Congestion Avoidance and Control

Van Jacobson
Michael J. Karles

SIGCOMM ‘88
Outline:

**INTRODUCTION:**
Conservation of packets principle in TCP

<table>
<thead>
<tr>
<th>PROBLEM</th>
<th>SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection doesn’t get to equilibrium</td>
<td>Getting to equilibrium: Slow-start</td>
</tr>
<tr>
<td>Injecting a new packet before an old packet has exited</td>
<td>Conservation at equilibrium: Round-trip timing</td>
</tr>
<tr>
<td>Equilibrium can’t be reached due to resource limits along the path</td>
<td>Adapting to the path: Congestion avoidance</td>
</tr>
</tbody>
</table>
Conservation of packets in TCP

At equilibrium: inject packet into network only when one is removed

- Rate control by sliding window:
  - self clocking, adjusted to bandwidth
  - wide dynamic range
  - transmission is smooth, once it is smooth

Issues:
- Needs to get to equilibrium, while:
  - avoiding sending burst of packets
  - avoiding retransmissions
Problem 1: getting to equilibrium

Slow-Start:
- Add a congestion window $cwnd$
- when restarting, set $cwnd=1$
- send $\min(cwnd, \text{window size})$ packets
- Increase $cwnd$ by 1 for each ACK received
Conservation at equilibrium

Problem: Injecting a new packet before an old packet has exited

- Retransmission timeout (RTO):
  - wait $\beta RTT$ before retransmitting
  - Needs RTT estimate $R_{n+1} \leftarrow \alpha R_n + (1 - \alpha)M_n$
  - Not estimating variance ($\beta = 2$)

Result: poor RTT estimate
(become critical under heavy load)

Solution: estimate $\beta$, consider the variance of RTT

Figure 5: Performance of an RFC793 retransmit timer
Figure 6: Performance of a Mean+Variance retransmit timer
Congestion avoidance

Main Problem: equilibrium cannot be reached

- Packet lost
  - usually due to insufficient buffer capacity in a congested network

Dropped packet = congested network

Problem 1: In a congested network queue length increases exponentially

Solution: multiplicative window size decrease on congestion

- Problem 2: The network does not tell us if the connection using less bandwidth than it can
  - Need to gradually increase bandwidth
  - Overestimating bandwidth is costly

Solution: additive widow size increase (by one packet per RTT)

1. On timeout set cwnd = cwnd/2
2. on each ACK set cwnd = cwnd + 1/cwnd
3. send min(receiverWnd, cwnd)