Announcements:

- Assignment 2 and Lab 2 are posted
Chapter 3 outline

- Principles of reliable data transfer
- Connection-oriented transport: TCP
- Principles of congestion control
  - TCP congestion control
TCP Round Trip Time (RTT) and Timeout

Why need to estimate RTT?
- “timeout” and “retransmit” needed to address packet loss
- need to know when to timeout and retransmit

Ideal world:
- exact RTT is needed

Real world:
- RTTs change over time:
  - packets may take different paths
  - network load changes over time
- RTTs can only be estimated

Some intuition
- What happens if too short?
  - Premature timeout
  - Unnecessary retransmissions
- What happens if too long?
  - Slow reaction to segment loss
TCP Round Trip Time (RTT) and Timeout

Technique: Exponential Weighted Moving Average (EWMA)

\[ \text{EstimatedRTT}(\text{current}) = (1-\alpha) \times \text{EstimatedRTT}(\text{previous}) + \alpha \times \text{SampleRTT}(\text{recent}) \]

\[ 0 < \alpha < 1; \text{ typical value: } \alpha = 0.125 \]

- **SampleRTT:**
  - measured time from segment transmission until ACK receipt
  - current value of RTT
  - Ignore retransmission

- **EstimatedRTT:**
  - estimated based on past & present; smoother than SampleRTT
  - to be used to set timeout period
TCP Round Trip Time (RTT) and Timeout

Technique: Exponential Weighted Moving Average (EWMA)

\[
\text{EstimatedRTT}^{(\text{current})} = (1-\alpha) \times \text{EstimatedRTT}^{(\text{previous})} + \alpha \times \text{SampleRTT}^{(\text{recent})}
\]

\[0 < \alpha < 1; \text{ typical value: } \alpha = 0.125\]

Illustration:
Assume: we received \( n \) RTT samples so far:
Let’s order them as: \( n \) is the most recent one,
\( n-1 \) is the 2\( ^{nd} \) most recent, etc.

\[
\text{EstimatedRTT}^{(n)} = (1-\alpha) \times \text{EstimatedRTT}^{(n-1)} + \alpha \times \text{SampleRTT}^{(n)}
\]

(\( \text{EstimatedRTT}^{(n)} \): estimated RTT after receiving ACK of \( n \)th Packet.)

Example: Suppose 3 ACKs returned with \( \text{SampleRTT}^{(1)}, \text{SampleRTT}^{(2)}, \text{SampleRTT}^{(3)} \)

Question: What would be \( \text{EstimatedRTT} \) after receiving the 3 ACKs?
Assume: \( \text{EstimatedRTT}^{(0)} = 2 \) seconds (initial/default value)
TCP Round Trip Time (RTT) and Timeout

Technique: Exponential Weighted Moving Average (EWMA)

\[
\text{EstimatedRTT}(\text{current}) = (1-\alpha) \times \text{EstimatedRTT}(\text{previous}) + \alpha \times \text{SampleRTT}(\text{recent})
\]

\[0 < \alpha < 1; \text{ typical value: } \alpha = 0.125\]

What happens if \(\alpha\) is too small (say very close 0):

- A sudden, real change in network load does not get reflected in \(\text{EstimatedRTT}\) fast enough
- May lead to under- or overestimation of RTT for a long time

What happens if \(\alpha\) is too large (say very close 1):

- Transient fluctuations/changes in network load affects \(\text{EstimatedRTT}\) and makes it unstable when it should not
- Also leads to under- or overestimation of RTT
Example RTT estimation:

RTT: gaia.cs.umass.edu to fantasia.eurecom.fr

SampleRTT
Estimated RTT
TCP Round Trip Time (RTT) and Timeout

Setting the timeout

- timeout = EstimatedRTT, any problem with this???
- add a “safety margin” to EstimatedRTT
  - large variation in EstimatedRTT → larger safety margin
- see how much SampleRTT deviates from EstimatedRTT:

\[
\text{DevRTT} = (1-\beta)\times\text{DevRTT} + \beta\times|\text{SampleRTT}-\text{EstimatedRTT}|
\]
(typically, \( \beta = 0.25 \))

Then set timeout interval:

\[
\text{TimeoutInterval} = \text{EstimatedRTT} + 4\times\text{DevRTT}
\]
TCP: Overview

- **point-to-point:**
  - one sender, one receiver
- **reliable, in-order byte stream:**
  - no “message boundaries”
- **pipelined:**
  - TCP congestion and flow control set window size
- **send & receive buffers**

- **full duplex data:**
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- **connection-oriented:**
  - handshaking (exchange of control msgs) init's sender, receiver state before data exchange
- **flow controlled:**
  - sender will not overwhelm receiver

RFCs: 793, 1122, 1323, 2018, 2581
TCP: a reliable data transfer

- TCP creates rdt service on top of IP’s unreliable service
- Pipelined segments
- Cumulative acks
- TCP uses single retransmission timer

- Retransmissions are triggered by:
  - timeout events
  - duplicate acks

- Initially consider simplified TCP sender:
  - ignore duplicate acks
  - ignore flow control, congestion control
TCP sender events:

**Data received from app:**
- Create segment with Seq #
- Seq # is byte-stream number of first data byte in segment
- start timer if not already running (think of timer as for “oldest unACK’ed segment”)
- expiration interval: TimeOutInterval

**Timeout:**
- retransmit segment that caused timeout
- restart timer

**Ack received:**
- If it acknowledges previously unACK’ed segments
  - update what is known to be ACK’ed
  - start timer if there are outstanding segments
TCP Seq. #'s and ACKs

Seq. #'s:
- byte stream "number" of first byte in segment's data

ACKs:
- Seq # of next byte expected from other side
- cumulative ACK

Host A
- User types 'C'
- Seq=42, ACK=79, data = 'C'
- Host ACKs receipt of 'C', echoes back 'C'

Host B
- Seq=79, ACK=43, data = 'C'
- Seq=43, ACK=80
- Host ACKs receipt of echoed 'C'

simple telnet scenario
TCP: retransmission scenarios

- Host A
  - Seq=92, 8 bytes data
  - ACK=100
  - X
  - loss

- Host B
  - Seq=92, 8 bytes data

- Host A
  - Seq=92 timeout

- Host B
  - Seq=92 timeout

- Premature timeout

Loss ACK scenario
TCP retransmission scenarios (more)

Cumulative ACK scenario

Host A

Seq=92, 8 bytes data

Seq=100, 20 bytes data

loss

Host B

ACK=100

ACK=120

time
timeout

Chapter 3, slide: 67
<table>
<thead>
<tr>
<th>Event at Receiver</th>
<th>TCP Receiver action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed</td>
<td>Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK</td>
</tr>
<tr>
<td>Arrival of in-order segment with expected seq #. One other segment has ACK pending</td>
<td>Immediately send single cumulative ACK, ACKing both in-order segments</td>
</tr>
<tr>
<td>Arrival of out-of-order segment higher-than-expect seq. #. Gap detected</td>
<td>Immediately send <em>duplicate ACK</em>, indicating seq. # of next expected byte</td>
</tr>
<tr>
<td>Arrival of segment that partially or completely fills gap</td>
<td>Immediate send ACK, provided that segment starts at lower end of gap</td>
</tr>
</tbody>
</table>
Fast Retransmit

- Suppose: Packet 0 gets lost
  - Q: When will retransmission of Packet 0 happen? Why does it happen at that time?
  - A: typically at $t_1$; we think it is lost when timer expires

- Can we do better??
  Think of what means to receive many duplicate ACKs:
  - it means Packet 0 is lost
  - Why wait until timeout when we know packet 0 is lost?
    => Fast retransmit
    => better performance

- Why several dupl. ACKs, not 1 or 2?
  - Think of what happens when pkt0 arrives after pkt1 (delayed, not lost)
  - Think of what happens when pkt0 arrives after pkt1 & 2, etc.
Fast Retransmit: recap

- Receipt of duplicate ACKs indicate loss of segments
  - Sender often sends many segments back-to-back
  - If segment is lost, there will likely be many duplicate ACKs.

This is how TCP works:
- If sender receives 3 ACKs for the same data, it supposes that segment after ACK’ed data was lost:
  - **Fast Retransmit:**
    - resend segment before timer expires
    - better performance
Review questions

Problem:
- TCP connection between A and B
- B received up to 248 bytes
- A sends back-to-back 2 segments to B with 40 and 60 bytes
- B ACKs every packet it receives

Q1: What is Seq# in 1st and 2nd segments from A to B?

Q2: Suppose: 1st segment gets to B first. What is ACK # in 1st ACK?
**Problem:**
- TCP connection between A and B
- B received up to 248 bytes
- A sends back-to-back 2 segments to B with 40 and 60 bytes
- B ACKs every packet it receives

**Q1:** What is Seq# in 1st and 2nd segments from A to B?

**Q2:** Suppose: 1st segment gets to B first. What is ACK # in 1st ACK?

**Q3:** Suppose: 2nd segment gets to B first. What is ACK # in 1st ACK? What is ACK # in 2nd ACK?
Review questions

Problem:
- TCP connection between A and B
- B received up to 248 bytes
- A sends back-to-back 2 segments to B with 40 and 60 bytes
- B ACKs every packet it receives

Q1: What is Seq# in 1st and 2nd segments from A to B?

Q2: Suppose: 1st segment gets to B first. What is ACK # in 1st ACK?

Q3: Suppose: 2nd segment gets to B first. What is ACK # in 1st ACK? What is ACK # in 2nd ACK?
Review questions

Problem:
- TCP connection between A and B
- B received up to 248 bytes
- A sends back-to-back 2 segments to B with 12 and 20 bytes, respectively
- B ACKs every packet it receives

Now suppose:
- 2 segments get to B in order.
- 1\textsuperscript{st} ACK is lost
- 2\textsuperscript{nd} ACK arrives after timeout

Question: Fill out all packet Seq #'s, and ACKs' ACK #'s in the timing diagram.