Congestion Avoidance and Control


Fixes to TCP in BSD
Handwaving arguments
Less rigorous math
Lots of “magical” hacks

1986

Argentina won the World Cup.
Challenger exploded.
Internet had a congestion collapse!

TCP throughput from LBL to UC Berkeley (two hops) dropped from 32K bps to 40 bps.
Congestion Collapse:
sender sends too fast
routers delay/drop packets
sender retransmit
no useful data getting through

Observation: a TCP connection should obey
Conservation of Packets

In equilibrium state, a new packet is not inserted until an old packet leaves.

I. Getting to the equilibrium state

Equilibrium state: self-clocking
How to start the ‘clock’?

Slow Start

Add a new variable $cwnd$.
Start/Restart: $cwnd = 1$.
Upon receiving ACK, $cwnd++$.
Send at most $\min(cwnd, rwin)$.

Never send more than 2x the max possible rate.

(previously 200x is possible!)
In equilibrium state, a new packet is not inserted until an old packet leaves.

2. Conservation at Equilibrium

Something’s wrong with TCP timer

TCP (RFC793)

\[ R_i \leftarrow (1 - \alpha) R_{i-1} + (\alpha) M_i \]
\[ RTO_i \leftarrow \beta R_i \]

R: smoothed RTT
M: measured RTT
RTO: timeout value

Variation in RTT is inversely proportional to (1 - load)
$\beta = 2$ (recommended) tolerates only 30% load

**Idea:** estimate the variation and use in calculating RTO

Measuring Variation

- **variance:** costly (need to square)
- **mean error:** simpler

---

\[
R_i \leftarrow (1 - \alpha)R_{i-1} + (\alpha)M_i \\
R_i \leftarrow R_{i-1} + \alpha(M_i - R_{i-1}) \\
V_i \leftarrow V_{i-1} + \alpha(|M_i - R_{i-1}| - V_{i-1}) \\
RTO_i \leftarrow R_i + kV_i
\]
To prevent spurious timeout,

\[ RTO_i > R_{i+1} \]

To pick a value of \( k \), consider bandwidth-dominated link.

\[ R \text{ doubles each round during slow-start.} \]

\[
\begin{align*}
RTO_i & > R_{i+1} \\
R_i + kV_i & > 2R_i \\
R_i + k(R_i - R_{i-1}) & > 2R_i \\
R_i + k(R_i - \frac{1}{2}R_i) & > 2R_i \\
k\left(\frac{1}{2}\right) & > 1 \\
k & > 2
\end{align*}
\]

\[ RTO_i = R_i + 4V_i \]
Idea: adjust cwnd when congestion happens

Assume: congestion leads to packet loss, leads to timeout.

On timeout, cwnd /= 2
On ACK, cwnd += 1/cwnd

3. Moving towards new equilibrium when path changes
Why drop by half?
1. Slow-start:
   we know R/2 works
2. Steady state:
   a new flow probably?