Computer Networks

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

(§1.3.4, 6.1.2-6.1.4)

Originally By David Wetherall (djw@), Modified By Qian Yan and Esther Jang (ether@)

UNIVERSITY of WASHINGTON

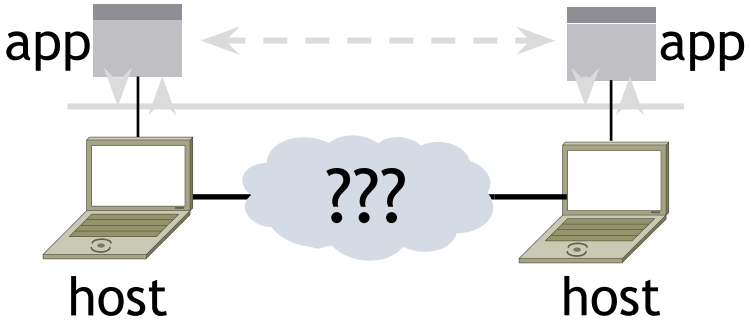
About Me

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

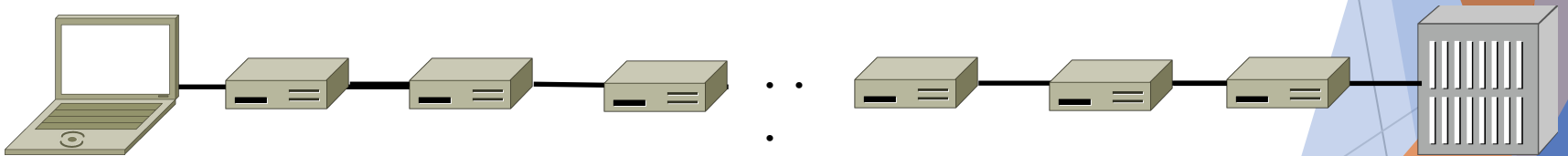
- Apps talk to other apps with no real idea of what is inside the network
  - This is good! But you may be curious …
- Peeking inside the Network with Traceroute



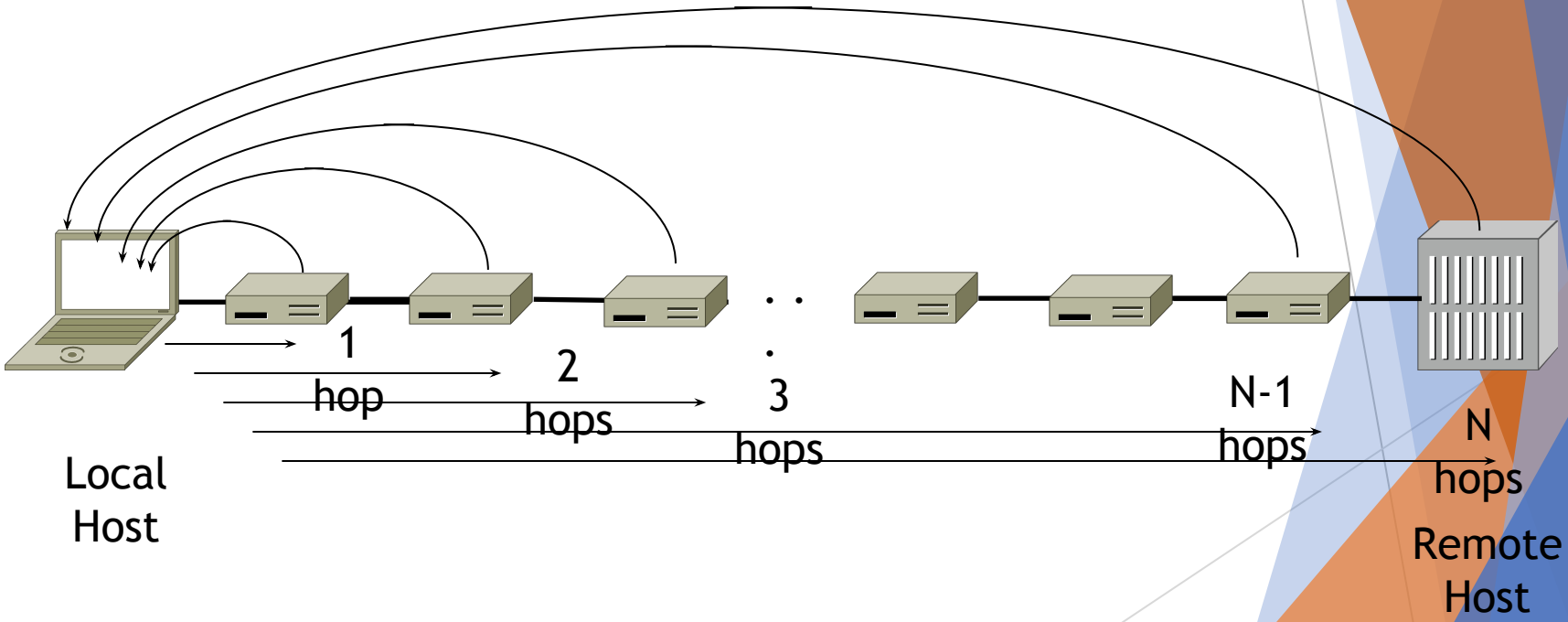
Traceroute

Traceroute

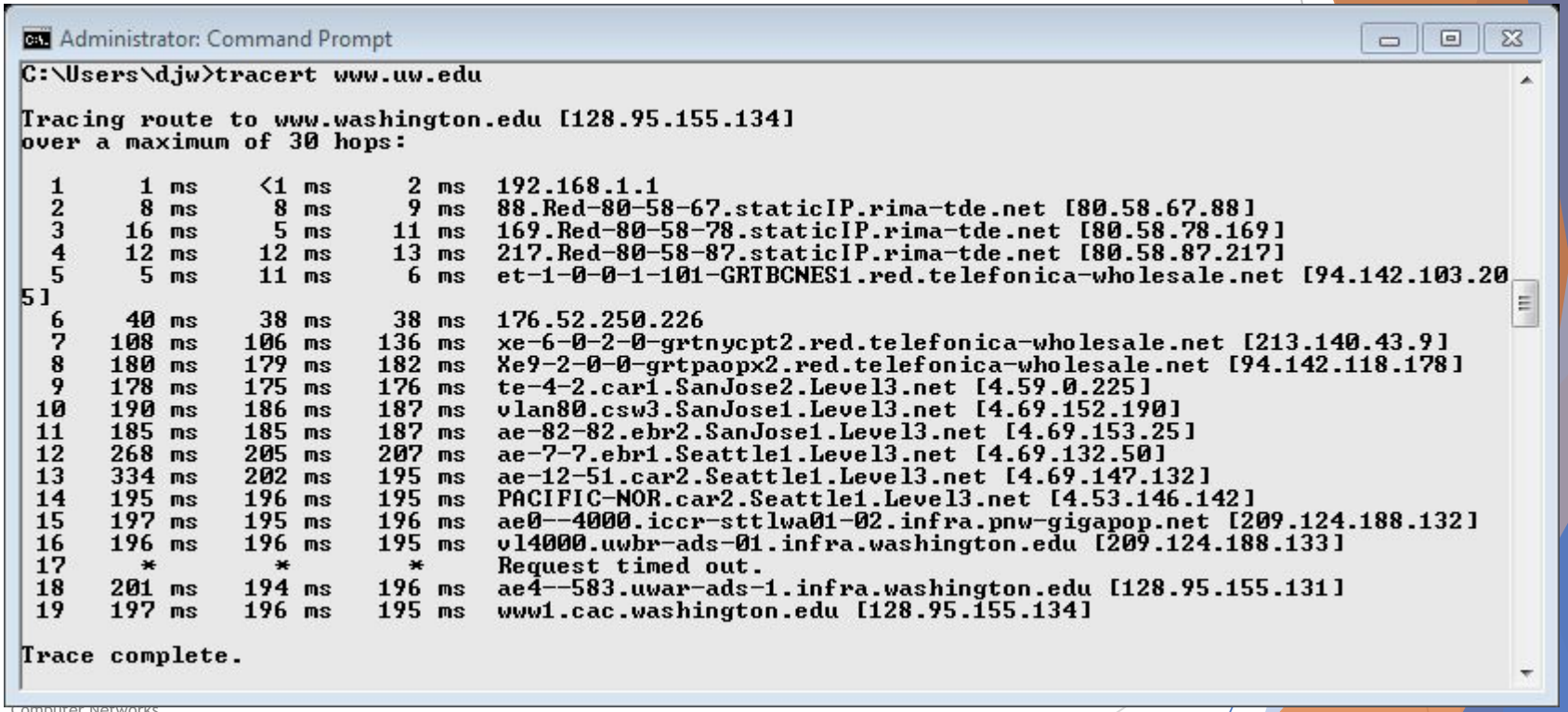
- Probes successive hops to find network path
- TTL: time-to-live



Traceroute

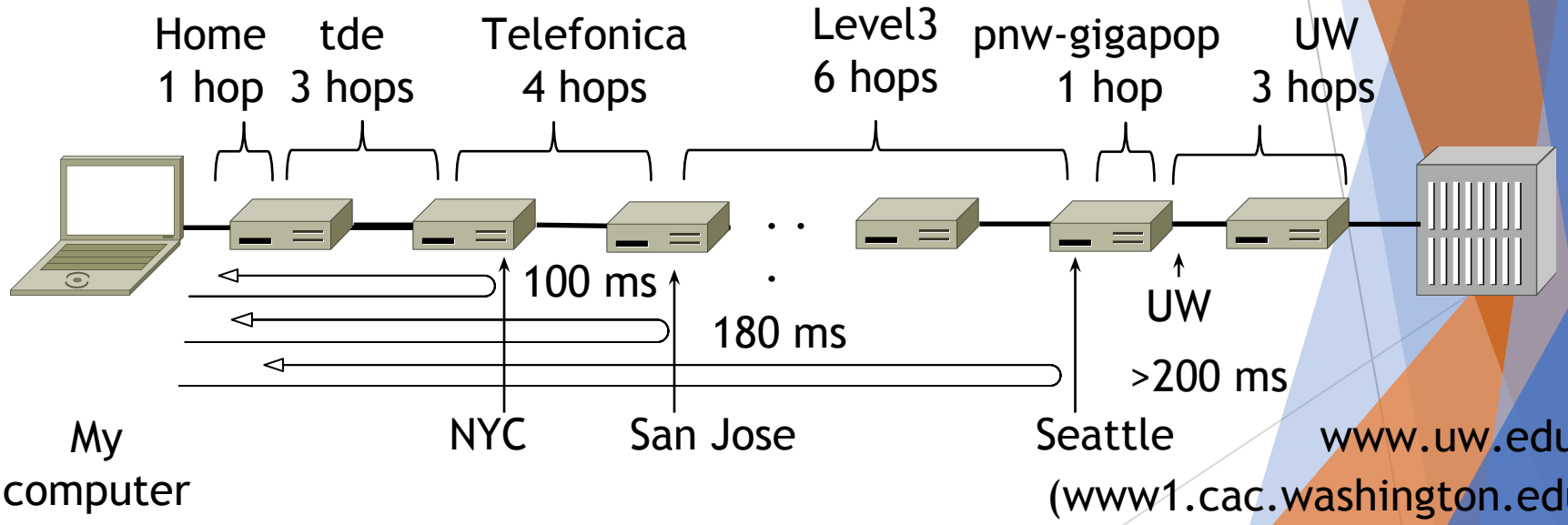


Using Traceroute



Using Traceroute (2)

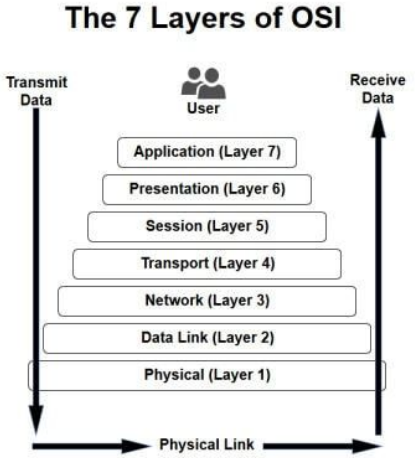
- ISP names and places are educated guesses

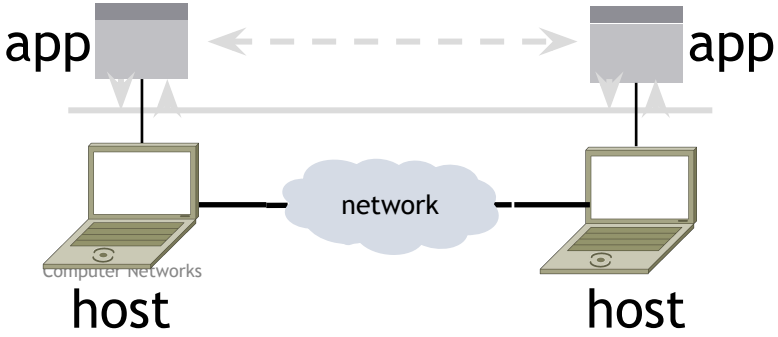


Client-server interaction

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 …

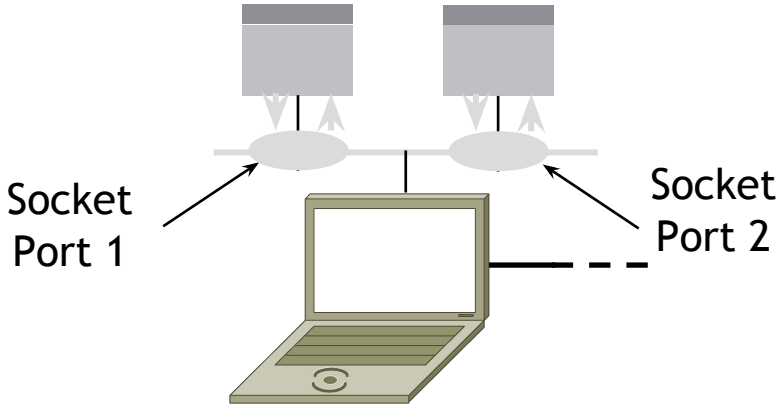
Socket API (Generalized)

- 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

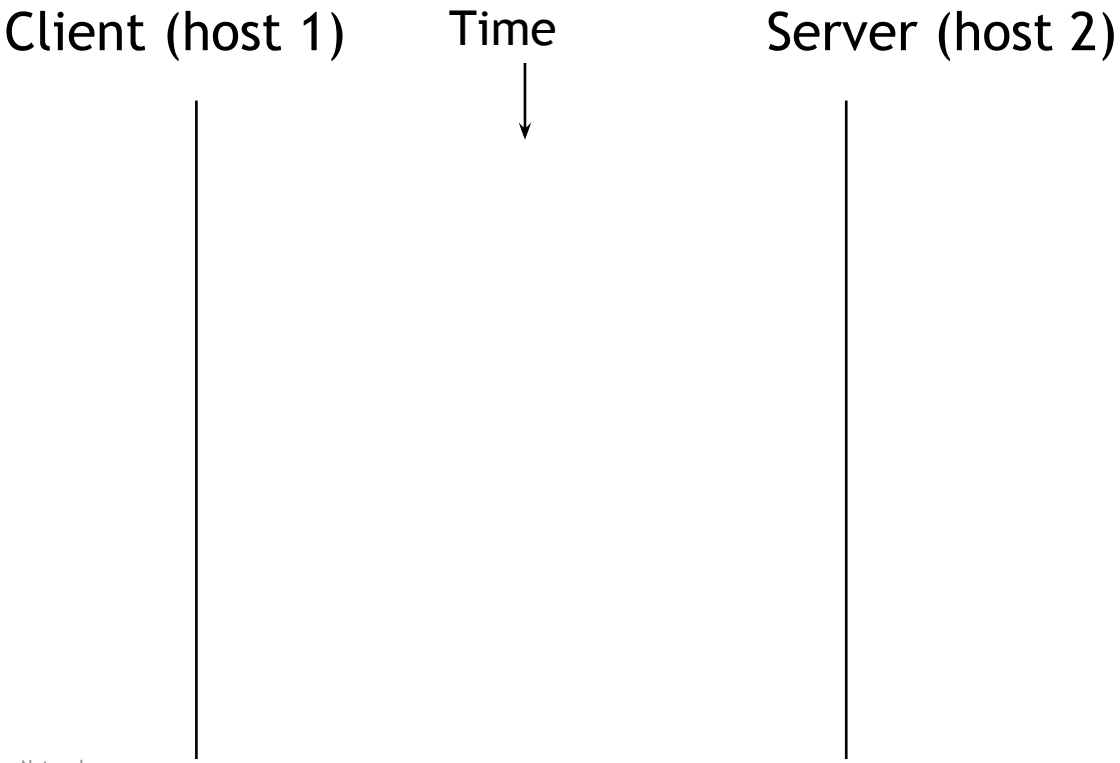


Socket API (3)

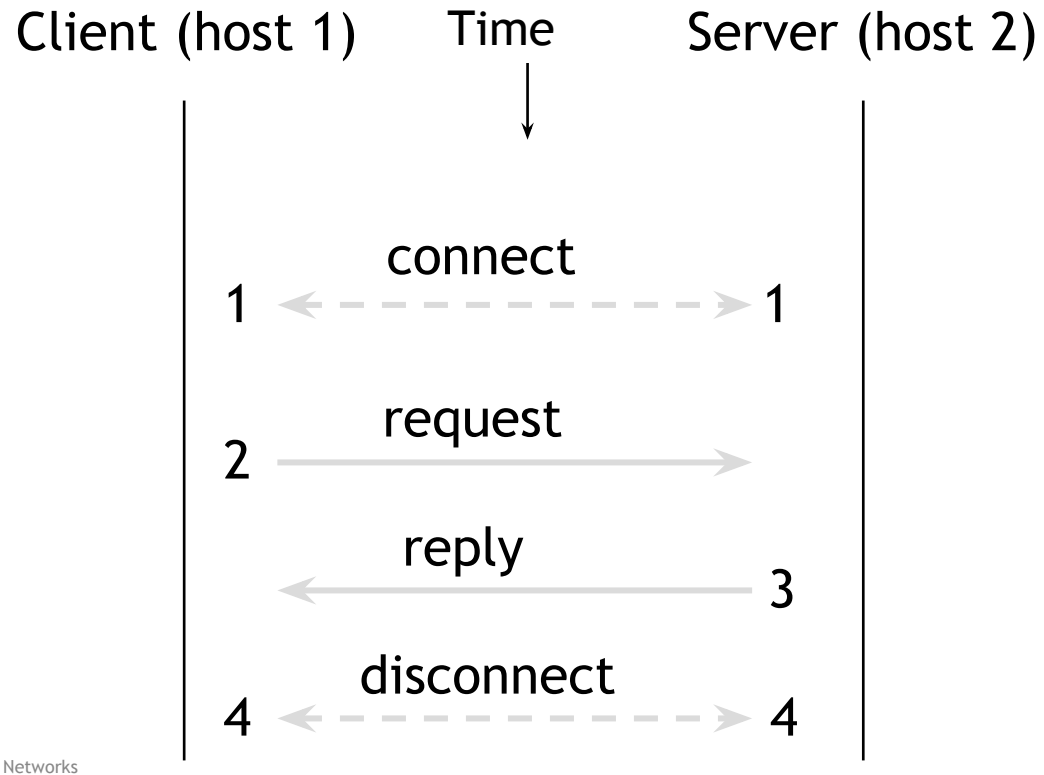
Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address (port) with a socket
LISTEN	Announce willingness to accept connections; (give queue size)
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data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

https://docs.oracle.com/javase/8/docs/api/java/net/Socket.html
https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html

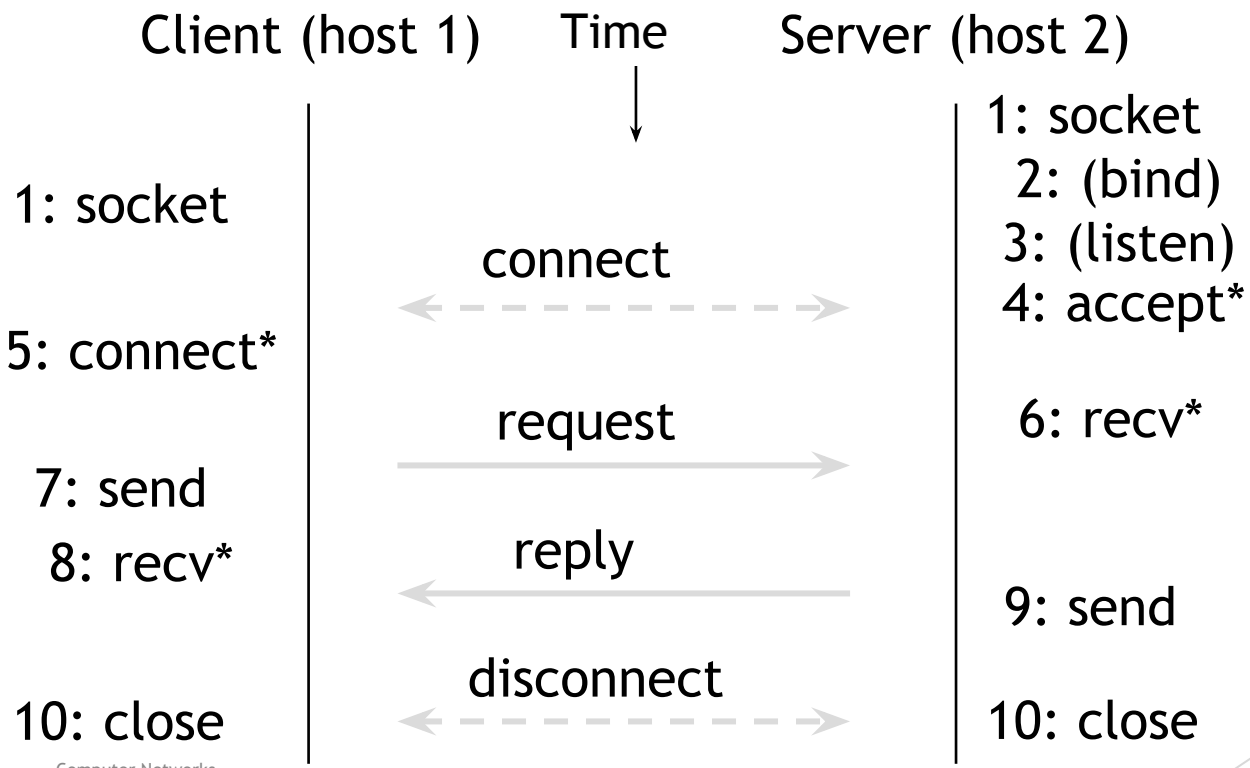
Using Sockets



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