Network Programming [CACS355]

Er.Sital Prasad Mandal

BCA- VI
Mechi Campus
Bhadrapur, Jhapa, Nepal

(Email : info.sitalmandal@gmail.com)

Text Book

- Elliotte Rusty Harold, “Java Network Programming” O’Reilly, 2014
- David Reilly, Michael Reilly, “ Java Netwoking Programming and DistributedComputing”

Course Content - 1

1. Introduction

1.1. Network Programming Features and Scope

1.2. Network Programming Language, Tools & Platfoms

1.3. Client and Server Applications

1.4. Client Server model and Software Design

1. 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)

1. Introduction

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

1. Introduction

Do You Know ?

- How to perform connection-oriented Socket Programming in networking ?
- How to display the data of any online web page ?
- How to get the IP address of any host name e.g. www.google.com ?
- How to perform connection-less socket programming in networking ?

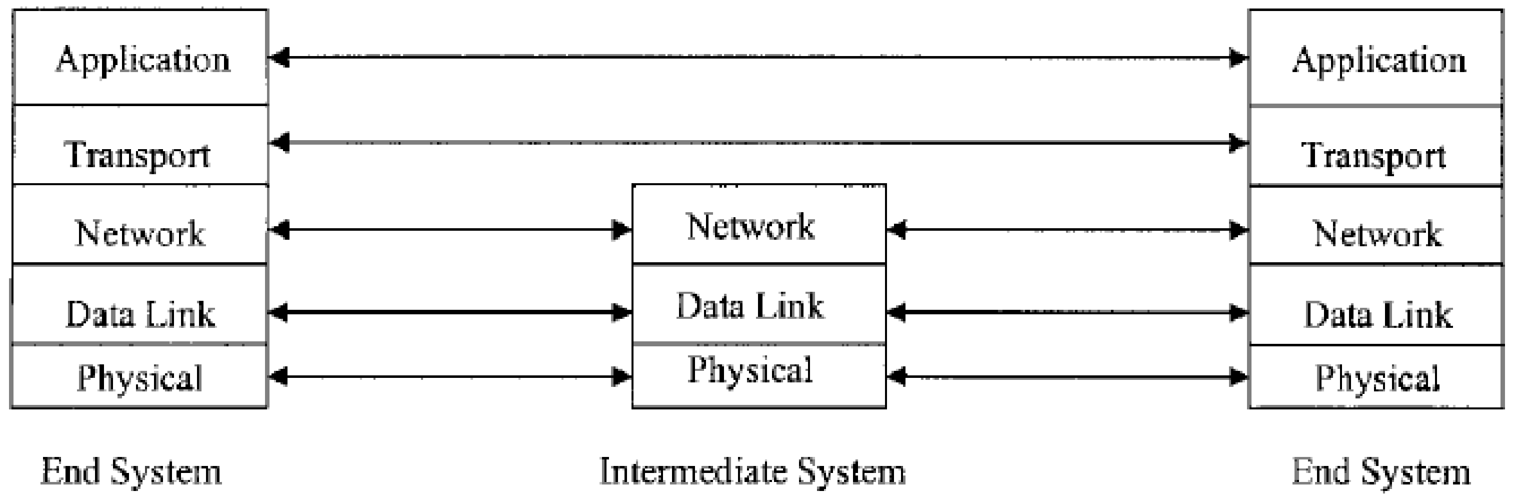
1. Introduction

Do You Know ?

- How to perform connection-oriented Socket Programming in networking ?
- How to display the data of any online web page ?
- How to get the IP address of any host name e.g. www.google.com ?
- How to perform connection-less socket programming in networking ?

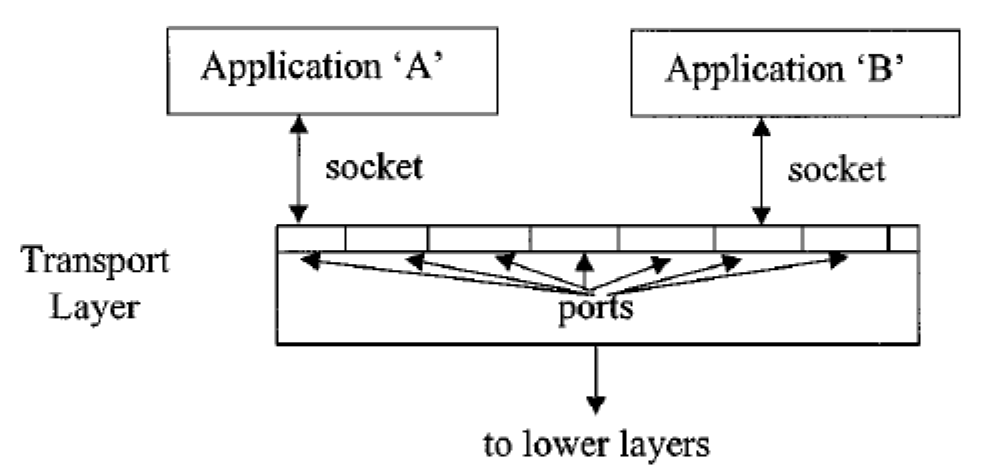
1. Introduction

EXHIBIT 1 — Layered Communications Model of the Internet



1. Introduction

EXHIBIT 2 — Socket-and-Port Transmission of Data



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.

- **Connection-Oriented Networking**
- **Connectionless Networking**
- **Multicast Connectionless Networking**
- **High-Level Java Networking Abstractions**
- **URL-Based Programming**
- **Remote Method Invocation (RMI)**

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 broken. 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)**.

Connection-Oriented Networking

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).

- high-level **methods** such as getInputStream and getOutputStream as well as lower-level **methods** such as getInetAddress, getPort, and set- TcpNoDelay.

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.

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.

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.

URL-Based Programming

In general terms, URLbased 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-Based Programming

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

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)

Network Programming Scope

- **Growing Opportunities**
- **Automation**
- **Innovation**
- **Job Opportunities**

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.

Network Programming Language, Tools & Platfoms

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!**
- **Writing communicating programs in different languages is a good exercise**

Network Programming Language, Tools & Platfoms

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.**

Each of these are briefly outlined below :

Network Programming Language, Tools & Platfoms

- Java Developers Kit (JDK) : As for the JDK utilities, there are seven main programs in the kit :

1. javac: The Java compiler. This program compiles Java source codes into byte codes.
2. java: The Java interpreter. This program runs Java byte codes.
3. javadoc: Generates API documentation in HTML format from Java source code.
4. appletviewer: A Java interpreter that executes Java applet (a special kind of Java Program) classes.
5. jdb: The Java debugger. Helps us find and fix bugs in Java programs.
6. javap: The Java disassembler. Displays the accessible functions and data in a compiled class file. It also displays the meaning of the byte codes.
7. javah: Creates header files that can be used to make routines, that can call Java routines,or make routines that can be called by Java programs.

Network Programming Language, Tools & Platfoms

Java Packages

- The Java language provides suite of packages that include a windowing tool kit, utilities, general I/O, tools, and networking.
- Following mentioned six packages are very popular:

1.java.applet: This package includes classes designed for use within an applet. There is one class, Applet and three interfaces AppletContex, AppletStub, and AudioClip.

2.java.awt: The Abstract Windowing Toolkit (AWT) package, awt, contains classes used to generate widgets and GUI (Graphical User Interface) components. This package includes the classes: Button, CheckBox, Choice, Component, Graphics, Menu, Panel, TextArea, and TextField.

3.java.io: The I/O Package include file input and output classes such as the classes FileInputStream and FileOutputStream. File I/O in subject to Security Control in applets.

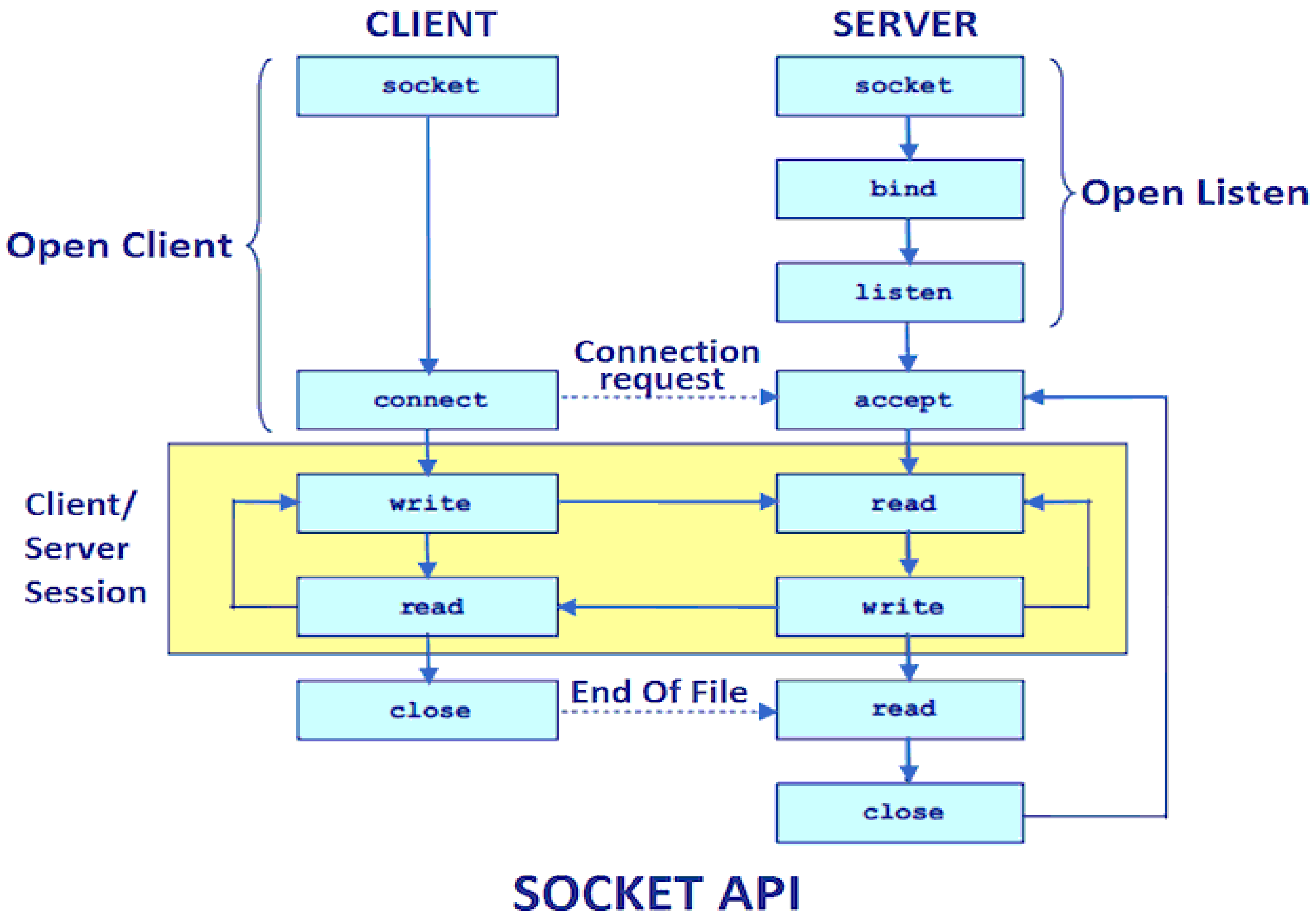
Network Programming Language, Tools & Platfoms

4. **java.lang:** This package includes the Java language classes, including Object, Thread, Exception, System, Integer, Float, Math, String, and so on.
5. **java.net:** This class supports the TCP/IP networking protocols and includes the Socket, ServerSocket, DatagramPacket, DategramSocket, URL, and URLConnection classes among others.
6. **java.util:** This class packages contains miscellaneous classes that are useful for a variety of programming.

1.3 Client 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.**

1.3 Client Server Applications



1.3 Client Server Applications

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.

1.3 Client Server Applications

Example of Java Networkig 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);
    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.

Example

File: MyServer.java

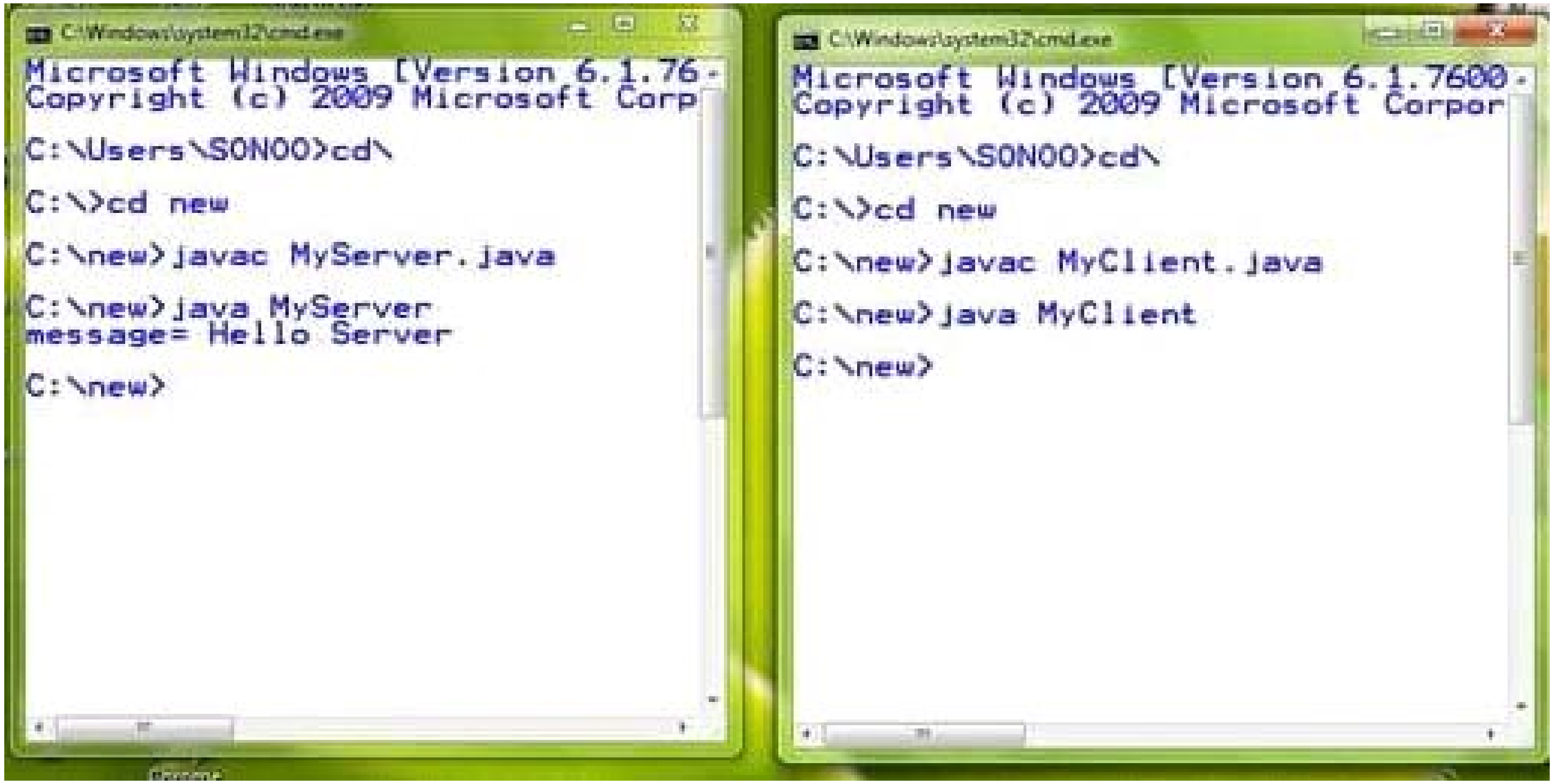
- **import java.io.*;**
- **import java.net.*;**
- **public class MyServer {**
- **public static void main(String[] args){**
- **try{**
- **ServerSocket ss=new ServerSocket(6666);**
- **Socket s=ss.accept();//establishes connection**
- **DataInputStream dis=new DataInputStream(s.getInputStream());**
- **String str=(String)dis.readUTF(); //return utf to string**
- **System.out.println("message= "+str);**
- **ss.close();**
- **}catch(Exception e){System.out.println(e);}**
- **}**
- **}**

File: MyClient.java

- **import java.io.*;**
- **import java.net.*;**
- **public class MyClient {**
- **public static void main(String[] args) {**
- **try{**
- **Socket s=new Socket("localhost",6666);**
- **DataOutputStream dout=new DataOutputStream(s.getOutputStream());**
- **dout.writeUTF("Hello Server");**
- **dout.flush();**
- **dout.close();**
- **s.close();**
- **}catch(Exception e){System.out.println(e);}**
- **}**
- **}**

To execute this program open two command prompts and execute each program at each command prompt as displayed in the below figure. After running the client application, a message will be displayed on the server console. UTF-*Stands for "Unicode Transformation Format.*

Example

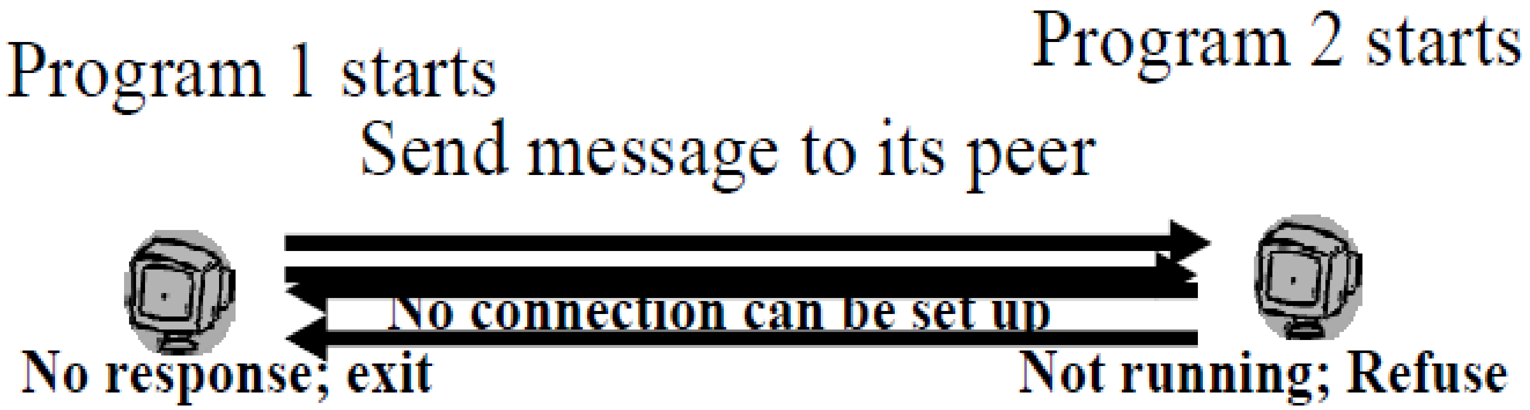


```
Microsoft Windows [Version 6.1.76
Copyright (c) 2009 Microsoft Corp

C:\Users\SONOO>cd\

C:\>cd new

C:\new>javac MyServer.java

C:\new>java MyServer
message= Hello Server

C:\new>
```

```
Microsoft Windows [Version 6.1.7600.
Copyright (c) 2009 Microsoft Corpor

C:\Users\SONOO>cd\

C:\>cd new

C:\new>javac MyClient.java

C:\new>java MyClient

C:\new>
```

Motivation

Scenario: A user tries to start two programs on separate machines and have them communicate.



The Client/Server Model

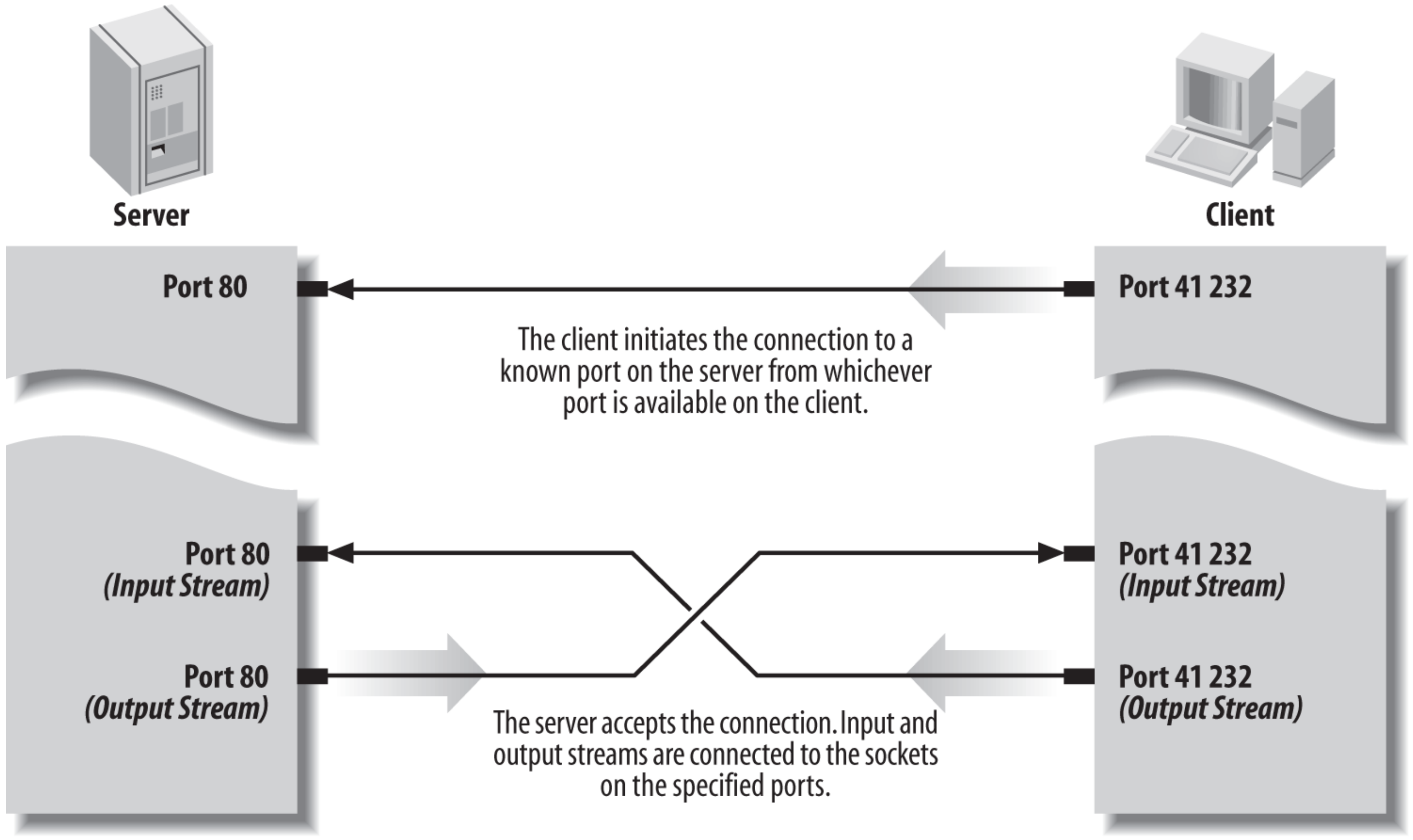

Server
Client
Port 80
Port 41 232
The client initiates the connection to a known port on the server from whichever port is available on the client.
Port 80 (Input Stream)
Port 80 (Output Stream)
Port 41 232 (Input Stream)
Port 41 232 (Output Stream)
The server accepts the connection. Input and output streams are connected to the sockets on the specified ports.

Figure 1-5. A client/server connection

The Client/Server Model

❖ One side in any pair of communicating application must start execution and wait for the other side to contact it.

❖ Since the client-server model places responsibility for meeting problem on application, TCP/IP does not need to provide mechanisms that automatically create a running program when a message arrives. *Instead, a program must be waiting to accept communication before any request arrive.*

❖ 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.

❖ Servers like Apache respond to requests from web clients like Firefox.

❖ FTP is an older 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.

The Client/Server Model

❖Not all applications fit easily into a client/server model.

❖The telephone system is the classic example of *a peer-to-peer network*. Each phone can either call another phone or be called by another phone. You don’t have to buy one phone to send calls and another to receive them.

❖Java does not have explicit peer-to-peer communication in its core networking API.

❖Alternatively, the peers can communicate with each other through an *intermediate server* program that forwards data from one peer to the other peers. This neatly solves the discovery problem of how two peers find each other.

Software Design Standard v.s. Nonstandard

Client Software Design

- ❑Standard application services
  - ✓Defined by TCP/IP
  - ✓Assigned well-known, universally recognized protocol port
- ❑Non-standard application services
  - ✓All other services which is not standard
  - ✓Or, locally-defined application services

Software Design Standard v.s. Nonstandard

Client Software Design

Be aware of the standard when outside the local environment

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

Java Networking Terminology

The widely used java networking terminologies are given below:

1. IP Address
2. Protocol
3. Port Number
4. MAC Address
5. Connection-oriented and connection-less protocol
6. Socket

Introduction

1) IP Address

IP address is a unique number assigned to a node of a network e.g. 192.168.0.1 . It is composed of octets that range from 0 to 255. It is a logical address that can be changed.

2) Protocol

A protocol is a set of rules basically that is followed for communication. For example:

- TCP
- FTP
- Telnet
- SMTP
- POP etc.

3) Port Number

The port number is used to uniquely identify different applications. It acts as a communication endpoint between applications.

The port number is associated with the IP address for communication between two applications.

Introduction

4) MAC Address

MAC (Media Access Control) Address is a unique identifier of NIC (Network Interface Controller). A network node can have multiple NIC but each with unique MAC.

5) Connection-oriented and connection-less protocol

In connection-oriented protocol, acknowledgement is sent by the receiver. So it is reliable but slow. The example of connection-oriented protocol is TCP.

But, in connection-less protocol, acknowledgement is not sent by the receiver. So it is not reliable but fast. The example of connection-less protocol is UDP.

6) Socket

A socket is an endpoint between two way communication.