Lecture 6: 2/9/2010

Transport and Queuing

TCP: Reliable, in-order — sequence numbers
acknowledgements — for the last in sequence packet received
TCP also does congestion control — regulating sending
rate to match n/w capacity (or available resources)

E.g., fair queuing

Why can't the n/w do it?
No proper end-host support
   can lead to congestion collapse
where the amount of useful work
the n/w is doing becomes vanishingly
small

Note: end-hosts can send as fast as
their NIC allows

Flow B: 25% useful work!

Congestion control: figuring of what rate to send to fit n/w
capacity optimally, given traffic of all other flows.

Challenges: 1. Quickly figure out available head room
   and match it
2. Back off when available capacity lowers or
   others want to use
3. Fine time-scale adaptation

Various ways of achieving this.
TCP embodies one specific set of mechanisms.
Let's discuss the mechanisms in TCP and the reasoning behind them.

1. **Slow-start** → @establish ACK clocking to help connection maintain equilibrium.
   - figure out available capacity
     - (or some rough estimate of it)
   - MAX Total available capacity = BW x delay + Buffering = BW x D + B
   - E exponential search, from 0 to 2(BW x D + B)

   Overshoot capacity by almost 10x twice.
   - CWND tracks exponential growth → # packets outstanding variable as through as last estimate of capacity
   - Establish ACK clocking:
     - at the end of search in 1 above → bunch of losses → time out → loose ACK clocking

   Another slow-start to quickly regain ack-clocking
     - up to throughput.

   After that explore slowly between throughput and true available capacity. Therefore
     - CWND → guides the search

2. **Congestion Avoidance**: and control
   - T0
     - amount of data outstanding over time
     - CP = decrease function: which next sending rate to explore
     - decrease function: How to adjust sending rate
choices driven by fairness and efficiency stability criteria.

\[ R \leftarrow \alpha R + (1-\alpha) N \]
\[ R \varepsilon \text{ EMA} \]
\[ R \rightarrow R_f + D \varepsilon \]

Two sender

\[ a_f = 2 \]
\[ a_e = 1 \]
\[ N_f = 1/2 \]

Go the overall algorithm.

\[ \text{Loose grain} \rightarrow \text{RTO} \rightarrow c_s \rightarrow R_f \rightarrow \text{RTO} \]

3. Feedback from timeouts, losses and Retransmission?

\[ \text{Duplicate acknowledgments to figure out} \]
\[ \text{loss} \rightarrow \text{fast retransmit} \]

Waiting for timeout and half window full of packets

\[ \text{to empty} \rightarrow \text{loss of Ack clocking} \]

Using incoming Retrks to check out packets

\[ \text{for every other Ack} \] (RENO).