Ex No 3	Simulation of Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP)
25/May/2021	

AIM:

To simulate the working principle of Address Resolution Protocol and (ARP) and Reverse Address Resolution Protocol (RARP) in Java

THEORY:

1. **IP Address**
   An IP address is a unique address that identifies a device on the internet or a local network. IP stands for "Internet Protocol," which is the set of rules governing the format of data sent via the internet or local network.

2. **Mac Address**
   MAC Addresses are unique **48-bits** hardware number of a computer, which is embedded into network card (known as **Network Interface Card**) during the time of manufacturing. MAC Address is also known as **Physical Address** of a network device.

3. **ARP – Working Principle**
   - The source broadcasts the Address Resolution Protocol (ARP) request message to the local network
   - The message is received by each device on the LAN since it is a broadcast. Each device compares the Target Protocol Address with its own Protocol Address
   - When the targeted device checks the Target Protocol Address, it will find a match and will generate an Address Resolution Protocol (ARP) reply message.
   - The destination device will update its Address Resolution Protocol (ARP) cache and finally establishes the connection

4. **RARP – Working Principle**
   - Reverse ARP is a networking protocol used by a client machine in a local area network to request its Internet Protocol address (IPv4) from the gateway-router's ARP table. The network administrator creates a table in gateway-router, which is used to map the MAC address to corresponding IP address
   - A special host configured inside the local area network, called as RARP-server is responsible to reply for these kind of broadcast packets. 19IT117 Page 29

- Now the RARP server attempt to find out the entry in IP to MAC address mapping table. If any entry matches in table, RARP server send the response packet to the requesting device along with IP address.

ARP USING TCP

ALGORITHM:

Client

Step 1: Start

Step 2: Create a class named ARPClient

Step 3: public static void Exception

Step 4: create a DataOutputStream and DatainputStream

Step 5: Get the input ip address

Step 6: out flush the input to the server

Step 7: receive and display the output from the server

Step 8: Stop

Server

Step 1: Start

Step 2: Create a class named ARPServer

Step 3: public static void Exception

Step 4: create a DataOutputStream and DatainputStream

Step 5: Read the value from the client and store it in the str

Step 6: create int flag with value 0

Step 7: Declare the value of ip and mac address

Step 8: create a for loop with i=0,i<mac length and i++

step 8.1:if the value of str is equal to ip[i]

Step 8.1.1:Write out the value of mac[i] to the client

Step 8.1.2: flag =1

Step 9: if flag is equal to 0 then give the output as not found

Step 10: Stop

CODING:

CLIENT

```
package lab_rec;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.util.Scanner;
public class ARPClient
{
   public static void main(String[] args) throws Exception
   {
      Scanner sc = new Scanner(System.in);
      String ip = "localhost";
      int port = 4321;
      Socket s = new Socket(ip, port);
      DataOutputStream out = new DataOutputStream(s.getOutputStream());
      DataInputStream in = new DataInputStream(s.getInputStream());
      System.out.println("Enter a logical address : ");
      String num = sc.next();
      out.writeUTF(num);
      out.flush();
      String str =  (String)in.readUTF();
      System.out.println("IP ADDRESS : "+num);
      System.out.println("MAC ADDRESS : "+str);
```

```
        out.close();
        in.close();
        s.close();
    }
}
```

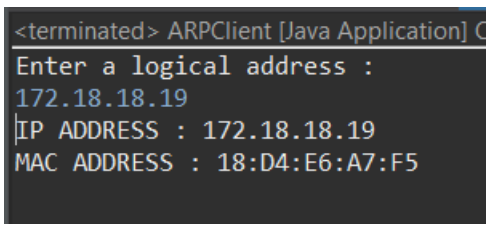
SERVER

```
package lab_rec;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
public class ARPServer {
    public static void main(String[] args) throws Exception {
        int port = 4321;
        ServerSocket ss = new ServerSocket(port);
        Socket s = ss.accept();
        DataInputStream in = new DataInputStream(s.getInputStream());
        DataOutputStream out = new DataOutputStream(s.getOutputStream());
            String str = (String) in.readUTF();
            System.out.println("\nCLIENT : \n");
            System.out.println("MAC address of system with IP " + str + " ?");
            int flag =0;
            String ip[] = {"10.0.3.186","172.17.18.18","172.18.18.19","172.10.18.17"};
            String mac[] =
{"D4:3D:7E:12:A3:D9","00:11:AE:72:E5","18:D4:E6:A7:F5","F4:E3:2K:F4:G3"};
            for (int i = 0; i < mac.length; i++)
            {
```

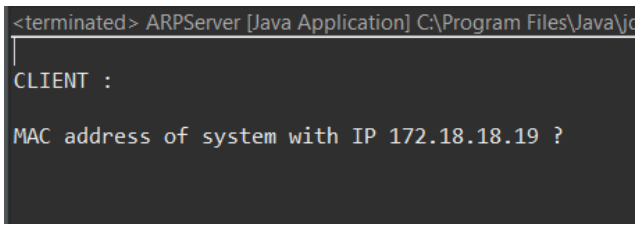
```
        if(str.equals(ip[i]))
        {
                out.writeUTF(mac[i]);
                flag=1;
                break;
        }
       }
       if(flag==0)
       {
        out.writeUTF("NOT FOUND");
       }
    out.flush();
    in.close();
    out.close();
    s.close();

  }
}
```

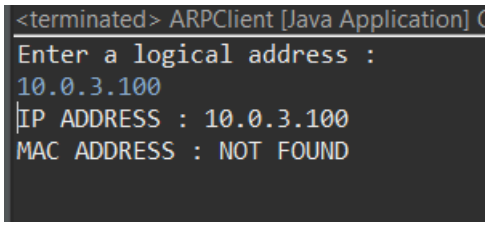
OUTPUT:

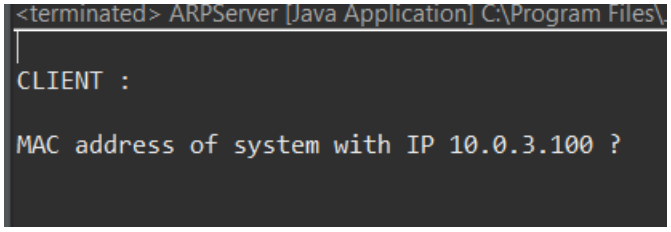
CLIENT



SERVER







ARP USING UDP

ALGORITHM:

Client

Step 1: Start

Step 2: Create a class named ARPClient2

Step 3: public static void Exception

Step 4: Create a DatagramSocket as ds1 and ds2.

Step 5: Get the ip address as input

Step 6: Create a array arr1 to convert the input into bytes

Step 7: Create DatagramPacket dp1 to send the input to the server

Step 8: Create a array arr2 to convert the output bytes into data

Step 9: Print the ip and mac address

Step 10: Stop

Server:

Step 1: Start

Step 2: Create a class named ARPServer2

Step 3: public static void Exception

Step 4: Create a DatagramSocket as ds1 and ds2.

Step 5: read the input from the client

Step 6: Declare the values of ip and mac address

Step 7: create int flag =0 and string str

Step 8: create a for loop with i=0,i<mac length and i++

step 8.1:if the value of s1 is equal to ip[i]

Step 8.1.1:store the mac[i] value in str

Step 8.1.2: flag =1

Step 9: if flag is equal to 0 then give not found as str value

Step 10: send the output to the client

Step 11: Stop

CODING:

CLIENT

```
package lab_rec;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class ARPClient2 {
	public static void main(String[] args) throws Exception {
		int port1=5000,port2=5001;
		Scanner inp=new Scanner(System.in);
		InetAddress ip = InetAddress.getLocalHost();
		DatagramSocket ds1 = new DatagramSocket();
		DatagramSocket ds2 = new DatagramSocket(port2);
		System.out.print("Enter a logical address : ");
		String s1 = inp.next();
		byte arr1[]=s1.getBytes();
		DatagramPacket dp1 = new DatagramPacket(arr1, arr1.length, ip, port1);
		ds1.send(dp1);
		byte arr2[] = new byte[1000];
```

```
                DatagramPacket dp2 = new DatagramPacket(arr2, arr2.length);
                ds2.receive(dp2);
                String s2=new String(arr2);
                System.out.print("IP ADDRESS : " + s1 + "\n");
                System.out.print("MAC ADDRESS : " + s2+ "\n");
        }
}
```

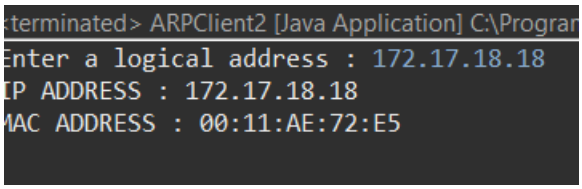
SERVER

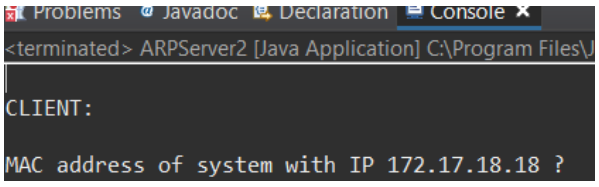
```
package lab_rec;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class ARPServer2 {
        public static void main(String[] args) throws Exception {
                int port1=5000,port2=5001;
                Scanner inp=new Scanner(System.in);
                InetAddress ip = InetAddress.getLocalHost();
                DatagramSocket ds1 = new DatagramSocket(port1);
                DatagramSocket ds2 = new DatagramSocket();
                byte arr1[] = new byte[1000];
                DatagramPacket dp1 = new DatagramPacket(arr1, arr1.length);
                ds1.receive(dp1);
                String s1=new String(arr1);
                System.out.println("\nCLIENT:\n");
                 System.out.println("MAC address of system with IP " + s1.trim() + " ?");
                 int flag=0;
```

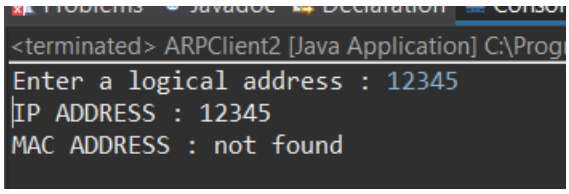
```
String ipadd[] = {"10.0.3.186","172.17.18.18","172.18.18.19","172.10.18.17"};
String mac[] = {"D4:3D:7E:12:A3:D9","00:11:AE:72:E5","18:D4:E6:A7:F5","F4:E3:2K:F4:G3"};
String str="";
for (int i = 0; i < mac.length; i++)
{
  if(ipadd[i].equals(s1.trim()))
  {
        str=mac[i];
        flag=1;
        break;
  }
}
        if(flag==0)
        {
            str="not found";
        }
        byte arr2[]=str.getBytes();
        DatagramPacket dp2 = new DatagramPacket(arr2, arr2.length, ip, port2);
        System.out.println(s1);
        ds2.send(dp2);
  }
}
```

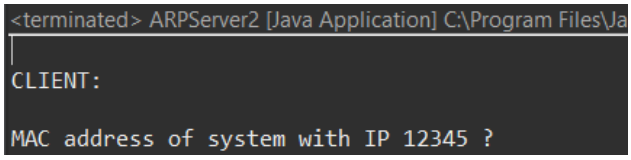
OUTPUT:

CLIENT **SERVER**









RARP USING TCP

ALGORITHM:

Client

Step 1: Start

Step 2: Create a class named RARPClient

Step 3: public static void Exception

Step 4: create a DataOutputStream and DatainputStream

Step 5: Get the input mac address

Step 6: out flush the input to the server

Step 7: receive and display the output from the server

Step 8: Stop

server

Step 1: Start

Step 2: Create a class named RARPServer

Step 3: public static void Exception

Step 4: create a DataOutputStream and DatainputStream

Step 5: Read the value from the client and store it in the str

Step 6: create int flag with value 0

Step 7: Declare the value of ip and mac address

Step 8: create a for loop with i=0,i<ip length and i++

step 8.1:if the value of str is equal to mac[i]

Step 8.1.1:Write out the value of ip[i] to the client

Step 8.1.2: flag =1

Step 9: if flag is equal to 0 then give the output as not found

Step 10: Stop

CODING:

CLIENT

```
package lab_rec;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.util.Scanner;
public class RARPClient
{
   public static void main(String[] args) throws Exception
   {
      Scanner sc = new Scanner(System.in);
      String ip = "localhost";
      int port = 4321;
      Socket s = new Socket(ip, port);
      DataOutputStream out = new DataOutputStream(s.getOutputStream());
      DataInputStream in = new DataInputStream(s.getInputStream());
      System.out.println("Enter a logical address : ");
```

```
        String num = sc.nextLine();
        out.writeUTF(num);
        out.flush();
        String str =  (String)in.readUTF();
        System.out.println("MAC ADDRESS : "+num);
        System.out.println("ID ADDRESS : "+str);
        out.close();
        in.close();
        s.close();
    }
}
```

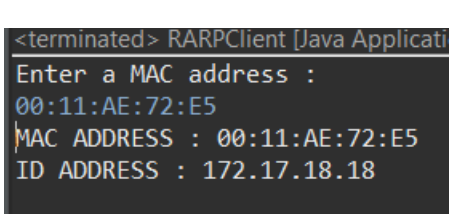
SERVER

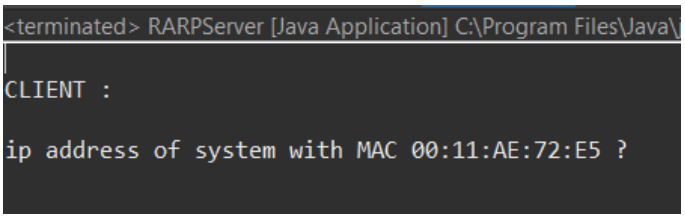
```
package lab_rec;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
public class RARPServer {
    public static void main(String[] args) throws Exception {
        int port = 4321;
        ServerSocket ss = new ServerSocket(port);
        Socket s = ss.accept();
        DataInputStream in = new DataInputStream(s.getInputStream());
        DataOutputStream out = new DataOutputStream(s.getOutputStream());
            String str = (String) in.readUTF();
            System.out.println("\nCLIENT : \n");
            System.out.println("ip address of system with IP " + str + " ?");
```

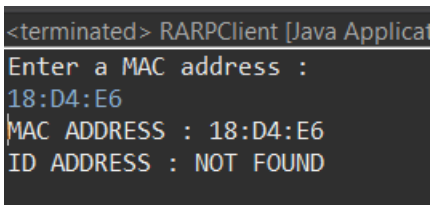
```
        int flag = 0;
        String ip[] = {"10.0.3.186","172.17.18.18","172.18.18.19","172.10.18.17"};
        String mac[] = {"D4:3D:7E:12:A3:D9","00:11:AE:72:E5","18:D4:E6:A7:F5","F4:E3:2K:F4:G3"};
        for (int i = 0; i < mac.length; i++)
        {
         if(mac[i].equals(str.trim()))
         {
                out.writeUTF(ip[i]);
                flag=1;
                break;
         }
        }
        if(flag==0)
         {
                out.writeUTF("NOT FOUND");
         }
      out.flush();
      in.close();
      out.close();
      s.close();
   }
}
```

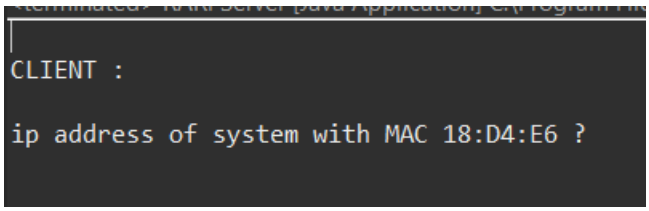
OUTPUT:

CLIENT **SERVER**









RARP USING UDP

ALGORITHM:

Client

Step 1: Start

Step 2: Create a class named RARPClient2

Step 3: public static void Exception

Step 4: Create a DatagramSocket as ds1 and ds2.

Step 5: Get the mac address as input

Step 6: Create a array arr1 to convert the input into bytes

Step 7: Create DatagramPacket dp1 to send the input to the server

Step 8: Create a array arr2 to convert the output bytes into data

Step 9: Print the ip and mac address

Step 10: Stop

Server:

Step 1: Start

Step 2: Create a class named RARPServer2

Step 3: public static void Exception

Step 4: Create a DatagramSocket as ds1 and ds2.

Step 5: read the input from the client

Step 6: Declare the values of ip and mac address

Step 7: create int flag =0 and string str

Step 8: create a for loop with i=0,i<ip length and i++

step 8.1:if the value of s1 is equal to mac[i]

Step 8.1.1:store the ip[i] value in str

Step 8.1.2: flag =1

Step 9: if flag is equal to 0 then give not found as str value

Step 10: send the output to the client

Step 11: Stop

CODING:

CLIENT

```
package lab_rec;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class RARPClient2 {
	public static void main(String[] args) throws Exception {
		int port1=5000,port2=5001;
		Scanner inp=new Scanner(System.in);
		InetAddress ip = InetAddress.getLocalHost();
		DatagramSocket ds1 = new DatagramSocket();
		DatagramSocket ds2 = new DatagramSocket(port2);
		System.out.print("Enter a MAC address : ");
		String s1 = inp.next();
```

```
                byte arr1[]=s1.getBytes();
                DatagramPacket dp1 = new DatagramPacket(arr1, arr1.length, ip, port1);
                ds1.send(dp1);
                byte arr2[] = new byte[1000];
                DatagramPacket dp2 = new DatagramPacket(arr2, arr2.length);
                ds2.receive(dp2);
                String s2=new String(arr2);
                System.out.print("MAC ADDRESS : " + s1 + "\n");
                 System.out.print("IP ADDRESS : " + s2+ "\n");
        }
}
```

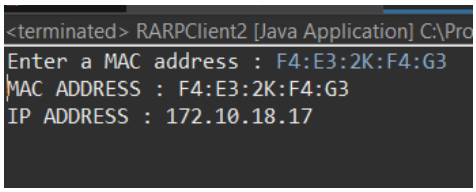
SERVER

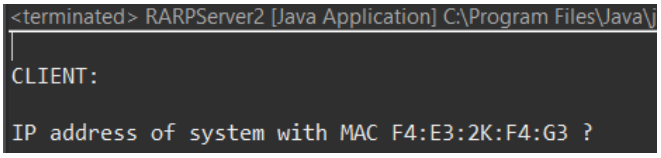
```
package lab_rec;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class RARPServer2 {
        public static void main(String[] args) throws Exception {
                int port1=5000,port2=5001;
                Scanner inp=new Scanner(System.in);
                InetAddress ip = InetAddress.getLocalHost();
                DatagramSocket ds1 = new DatagramSocket(port1);
                DatagramSocket ds2 = new DatagramSocket();
                byte arr1[] = new byte[1000];
                DatagramPacket dp1 = new DatagramPacket(arr1, arr1.length);
                ds1.receive(dp1);
```

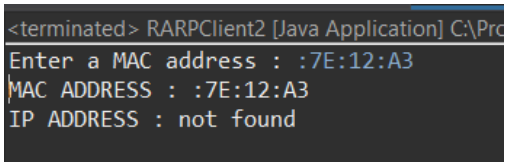
```
                String s1=new String(arr1);
                System.out.println("\nCLIENT:\n");
                 System.out.println("IP address of system with MAC " + s1.trim() + " ?");
                 int flag=0;
                String ipadd[] = {"10.0.3.186","172.17.18.18","172.18.18.19","172.10.18.17"};
    String mac[] = {"D4:3D:7E:12:A3:D9","00:11:AE:72:E5","18:D4:E6:A7:F5","F4:E3:2K:F4:G3"};
    String str="";
    for (int i = 0; i < ipadd.length; i++)
    {
       if(mac[i].equals(s1.trim()))
       {
                str=ipadd[i];
                flag=1;
                break;
       }
    }
                if(flag==0)
                {
                        str="not found";
                }
                byte arr2[]=str.getBytes();
                DatagramPacket dp2 = new DatagramPacket(arr2, arr2.length, ip, port2);
                System.out.println(s1);
                ds2.send(dp2);
       } }
```

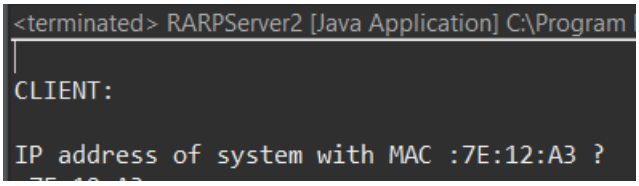
OUTPUT:

CLIENT **SERVER**









KEY TAKEAWAYS:

- ARP and RARP
- Database using java
- Basic sql commands
- Different types of IP addresses

RESULT:

the working principle of Address Resolution Protocol and (ARP) and Reverse Address Resolution Protocol (RARP) in Java has been successfully implemented

Rubric for Evaluation

Parameter	Max Marks	Marks Obtained
Complexity of the Application chosen	3	
Uniqueness of the Code	10	
Use of Comment lines and standard coding practices	2	
Viva	5	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	
Signature of the faculty with Date		