Congestion Control In High Bandwidth-Delay Nets [Katabi02a]

CSci551: Computer Networks
SP2006 Thursday Section

John Heidemann

Context

- limitations of TCP over large bw-delay links
  - need large window, and slow may take a while, recovery from pkt loss takes a while
- ~1990: routers could barly keep up
  - things like FQ cannot be used (even today) because of costs of per flow state
  - things like R&J used very simple feedback to keep routers simple
  - what about today?
- current status:
  - SIGCOMM paper was just simulations
  - ISI and others are now implementing
    - some slides and much discussion from Aaron Falk and Ted Faber at ISI
  - eventually may be IETF standard

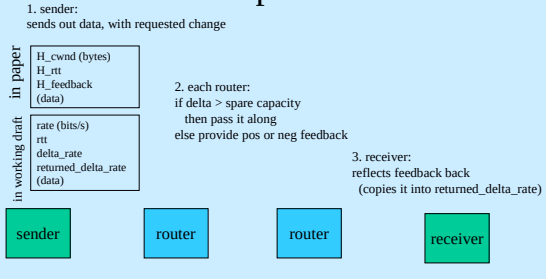
Key ideas

- explict feedback from routers to senders
  - talk about rates, not just congested/not
  - takes advantage of greater router compute power (than in 1990)
    - but still avoid per-flow state
- separate fairness and utilization
  - also separate congestion signals (congestion loss) from packet loss (corruption loss)
- protocol for high bw-delay links

What is XCP?

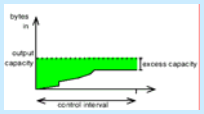
- congestion control protocol
  - intended to layer over some transport protocol like TCP
- components in end-points *and* routers
  - use *explicit* signaling
  - and avoid per-flow state in the routers
- goals:
  - high utilization
  - good fairness
  - low queueing delay

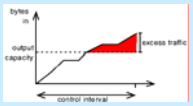
XCP Operation



Router Feedback: Utilization

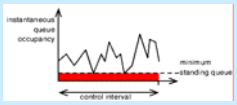
goals:
• keep utilization high (nearly 1)
• keep queues small
=> *efficiency controller*
(why? keep utilization to send as much as possible, but avoid queues to handle bursts)





$$\phi = \alpha \cdot d \cdot S - \beta \cdot Q_s$$

d: mean RTT
S: spare bandwidth
Q: mean queue size
α, β: constants



φ: desired router feedback over all flows

Router Feedback: Fairness

goals:
• provide min-max fairness to each flow
=> *fairness controller*

mechanisms:
• AIMD
• if φ>0, increase all evenly;
  if φ<0, decrease relative to usage
• and always *shuffle* some bw

and do all this without any per-flow state (!)

•why AIMD? promote stability

•why increase evenly? bring small flows up to the mean

•why decrease in proportion? bring large flows down quickly

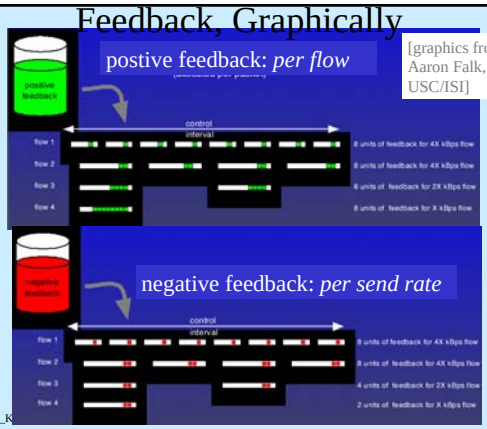
•why shuffle? to handle steady state unfair allocations

• you can get unfair allocations because flows come and go

Begin Digression

• digression on how this *actually* works

Feedback, Graphically



[graphics from Aaron Falk, USC/ISI]

Goal: Feedback Without Per-Flow State

• feedback is easy with per-flow state, but that's costly
• how without per-flow state?
  – router has "pool" of feedback (residue_pos_fbk, residue_neg_fbk)
  – estimate these each control period (= mean RTT)
  – allocation:
    • each packet is labeled w/RTT and cwnd (or rate)
    => can compute expected number of pkts per RTT and then give feedback to each packet as it arrives

Understanding XCP Feedback

• if you have per-flow state, it's easy:
  – $T_p := h + \max(\varphi, 0)$: positive feedback
  – $T_n := h + \max(-\varphi, 0)$: negative feedback
• over each control interval d you want to give out T_p and T_n
• assume per-flow state, so you know N flows and M packets
• then $T_{p,flow} = T_p / N$; $T_{n,pkt} = T_n / M$
• but how to do this *without* explicit N and M?

Idea

• each packet carries $cwnd_i$, rtt_i, so we know that flow's rate
  1. sum these from the prior ctl interval
  2. use values in pkt to estimate pkt's contribution to total traffic
  3. assume last ctl interval is like this ctl interval and pkts don't lie
• ξ_p , ξ_n become normalization constants to dole out T_p , T_n per pkt

Challenges

- different packet sizes
- different flow RTTs
- control interval != RTT

- also
  - traffic may (or *will*) change
  - potentially, users could lie

Negative Feedback

- idea: penalize relative to rate
  - Δ throughput$_i$ ~ throughput$_i$
  - Δ cwnd$_i$ / rtt$_i$ ~ cwnd$_i$ / rtt$_i$
  - Δ cwnd$_i$ ~ cwnd$_i$ (if const. rtt$_i$)
- if T_n is total penalty, then per pkt penalty n_i is just T_n / E[total pkts in d]

Negative Feedback: Complications

- different size pkts? $n_i \sim s_i / \Sigma_{\forall \text{pkts in } d} s_i$
  - large pkt => more thrpt => more fdbk
- different RTTs? $n_i \sim \text{rtt}_i / d$ (scale by rtt)
  - short rtt => more pkts => more feedback
- from paper:
  - $n_i = s_i * \text{rtt}_i * \xi_n$
  - $\xi_n = T_n / (d * \Sigma_{\forall \text{pkts in } d} s_i)$

Putting it Together

$$n_i = T_n * \frac{\text{rtt}_i * s_i}{(d * \Sigma_{\forall \text{pkts in } d} s_i)}$$

packet weight

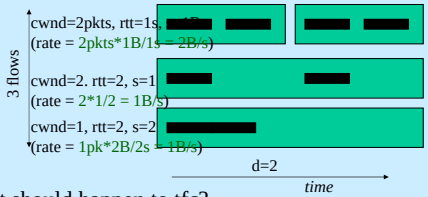
traffic weight

scale for rtt

scale for pkt size

- $n_i = s_i * \text{rtt}_i * \xi_n$
- $\xi_n = T_n / (d * \Sigma_{\forall \text{pkts in } d} s_i)$

Example of Negative Feedback



if Tn = 4B, what should happen to tfc?
cut in half
what is $\xi_n = T_n / (d * \Sigma_{\forall \text{pkts in } d} s_i)$
4B/(2s * 8B) = 1/2s

penalty to flow 1?
4* ½ = 2pkts

Positive Feedback

- idea: allocate per flow
  - Δ throughput$_i$ ~ constant
  - Δ cwnd$_i$ / rtt$_i$ ~ constant
  - Δ cwnd$_i$ ~ constant * rtt$_i$
- compare

$n_i = \xi_n * s_i * \text{rtt}_i$ $\quad$ $p_i = \xi_n * s_i * \text{rtt}_i^2 / \text{cwnd}_i$

$\xi_n = T_n / (d * \Sigma_{\forall \text{pkts in } d} s_i)$ $\quad$ $\xi_p = T_p / (d * \Sigma_{\forall \text{pkts in } d} (s_i \text{rtt}_i / \text{cwnd}_i))$

basic change: add a factor for rate

(end of digression)

So Why is XCP Cool?

- complete feedback from routers (target rate)
  - much more than just "slow down"
- allows flows to reach high utilization quickly
  - no slow-start to find target; no aggressive window halving like TCP
  - both are really important for high bw-delay flows
- and does so without per-flow state
  - so is probably achievable in routers

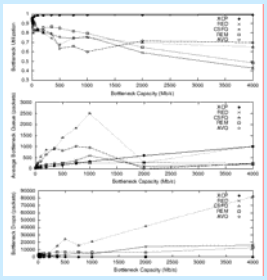
Does it Work?

- control theoretic analysis
  - see paper
  - but doesn't consider feedback delay (the hard part!)
- extensive simulations
  - compare XCP, TCP w/RED, REM, AVQ, CSFQ
  - consider capacity, feedback delay, number of flows, size of flows, fairness, topology, dynamics

Evaluation vs. Bandwidth

[Katabi02a, figure 4]

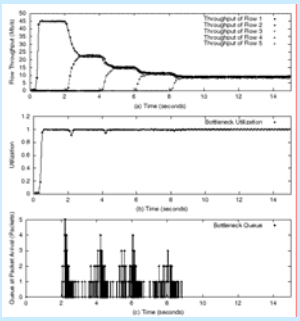
- vary bandwidth, holding RTT=80ms
- XCP has excellent utilization, very low queueing and drops
- why?
  - explicit feedback
    - can quickly react to changes, positive or negative



Dynamics

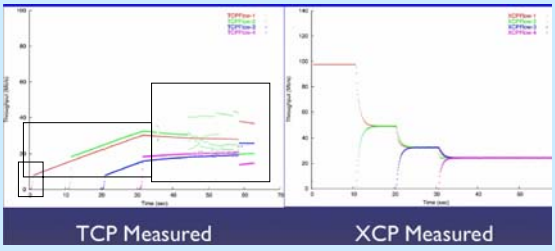
[Katabi02a, figure 10]

- as flows are added
  - quickly reach fair share (top)
  - utilization stays high (middle)
  - queues are fairly short (bottom)



Experimental Data

Data from Falk et al at ISI; implementation over FreeBSD; www.isi.edu/isi-xcp



xxx xxx

Other questions/observations?

- compatibility?
  - completely incompatible…needs new end hosts and new routers
  - maybe deploy in new networks (like satellites)
  - or maybe do it an an overlay network (if you have dedictated bw in the overlay, and can get feedback from routers in the overlay)
- compare it to TCP
  - TCP friendlies