TCP modeling

Performance Modeling Lecture #7

Slides adapted from Kurose and Ross

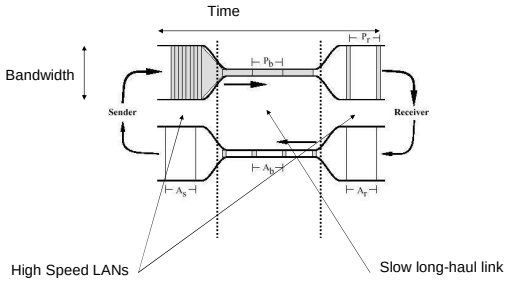
Outline

- Introduction
- TCP Modelling
- Fairness
- The TFRC equation
- HTTP modelling

Introduction

- TCP modelling can be used to determine:
  - TCP throughput
  - Application layer performance for HTTP etc.
  - Response times

TCP self-clocking in equilibrium



TCP throughput

- What's the average throughout ot TCP as a function of window size and RTT?
  - Ignore slow start
- Let W be the window size when loss occurs.
- When window is W, throughput is W/RTT
- Just after loss, window drops to W/2, throughput to W/2RTT.
- Average throughout: .75 W/RTT

Problems for high-speed networks

- Example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput
- Requires window size W = 83,333 in-flight segments
- Throughput in terms of loss rate:

$$\frac{1.22 \cdot MSS}{RTT\sqrt{L}}$$

- ➜ L = $2 \cdot 10^{-10}$ *Wow*
- New versions of TCP for high-speed needed!

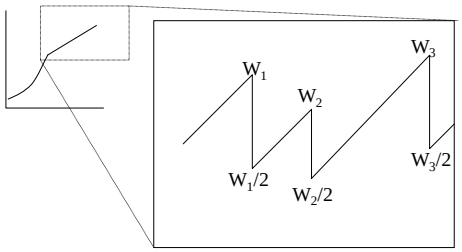
Reno TCP

- Increase congestion window size
  - slow start (cwnd < ssh): cwnd += 1
  - steady state (cwnd ≥ ssh): cwnd += 1/cwnd
- Decrease congestion window size
  - duplicated acknowledges: cwnd = cwnd/2
  - timeout: cwnd = 1
    - ssh = cwnd/2

The Magical $(1/p)^{1/2}$

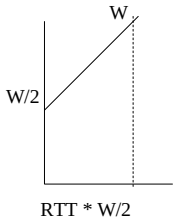
- Show in a simplified analysis
  - infinitely long TCP connections
    - only in the steady state
    - cwnd += 1 per RTT
  - no timeouts
    - only duplicated acknowledges
    - cwnd /= 2 per drop
- Average Bandwidth =
  - MSS/RTT * $(3/2p)^{1/2}$

Saw Tooth Behavior



Deriving BW

W: average tooth tip
W/2: average tooth dip



Total number of packets sent between two packet drops is:
$(W/2 + W) * (W/2) /2 = (3/8)W^2$

p: probability of packet loss
$(3/8)\ W^2 = 1/p$

$W = (8/3p)^{1/2}$
$BW = MSS * (3/8)\ W^2 / (RTT * W/2)$
$= MSS/RTT * (3/2p)^{1/2}$

Padhye's TCP model

- Based on TCP steady-state response function
  - gives upper bound for transmission rate T (bytes/sec):

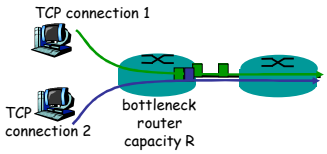
$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)}$$

s: packet size
R: rtt
t_{RTO}: TCP retransmit timeout
p: steady-state loss event rate (the difficult part!)

- well known example: **TFRC** - TCP-friendly rate control protocol
  - smooth sending rate

TCP Fairness

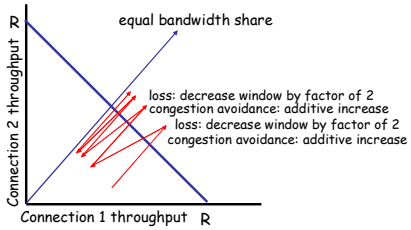
Fairness goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K



Why is TCP fair?

Two competing sessions:

- Additive increase gives slope of 1, as throughout increases
- multiplicative decrease decreases throughput proportionally



Fairness (more)

Fairness and UDP

- Multimedia apps often do not use TCP
  - do not want rate throttled by congestion control
- Instead use UDP:
  - pump audio/video at constant rate, tolerate packet loss
- Research area: TCP friendly

Fairness and parallel TCP connections

- nothing prevents app from opening parallel cnctions between 2 hosts.
- Web browsers do this
- Example: link of rate R supporting 9 cnctions;
  - new app asks for 1 TCP, gets rate R/10
  - new app asks for 11 TCPs, gets R/2 !

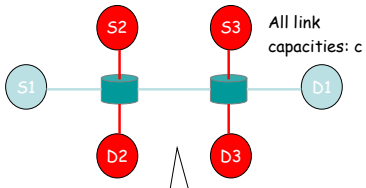
Fairness

- ATM ABR: Max-Min-fairness
  - "A (..) allocation of rates is max-min fair iff an increase of any rate (..) must be at the cost of a decrease of some already smaller rate."
  - One resource: mathematical definition satisfies "general" understanding of fairness - resource is divided equally among competitors
  - *Usually* requires knowledge of flows in routers (switches) - scalability problem!
- Internet:
  - TCP dominant, but does not satisfy Max-Min-fairness criterion!
  - Ack-clocked - flows with shorter RTT react sooner (slow start, ..) and achieve better results
  - Therefore, Internet definition of fairness: TCP-friendliness

  "A flow is TCP-compatible (TCP-friendly) if, in steady state, it uses no more bandwidth than a conformant TCP running under comparable conditions."

Proportional Fairness

F. Kelly: Network should solve a global optimization problem (maximize log utility function)



Max-Min-fairness suboptimal:
S1 = S2 = S3 = c/2

Proportional fairness:

"An allocation of rates $\vec{x}$ is proportionally fair iff, for any other (..) allocation $\vec{y}$, we have: $\sum_{s=1}^{S} \frac{y_s - x_s}{x_s} <= 0$ "

Proportionally fair allocation:
S1 = c/3, S2 = S3 = 2c/3

(***roughly*** *approximated by AIMD!)*

Issues with TCP-friendliness

- TCP regularly increases the queue length and causes loss
  ⇒ detect congestion when it is already (ECN: almost) too late!
  - possible to have more throughput with smaller queues and less loss
    ... but: exceed rate of TCP under similar conditions ⇒ not TCP-friendly!
- What if I send more than TCP *in the absence* of competing TCP's?
  - can such a mechanism exist?
  - yes! TCP itself, with max. window size = bandwidth * RTT
  - Does this mean that TCP is not TCP-friendly?
- Details missing from the definition:
  - parameters + version of „conformant TCP"
  - duration! short TCP flows are different than long ones
- TCP-friendliness = compatibility of new mechanisms with old mechanism
  - there was research since the 80's! e.g. new knowledge about network measurements
- TCP rate depends on RTT - how does this relate to „fairness"?

Delay modeling

Q: How long does it take to receive an object from a Web server after sending a request?

Ignoring congestion, delay is influenced by:

- TCP connection establishment
- data transmission delay
- slow start

Notation, assumptions:

- Assume one link between client and server of rate R
- S: MSS (bits)
- O: object size (bits)
- no retransmissions (no loss, no corruption)

Window size:

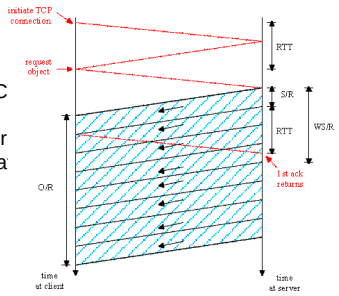
- First assume: fixed congestion window, W segments
- Then dynamic window, modeling slow start

Fixed congestion window (1)

First case:

WS/R > RTT + S/R: AC for first segment in window returns befor window's worth of da sent

delay = 2RTT + O/R

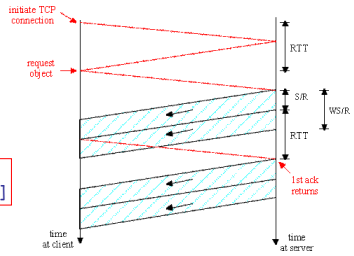


Fixed congestion window (2)

Second case:

- WS/R < RTT + S/R: wait for ACK after sending window's worth of data sent

delay = 2RTT + O/R + (K-1)[S/R + RTT - WS/R]



TCP Delay Modeling: Slow Start (1)

Now suppose window grows according to slow start

Will show that the delay for one object is:

$$Latency = 2RTT + \frac{O}{R} + P\left[RTT + \frac{S}{R}\right] - (2^P - 1)\frac{S}{R}$$

where *P* is the number of times TCP idles at server:

$$P = \min\{Q, K-1\}$$

- where Q is the number of times the server idles if the object were of infinite size.
- and K is the number of windows that cover the object.

TCP Delay Modeling: Slow Start (2)

Delay components:

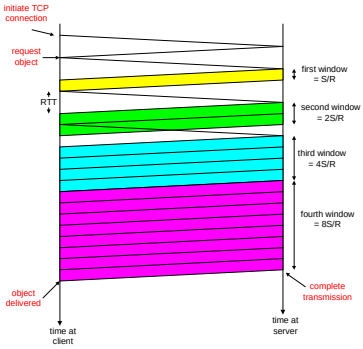
- 2 RTT for connection estab and request
- O/R to transmit object
- time server idles due to slow start

Server idles:
P = min{K-1,Q} times

Example:

- O/S = 15 segments
- K = 4 windows
- Q = 2
- P = min{K-1,Q} = 2

Server idles P=2 times



TCP Delay Modeling (3)

$\frac{S}{R} + RTT$ = time from when server starts to send segment until server receives acknowledgement

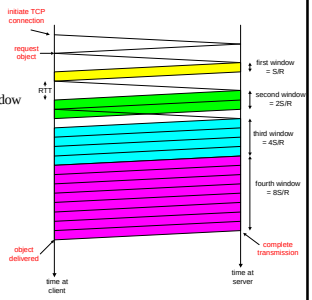
$2^{k-1}\frac{S}{R}$ = time to transmit the kth window

$\left[\frac{S}{R} + RTT - 2^{k-1}\frac{S}{R}\right]^+$ = idle time after the *k*th window

$$\text{delay} = \frac{O}{R} + 2RTT + \sum_{p=1}^{P} idleTime_p$$

$$= \frac{O}{R} + 2RTT + \sum_{k=1}^{P}\left[\frac{S}{R} + RTT - 2^{k-1}\frac{S}{R}\right]$$

$$= \frac{O}{R} + 2RTT + P\left[RTT + \frac{S}{R}\right] - (2^P - 1)\frac{S}{R}$$



TCP Delay Modeling (4)

Recall K = number of windows that cover object

How do we calculate K ?

$$K = \min\{k : 2^0 S + 2^1 S + \cdots + 2^{k-1} S \geq O\}$$

$$= \min\{k : 2^0 + 2^1 + \cdots + 2^{k-1} \geq O/S\}$$

$$= \min\{k : 2^k - 1 \geq \frac{O}{S}\}$$

$$= \min\{k : k \geq \log_2(\frac{O}{S} + 1)\}$$

$$= \left\lceil \log_2(\frac{O}{S} + 1) \right\rceil$$

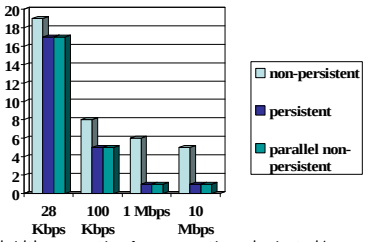
Similarly, for Q: $Q = \left\lfloor \log_2\left(1 + \frac{RTT}{S/R}\right) \right\rfloor + 1$

HTTP Modeling

- Assume Web page consists of:
  - *1* base HTML page (of size *O* bits)
  - *M* images (each of size *O* bits)
- Non-persistent HTTP:
  - *M+1* TCP connections in series
  - *Response time = (M+1)O/R + (M+1)2RTT + sum of idle times*
- Persistent HTTP:
  - *2 RTT* to request and receive base HTML file
  - *1 RTT* to request and receive M images
  - *Response time = (M+1)O/R + 3RTT + sum of idle times*
- Non-persistent HTTP with X parallel connections
  - Suppose M/X integer.
  - 1 TCP connection for base file
  - M/X sets of parallel connections for images.
  - *Response time = (M+1)O/R + (M/X + 1)2RTT + sum of idle times*

HTTP Response time (in seconds)

RTT = 100 msec, *O* = 5 Kbytes, M=10 and X=5

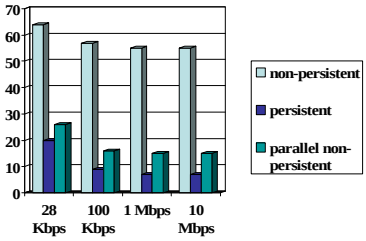


For low bandwidth, connection & response time dominated by transmission time.

Persistent connections only give minor improvement over parallel connections.

HTTP Response time (in seconds)

RTT =1 sec, *O* = 5 Kbytes, M=10 and X=5



For larger RTT, response time dominated by TCP establishment & slow start delays. Persistent connections now give important improvement: particularly in high delay•bandwidth networks.