

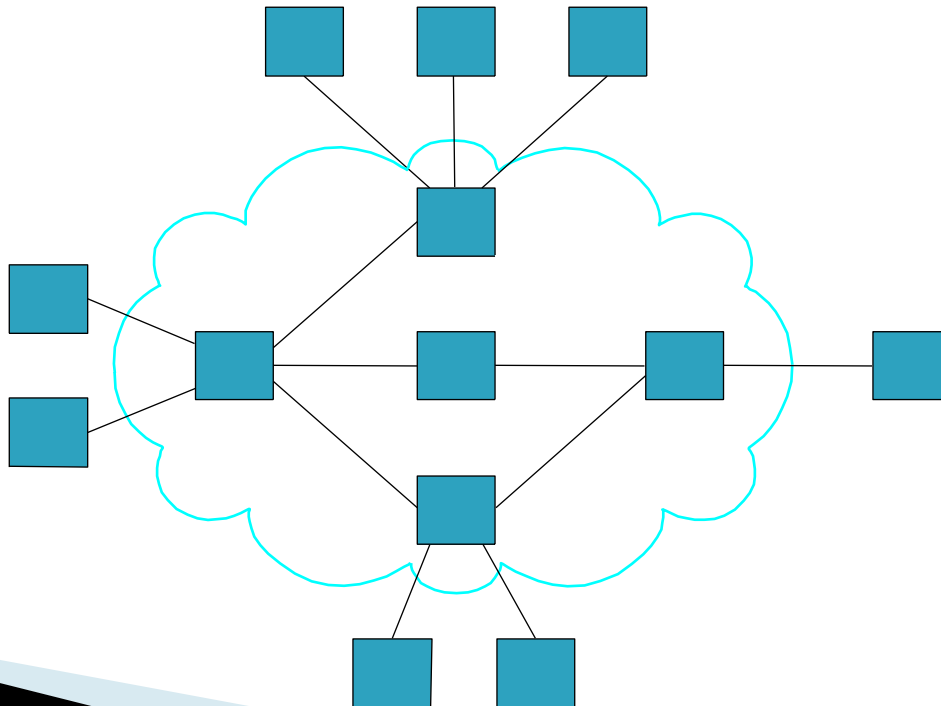
JAVA Socket Programming

Edited by Ehsan Edalat



Computer Network

A **computer network** is an interconnected collection of autonomous computers.



Protocol

A **network protocol** defines rules and conventions for communication between **network** devices.

Network protocols include mechanisms for devices to **identify** and make connections with each other, as well as **formatting rules** that specify how data is packaged into sent and received messages.

Network Architecture

A network architecture is a set of **layers** and **protocols** used to reduce network design complexity.

The TCP/IP Protocol Suite (also called the Internet Architecture) is an important example of a network architecture.

TCP/IP Protocol Suite

Application

- Various applications (FTP,HTTP,...)

Transport

- Reliable, end-to-end byte stream (TCP)

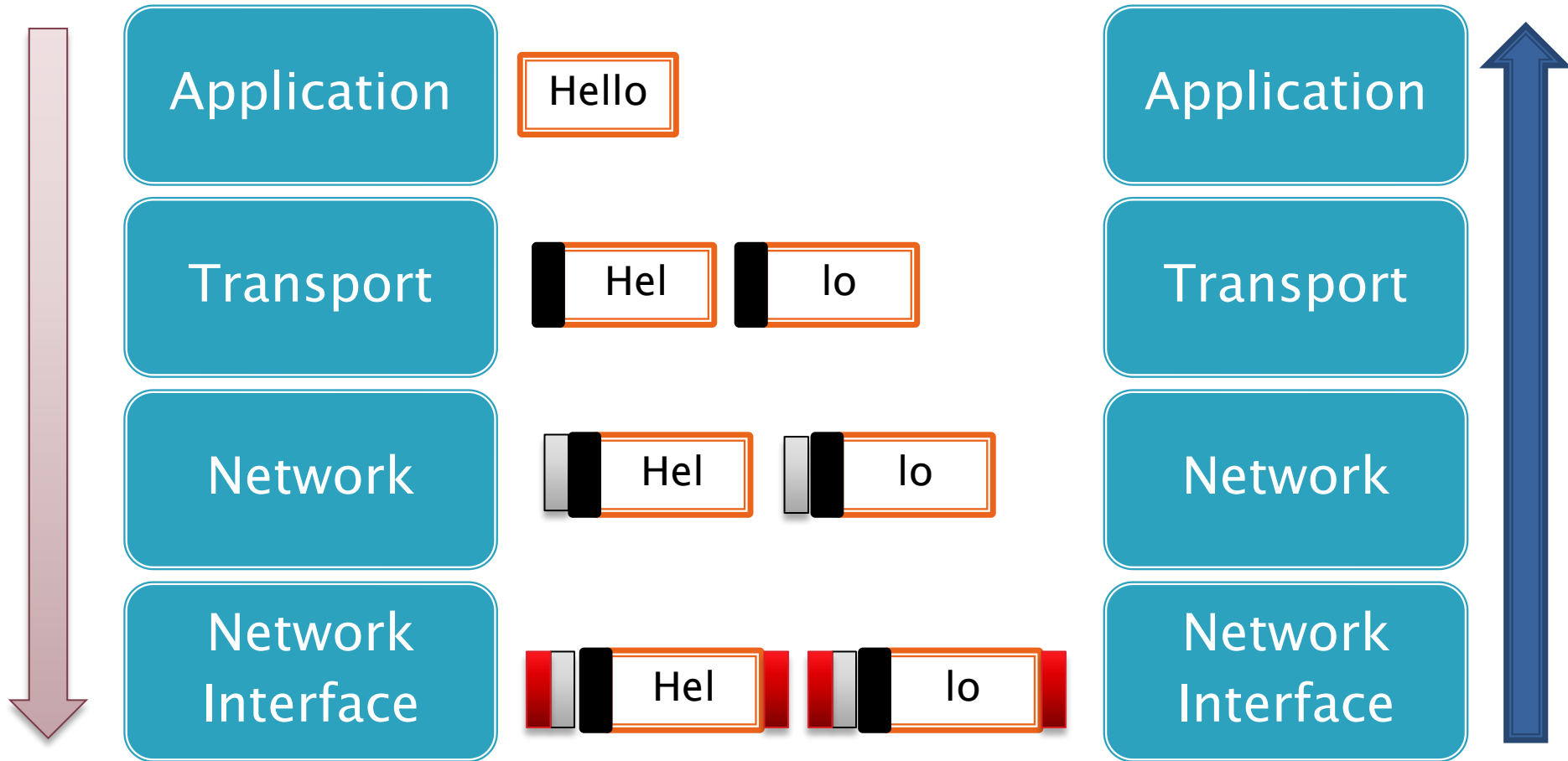
Network

- Unreliable end-to-end transmission of packets

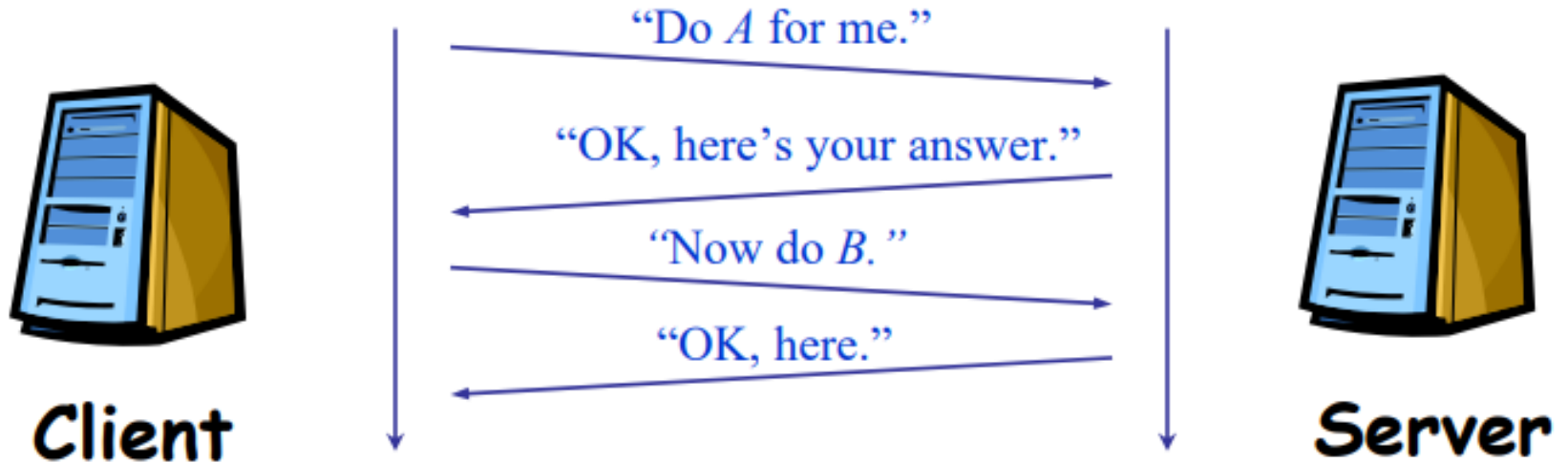
Network Interface

- transmission of raw bits

TCP/IP Protocol Suite

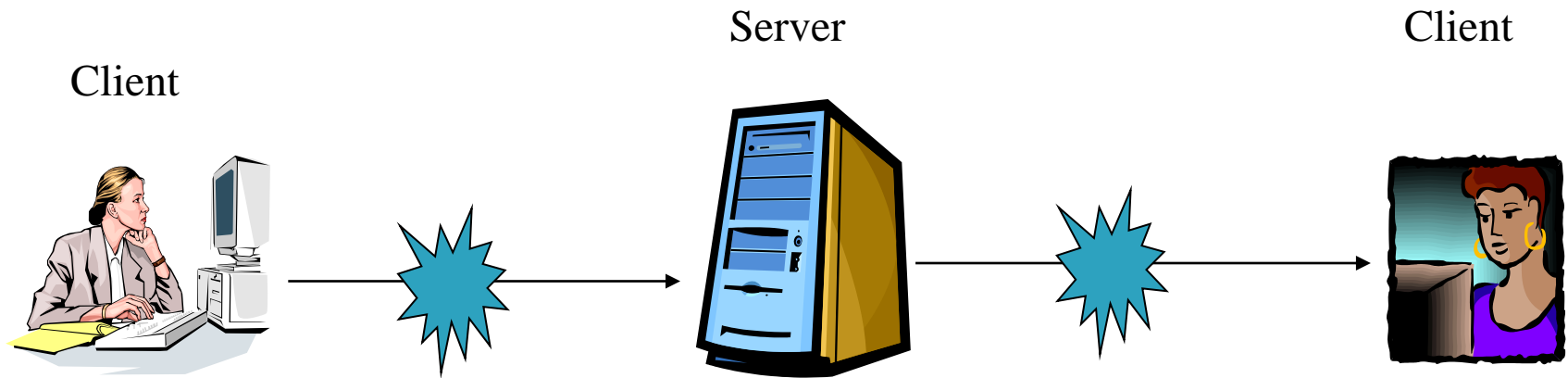


Server and Client



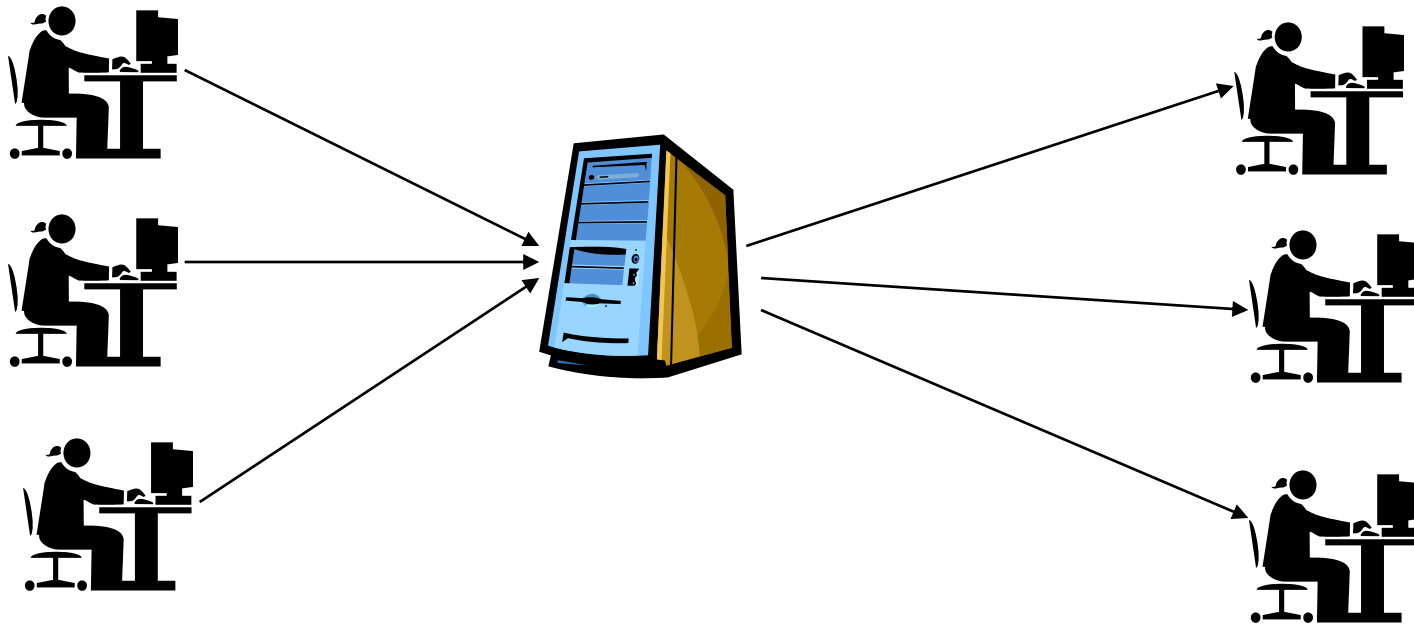
Server and Client

► Email

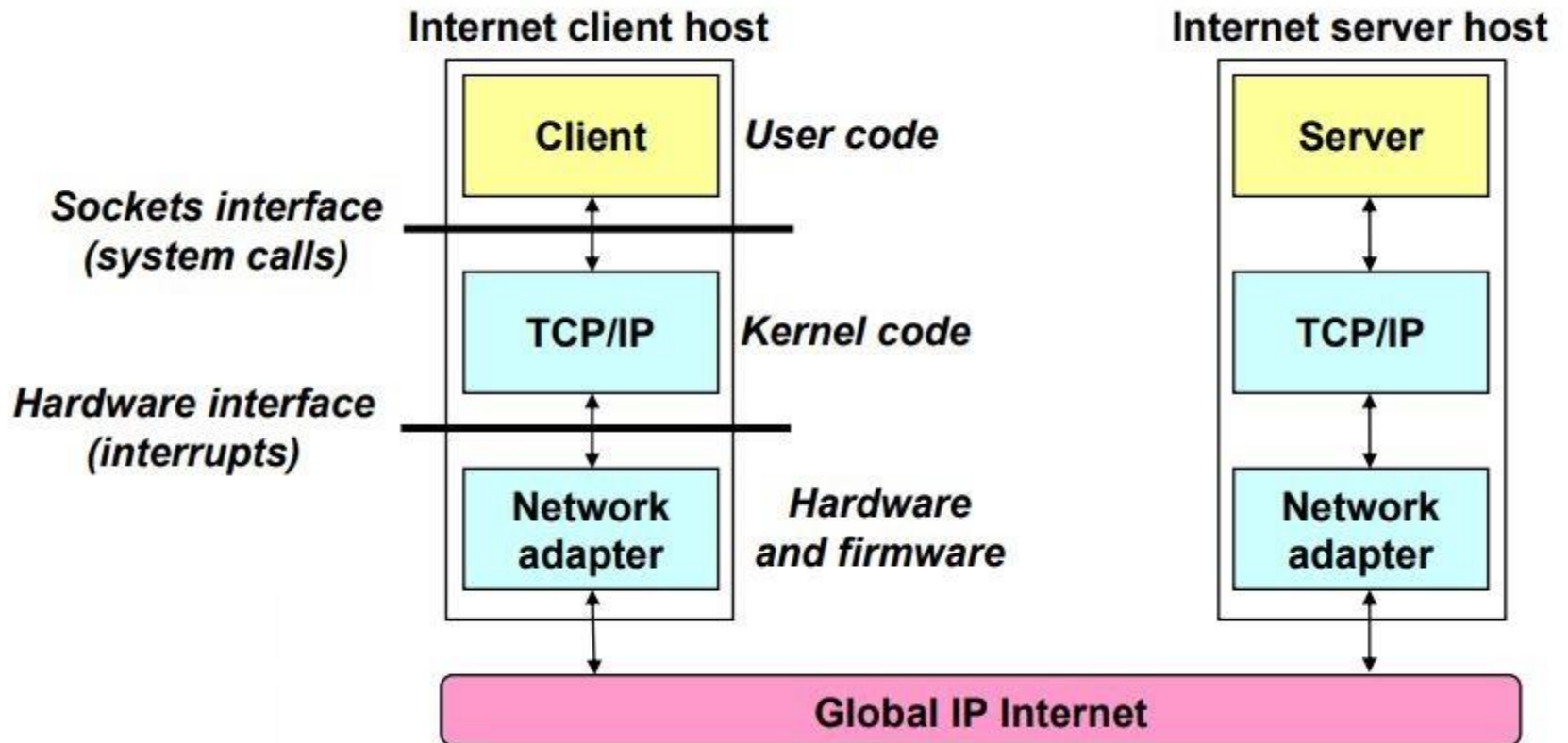


Server and Client

► Chatroom



An Internet Application



A Programmer's View of the Internet

- ▶ Hosts are mapped to a set of 32-bit **IP addresses**.
 - 185.211.88.129
- ▶ The set of IP addresses is mapped to a set of identifiers called **Internet domain names**.
 - 185.211.88.129 is mapped to ce.aut.ac.ir
- ▶ A process on one Internet host can communicate with a process on another Internet host over a **connection**.

Internet Connections

- ▶ Most clients and servers communicate by sending streams of bytes over **connections**
 - e.g., using TCP, the Transmission Control Protocol
- ▶ A **socket** is an endpoint of a connection between two processes
 - Java APIs
- ▶ Or: the interface between user and network

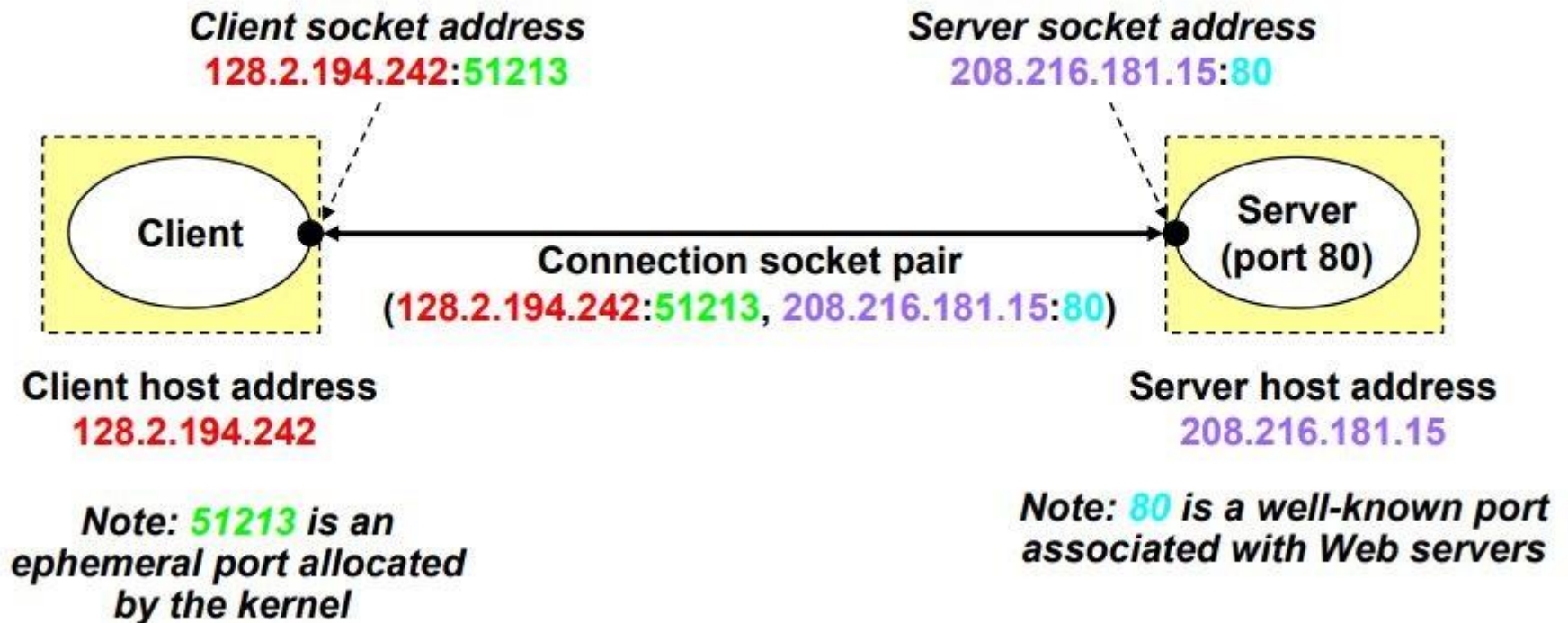
Sockets



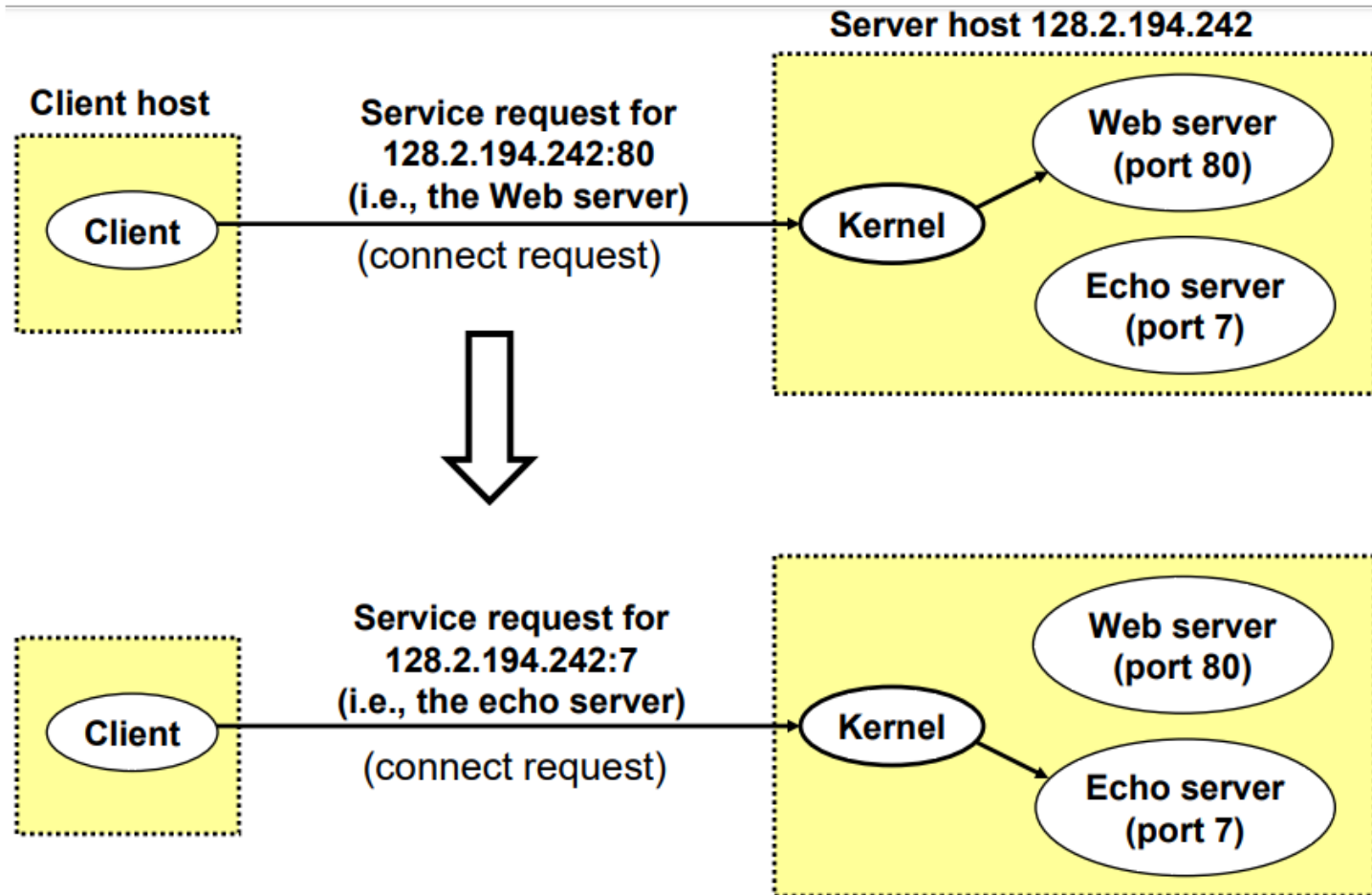
Sockets

- ▶ A host might have many open connections, possibly held by different processes.
- ▶ A **port** is a unique communication endpoint on a host, named by a 16-bit integer, and associated with a process.

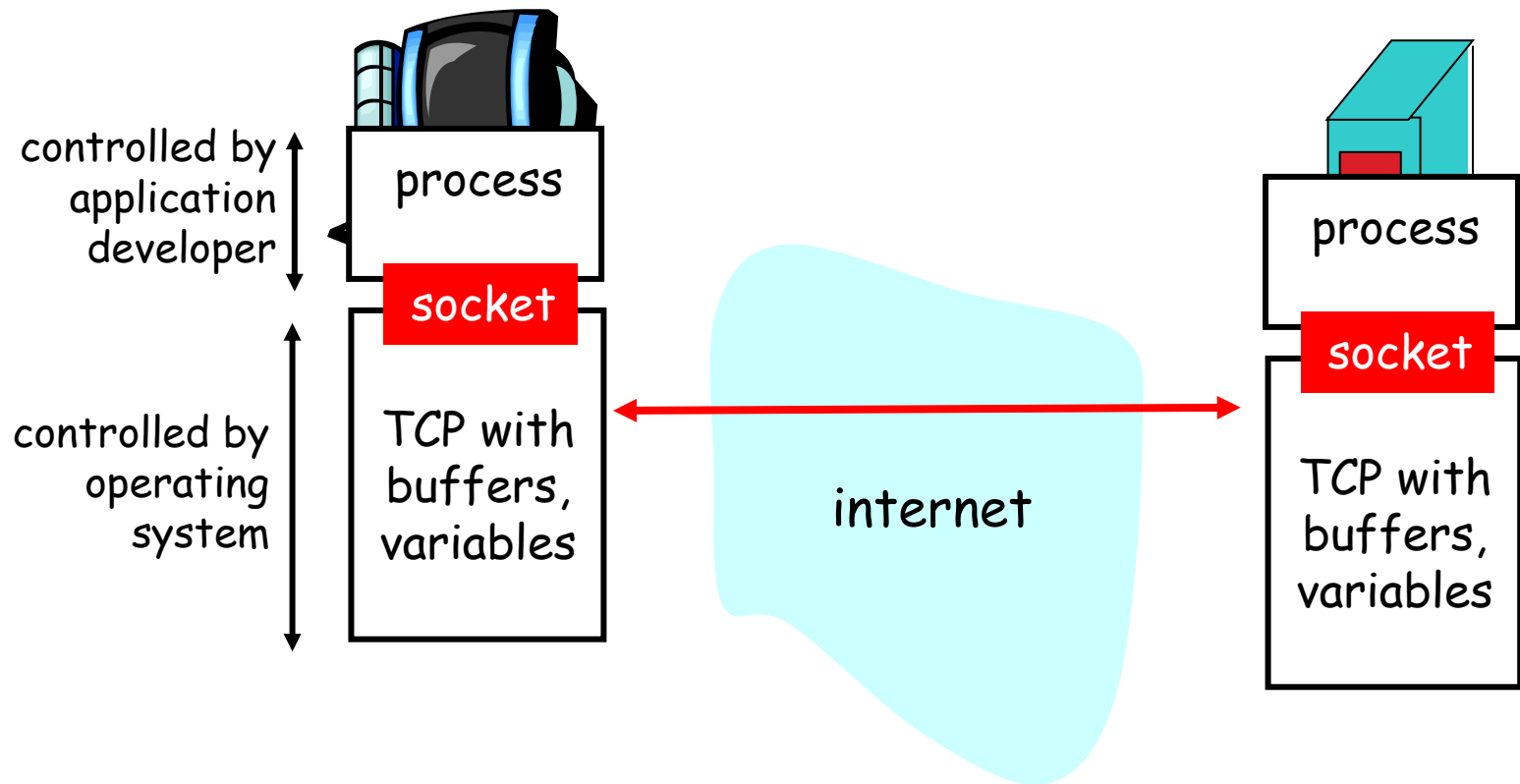
Sockets



Sockets



Sockets



What is a socket?

▶ Socket

- The combination of an IP address and a port number.
- Two types
 - Stream socket : reliable two-way connected communication streams (TCP)
 - Datagram socket (UDP)

▶ Socket pair

- Specified the two end points that uniquely identifies each TCP connection in an internet.
- 4-tuple: (client IP address, client port number, server IP address, server port number)

TCP Sockets for server and client

▶ Server

■ Welcoming socket

- Welcomes some initial contact from a client.

■ Connection socket

- Is created at initial contact of client.
- New socket that is dedicated to the particular client.

▶ Client

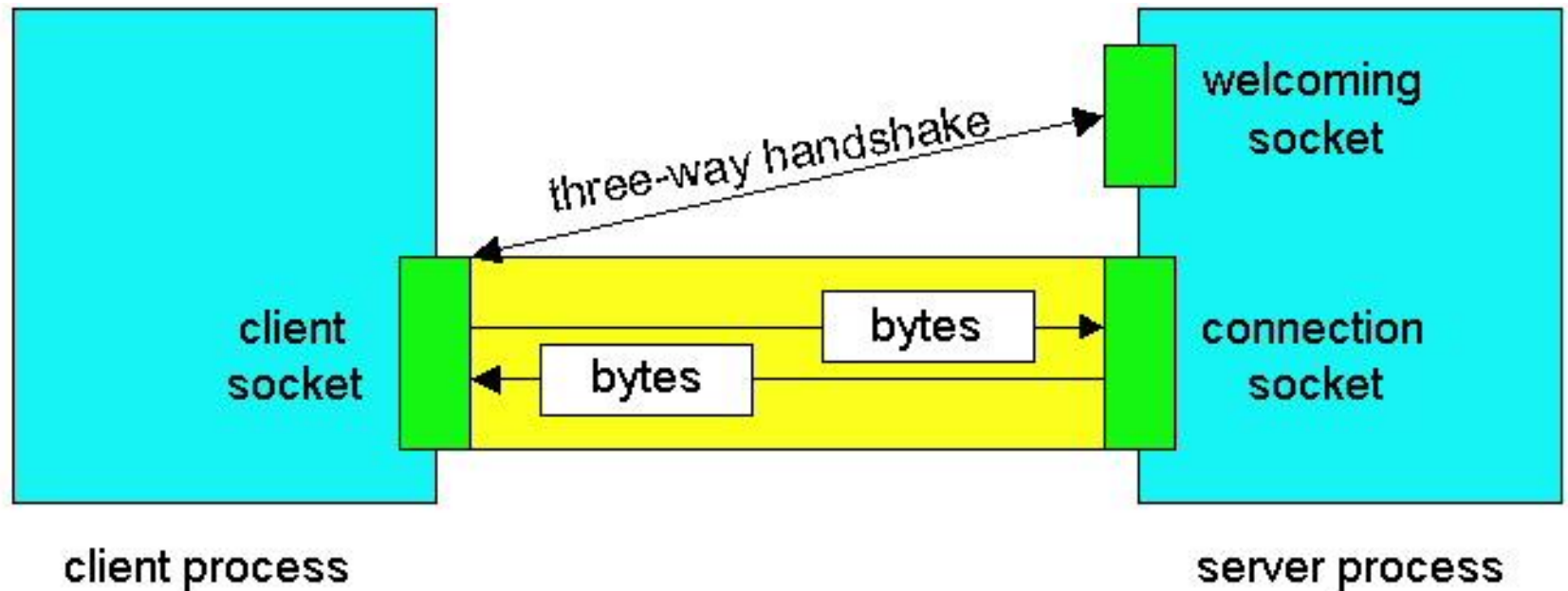
■ Client socket

- Initiate a TCP connection to the server by creating a socket object. (Three-way handshake)
- Specify the address of the server process, namely, the IP address of the server and the port number of the process.

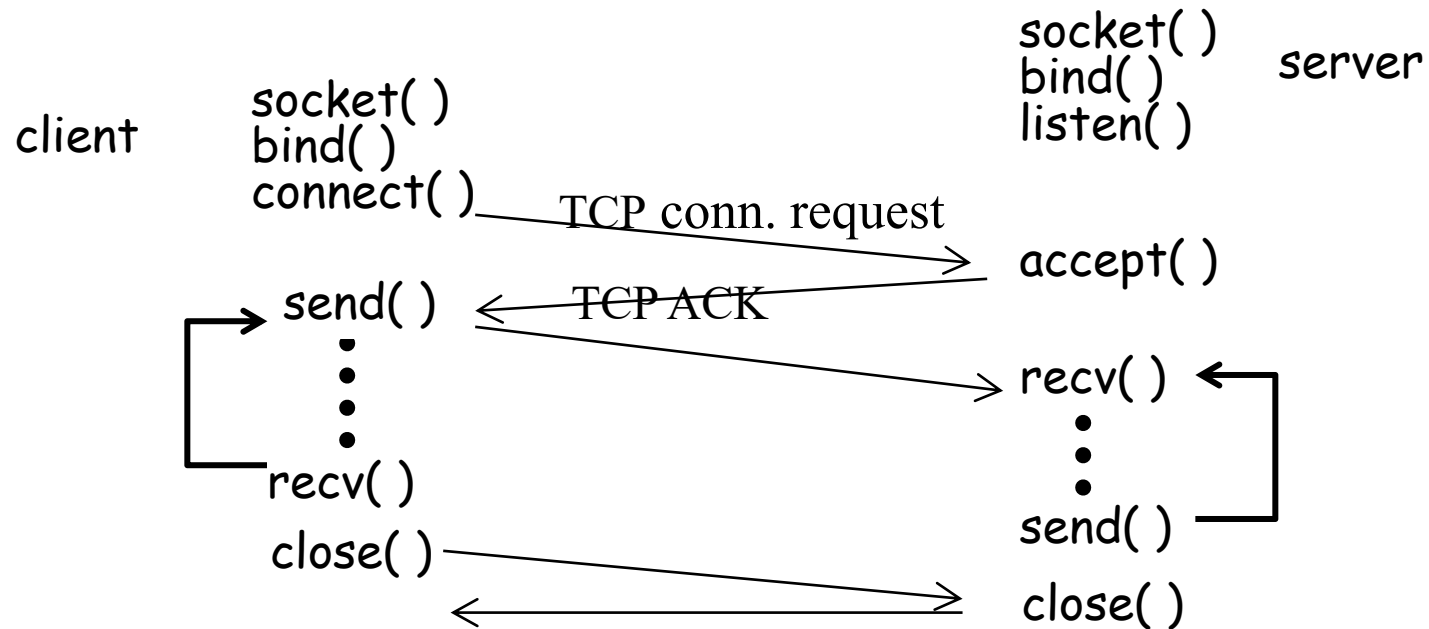
Socket functional calls

- ▶ **socket ()**: Create a socket
- ▶ **bind()**: bind a socket to a local IP address and port #
- ▶ **listen()**: passively waiting for connections
- ▶ **connect()**: initiating connection to another socket
- ▶ **accept()**: accept a new connection
- ▶ **Write()**: write data to a socket
- ▶ **Read()**: read data from a socket
- ▶ **sendto()**: send a datagram to another UDP socket
- ▶ **recvfrom()**: read a datagram from a UDP socket
- ▶ **close()**: close a socket (tear down the connection)

TCP Sockets



Socket-programming using TCP



Client/server socket interaction: TCP

Server (running on **hostid**)

Client

```
create socket,  
port=x, for  
incoming request:  
welcomeSocket =  
ServerSocket()
```



Client/server socket interaction: TCP

Server (running on **hostid**)

Client

```
create socket,  
port=x, for  
incoming request:  
welcomeSocket =  
    ServerSocket()  
    ↓  
wait for incoming  
connection request  
connectionSocket =  
welcomeSocket.accept()
```


Client/server socket interaction: TCP

Server (running on **hostid**)

```
create socket,  
port=x, for  
incoming request:  
welcomeSocket =  
    ServerSocket()
```



```
wait for incoming  
connection request  
connectionSocket =  
welcomeSocket.accept()
```

Client

```
create socket,  
connect to hostid, port=x  
clientSocket =  
    Socket()
```

Client/server socket interaction: TCP

Server (running on **hostid**)

Client

create socket,
port=**x**, for
incoming request:
welcomeSocket =
 ServerSocket()

↓
wait for incoming
connection request
connectionSocket =
 welcomeSocket.accept()

← — — — — — TCP — — — — — →
 connection setup

create socket,
connect to **hostid**, port=**x**
clientSocket =
 Socket()

Client/server socket interaction: TCP

Server (running on **hostid**)

Client

create socket,
port=**x**, for
incoming request:
welcomeSocket =
 ServerSocket()

wait for incoming
connection request
connectionSocket =
 welcomeSocket.accept()

← — — — — TCP — — — — →
 connection setup

create socket,
connect to **hostid**, port=**x**
clientSocket =
 Socket()

send request using
clientSocket

Client/server socket interaction: TCP

Server (running on **hostid**)

Client

create socket,
port=**x**, for
incoming request:
welcomeSocket =
ServerSocket()

wait for incoming
connection request
connectionSocket =
welcomeSocket.accept()

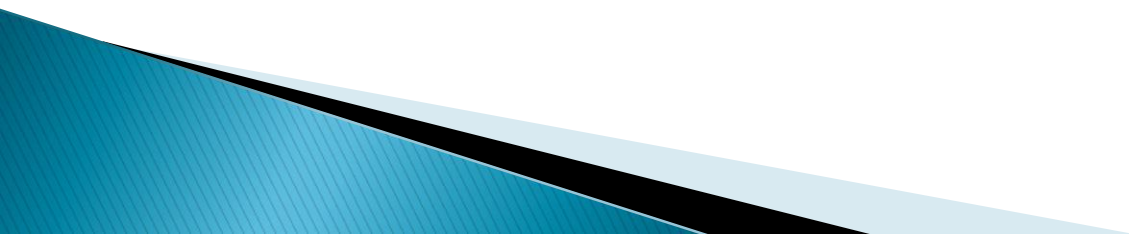
read request from
connectionSocket

write reply to
connectionSocket

TCP
connection setup

create socket,
connect to **hostid**, port=**x**
clientSocket =
Socket()

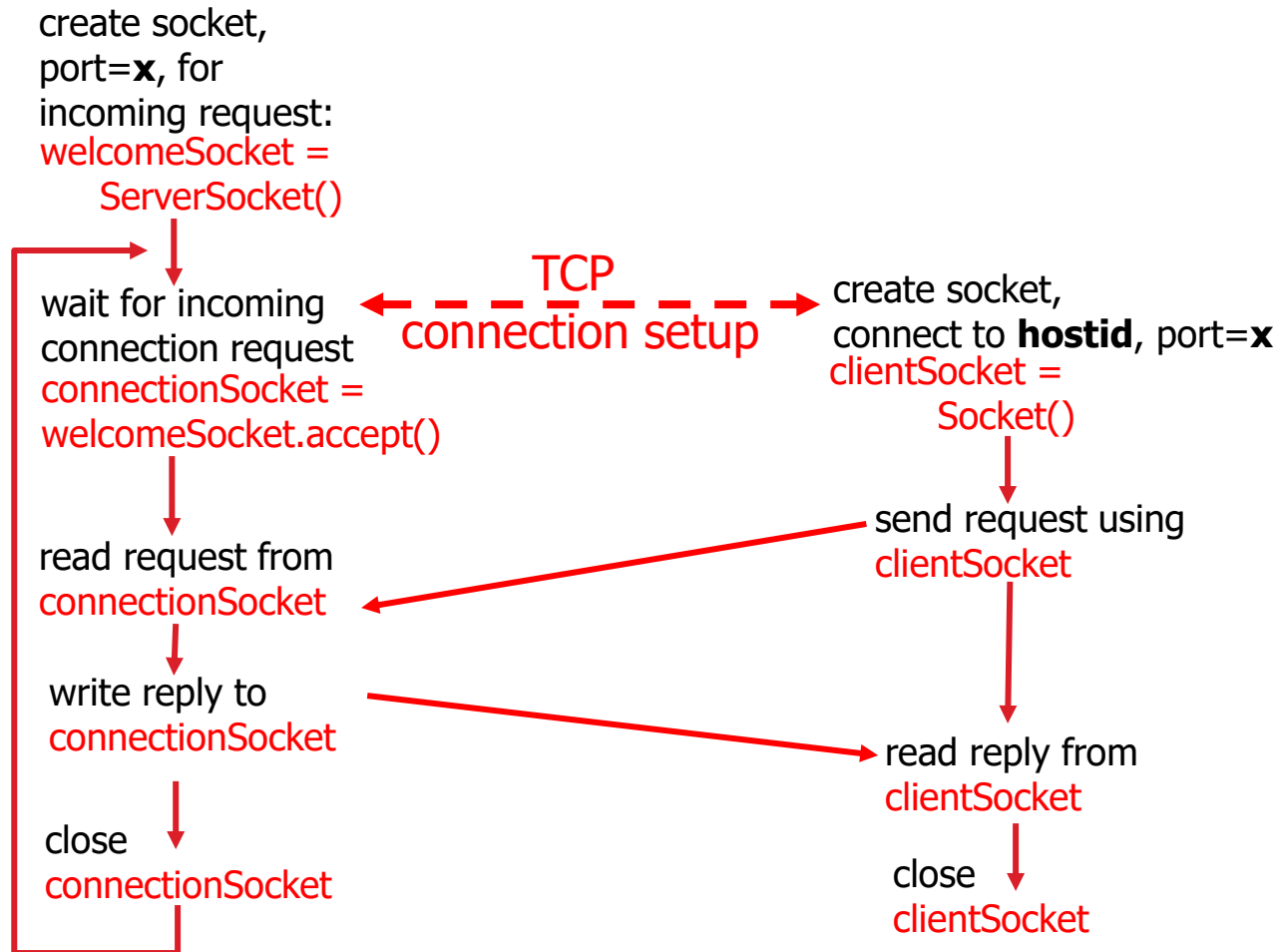
send request using
clientSocket



Client/server socket interaction: TCP

Server (running on **hostid**)

Client



Socket-programming using TCP

- ▶ See single-thread-server Example.

Concurrent server

- ▶ Servers need to handle a new connection request while processing previous requests.
 - Most TCP servers are designed to be concurrent.
- ▶ When a new connection request arrives at a server, the server accepts and invokes a new process to handle the new client.

Socket-programming using TCP

- ▶ See multi-thread-server Example.

Writing a HTTP Client with HttpURLConnection

- ▶ See HttpDownloader Example.