Overview: TCP Basics

- Recap: using SEQ and ACK numbers
  - SEQ random initial number
  - "Acknowledge" next byte expected
  - Read chapter to review

- TCP congestion control
  - Read through multiple times!

- TCP flow control

TCP: SEQ and ACK numbers

<table>
<thead>
<tr>
<th>Host A</th>
<th>Host B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq=92, 8 bytes data</td>
<td>Seq=100, 20 bytes data</td>
</tr>
<tr>
<td>ACK=100</td>
<td>ACK=100</td>
</tr>
<tr>
<td>time</td>
<td>time</td>
</tr>
</tbody>
</table>

TCP: Cumulative ACK

<table>
<thead>
<tr>
<th>Host A</th>
<th>Host B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq=92, 8 bytes data</td>
<td>Seq=100, 20 bytes data</td>
</tr>
<tr>
<td>ACK=100</td>
<td>ACK=120</td>
</tr>
<tr>
<td>timeout</td>
<td>timeout</td>
</tr>
</tbody>
</table>

Cumulative ACK scenario
TCP: retransmission from timeout

TCP possible sender events:

1) Data received from application:
   1. Create a segment and assign a SEQ number
      - SEQ # is byte-stream number of first data byte in segment
   2. Start timer if it is not already running
      - Timer is for the oldest un-acked segment
      - Expiration interval: TimeOutInterval

2) Timeout (ACK not received):
   1. Retransmit segment that caused the timeout
   2. Restart the timer

3) ACK received for previously unacked segments
   1. Update what is known to be acked
   2. Start timer if there are outstanding segments

TCP: retransmission scenarios

TCP: Fast Retransmit upon 3 Dup ACK

1) What is/was ‘A’ s next step?
2) What does ‘B’ then do?
New Today:
Principles of Congestion Control

- Packet loss is caused by overflowing router buffers
- Retransmission treats the symptom
- Congestion control treats the cause

TCP Congestion Control

- Three questions
  1. How does a sender sense congestion?
  2. How does a sender limit its sending rate?
  3. What algorithm is used to change the send-rate?

TCP Congestion Control: details

How does sender perceive congestion?
- A loss event is
  - A timeout or
  - 3 duplicate ACKs

How does sender limit its send rate?
- TCP sender reduces the send rate via changing a variable value, the "CongWin," after a loss event

TCP Congestion Control Window

- CongWin is dynamic, function of perceived network congestion
- Sender limits transmission:
\[ \text{LastByteSent} - \text{LastByteAcked} \leq \text{CongWin} \]
- Changing CongWin changes the SendRate
TCP Congestion Control Algorithm

Three major mechanisms:
1) Slow start
2) Congestion Avoidance
   • AIMD = additive increase, multiplicative decrease
3) Fast Recovery:
   • Reaction to timeout events versus 3 duplicate ACKs

TCP Slow Start

1) When connection begins, CongWin = 1 MSS
   • Available bandwidth probably much greater
   • Desirable to quickly ramp up to respectable rate
2) Increase rate exponentially fast until first loss event
   • Grow window 1MSS for each ACK received

Summary: initial rate is slow but ramps up exponentially fast

TCP: detecting, reacting to loss

- Loss indicated by timeout
  • cwnd set to 1 MSS;
  • Window grows exponentially (as in slow start) to threshold, then grows linearly
- Loss indicated by 3 duplicate ACKs
  • Duplicate ACKs indicate network capable of delivering some segments
  • cwnd is cut in half window then grows linearly
TCP Slow Start with Exp. Increase

- When connection begins, increase rate exponentially until first loss event:
  - Double CongWin every RTT
  - Increment CongWin for every ACK received

TCP: Congestion algorithm switching from slow start to CA

Implementation:
- variable ssthresh
- on loss event, ssthresh is set to 1/2 of cwnd just before loss event

Reaction to Loss Events

- Exponential increase switches to linear increase when CongWin gets to the ‘threshold’ value (size)

 Identify everything on this graph
Summary: TCP Congestion Control

**Increase Sending Rate Phase Options:**
1. When CongWin is below Threshold, sender in slow-start phase, window grows exponentially.
2. When CongWin is above Threshold, sender is in congestion-avoidance phase, window grows linearly.

**Decrease Sending Rate Phase Options:**
1. When a triple duplicate ACK occurs, Threshold set to CongWin/2 and CongWin set to Threshold.
2. When timeout occurs, Threshold set to CongWin/2 and CongWin is set to 1 MSS.