15-441 Computer Networking

Lecture 19 – TCP Performance

Outline

- TCP congestion avoidance
- TCP slow start
- TCP modeling

Additive Increase/Decrease

- Both $X_1$ and $X_2$ increase/decrease by the same amount over time
  - Additive increase improves fairness and additive decrease reduces fairness

Multiplicative Increase/Decrease

- Both $X_1$ and $X_2$ increase by the same factor over time
  - Extension from origin – constant fairness
**What is the Right Choice?**

- Constraints limit us to AIMD
  - Improves or keeps fairness constant at each step
  - AIMD moves towards optimal point

**TCP Congestion Control**

- Changes to TCP motivated by ARPANET congestion collapse
- Basic principles
  - AIMD
  - Packet conservation
  - Reaching steady state quickly
  - ACK clocking

**Implementation Issue**

- Operating system timers are very coarse – how to pace packets out smoothly?
- Implemented using a congestion window that limits how much data can be in the network.
  - TCP also keeps track of how much data is in transit
  - Data can only be sent when the amount of outstanding data is less than the congestion window.
  - The amount of outstanding data is increased on a “send” and decreased on “ack”
  - (last sent – last acked) < congestion window
  - Window limited by both congestion and buffering
  - Sender’s maximum window = Min (advertised window, cwnd)

**ACK Clocking**

- Congestion window helps to “pace” the transmission of data packets
- In steady state, a packet is sent when an ack is received
  - Data transmission remains smooth, once it is smooth
  - Self-clocking behavior
**AIMD**
- Distributed, fair and efficient
- Packet loss is seen as sign of congestion and results in a multiplicative rate decrease
  - Factor of 2
  - TCP periodically probes for available bandwidth by increasing its rate

**Congestion Avoidance**
- If loss occurs when $cwnd = W$
  - Network can handle $0.5W \sim W$ segments
  - Set $cwnd$ to $0.5W$ (multiplicative decrease)
- Upon receiving ACK
  - Increase $cwnd$ by $(1 \text{ packet})/cwnd$
    - What is 1 packet? → 1 MSS worth of bytes
    - After $cwnd$ packets have passed by → approximately increase of 1 MSS
- Implements AIMD

**Congestion Avoidance Sequence Plot**

**Congestion Avoidance Behavior**
Packet Conservation

- At equilibrium, inject packet into network only when one is removed
  - Sliding window and not rate controlled
  - But still need to avoid sending burst of packets → would overflow links
    - Need to carefully pace out packets
    - Helps provide stability
- Need to eliminate spurious retransmissions
  - Accurate RTO estimation
  - Better loss recovery techniques (e.g. fast retransmit)

Outline

- TCP congestion avoidance
  - TCP slow start
  - TCP modeling

Congestion Avoidance Behavior

Reaching Steady State

- Doing AIMD is fine in steady state but slow…
- How does TCP know what is a good initial rate to start with?
  - Should work both for a CDPD (10s of Kbps or less) and for supercomputer links (10 Gbps and growing)
  - Quick initial phase to help get up to speed (slow start)
Slow Start Packet Pacing

- How do we get this clocking behavior to start?
  - Initialize cwnd = 1
  - Upon receipt of every ack, cwnd = cwnd + 1
- Implications
  - Window actually increases to $W \times \log_2(W)$
  - Can overshoot window and cause packet loss

Slow Start Example

Return to Slow Start

- If packet is lost we lose our self clocking as well
  - Need to implement slow-start and congestion avoidance together
- When retransmission occurs set ssthresh to 0.5w
  - If cwnd < ssthresh, use slow start
  - Else use congestion avoidance
TCP Saw Tooth Behavior

Outline

- TCP congestion avoidance
- TCP slow start
- TCP modeling

TCP Performance

- Can TCP saturate a link?
- Congestion control
  - Increase utilization until... link becomes congested
  - React by decreasing window by 50%
  - Window is proportional to rate * RTT
- Doesn’t this mean that the network oscillates between 50 and 100% utilization?
  - Average utilization = 75%??
  - No...this is "not" right!

TCP Congestion Control

Rule for adjusting $W$
- If an ACK is received: $W \leftarrow W + \frac{1}{W}$
- If a packet is lost: $W \leftarrow W / 2$

Window size

$W_{\text{max}}$

Only $W$ packets may be outstanding
**Single TCP Flow**

*Router without buffers*

- $W = 1$
- $\text{util} = 0\%$

**Summary Unbuffered Link**

- The router can’t fully utilize the link
  - If the window is too small, link is not full
  - If the link is full, next window increase causes drop
  - With no buffer it still achieves 75% utilization

**TCP Performance**

- In the real world, router queues play important role
  - Window is proportional to rate * RTT
    - But, RTT changes as well the window
  - Window to fill links = propagation RTT * bottleneck bandwidth
    - If window is larger, packets sit in queue on bottleneck link

**TCP Performance**

- If we have a large router queue → can get 100% utilization
  - But, router queues can cause large delays
- How big does the queue need to be?
  - Windows vary from $W \to W/2$
    - Must make sure that link is always full
    - $W/2 > \text{RTT} \times \text{BW}$
    - $W = \text{RTT} \times \text{BW} + Q_{\text{size}}$
    - Therefore, $Q_{\text{size}} > \text{RTT} \times \text{BW}$
  - **Ensures 100% utilization**
  - Delay?
    - Varies between RTT and $2 \times \text{RTT}$
Single TCP Flow
Router with large enough buffers for full link utilization

Summary Buffered Link
- With sufficient buffering we achieve full link utilization
- The window is always above the critical threshold
- Buffer absorbs changes in window size
  - Buffer Size = Height of TCP Sawtooth
  - Minimum buffer size needed is 2T*C
- This is the origin of the rule-of-thumb

TCP (Summary)
- General loss recovery
  - Stop and wait
  - Selective repeat
- TCP sliding window flow control
- TCP state machine
- TCP loss recovery
  - Timeout-based
    - RTT estimation
  - Fast retransmit
  - Selective acknowledgements

TCP (Summary)
- Congestion collapse
  - Definition & causes
- Congestion control
  - Why AIMD?
  - Slow start & congestion avoidance modes
  - ACK clocking
  - Packet conservation
- TCP performance modeling
  - How does TCP fully utilize a link?
    - Role of router buffers