

Computer Networks

The Socket API

(§1.3.4, 6.1.2-6.1.4)



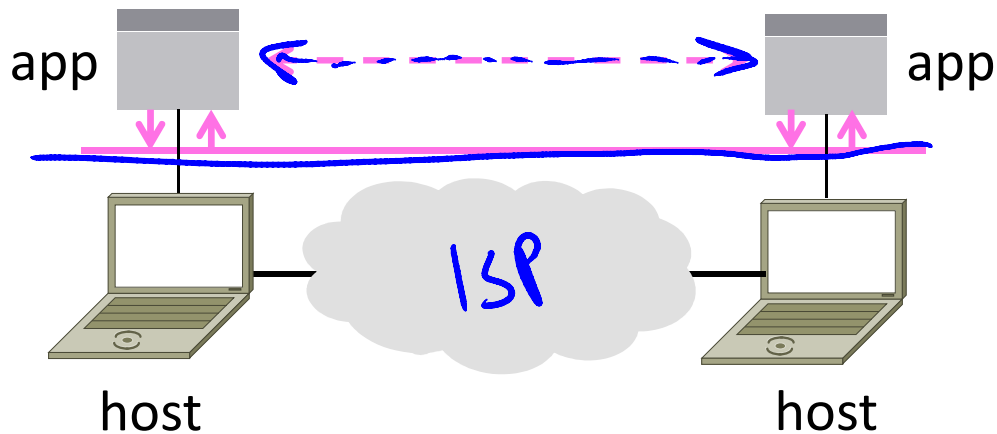
David Wetherall (djw@uw.edu)

Professor of Computer Science & Engineering

UNIVERSITY *of* WASHINGTON

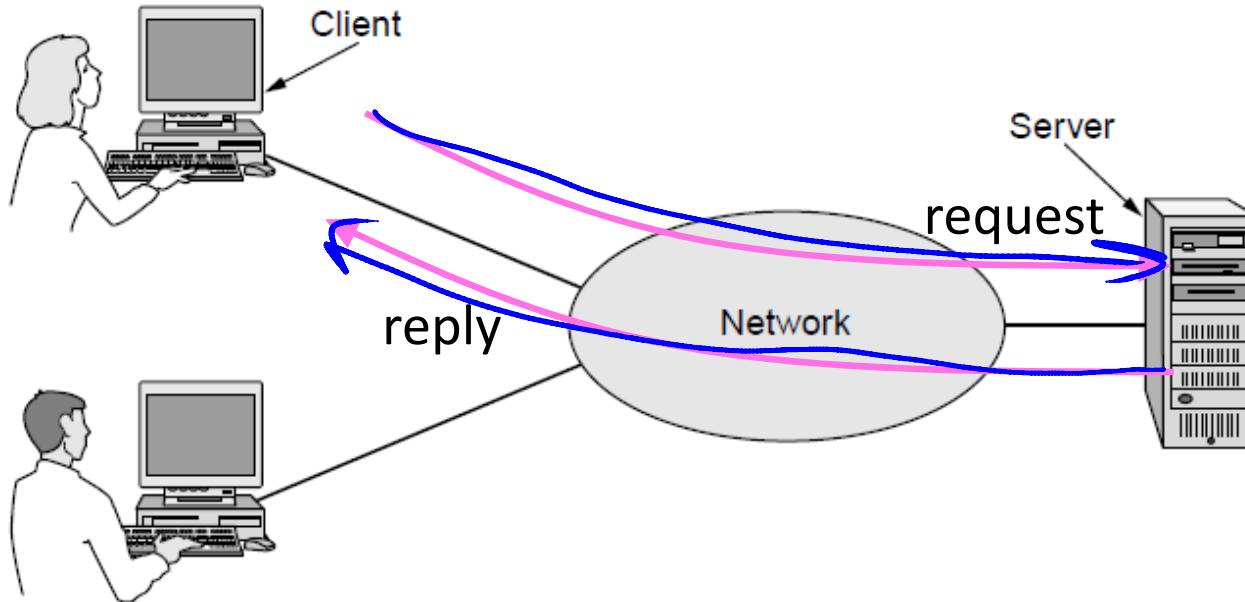
Network-Application Interface

- Defines how apps use the network
 - Lets apps talk to each other via hosts; hides the details of the network



Motivating Application

- Simple client-server setup



Motivating Application (2)

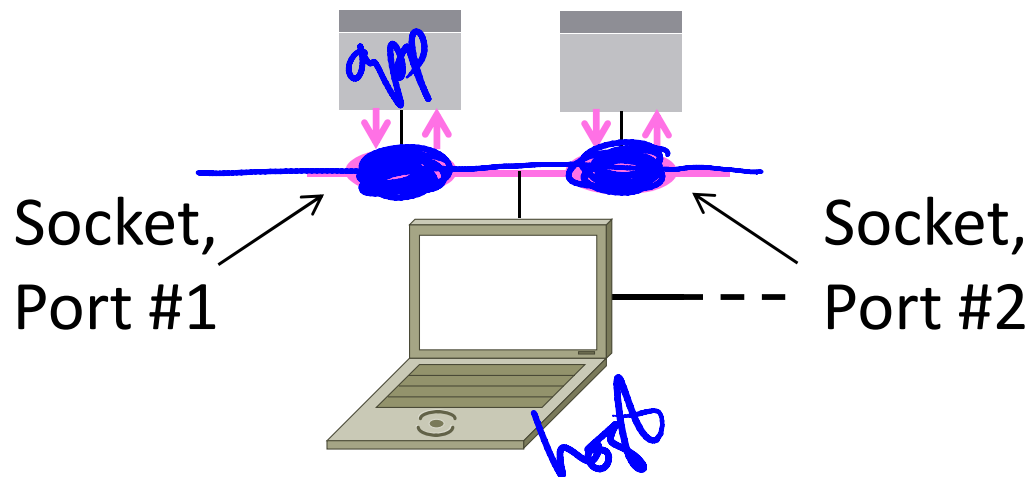
- Simple client-server setup
 - Client app sends a request to server app
 - Server app returns a (longer) reply
- This is the basis for many apps!
 - ➔ File transfer: send name, get file (§6.1.4)
 - ➔ Web browsing: send URL, get page
 - ➔ Echo: send message, get it back
- Let's see how to write this app ...

Socket API

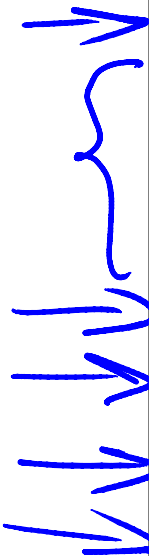
- Simple abstraction to use the network
 - The **network service API** used to write all Internet applications
 - Part of all major OSes and languages; originally Berkeley (Unix) ~1983
- Supports two kinds of network services
 - ➔ Streams: reliably send a stream of bytes »
 - Datagrams: unreliably send separate messages. (Ignore for now.)

Socket API (2)

- Sockets let apps attach to the local network at different ports



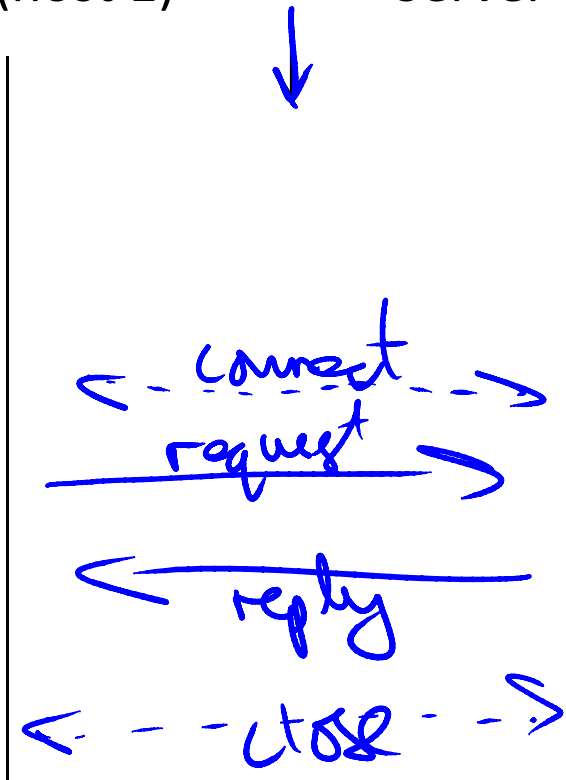
Socket API (3)



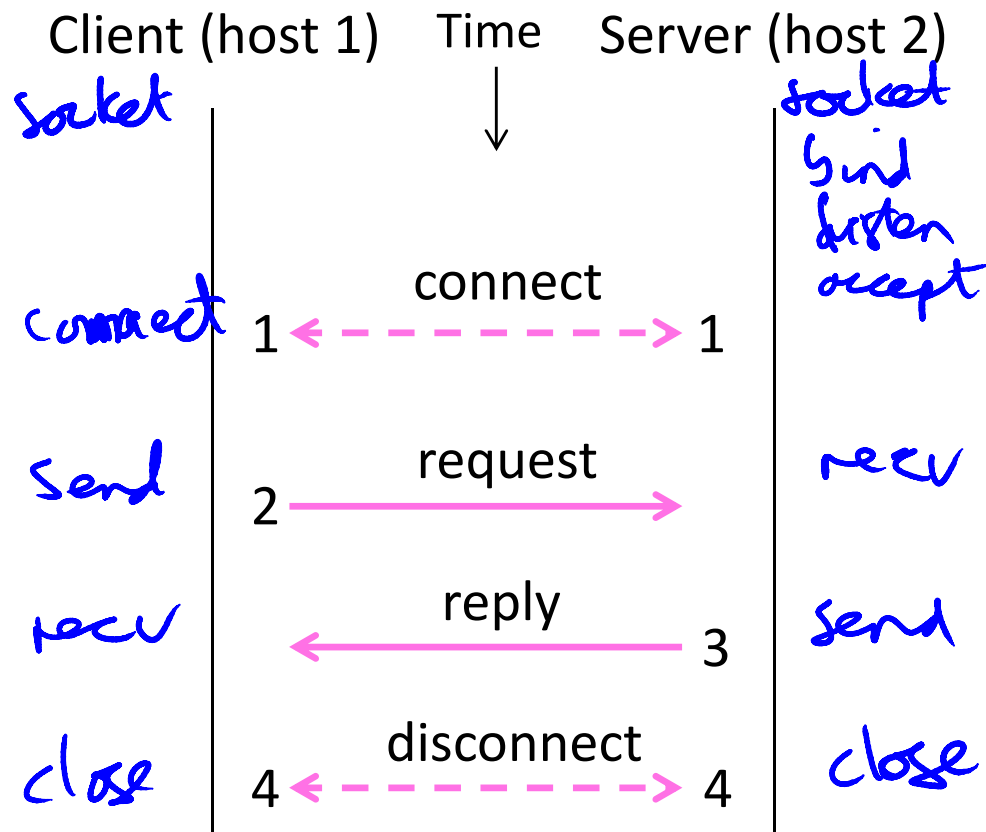
Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the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

Using Sockets

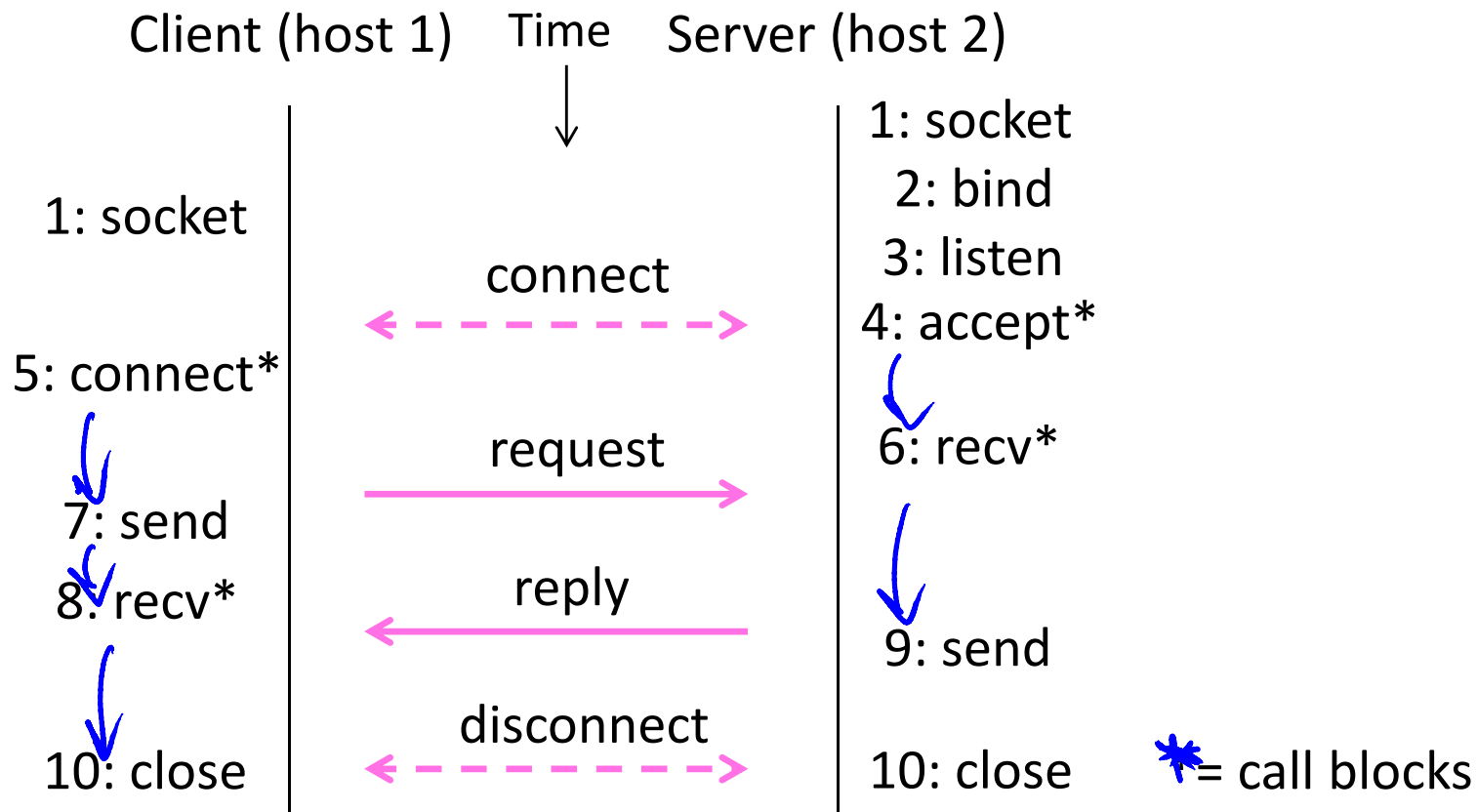
Client (host 1) Time Server (host 2)



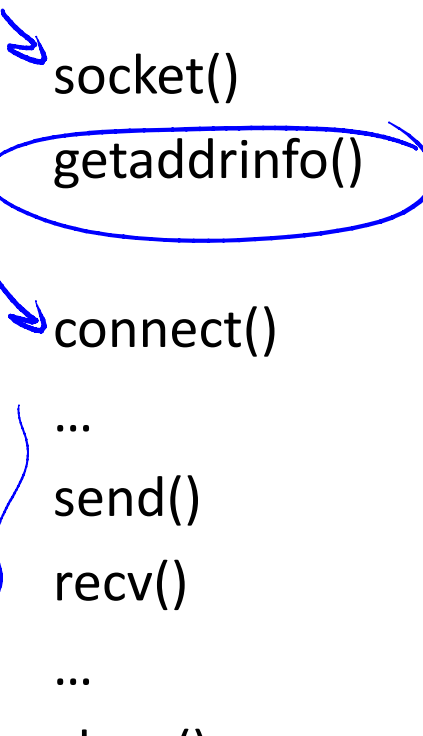
Using Sockets (2)



Using Sockets (3)

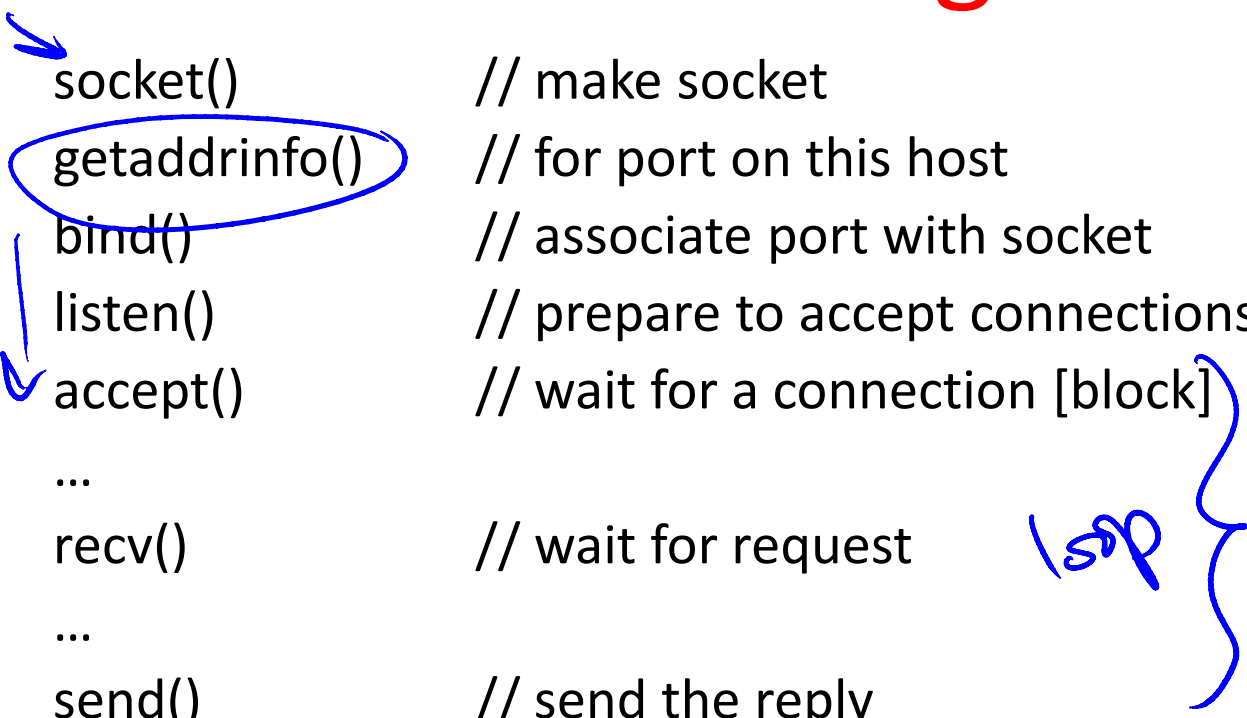


Client Program (outline)



```
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
```

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