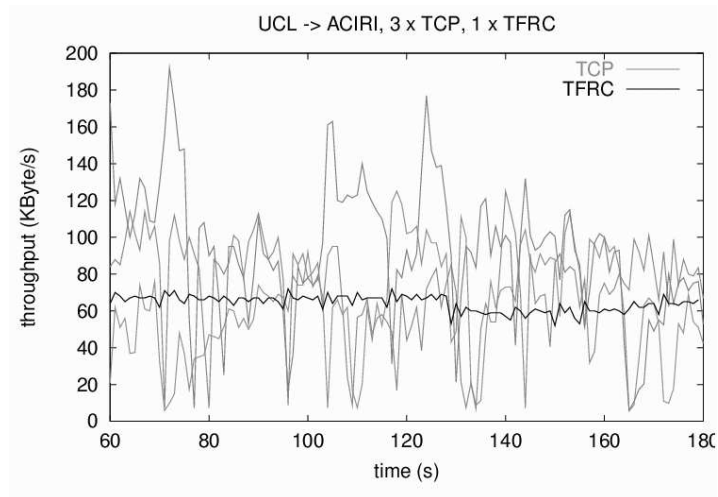
- The devil's in the details
 - How to measure the loss rate?
 - How to respond to persistent congestion?
 - How to use RTT and prevent oscillatory behavior?
- Not as simple as first thought

7

TFRC Performance (Simulation)



TFRC Performance (Experimental)



9

Outline

- TCP-Friendly Rate Control (TFRC)
- ATM Congestion Control
- eXplicit Control Protocol

10

ATM Congestion Control

- **Credit Based**

- Sender is given “credit” for number of octets or packets it may send before it must stop and wait for additional credit.

- **Rate Based**

- Sender may transmit at a rate up to some limit.
- Rate can be reduced by control message.

11

Case study: ATM ABR congestion control

ABR: available bit rate:

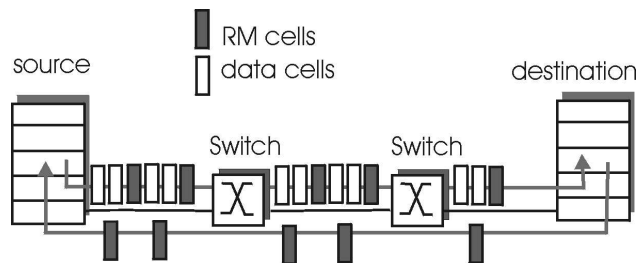
- “elastic service”
- if sender’s path “underloaded”:
 - Sender should use available bandwidth
- if sender’s path congested:
 - Sender throttled to minimum guaranteed rate

RM (resource management) cells:

- Sent by sender, interleaved with data cells
- Bits in RM cell set by switches (“*network-assisted*”)
 - NI bit: no increase in rate (mild congestion)
 - CI bit: congestion indication
- RM cells returned to sender by receiver, with bits intact

12

EXPLICIT Case study: ATM ABR congestion control



- Two-byte ER (explicit rate) field in RM cell
 - Congested switch may lower ER value in cell
 - Sender' send rate thus minimum supportable rate on path
- EFCI bit in data cells: set to 1 in congested switch
 - If data cell preceding RM cell has EFCI set, sender sets CI bit in returned RM cell

13

ABR Cell Rate Feedback Rules

- if CI == 1
 - Reduce ACR to a value \geq MCR
- else if NI == 0
 - Increase ACR to a value \leq PCR
- if ACR > ER
 - set ACR = $\max(\text{ER}, \text{MCR})$

ACR = Allowed Cell Rate
MCR = Minimum Cell Rate
PCR = Peak Cell Rate
ER = Explicit Rate

14

Outline

- TCP-Friendly Rate Control (TFRC)
- ATM Congestion Control
- eXplicit Control Protocol

15

TCP congestion control performs poorly as bandwidth or delay increases

Shown analytically in [Low01] and via simulations



Because TCP lacks fast response

- Spare bandwidth is available \Rightarrow TCP increases by 1 pkt/RTT even if spare bandwidth is huge
- When a TCP starts, it increases exponentially \Rightarrow Too many drops \Rightarrow Flows ramp up by 1 pkt/RTT, taking forever to grab the large bandwidth

16

Solution: Decouple Congestion Control from Fairness

High Utilization;
Small Queues;
Few Drops

Bandwidth
Allocation
Policy

17

Solution: Decouple Congestion Control from Fairness

Coupled because a *single* mechanism controls both

Example: In TCP, Additive-Increase Multiplicative-
Decrease (AIMD) controls both

How does decoupling solve the problem?

1. To control congestion: use MIMD which shows fast response
2. To control fairness: use AIMD which converges to fairness

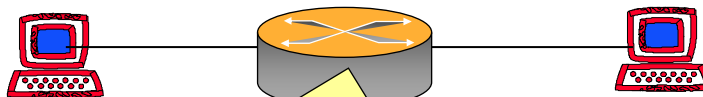
18

Characteristics of XCP Solution

1. Improved Congestion Control (in high bandwidth-delay & conventional environments):
 - Small queues
 - Almost no drops
2. Improved Fairness
3. Scalable (no per-flow state)
4. Flexible bandwidth allocation: min-max fairness, proportional fairness, differential bandwidth allocation,...

19

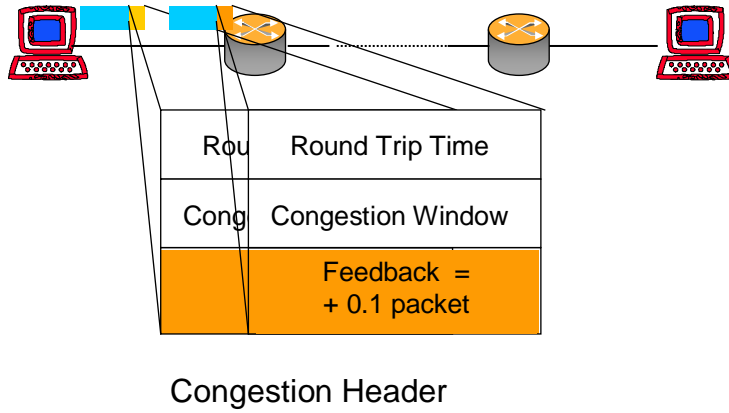
XCP: An eXplicit Control Protocol



1. Congestion Controller
2. Fairness Controller

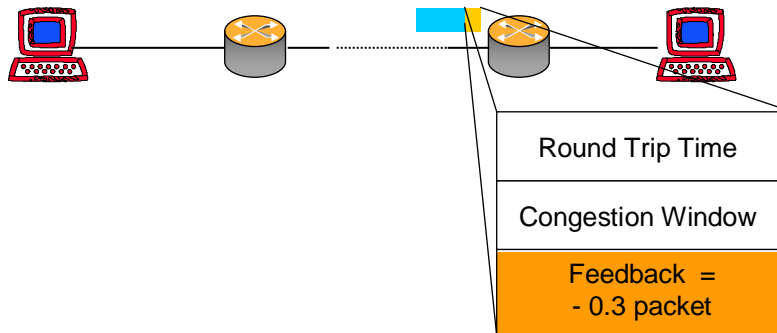
20

How does XCP Work?



21

How does XCP Work?



22

How does XCP Work?



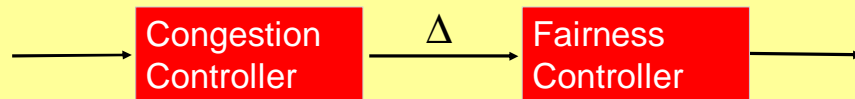
Congestion Window = Congestion Window + Feedback

XCP extends ECN and CSFQ

Routers compute feedback without any per-flow state

23

How Does an XCP Router Compute the Feedback?



queue

MIMD

Algorithm:

Aggregate traffic changes by Δ

$\Delta \sim$ Spare Bandwidth

$\Delta \sim$ - Queue Size

So, $\Delta = \alpha d_{avg} \text{ Spare} - \beta \text{ Queue}$

Congestion He

AIMD

Algorithm:

If $\Delta > 0 \Rightarrow$ Divide Δ equally between flows

If $\Delta < 0 \Rightarrow$ Divide Δ between flows proportionally to their current rates

24

Getting the devil out of the details ...

Congestion Controller

$$\Delta = \alpha d_{avg} Spare - \beta Queue$$

Theorem: System converges to optimal utilization (i.e., stable) for any link bandwidth, delay, number of sources if:

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

No Parameter Tuning

Fairness Controller

Algorithm:

If $\Delta > 0 \Rightarrow$ Divide Δ equally between flows

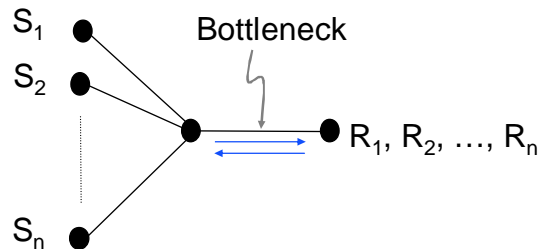
If $\Delta < 0 \Rightarrow$ Divide Δ between flows proportionally to their current rates

Need to estimate number of flows N

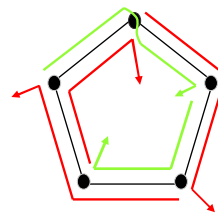
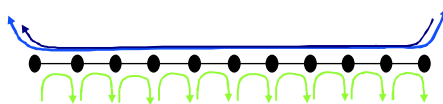
$$N = \sum_{pkts \text{ in } T} \frac{1}{T \times (Cwnd_{pkt} / RTT_{pkt})}$$

No Per-Flow State

Subset of Results

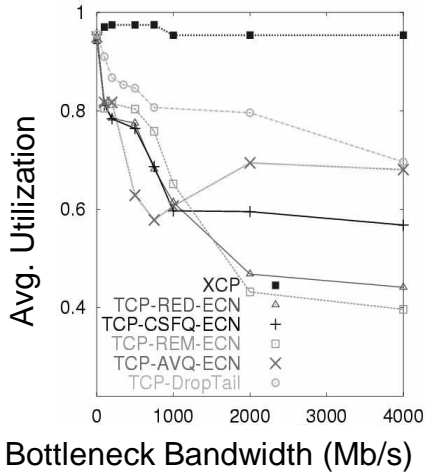


Similar behavior over:

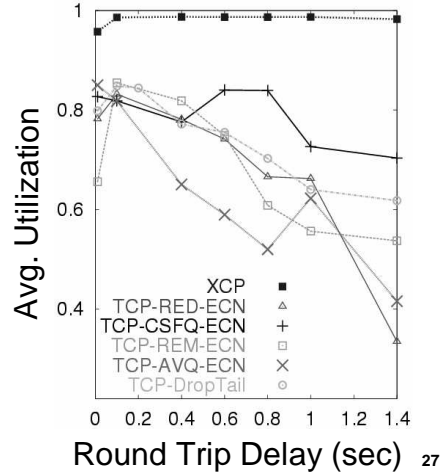


XCP Remains Efficient as Bandwidth or Delay Increases

Utilization as a function of Bandwidth

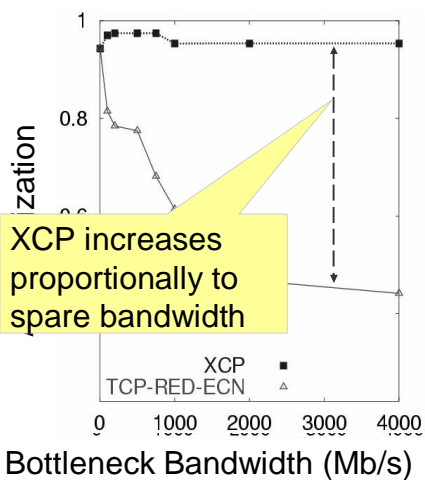


Utilization as a function of Delay

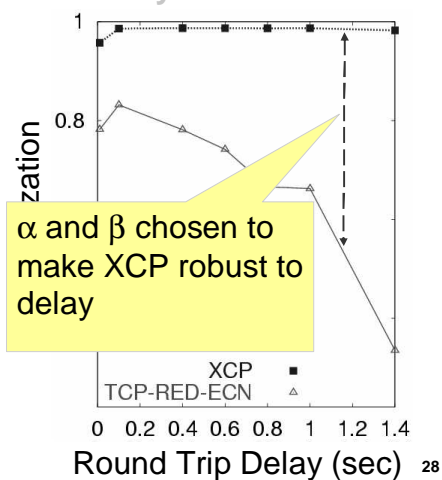


XCP Remains Efficient as Bandwidth or Delay Increases

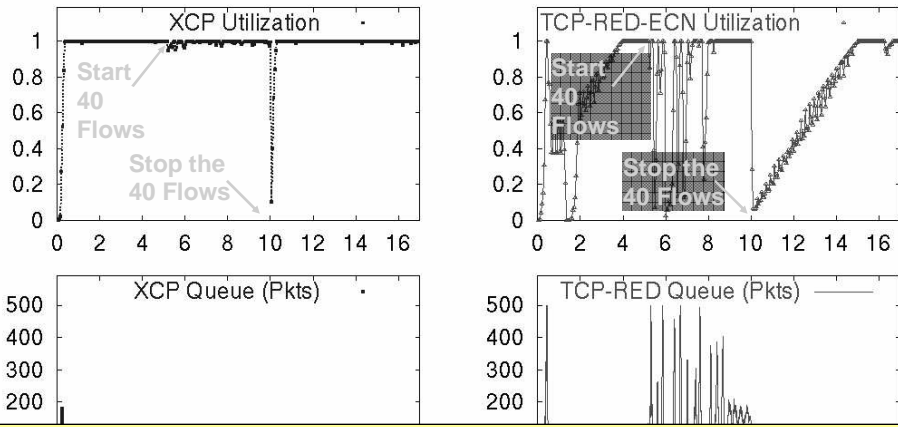
Utilization as a function of Bandwidth



Utilization as a function of Delay



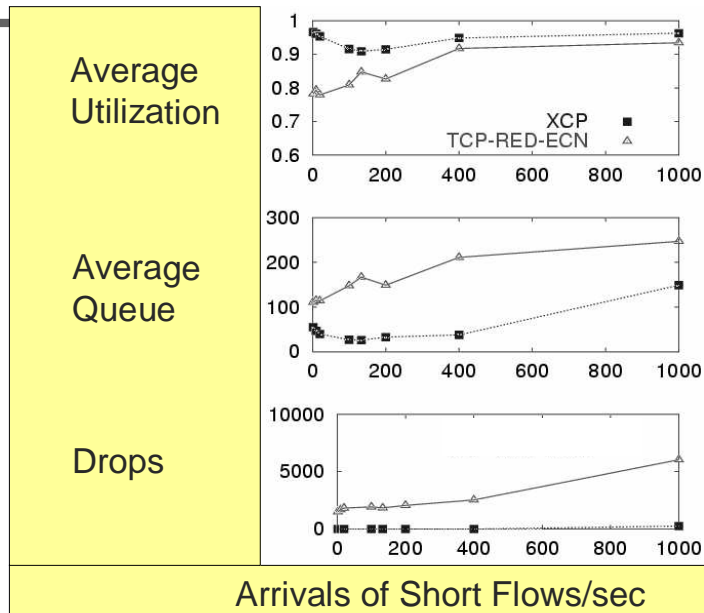
XCP Shows Faster Response than TCP



XCP shows fast response!

29

XCP Deals Well with Short Web-Like Flows



30

XCP is Fairer than TCP

