TCP Connection Management
Kurose & Ross, Chapter 3 (5th ed.)

Many slides adapted from:
J. Kurose & K. Ross \nComputer Networking: A Top Down Approach (5th ed.)
Addison-Wesley, April 2009.
Copyright 1996-2010, J.F Kurose and K.W. Ross, All Rights Reserved.
TCP: Overview

- point-to-point:
  - one sender, one receiver
- reliable, in-order byte steam:
  - 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) inits sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver

RFCs: 793, 1122, 1323, 2018, 2581
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 number</td>
<td>Sequence number for the data segment</td>
</tr>
<tr>
<td>acknowledgement number</td>
<td>Acknowledgement number for the data segment</td>
</tr>
<tr>
<td>head len</td>
<td>Length of the header</td>
</tr>
<tr>
<td>not used</td>
<td>Not used</td>
</tr>
<tr>
<td>Urg</td>
<td>Urgent data flag</td>
</tr>
<tr>
<td>valid</td>
<td>Validity of ACK flag</td>
</tr>
<tr>
<td>PSH</td>
<td>Push data flag</td>
</tr>
<tr>
<td>count</td>
<td>Counting by bytes of data</td>
</tr>
<tr>
<td>checksum</td>
<td>Checksum</td>
</tr>
<tr>
<td>Urg data pnter</td>
<td>Urgent data pointer</td>
</tr>
<tr>
<td>Receive window</td>
<td>Receiver window</td>
</tr>
<tr>
<td>Options</td>
<td>Variable length options</td>
</tr>
<tr>
<td>application data (variable length)</td>
<td>Application data (variable length)</td>
</tr>
<tr>
<td>Internet checksum (as in UDP)</td>
<td>Internet checksum (as in UDP)</td>
</tr>
</tbody>
</table>

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

- # bytes rcvr willing to accept
- counting by bytes of data (not segments!)
TCP Connection Management

Recall: TCP sender, receiver establish “connection” before exchanging data segments

- initialize TCP variables:
  - seq. #s
  - buffers, flow control info (e.g. RcvWindow)

- client: connection initiator
  Socket clientSocket = new Socket("hostname","port number");

- server: contacted by client
  Socket connectionSocket = welcomeSocket.accept();

Three way handshake:

Step 1: client host sends TCP SYN segment to server
  - specifies initial seq #
  - no data

Step 2: server host receives SYN, replies with SYNACK segment
  - server allocates buffers
  - specifies server initial seq. #

Step 3: client receives SYNACK, replies with ACK segment, which may contain data
Three-way handshake

Host 1

SYN = 1 (SEQ = x)

SYN = 1 ACK = 1
(SEQ = y, ACK = x+1)

ACK = 1
(SEQ = x+1, ACK = y+1)

Host 2
TCP Connection Management (cont.)

**Closing a connection:**

client closes socket:
   ```java
clientSocket.close();
```

**Step 1:** client end system sends TCP FIN control segment to server

**Step 2:** server receives FIN, replies with ACK. Closes connection, sends FIN.
**TCP Connection Management (cont.)**

**Step 3:** client receives FIN, replies with ACK.
- Enters “timed wait” - will respond with ACK to received FINs

**Step 4:** server, receives ACK. Connection closed.

**Note:** with small modification, can handle simultaneous FINs.
TCP’s statechart

- On board
  - Statechart appears in RFC 793
- Discussion of:
  - TIME_WAIT state
  - Syn flood attacks