Transport Layer

Derek Leonard
Hendrix College

October 13, 2008
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 (cont)
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: Overview [RFCs: 793, 1122, 1323, 2018, 2581]

- Point-to-point:
  - One sender, one receiver
- Reliable, in-order *byte steam*:
  - Message boundaries are not visible to the application
- 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) initializes sender/receiver state before data exchange
- Flow controlled:
  - Sender will not overwhelm receiver
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 Segment Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source port #</td>
<td>Source port number</td>
</tr>
<tr>
<td>dest port #</td>
<td>Destination port number</td>
</tr>
<tr>
<td>sequence #</td>
<td>Sequence number</td>
</tr>
<tr>
<td>acknowledgement number</td>
<td>Acknowledgement number</td>
</tr>
<tr>
<td>head len</td>
<td>Header length</td>
</tr>
<tr>
<td>URG</td>
<td>URG flag</td>
</tr>
<tr>
<td>ACK</td>
<td>ACK flag</td>
</tr>
<tr>
<td>PSH</td>
<td>PSH flag</td>
</tr>
<tr>
<td>RST, SYN, FIN</td>
<td>RST, SYN, FIN flags</td>
</tr>
<tr>
<td>Urg data pnter</td>
<td>Urgent data pointer</td>
</tr>
<tr>
<td>Receive window</td>
<td>Receive window size</td>
</tr>
<tr>
<td>checksum</td>
<td>Internet checksum</td>
</tr>
<tr>
<td>application data</td>
<td>Application data (variable length)</td>
</tr>
</tbody>
</table>

URG: urgent data (generally not used)
ACK: ACK # valid
PSH: push data now (generally not used)
RST, SYN, FIN: connection estab (setup, teardown commands)
Internet checksum (as in UDP)
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

Q: how receiver handles out-of-order segments?
A: TCP spec doesn’t say, up to implementor

Simple telnet scenario
TCP Seq. #'s and ACKs

FTP Example:

- Suppose MSS = 1,000 bytes and the sender has a large file to transmit (we ignore seq field in ACKs since no data is traveling back)

What is the sender’s window size?

- Host A
  - seq = 0
  - seq = 1000
  - seq = 2000
  - seq = 3000
  - seq = 4000

- Host B
  - ACK = 1000
  - ACK = 2000
  - ACK = 2000
  - ACK = 2000
  - ACK = 5000

timeout
TCP Round Trip Time and Timeout

**Q:** how to set TCP timeout value?
- “Larger” than RTT
  - But RTT varies
- Too short:
  - premature timeout
  - Unnecessary retransmissions
- Too long: slow reaction to segment loss

**Q:** how to estimate RTT?
- **SampleRTT:** measured time from segment transmission until ACK receipt
  - Ignore retransmissions
- **SampleRTT** will vary, want estimated RTT “smoother”
  - Average several recent measurements, not just current **SampleRTT**
TCP Round Trip Time and Timeout

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

- **Exponentially weighted moving average (EWMA)**
  - Influence of past sample decreases exponentially fast
  - Typical value: \( \alpha = 0.125 = 1/8 \)

- **Task: derive a non-recursive formula for** \( \text{EstimatedRTT}(n) \)
  - Assume \( \text{EstimatedRTT}(0) = \text{SampleRTT}(0) \)
  - Let \( Y(n) = \text{EstimatedRTT}(n) \) and \( y(n) = \text{SampleRTT}(n) \)

\[
Y(n) = (1 - \alpha)^n y(0) + \alpha \sum_{i=0}^{n-1} (1 - \alpha)^i y(n - i)
\]
Example RTT Estimation:
Setting the timeout:

EstimatedRTT plus a “safety margin”

- Larger variation in EstimatedRTT → larger safety margin

First estimate how much SampleRTT deviates from EstimatedRTT (typically, $\beta = 0.25$):

$$\text{DevRTT}(n) = (1-\beta)\times \text{DevRTT}(n-1) + \beta \times |\text{SampleRTT}(n) - \text{EstimatedRTT}(n)|$$

Then set timeout value (RTO):

$$\text{Timeout}(n) = \text{EstimatedRTT}(n) + 4\times \text{DevRTT}(n)$$
Example Timeout Estimation: