UNIT 1
INTRODUCTION
LH – 3HRS

PRESENTED BY:

ER. SHARAT MAHARJAN

NETWORK PROGRAMMING

CONTENTS (LH – 5HRS)

1.1 Network Programming Features and Scope

1.2 Network Programming Language, Tools & Platforms

1.3 Client and Server Applications

1.4 Client server model and software design

1.1 Network Programming Features and Scope

Introduction

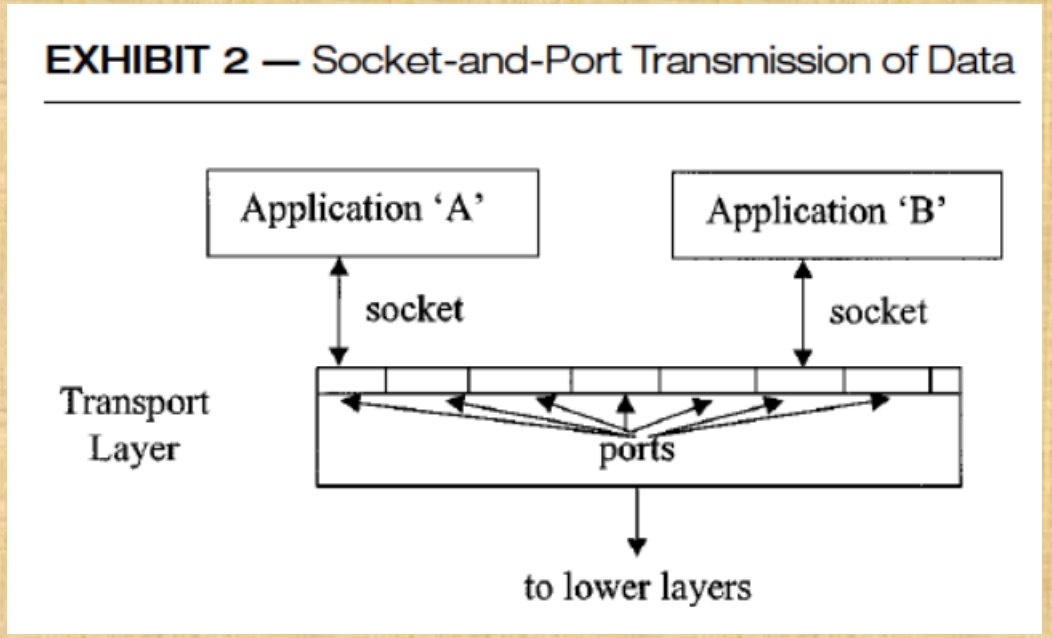
- Network Programming involves writing programs that communicate with other programs across a computer network.
- A server is an application that provides a "service" to various clients who request the service.
- There are many client/server scenarios in real life:
  - Bank tellers (server) provide a service for the account owners (client)
  - Waitresses (server) provide a service for customers (client)
  - Travel agents (server) provide a service for people wishing to go on vacation (client)

Java Networking Programming:

- Java Networking Programming is a concept of connecting two or more computing devices together so that we can share resources with the help of Coding.
- Java socket programming provides facility to share data between different computing devices.

Advantage of Java Networking

1. sharing resources
2. centralize software management



NETWORKING Programming Features

In this section, Java networking primitives will be discussed that enable the programmer to develop applications that interact directly with the layer 4 (Transport).

These primitives are found within the java.net package.

1. Connection-Oriented Networking
2. Connectionless Networking
3. Multicast Connectionless Networking
4. High-Level Java Networking Abstractions
5. URL-Based Programming
6. Remote Method Invocation (RMI)

1. Connection-Oriented Networking

- First, a connection is established between two parties. Once established, communications take place by having each party send and listen as needed. Once finished, the connection is closed. As discussed above, rather than using a telephone, each party “speaks” into and “listens” from a socket.
- For connection-oriented networking, there are actually two classes of socket objects in Java: “Socket,” which implements client sockets, and “ServerSocket,” which implements server sockets.
- A socket is defined as an end-point in the flow of communication between any two programs or channels. The sockets are created by using a set of requests in programming, also called socket API (Application Programming Interface).

For example, among the constructors for the **client-side socket** are the following:

- **Socket (InetAddress, int)** — creates a socket and connects it to the specified port on the host at the specified IP address.
- **Socket (String, int)** — creates a socket and connects it to the specified port on the host named in String.

Constructors on the **server side** include the following:

- **ServerSocket (int)** — creates a server socket and binds it to the specified port on the local host.
- **ServerSocket (int, int)** — same as above, but allows the programmer to specify the maximum allowable backlog of pending requests.
- **ServerSocket (int, int, InetAddress)** — same as the previous constructor, but allows programmer to specify which network interface on multi-homed machines (e.g., machines sitting on firewalls).

2. Connectionless Networking

- Java provides a separate type of socket class for this purpose: the **DatagramSocket**. In this case, there is no distinction between a server socket and a client socket, since datagrams are sent and received outside the context of a connection.
- Fundamentally a simpler concept, the DatagramSocket has only three constructors:
  - **DatagramSocket ( )** — This constructor creates a socket and binds it to any available port on the local machine.
  - **DatagramSocket (int**) — This constructor creates a socket and binds it to the specified port on the local machine.
  - **DatagramSocket (int, InetAddress)** — This constructor creates a socket and binds it to the specified port/interface combination on the local machine.
- DatagramPacket methods are provided to get or set the datagram's address, port, data, or length as needed.

3. Multicast Connectionless Networking

- Connectionless networking can also be accomplished between multiple parties, as opposed to the limited notion of point-to-point networking.
- Java provides a **MulticastSocket** class for this purpose as a subclass of DatagramSocket. Of particular interest are the methods **joinGroup** and **leaveGroup**, which allow the system to join and leave a particular multicast group, and the methods **getTTL** and **setTTL**, which handle the datagram's time-to-live attribute.

4. High-Level Java Networking Abstractions

- Up to this point, the discussion has been limited to fairly low-level networking concepts, focused at the transport layer service interface, the socket.
- Although extremely powerful when compared with the networking features of most other languages, these features are little more than primitives within the context of Java networking features.

5. URL-Based Programming

- In general terms, URL based programming allows the programmer to focus on the concepts associated with actually handling a remote object, rather than on all the lower- level mechanisms involved in creating a socket and binding it to a port, establishing a connection to the object's machine, locating the object, and retrieving information from or sending information to the object.
- This class, whose name stands for Uniform Resource Locator, uses a standard notation for representing a resource on the network.
- The Java URL class provides four constructors to allow flexibility in the way a URL object is created:

- **URL (String)** — allows the creation of a URL object by specifying a complete, familiar URL specification such as http://www.yahoo.com/
- **URL (String, String, int, String)** — allows the creation of a URL object by separately specifying the **protocol, host name, port, and file name**
- **URL (String, String, String)** — same as the previous constructor, except that the default port is assumed
- **URL (URL, String)** — allows the creation of a URL by specifying its path relative to an existing URL object

6. Remote Method Invocation (RMI)

- RMI provides the Java programmer with the capability of developing truly distributed, yet fully cooperative Java-only applications.
- These cooperating Java components can be peer applications, client and server applications, or client applets interacting with server applications.
- Compared with URL-based programming, RMI allows an order of magnitude increase in the degrees of complexity and sophistication of the resulting networked applications.
- defined in the java.rmi family of packages
- At a very high level, RMI requires the development of two components:
  - a Java object that implements a method through a remote interface, and a Java object that remotely invokes that method.
  - These two objects may be on the same machine or on different machines

Network Programming Scope

- **Professions and Industries:** – Network programming developers, software engineers, back end developers, java programmers – Used by employers in information technology, engineering, professional services and design
- **Major Organizations**: Google, Pinterest, Instagram, YouTube, DropBox…
- **Specializations and Industries:** Web and Internet development (frameworks, micro-frameworks and advanced content management systems); scientific and numeric computing; desktop graphical user interfaces (GUIs)

1.2 Network Programming Language, Tools & Platforms

Network Programming Language

- Network Programming involves writing programs that communicate with other, programs across a computer network.
- There are many issues that arise when doing network programming which do not appear when doing single program applications.
- However, JAVA makes networking applications simple due to the easy-to-use libraries.

Languages and Tools & Platforms

It is available for many languages on many platforms:

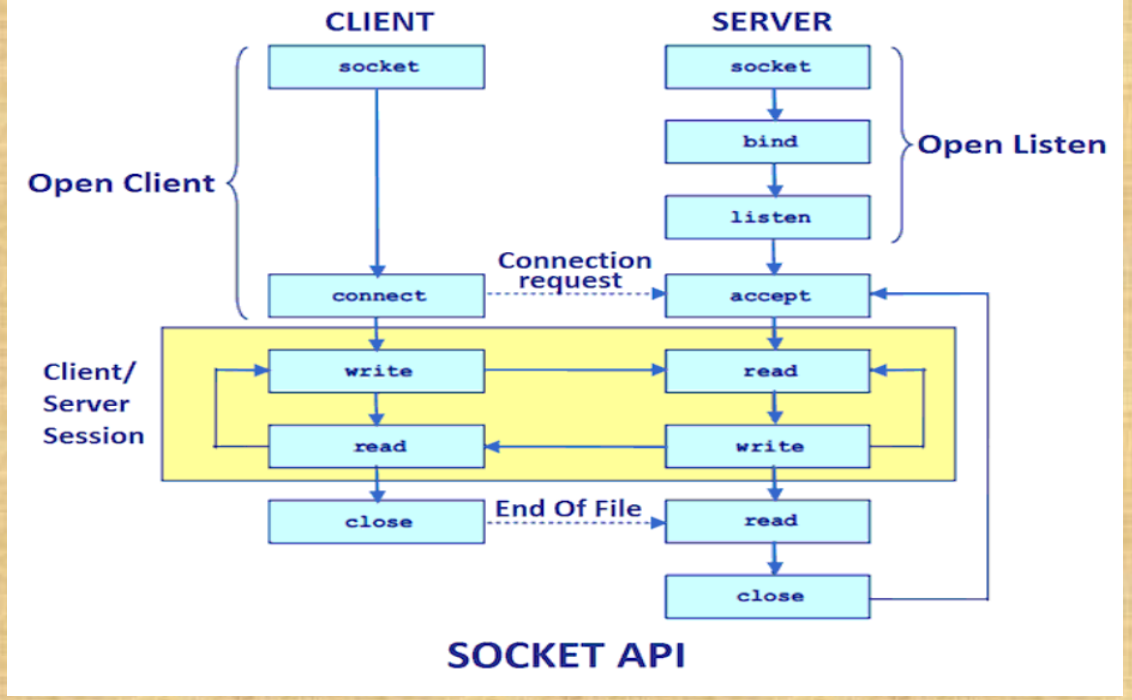
- C, Java, Perl, Python,...
- Support in Windows …..
- Programs written in any language and running on any platform can communicate with each other!

Tools Available

- The tools those are essential/ available for Network Programming environment are JDK(Java Developers Kit), Java Packages, Java Enabled web browsers and other Third party tools.

1.3 Client and Server Applications

- Java Socket programming is used for communication between the applications running on different JRE.
- Java Socket programming can be connection-oriented or connection-less.
- Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.
- The client in socket programming must know two information:
  - 1. IP Address of Server, and
  - 2. Port number.



- Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

Example of Java Networking Programming

- **Creating Server**: To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.
  - **ServerSocket ss=new ServerSocket(6666);** //opening a port
  - **Socket s=ss.accept();**//establishes connection and waits for the client
- **Creating Client**: To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.
  - **Socket s=new Socket("localhost",6666);**
- Let's see a simple of Java socket programming where client sends a text and server receives and prints it.

1.4 Client server model and software design

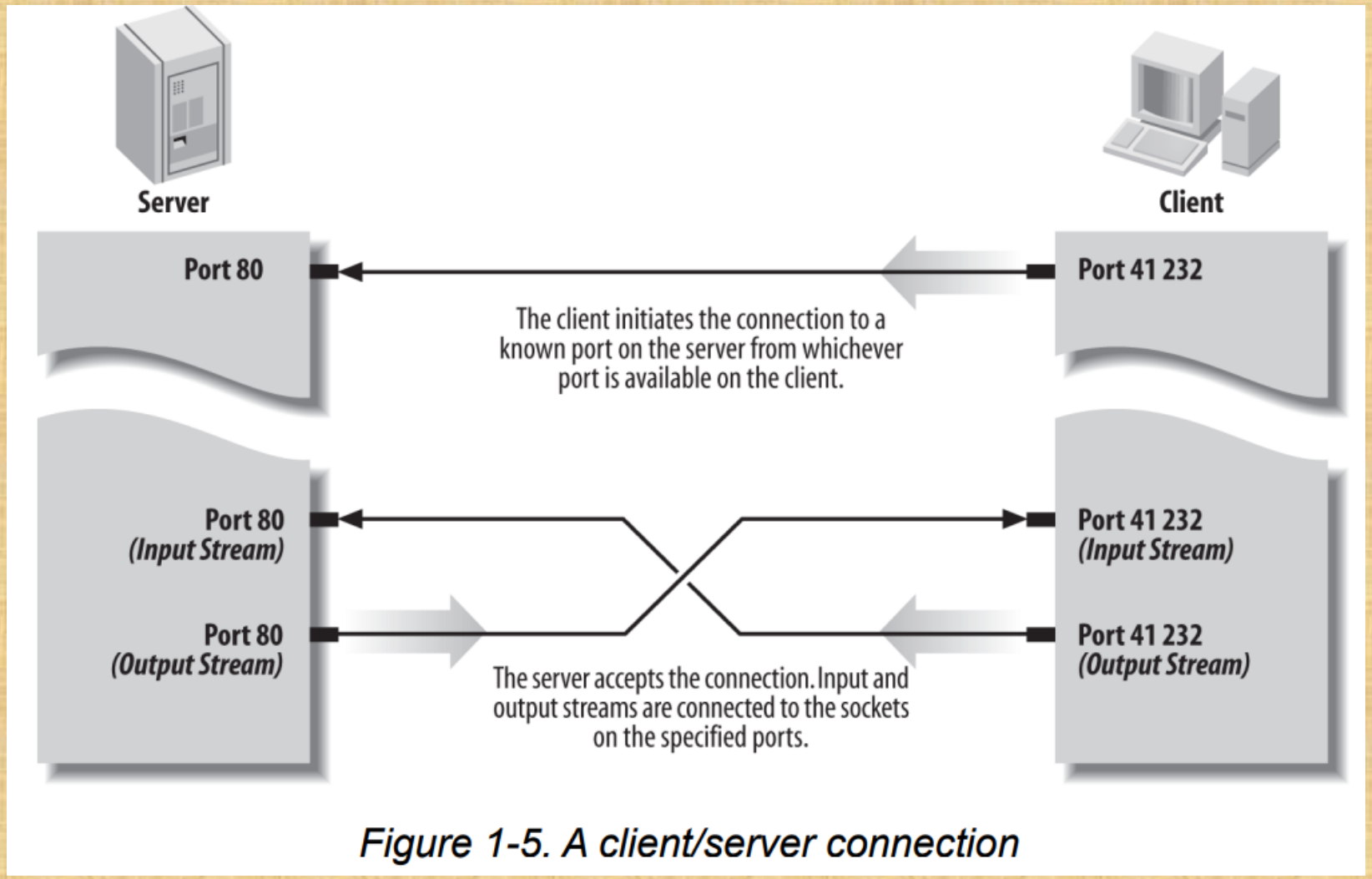


Figure 1-5. A client/server connection

- One side in any pair of communicating application must start execution and wait for the other side to contact it.
- A more reliable distinction is that a client initiates a conversation while a server waits for clients to start conversations with it. Figure 1-5 illustrates both possibilities.
- FTP is a service that fits the client/server model. People often use FTP to upload files from the client to the server, but it is still true that an FTP client initiates the connection and the FTP server responds.

Client Software Design

- Standard application service examples:
  - Remote login, TELNET protocol
  - E-mail client, SMTP or POP protocol
  - File transfer client, FTP protocol
  - Web browser, HTTP protocol
- Non-standard application service examples
  - Music or video transfer
  - Voice communication
  - Distributed database access

THANK YOU FOR YOUR ATTENTION