

**Name : Raj Koyani**

**Reg No:-**

**21MIS1017CN**

**(LAB:3)**

(a) Write a program to implement a simple message transfer from client to server using UDP sockets.

Client Side:

1. The client program prompts the user to enter a message to be sent to the server.
2. The client creates a UDP socket for communication.
3. The client converts the message to bytes and creates a UDP packet containing the message and the server's address and port.
4. The client sends the UDP packet to the server using the UDP socket.

Server Side:

5. The server program creates a UDP socket and binds it to a specific port to listen for incoming messages.
6. The server waits to receive a UDP packet from the client.
7. When a UDP packet is received, the server extracts the message from the packet.
8. The server processes the received message as needed (e.g., displays it, performs operations, etc.).
9. The server can optionally send a response message back to the client.
10. The server and client can continue the communication loop until a termination condition is met.

Code :

Server:

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

public class serverudp {
    public static void main(String[] args) {
        try {
            // Create a DatagramSocket and bind it to a specific port
            DatagramSocket serverSocket = new DatagramSocket(1234);
```

```

        byte[] receiveBuffer = new byte[1024];

        while (true) {
            // Create a DatagramPacket to receive data from the client
            DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer, receiveBuffer.length);

            // Receive data from the client
            serverSocket.receive(receivePacket);

            // Extract the data from the received packet
            String clientMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());

            System.out.println("Message from client: " + clientMessage);

            // Process the received message (e.g., send a response back to
the client)

            // Convert the response message to bytes
            String serverMessage = "Hello, client!";
            byte[] sendBuffer = serverMessage.getBytes();

            // Get the client's address and port from the received packet
            String clientAddress =
receivePacket.getAddress().getHostAddress();
            int clientPort = receivePacket.getPort();

            // Create a DatagramPacket to send the response back to the
client
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length,
                receivePacket.getAddress(), receivePacket.getPort());

            // Send the response back to the client
            serverSocket.send(sendPacket);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

Client:

```

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

```

```
import java.net.DatagramSocket;
import java.net.InetAddress;

public class clientudp {
    public static void main(String[] args) {
        try {
            // Create a DatagramSocket
            DatagramSocket clientSocket = new DatagramSocket();

            // Get the server's address
            InetAddress serverAddress = InetAddress.getByName("localhost");

            // Convert the message to bytes
            String message = "Hello, server!";
            byte[] sendBuffer = message.getBytes();

            // Create a DatagramPacket to send the message to the server
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, serverAddress, 1234);

            // Send the message to the server
            clientSocket.send(sendPacket);

            byte[] receiveBuffer = new byte[1024];

            // Create a DatagramPacket to receive the response from the server
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);

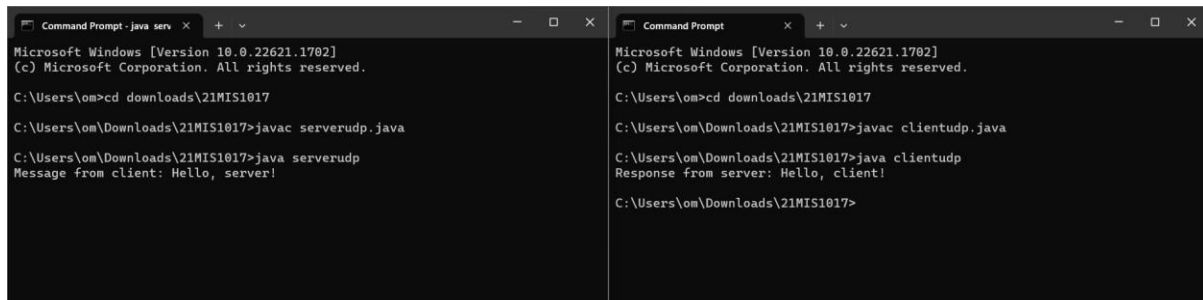
            // Receive the response from the server
            clientSocket.receive(receivePacket);

            // Extract the response message from the received packet
            String serverMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());

            System.out.println("Response from server: " + serverMessage);

            // Close the socket
            clientSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:



```
Command Prompt - java serv x + v
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\om>cd downloads\21MIS1017

C:\Users\om\Downloads\21MIS1017>javac serverudp.java

C:\Users\om\Downloads\21MIS1017>java serverudp
Message from client: Hello, server!

Command Prompt x + v
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\om>cd downloads\21MIS1017

C:\Users\om\Downloads\21MIS1017>javac clientudp.java

C:\Users\om\Downloads\21MIS1017>java clientudp
Response from server: Hello, client!

C:\Users\om\Downloads\21MIS1017>
```

(b) Write a UDP-based server code to get the date of birth of the client and calculate the age as of today. The client has to enter, the year, month, and day of birth.

### Problem Definition:

To write a UDP-based server code to get the date of birth of the client and calculate the age as of today. The client has to enter, the year, month, and day of birth.

### Method:

Server Side:

1. The server initializes and binds to a specific port to listen for incoming UDP packets.
2. The server creates a DatagramSocket to handle UDP communication.

Client Side:

3. The client prompts the user to enter their year, month, and day of birth.
4. The client constructs a UDP packet containing the date of birth information.
5. The client sends the UDP packet to the server using a DatagramSocket.

Server Side:

6. The server receives the UDP packet containing the date of birth information.
7. The server extracts the year, month, and day of birth from the received packet.
8. The server calculates the age based on the current date and the provided date of birth.
9. The server prepares a response message containing the calculated age.
10. The server sends the response message back to the client using a DatagramSocket.

Client Side:

11. The client receives the response message from the server.
12. The client displays the received age calculated by the server

Code:

Server:

```
import java.net.*;

public class serverage {
    public static void main(String[] args) {
        try {
            DatagramSocket socket = new DatagramSocket(5000);

            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);
            socket.receive(receivePacket);

            String clientData = new String(receivePacket.getData());
            String[] dateOfBirth = clientData.trim().split("-");
            int year = Integer.parseInt(dateOfBirth[0]);
            int month = Integer.parseInt(dateOfBirth[1]);
            int day = Integer.parseInt(dateOfBirth[2]);

            // Get the current date
            java.util.Date currentDate = new java.util.Date();
            java.util.Calendar calendar = java.util.Calendar.getInstance();
            calendar.setTime(currentDate);
            int currentYear = calendar.get(java.util.Calendar.YEAR);
            int currentMonth = calendar.get(java.util.Calendar.MONTH) + 1; //
Month is zero-based
            int currentDay = calendar.get(java.util.Calendar.DAY_OF_MONTH);

            // Calculate the age
            int age = currentYear - year;
            if (currentMonth < month || (currentMonth == month && currentDay <
day)) {
                age--; // Subtract 1 if the birthday hasn't occurred yet this
year
            }

            // Send the age back to the client
            InetAddress clientAddress = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();
            String response = "Your age is: " + age;
            byte[] sendBuffer = response.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, clientAddress, clientPort);
            socket.send(sendPacket);

            socket.close();
        }
    }
}
```

```
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Client:

```
import java.io.IOException;  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
  
public class clientudp {  
    public static void main(String[] args) {  
        try {  
            // Create a DatagramSocket  
            DatagramSocket clientSocket = new DatagramSocket();  
  
            // Get the server's address  
            InetAddress serverAddress = InetAddress.getByName("localhost");  
  
            // Convert the message to bytes  
            String message = "Hello, server!";  
            byte[] sendBuffer = message.getBytes();  
  
            // Create a DatagramPacket to send the message to the server  
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,  
sendBuffer.length, serverAddress, 1234);  
  
            // Send the message to the server  
            clientSocket.send(sendPacket);  
  
            byte[] receiveBuffer = new byte[1024];  
  
            // Create a DatagramPacket to receive the response from the server  
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,  
receiveBuffer.length);  
  
            // Receive the response from the server  
            clientSocket.receive(receivePacket);  
  
            // Extract the response message from the received packet  
            String serverMessage = new String(receivePacket.getData(), 0,  
receivePacket.getLength());  
        }  
    }  
}
```

```

        System.out.println("Response from server: " + serverMessage);

        // Close the socket
        clientSocket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

Output:

```

Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\om>cd downloads\21MIS1017

C:\Users\om\Downloads\21MIS1017>javac serverage.java

C:\Users\om\Downloads\21MIS1017>java serverage

C:\Users\om\Downloads\21MIS1017>
Your age is: 33

Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\om>cd downloads\21MIS1017

C:\Users\om\Downloads\21MIS1017>javac clientage.java

C:\Users\om\Downloads\21MIS1017>java clientage

C:\Users\om\Downloads\21MIS1017>
Your age is: 33

C:\Users\om\Downloads\21MIS1017>

```

- c) Write a UDP-based server code to broadcast a message to the nodes in a LAN

### Problem Definition:

To write a Java Code for implementation of server code to broadcast a message to the nodes in a LAN.

### Method:

1. The server initializes and listens on a specific port to accept incoming connections.
2. Clients within the LAN establish connections to the server by specifying its IP address and port number.
3. The server keeps track of the connected clients or maintains a list of their socket connections.
4. When a client sends a message to the server, the server stores the message or the latest received message.
5. The server iterates through the list of connected clients.
6. For each client, the server sends the stored message or the new message using the client's socket connection.
7. Clients continuously listen for messages from the server using their respective socket connections.

8. When a message is received, the client can process or display it to the user.
9. This message broadcasting process continues until a termination condition is met.
10. Upon termination, the server closes its server socket, and each client closes its socket connection.

Code:

Server:

```
import java.net.*;

public class servercast {
    public static void main(String[] args) {
        try {
            int port = 5000;

            // Create a socket with a broadcast address
            DatagramSocket socket = new DatagramSocket();
            socket.setBroadcast(true);

            // Message to be broadcasted
            String message = "Hello, LAN nodes!";
            byte[] sendBuffer = message.getBytes();

            // Create a packet with the broadcast address and port
            InetAddress broadcastAddress =
InetAddress.getByName("255.255.255.255");
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, broadcastAddress, port);

            // Send the packet to the broadcast address
            socket.send(sendPacket);

            System.out.println("Message broadcasted successfully.");

            socket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Client:

```
import java.net.*;

public class clientcast {
```



```

public static void main(String[] args) {
    try {
        int port = 5000;

        // Create a socket to listen for broadcast messages
        DatagramSocket socket = new DatagramSocket(port);
        socket.setBroadcast(true);

        byte[] receiveBuffer = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);

        // Listen for incoming broadcast messages
        socket.receive(receivePacket);

        String receivedMessage = new
String(receivePacket.getData()).trim();
        System.out.println("Received message: " + receivedMessage);

        socket.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

Output:

Command Prompt (Left)	Command Prompt (Right)
Microsoft Windows [Version 10.0.22621.1702] (c) Microsoft Corporation. All rights reserved. C:\Users\om>cd downloads\21MIS1017	Microsoft Windows [Version 10.0.22621.1702] (c) Microsoft Corporation. All rights reserved. C:\Users\om>cd downloads\21MIS1017
C:\Users\om\Downloads\21MIS1017>javac clientcast.java	C:\Users\om\Downloads\21MIS1017>javac servercast.java
C:\Users\om\Downloads\21MIS1017>java clientcast	C:\Users\om\Downloads\21MIS1017>java servercast
Received message: Hello, LAN nodes!	Message broadcasted successfully.
C:\Users\om\Downloads\21MIS1017>java clientcast	C:\Users\om\Downloads\21MIS1017>java servercast
Received message: Hello, LAN nodes!	Message broadcasted successfully.
C:\Users\om\Downloads\21MIS1017>	C:\Users\om\Downloads\21MIS1017>

(d) Write a code to implement border gateway protocol (BGP).

**Problem Definition:**

To write a Java Code for implementation of simple Message Transfer using Border Gateway Protocol (BGP).

**Method:**

1. The server program listens for incoming connections on port 5000 using a ServerSocket.
2. When a client connects to the server, the server accepts the connection using the accept() method, which blocks until a connection is established.
3. The server retrieves the client's IP address and prints it.
4. The server sets up input and output streams to communicate with the client.
5. The server receives the message sent by the client using the input stream's read() method.
6. The server can process the received message as needed.
7. The server sends a response message back to the client using the output stream's write() method.
8. The server closes the input/output streams and the client socket to release resources.

**Code:**

```
import java.util.ArrayList;
import java.util.List;

class BGPRoute {
    private String prefix;
    private String nextHop;

    public BGPRoute(String prefix, String nextHop) {
        this.prefix = prefix;
        this.nextHop = nextHop;
    }

    public String getPrefix() {
        return prefix;
    }

    public String getNextHop() {
        return nextHop;
    }
}

class BGPRouter {
    private List<BGPRoute> routingTable;

    public BGPRouter() {
        routingTable = new ArrayList<>();
    }

    public void addRoute(String prefix, String nextHop) {
```

```

        BGPRoute route = new BGPRoute(prefix, nextHop);
        routingTable.add(route);
    }

    public void removeRoute(String prefix) {
        routingTable.removeIf(route -> route.getPrefix().equals(prefix));
    }

    public void printRoutingTable() {
        for (BGPRoute route : routingTable) {
            System.out.println("Prefix: " + route.getPrefix() + ", Next Hop: "
+ route.getNextHop());
        }
    }
}

public class Main {
    public static void main(String[] args) {
        BGPRouter router = new BGPRouter();

        router.addRoute("10.0.0.0/24", "192.168.1.1");
        router.addRoute("192.168.0.0/16", "10.0.0.1");

        router.printRoutingTable();

        router.removeRoute("10.0.0.0/24");

        router.printRoutingTable();
    }
}

```

Output:

```

Command Prompt
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\om>cd downloads\21MIS1017

C:\Users\om\Downloads\21MIS1017>javac Main.java

C:\Users\om\Downloads\21MIS1017>java Main
Prefix: 10.0.0.0/24, Next Hop: 192.168.1.1
Prefix: 192.168.0.0/16, Next Hop: 10.0.0.1
Prefix: 192.168.0.0/16, Next Hop: 10.0.0.1

C:\Users\om\Downloads\21MIS1017>|

```

