

Distributed System and Cloud Computing Lab Manual

**KHYATI MANVAR,
DR. RASHMITA PRADHAN,
DIVAKAR JHA**

Faculty, Master of Computer Application (M.C.A.)

Late Bhausaheb Hiray S.S. Trust's Institute of Computer Application

DISTRIBUTED SYSTEM AND CLOUD COMPUTING LAB MANUAL

DISTRIBUTED SYSTEM AND CLOUD COMPUTING LAB MANUAL

LATE BHAUSAHEB HIRAY S.S. TRUST'S INSTITUTE OF COMPUTER
APPLICATION BANDRA, EAST MUMBAI-51

Author: Khyati Manvar

Co- Authors: Dr. Rashmita Pradhan, Divakar Jha

Faculty, Master of Computer Application (M.C.A.)

Late Bhausaheb Hiray S.S. Trust's Institute of Computer Application



First Edition, 2023

Copyright © Late Bhausaheb Hiray S.S. Trust's Institute Of Computer Application, Bandra (E), Mumbai-51, 2023

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the author, except in the case of brief quotations embodied in critical reviews and certain other non-commercial uses permitted by copyright law. For permission requests, write to the publisher at the address below.

This book can be exported from India only by the publishers or by the authorized suppliers. Infringement of this condition of sale will lead to Civil and Criminal prosecution.

Paperback ISBN: 978-81-19221-76-9

eBook ISBN: 978-81-19221-86-8

WebPDF ISBN: 978-81-19221-84-4

Note: Due care and diligence has been taken while editing and printing the book; neither the author nor the publishers of the book hold any responsibility for any mistake that may have inadvertently crept in.

The publishers shall not be liable for any direct, consequential, or incidental damages arising out of the use of the book. In case of binding mistakes, misprints, missing pages, etc., the publishers' entire liability, and your exclusive remedy, is replacement of the book within one month of purchase by similar edition/reprint of the book.

Printed and bound in India by
16Leaves
2/579, Singaravelan Street
Chinna Neelankarai
Chennai – 600 041, India

info@16leaves.com
www.16Leaves.com
Call: 91-9940638999

Title of the Program

1. Remote Process Communication	1
2. Remote Procedure Call	17
3. Remote Method Invocation	23
4. Remote Object Communication	31
5. Program on Mutual Exclusion	39
6. Implementation of Cloud Computing Services	47
7. Implementation of Identity Management using Cloud Computing concept	55
8. App Development using Cloud Computing	65
9. Application development using Google App Engine	75

Chapter 1 Remote Process Communication

Description:

Socket Programming refers to writing programs that execute across multiple computers in which the devices are all connected to each other using a network. A Server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request. The client and server can communicate by writing to or reading from their socket.

A **Socket** is simply an endpoint for communication between machines.

There are two communication protocol that one can use for Socket Programming:

- 1) Transmission control protocol (TCP)
- 2) User Datagram Protocol (UDP)

1) Transmission Control Protocol.

TCP is a connection oriented protocol. In order to do communication over the TCP Protocol, a connection must be established between the pair of sockets. While one of the socket listen for connection request (Server), the other ask for connection (client). Once two socket have been connected, they can be used to transmit data in both direction.

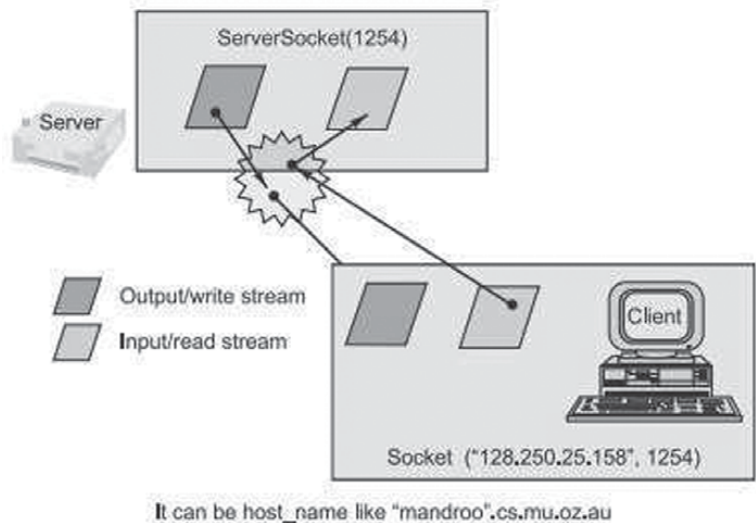


Fig: Socket-based client and server programming.

Classes used for Socket programming:

1) Socket

Socket class represents the socket that both the client and the server use to communicate with each other.

2) ServerSocket

The ServerSocket class used by server application to obtain a port and listen for client requests.

Steps for creating a simple Server Program:

Step 1) Open the Server Socket.

```
ServerSocket ss=new ServerSocket(PORT);
```

Step 2) Wait for the client request.

```
Socket client=ss.accept();
```

Step 3) Create I/O Streams for the communicating to the client.

```
DataInputStream dis=new DataInputStream(client.getInputStream());  
DataOutputStream dos=new DataOutputStream(client.getOutPutStream());
```

Step 4) Perform communication with the client.

```
Receive from client: => String str=dis.readUTF();  
Send data to client: => dos.writeUTF("Hello");
```

Step 5) Close the socket.

```
Client.close();
```

Steps for creating a simple Client Program:

Step 1) Create a Socket Object.

```
Socket s=new Socket(server,port_id);
```

Step 2) Create I/O Streams for the communicating to the server.

```
DataInputStream dis=new  
DataInputStream(client.getInputStream());  
DataOutputStream dos=new DataOutputStream(client.getOutPutStream());
```

Step 3) Perform communication with the server.

```
Receive data from server: => String str=dis.readUTF();  
Send data to the server: => dos.writeUTF("Hello");
```

Step 4) Close the socket.

```
s.close();
```

2) User Datagram Protocol.

UDP is a connection less protocol that allows for packets of data to be transmitted. Datagram Packets are used to implement a connection less packet delivery service supported by the UDP Protocol. Each message is transferred from source machine to destination based on information contained within that packet.

Msg length Host serverPort

The format of Datagram Packet is:

The class **DatagramPacket** contain several constructors that can be used for creating packet object. For Example: **DatagramPacket(byte[] buff, int length, InetAddress address, int port);**

The class datagramSocket support various methods that can be used for transmitting or receiving data over the network, The two key methods are:

void send(DatagramPacket p) => send a datagram packet from this socket.

void receive(DatagramPacket p) => receive a datagram packet from this socket.

Program I.I: Develop a program for one way client and server communication using java Socket, where client sends a message to the server, then the server reads the message and print it. <<<

Source Code:

Filename: DemoClient.java

```
import java.net.*; import java.io.*;

public class DemoClient
{
    public static void main(String args[]) throws Exception
    {
        Socket s=new Socket("localhost",1234);
        DataOutputStream dis=new DataOutputStream(s.getOutputStream()); dis.writeUTF("Hello!! How are you");
        s.close();
    }
}
```

Filename: DemoServer.java:

```
import java.net.*; import java.io.*;

public class DemoServer
{
    public static void main(String args[]) throws Exception
    {
        ServerSocket ss=new ServerSocket(1234); Socket s=ss.accept();
        DataInputStream dis=new DataInputStream(s.getInputStream()); String msg=dis.readUTF();
        System.out.println("Message from client:" +msg);
    }
}
```

OUTPUT:**DemoClient.java**

```
C:\Windows\system32\cmd.exe
G:\adc>javac DemoClient.java
G:\adc>java DemoClient
G:\adc>
```

DemoServer.java

```
C:\Windows\system32\cmd.exe
G:\adc>javac DemoServer.java
G:\adc>java DemoServer
Message from client : Hello !! How are you
G:\adc>
```

Program 1.2 Develop a program for client server chat using java socket. <<<**Source Code:****Filename: MyClient.java**

```
import java.net.*;
import java.io.*;
import java.util.*;

public class My_Client
{
```



```

public static void main(String args[]) throws Exception
{
    String str;
    Socket s=new Socket("localhost",3333);
    DataInputStream dis=new DataInputStream(s.getInputStream());
    DataOutputStream dos=new DataOutputStream(s.getOutputStream());
    Scanner in=new Scanner(System.in);
    while(true)
    {
        System.out.print("Client says:");
        str=in.nextLine();
        dos.writeUTF(str);
        str=dis.readUTF();
        System.out.println("Server says:" +str);
        if(str.equals("exit"))
            break;
    }
    s.close();
}
}

```

Filename: MyServer.java:

```

import java.net.*;
import java.io.*;
import java.util.*;
public class My_Server
{
    public static void main(String args[]) throws Exception
    {
        String str;
        ServerSocket ss=new ServerSocket(3333);
        Socket s=ss.accept();
        DataInputStream dis=new DataInputStream(s.getInputStream());
        DataOutputStream dos=new DataOutputStream(s.getOutputStream());
        Scanner in=new Scanner(System.in);
        while(true)
        {
            str=dis.readUTF();
            if (str.equals("exit"))

```

```

        {
            dos.writeUTF("exit");
            break;
        }
        System.out.println(" Client says:" +str);
        System.out.print("Server says:");
        str=in.nextLine();
        dos.writeUTF(str);
    }
    ss.close();
    s.close();
}
}

```

OUTPUT:

My_Client.java



```

C:\Windows\system32\cmd.exe
g:\adc>javac My_Client.java
g:\adc>java My_Client
Client says : hii
Server says : hello
Client says : how are you
Server says : i am fine
Client says : exit
Server says : exit
g:\adc>_

```

My_Server.java



```

C:\Windows\system32\cmd.exe
g:\adc>javac My_Server.java
g:\adc>java My_Server
Client says : hii
Server says : hello
Client says : how are you
Server says : i am fine
g:\adc>_

```

Program 1.3: Develop a program for one way client and server communication using Datagram Socket. <<<

Filename: UDPClient.java

```
import java.net.*;
import java.io.*;
public class UDPClient
{
    public static void main(String args[]) throws Exception
    {
        String s="How are you";
        DatagramSocket ds=new DatagramSocket();
        InetAddress ip=InetAddress.getByName("localhost");
        DatagramPacket p=new DatagramPacket(s.getBytes(),s.length(),ip,2222);
        ds.send(p);
    }
}
```

Filename: UDPServer.java:

```
import java.net.*;
import java.io.*;
public class UDPServer
{
    public static void main(String args[]) throws Exception
    {
        DatagramSocket ds=new DatagramSocket(2222);
        byte[] b=new byte[1024]; DatagramPacket p=new DatagramPacket(b,1024);
        ds.receive(p);
        String msg=new String(p.getData(),0,p.getLength());
        System.out.println("Message from client:" +msg);
    }
}
```