Announcements

- **Reading**
  - Today: Chapter 6 (6.3 and 6.4)

- **Midterms: Requests for re-grade due by 10/19**

- **HW #1 (Due Today)**

- **Class Project Notes:**
  - My comment about the parameter to config_get_info was **wrong**. It should take an array of 16 chars (i.e. a v6 address). See class web page for info about parseV6Addr which converts a string to a v6 address
  - the bit field notation int x:y; is **very** useful for packet headers
Protocol State Machines

<LISTEN, P1>  
<CONNECT, P1>  
<LISTEN, ~P2>, A2

<Connect, ~P1>, A3  
<Clear_req, *>  
<DISCON, P4>, A5  
<Credit, *>, A7

<SEND, ~P5>, A8  
<Call_acc, *>  
<Clear_req, *>, A10  
<data, *>, A12

<DISCON, ~P4>, A6

Sending

<DISCON, P4>, A5

Queued

<SEND, P5>, A7  
<Clear_req, *>, A10  
<Credit, *>, A11

<LISTEN, *>  
<Recv, *>, A9

Receiving

<Clear_req, *>, A10

<Clear_req, *>, A4

<call_req, ~P3>, A1  
<call_req, P3>, A1

<LISTEN, P2>, A1

<timeout, *>  
<clear_Req, *>  
<clear_Conf, *>

Disconnecting

<Clear_req, *>  
<DISCON, ~P4>, A6

Established

<call_req, P3>, A1

<LISTEN, P1>

<Connect, ~P1>, A3

Waiting

Idle

<timeout, *>

<Clear_req, *>  
<LISTEN, ~P2>, A2

<timeout, *>  
<clear_Req, *>  
<clear_Conf, *>

Disconnecting
## 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 Protocol

- **TSAPs**
  - Use <host, port> combination
  - Well known ports provide services
    - first 256 ports
    - SMTP 25, Telnet 23, Ftp 21, HTTP 80

- **Provides a byte stream**
  - this is **not** a message stream
  - a message (single call to send) may be split, merged, etc.

- **Urgent Data field**
  - provides cut through delivery *within* a transport connection
  - used to send breaks or other high priority info
TCP Packet Format

- Permits ACKs to be piggy packed
  - ACK is next byte expected
  - ACK is only valid if ACK bit is set
- Sequence number
  - first byte in packet
- Also used for connection establishment

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment Number</td>
<td>Window Size</td>
</tr>
<tr>
<td></td>
<td>Urgent Pointer</td>
</tr>
</tbody>
</table>

0 Or More Options

32 bits
TCP Connection Management

- **Three-way Handshake**
- **Initial Sequence Numbers**
  - Use a 4 micro-second clock
  - hosts must wait $T$ (120 seconds) before a reboot
- **Connection Closure**
  - Each side uses a FIN and FIN_ACK message
  - A FIN times out after $2T$ (240 seconds)
  - Keep alives used to timeout half dead connections
TCP Flow Control

- **Use Variable Sized Sliding Window**
  - ACK indicates start of window
  - Window size indicates current size of window

- **Receiver can send a window of 0**
  - Indicates that it wants to pause the connection
  - Urgent data need not follow this request

- **Window size of 16 bits is too small**
  - 64K Bytes
  - Only a small fraction of the in-flight bytes when
    - Bandwidth is high
    - Delay is high
  - Solution: window shift option:
    - Bit shift window up to 16 bits
    - Permits up to $2^{32}$ byte windows
    - Reduces window granularity
TCP Congestion Control

- **Detecting Congestion**
  - In general it is difficult
  - But, consider why a packet might be dropped
    - link error - but links are very reliable now
    - buffer overflow --> congestion
  - Use re-transmission timeouts as an estimate of congestion

- **Dealing with Congestion**
  - add a second window (congestion window)
    - limit transmissions to min(recv window, congestion window)
  - start with congestion window = max segment window
    - initial max segment is one kilo-byte
    - on a ACK without a timeout
      - if window < threshold, increment by one max segment
        otherwise increment by initial max segment
  - on timeout
    - cut threshold in half
    - set window size to initial max segment