Announcements

- **Reading**
  - Chapter 6 (6.4 & 6.6)

- **Midterm #1**
  - Re-grades were returned
    - Changes ranged from –7 to +10
Multiplexing in the Transport Layer

- **Upward multiplexing**
  - putting multiple transport connections onto one network connection
  - used to accommodate pricing strategies that charge for connections

- **Downward multiplexing**
  - using several network connections per transport connection
  - permits use of multiple copies of network resources
    - if the network layer uses sliding windows
      - a high latency network may under utilize the link
      - multiple connections each get a window
    - per connection buffer allocation
      - get more buffers
    - round-robin scheduling
      - get a larger share of link bandwidth
Crash Recovery

- **Router or Link Crashes**
  - Data in transit can be lost.
  - End nodes have sufficient state to recover lost data.
  - Transport protocol can hide network failures from the application.

- **Host Crashes**
  - Transport level state will be lost at one end.
  - Does the transport layer have sufficient info to recover? **No!**
    - Information must flow down to network and up to transport user
      - ACKs go down, and data goes up.
      - It is not possible to make these two operations atomic.
    - lack of stable storage causes this problem
  - Result, higher up layer must deal with host crashes
Protocol State Machines

Idle

<Connect, ¬P1>, A3
<Clear_req, *>, A4
<DISCON, P4>, A5
<timeout, *>, A7
<Call_acc, *>, A1
<SEND, ¬P5>, A8
<Clear_req, *>, A10
<Call_req, ¬P3>, A4
<Call_req, P3>, A1
<LISTEN, P2>, A1
<LISTEN, ¬P2>, A2

Waiting

<Send, ¬P5>, A7

Established

<Send, P5>, A7
<Clear_req, *>, A10
<Credit, *>, A11
<Timeout, *>, A3

Sending

<DISCON, ¬P4>, A6

Receiving

Disconnecting

<Recv, *>, A9
<Clear_req, *>, A10
<data, *>, A12
<clear_req, *>, A6
<clear_Conf, *>
<clear Req, *>

CMSC 417 – F01 (lec15)
## Predicates And State Transitions

<table>
<thead>
<tr>
<th>Pred</th>
<th>Meaning</th>
<th>Act</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Connection table full</td>
<td>A1</td>
<td>Send Call_acc</td>
</tr>
<tr>
<td>P2</td>
<td>Call_req pending</td>
<td>A2</td>
<td>Wait for Call_req</td>
</tr>
<tr>
<td>P3</td>
<td>LISTEN Pending</td>
<td>A3</td>
<td>Send Call_req</td>
</tr>
<tr>
<td>P4</td>
<td>Clear_req Pending</td>
<td>A4</td>
<td>Start Timer</td>
</tr>
<tr>
<td>P5</td>
<td>Credit Available</td>
<td>A5</td>
<td>Send Clear_conf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A6</td>
<td>Send Clear_req</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A7</td>
<td>Send message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A8</td>
<td>Wait for credit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A9</td>
<td>Send Credit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A10</td>
<td>Set Clr_req_recv flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A11</td>
<td>Record credit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A12</td>
<td>Accept message</td>
</tr>
</tbody>
</table>
TCP Timer Management

- **Problem:** How to pick timeout value?
  - need to estimate round-trip latency
  - need low variance in round trip latency
- **Solution:** dynamic estimates of RTT
  - \( RTT = \alpha \cdot RTT + (1 - \alpha) \cdot M \)
    - \( M \) time of an ACK
    - \( \alpha = 7/8 \)
  - Need to pick retransmission time
    - old policy, use \( \text{Timeout} = RTT \beta \), with \( \beta = 2 \)
    - estimate standard deviation of RTT using mean deviation
      \[ D = \alpha D + (1 - \alpha) \cdot |RTT - M| \]
      - \( \text{Timeout} = RTT + 4 \cdot D \)
  - How to update RTT on retransmission's
    - double Timeout on a retransmission
Other TCP Timers

- **Persistence Timer**
  - Prevents deadlock due to dropped window packets
    - This is a problem if the window is set to 0

- **Keepalive Timer**
  - Prevents half dead connections
  - may consume bandwidth
  - may kill live connections when net hiccups

- **TIMED Wait**
  - prevents re-use of a connection before max packet life is over
  - set to twice max packet lifetime