

Computer Networks

The Socket API (Project 1) & Traceroute (HW 1)

(§1.3.4, 6.1.2-6.1.4)

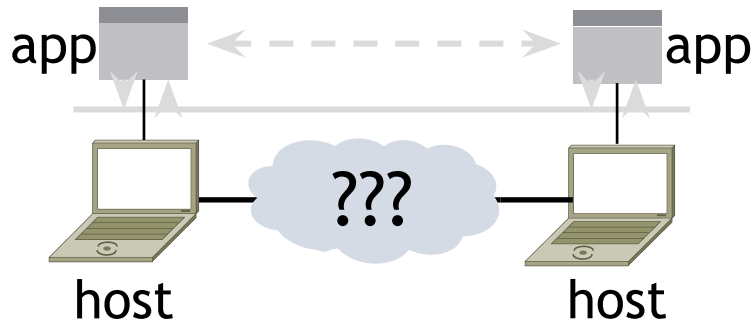
About Me

Esther Jang

- ▶ 3rd year PhD student
- ▶ Information and Communications Technology for Development
- ▶ Community Cellular Network deployments
- ▶ My goal is to get better at teaching.

Traceroute

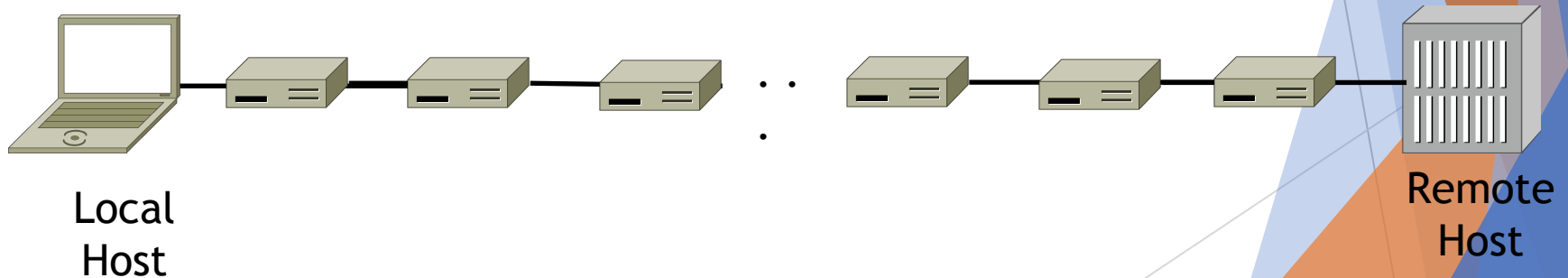
- ▶ Apps talk to other apps with no real idea of what is inside the network
 - ▶ This is good! But you may be curious ...
- ▶ Peeking inside the Network with Traceroute



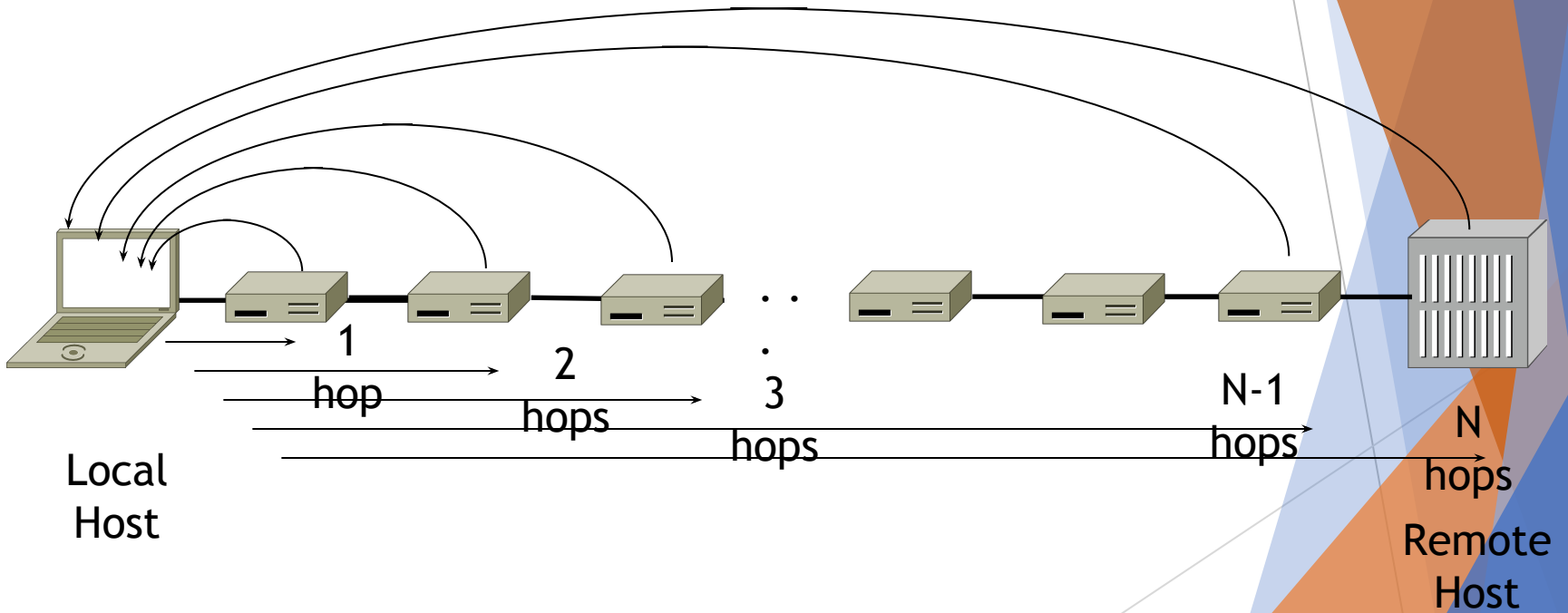
Traceroute

Traceroute

- ▶ Probes successive hops to find network path
- ▶ TTL: time-to-live



Traceroute



Using Traceroute

```
Administrator: Command Prompt
C:\Users\djw>tracert www.uw.edu

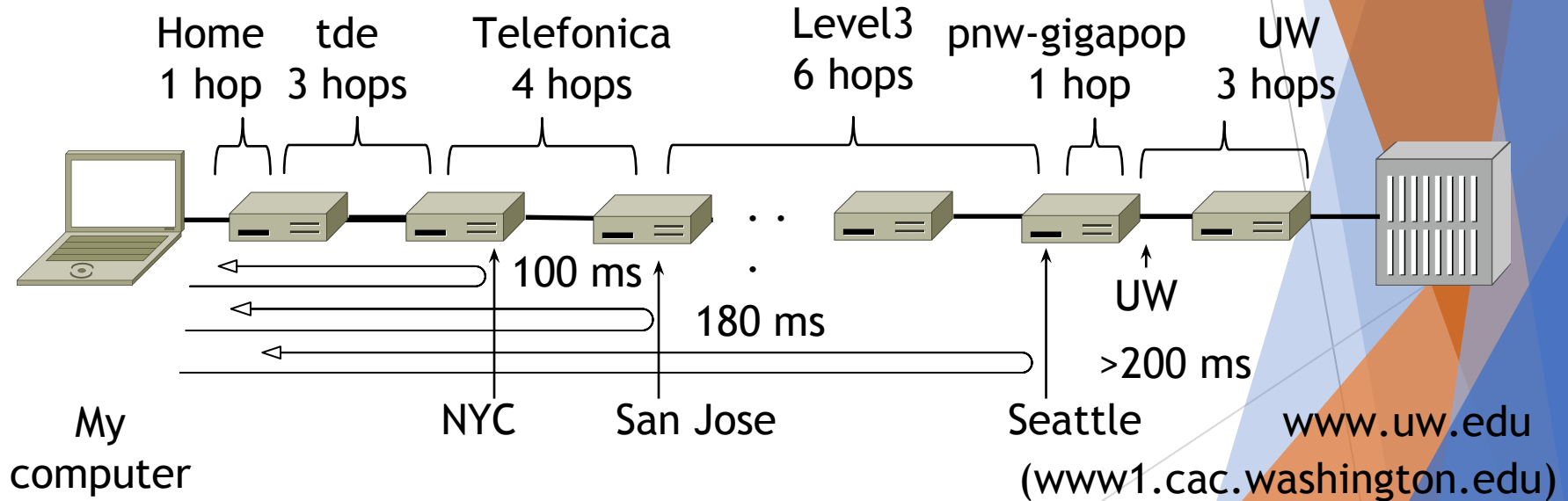
Tracing route to www.washington.edu [128.95.155.134]
over a maximum of 30 hops:

  0  1 ms    <1 ms   2 ms    192.168.1.1
  1  8 ms    8 ms    9 ms    88.Red-80-58-67.staticIP.rima-tde.net [80.58.67.88]
  2 16 ms    5 ms   11 ms   169.Red-80-58-78.staticIP.rima-tde.net [80.58.78.169]
  3 12 ms   12 ms  13 ms   217.Red-80-58-87.staticIP.rima-tde.net [80.58.87.217]
  4  5 ms    11 ms   6 ms    et-1-0-0-1-101-GRITBCNES1.red.telefonica-wholesale.net [94.142.103.205]
  5 40 ms   38 ms  38 ms   176.52.250.226
  6 108 ms  106 ms 136 ms  xe-6-0-2-0-grtnycpt2.red.telefonica-wholesale.net [213.140.43.9]
  7 180 ms  179 ms 182 ms  Xe9-2-0-0-grtpaopx2.red.telefonica-wholesale.net [94.142.118.178]
  8 178 ms  175 ms 176 ms  te-4-2.car1.SanJose2.Level3.net [4.59.0.225]
  9 190 ms  186 ms 187 ms  vlan80.csw3.SanJose1.Level3.net [4.69.152.190]
 10 185 ms  185 ms 187 ms  ae-82-82.ebr2.SanJose1.Level3.net [4.69.153.25]
 11 268 ms  205 ms 207 ms  ae-7-7.ebr1.Seattle1.Level3.net [4.69.132.50]
 12 334 ms  202 ms 195 ms  ae-12-51.car2.Seattle1.Level3.net [4.69.147.132]
 13 195 ms  196 ms 195 ms  PACIFIC-NOR.car2.Seattle1.Level3.net [4.53.146.142]
 14 197 ms  195 ms 196 ms  ae0--4000.iccr-sttlwa01-02.infra.pnw-gigapop.net [209.124.188.132]
 15 196 ms  196 ms 195 ms  v14000.uwbr-ads-01.infra.washington.edu [209.124.188.133]
 16 *      *      *
 17 *      *      *
 18 201 ms  194 ms 196 ms  ae4--583.uwar-ads-1.infra.washington.edu [128.95.155.131]
 19 197 ms  196 ms 195 ms  www1.cac.washington.edu [128.95.155.134]

Trace complete.
```

Using Traceroute (2)

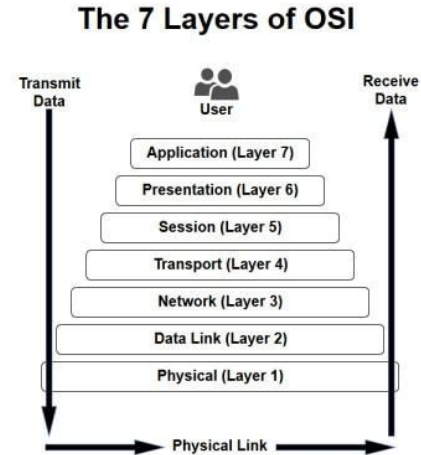
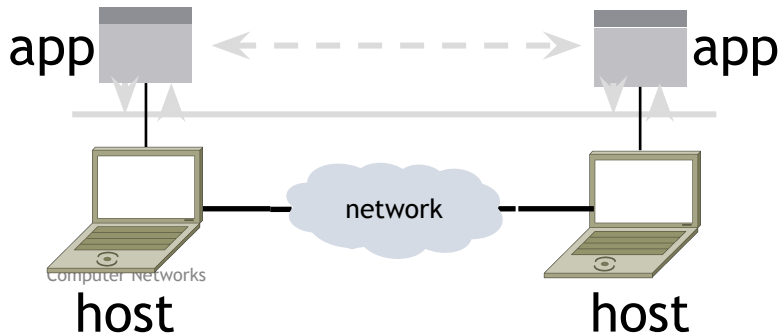
- ISP names and places are educated guesses



Client-server interaction

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 server
 - ▶ Wait for a reply
 - ▶ Extract the information from the reply
 - ▶ Continue...
- ▶ Simple Server
 - ▶ Server handles the Client requests
 - ▶ Multi-threaded

Project 1

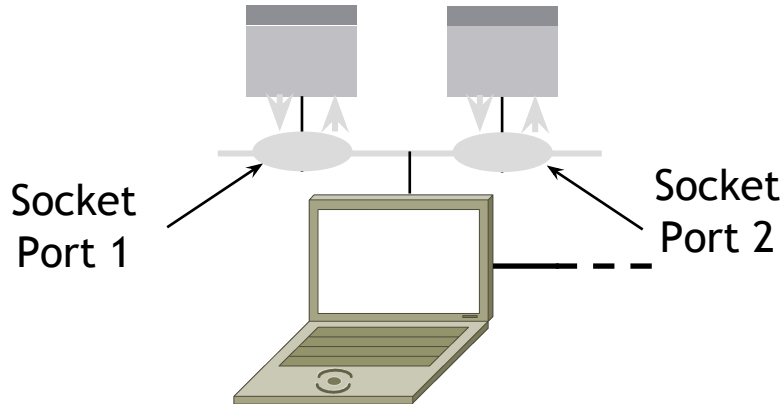
- ▶ This is the basis for many apps!
 - ▶ File transfer: send name, get file (§6.1.4)
 - ▶ Web browsing: send URL, get page
 - ▶ Echo: send message, get it back
- ▶ Let's see how to write this app ...

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)

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

Using Sockets

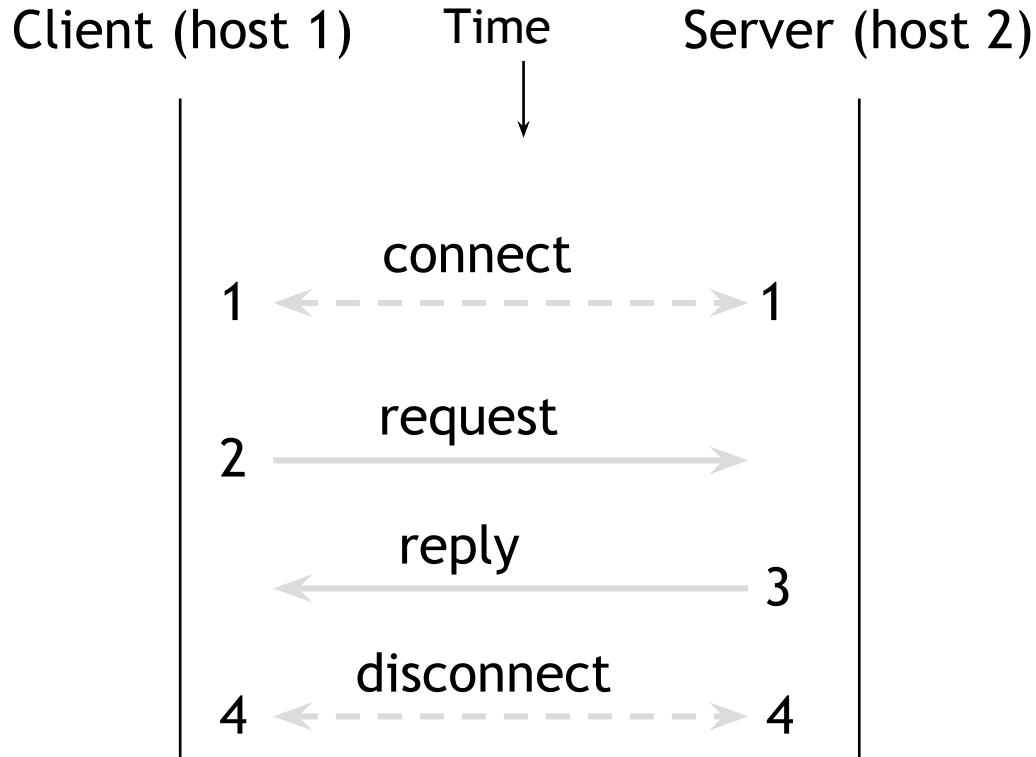
Client (host 1)

Time

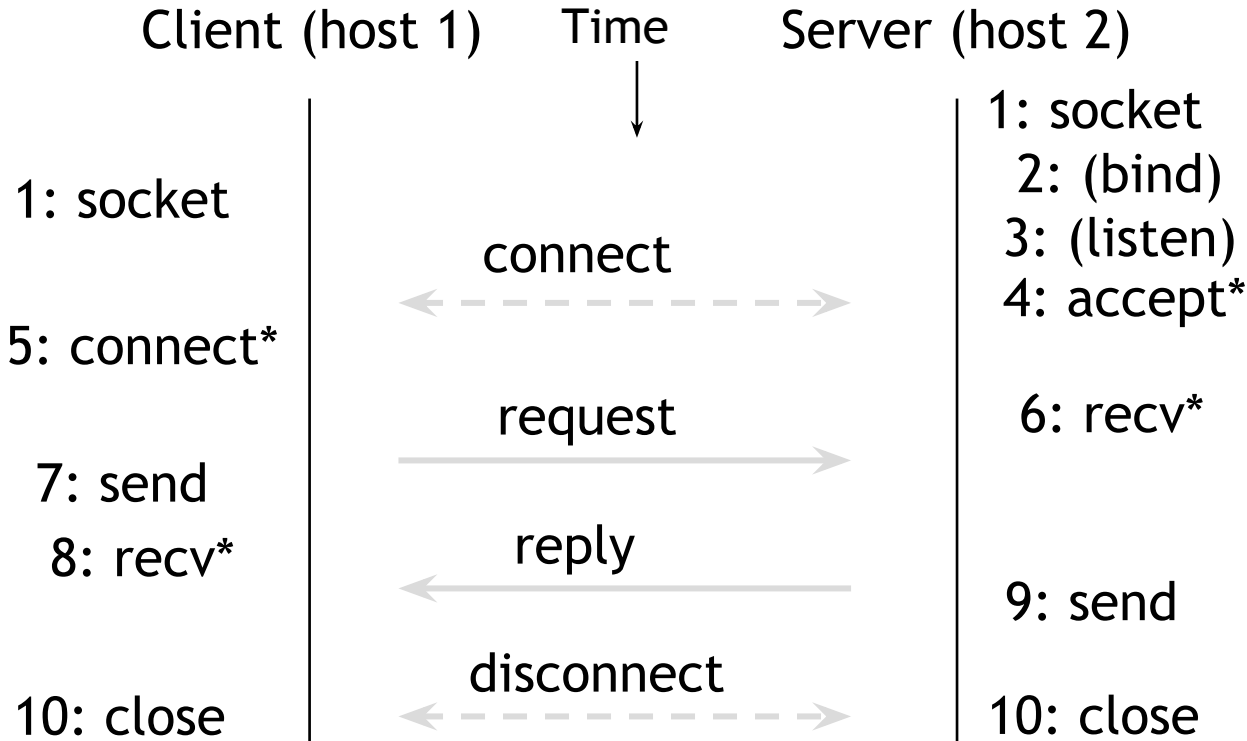
Server (host 2)



Using Sockets (2)



Using Sockets (3)



*= call blocks

Client Program (outline)

```
socket()      // make socket
getaddrinfo() // server and port name
              // www.example.com:80
connect()     // connect to server [block]
...
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
...
send()            // send the reply
close()           // eventually disconnect
```

Java Examples with Socket & ServerSocket

▶ Server

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

▶ Client

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

- <http://cs.lmu.edu/~ray/notes/javanetexamples/>
- <https://docs.oracle.com/javase/tutorial/networking/datagrams/clientServer.html>
- <https://docs.oracle.com/javase/tutorial/networking/sockets/index.html>

END

© 2013 D. Wetherall

Slide material from: TANENBAUM, ANDREW S.; WETHERALL, DAVID J., **COMPUTER NETWORKS**, 5th Edition, © 2011. Electronically reproduced by permission of Pearson Education, Inc., Upper Saddle River, New Jersey