CS268: Beyond TCP Congestion Control

Kevin Lai
February 4, 2003

TCP Problems

- When TCP congestion control was originally designed in 1988:
  - Maximum link bandwidth: 10Mb/s
  - Users were mostly from academic and government organizations (i.e., well-behaved)
  - Almost all links were wired (i.e., negligible error rate)
- Thus, current problems with TCP:
  - High bandwidth-delay product paths
  - Selfish users
  - Wireless (or any high error links)

High Bandwidth-Delay Product Paths

- Motivation
  - 10Gb/s links now common in Internet core
    - as a result of Wave Division Multiplexing (WDM), link bandwidth 2x/9 months
  - Some users have access to and need for 10Gb/s end-to-end
    - e.g., very large scientific, financial databases
  - Satellite/Interplanetary links have a high delay
- Problems
  - slow start
  - Additive increase, multiplicative decrease (AIMD)
- Congestion Control for High Bandwidth-Delay Product Networks. Dina Katabi, Mark Handley, and Charlie Rohrs. Proceedings on ACM Sigcomm 2002.

Slow Start

- TCP throughput controlled by congestion window (cwnd) size
- In slow start, window increases exponentially, but may not be enough
- example: 10Gb/s, 200ms RTT, 1460B payload, assume no loss
  - Time to fill pipe: 18 round trips = 3.6 seconds
  - Data transferred until then: 382MB
  - Throughput at that time: 382MB / 3.6s = 850Mb/s
  - 8.5% utilization → not very good
- Loose only one packet → drop out of slow start into AIMD (even worse)

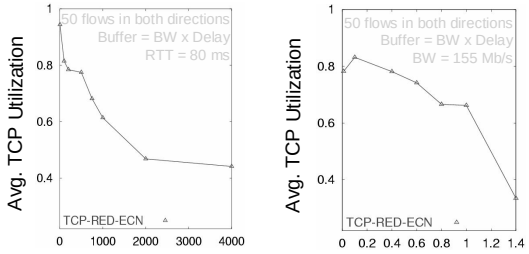
AIMD

- In AIMD, cwnd increases by 1 packet/ RTT
- Available bandwidth could be large
  - e.g., 2 flows share a 10Gb/s link, one flow finishes → available bandwidth is 5Gb/s
  - e.g., suffer loss during slow start → drop into AIMD at probably much less than 10Gb/s
- time to reach 100% utilization is proportional to available bandwidth
  - e.g., 5Gb/s available, 200ms RTT, 1460B payload → 17,000s

5

Simulation Results

Shown analytically in [Low01] and via simulations



6

Proposed 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

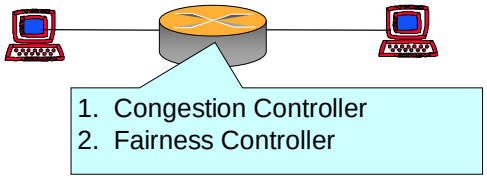
7

Characteristics of 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,…

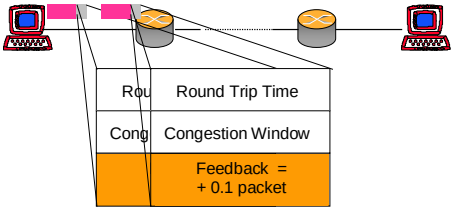
8

XCP: An eXplicit Control Protocol



9

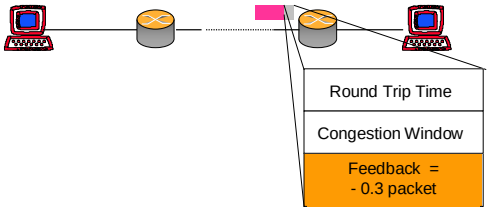
How does XCP Work?



Congestion Header

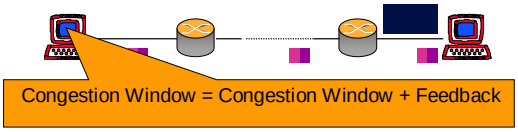
10

How does XCP Work?



11

How does XCP Work?



XCP extends ECN and CSFQ

Routers compute feedback without any per-flow state

12

How Does an XCP Router Compute the Feedback?

Congestion Controller

Goal: Matches input traffic to link capacity & drains the queue

Looks at aggregate traffic & queue

MIMD

Algorithm:

Aggregate traffic changes by Δ

Δ ~ Spare Bandwidth

Δ ~ - Queue Size

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

Fairness Controller

Goal: Divides Δ between flows to converge to fairness

Looks at a flow's state in Congestion He

AIMD

Algorithm:

If Δ > 0 ⇒ Divide Δ equally between flows

If Δ < 0 ⇒ Divide Δ between flows proportionally to their current rates

13

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:

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

No Parameter Tuning

Fairness Controller

Algorithm:

If Δ > 0 ⇒ Divide Δ equally between flows

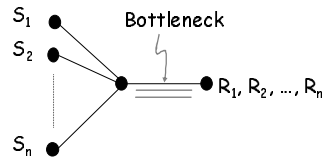
If Δ < 0 ⇒ Divide Δ between flows proportionally to their current rates

Need to estimate number of flows *N*

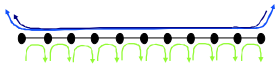
$$N = \sum_{pkts\ in\ T} \frac{1}{T \times (Cwnd_{pkt} / RTT_{pkt})}$$

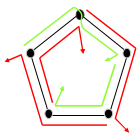
No Per-Flow State

Subset of Results



Similar behavior over:

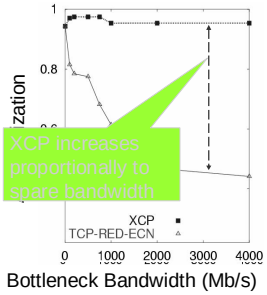




15

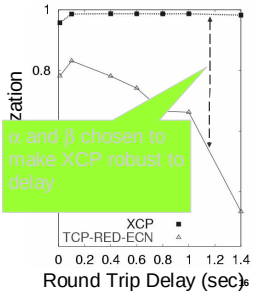
XCP Remains Efficient as Bandwidth or Delay Increases

Utilization as a function of Bandwidth



Bottleneck Bandwidth (Mb/s)

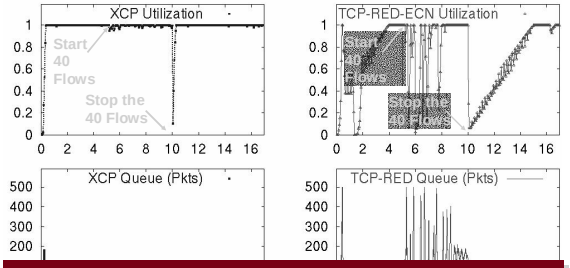
Utilization as a function of Delay



Round Trip Delay (sec)

16

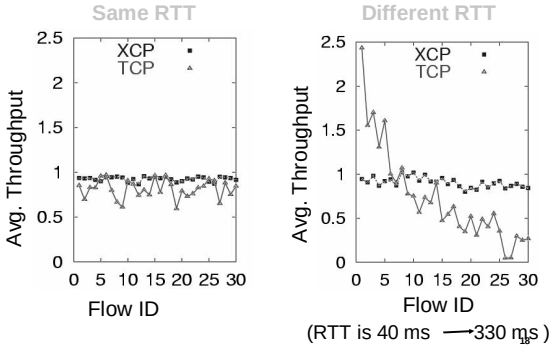
XCP Shows Faster Response than TCP



XCP shows fast response!

17

XCP is Fairer than TCP



18

XCP Summary

- XCP
  - Outperforms TCP
  - Efficient for any bandwidth
  - Efficient for any delay
  - Scalable (no per flow state)
- Benefits of Decoupling
  - Use MIMD for congestion control which can grab/release large bandwidth quickly
  - Use AIMD for fairness which converges to fair bandwidth allocation

19

Selfish Users

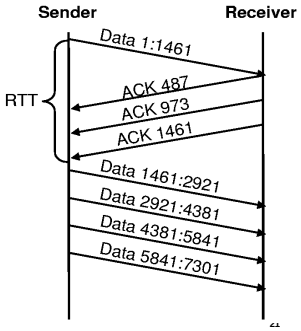
- Motivation
  - Many users would sacrifice overall system efficiency for more performance
  - Even more users would sacrifice fairness for more performance
  - Users can modify their TCP stacks so that they can receive data from a normal server at an un-congestion controlled rate.
- Problem
  - How to prevent users from doing this?
  - General problem: How to design protocols that deal with lack of trust?

- TCP Congestion Control with a Misbehaving Receiver. Stefan Savage, Neal Cardwell, David Wetherall and Tom Anderson. ACM Computer Communications Review, pp. 71-78, v 29, no 5, October, 1999.
- Robust Congestion Signaling. David Wetherall, David Ely, Neil Spring, Stefan Savage and Tom Anderson. IEEE International Conference on Network Protocols, November 2001

20

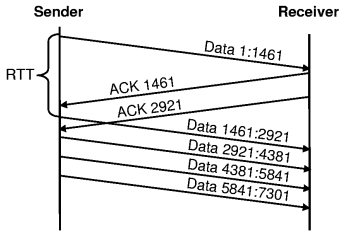
Ack Division

- Receiver sends multiple, distinct acks for the same data
- Max: one for each byte in payload
- Smart sender can determine this is wrong



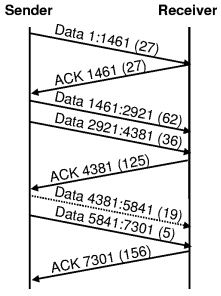
Optimistic Acking

- Receiver acks data it hasn't received yet
- No robust way for sender to detect this on its own



Solution: Cumulative Nonce

- Sender sends random number (nonce) with each packet
- Receiver sends cumulative sum of nonces
- if receiver detects loss, it sends back the last nonce it received
- Why cumulative?

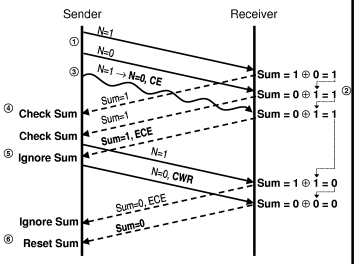


ECN

- Explicit Congestion Notification
  - Router sets bit for congestion
  - Receiver should copy bit from packet to ack
  - Sender reduces cwnd when it receives ack
- Problem: Receiver can clear ECN bit
  - or increase XCP feedback
- Solution: Multiple unmarked packet states
  - Sender uses multiple unmarked packet states
  - Router sets ECN mark, clearing original unmarked state
  - Receiver returns packet state in ack
    - receiver must guess original state to unmark packet

ECN

- Receiver must either return ECN bit or guess nonce
- More nonce bits → less likelihood of cheating
  - 1 bit is sufficient



25

Selfish Users Summary

- TCP allows selfish users to subvert congestion control
- Adding a nonce solves problem efficiently
  - must modify sender and receiver
- Many other protocols not designed with selfish users in mind, allow selfish users to lower overall system efficiency and/or fairness
  - e.g., BGP

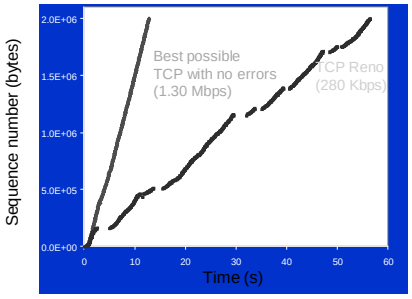
26

Wireless

- Wireless connectivity proliferating
  - Satellite, line-of-sight microwave, line-of-sight laser, cellular data (CDMA, GPRS, 3G), wireless LAN (802.11a/b), Bluetooth
  - More cell phones than currently allocated IP addresses
- Wireless → non-congestion related loss
  - signal fading: distance, buildings, rain, lightning, microwave ovens, etc.
- Non-congestion related loss →
  - reduced efficiency for transport protocols that depend on loss as implicit congestion signal (e.g. TCP)

27

Problem



2 MB wide-area TCP transfer over 2 Mbps Lucent WaveLAN
(from Hari Balakrishnan)

28

Solutions

- Modify transport protocol
- Modify link layer protocol
- Hybrid

29

Modify Transport Protocol

- Explicit Loss Signal
  - Distinguish non-congestion losses
  - Explicit Loss Notification (ELN) [BK98]
  - If packet lost due to interference, set header bit
  - Only needs to be deployed at wireless router
  - Need to modify end hosts
  - How to determine loss cause?
  - What if ELN gets lost?

30

Modify Transport Protocol

- TCP SACK
  - TCP sends cumulative ack only→cannot distinguish multiple losses in a window
  - Selective acknowledgement: indicate exactly which packets have not been received
  - Allows filling multiple “holes” in window in one RTT
  - Quick recovery from a burst of wireless losses
  - Still causes TCP to reduce window

31

Modify Link Layer

- How does IP convey reliability requirements to link layer?
  - not all protocols are willing to pay for reliability
  - Read IP TOS header bits(8)?
    - must modify hosts
  - TCP = 100% reliability, UDP = doesn't matter?
    - what about other degrees?
  - consequence of lowest common denominator IP architecture
- Link layer retransmissions
  - Wireless link adds seq. numbers and acks below the IP layer
  - If packet lost, retransmit it
  - May cause reordering
  - Causes at least one additional link RTT delay
  - Some applications need low delay more than reliability e.g. IP telephony
  - easy to deploy

32

Modify Link Layer

- Forward Error Correction (FEC) codes
  - k data blocks, use code to generate n>k coded blocks
  - can recover original k blocks from any k of the n blocks
  - n-k blocks of overhead
  - trade bandwidth for loss
  - can recover from loss in time independent of link RTT
    - useful for links that have long RTT (e.g. satellite)
  - pay n-k overhead whether loss or not
    - need to adapt n, k depending on current channel conditions

33

Hybrid

- Indirect TCP [BB95]
  - Split TCP connection into two parts
  - regular TCP from fixed host (FH) to base station
  - modified TCP from base station to mobile host (MH)
  - base station fails?
  - wired path faster than wireless path?
- TCP Snoop [BSK95]
  - Base station snoops TCP packets, infers flow
  - cache data packets going to wireless side
  - If dup acks from wireless side, suppress ack and retransmit from cache
  - soft state
  - what about non-TCP protocols?
  - what if wireless not last hop?

34

Conclusion

- Transport protocol modifications not deployed
  - SACK was deployed because of general utility
- Cellular, 802.11b
  - link level retransmissions
  - 802.11b: acks necessary anyway in MAC for collision avoidance
  - additional delay is only a few link RTTs (<5ms)
- Satellite
  - FEC because of long RTT issues
- Link layer solutions give adequate, predictable performance, easily deployable

35