CMPE 150 -- Introduction to Computer Networks

- Instructor: Patrick Mantey
  mantey@soe.ucsc.edu
  http://www.soe.ucsc.edu/~mantey/
- Office: Engr. 2 Room 595J
- Office hours: Tues 3-5 PM, Mon 5-6 PM*
- TA: Anselm Kia
  akia@soe.ucsc.edu
- Web site: http://www.soe.ucsc.edu/classes/cmpe150/Winter09/
- Text: Tannenbaum: Computer Networks
  (4th edition – available in bookstore, etc. )
<table>
<thead>
<tr>
<th>Lecture</th>
<th>Date</th>
<th>Topic</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6-Jan</td>
<td>Intro / Overview</td>
<td>Chapter 1</td>
</tr>
<tr>
<td>2</td>
<td>8-Jan</td>
<td>Internet Architecture</td>
<td>Chapter 1, 2</td>
</tr>
<tr>
<td>3</td>
<td>13-Jan</td>
<td>Physical Layer</td>
<td>Chapter 2</td>
</tr>
<tr>
<td>4</td>
<td>15-Jan</td>
<td>Physical Layer</td>
<td>Chapter 2</td>
</tr>
<tr>
<td>5</td>
<td>20-Jan</td>
<td>Link Layer</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>6</td>
<td>22-Jan</td>
<td>Link Layer / MAC</td>
<td>Chapter 3, 4</td>
</tr>
<tr>
<td>7</td>
<td>27-Jan</td>
<td>MAC Sublayer</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>8</td>
<td>29-Jan</td>
<td>Wireless</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>9</td>
<td>3-Feb</td>
<td>Bridges (and Review)</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>10</td>
<td>5-Feb</td>
<td>Midterm Exam 1</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>11</td>
<td>10-Feb</td>
<td>Intro to Network Layer</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>12</td>
<td>12-Feb</td>
<td>Routing and Routers</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>13</td>
<td>17-Feb</td>
<td>Congestion Control, QOS</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>14</td>
<td>19-Feb</td>
<td>Internet / Networking / IP</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>15</td>
<td>24-Feb</td>
<td>Transport Layer / Services / Protocols</td>
<td>Chapter 6</td>
</tr>
<tr>
<td>16</td>
<td>26-Feb</td>
<td>Internet Transport Protocols: UDP, TCP</td>
<td>Chapter 6</td>
</tr>
<tr>
<td>17</td>
<td>3-Mar</td>
<td>RPC/ TCP / Performance / Applications / Review</td>
<td>Chapter 6, 7</td>
</tr>
<tr>
<td>18</td>
<td>5-Mar</td>
<td>Midterm Exam 2</td>
<td>Chapter 7</td>
</tr>
<tr>
<td>19</td>
<td>10-Mar</td>
<td>Applications / Multimedia</td>
<td>Chapter 7</td>
</tr>
<tr>
<td>20</td>
<td>12-Mar</td>
<td>Buffer / Review</td>
<td>Chapter 7</td>
</tr>
<tr>
<td><strong>Final Exam</strong></td>
<td>18-Mar</td>
<td>12-3 PM</td>
<td></td>
</tr>
</tbody>
</table>
Text Readings

Today:
- Chapter 6, Sections, 6.2.3-6, 6.4 (UDP), 6.5 (TCP), 6.6 (TCP Performance)

Tuesday March 10
- Chapter 7.1 (DNS), 7.2-3
- Chapter 7.4 (Multimedia)
Internet Layering

Level 5  -- **Application** Layer
          (rlogin, ftp, SMTP, POP3, IMAP, HTTP..)

Level 4  -- **Transport** Layer (a.k.a. Host-to-Host)
          (*TCP, UDP*)

Level 3  -- **Network** Layer (a.k.a. Internet)
          (*IP, ICMP, ARP*)

Level 2  -- **(Data) Link** Layer / MAC sub-layer
          (a.k.a. Network Interface or Network Access Layer)

Level 1  -- **Physical** Layer
Transport Layer – Connection Oriented Service

- Addressing / Access Points
- Connection Request / Establishment
  - Delayed Duplicate Packet Problem
  - Timer management
  - Connection points
- Connection Release
Connection Release

• Asymmetric release: telephone system.
  – When one party hangs up, connection breaks.
  – May cause data loss.

• Symmetric release:
  – Treats connection as 2 separate unidirectional connections.
  – Requires each to be released separately.
Symmetric Release

• How to determine when all data has been sent and connection could be released?

• 2-army problem:

  Blue army 1
  ________________________
  . White army larger than either blue armies.
  . Blue army together is larger.

  White army

  Blue army 2

  . If each blue army attacks, it’ll be defeated. They win if attack together.
2-Army Problem 1

• To synchronize attack, they must use messengers that need to cross valley: unreliable.

• Is there a protocol that allows blue army to win? No.
  – Blue army 1 sends message to blue army 2.
  – Blue army 2 sends ACK back.
  – Blue army 2 is not sure whether ACK was received.
2-Army Problem 2

• Use 2-way handshake.
  – Blue army 1 ACKs back but it’ll never know if the ACK was received.

• Applying to connection release:
  – Neither side is prepared to disconnect until convinced other side is prepared to disconnect.
  – In practice, hosts are willing to take risks.
Connection Release Protocol

Send DR+ start timer

DR

Release connection

Send ACK

DR: disconnection request.

DR

Send DR+ start timer

ACK

Release connection
Connection Release Scenarios

1

Send DR+
start timer

Release connection

Send ACK

DR

DR

ACK

DR: disconnection request.

Send DR+
start timer

Timeout:
Release connection
Connection Release Scenarios

2

Send DR+ start timer

Timeout: send DR+ start timer

Send DR+ start timer

DR: disconnection request.

Send DR+ start timer

Release connection

Send DR+ start timer

ACK

DR

DR

DR

DR

DR

DR
Transport Layer

- Assumes Network is unreliable
- Sender buffers all TPDUs until acknowledged
- Receiver may also buffer
  - Difficult to decide buffer size as TPDUs
  - Dynamic buffer allocation may be best strategy
  - Buffer size negotiated by exchange of control packets
  - Deadlock can occur if control packets get lost
    - Solution is periodic sending of control TPDUs containing acknowledgements and buffer status
Flow Control and Buffering

- Premise: Network is unreliable
  - Consequence: Sender must buffer all TPDUs sent
  - May need to retransmit

- Receiver needs to buffer
  - Out of order TPDUs
  - Flow control to application(s) (maximize transmission speed)

- Sender TP layer also buffers to even flow from its application(s) (smooth out bursts to match network)

- Buffering negotiated between sender and receiver

- Hosts (clients and servers) may have each have many connections (in contrast to link layer point-to-point)

- Buffer pool
Flow Control and Buffering

(a) Chained fixed-size buffers.  (b) Chained variable-sized buffers.  
(c) One large circular buffer per connection.
Multiplexing

• Upward Multiplexing
  – One IP address shared by multiple Transport Layer processes (ports)

• Downward Multiplexing
  – Use multiple virtual circuits to get more bandwidth (e.g. join multiple ISND lines to get a higher bandwidth)
Multiplexing

(a) Upward multiplexing

(b) Downward multiplexing

Layer

4

3

2

1

To router

Router lines

Transport address

Network address
Crash Recovery

- Transport layer can handle crashes of network layer
  - Transport expects errors in network
- Machine crash causes transport layer crash
  - Recovery more challenging
  - (Related to database “commit”)
  - Cannot be managed at Transport Layer
## Crash Recovery

### Strategy used by receiving host

<table>
<thead>
<tr>
<th>Strategy used by sending host</th>
<th>First ACK, then write</th>
<th>First write, then ACK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AC(W)</td>
<td>AWC</td>
</tr>
<tr>
<td>Always retransmit</td>
<td>OK</td>
<td>DUP</td>
</tr>
<tr>
<td>Never retransmit</td>
<td>LOST</td>
<td>OK</td>
</tr>
<tr>
<td>Retransmit in S0</td>
<td>OK</td>
<td>DUP</td>
</tr>
<tr>
<td>Retransmit in S1</td>
<td>LOST</td>
<td>OK</td>
</tr>
</tbody>
</table>

**OK** = Protocol functions correctly  
**DUP** = Protocol generates a duplicate message  
**LOST** = Protocol loses a message

### Client States:

- **S0** – no TPDUs outstanding  
- **S1** – one TPDU outstanding
End-to-end Ack

- Perfect end-to-end acknowledgement means
  1. if you get it the `ack`, the task was completed
  2. if you do not get the `ack`, the task was not completed

- (Can’t do both at the same time and guarantee either both or neither were done.)

- Example: database update (transaction “commit”)

- “Probably impossible” (Saltzer, 1984)
Recovery “Principal”

• Layer above, with enough status information, can recover from crash of the layer below.

Recovery from crash at layer N needs to be done at layer N+1 (who must have necessary status info.)
The Internet Transport Protocols: UDP (User Datagram Protocol)

• Introduction to UDP
  – Connectionless
  – IP with a short header
    • Carries source and destination ports
    • Sends segments with 8 byte header
Introduction to UDP

- The UDP header.

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP length</td>
<td>UDP checksum</td>
</tr>
</tbody>
</table>
The Internet Transport Protocols: UDP (User Datagram Protocol)

• Introduction to UDP
  – Connectionless
  – IP with a short header (carries ports)
  – Sends segments with 8 byte header
  – No flow control
  – No error control
  – No retransmission for bad segments
The Internet Transport Protocols: UDP (User Datagram Protocol)

- Introduction to UDP
  - Connectionless
  - IP with a short header (carries ports)
  - *Used with client-server and real-time multimedia*
The Internet Transport Protocols: UDP
(User Datagram Protocol)

• Introduction to UDP
  – Connectionless
  – IP with a short header (carries ports)
    – *Used with client-server and real-time multimedia*
  
• Remote Procedure Call

• The Real-Time Transport Protocol
Remote Procedure Call

Steps in making a remote procedure call:

1. Client CPU
   - Client makes a local procedure call
   - Client stub
   - 1

2. Operating system
   - 2

3. Network
   - 3

4. Server CPU
   - Server stub
   - 4

5. Server
   - 5

Client makes a local procedure call -- an alternative to sockets
Real-time Transport Protocol (RTP)

- Basic function: support multiplexing of real-time data streams into a single stream of UDP packets
- Header field specifies encoding
- May include timestamps – give relative time for each packet (for synchronization of audio and video, and to help reduce jitter)
- RTP numbers packets sequentially – so receiver can determine if any missing
- No resend – better for multimedia if receiver just interpolates to create missing packet
(a) The position of RTP in the protocol stack.
(b) Packet nesting.
The Real-Time Transport Protocol (2)

The RTP header.

<table>
<thead>
<tr>
<th>Ver.</th>
<th>P</th>
<th>X</th>
<th>CC</th>
<th>M</th>
<th>Payload type</th>
<th>Sequence number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timestamp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Synchronization source identifier</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Contributing source identifier</td>
<td></td>
</tr>
</tbody>
</table>
The Internet Transport Protocols: TCP

- Introduction to TCP
- The TCP Service Model
- The TCP Protocol
- The TCP Segment Header
- TCP Connection Establishment
- TCP Connection Release
- TCP Connection Management Modeling
- TCP Transmission Policy
- TCP Congestion Control
- TCP Timer Management
- Wireless TCP and UDP
- Transactional TCP
Transmission Control Protocol (TCP)

- Reliable end-to-end
- Expects unreliable network
- 1970s – Vint Cerf and Robert Kahn
- Minimize network dependencies
- Library or user process or in OS on host machines
- Ports (TSAPs)
- Byte stream oriented (not messages)
- Full duplex – point-to-point
  (no multicasting or broadcasting)
The TCP Service Model

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>FTP</td>
<td>File transfer</td>
</tr>
<tr>
<td>23</td>
<td>Telnet</td>
<td>Remote login</td>
</tr>
<tr>
<td>25</td>
<td>SMTP</td>
<td>E-mail</td>
</tr>
<tr>
<td>69</td>
<td>TFTP</td>
<td>Trivial File Transfer Protocol</td>
</tr>
<tr>
<td>79</td>
<td>Finger</td>
<td>Lookup info about a user</td>
</tr>
<tr>
<td>80</td>
<td>HTTP</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>110</td>
<td>POP-3</td>
<td>Remote e-mail access</td>
</tr>
<tr>
<td>119</td>
<td>NNTP</td>
<td>USENET news</td>
</tr>
</tbody>
</table>

Some assigned ports – over 300 assigned “well-known” ports have number < 1024.
The TCP Service Model

Sends byte stream (not a message stream)
(a) Four 512-byte segments sent as separate IP datagrams.
(b) The 2048 bytes of data delivered to the application in a single READ CALL.
The TCP Segment Header

Source port | Destination port

Sequence number

Acknowledgement number

TCP header length | U R G A C S H P R S Y T N F I N

Window size

Checksum | Urgent pointer

Options (0 or more 32-bit words)

Data (optional)

TCP Header.
The TCP Segment Header (2)

The pseudoheader included in the TCP checksum.

<table>
<thead>
<tr>
<th>Source address</th>
<th>Protocol = 6</th>
<th>TCP segment length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

32 Bits
TCP and High Bandwidth

- 64-KB window problem
- T3 (44.736 M bps) 12 msec to output full 64-KB window.
- Transcontinental line round trip delay approx. 50 msec
- So significant idle time on line
- (Worse on satellite links!!)
- “Fix” is larger window field (now 16bits)
RFC 1323 – Window Scale Option

• Sender and receiver negotiate a “window size” shift
• Up to 14 bits (left)
• Can allow $2^{30}$ bytes in TCP window
TCP Connection Establishment

3-Way Handshake
(a) TCP connection establishment in the normal case.
(b) Call collision (simultaneous attempt).
Connection Release

- Full duplex connection
- Treat as pair of simplex connections
- Either sends a FIN
- Then other sends
- Timers to avoid “two army” problem
- Timer set for two maximum packet lifetimes
TCP Connection Management Modeling

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSED</td>
<td>No connection is active or pending</td>
</tr>
<tr>
<td>LISTEN</td>
<td>The server is waiting for an incoming call</td>
</tr>
<tr>
<td>SYN RCVD</td>
<td>A connection request has arrived; wait for ACK</td>
</tr>
<tr>
<td>SYN SENT</td>
<td>The application has started to open a connection</td>
</tr>
<tr>
<td>ESTABLISHED</td>
<td>The normal data transfer state</td>
</tr>
<tr>
<td>FIN WAIT 1</td>
<td>The application has said it is finished</td>
</tr>
<tr>
<td>FIN WAIT 2</td>
<td>The other side has agreed to release</td>
</tr>
<tr>
<td>TIMED WAIT</td>
<td>Wait for all packets to die off</td>
</tr>
<tr>
<td>CLOSING</td>
<td>Both sides have tried to close simultaneously</td>
</tr>
<tr>
<td>CLOSE WAIT</td>
<td>The other side has initiated a release</td>
</tr>
<tr>
<td>LAST ACK</td>
<td>Wait for all packets to die off</td>
</tr>
</tbody>
</table>

The states used in the TCP connection management finite state machine.
TCP Connection Management Modeling (2)

TCP connection management finite state machine. The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events. Each transition is labeled by the event causing it and the action resulting from it, separated by a slash.
TCP Transmission Policy

Window management in TCP – managed separately from acknowledgements.
Improvement

- Excessive handshaking with window updates and acknowledgments
- Improvement: Delay acknowledgements and window updates for 500 msec to possibly piggy-back with data being sent
- Still sending 41-byte packets containing as little as 1 byte of data
- Nagle’s algorithm – send first byte, buffer rest until first byte is acknowledged (or have ½ window)
- Good for keystrokes, not good for other apps.
TCP Transmission Policy (2)

Silly window syndrome.
Congestion Control

• Why do it at the transport layer?
• (Already have congestion control and flow control at the network and link layers respectively)
Congestion Control

• Why do it at the transport layer?
  – Real fix to congestion is to slow down the sender.
Congestion Control

• Why do it at the transport layer?
  – Real fix to congestion is to slow down the sender.

• Use law of “conservation of packets”.
  – Keep number of packets in the network constant.
  – Don’t inject new packet until old one leaves.

• Congestion indicator: packet loss.
TCP Congestion Control 1

- Window based -- like flow control
  - Sender keeps *congestion window* (cwin).
  - Each sender keeps 2 windows: receiver’s advertised window and congestion window.
  - Number of bytes that may be sent is \( \min(\text{advertised window, cwin}) \).
TCP Congestion Control

- A fast network feeding a low capacity receiver.
- A slow network feeding a high capacity receiver.
TCP Congestion Control 2

• Slow start [Jacobson 1988]:
  – Connection’s congestion window starts at 1 segment.
  – If segment ACKed before time out, cwin=cwin+1.
  – As ACKs come in, current cwin is increased by 1.
  – Exponential increase.
TCP Congestion Control 3

• Congestion Avoidance:
  – Third parameter: *threshold*.
  – Initially set to 64KB.
  – If timeout, threshold=cwin/2 and cwin=1.
  – Re-enters slow-start until cwin=threshold.
  – Then, cwin grows linearly until it reaches receiver’s advertised window.
TCP Congestion Control: Example

cwin

threshold

timeout

threshold

time
TCP Retransmission Timer

• When segment sent, retransmission timer starts.
  – If segment ACKed, timer stops.
  – If time out, segment retransmitted and timer starts again.
How to set timer?

• Based on round-trip time: time between a segment is sent and ACK comes back.
• If timer is too short, unnecessary retransmissions.
• If timer is too long, long retransmission delay.
Jacobson’s Algorithm 1

- Determining the round-trip time:
  - TCP keeps $RTT$ variable.
  - When segment sent, TCP measures how long it takes to get ACK back ($M$).
  - $RTT = \alpha \times RTT + (1-\alpha)M$.
  - $\alpha$: smoothing factor; determines weight given to previous estimate.
  - Typically, $\alpha = 7/8$.  

\[ RTT = \alpha \times RTT + (1-\alpha)M. \]
Jacobson’s Algorithm 2

• Determining timeout value:
  – Measure RTT variation, or |RTT-M|.
  – Keeps smoothed value of cumulative variation
    \[ D = \alpha D + (1-\alpha)|RTT-M| \].
  – Alpha may or may not be the same as value used to
    smooth RTT.
  – Timeout = RTT + 4*D.
Karn’s Algorithm

• How to account for ACKs of retransmitted segments?
  – Count it for first or second transmission?
  – Karn proposed not to update RTT on any retransmitted segment.
  – Instead RTT is doubled on each failure until segments get through.
Persistence Timer

- Prevents deadlock if an window update packet is lost and advertised window = 0.
- When persistence timer goes off, sender probes receiver; receiver replies with its current advertised window.
- If = 0, persistence timer is set again.
Keepalive Timer

• Goes off when a connection is idle for a long time.

• Causes one side to check whether the other side is still alive.

• If no answer, connection terminated.
TCP Timer Management

- (a) Probability density of ACK arrival times in the data link layer.
- (b) Probability density of ACK arrival times for TCP.