Time & Scope

- Next Mon: 12:30 - 1:20
  - Online on canvas
  - No need to come to lecture
- Everything up until TCP congestion control
Internet Reference Model - Layering

- The classic OSI model has seven layers
- In practice, there are more like four

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
</table>
| Application | Application-specific data  
|            | *HTTP, SMTP, POP (project 0)*                                                 |
| Transport  | Move data between applications anywhere on the Internet.  
|            | *UDP, TCP*                                                                   |
| Network    | Move data from one machine to another, anywhere on the Internet.  
|            | *IP*                                                                         |
| Link       | Move data between nodes that can hear each other’s transmissions.  
|            | *Ethernet, WiFi*                                                             |
Encapsulation/Decapsulation

message
segment
datagram
frame

source

application
transport
network
link
physical

destination

application
transport
network
link
physical

switch

router
Application Layer - DNS

- Built on UDP messages, port 53
- Names ↔ Addresses
- Resolution
- Iterative vs recursive
- Cache
Application Layer - HTTP

• Static vs Dynamic Web pages
• URL structures
• Browser steps:
  • Resolve the server name to an IP address (DNS)
  • Set up a TCP connection to the server
  • Send HTTP request for the page
  • Wait for and then read HTTP response
  • (Assuming no errors) Process response data and render page
  • Clean up any idle TCP connections
• HTTP methods
• HTTP response codes
• HTTP 1.0 vs HTTP 1.1 vs HTTP 2.0
## Transport

<table>
<thead>
<tr>
<th></th>
<th>Reliable</th>
<th>Unreliable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Packets</strong></td>
<td></td>
<td>Datagrams (UDP)</td>
</tr>
<tr>
<td><strong>Bytestream</strong></td>
<td></td>
<td>Streams (TCP)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streams</td>
<td>Datagrams</td>
</tr>
<tr>
<td>Connections</td>
<td>Connectionless</td>
</tr>
<tr>
<td>Bytes are delivered to receiving app reliably (once, and in order)</td>
<td>Packets may be lost, reordered, duplicated (but not corrupted)</td>
</tr>
<tr>
<td>Arbitrary length content</td>
<td>Fixed maximum datagram size</td>
</tr>
<tr>
<td>Connection latency</td>
<td>No delay</td>
</tr>
<tr>
<td>Segment delivery latency (&quot;nagling&quot;)</td>
<td>Datagram is sent now</td>
</tr>
<tr>
<td>Flow control matches sender’s rate to receiver’s capability</td>
<td>No flow control (can lead to many lost datagrams)</td>
</tr>
<tr>
<td>Congestion control matches sender’s rate to network’s capability</td>
<td>No congestion control (can lead to many lost datagrams)</td>
</tr>
</tbody>
</table>
Transport – Mux/demux – UDP

```java
DatagramSocket
mySocket2 = new DatagramSocket
(9157);

DatagramSocket
serverSocket = new DatagramSocket
(6428);

DatagramSocket
mySocket1 = new DatagramSocket
(5775);
```
Transport – Mux/demux – TCP

host: IP address A
source IP, port: B,80
dest IP, port: A,9157

source IP, port: A,9157
dest IP, port: B,80

host: IP address C
source IP, port: C,9157
dest IP, port: B,80

source IP, port: C,5775
dest IP, port: B,80

threaded server
Transport – TCP

• Reliable transfer – retransmission
  • Stop-and-wait
  • Sliding windows
  • Go-Back-N
  • Selective Repeat
Transport – TCP

- Connection establishment
- Flow control
- Sliding windows
- Connection teardown