Sri Eshwar
College of Engineering
Coimbatore | Tamilnadu
An Autonomous Institution
Affiliated to Anna University, Chennai

LABORATORY RECORD

R19IT351

DATA COMMUNICATION

AND

NETWORKING LABORATORY

Sri Eshwar College of Engineering
Coimbatore | Tamilnadu
An Autonomous Institution
Affiliated to Anna University, Chennai

BONAFIDECERTIFICATE

Certified that this is the Bonafide record of work done by, Mr/Ms ……

Register Number ……... of **III year B.Tech Information Technology** in **R19IT351– Data Communication and Networking Laboratory** during **V Semester** of the academic year **2024-2025 (ODD Semester)**

Signature of Faculty In-Charge **Head of the Department**

Submitted for the practical examination scheduled on…….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

Experiment No. &Title: 1) Configure a simple peer to peer network connection with three systems using any topology and understand its connectivity by Standard Cabling, Cross Cabling

Date : 20/08/2024

I. General Objective (Aim)

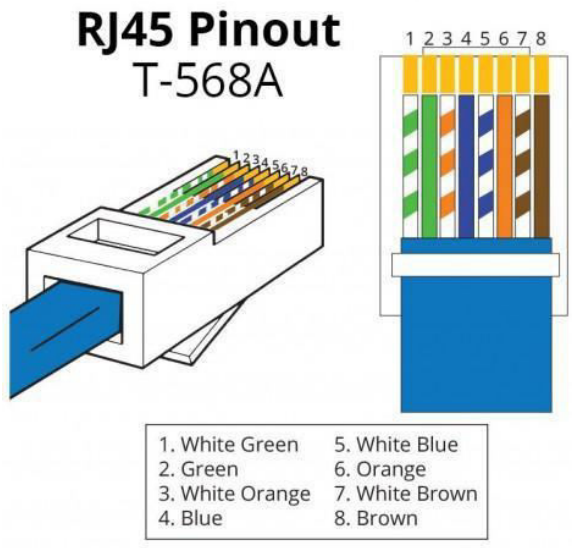
To study about the simple peer to peer network connection with three systems using any topology and understand its connectivity by Standard Cabling, Cross Cabling

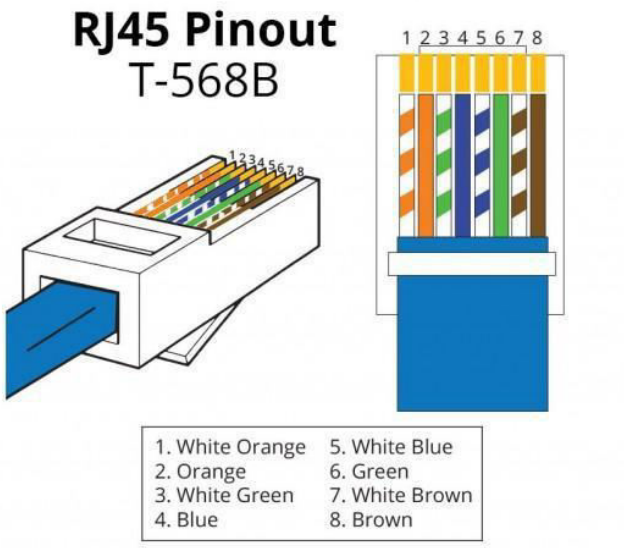
II. Conduct of Experiment

➢ **List of Equipment / Software(s)**

Crimping tool
RJ45 Connector
RJ45 Cable
Cable tester

➢ **Design / Circuit diagrams**





III. Procedure

RJ-45 cable

RJ stands for Registered Jacks. These are used in telephone and data jack wiring registered with FCC. RJ-11 is a 6-position, 4-conductor jack used in telephone wiring, and RJ-45 is a 8-position, 8-conductor jack used in 10BaseT and 100BaseT Ethernet wiring.

RJ45 color-coded scheme

RJ-45 conductor data cable contains 4 pairs of wires each consists of a solid colored wire and a strip of the same color. There are two wiring standards for RJ-45 wiring: T-568A and T-568B. Although there are 4 pairs of wires, 10BaseT/100BaseT Ethernet uses only 2 pairs: Orange and Green. The other two colors (blue and brown) may be used for a second Ethernet line or for phone connections. The two wiring standards are used to create a cross-over cable (T-568A on one end, and T-568B on the other end), or a straight-through cable (T-568B or T-568A on both ends).

Cable crimping

Crimping is joining two pieces of material (usually a wire and a metal plate) by deforming one or both of them to hold the other. The bend or deformity is called the crimp.

Process

Typically, the metals are joined together via a special connector. Stripped wire (often stranded) is inserted through the correctly sized opening of the connector, and a crimper is used to tightly squeeze the opening against the wire. Depending on the type of connector used, it may be attached to a metal plate by a separate screw or bolt or it could be simply screwed on using the connector itself to make the attachment like an F connector.

Uses

Crimping is most extensively used in metalworking. Crimping is commonly used to in their cartridge cases, for rapid but lasting electrical connections, securing lids on metal food cans, and many other applications.

Standard cabling

This type of cable is used in structured cabling for computer networks such as Ethernet over twisted pair. The cable standard provides performance of up to 100 MHz and is suitable for 10BASE-T, 100BASE-TX (Fast Ethernet), and 1000BASE-T (Gigabit Ethernet).

Cross cabling

A crossover cable connects two devices of the same type, for example DTE-DTE or DCE-DCE, usually connected asymmetrically (DTE-DCE), by a modified called a cross link. Such distinction of devices was introduced by IBM.

The crossing wires in a cable or in a connector adaptor allows:

a) Connecting two devices directly, output of one to input of the other,
b) Letting two terminals (DTE) devices communicate without an interconnecting hub knot, i.e. PCs.
c) Linking two or more hubs, switches or routers (DCE) together, possibly to work as one wider device.

Establishing LAN connections

1. Verify that you have a crossover Ethernet cable.
2. Plug each end of the Ethernet cable into an Ethernet network port on each computer to connect the computers together with the cable.
3. Go to any one of your computers, and click on the "Start" menu.
4. Select "Control Panel," then type "network" into the search box provided to you within Control Panel.
5. Select "Network and Sharing Center" from the options displayed in the window.
6. Select and open the icon labeled "Unidentified network" from the network map at the top of the Network and Sharing Center window.
7. Click on the message that prompts to change the network discovery and file sharing settings, then click on the option that reads, "Turn on network discovery and file sharing.

IV. Results (Simulation/Hardware output, Tabulation and Graph)

Thus the RJ45 cable and its color coding scheme has been studied.

Experiment No. &Title: 2 Write a program to implement the client server communication using socket connection

Date : 27/08/2024

I. General Objective (Aim)

To implement a program for communicating the client and server using socket connection.

II. Conduct of Experiment

- **Algorithm or Programming / Process Mapping**

Step 1: Start the Program.
Step 2: Create the Server Socket.
Step 3: Create the Client Socket.
Step 4: Initiate the request for connection from the client side.
Step 5: Server accepts the request.
Step 6: Start communication by exchanging messages between client and server
Step 7: Client sends the directory path name to the Server.
Step 8: Server lists the entire file inside the directory.
Step 9: Stop the Program.

Program:

Client program

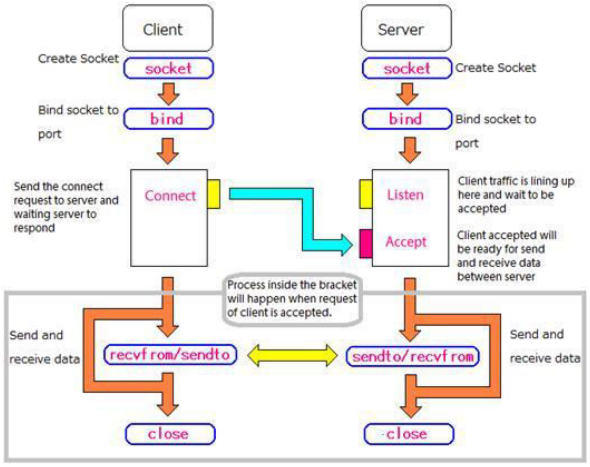
```
import java.io.*;
import java.net.*;
public class tcpc
{
    public static void main(String args[])throws
    Exception {
            int port=500;
```

```
            Socket s=new Socket(InetAddress.getLocalHost(),port);
            BuffereReader dis=new BufferedReader(new
               InputStreamReader(s.getInputStream()));

            String str;

            while((str=dis.readLine())!= null)
            {
                    System.out.println(str);
            }
            dis.close();
            s.close();
    }
}
```

Server program

```
import java.net.*;
import java.io.*;
public class tcps
{   public static void main(String args[]) throws Exception
    {
    int port=500;
    System.out.println("\n Waiting for connection
    ..."); ServerSocket ss=new ServerSocket(500);
    Socket s=ss.accept();
    System.out.println("\n Connection accepted ...");
    System.out.println("\n Processing the files in the directory " + args[0] +
    "..."); File directory=new File(args[0]);
    PrintWriter out=new
    PrintWriter(s.getOutputStream()); String[]
    fileList=directory.list();
    for(int i=0;i<fileList.length;i++)
    out.println(fileList[i]);
    System.out.println("\n List sent to the client...");
    out.close();
    s.close();
    }}
```

➢ **Design / Circuit diagrams**



III. Procedure

a) Implement Server side and Client side program separately
b) Start server side program first
c) Start client side program next
d) Send and receive data

IV. Results (Simulation/Hardware output, Tabulation and Graph)

Output:

Client side:

```
E:\jdk1.3\bin>javac tcpc.java
E:\jdk1.3\bin>java tcpc
ex1.html
ex1.html.doc
E:\jdk1.3\bin>
```

Server side:

```
E:\jdk1.3\bin>javac tcps.java
E:\jdk1.3\bin>java tcps e:\vas
 Waiting for connection...
 Connection accepted...
 Processing the files in the directory e:\vas...
 List sent to the client...
E:\jdk1.3\bin>
```

Thus the program for communicating the client and server using socket connection is done and output is verified successfully.

Experiment No. &Title: 3) write a program for transferring a file between nodes in a network

Date : 3/09/2024

I. General Objective (Aim)

To implement a program for transferring a file between nodes in a network using File Transfer Protocol.

II. Conduct of Experiment

- **Algorithm or Programming / Process Mapping**

Step 1: Start the Program.
Step 2: Create the Server Socket.
Step 3: Create the Client Socket.
Step 4: Client initiates the request for establishing connection.
Step 5: Server accepts the request.
Step 6: Client sends the name of the file to be transferred to the Server.
Step 7: The transferred file is stored in the default file abc.txt.
Step 8: Messages are displayed in the corresponding windows indicating the successful transfer of file.
Step 9: Stop the Program.

Program:

Client program

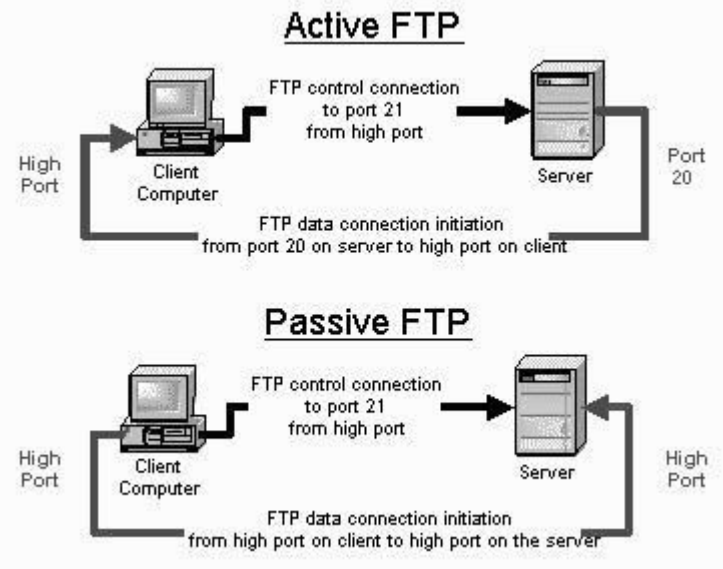
```
import java.io.*;
import java.net.*;
public class ftpc
{
        public static void main(String args[])throws
        IOException {
                Socket s=null;
                DataInputStream si=null;
                s=new Socket(InetAddress.getLocalHost(),55555);
                si=new DataInputStream(s.getInputStream());
                BufferedReader inp=new BufferedReader(new
                InputStreamReader(System.in)); //DataInputStream inp=new
```

```
                DataInputStream(System.in); DataOutputStream so=new
                DataOutputStream(s.getOutputStream()); String str;
                System.out.println("\n Enter the filename(path) : ");
                str=inp.readLine();
                so.writeBytes(str);
                so.writeBytes("\n");
                FileOutputStream fos=new FileOutputStream("abc.txt");
                int str1;
                while((str1=si.read())!=-1)
                fos.write((char)str1);
                System.out.println("File received successfully");
                si.close();
        }
}
```

Server program

```
import java.io.*;
import java.net.*;
class ftps
{
        public static void main(String args[])throws IOException
        {

                Socket s=null;
                ServerSocket ss=null;
                DataOutputStream sso=null;
                ss=new ServerSocket(55555);
                s=ss.accept();
                sso=new DataOutputStream(s.getOutputStream());
                BufferedReader sin=new BufferedReader(new
                InputStreamReader(s.getInputStream())); String s1;
                s1=sin.readLine();
                FileInputStream fos=new FileInputStream(s1);
                int str;
                while ((str=fos.read())!=-1)
                sso.writeBytes(" "+(char)str);
                System.out.println("File has been sent successfully");
                sso.close();
                s.close();
        }
}
```

➢ **Design / Circuit diagrams**



III. Procedure

a) Implement Server side and Client side program separately
b) Start server side program first
c) Start client side program next
d) Send and receive files using FTP

IV. Results (Simulation/Hardware output, Tabulation and Graph)

Output:

Client side:

D:\java>javac ftpc.java
D:\java>java ftpc
Enter the filename (path):
C:\first.txt
File received successfully

Server side:

D:\java>javac ftps.java
D:\java>java ftps
File has been sent successfully
first.txt : This is the test file for file transfer protocol.

That's all this file contains.

Thus a file can be transferred from client to server successfully and display was implemented using Java program.

Experiment No. &Title: 4) Write a program to Simulate Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP).

Date : 10/09/2024

I. General Objective (Aim)

To simulate Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP)

II. Conduct of Experiment

- **Algorithm or Programming / Process Mapping**

ARP Client side

Step 1: Start the Program.
Step 2: Establish a connection between the Client and Server. Socket ss=new Socket(InetAddress.getLocalHost(),1100);
Step 3: Create instance output stream writer PrintWriter ps=new PrintWriter(s.getOutputStream(),true);
Step 4: Get the IP Address to resolve its physical address.
Step 5: Send the IPAddress to its output Stream.ps.println(ip);
Step 6: Print the Physical Address received from the server.
Step 7: Stop the program.

ARP Server Side

Step 1: Start the Program.
Step 2: Accept the connection request by the client. ServerSocket ss=new ServerSocket(2000);Socket s=ss.accept();
Step 3: Get the IPaddress from its inputstream. BufferedReader br1=new BufferedReader(newInputStreamReader(s.getInputStream())); ip=br1.readLine();
Step 4: During runtime execute the processRuntime r=Runtime.getRuntime(); Process p=r.exec("arp -a "+ip);
Step 5: Send the Physical Address to the client.
Step 6: Stop the program.

Program – ARP Client

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

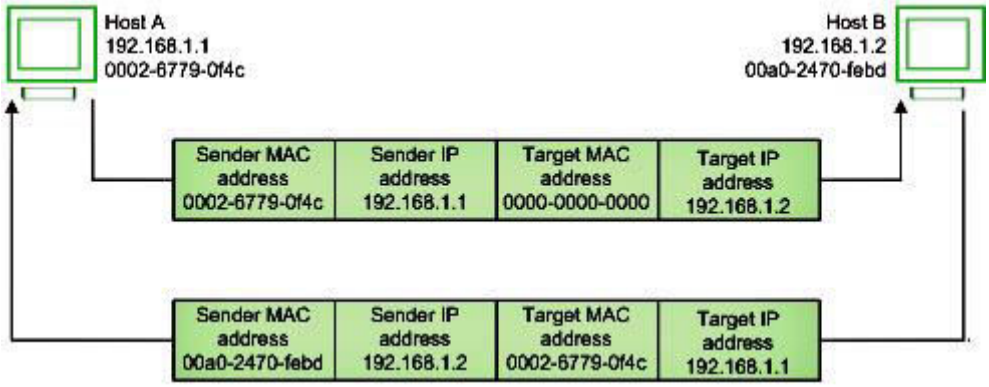
```
public class ArpClient {
   public static void main(String[] args) {
      String serverAddress = "127.0.0.1"; // Change to the ARP server's IP address
      int port = 9876; // Same port as the ARP server

      try {
         DatagramSocket socket = new DatagramSocket();
         String arpRequest = "ARP Request: What is my MAC address?";
         byte[] buffer = arpRequest.getBytes();

         // Send ARP request to the server
         InetAddress address = InetAddress.getByName(serverAddress);
         DatagramPacket packet = new DatagramPacket(buffer, buffer.length,
address, port);
         socket.send(packet);
         System.out.println("Sent ARP request: " + arpRequest);

         // Receive ARP response from the server
         byte[] responseBuffer = new byte[1024];
         DatagramPacket responsePacket = new DatagramPacket(responseBuffer,
responseBuffer.length);
         socket.receive(responsePacket);
         String response = new String(responsePacket.getData(), 0,
responsePacket.getLength());
         System.out.println("Received: " + response);

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

Program – ARP Server

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

public class ArpServer {

   public static void main(String[] args) {
      int port = 9876; // Port number for the ARP server
      try {
         DatagramSocket socket = new DatagramSocket(port);
         System.out.println("ARP Server is running on port " + port);

         while (true) {
            // Buffer for incoming data
            byte[] buffer = new byte[1024];

            // Create a packet to receive data
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
            socket.receive(packet); // Wait for incoming packets

            // Process the received ARP request
```

```
                String received = new String(packet.getData(), 0, packet.getLength());
                System.out.println("Received ARP request: " + received);

                // Simulate an ARP response
                String macAddress = "00:1A:2B:3C:4D:5E"; // Example MAC address
                String ipAddress = packet.getAddress().getHostAddress();
                String arpResponse = "ARP Response: IP: " + ipAddress + " is at " +
macAddress;

                // Send ARP response back to the client
                InetAddress clientAddress = packet.getAddress();
                int clientPort = packet.getPort();
                DatagramPacket responsePacket = new
DatagramPacket(arpResponse.getBytes(), arpResponse.length(), clientAddress,
clientPort);
                socket.send(responsePacket);
                System.out.println("Sent ARP response: " + arpResponse);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

➢ **Design / Circuit diagrams**



Program – RARP Server

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

public class RARPServer {
   private static final int RARP_PORT = 102; // RARP typically uses port 102
   private static final String TARGET_MAC = "00:11:22:33:44:55";
   private static final String TARGET_IP = "192.168.1.10";

   public static void main(String[] args) {
      try {
         DatagramSocket socket = new DatagramSocket(RARP_PORT);
         System.out.println("RARP Server is running on port " + RARP_PORT);

         byte[] receiveBuffer = new byte[1024];

         while (true) {
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);
            socket.receive(receivePacket);
            String request = new String(receivePacket.getData(), 0,
receivePacket.getLength());

            System.out.println("Received RARP request: " + request);

            String macAddress = request.trim();

            if (TARGET_MAC.equals(macAddress)) {
               byte[] responseBuffer = TARGET_IP.getBytes();
               InetAddress clientAddress = receivePacket.getAddress();
               int clientPort = receivePacket.getPort();

               DatagramPacket responsePacket = new
DatagramPacket(responseBuffer, responseBuffer.length, clientAddress,
clientPort);
               socket.send(responsePacket);
               System.out.println("Sent RARP response: " + TARGET_IP);
            }
         }
      } catch (IOException e) {
         System.err.println("Error in RARP Server: " + e.getMessage());
      }
   }
}
```

Program – RARP Client

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

public class RARPClient {
   public static void main(String[] args) {
            String serverIp = "192.168.1.100"; // Replace with your RARP server's IP
      int port = 2048; // Default RARP port

      byte[] clientMacAddress = new byte[] { (byte) 0x00, (byte) 0x1A, (byte) 0x2B,
(byte) 0x3C, (byte) 0x4D, (byte) 0x5E };

      try {
         DatagramSocket socket = new DatagramSocket();
         InetAddress serverAddress = InetAddress.getByName(serverIp);
         byte[] requestData = createRARPRequest(clientMacAddress);
         DatagramPacket requestPacket = new DatagramPacket(requestData,
requestData.length, serverAddress, port);
         socket.send(requestPacket);

         byte[] responseData = new byte[1024]; // Buffer for response
         DatagramPacket responsePacket = new DatagramPacket(responseData,
responseData.length);
         socket.receive(responsePacket);

         String ipAddress = extractRARPResponse(responseData);
         System.out.println("Received IP Address: " + ipAddress);
         socket.close();

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

   private static byte[] createRARPRequest(byte[] clientMacAddress) {
      byte[] request = new byte[6 + clientMacAddress.length];
      System.arraycopy(clientMacAddress, 0, request, 6, clientMacAddress.length);
      return request;
   }

   private static String extractRARPResponse(byte[] responseData) {
      StringBuilder ipAddress = new StringBuilder();
      for (int i = 0; i < 4; i++) {
         if (i > 0) {
            ipAddress.append('.');
```

```
            }
            ipAddress.append(responseData[i] & 0xFF);
        }
        return ipAddress.toString();
    }
}
```

III. Procedure

a) Implement Server side and Client side program separately
b) Start server side program first
c) Start client side program next
d) Send and receive data

IV. Results (Simulation/Hardware output, Tabulation and Graph)

Output(ARP):

```
C:\Networking Programs>java ArpServer
C:\Networking Programs>java ArpClient
Enter the IPADDRESS:
192.168.11.58
ARP From Server::
Interface: 192.168.11.57 on Interface 0x1000003
Internet Address Physical Address Type
192.168.11.58 00-14-85-67-11-84 dynamic
```

Output(RARP):

```
I:\ex>java Serverrarp12
 I:\ex>java Clientrarp12
Enter the Physical address (MAC):
6A:08:AA:C2
The Logical Address is(IP): 165.165.80.80
```

Thus ARP and RARP successfully simulated using Java program.

Experiment No. &Title: 5) Write a program for downloading a file from HTTP server.
Date : 17/09/2024

I. General Objective (Aim)

To Write a program for downloading a file from HTTP server

II. Conduct of Experiment

- **Algorithm or Programming / Process Mapping**

Client side

Step 1: Start the Program.
Step 2: Create a TCP connection to the server's IP and port (ie port 80 for HTTP)
Step 3: Send a GET request to the server in the format
Step 4: Receive response and Check the response code in the HTTP headers
Step 5: Read the content Download the file and write it to a file on the local disk
Step 6: Close the TCP connection after the download completes
Step 7: Stop the program.

Server Side

Step 1: Start the Program.
Step 2: Start an HTTP server to listen for incoming client requests on port 80
Step 3: Accept a client's TCP connection
Step 4: Parse the HTTP request GET method and requested file path
Step 5: If the file exists,
respond with HTTP headers and the file content in chunks.
If the file does not exist,
respond with 404 Not Found
Step 6: Close the TCP connection after responding to the client

Program - HTTP Client

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

public class SimpleHttpClient {

    public static void main(String[] args) {
        String fileURL = "http://localhost:8000/download";
        String savePath = "downloaded_sample.txt";

        try {
            URL url = new URL(fileURL);
            HttpURLConnection httpConn = (HttpURLConnection) url.openConnection();
            int responseCode = httpConn.getResponseCode();

            // Check for successful response code
            if (responseCode == HttpURLConnection.HTTP_OK) {
                InputStream inputStream = httpConn.getInputStream();
                FileOutputStream outputStream = new FileOutputStream(savePath);

                byte[] buffer = new byte[1024];
                int bytesRead;

                System.out.println("Downloading file...");

                while ((bytesRead = inputStream.read(buffer)) != -1) {
                    outputStream.write(buffer, 0, bytesRead);
                }

                outputStream.close();
                inputStream.close();

                System.out.println("File downloaded to " + savePath);
            } else {
                System.out.println("No file to download. Server replied HTTP code: " + responseCode);
            }
            httpConn.disconnect();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Program - Server

```
import com.sun.net.httpserver.HttpServer;
import com.sun.net.httpserver.HttpHandler;
import com.sun.net.httpserver.HttpExchange;

import java.io.*;
import java.net.InetSocketAddress;

public class SimpleHttpServer {

    public static void main(String[] args) throws IOException {
        HttpServer server = HttpServer.create(new InetSocketAddress(8000), 0);
        server.createContext("/download", new FileHandler());
        server.setExecutor(null); // creates a default executor
        System.out.println("HTTP server started on port 8000. Access file at
http://localhost:8000/download");
        server.start();
    }
```

```
    static class FileHandler implements HttpHandler {
        @Override
        public void handle(HttpExchange exchange) throws IOException {
            File file = new File("sample.txt"); // Path to the file you want to serve
            if (!file.exists()) {
                String response = "File not found";
                exchange.sendResponseHeaders(404, response.length());
                OutputStream os = exchange.getResponseBody();
                os.write(response.getBytes());
                os.close();
            } else {
                exchange.sendResponseHeaders(200, file.length());
                OutputStream os = exchange.getResponseBody();
                FileInputStream fis = new FileInputStream(file);
                byte[] buffer = new byte[1024];
                int count;
                while ((count = fis.read(buffer)) != -1) {
                    os.write(buffer, 0, count);
                }
                fis.close();
                os.close();
            }
        }
    }
}
```

III. Procedure

a) Implement Server side and Client side program separately
b) Start server side program first
c) Start client side program next
d) Send and receive data

IV. Results (Simulation/Hardware output, Tabulation and Graph)

Output:

Server side:

Server started on port 8080

Client side :

File downloaded successfully to downloaded_file.txt

Thus program for downloading a file from HTTP server implemented successfully.

Experiment No. &Title: 6) Implement star topology for connecting 8 machines for an organization in packet tracer.

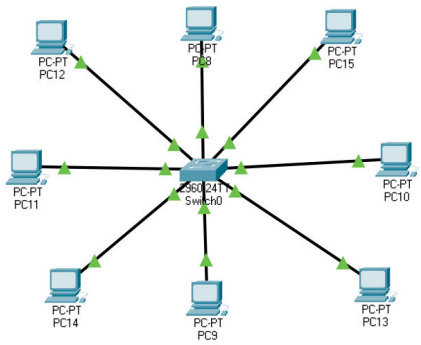
Date : 01/10/2024

I. General Objective (Aim)

To Implement star topology for connecting 100 machines for an organization in packet tracer.

II. Conduct of Experiment

Topology



IP TABLE

Device	Interface	IP Address	Subnetmask	Default Gateway
Host 1	Fast Ethernet	192.168.1.1	255.255.255.0	192.168.1.254
Host 2 ..	Fast Ethernet	192.168.1.2	255.255.255.0	192.168.1.254
…Host 8	Fast Ethernet	192.168.1.8	255.255.255.0	192.168.1.254

Steps involved in configuration

Step 1. Configure system 1 to system 8 with a host name of hostX. Configure the appropriate IPaddresses on the interfaces; refer to the IP Addresses table.

Step 2. Configure Switch with a host name of Switch0.

Step 3. Make UTP connection between hosts and switch.

Step 6. Verify your configuration by pinging from Host1 to Host8. The ping from one host to other should be successful.

III. Result

Thus the star topology for connecting 8 machines for an organization in packet tracer is tested and verified.

Experiment No. &Title: 7) Implement appropriate IP Subnetting for the following Scenario (Departments)
1. Purchase 12 PCs,
2.Marketing 6 PCs,
3.Production 18 PCs and simulate in packet tracer.

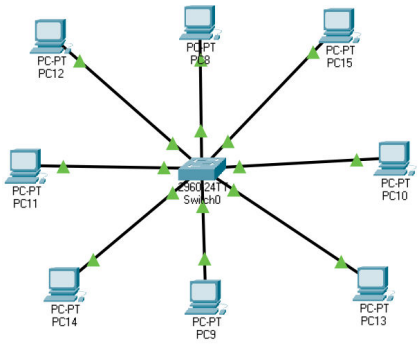
Date : 08/10/2024

I. General Objective (Aim)

To Implement realtime subnetting for given departments and simulate in packet tracer.

II. Conduct of Experiment

Topology



IP TABLE

Departments	Count	IP Range	Usable IPs	Un used IPs	Subnetmask
Purchase	12	192.168.1.1 – 192.168.1.14	14	2	255.255.255.240
Marketing	6	192.168.1.65 - 192.168.1.70	6	0	255.255.255.248
Production	18	192.168.1.33 - 192.168.1.62	30	12	255.255.255.224

Steps involved in configuration

Step 1. Configure systems host name with respect to department. Configure the appropriate IP addresses on the interfaces; refer to the IP Addresses table.

Step 2. Configure Switches with respect to departments.

Step 3. Make UTP connection between hosts and the departments switches.

Step 6. Verify your configuration by pinging from Hosts to Hosts. The ping from one host to other should be successful within the department.

III. Result

Thus the real-time subnetting for given departments calculated and simulated in packet tracer.

Experiment No. &Title: 8) Implement and configure IIS server and FTP server.

Date : 15/10/2024

I. General Objective (Aim)

To Implement and configure IIS server and FTP server in windows.

II. Conduct of Experiment

Step 1: Install IIS (Internet Information Services)

1. **Open Windows Features**:
   - Press Windows + R to open the Run dialog.
   - Type optionalfeatures and press Enter to open "Windows Features."
2. **Enable IIS**:
   - In the "Windows Features" window, scroll down and check the following options:
     - **Internet Information Services** (IIS)
     - **Web Management Tools** (includes IIS Management Console)
     - **World Wide Web Services**
       - Under this, select "Web Server" and "FTP Server" (you may also enable other features depending on your needs).
3. **Click OK**:
   - Windows will install the selected features. It may take a few minutes.
4. **Verify IIS Installation**:
   - Once the installation is complete, open your browser and go to http://localhost. You should see the default IIS welcome page indicating that IIS is installed and running.

Step 2: Install FTP Server Feature

1. **Open Windows Features Again**:
   - Press Windows + R, type optionalfeatures, and press Enter.
2. **Enable FTP Server**:
   - In the "Windows Features" window, expand **Internet Information Services** > **FTP Server** and check:
     - **FTP Service**
     - **FTP Extensibility**
3. **Click OK** to install the FTP feature.

Step 3: Configure FTP Site in IIS

1. **Open IIS Manager**:
    - Press Windows + R, type inetmgr, and press Enter. This will open IIS Manager.
2. **Add FTP Site**:
    - In the **Connections** pane on the left, expand your server node (it will show the name of your PC).
    - Right-click on **Sites** and choose **Add FTP Site**.
3. **Configure FTP Site Settings**:
    - **FTP Site Name**: Choose a name for your FTP site.
    - **Physical Path**: Browse and select the folder you want to use as the root directory for FTP.
    - **Binding and SSL**:
        - **IP Address**: Select your machine's IP address or leave it as All Unassigned.
        - **Port**: By default, FTP uses port 21, so leave this as is.
        - **SSL**: Choose **No SSL** unless you want to configure secure FTP (FTPS).
4. **Configure Authentication and Authorization**:
    - **Authentication**: Select **Basic Authentication**.
    - **Authorization**: Choose **All users** or specific users/groups and assign the permissions (Read, Write, etc.).
5. **Finish**: Click **OK** to create the FTP site.

Step 4: Configure Windows Firewall for FTP

1. **Open Windows Firewall**:
    - Press Windows + R, type wf.msc, and press Enter to open the Windows Firewall with Advanced Security.
2. **Allow FTP Server**:
    - In the left pane, click **Inbound Rules**.
    - In the right pane, click **New Rule...**.
    - Select **Port**, then click **Next**.
    - Choose **TCP** and specify port 21 (for FTP) and click **Next**.
    - Allow the connection and click **Next**.
    - Specify the profiles (Domain, Private, Public) where this rule should apply, then click **Next**.
    - Name the rule (e.g., "Allow FTP") and click **Finish**.
3. **Allow Passive Mode FTP (if necessary)**:
    - If you're using passive mode FTP, you need to open a range of ports for the passive FTP connection. The range can be configured in IIS and needs to be allowed in the firewall.

Step 5: Test the FTP Server

1. **Test FTP Locally**:
    - Open File Explorer and type ftp://localhost in the address bar. You should be able to connect and view the FTP root directory.
2. **Test FTP Remotely**:
    - From another device or computer, open File Explorer or an FTP client like FileZilla.
    - Connect using ftp://<your-server-ip> or ftp://localhost (if testing locally).
    - Enter your FTP credentials to log in and confirm the setup.

Optional Step 6: Set Up FTP User Accounts (if needed)

If you want to restrict FTP access to specific users, follow these steps:

1. **Create Local User Accounts**:
    - Go to **Control Panel** > **User Accounts** > **Manage User Accounts**.
    - Create a new user for FTP access or modify an existing one.
2. **Assign Permissions to the FTP Site**:
    - Go back to IIS Manager and select your FTP site.
    - Under **FTP Authorization Rules**, click **Add Allow Rule**.
    - Choose the user/group you created and assign appropriate permissions.

IV. Result

Thus the Implementation and configuration of IIS server and FTP server completed and checked.

Experiment No. &Title: 9) Simulation of RIP using CISCO Packet tracer

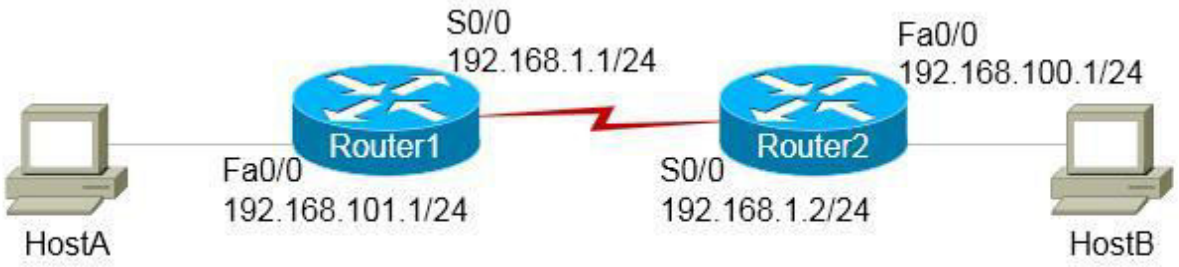
Date : 29/10/2024

I. General Objective (Aim)

To Configure Routing information protocol (RIP) and verify connectivity within the connected topology

II. Conduct of Experiment

Topology



IP TABLE

Device	Interface	IP Address	Subnetmask	Default Gateway
Host A	**Fast Ethernet**	**192.168.101.2**	**255.255.255.0**	**192.168.101.1**
Host B	**Fast Ethernet**	**192.168.100.2**	**255.255.255.0**	**192.168.100.1**
Router 1	**Fast Ethernet** **S0/0**	**192.168.101.1** **192.168.1.1**	**255.255.255.0** **255.255.255.0**	-
Router 2	**Fast Ethernet** **S0/0**	**192.168.100.1** **192.168.1.2**	**255.255.255.0** **255.255.255.0**	-

Command summary(Router)

enable	Enters privileged EXEC mode
configure terminal	Enters global configuration mode from privileged EXEC mode
interface *type number*	Changes from global configuration mode o interface configuration mode
ip address *ip-address subnet-mask*	enters router configuration mode for EIGRP
no shutdown	Enables an interface
router rip	Configure RIP
Rip v2	Enable advanced RIP routing
network *network-address*	activates the specified routing
Show ip interface brief	Displays the details of the interfaces
Show ip route	Displays the IP routing table

Steps involved in RIP configuration

Step 1. Configure Router1 with a host name of Router1. Configure the appropriate IPaddresses on the interfaces; refer to the IP Addresses table. A DCE cable is connected to Router1. The Serial link should have a speed of 64 Kbps. Enable the interfaces.

Step 2. Configure Router2 with a host name of Router2. Configure the appropriate IPaddresses on the interfaces; refer to the IP Addresses table. Enable the interfaces.

Step 3. On Router1, display the routing table and review the routes displayed. Note thatthe only routes displayed are the ones that are directly connected.

Step 4. On Router1 and Router2, configure RIP so that all devices can ping any otherdevice.

Step 5. On Router1, display the routing table and review the routes displayed. Note thatthe routes displayed are both those directly connected and the route advertised by Router2.

Step 6. Verify your configuration by pinging from HostA to HostB (192.168.100.2). Theping should be successful.

III. Result

Thus the simulation of RIP protocol is configured and tested.

Experiment No. &Title: 10) Simulation of OSPF using CISCO Packet tracer

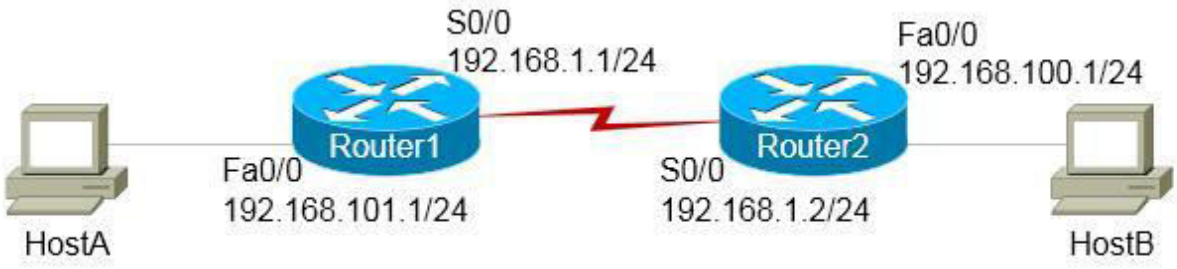
Date : 5/11/2024

I. General Objective (Aim)

To Configure Open shortest path first (OSPF) and verify connectivity within the connected topology.

II. Conduct of Experiment

Topology



IP TABLE

Device	Interface	IP Address	Subnetmask	Default Gateway
Host A	**Fast Ethernet**	**192.168.101.2**	**255.255.255.0**	**192.168.101.1**
Host B	**Fast Ethernet**	**192.168.100.2**	**255.255.255.0**	**192.168.100.1**
Router 1	**Fast Ethernet** **S0/0**	**192.168.101.1** **192.168.1.1**	**255.255.255.0** **255.255.255.0**	-
Router 2	**Fast Ethernet** **S0/0**	**192.168.100.1** **192.168.1.2**	**255.255.255.0** **255.255.255.0**	-

Command summary(Router)

enable	Enters privileged EXEC mode
configure terminal	Enters global configuration mode from privileged EXEC mode
interface *type number*	Chabges from global configuration mode o interface configuration mode
ip address *ip-address subnet-mask*	enters router configuration mode for EIGRP
no shutdown	Enables an interface
router ospf *process-id*	enters router configuration mode for an OSPF process
network *network-address wildcard-mask* **area** *area-id*	activates OSPF on the specified network and places the matching interface in the specified area.
Show ip interface brief	Displays the details of the interfaces
Show ip route	Displays the IP routing table

Steps involved in OSPF configuration

Step 1. Configure Router1 with a host name of Router1. Configure theappropriate IP addresses on the interfaces; refer to the IP Addresses table. A DCE cable is connected to Router1. The Serial link should have a speed of 64 Kbps. Enable the interfaces.

Step 2. Configure Router2 with a host name of Router2. Configure theappropriate IP addresses on the interfaces; refer to the IP Addresses table. Enable the interfaces.

Step 3. On Router1, display the routing table and review the routes displayed. Note that the only routes displayed are the ones that are directly connected.

Step 4. On Router1 and Router2, configure OSPF so that all devices can ping anyother device.

Step 5. On Router1, display the routing table and review the routes displayed.Note that the routes displayed are both those directly connected and the route advertised by Router2.

Step 6. Verify your configuration by pinging from HostA to HostB(192.168.100.2). The ping should be successful.

III. Result

Thus the simulation of OSPF protocol is configured and tested.