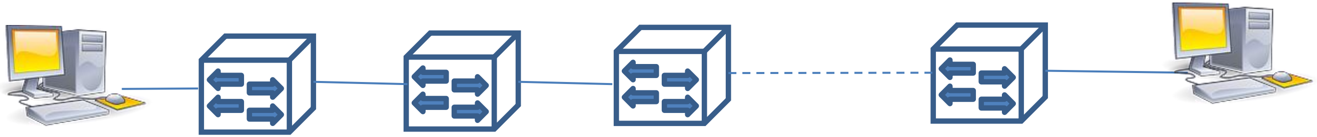
TCP Congestion Control Mechanisms

Kameswari Chebrolu

Seminal Paper: Congestion Avoidance and Control
by Van Jacobson and Michael J. Karels

Congestion Control: Challenge



- Need to estimate W (of sliding window) such that each flow gets its fair share
  - Estimate small → underutilization; Estimate large → Congestion
- W will vary over time
- Congestion Control: Preventing sources from sending too much data too fast and thereby ‘congest’ the network

Idea

- View network as a pipe
- Estimate Bandwidth-delay product (capacity) dynamically
  - Uses the variable Congestion Window (CW) to track it
- Use self clocking to pump packets into the network

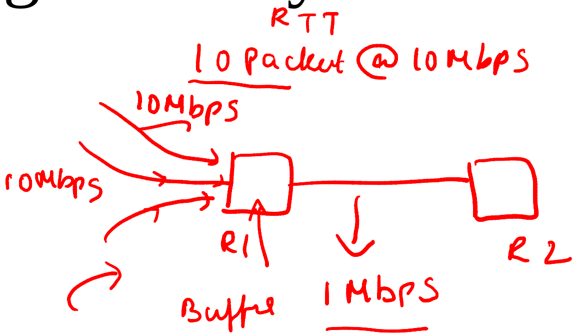
Approach

- Getting to Equilibrium → filling up the pipe
- Conservation at equilibrium
- Adapting to Path

Getting to Equilibrium

BDP – 1, 10, 100, 200

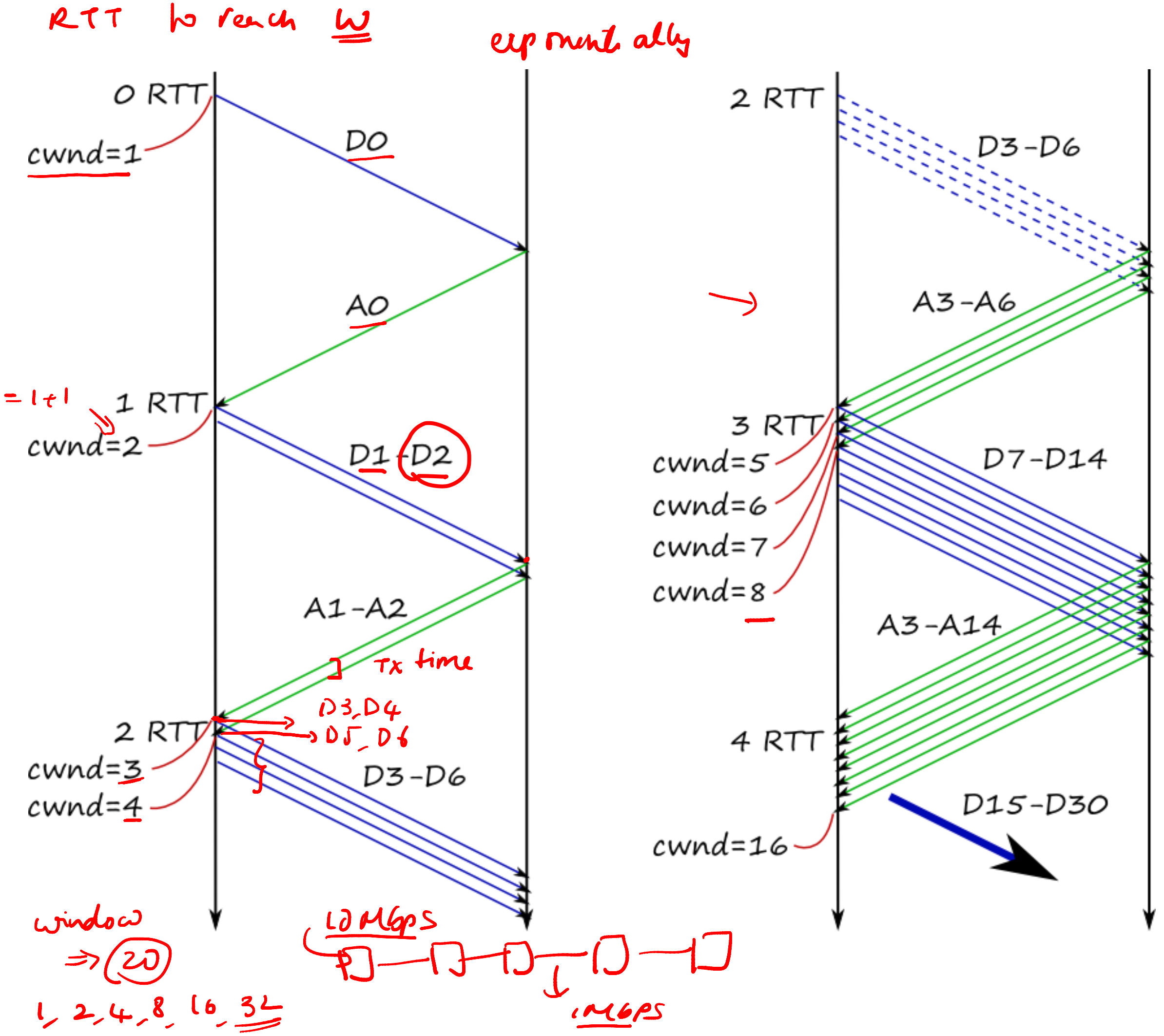
- What value of CW to choose initially?
  - Too large: pushes network into congestion
  - Just right: bursty transmissions can lead to losses



Slow Start

- Add a variable cwnd (congestion window)
  - Captures the number of outstanding data in the network
- At start, set cwnd=1
- On each ack for new data, increase cwnd by 1

- Takes $\log_2 W$ round trip times to get to W
- Sends data at most twice the bottleneck link on the path
- Bound to overestimate capacity at some time in future



Conservation at Equilibrium

- Don't put a packet unless a packet is removed
  - Particularly important when the network is congested
  - Can potentially happen on timeouts → proper RTT estimation crucial
  - Delayed packets should not be interpreted as lost

TO = 200ms

2 × 100

RTT

RTT
400ms

RTT Estimation: Original Algorithm

- Measure SampleRTT for sequence/ack combo
- EstimatedRTT = a*EstimatedRTT + (1-a)*SampleRTT
  - Small a heavily influenced by temporary fluctuations
  - Large a not quick to adapt to real changes
  - a is between 0.8-0.9

variability

- Timeout = 2 * EstimatedRTT

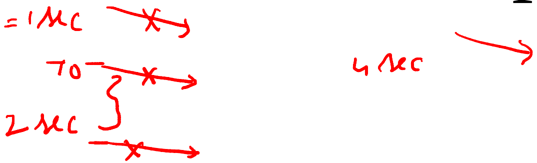
Jacobson/Karels Algorithm

- Algorithm takes into account variance of RTTs
  - If variance is small, EstimatedRTT can be trusted
  - If variance is large, timeout should not depend heavily on EstimatedRTT

- Difference = SampleRTT - EstimatedRTT
- EstimatedRTT = EstimatedRTT + (d * Difference)
- Deviation = Deviation + d (|Difference| - Deviation)), where d ~ 0.125
- Timeout = u * EstimatedRTT + q * Deviation, where u = 1 and q = 4

→ 0 → dominate

- Exponential Timeout backoff: controls spacing between retransmits



Exponential Damping

- From control theory: An unstable system can be stabilized by adding exponential damping

- “A network subject to random load shocks and prone to congestive collapse can be stabilized by adding exponential damping to its primary excitation (Traffic sources)”

Adapting to Path

- Estimating process can over or underestimate W; need to correct this
- Available bandwidth also changes over time; need to adapt to this
- Need a feedback mechanism from the network that the estimate is wrong

Overestimation

- Overestimation leads to congestion
- Feedback: If losses are due to congestion and timers are working correctly → Timeout indicates congestion

TO

- How to change the congestion window?

$\boxed{W}$ - C

$\boxed{TO}$ →

W/C

  - Additive decrease or multiplicative decrease?
  - Multiplicative decrease yields better stability
    - $W_i = dW_{i-1}$ (d<1, typically 0.5)

$W = 10$

$W = 5$

Underestimation

$w = 5 \longrightarrow 15$

- Underestimation leads to lower utilization

$w + c$

$c \times \underline{\underline{w}}$

- Additive increase or multiplicative increase?
  - Exponential increase leads to instability; overestimation is inevitable
  - Additive increase
    - $W_i = W_{i-1} + u$ ($u \ll W_{max}$; typical u is 1)
    - Increase window by 1 segment every RTT

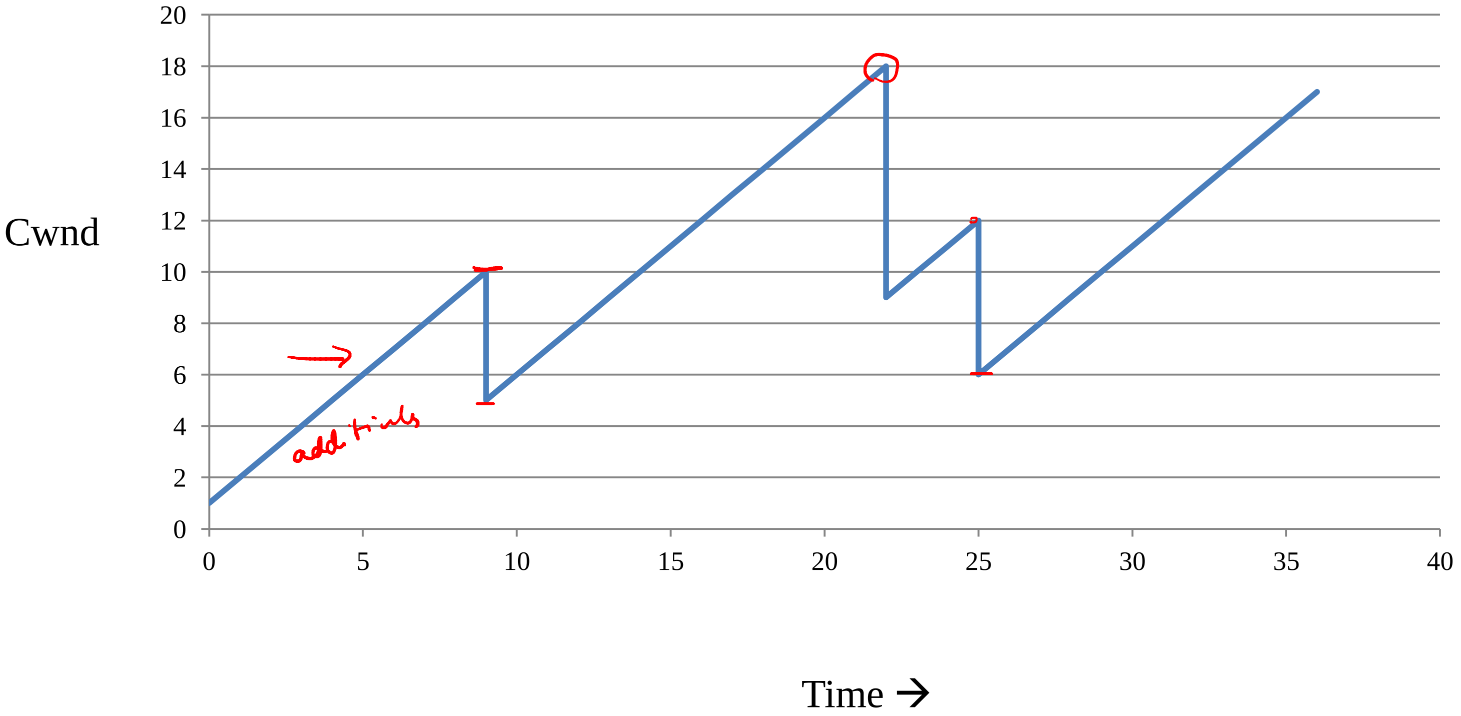
Congestion Avoidance

- Additive Increase, Multiplicative Decrease
- On detecting congestion, set cwnd to half the window size (multiplicative decrease)

self clocking

- On each ack of new data, increase cwnd by 1/cwnd (additive increase)

RTT → cwnd by 1 segment ✓


Cwnd
20
18
16
14
12
10
8
6
4
2
0
0
5
10
15
20
25
30
35
40
Time →
add tivls

Summary

- Congestion control is a difficult task
  - Prevent underutilization; ensure no congestion; ensure fairness
- TCP relies on a variety of techniques to achieve this
  - Slow start, RTT estimation, Congestion avoidance (AIMD)
- Ahead: Putting it all together in TCP versions