Chapter 3: Roadmap

3.1 Transport-layer services
3.2 Multiplexing and demultiplexing
3.3 Connectionless transport: UDP
3.4 Principles of reliable data transfer
3.5 Connection-oriented transport: TCP
   - Segment structure
   - Reliable data transfer
   - Flow control
   - Connection management
3.6 Principles of congestion control
3.7 TCP congestion control
TCP Congestion Control

- **End-to-end** control (no network assistance)
- Sender limits transmission: \( \text{LastByteSent} - \text{LastByteAcked} \leq \text{CongWin} \)
- **CongWin** is a function of perceived network congestion
- The **effective** window is the minimum of \( \text{CongWin} \), flow-control window carried in the ACKs, and sender’s own buffer space

- **How does sender perceive congestion?**
  - Loss event = timeout or 3 duplicate acks
- **TCP sender reduces rate** (\( \text{CongWin} \)) after loss event
- **Three mechanisms:**
  - Slow start
  - Conservative after timeouts
  - AIMD (congestion avoidance)
**TCP Slow Start**

- When connection begins, $\text{CongWin} = 1$ MSS
  - *Example*: MSS = 500 bytes and RTT = 200 msec
  - Q: initial rate?
  - A: 20 Kbits/s
- Available bandwidth may be much larger than MSS/RTT
  - Desirable to quickly ramp up to a “respectable” rate
- Solution: **Slow Start (SS)**
  - When a connection begins, it increases rate exponentially fast until first loss or receiver window is reached
  - Term “slow” is used to distinguish this algorithm from earlier TCPs which directly jumped to some huge rate
**TCP Slow Start (More)**

- Let $W$ be congestion window in pkts and $B = \text{CongWin}$ be the same in bytes ($B = \text{MSS} \times W$)

- Slow start
  - Double $\text{CongWin}$ every RTT

- Done by incrementing $\text{CongWin}$ for every ACK received:
  - $W = W + 1$ per ACK
    (or $B = B + \text{MSS}$)

- **Summary**: initial rate is slow but ramps up exponentially fast
**Congestion Avoidance**

- **TCP Tahoe loss** (timeout or triple dup ACK):
  - Threshold = CongWin/2
  - CongWin is set to 1 MSS
  - Slow start until threshold is reached; then move to linear probing
- **TCP Reno loss**:
  - Timeout: same as Tahoe
  - 3 dup ACKs: CongWin is cut in half, then continue linear probing (called fast recovery, now part of AIMD)

**Fast Recovery Philosophy**:

- Three dup ACKs indicate that network is capable of delivering subsequent segments
- Timeout before 3-dup ACK is more alarming
TCP Reno AIMD (Additive Increase, Multiplicative Decrease)

Additive increase: increase $\text{CongWin}$ by 1 MSS every RTT in the absence of loss events: *probing*

Multiplicative decrease: cut $\text{CongWin}$ in half after fast retransmit (3-dup ACKs)

Peaks are different: # of flows or RTT changes

congestion window

- 24 Kbytes
- 16 Kbytes
- 8 Kbytes

3-dup ACK (loss)

Time
TCP Reno Equations

• To better understand TCP, we next examine its AIMD equations (congestion avoidance)

• General form (loss detected through 3-dup ACK):

\[ W = \begin{cases} \frac{W}{2} \quad \text{per loss} \\ W + \frac{1}{W} \quad \text{per ACK} \end{cases} \]

• Reasoning
  - For each window of size \( W \), we get exactly \( W \) acknowledgments in one RTT (assuming no loss!)
  - This increases window size by roughly 1 packet per RTT

• Performing actions on packet arrival is lower overhead than waking up on timers
TCP Reno Equations

\[ W = \begin{cases} \frac{W}{2} & \text{per ACK per loss} \\ \frac{1}{W} & \text{per ACK} \end{cases} \]

- What is the equation in terms of \( B = \text{MSS} \times W \)?

\[
B = \begin{cases} 
B + \frac{\text{MSS}^2}{B} & \text{per ACK} \\
B/2 & \text{per loss}
\end{cases}
\]

- Equivalently, TCP increases \( B \) by \( \text{MSS} \) per RTT.

- What is the rate of TCP given that its window size is \( B \) (or \( W \))?

- Since TCP sends a full window of pkts per RTT, its ideal rate can be written as:

\[
r = \frac{B}{\text{RTT} + \frac{L}{R}} \approx \frac{B}{\text{RTT}} = \frac{\text{MSS} \times W}{\text{RTT}}
\]
# TCP Reno Sender Congestion Control

<table>
<thead>
<tr>
<th>Event</th>
<th>State</th>
<th>TCP Sender Action</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK receipt for previously unacked data</td>
<td>Slow Start (SS)</td>
<td>CongWin += MSS, If (CongWin &gt;= ssthresh) {</td>
<td>Results in a doubling of CongWin every RTT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set state to “Congestion Avoidance”</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACK receipt for previously unacked data</td>
<td>Congestion Avoidance (CA)</td>
<td>CongWin += MSS² / CongWin</td>
<td>Additive increase, resulting in increase of CongWin by 1 MSS every RTT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loss event detected by triple duplicate ACK</td>
<td>SS or CA</td>
<td>ssthresh = max(CongWin/2, MSS)</td>
<td>Fast recovery, implementing multiplicativde decrease</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CongWin = ssthresh</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set state to “Congestion Avoidance”</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeout</td>
<td>SS or CA</td>
<td>ssthresh = max(CongWin/2, MSS)</td>
<td>Enter slow start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CongWin = MSS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set state to “Slow Start”</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duplicate ACK</td>
<td>SS or CA</td>
<td>Increment duplicate ACK count for segment being acked</td>
<td>CongWin and Threshold not changed</td>
</tr>
</tbody>
</table>
TCP Reno Congestion Control

• Summary:

- **Timeout** $W = 1$
- **New ACK** $W = W + 1$
- **Timeout** $W = 1$
- **reach threshold or triple dup ACK**

- **New ACK** $W = W + 1/W$
- **Triple dup ACK** $W = W/2$
TCP Throughput

• What’s the average throughout of TCP as a function of max window size $W$ and $RTT$?
  – Ignore slow start and assume perfect AIMD (no timeouts)
• Let $W$ be the window size when loss occurs
  – At that time, throughput is $W \times \frac{MSS}{RTT}$
  – Just after loss, window drops to $W/2$, throughput is halved
• Average rate:

$$r_{av} = \frac{3}{4} \times \frac{W \times MSS}{RTT} = \frac{W_{av} \times MSS}{RTT}$$
TCP Model

- **Example**: 1500-byte segments, 100 ms RTT, want 10 Gbps average throughput $r_{av}$
  - Requires max window size $W = 111,111$ in-flight segments, 166 MB of buffer space ($W_{av} = 83,333$ packets)
  - But there are bigger issues as discussed below
- **Next**: derive average throughput in terms of loss rate
  - Assume packet loss probability is $p$
  - Roughly one packet lost for every $1/p$ sent packets
- **Step 1**: derive the number of packets transmitted in one oscillation cycle
TCP Model

- Examine time in terms of RTT units
  - At each step, window increases by 1 packet
- The number of packets sent between two losses:
  \[
  \text{sent} = \frac{W}{2} + \left( \frac{W}{2} + 1 \right) + \left( \frac{W}{2} + 2 \right) + \ldots + W
  \]
- Combining \( W/2 \) terms, we have:
  \[
  \text{sent} = \frac{W}{2} \left( \frac{W}{2} + 1 \right) + \sum_{i=1}^{W/2} i
  \]
TCP Model

• Thus we arrive at:

\[ sent = \frac{3}{8} W^2 + \frac{3}{4} W \]

• Step 2: now notice that this number equals \( \frac{1}{p} \)
  - Ignoring the linear term, we approximately get:

\[ \frac{1}{p} \approx \frac{3}{8} W^2 \]

• In other words:

\[ W = \sqrt{\frac{8}{3p}} \]
TCP Model

• **Step 3**: writing in terms of average rate:

\[ r_{av} = \frac{W_{av} \times MSS}{RTT} = \frac{3}{4}W \times MSS \quad \frac{3}{4} \sqrt{\frac{8}{3p}} \times MSS \]

• Simplifying:

\[ r_{av} = \frac{\sqrt{3/2} \times MSS}{RTT \sqrt{p}} \approx \frac{1.22 \times MSS}{RTT \sqrt{p}} \]

• This is the famous formula of AIMD throughput
  
  - **Note**: homework #3 does not use congestion control and its rate is a different function of \( p \)