Problem Statement : **Write a program using UDP sockets for wired network to implement**
a. Peer to Peer Chat
b. Multiuser Chat

Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.

1. **Peer to peer UDP chat**

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;


public class P2P_ChatRead extends Thread {
      private DatagramSocket socket;
      private String message;
      private boolean chatting = true;


      public P2P_ChatRead (DatagramSocket socket) {
            this.socket = socket;
      }


      private void receiveAndShowMessage () {
            byte [] buffer = new byte [256];

        DatagramPacket inPacket = new DatagramPacket (buffer, buffer.length);

        try {
                  socket.receive (inPacket);
            } catch (IOException e) {
                  System.err.println (e);
            }

        message = new String (inPacket.getData (), 0, inPacket.getLength ());

        System.out.print ("Caller (" + socket.getLocalAddress () + ")>");
            System.out.println (message);
      }


      @Override
      public void run () {
            while (chatting) {
                  receiveAndShowMessage ();
            }
      }
}


/**
 *
 */
```

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Scanner;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;


/*
 * P2P Messenger client, based on threads. It is possible to connect with
another one
 * computer, after entering the IP address. The default address is localhost.
<br />
 * @author Dawid Samolyk
 */
public class P2P_ChatClient {
      private static DatagramSocket socket;
      private static InetAddress address;
      private static Scanner scanner = new Scanner (System.in);
      private static String info = new String ("Simple P2P Chat. Enter to
exit. Default host: localhost.");


      private static void createConnection () {
            try {
                  socket = new DatagramSocket ();
            } catch (IOException e) {
                  System.err.println ("Can not find connection!");
                  System.exit (1);
            }
      }

      private static void closeConnection () {
            socket.close ();
      }


      private static void enterHostIP () {
            System.out.print ("Enter host IP:");
            String hostIP = scanner.nextLine ();


            if (hostIP == "" || hostIP == null || hostIP == "") {
                  hostIP = "127.0.0.1";
            }


            try {
                  address = InetAddress.getByName (hostIP);
            } catch (UnknownHostException e) {
                  System.err.println (e);
            }
      }
	public static void main (String [] args) {
		System.out.println (info);


		enterHostIP ();
		createConnection ();
		sendTest ();


		startConversation ();


		closeConnection ();
	}


	/**
	 * Send a test package to the server so that it can determine the IP
address and port of the client.
	 */
	private static void sendTest () {
		String testMessage = new String ("");
		byte [] buffer = testMessage.getBytes ();


		DatagramPacket outPacket = new DatagramPacket (buffer, 0,
buffer.length, address, 4444);


		try {
			socket.send (outPacket);
		} catch (IOException e) {
			System.err.println (e);
		}
	}


	private static void startConversation () {
		ExecutorService executor = Executors.newFixedThreadPool (2);


		P2P_ChatWrite write = new P2P_ChatWrite (socket, address, 4444);
		P2P_ChatRead read = new P2P_ChatRead (socket);


		while (write.chatting == true) {
			executor.execute (write);
			executor.execute (read);
		}


		executor.shutdown ();
	}
}

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;


public      class P2P_ChatWrite extends Thread {
      private DatagramSocket socket;
      private InetAddress address;
      private int port;
      private String message;
      private static Scanner scanner = new Scanner (System.in);
      public boolean chatting = true;


      public P2P_ChatWrite (DatagramSocket socket, InetAddress address, int
port) {
            this.socket = socket;
            this.address = address;
            this.port = port;
      }


      @Override
      public void run () {
            try {


                  while (chatting) {
                        sendMessage ();
                  }


            } catch (Throwable e) {
                  System.err.println (e);
            }
      }


      private void sendMessage () throws Throwable {
            message = scanner.nextLine ();


            if (message.equals ("END")) {
                  chatting = false;
                  System.exit (1);


            } else if (message != null) {
                  byte [] buffer = message.getBytes ();
                  DatagramPacket outPacket = new DatagramPacket (buffer, 0,
buffer.length, address, port);


                  socket.send (outPacket);
            }
      }
}
```

```java
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.Scanner;


/**
 * P2P messenger server, based on threads. <br />
 * @author Dawid Samolyk
 */
public class P2P_ChatServer {
      private static DatagramSocket socket;
      private static InetAddress address;
      private static int port;
      private static String info = new String ("Simple P2P Chat. Enter to
exit.");

      private static void closeConnection () {
            socket.close ();
      }


      private static void createConnection () {
            try {
                  socket = new DatagramSocket (4444);
            } catch (IOException e) {
                  System.err.println ("Could not listen on port: 4444");
                  System.exit (1);
            }
      }


      public static void main (String [] args) {
            System.out.println (info);


            createConnection ();
            setClientsAddressAndPort ();


            startConversation ();


            closeConnection ();
      }


      /**
       * Receiving from the customer the test package on the basis of which
it is determined
       * is the address and port on which to send messages to the customer.
       */
      private static void setClientsAddressAndPort () {
            byte [] inBuf = new byte [256];
            DatagramPacket inPacket = new DatagramPacket (inBuf,
inBuf.length);


            try {
                  socket.receive (inPacket);
            } catch (IOException e) {
                  System.err.println (e);
            }


            port = inPacket.getPort ();
            address = inPacket.getAddress ();

            System.out.println ("Client connected!"
                        + "IP:"
                        + address
                        + ", port:"
                        + port);
      }


      private static void startConversation () {
            ExecutorService executor = Executors.newFixedThreadPool (2);


            P2P_ChatWrite write = new P2P_ChatWrite (socket, address, port);
            P2P_ChatRead read = new P2P_ChatRead (socket);


            while (write.chatting == true) {
                  executor.execute (write);
                  executor.execute (read);
            }


            executor.shutdown ();
      }
}
```

```
Output :
G:\CO5G\CN>javac P2P_ChatClient.java

G:\CO5G\CN>java P2P_ChatClient
Simple P2P Chat. Enter to exit. Default host: localhost.
Enter host IP:127.0.0.1
Hello


G:\CO5G\CN>javac P2P_ChatServer.java

G:\CO5G\CN>java P2P_ChatServer
Simple P2P Chat. Enter to exit.
Client connected!IP:/127.0.0.1, port:59031

Caller (0.0.0.0/0.0.0.0)>Hello
Hii
```

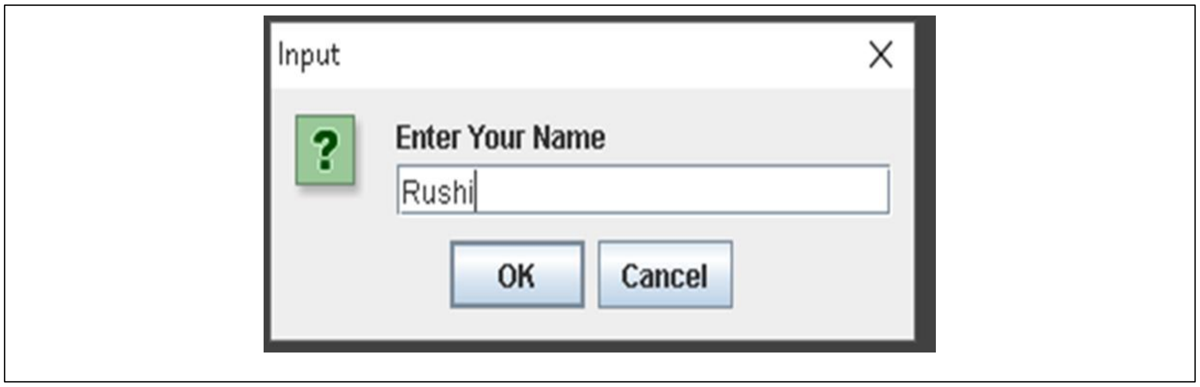
2. **Multiuser UDP chat**

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.ArrayList;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.WindowConstants;


public class ChatRoom extends JFrame{
    public static final int HOST_MODE=0;
    public static final int CLIENT_MODE=1;
    JButton btn_send;
    JScrollPane jScrollPane1;
    JTextArea jTextArea1;
    JLabel lbl_ipNroomName;
    JTextField txt_mymsg;
    int mode;
    String Name;
    String roomname;
    InetAddress hostip;
    ChatRoom pt;
    DatagramSocket socket;
    ArrayList<client> ClientList;
    byte[] b;

public ChatRoom(String myname,int mod,String ip,String room)
{
     try{
        Name=myname;
        mode=mod;
        hostip=InetAddress.getByName(ip);
        roomname=room;
        setLayout(null);
        setSize(400,460);
        lbl_ipNroomName = new JLabel("",SwingConstants.CENTER);
        txt_mymsg = new JTextField();
        btn_send = new JButton("Send");
        jScrollPane1 = new JScrollPane();
        jTextArea1 = new JTextArea(8,15);
        ClientList=new ArrayList<>();
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        add(lbl_ipNroomName);
        lbl_ipNroomName.setBounds(10,10,getWidth()-30,40);
        add(txt_mymsg);
        pt=this;

txt_mymsg.setBounds(10,lbl_ipNroomName.getY()+lbl_ipNroomName.getHeight(),get
Width()-130,30);
        add(btn_send);
        btn_send.setBounds(txt_mymsg.getWidth()+20,txt_mymsg.getY(),80,30);
        jScrollPane1.setViewportView(jTextArea1);
        add(jScrollPane1);

jScrollPane1.setBounds(10,btn_send.getY()+40,lbl_ipNroomName.getWidth(),getHe
ight()-20-jScrollPane1.getY()-110);
        btn_send.setEnabled(false);
        jTextArea1.setEditable(false);
        txt_mymsg.setEnabled(false);
        btn_send.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
        String s=txt_mymsg.getText();
        if(s.equals("")==false)
            {
            if(mode==HOST_MODE)
                broadcast(Name+": "+s);
            else
                sendToHost(Name+": "+s);
            txt_mymsg.setText("");
            }
            }
        });

        if(mode==HOST_MODE)
            {
            socket=new DatagramSocket(37988);
            lbl_ipNroomName.setText("My
IP:"+InetAddress.getLocalHost().getHostAddress());
            }
        else
            {
            socket=new DatagramSocket();
            String reqresp="!!^^"+Name+"^^!!";
            DatagramPacket pk=new
DatagramPacket(reqresp.getBytes(),reqresp.length(),hostip,37988);
            socket.send(pk);
            b=new byte[300];
            pk=new DatagramPacket(b,300);
            socket.setSoTimeout(6000);
            socket.receive(pk);
            reqresp=new String(pk.getData());
            if(reqresp.contains("!!^^"))
                {
                roomname=reqresp.substring(4,reqresp.indexOf("^^!!"));
                lbl_ipNroomName.setText("ChatRoom: "+roomname);
                btn_send.setEnabled(true);
                txt_mymsg.setEnabled(true);
                }
            else{
                JOptionPane.showMessageDialog(pt,"No response from the
server");System.exit(0);
                }
            }
        Messenger.start();
        }catch(Exception ex){JOptionPane.showMessageDialog(null,ex);}
}


public static void main(String args[]) {
        try {
        String host="",room="";
        String name=JOptionPane.showInputDialog("Enter Your Name");
        if(name==null||name.equals(""))
            {JOptionPane.showMessageDialog(null, "Name cannot be
blank");return;}
        int mode=JOptionPane.showConfirmDialog(null,"Create a chatroom or
connect to existing one?\nYes - Create Chat Room\nNo - Jion a Chat
Room","Create or Join?",JOptionPane.YES_NO_OPTION);
        if(mode==1)
            {
            host=JOptionPane.showInputDialog("Enter the host ip address");
            if(host==null||host.equals(""))
                {JOptionPane.showMessageDialog(null, "IP of host is
mandatory");return;}
            }
        else
            room=JOptionPane.showInputDialog("Name your chat room");
        ChatRoom obj= new ChatRoom(name,mode,host,room);
        obj.setVisible(true);
        } catch (Exception ex) {JOptionPane.showMessageDialog(null,ex);}
    }

public void broadcast(String str)
{
try {
DatagramPacket pack=new DatagramPacket(str.getBytes(),str.length());
for(int i=0;i<ClientList.size();i++)
    {
    pack.setAddress(InetAddress.getByName(ClientList.get(i).ip));
    pack.setPort(ClientList.get(i).port);
    socket.send(pack);
    }
jTextArea1.setText(jTextArea1.getText()+"\n"+str);
} catch (Exception ex) {JOptionPane.showMessageDialog(pt,ex);}
}

public void sendToHost(String str)
{
DatagramPacket pack=new
DatagramPacket(str.getBytes(),str.length(),hostip,37988);
try {socket.send(pack);} catch (Exception ex)
{JOptionPane.showMessageDialog(pt,"Sending to server failed");}
}

Thread Messenger=new Thread()
{
public void run()
{
try {
```

```
while(true)
    {
    b=new byte[300];
    DatagramPacket pkt=new DatagramPacket(b,300);
    socket.setSoTimeout(0);
    socket.receive(pkt);
    String s=new String(pkt.getData());
    if(mode==HOST_MODE)
        {
        if(s.contains("!!^^"))
            {
            client temp=new client();
            temp.ip=pkt.getAddress().getHostAddress();
            temp.port=pkt.getPort();
            broadcast(s.substring(4,s.indexOf("^^!!"))+" joined.");
            ClientList.add(temp);
            s="!!^^"+roomname+"^^!!";
            pkt=new
DatagramPacket(s.getBytes(),s.length(),InetAddress.getByName(temp.ip),temp.po
rt);
            socket.send(pkt);
            btn_send.setEnabled(true);
            txt_mymsg.setEnabled(true);
            }
        else
            {
            broadcast(s);
            }
        }
    else
        {
        jTextArea1.setText(jTextArea1.getText()+"\n"+s);
        }
    }
}catch (IOException ex)
{JOptionPane.showMessageDialog(pt,ex);System.exit(0);}
}
};
}

class client
{
public String ip;
public int port;
public String name;
}
```

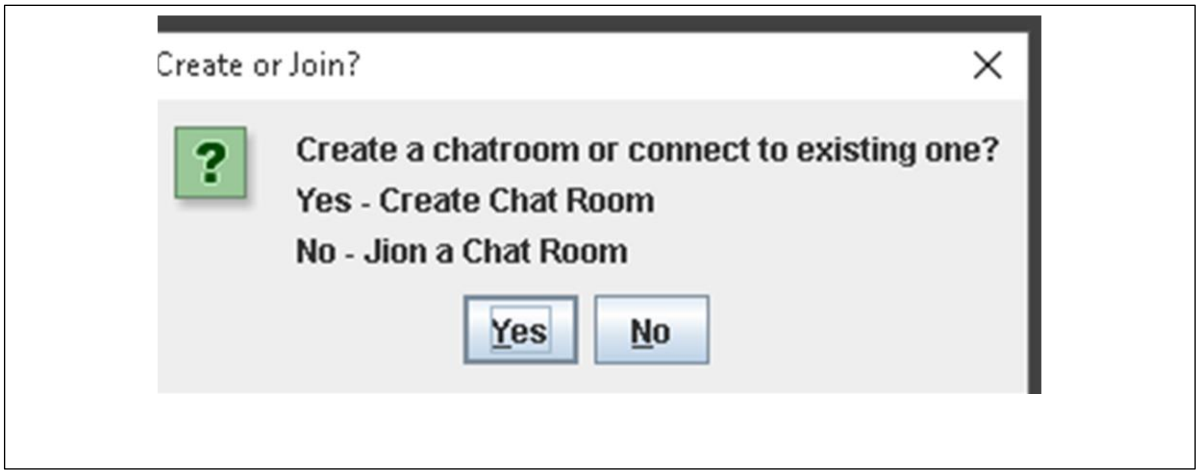
Output :

```
G:\CO5G\CN>java ChatRoom
```


Input

Enter Your Name

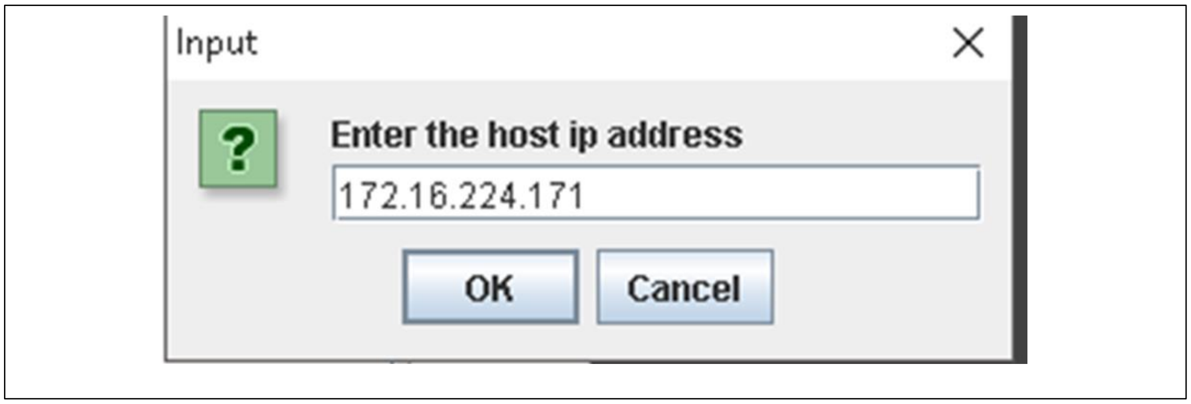
Rushi


Create or Join?

Create a chatroom or connect to existing one?

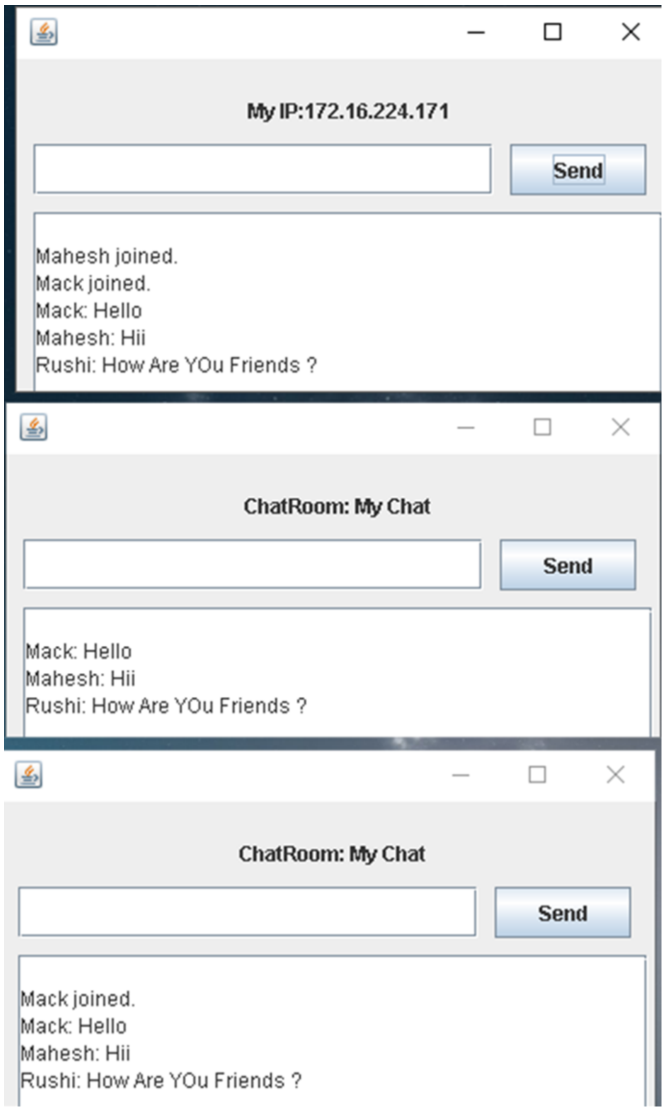
Yes - Create Chat Room

No - Jion a Chat Room


Input

Enter the host ip address

172.16.224.171



```
Mahesh joined.
Mack joined.
Mack: Hello
Mahesh: Hii
Rushi: How Are YOu Friends ?
```

```
Mack: Hello
Mahesh: Hii
Rushi: How Are YOu Friends ?
```

```
Mack joined.
Mack: Hello
Mahesh: Hii
Rushi: How Are YOu Friends ?
```