Introduction

Lecture 1

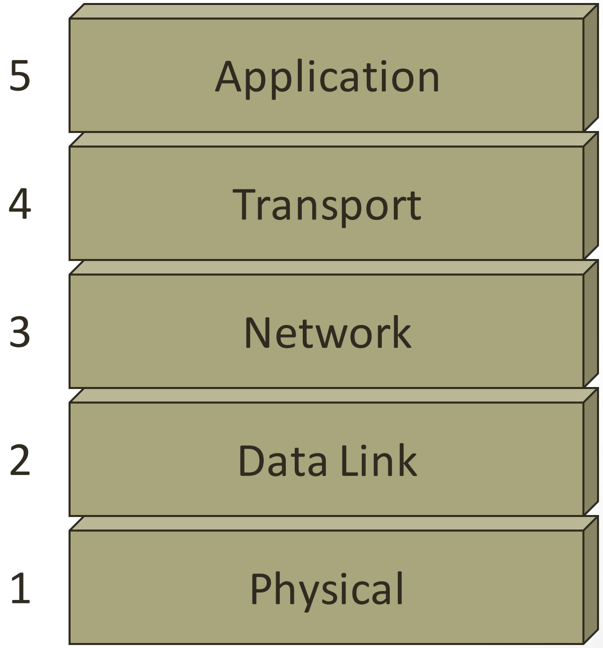
Topic Outline

- Review of TCP/IP
- Overview of network programming.
- Simple client-server programs.

Internet Layered Architecture

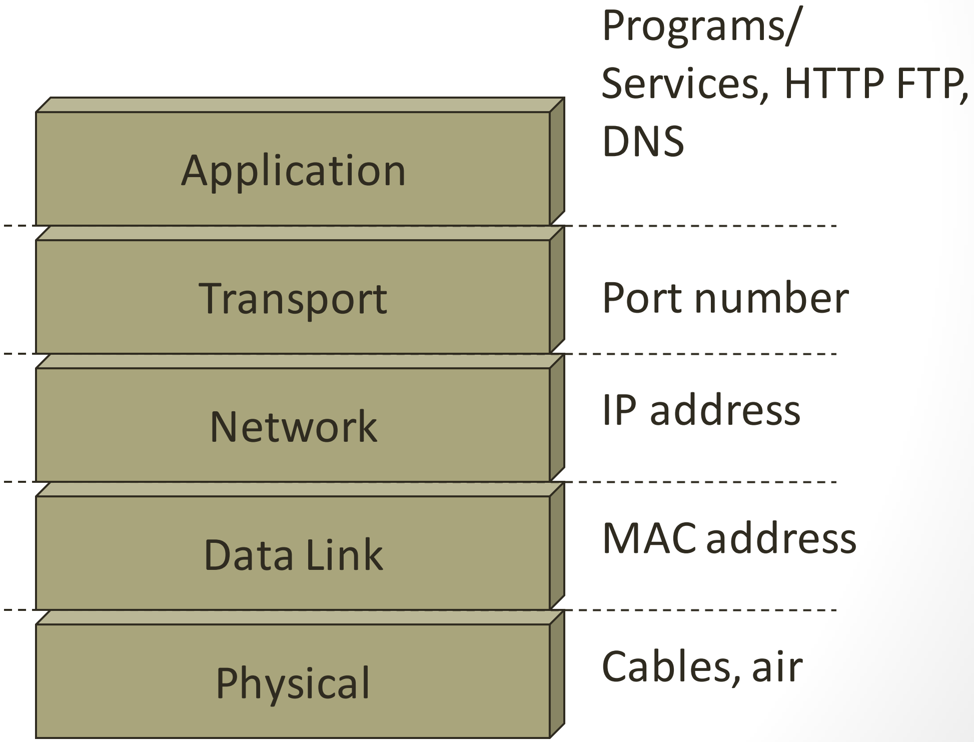
- Internet Layered Architecture is a “standard” of how data should be organized in the network by the hosts.

- There are 5 layers in Internet model.
  - Layer 1 is called physical layer
  - Layer 2 is called data link layer
  - Layer 3 is called network layer
  - Layer 4 is called transport layer
  - Layer 5 is called application layer



Corresponding Layer in Addressing

- Layer in the address scheme:
- Application Layer
  - Deals with client program and services such as FTP, HTTP, DNS, telnet, etc
- Transport Layer
  - Deals with port numbers
- Network Layer
  - Deals with IP address
- Data Link Layer
  - Deals with MAC address
- Physical Layer
  - Deals with electrical signal, cables and air



Application

Programs/ Services, HTTP FTP, DNS

Transport

Port number

Network

IP address

Data Link

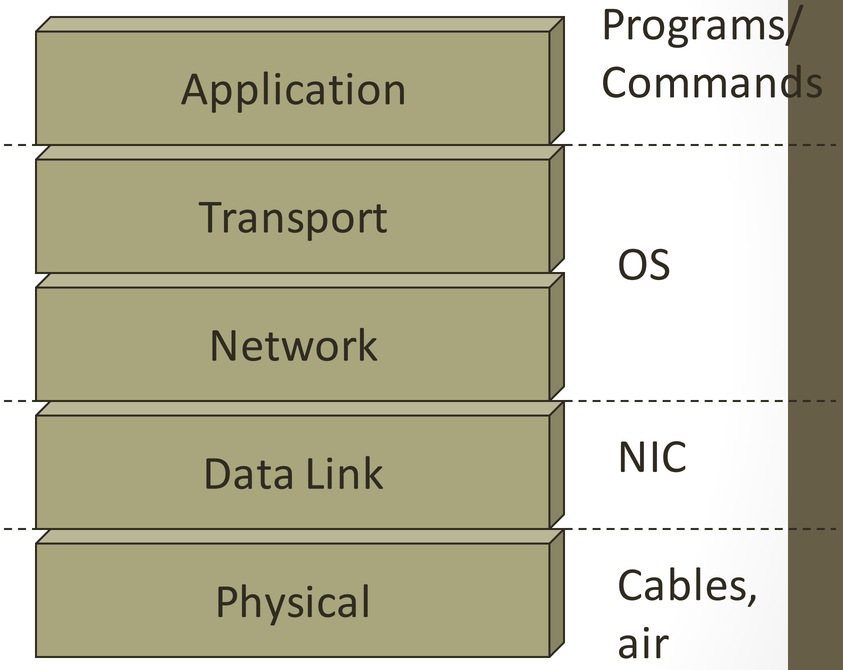
MAC address

Physical

Cables, air

Corresponding Layer in a Host

- For easy visualization of layered model in PC
- Layer 1 is made up of
  - Cables, transmission and reception of NIC
- Layer 2
  - Processing part of NIC
- Layer 3, 4, 5
  - CPU, and RAM



Why Layered Model?

Dealing with complex systems:

- Explicit structure allows identification, relationship of complex system's pieces
- Modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't affect rest of system
- Easy to swap in and out (upgrade) for each layer.
- Special people trained for each layer.

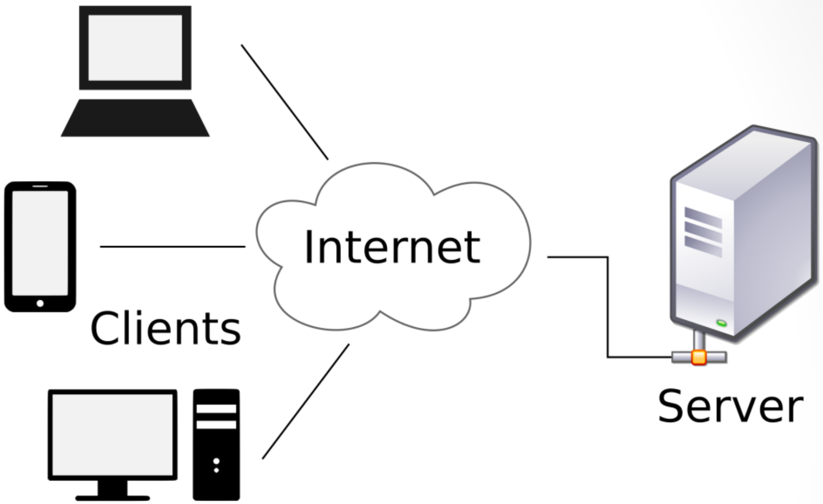
Overview of Network Programming

- **Network applications** are widely used
  - Web, email, & even many of the mobile apps that we use daily (e.g. WhatsApp, WeChat, etc.)

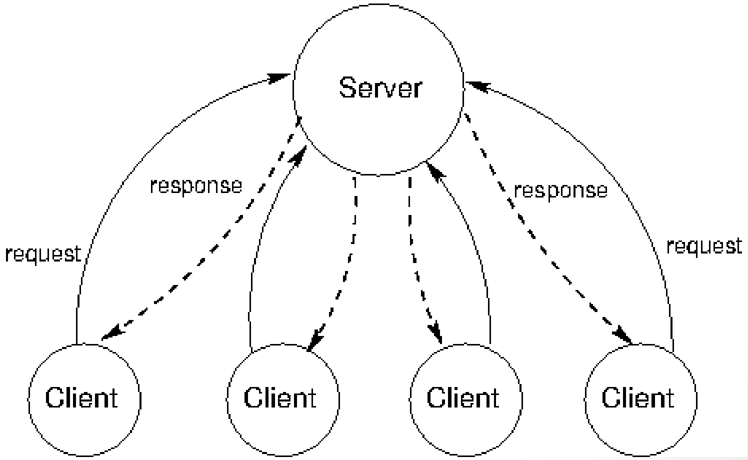
- **Network Programming** involves writing computer programs that enable processes to communicate with each other across a computer network.
  - Interestingly, all network applications are based on the same basic programming model, have similar overall logical structures, and rely on the same programming interface.

Client-Server Model

- Distributed application structure that partitions tasks or workloads between the providers of a resource or service, called **servers**, and service requesters, called **clients**.

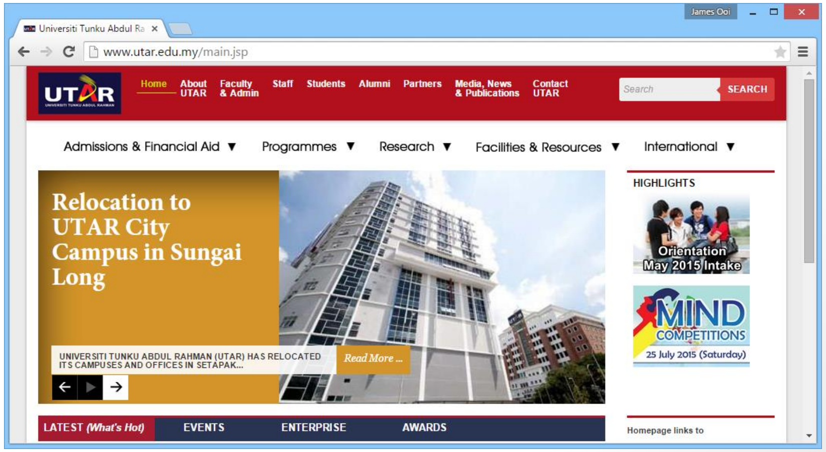


- A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function.



Examples of Client-Server Applications

- Web
  - Web server – e.g. Apache, Nginx, Microsoft IIS, etc.
  - Web browser (client) – e.g. Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari, etc.


Universiti Tunku Abdul Ra ×

www.utar.edu.my/main.jsp

James Ooi

UTAR

Home | About UTAR | Faculty & Admin | Staff | Students | Alumni | Partners | Media, News & Publications | Contact UTAR

Search | SEARCH

Admissions & Financial Aid ▼ | Programmes ▼ | Research ▼ | Facilities & Resources ▼ | International ▼

Relocation to UTAR City Campus in Sungai Long

UNIVERSITI TUNKU ABDUL RAHMAN (UTAR) HAS RELOCATED ITS CAMPUSES AND OFFICES IN SETAPAK...

Read More ...

HIGHLIGHTS

Orientation May 2015 Intake

MIND COMPETITIONS 25 July 2015 (Saturday)

LATEST (What's Hot) | EVENTS | ENTERPRISE | AWARDS

Homepage links to

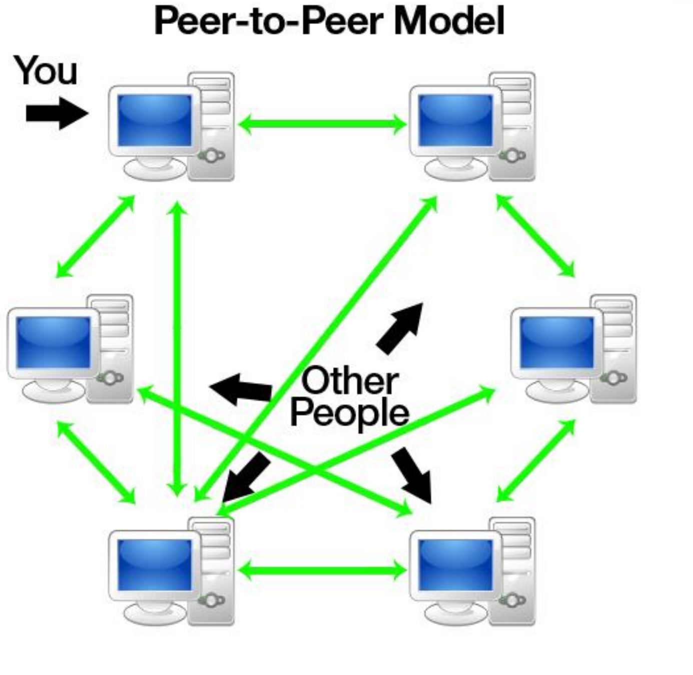
Examples of Client-Server Applications

- Email
  - Email Server – sendmail, qmail, Microsoft Exchange, etc.
  - Email Client – Microsoft Outlook, Mozilla Thunderbird, etc & web-based email such as Gmail, etc.

- Database
  - Database Server – e.g. MySQL Server
  - Database Client – e.g. phpmyadmin, a web-based MySQL client

Peer-to-Peer Model

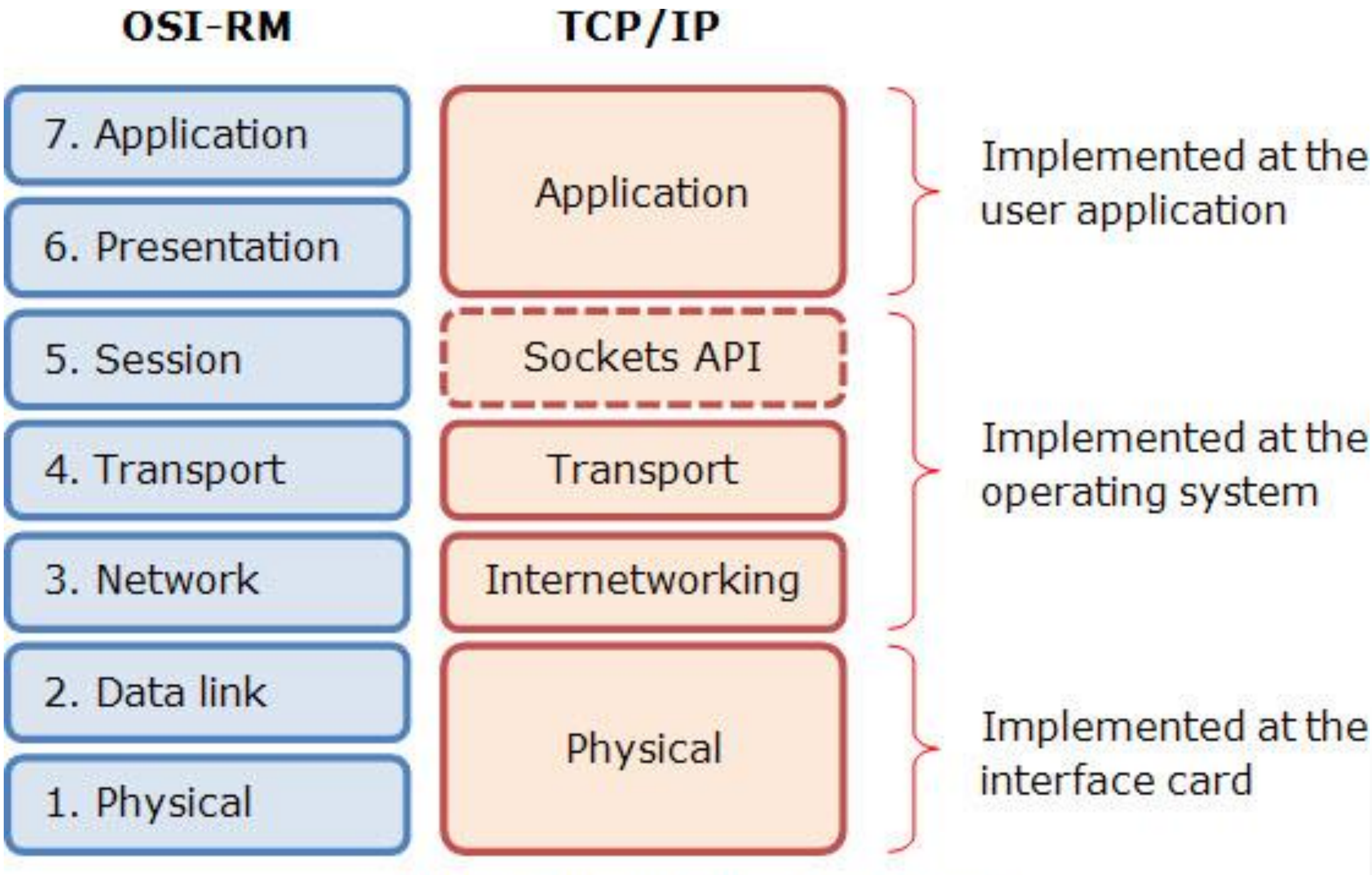
- Distributed application architecture that partitions tasks or work loads between peers.
- Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.
- Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts.
- Peers are both suppliers and consumers of resources.



Sockets API

- **Network socket**
  - An endpoint of an inter-process communication across a computer network.
  - Today, most communication between computers is based on the Internet Protocol; therefore most network sockets are Internet sockets.
- **Sockets API**
  - An API that allows application programs to control and use network sockets.
- **Socket address**
  - Combination of an IP address and a port number.
  - Based on this address, internet sockets deliver incoming data packets to the appropriate application process or thread.

Sockets API



OSI-RM

7. Application

6. Presentation

5. Session

4. Transport

3. Network

2. Data link

1. Physical

TCP/IP

Application

Sockets API

Transport

Internetworking

Physical

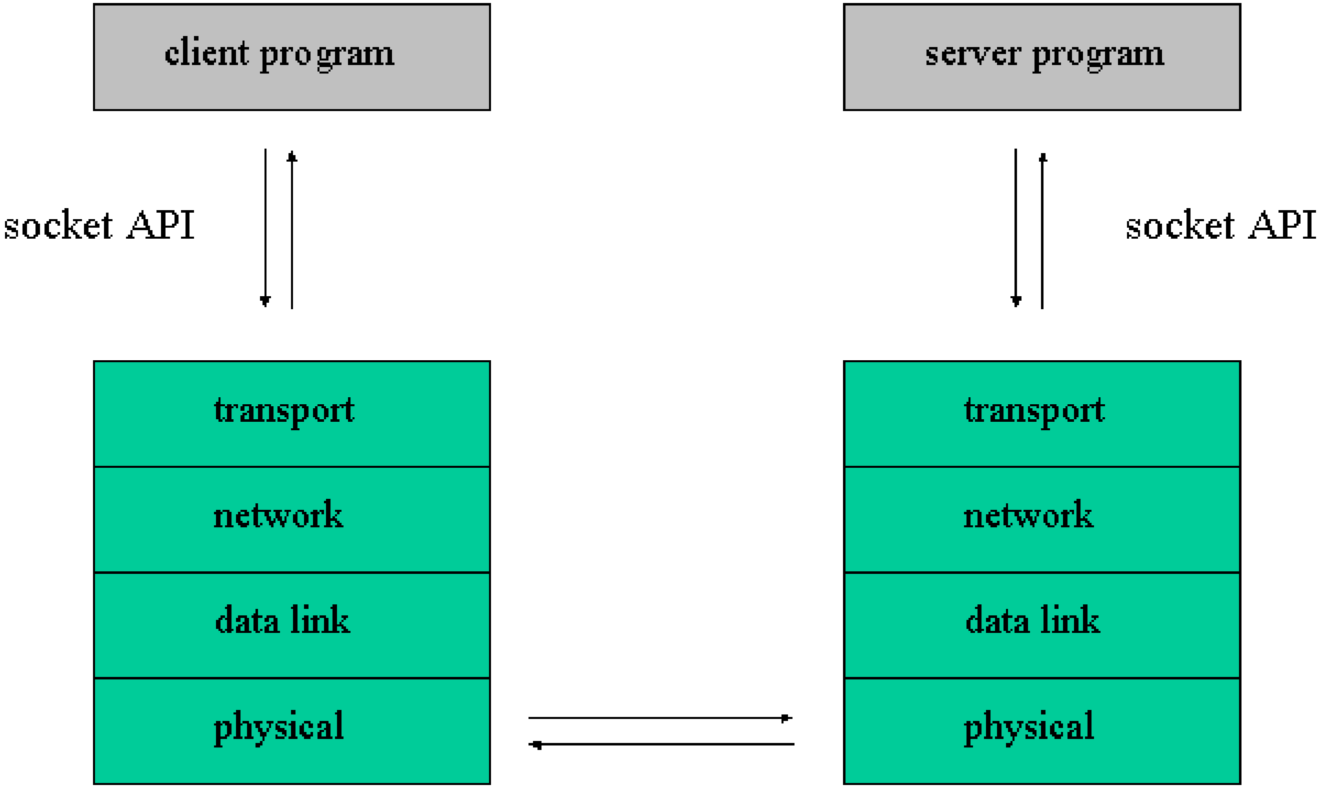
Implemented at the user application

Implemented at the operating system

Implemented at the interface card

OSI-RM and TCP/IP network architectures

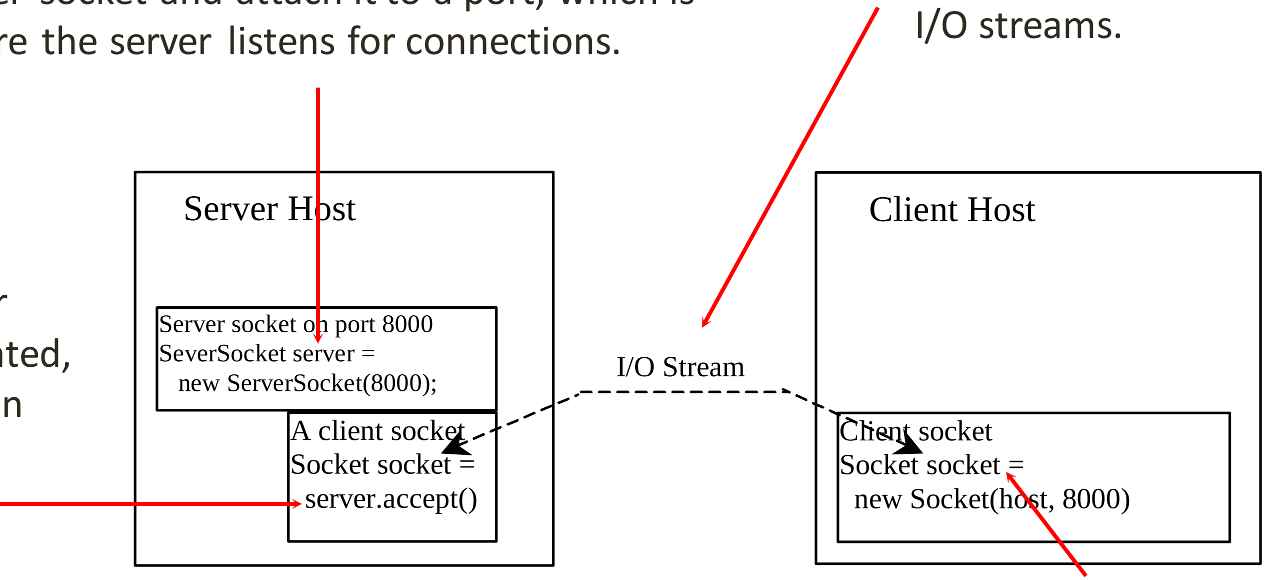
Sockets API


client program
server program
socket API
socket API
transport
network
data link
physical
transport
network
data link
physical

Client-Server Communications using Sockets

The server must be running when a client starts. The server waits for a connection request from a client. To establish a server, you need to create a server socket and attach it to a port, which is where the server listens for connections.

After the server accepts the connection, communication between server and client is conducted the same as for I/O streams.



Server Host

```
Server socket on port 8000
SeverSocket server =
  new ServerSocket(8000);
```

```
A client socket
Socket socket =
  server.accept()
```

I/O Stream

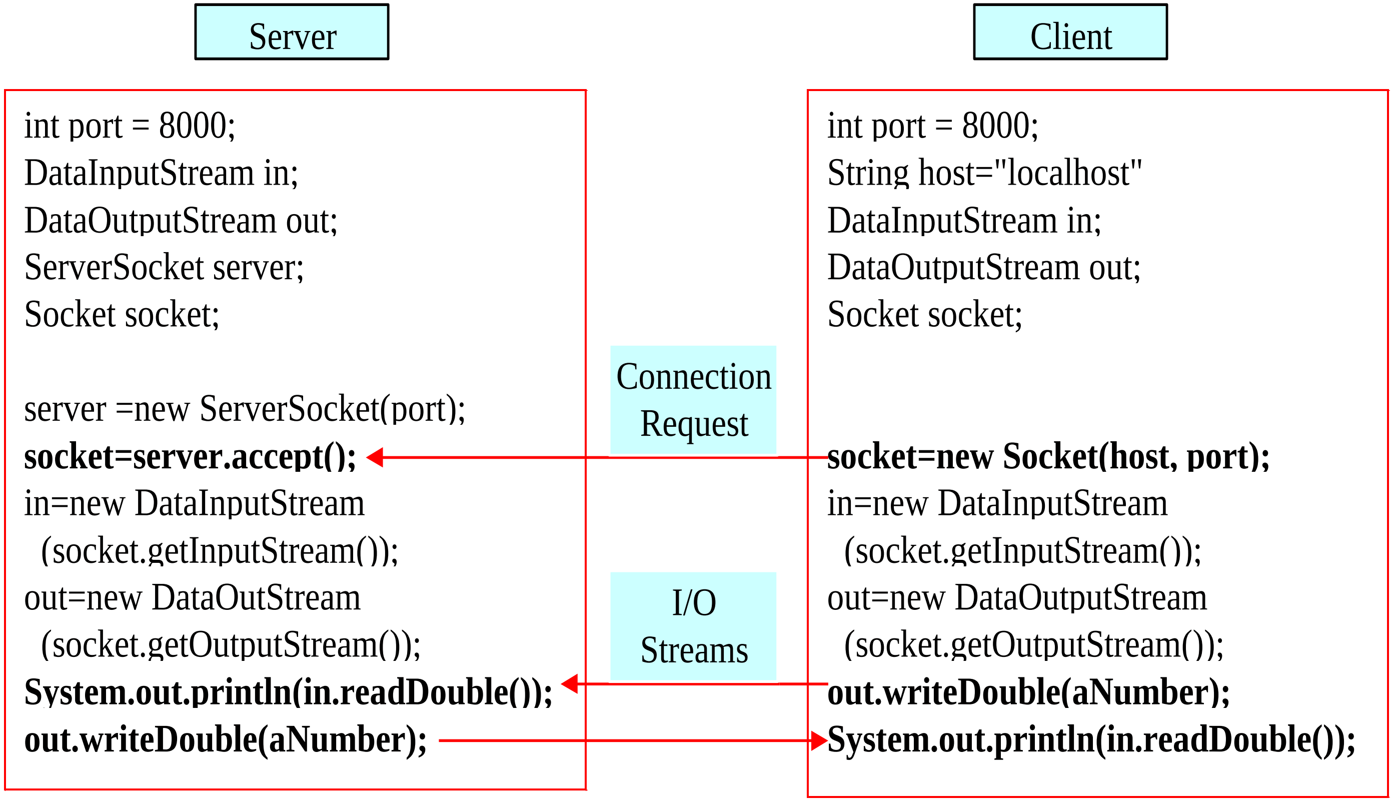
Client Host

```
Client socket
Socket socket =
  new Socket(host, 8000)
```

After a server socket is created, the server can use this statement to listen for connections.

The client issues this statement to request a connection to a server.

Data Transmission through Sockets



Server

```
int port = 8000;
DataInputStream in;
DataOutputStream out;
ServerSocket server;
Socket socket;

server =new ServerSocket(port);
socket=server.accept();
in=new DataInputStream
  (socket.getInputStream());
out=new DataOutStream
  (socket.getOutputStream());
System.out.println(in.readDouble());
out.writeDouble(aNumber);
```

Connection Request

I/O Streams

Client

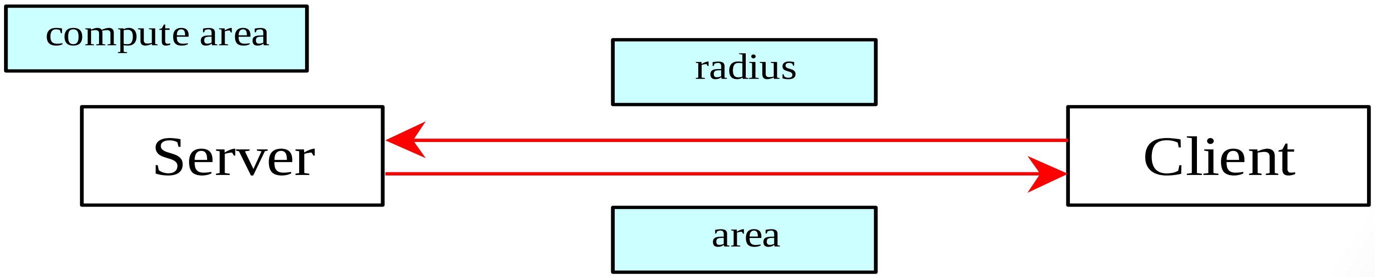
```
int port = 8000;
String host="localhost"
DataInputStream in;
DataOutputStream out;
Socket socket;


socket=new Socket(host, port);
in=new DataInputStream
  (socket.getInputStream());
out=new DataOutputStream
  (socket.getOutputStream());
out.writeDouble(aNumber);
System.out.println(in.readDouble());
```

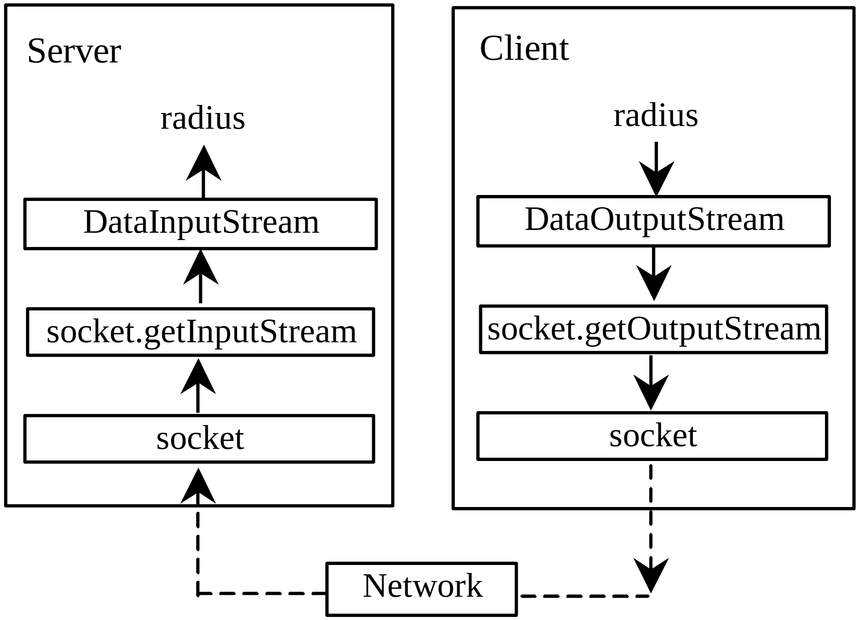
```
InputStream input = socket.getInputStream();
OutputStream output = socket.getOutputStream();
```

A Client/Server Example

- Problem: Write a client to send data to a server. The server receives the data, uses it to produce a result, and then sends the result back to the client. The client displays the result on the console. In this example, the data sent from the client is the radius of a circle, and the result produced by the server is the area of the circle.



A Client/Server Example, cont.



Server

radius

DataInputStream

socket.getInputStream

socket

Client

radius

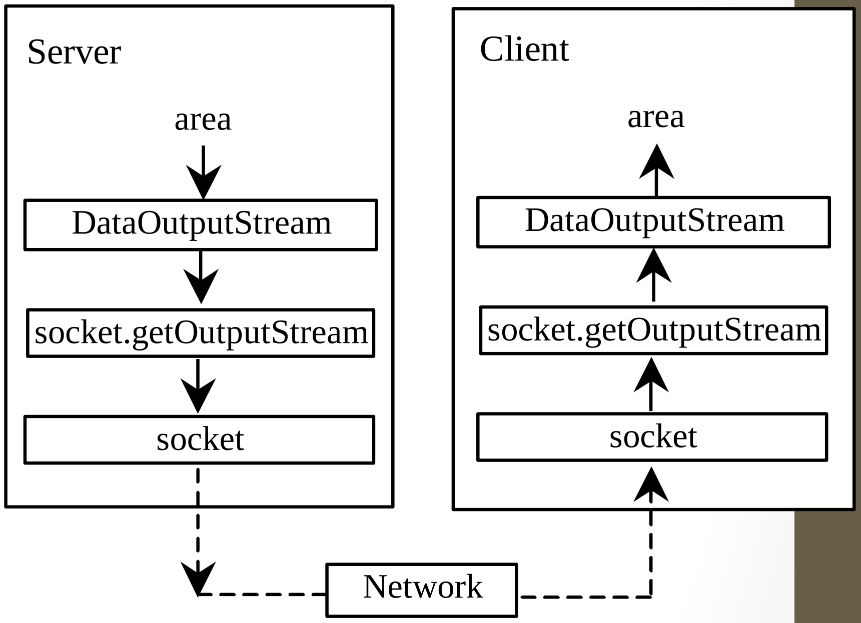
DataOutputStream

socket.getOutputStream

socket

Network

(A)



Server

area

DataOutputStream

socket.getOutputStream

socket

Client

area

DataOutputStream

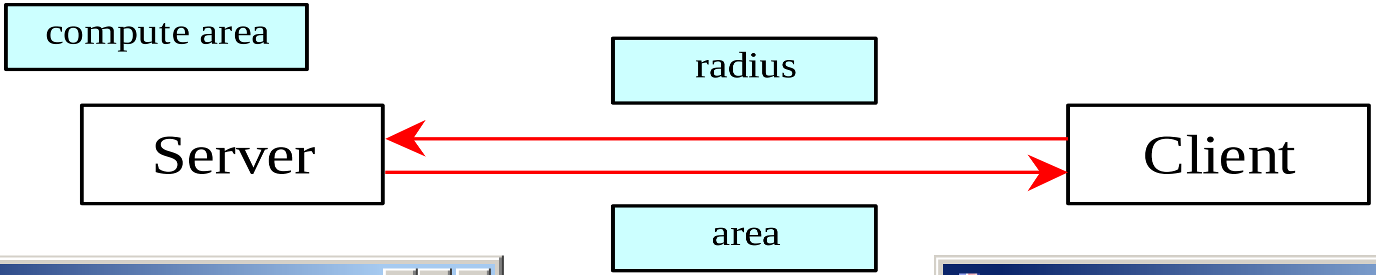
socket.getOutputStream

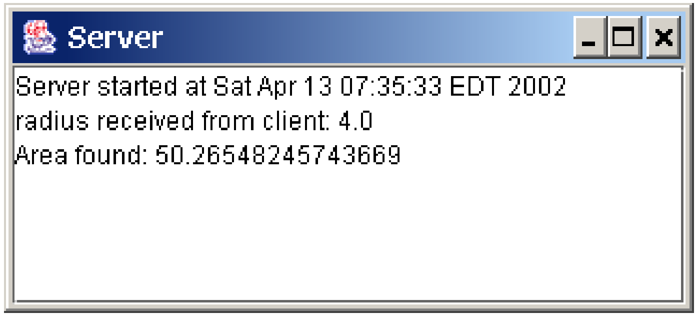
socket

Network

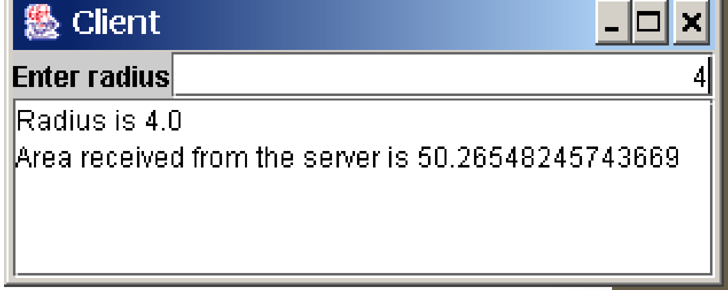
(B)

A Client/Server Example, cont.





```
Server started at Sat Apr 13 07:35:33 EDT 2002
radius received from client: 4.0
Area found: 50.26548245743669
```



Enter radius: 4

```
Radius is 4.0
Area received from the server is 50.26548245743669
```

Server.java

Client.java

Note: Start the server, then the client.