Outline

- Administrivia
- Icebreaker
- Project 1 and Socket API
- Traceroute
Administrivia

- 3 Projects (15% + 17% + 18%)
  - Post on Ed if you cannot find a partner
- Assignments (10%)
  - Assignment 1: Read the textbook!
  - What is RTT?
- Surprise Quizzes (5%)
- Midterm (15%) and Final (20%)
Ice Breakers

- Introduce yourself
  - Name
  - Pronouns
  - Favorite non-cs class
- As a group, decide on a favorite ice cream flavor
Network-Application Interface

- Defines how apps use the network
  - Application Layer APIs
  - Lets apps talk to each other
  - hides the other layers of the network
Project 1

- **Simple Client**
  - Send requests to attu server
  - Wait for a reply
  - Extract the information from the reply
  - Continue...

- **Simple Server**
  - Server handles the Client requests
  - Multi-threaded
Project 1

- This is the basis for many apps!
  - File transfer: send name, get file (§6.1.4)
  - Web browsing: send URL, get page
  - Echo: send message, get it back

- Let’s see how to write this app ...
Socket API (Generalized)

- Simple application-layer abstractions (APIs) to use the network
  - The network service API used to write all Internet applications
  - Part of all major OSes and languages; originally Berkeley (Unix) ~1983

- Two kinds of sockets
  - Streams (TCP): reliably send a stream of bytes
  - Datagrams (UDP): unreliably send separate messages
Socket API (2)

- **Sockets** let apps attach to the local network at different **ports**
- **Ports** are used by OS to distinguish services/apps using internet
## Socket API (3)

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCKET</td>
<td>Create a new communication endpoint</td>
</tr>
<tr>
<td>BIND</td>
<td>Associate a local address (port) with a socket</td>
</tr>
<tr>
<td>LISTEN</td>
<td>Announce willingness to accept connections; (give queue size)</td>
</tr>
<tr>
<td>ACCEPT</td>
<td>Passively establish an incoming connection</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Actively attempt to establish a connection</td>
</tr>
<tr>
<td>SEND</td>
<td>Send some data over the connection</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>Receive some data from the connection</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Release the connection</td>
</tr>
</tbody>
</table>

**Computer Networks**

[https://docs.oracle.com/javase/8/docs/api/java/net/Socket.html](https://docs.oracle.com/javase/8/docs/api/java/net/Socket.html)
[https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html](https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html)
Using TCP Sockets

Client (host 1) \( \text{Time} \) Server (host 2)
Using TCP Sockets (2)

Client (host 1) \[\text{Time}\] Server (host 2)

connect

1 ← --- → 1
Using TCP Sockets (2)

Client (host 1)  Time  Server (host 2)

1 ← connect ← 1

2  request  
Using TCP Sockets (2)

Client (host 1)  Time  Server (host 2)

connect

1 ← connect → 1

request

2 → request

reply

1 ← reply → 3

Computer Networks
Using TCP Sockets (2)

Client (host 1) \[\downarrow\] Time \[\downarrow\] Server (host 2)

1 \(\rightarrow\) connect \(\leftarrow\) 1

2 \(\rightarrow\) request

3 \(\leftarrow\) reply

4 \(\leftarrow\) disconnect \(\rightarrow\) 4
Using TCP Sockets (3)

Client (host 1)  Time  Server (host 2)

1: socket
2: (bind)
3: (listen)
4: accept*

*= call blocks
Using TCP Sockets (3)

Client (host 1)

1: socket

5: connect*

Server (host 2)

1: socket

2: (bind)

3: (listen)

4: accept*

Time

(connect)

*= call blocks
Using TCP Sockets (3)

Client (host 1)  Time  Server (host 2)

1: socket

connect

5: connect*

1: socket
2: (bind)
3: (listen)
4: accept*
5: connect*

6: recv*

*= call blocks
Using TCP Sockets (3)

Client (host 1)  Time  Server (host 2)

1: socket
5: connect*
7: send
8: recv*

connect

1: socket
2: (bind)
3: (listen)
4: accept*
6: recv*

request

*= call blocks
Using TCP Sockets (3)

Client (host 1)  Time  Server (host 2)

1: socket

connect

5: connect*

7: send
8: recv*

request

reply

1: socket
2: (bind)
3: (listen)
4: accept*

6: recv*

9: send
10: recv*

* = call blocks
Using TCP Sockets (3)

Client (host 1)  

1: socket  
5: connect*  
7: send  
8: recv*  
11: close  

Server (host 2)  

1: socket  
2: (bind)  
3: (listen)  
4: accept*  
6: recv*  
9: send  
10: recv*  
12: close

* = call blocks

request  
reply  
connect  
disconnect
# Using UDP Sockets

<table>
<thead>
<tr>
<th>Client (host 1)</th>
<th>Time</th>
<th>Server (host 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: socket</td>
<td></td>
<td>1: socket</td>
</tr>
<tr>
<td>5: connect*</td>
<td></td>
<td>2: (bind)</td>
</tr>
<tr>
<td>7: sendto</td>
<td></td>
<td>3: (listen)</td>
</tr>
<tr>
<td>8: recvfrom*</td>
<td></td>
<td>4: accept*</td>
</tr>
<tr>
<td>11: close</td>
<td></td>
<td>6: recvfrom*</td>
</tr>
</tbody>
</table>

10: recvfrom*  *= call blocks

9: sendto

12: close

*connect request*

reply

disconnect
Client Program (outline)

socket()  // make socket
getaddrinfo()  // server and port name
    // www.example.com:80
connect()  // connect to server

send()  // send request
recv()  // await reply [block]
...  // do something with data!
close()  // done, disconnect
Server Program
(outline)

socket()       // make socket
getaddrinfo()  // for port on this host
bind()         // associate port with socket
listen()       // prepare to accept connections
accept()       // wait for a connection [block]
...
recv()         // wait for request [block]
...
send()         // send the reply
close()        // eventually disconnect
Java Examples with Socket & ServerSocket

### Server

```java
ServerSocket listener = new ServerSocket(9090);
    try {
        while (true) {
            Socket socket = listener.accept();
            try {
                socket.getInputStream();
            } finally {
                socket.close();
            }
        }
    } finally {
        listener.close();
    }
```

### Client

```java
Socket socket = new Socket(server, 9090);
    out =
        new PrintWriter(socket.getOutputStream(), true);
    socket.close();
```

- [http://cs.lmu.edu/~ray/notes/javanetexamples/](http://cs.lmu.edu/~ray/notes/javanetexamples/)
- [https://docs.oracle.com/javase/tutorial/networking/datagrams/clientServer.html](https://docs.oracle.com/javase/tutorial/networking/datagrams/clientServer.html)
- [https://docs.oracle.com/javase/tutorial/networking/sockets/index.html](https://docs.oracle.com/javase/tutorial/networking/sockets/index.html)
Python Examples with socket

Server

```python
listener = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
listener.bind(server_address)

while True:
    try:
        connection, client_addr = listener.accept()
        try:
            connection.recv(n_bytes)
        finally:
            connection.close()
    except:
        listener.close()
```

Client

```python
socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket.connect(server_address)
socket.sendto(message, server_address)
socket.close();
```

- Python socket documentation
- UDP socket example
- socketserver (a little overkill)
Traceroute

- Apps talk to other apps with no real idea of what is inside the network
  - This is good! But you may be curious ...

- Peeking inside the Network with Traceroute
Traceroute

- Widely used command-line tool to let hosts peek inside the network
  - On all OSes (tracert on Windows)
  - Developed by Van Jacobson ~1987
  - Uses a network-network interface (IP) in ways we will explain later
Traceroute

- Probes successive hops to find network path
- Core mechanism: Time-To-Live (TTL)
  - TTL == 0?
    - Discard data, error (ICMP) report to sender:
      - Continue with TTL-1
Traceroute

Local Host

1 hop → 2 hops → 3 hops → N-1 hops → N hops

Remote Host

Computer Networks
Using Traceroute

```
C:\Users\djaw>tracert www.uw.edu

Tracing route to www.washington.edu [128.95.155.134] over a maximum of 30 hops:

1    1 ms <1 ms  2 ms  192.168.1.1
2    8 ms  8 ms  9 ms  88.Red-80-58-67.staticIP.rima-tde.net [80.58.67.88]
3   16 ms  5 ms  11 ms  169.Red-80-58-78.staticIP.rima-tde.net [80.58.78.169]
4   12 ms  12 ms  13 ms  217.Red-80-58-87.staticIP.rima-tde.net [80.58.87.217]
5    5 ms  11 ms  6 ms  et-1-0-0-1-101-GRIBONES1.red.telefonica-wholesale.net [94.142.103.20
6   38 ms  38 ms  38 ms  176.52.250.226
7   108 ms 106 ms 136 ms xe-6-0-2-0-grtncp2.red.telefonica-wholesale.net [213.140.43.9]
8   180 ms 179 ms 182 ms Ke9-2-0-0-grtpaopx2.red.telefonica-wholesale.net [94.142.118.178]
9   178 ms 175 ms 176 ms te-4-2.car1.SanJose2.Level3.net [4.59.0.225]
10   190 ms 186 ms 187 ms v1an80.csu3.SanJose1.Level3.net [4.69.152.190]
11   185 ms 185 ms 187 ms ae-82-82.eb1.SanJose1.Level3.net [4.69.153.25]
12   268 ms 205 ms 207 ms ae-7-7.eb1.Seattle1.Level3.net [4.69.132.50]
13   334 ms 202 ms 195 ms ae-12-51.car2.Seattle1.Level3.net [4.69.147.132]
14   195 ms 196 ms 195 ms PACIFIC-NOR.car2.Seattle1.Level3.net [4.53.146.142]
15   197 ms 195 ms 196 ms a0--4000.icrr-sttlwa01-02.infra.pnw-gigappp.net [209.124.188.132]
16   196 ms 196 ms 195 ms v14000.uwbr-ads-01.infra.washington.edu [209.124.188.133]
17    *    *    *    Request timed out.
18   201 ms 194 ms 196 ms a4--583.uwbr-ads-1.infra.washington.edu [128.95.155.131]
19   197 ms 196 ms 195 ms www1.cac.washington.edu [128.95.155.134]
```
Using Traceroute

<table>
<thead>
<tr>
<th>Hop</th>
<th>RTT 1</th>
<th>RTT 2</th>
<th>RTT 3</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 ms</td>
<td>&lt;1 ms</td>
<td>2 ms</td>
<td>192.168.1.1</td>
</tr>
<tr>
<td>2</td>
<td>8 ms</td>
<td>8 ms</td>
<td>9 ms</td>
<td>88.80-58-67.staticIP.rima-tde.net [80.58.67.68]</td>
</tr>
<tr>
<td>3</td>
<td>16 ms</td>
<td>5 ms</td>
<td>11 ms</td>
<td>169.80-58-78.staticIP.rima-tde.net [80.58.78.169]</td>
</tr>
<tr>
<td>4</td>
<td>12 ms</td>
<td>12 ms</td>
<td>13 ms</td>
<td>217.80-58-87.staticIP.rima-tde.net [80.58.87.217]</td>
</tr>
<tr>
<td>5</td>
<td>5 ms</td>
<td>11 ms</td>
<td>6 ms</td>
<td>et-1-0-0-1-101-GRIBONESI.red.telefonica-wholesale.net [94.142.103.20]</td>
</tr>
<tr>
<td>6</td>
<td>40 ms</td>
<td>38 ms</td>
<td>38 ms</td>
<td>176.52.250.226</td>
</tr>
<tr>
<td>7</td>
<td>108 ms</td>
<td>106 ms</td>
<td>136 ms</td>
<td>xe-6-0-2-0-grtncyt2.red.telefonica-wholesale.net [213.140.43.9]</td>
</tr>
<tr>
<td>8</td>
<td>180 ms</td>
<td>179 ms</td>
<td>182 ms</td>
<td>Ke9-2-0-0-grtppaqpx2.red.telefonica-wholesale.net [94.142.118.178]</td>
</tr>
<tr>
<td>9</td>
<td>178 ms</td>
<td>175 ms</td>
<td>176 ms</td>
<td>te-4-2.car1.SanJose2.Level3.net [4.59.0.225]</td>
</tr>
<tr>
<td>10</td>
<td>190 ms</td>
<td>186 ms</td>
<td>187 ms</td>
<td>vian80.cs.w3.SanJose1.Level3.net [4.69.152.190]</td>
</tr>
<tr>
<td>11</td>
<td>185 ms</td>
<td>185 ms</td>
<td>187 ms</td>
<td>ae-82-82.ebr2.SanJose1.Level3.net [4.69.153.25]</td>
</tr>
<tr>
<td>12</td>
<td>268 ms</td>
<td>205 ms</td>
<td>207 ms</td>
<td>ae-7-7.ebr1.Seattle1.Level3.net [4.69.132.50]</td>
</tr>
<tr>
<td>13</td>
<td>334 ms</td>
<td>202 ms</td>
<td>195 ms</td>
<td>ae-12-51.car2.Seattle1.Level3.net [4.69.147.132]</td>
</tr>
<tr>
<td>14</td>
<td>195 ms</td>
<td>196 ms</td>
<td>195 ms</td>
<td>PACIFIC-NOR.car2.Seattle1.Level3.net [4.53.146.142]</td>
</tr>
<tr>
<td>15</td>
<td>197 ms</td>
<td>195 ms</td>
<td>196 ms</td>
<td>ae0-4000.iccr-sttla01-02.infra.pnw-gigapop.net [209.124.188.132]</td>
</tr>
<tr>
<td>16</td>
<td>196 ms</td>
<td>196 ms</td>
<td>195 ms</td>
<td>v4000.wbr-ads-01.infra.washington.edu [209.124.188.133]</td>
</tr>
<tr>
<td>17</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>Request timed out.</td>
</tr>
<tr>
<td>18</td>
<td>201 ms</td>
<td>194 ms</td>
<td>196 ms</td>
<td>ae4-583.uwar-ads-1.infra.washington.edu [128.95.155.131]</td>
</tr>
<tr>
<td>19</td>
<td>197 ms</td>
<td>196 ms</td>
<td>195 ms</td>
<td>www1.cac.washington.edu [128.95.155.134]</td>
</tr>
</tbody>
</table>
Using Traceroute

Router settings affect results
Using Traceroute (2)

- ISP names and places are educated guesses

- My computer

- www.uw.edu (www1.cac.washington.edu)
END