CS5222 Computer Networks and Internets

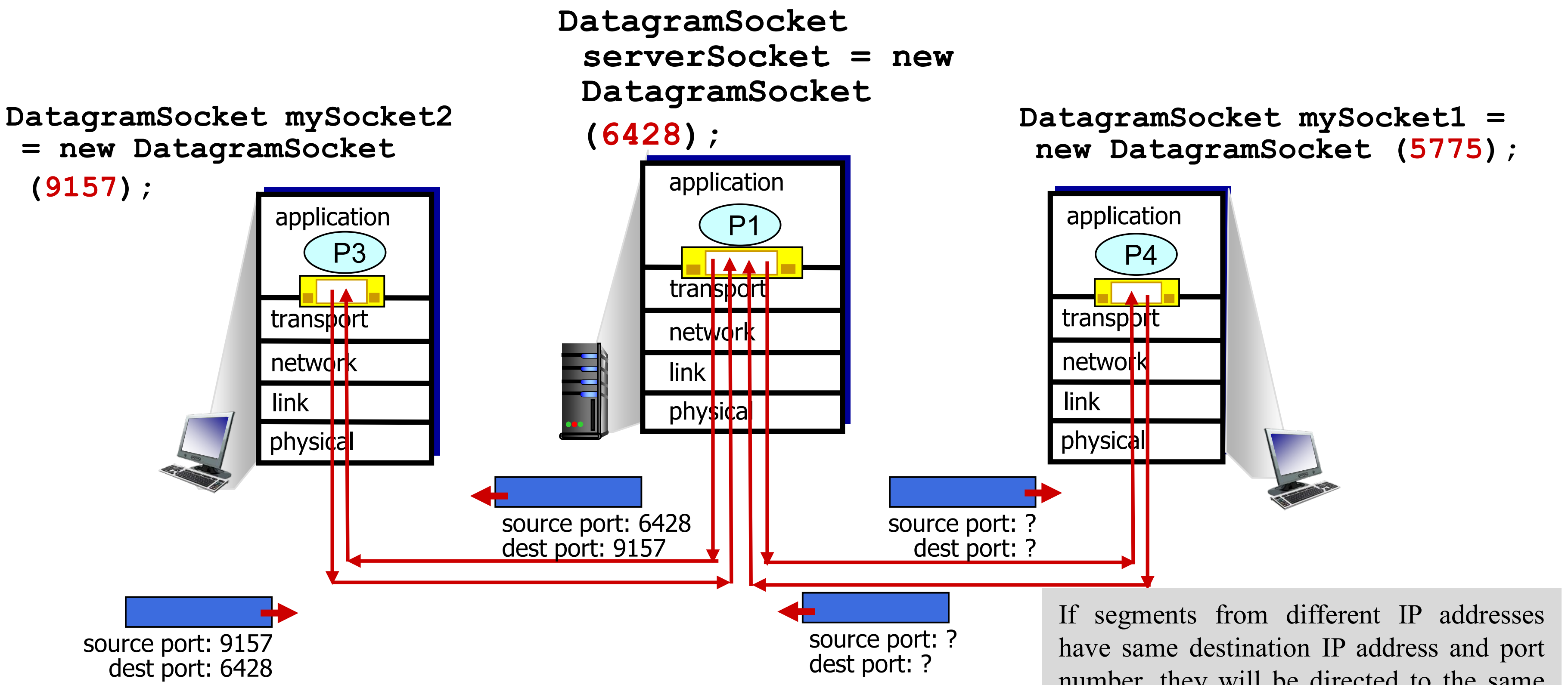
Tutorial on Socket Programming

Prof Weifa Liang

Weifa.liang@cityu.edu.hk

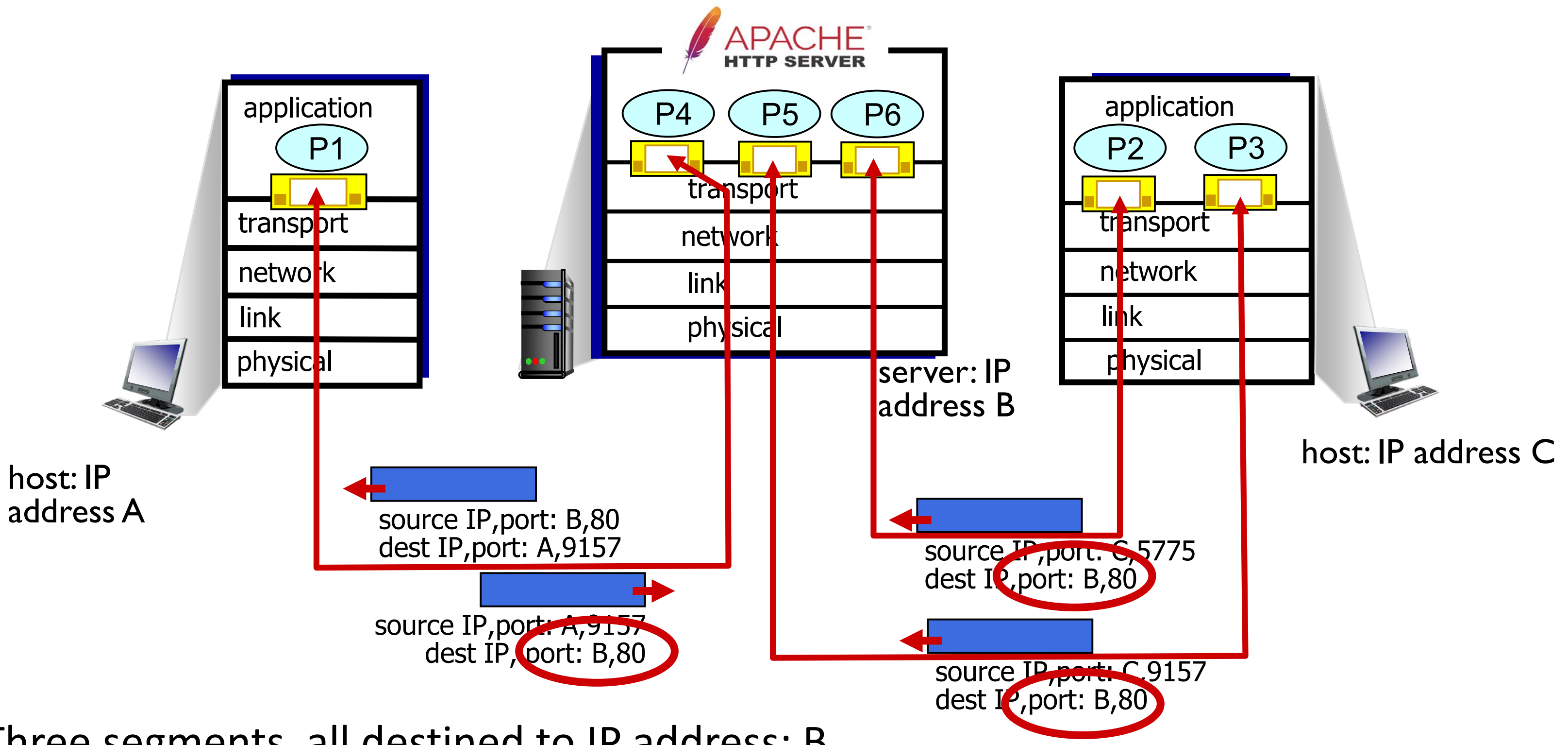
Slides based on book *Computer Networking: A Top-Down Approach.*

Connectionless demultiplexing: an example



If segments from different IP addresses have same destination IP address and port number, they will be directed to the same process

Connection-oriented demultiplexing: example

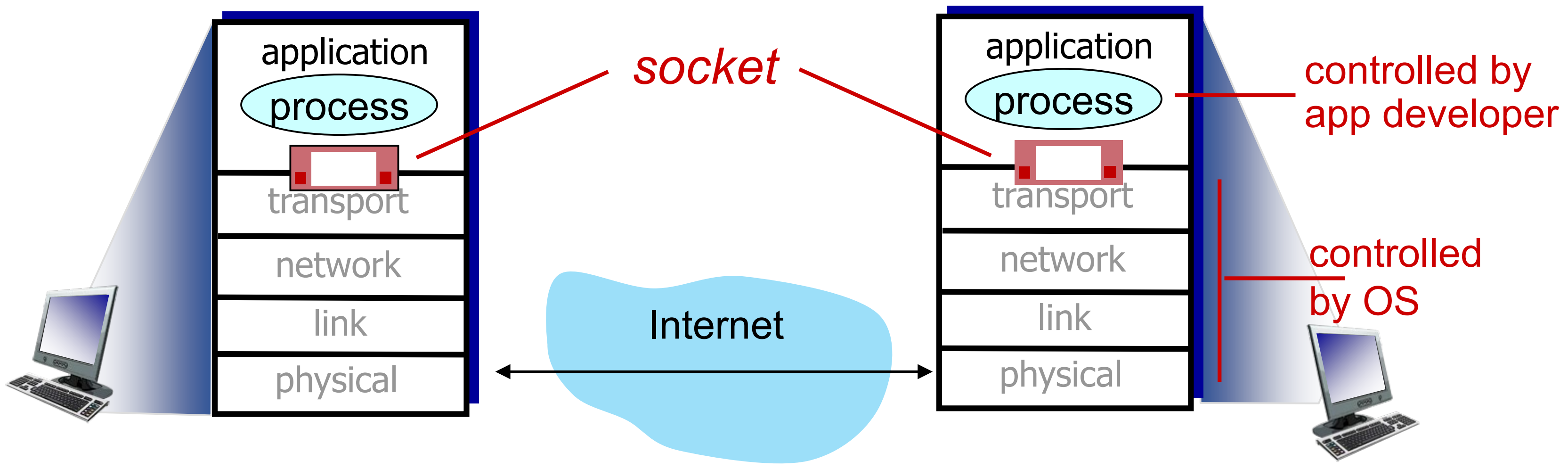


Three segments, all destined to IP address: B,
dest port: 80 are demultiplexed to *different* sockets

Introduction to Socket programming

goal: learn how to build client/server applications that communicate using sockets

socket: door between the application layer protocol and the end-to-end transport layer protocol



Socket programming

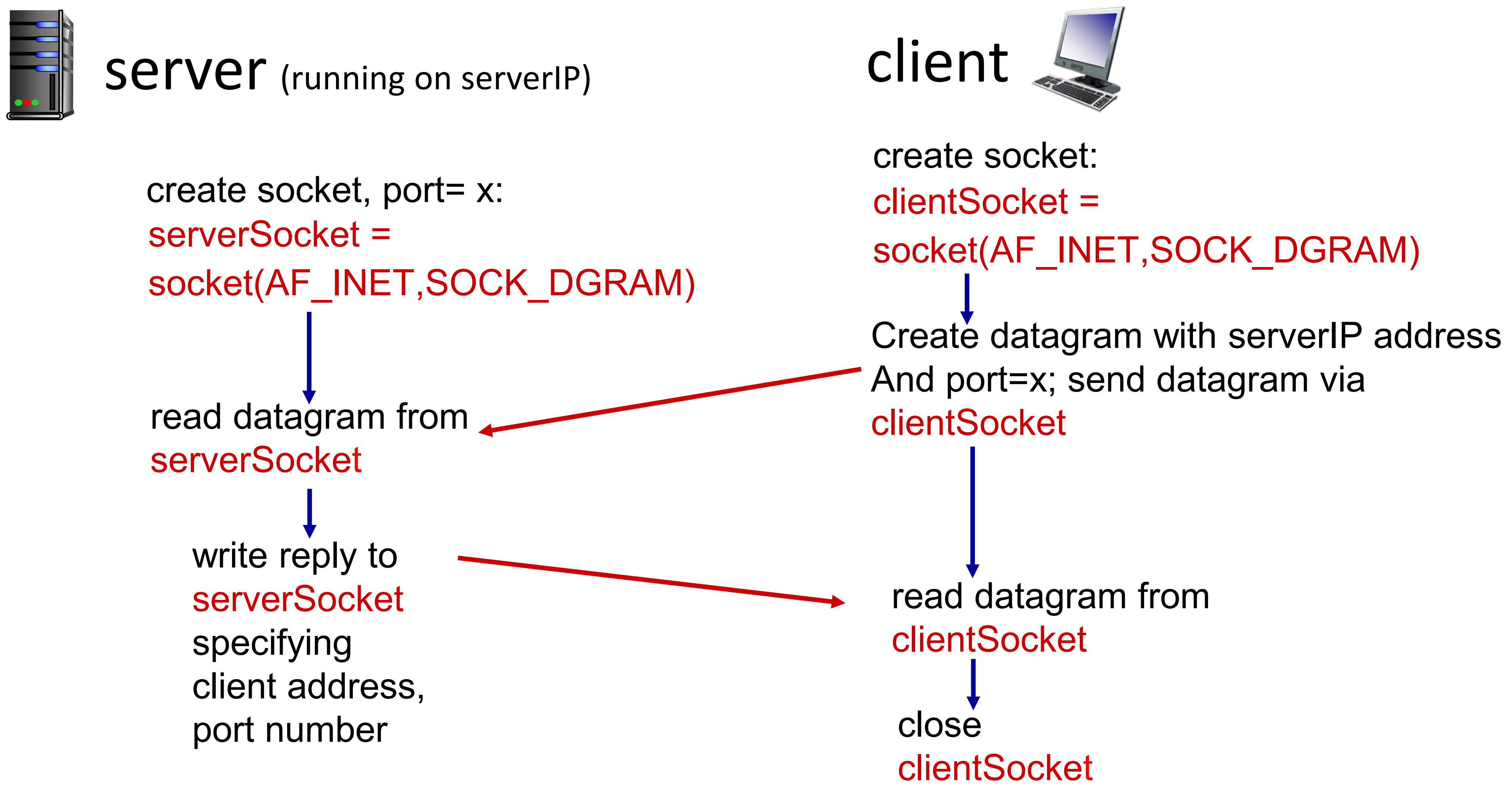
Two socket types for two transport services:

- *UDP:* unreliable datagram
- *TCP:* reliable, byte stream-oriented

An Example:

1. Client reads a line of characters (data) from its keyboard and sends the data to a server
2. Server receives the data and converts the characters (data) into uppercase
3. Server sends the modified data back to the client
4. Client receives the modified data and displays the line on its screen

Client/server socket interaction: UDP



Example app: UDP client

Python UDPClient

include Python's socket library →	`from socket import *`
	`serverName = 'hostname'`
	`serverPort = 12000`
create UDP socket for server →	`clientSocket = socket(AF_INET, SOCK_DGRAM)`
get user keyboard input →	`message = raw_input('Input lowercase sentence:')`
attach server name, port to message; send into socket →	`clientSocket.sendto(message.encode(), (serverName, serverPort))`
read reply characters from socket into string →	`modifiedMessage, serverAddress = clientSocket.recvfrom(2048)`
print out received string and close socket →	`print modifiedMessage.decode()`
	`clientSocket.close()`

Example app: UDP server

Python UDPServer

```
                                                  from socket import *
                                                  serverPort = 12000
                          create UDP socket  →   serverSocket = socket(AF_INET, SOCK_DGRAM)
        bind socket to local port number 12000  →   serverSocket.bind((", serverPort))
                                                  print (“The server is ready to receive”)
                               loop forever  →   while True:
Read from UDP socket into message, getting  →       message, clientAddress = serverSocket.recvfrom(2048)
client’s address (client IP and port)                modifiedMessage = message.decode().upper()
     send upper case string back to this client  →       serverSocket.sendto(modifiedMessage.encode(),
                                                                                  clientAddress)
```

Socket programming with TCP

Client must contact server

- server process must first be running
- server must have a created socket (door) that welcomes client's contact

Client contacts server by:

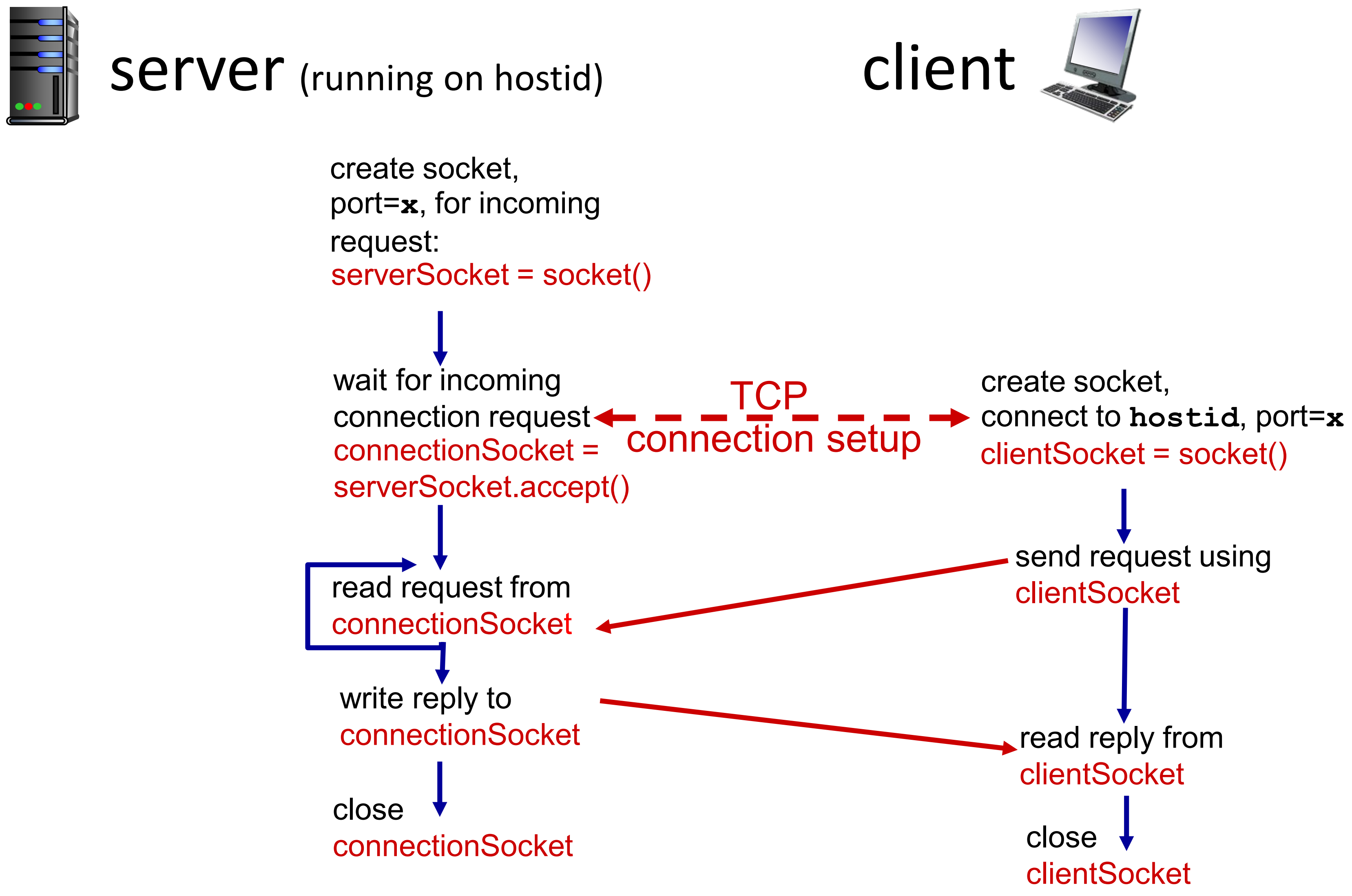
- Creating a TCP socket, specifying IP address, port number of server
- *when client creates socket:* client TCP establishes a TCP connection to server TCP

- when contacted by a client, *server TCP creates new socket* for server process to communicate with that particular client
  - allows server to talk with multiple clients
  - *source* port number used to distinguish different clients

Application viewpoint

TCP provides reliable, in-order byte-stream transfer ("pipe") between client and server processes

Client/server socket interaction: TCP



Example app: TCP client

Python TCPClient

include Python's socket library → from socket import *

create TCP socket for server, remote port 12000 → clientSocket = socket(AF_INET, SOCK_STREAM)

get user keyboard input → sentence = raw_input('Input lowercase sentence:')

send into socket → clientSocket.send(sentence.encode())

read reply characters from socket into string → modifiedSentence = clientSocket.recv(1024)

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print ('From Server: ', modifiedSentence.decode())
clientSocket.close()
```

Example app: TCP server

Python TCPServer

create TCP welcoming socket

server begins listening for incoming TCP requests

loop forever

server waits on accept() for incoming requests, new socket created on return

read bytes from socket (but not address as in UDP)

close connection to this client (but *not* welcoming socket)

```
 from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((‘’,serverPort))
serverSocket.listen(1)
print ‘The server is ready to receive’
while True:
    connectionSocket, addr = serverSocket.accept()

    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.
                                           encode())
    connectionSocket.close()
```