CIS 457 - Data Communications

Nathan Bowman

Images taken from Kurose and Ross book

TCP Congestion Control -- Performance

Saw how TCP congestion control operates

Next, take a *very* simplified look at how congestion control affects performance

First, consider cost to brief connections, such as performing web search

According to your book, returning search results requires server to send about 3 TCP windows of data

Also need to consider overhead of TCP connection establishment in this case

So, takes about 4 RTT to acquire search results, which is a lot of overhead

Overall, combination of handshake and slow start can make for poor perceived performance from users when connection is established

Improving delay requires lowering RTT (i.e., latency), which means servers must be close to clients

However, companies cannot put data centers everywhere

One way to get around this is called **TCP splitting**, and is somewhat similar to caching, but not quite the same

Companies can put many small "front-end" servers over wide geographic area so client likely to be near one (low RTT)

Front-end servers maintain open, high transmission-rate connection with back-end servers in data center

When client requests results from front end, front end sends to back end over already-established connection with large cwnd, allowing it to bypass handshake and slow start

Communication between front and back ends takes 1 RTT (to back end)

Client and front end still need 4 RTT (to front end) delay, but that RTT is different and much smaller

Disregarding client-to-front-end RTTs, have essentially reduced time by factor of 4

Next, we examine long-term performance of TCP

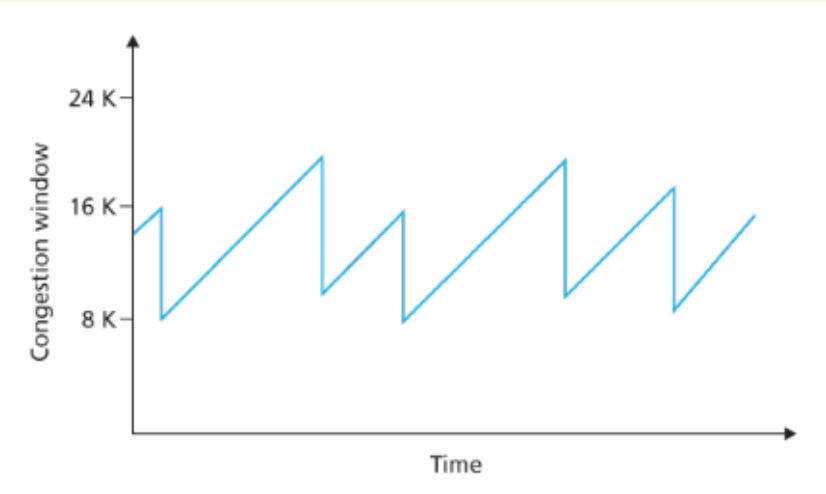
Ignore slow start because hosts will not spend much time there due to exponential growth

Focus mainly on congestion avoidance phase

When congestion not an issue, have linear increase of 1 MSS/RTT

Each triple duplicate ACK results in halving `cwnd`

This pattern known as **additive-increase, multiplicative-decrease** (AIMD)


24 K
16 K
8 K
Congestion window
Time

AIMD results in saw-tooth pattern

Visual demonstration that TCP is constantly probing for maximum possible bandwidth

Next, we compute (very simple approximation of) long-term throughput of congestion avoidance stage

Already saw that rate of sending depends on RTT and current congestion window

If current window is `w` bytes, current rate roughly `w`/RTT

Rate increases linearly by 1 MSS / RTT until loss occurs at some rate `W`

Assume W and RTT remain constant for given connection

Instantaneous rate varies from (W/2) / RTT to W / RTT

Since rate changes *linearly* between these, average rate simply

0.75 W / RTT

Other variants of TCP exist

For example, TCP Vegas attempts to detect congestion *before* packet loss by directly measuring RTT

When congestion detected this way, decrease linearly instead of multiplicatively

How do people come up with these?

According to your book, protocols originally developed using "engineering insight and experimentation"

I.e., try it and see what happens

Modern networking has more sound theoretical basis, which generally shows that TCP designers did a good job

TCP and UDP are primary transport-layer protocols on internet, but others exist

Newer protocols can have nice features or improved performance

However, difficult to get adoption when established protocols already so effective

Book mentions a few other transport protocols that have seen some adoption: DCCP, QUIC, DCTCP, SCTP