



**INSTITUTE OF ENGINEERING & MANAGEMENT  
SALT LAKE, KOLKATA**

**LAB MANUAL**

**Year : 2022 - 2026**

**Course Name : Computer Networks Laboratory**

**Course Code : PCCCS692**

**Semester : VI**

**Branch : CSBS**

**Computer Networks Laboratory (PCCCS692)**

**Name : Mafuj Gazi**

**University Roll No. : 12022002018004 Class Roll 07**

**Year : 3<sup>rd</sup> year Semester : 6<sup>th</sup> Semester**

**Session : 2024-2025**

# INDEX

**LAB : Computer Networks Laboratory**

YEAR : 3<sup>rd</sup> Year SEMESTER: 6<sup>th</sup>

Exp. No.	Name of Experiment	Date of Experiment	Date of Submission	Page	Remarks
01.	Exploring Basic Networking Commands	06/01/2025	22/01/2025	03-08	
02.	Packet Capture and Analysis using Wireshark	06/01/2025	22/01/2025	09-11	
03.	Socket Programming	15/01/2025	22/01/2025	12-18	

## Experiment No : 1

# Title: Exploring Basic Networking Commands

### Objective:

This lab introduces students to essential networking commands used for diagnosing, monitoring, and troubleshooting networks. By completing this assignment, students will develop hands-on experience with commands in Windows, Linux, or macOS.

### Identifying Network Configuration

1. **Objective:** Understand the network configuration of your system.
2. **Steps**
  - I. Run the following commands to view network configuration details:

- **Windows:** `ipconfig`
- **Linux/macOS:** `ifconfig` or `ip addr`

```
Connection-specific DNS Suffix  . :  
IPv6 Address. . . . . : 2405:201:802d:34:8038:153:7ca0:9be9  
Temporary IPv6 Address. . . . . : 2405:201:802d:34:e0b3:563f:6419:e492  
Link-local IPv6 Address . . . . . : fe80::25be:a260:7350:6aaf%21  
IPv4 Address. . . . . : 192.168.29.2  
Subnet Mask . . . . . : 255.255.255.0  
Default Gateway . . . . . : fe80::f2ed:b8ff:fe8e:e568%21  
                             192.168.29.1
```

- II. Note the following:

- IP Address                   **192.168.29.2**
- Subnet Mask               **255.255.255.0**
- Default Gateway       **fe80::f2ed:b8ff:fe8e:e568%21**

3. **Questions**

- I. What is your system's IP address?  
An IP address, or Internet Protocol address, is a unique identifier assigned to a device connected to a network. It's like a postal address, allowing data to be sent and received between devices on a network. There are two main types of IP addresses: IPv4 and IPv6.
  - II. What is the role of the default gateway in your network?  
In a network, the default gateway serves as an access point or a router that sends data from one network to another. Think of it as the middleman between your local network and the internet. When devices on your network need to communicate with devices on another network (such as accessing a website), they send the data packets to the default gateway. The gateway then forwards the packets to their destination outside the local network.

### Testing Network Connectivity

1. **Objective:** Use the **ping** command to test connectivity with other devices.
2. **Steps:**
  - I. Ping the following:
    1. Your default gateway.

```
C:\Users\Admin>ping fe80::f2ed:b8ff:fe8e:e568%21

Pinging fe80::f2ed:b8ff:fe8e:e568%21 with 32 bytes of data:
Reply from fe80::f2ed:b8ff:fe8e:e568%21: time=3ms
Reply from fe80::f2ed:b8ff:fe8e:e568%21: time=5ms
Reply from fe80::f2ed:b8ff:fe8e:e568%21: time=1ms
Reply from fe80::f2ed:b8ff:fe8e:e568%21: time=4ms

Ping statistics for fe80::f2ed:b8ff:fe8e:e568%21:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 5ms, Average = 3ms
```

2. A public server (e.g., 8.8.8.8).

```
C:\Users\Admin>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=30ms TTL=112
Reply from 8.8.8.8: bytes=32 time=29ms TTL=112
Reply from 8.8.8.8: bytes=32 time=29ms TTL=112
Reply from 8.8.8.8: bytes=32 time=29ms TTL=112

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 29ms, Maximum = 30ms, Average = 29ms
```

3. A domain name (e.g., [www.google.com](http://www.google.com)).

```
C:\Users\Admin>ping www.google.com

Pinging www.google.com [2404:6800:4002:81e::2004] with 32 bytes of data:
Reply from 2404:6800:4002:81e::2004: time=28ms
Reply from 2404:6800:4002:81e::2004: time=29ms
Reply from 2404:6800:4002:81e::2004: time=28ms
Reply from 2404:6800:4002:81e::2004: time=28ms

Ping statistics for 2404:6800:4002:81e::2004:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 28ms, Maximum = 29ms, Average = 28ms
```

- II. Record the round-trip time (RTT) for each ping.

### 3. Questions

- I. Was the ping to each target successful?

Yes the ping was successful to each target but it may fail under various circumstances.

- II. If a ping failed, what might be the reason?

Following are the reasons for a pin to fail :

- **Network Connectivity Issues:** If there is no physical or wireless connection between your device and the target device, the ping will fail.
- **Firewall or Security Settings:** Firewalls or security settings on either the source or target device may block ping requests, causing the ping to fail.
- **Target Device is Unreachable:** The target device might be turned off, disconnected from the network, or experiencing its own network issues.

- **Incorrect IP Address or Hostname:** If the IP address or hostname you are pinging is incorrect or does not exist, the ping will not succeed.
- **Network Congestion:** High traffic on the network can result in packet loss, causing ping requests to time out and fail.

## Tracing Routes

1. **Objective:** Use the traceroute command to map the path to a destination.
2. **Steps:**
  - I. Run:
    - **Windows:** `tracert www.google.com`
    - **Linux/macOS:** `traceroute www.google.com`

```
C:\Users\Admin>tracert www.google.com

Tracing route to www.google.com [2404:6800:4009:820::2004]
over a maximum of 30 hops:

  1      1 ms      1 ms      2 ms  2405:201:802d:34:f2ed:b8ff:fe8e:e568
  2      *        *        *      Request timed out.
  3     152 ms     6 ms     5 ms  2405:200:801:500::5
  4      *        *        *      Request timed out.
  5      5 ms      4 ms      4 ms  2405:200:806:3168:61::4
  6      *        *        *      Request timed out.
  7      6 ms      6 ms      6 ms  2405:200:801:500::c91
  8      *        *        *      Request timed out.
  9      5 ms      5 ms      5 ms  2405:200:801:500::1d
 10     29 ms     30 ms     30 ms  2001:4860:1:1::f48
 11     29 ms     29 ms     29 ms  2001:4860:1:1::f48
 12      *        *        *      Request timed out.
 13     29 ms     29 ms     28 ms  2001:4860:0:1::305a
 14     28 ms     28 ms     29 ms  2001:4860:0:1::7604
 15     46 ms     44 ms     45 ms  2001:4860::9:4001:7734
 16      *        *        *      Request timed out.
 17     44 ms     43 ms     46 ms  2001:4860:0:1::4fe9
 18     45 ms     45 ms     44 ms  bom07s30-in-x04.1e100.net [2404:6800:4009:820::2004]

Trace complete.
```

- II. Observe the hops the packets take to reach the destination.
3. **Questions**
    - I. How many hops did it take to reach [www.google.com](http://www.google.com)?  
It took a total of 18 (Eighteen) hops to reach www. Google.com

- II. Did any hops time out? If so, what could cause this?  
6 out of 18 hops timed out.

When performing a traceroute, hops may time out for several reasons:

1. **Firewall or Security Settings:** Firewalls on intermediate routers or the target device may block traceroute packets, causing timeouts.
2. **Network Congestion:** Heavy traffic on the network can result in packet delays or losses, leading to timeouts.
3. **Unresponsive Routers:** Some routers are configured not to respond to traceroute requests, causing the trace to show timeouts at those hops.
4. **Rate Limiting:** Network devices may limit the rate of responses to traceroute packets to prevent overloading, resulting in occasional timeouts.

5. **Device Issues:** The router or network device may be experiencing issues or high CPU usage, causing it to drop traceroute packets.

### Examining Active Connections

1. **Objective:** Identify active network connections using `netstat`.

2. **Steps:**

I. Run:

- Windows/Linux/macOS: `netstat -an`

```
C:\Users\Admin>netstat -an
```

#### Active Connections

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:3306	0.0.0.0:0	LISTENING
TCP	0.0.0.0:5040	0.0.0.0:0	LISTENING
TCP	0.0.0.0:5357	0.0.0.0:0	LISTENING
TCP	0.0.0.0:7680	0.0.0.0:0	LISTENING
TCP	0.0.0.0:33060	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49664	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49665	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49666	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49667	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49668	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49670	0.0.0.0:0	LISTENING
TCP	127.0.0.1:49671	127.0.0.1:49672	ESTABLISHED
TCP	127.0.0.1:49672	127.0.0.1:49671	ESTABLISHED
TCP	127.0.0.1:49673	127.0.0.1:49674	ESTABLISHED
TCP	127.0.0.1:49674	127.0.0.1:49673	ESTABLISHED
TCP	192.168.29.2:139	0.0.0.0:0	LISTENING
TCP	192.168.29.2:61107	20.212.88.117:443	ESTABLISHED
TCP	192.168.29.2:61191	20.50.201.204:443	TIME_WAIT
TCP	192.168.29.2:61193	20.189.173.9:443	ESTABLISHED

II. Identify:

- Any established TCP connections.  
Yes there exists some TCP connections that are established
- Any listening ports on your machine.  
Yes, there are some ports that are listening in my machine

3. **Questions**

- I. What are the most common protocols (e.g., TCP, UDP) used in the active connections?

The most common protocols used in active network connections include:

- TCP (Transmission Control Protocol):** This protocol is reliable and connection-oriented, ensuring that data packets are delivered in the correct order and without errors. It's widely used for applications that require guaranteed delivery, such as web browsing (HTTP/HTTPS), email (SMTP, IMAP, POP3), and file transfers (FTP).
- UDP (User Datagram Protocol):** Unlike TCP, UDP is connectionless and does not guarantee the delivery of data packets. It's faster but less reliable, making it suitable for

applications where speed is more critical than accuracy, such as video streaming, online gaming, and voice-over-IP (VoIP).

- II. Why might some ports be in a listening state?  
Ports are in a listening state when an application or service on a device is actively waiting for incoming connections.

## DNS and Name Resolution

**Objective:** Understand how DNS resolves domain names to IP addresses.

### Steps:

- I. Run:

**Windows:** nslookup [www.example.com](http://www.example.com)

```
C:\Users\Admin>nslookup www.example.com
Server:  reliance.reliance
Address:  2405:201:802d:34::c0a8:1d01

Non-authoritative answer:
Name:     a1422.dscr.akamai.net
Addresses: 2405:200:1630:a06::312c:c740
           2405:200:1630:a06::312c:c773
           49.44.176.51
           49.44.176.65
Aliases:  www.example.com
           www.example.com-v4.edgesuite.net
```

**Linux/macOS:** nslookup [www.example.com](http://www.example.com) or dig [www.example.com](http://www.example.com)

- II. Note the resolved IP address.

### • Questions

- I. What is the resolved IP address of [www.example.com](http://www.example.com)?  
The resolved IP address of [www.example.com](http://www.example.com) is 49.44.476.51 & 49.44.476.65
- II. What happens if you try to resolve a non-existent domain (e.g., [www.invalidexample.com](http://www.invalidexample.com))?

```
C:\Users\Admin>nslookup www.invalidexample.com
Server:  reliance.reliance
Address:  2405:201:802d:34::c0a8:1d01

*** reliance.reliance can't find www.invalidexample.com: Non-existent domain
```

It throughs a response of the domain being non-existent.

## Exploring ARP Cache

1. **Objective:** View the ARP cache on your system.
2. **Steps:**
  - I. Run:
    - **Windows/Linux/macOS:** arp -a



```
C:\Users\Admin>arp -a

Interface: 192.168.29.2 --- 0x15
    Internet Address      Physical Address      Type
    192.168.29.1          f0-ed-b8-8e-e5-68     dynamic
    192.168.29.255        ff-ff-ff-ff-ff-ff     static
    224.0.0.22            01-00-5e-00-00-16     static
    224.0.0.251           01-00-5e-00-00-fb     static
    224.0.0.252           01-00-5e-00-00-fc     static
    239.255.255.250       01-00-5e-7f-ff-fa     static
    255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

## II. Identify:

- MAC addresses of devices in the cache.
- Corresponding IP addresses.

## 3. Questions

### I. What is the purpose of the ARP cache?

The ARP (Address Resolution Protocol) cache serves a crucial role in network communication. Here's what it does:

1. **Maps IP Addresses to MAC Addresses:** The ARP cache stores the mappings between IP addresses (used for network-level routing) and MAC addresses (used for data link layer communication). This mapping allows devices to communicate within a local network.
2. **Improves Network Efficiency:** By keeping a record of the IP-to-MAC address mappings, the ARP cache eliminates the need to repeatedly broadcast ARP requests to resolve addresses, thus reducing network traffic and latency.
3. **Enhances Speed:** Cached entries mean quicker access to devices on the network, improving the speed of data transmission since the device does not have to wait for an ARP request/response cycle every time it communicates.

### II. How can outdated ARP entries affect network communication?

Outdated ARP entries can cause several issues in network communication, including:

1. **Misrouted Packets:** If an ARP cache contains outdated mappings, packets might be sent to the wrong MAC address. This misdirection can lead to failed communication attempts.
2. **Communication Failures:** Devices expecting to communicate with a specific MAC address may be unable to establish a connection if the address has changed but the ARP cache hasn't been updated.
3. **Increased Network Traffic:** Devices might repeatedly attempt to communicate with the wrong MAC address, generating unnecessary network traffic and potentially leading to congestion.



## Experiment No :2

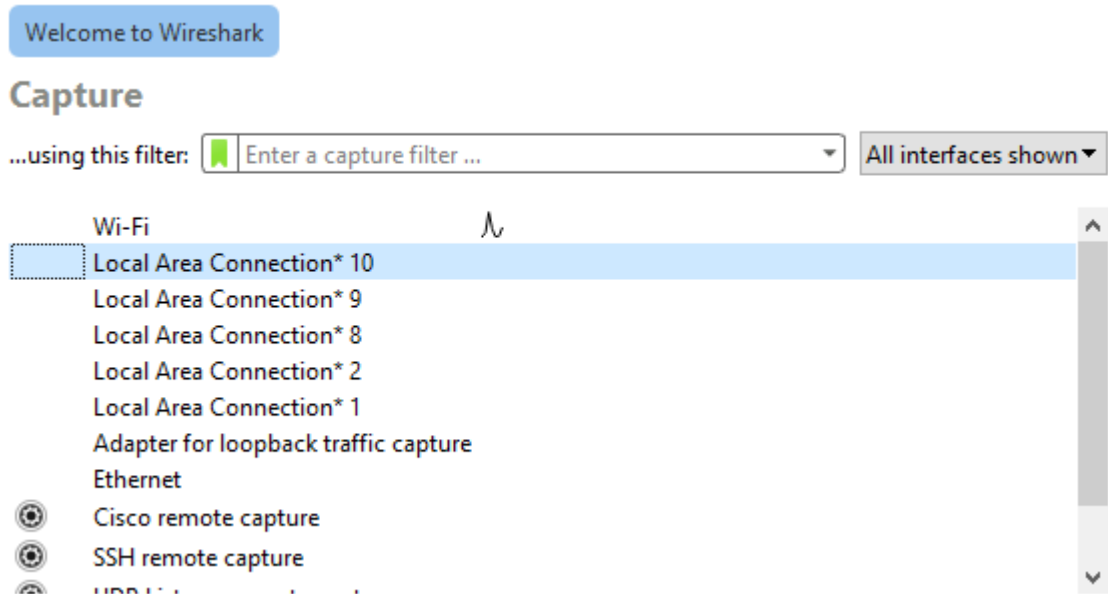
# Title: Packet Capture and Analysis Using Wireshark

### Objective:

This lab introduces students to network packet analysis using Wireshark. By completing the assignment, students will learn how to capture, filter, and analyse network traffic effectively.

### Getting Started with Wireshark

1. **Objective:** Familiarize students with the Wireshark interface and basic functionality.
2. **Steps:**
  1. Launch Wireshark and identify the available network interfaces.



2. Start a packet capture on the primary interface (e.g., Wi-Fi or Ethernet).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.29.2	224.0.0.22	IGMPv3	54	Membership Report / Join
2	0.499981	192.168.29.2	224.0.0.22	IGMPv3	54	Membership Report / Join
3	0.572035	2405:200:1606:1731:...	2405:201:802d:34:e0...	TLSv1.2	98	Application Data
4	0.624973	2405:201:802d:34:e0...	2405:200:1606:1731:...	TCP	74	61410 → 443 [ACK] Seq=1 A
5	1.778777	fe80::f2ed:b8ff:fe8...	ff02::1	ICMPv6	142	Router Advertisement from
6	1.778777	fe80::f2ed:b8ff:fe8...	ff02::1	ICMPv6	142	Router Advertisement from
7	1.779052	fe80::f2ed:b8ff:fe8...	ff02::1	ICMPv6	142	Router Advertisement from
8	3.063097	2405:201:802d:34:e0...	2603:1046:900:40::2	TCP	74	61475 → 443 [FIN, ACK] Se
9	3.063514	192.168.29.2	13.107.6.158	TCP	54	61471 → 443 [FIN, ACK] Se
10	3.103111	13.107.6.158	192.168.29.2	TCP	54	443 → 61471 [FIN, ACK] Se
11	3.103173	192.168.29.2	13.107.6.158	TCP	54	61471 → 443 [ACK] Seq=2 A

3. Browse a website (e.g., [www.example.com](http://www.example.com)) during the capture.
4. Stop the capture and save it as `capture_part1.pcap`.

### 3. Questions

- a. Which network interface did you use, and why?  
Wifi was used as it the available interface in my system connecting to the internet.
- b. How many packets were captured in total?

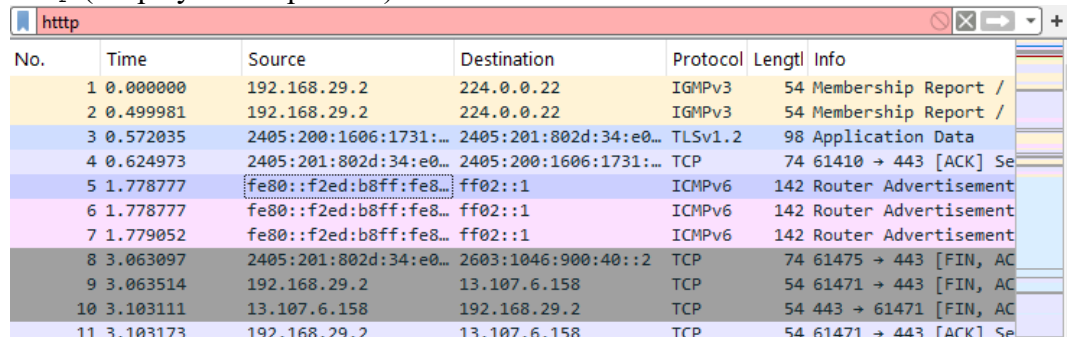
**Statistics**

<u>Measurement</u>	<u>Captured</u>	<u>Displayed</u>	<u>Marked</u>
Packets	4123	4123 (100.0%)	—
Time span, s	92.097	92.097	—
Average pps	44.8	44.8	—
Average packet size, B	707	707	—
Bytes	2913938	2913938 (100.0%)	0
Average bytes/s	31 k	31 k	—
Average bits/s	253 k	253 k	—

**Applying Filters**

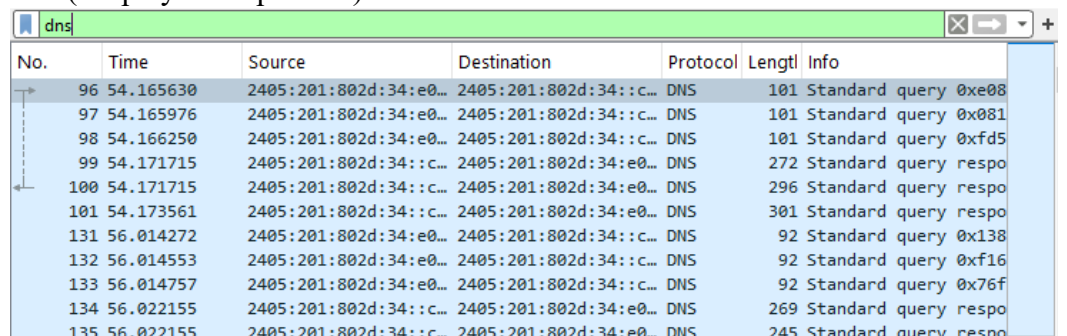
1. **Objective:** Learn to apply display filters to narrow down relevant packets.
2. **Tasks:**
  - I. Use the capture from Part 1.
  - II. Apply the following filters and note the results:

- http (Display HTTP packets)



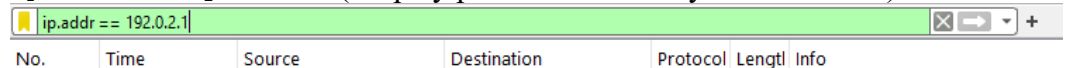
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.29.2	224.0.0.22	IGMPv3	54	Membership Report /
2	0.499981	192.168.29.2	224.0.0.22	IGMPv3	54	Membership Report /
3	0.572035	2405:200:1606:1731::...	2405:201:802d:34:e0...	TLSv1.2	98	Application Data
4	0.624973	2405:201:802d:34:e0...	2405:200:1606:1731::...	TCP	74	61410 → 443 [ACK] Seq
5	1.778777	fe80::f2ed:b8ff:fe8...	ff02::1	ICMPv6	142	Router Advertisement
6	1.778777	fe80::f2ed:b8ff:fe8...	ff02::1	ICMPv6	142	Router Advertisement
7	1.779052	fe80::f2ed:b8ff:fe8...	ff02::1	ICMPv6	142	Router Advertisement
8	3.063097	2405:201:802d:34:e0...	2603:1046:900:40::2	TCP	74	61475 → 443 [FIN, AC
9	3.063514	192.168.29.2	13.107.6.158	TCP	54	61471 → 443 [FIN, AC
10	3.103111	13.107.6.158	192.168.29.2	TCP	54	443 → 61471 [FIN, AC
11	3.103173	192.168.29.2	13.107.6.158	TCP	54	61471 → 443 [ACK] Se

- dns (Display DNS packets)



No.	Time	Source	Destination	Protocol	Length	Info
96	54.165630	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	101	Standard query 0xe08
97	54.165976	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	101	Standard query 0x081
98	54.166250	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	101	Standard query 0xfd5
99	54.171715	2405:201:802d:34::c...	2405:201:802d:34:e0...	DNS	272	Standard query respo
100	54.171715	2405:201:802d:34::c...	2405:201:802d:34:e0...	DNS	296	Standard query respo
101	54.173561	2405:201:802d:34::c...	2405:201:802d:34:e0...	DNS	301	Standard query respo
131	56.014272	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	92	Standard query 0x138
132	56.014553	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	92	Standard query 0xf16
133	56.014757	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	92	Standard query 0x76f
134	56.022155	2405:201:802d:34::c...	2405:201:802d:34:e0...	DNS	269	Standard query respo
135	56.022155	2405:201:802d:34::c...	2405:201:802d:34:e0...	DNS	245	Standard query respo

- ip.addr == <your IP> (Display packets related to your IP address)



No.	Time	Source	Destination	Protocol	Length	Info
-----	------	--------	-------------	----------	--------	------

- III. Identify the DNS query and response for [www.example.com](http://www.example.com).

dns.qry.name == "www.example.com"						
No.	Time	Source	Destination	Protocol	Length	Info
2455	70.138434	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	95	Standard query 0x35e
2456	70.140027	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	95	Standard query 0x19b
2457	70.140412	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	95	Standard query 0x217
2462	70.148961	2405:201:802d:34::c...	2405:201:802d:34:e0...	DNS	229	Standard query respo
2468	70.150673	2405:201:802d:34::c...	2405:201:802d:34:e0...	DNS	205	Standard query respo
2469	70.150673	2405:201:802d:34::c...	2405:201:802d:34:e0...	DNS	234	Standard query respo
2480	70.167060	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	95	Standard query 0xb87
2481	70.167331	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	95	Standard query 0xe84
2482	70.167526	2405:201:802d:34:e0...	2405:201:802d:34::c...	DNS	95	Standard query 0xaf8
2483	70.169871	2405:201:802d:34::c...	2405:201:802d:34:e0...	DNS	232	Standard query respo
2484	70.169871	2405:201:802d:34::c...	2405:201:802d:34:e0...	DNS	208	Standard query respo

### 3. Questions

- What is the IP address resolved for [www.example.com](http://www.example.com)?  
2405:201:802d:34::c0a8:1d01
- How many HTTP packets were captured?  
A total of 4123 packets were captured

### Analysing Protocols

- Objective:** Dive deeper into protocol details and packet structure.
- Steps:**
  - Select a single HTTP GET request packet.

No.	Time	Source	Destination	Protocol	Length	Info
3344	30.758774	2405:201:802d:34:e0...	2404:6800:4009:812:...	HTTP	274	GET /r/r1.crl HTTP/1.1
3351	30.802910	2404:6800:4009:812:...	2405:201:802d:34:e0...	HTTP	297	HTTP/1.1 304 Not Modified
3352	30.815850	2405:201:802d:34:e0...	2404:6800:4009:812:...	HTTP	274	GET /r/r4.crl HTTP/1.1
3354	30.860191	2404:6800:4009:812:...	2405:201:802d:34:e0...	HTTP	296	HTTP/1.1 304 Not Modified

- Expand the protocol layers (Ethernet, IP, TCP, HTTP) in the packet details pane.
- Note the source IP, destination IP, and the requested URL.

### 3. Questions

- What is the source and destination IP of the HTTP packet?  
Source - 2405:201:802d:34:e0b3:563f:6419:e492  
Destination - 2404:6800:4009:812::2003
- What is the URL requested in the GET packet?  
/r/r1.crl HTTP/1.1

### Capturing Specific Traffic

- Objective:** Use capture filters to focus on specific traffic.
- Tasks:**
  - Restart Wireshark and apply the following capture filter: port 53 (DNS traffic).
  - Initiate a new DNS query by visiting a new website (e.g., [www.google.com](http://www.google.com)).
  - Stop the capture and save it as `capture_part4.pcap`.

### 3. Questions

- What is the DNS query sent for [www.google.com](http://www.google.com)?  
Standard query 0x9498 A play.google.com
- What was the response from the DNS server?  
Standard query response 0x0788 AAAA play.google.com AAAA 2404:6800:4009:81f::200e

## Experiment No :3

# Title : Socket Programming

### Objective :

To write a code in Java and C to implement Client-Server model where the server listens for incoming connection and client initiates the connection. Also to develop a Multithreaded server to connect and serve multiple clients

### Inputs :

*CustomClient.java*

```
import java.io.*;
import java.net.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class CustomClient {
    public static void main(String[] args) {
        try {
            // Connect to the server on localhost at port 12345
            Socket socket = new Socket("localhost", 12345);
            System.out.println("Connected to the server");

            // Set up input and output streams
            OutputStream output = socket.getOutputStream();
            PrintWriter writer = new PrintWriter(output, true);
            InputStream input = socket.getInputStream();
            BufferedReader reader = new BufferedReader(new InputStreamReader(input));

            // Scanner to read user input
            Scanner scanner = new Scanner(System.in);
            String message;
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

            System.out.println("Type your messages below (type 'exit' to quit):");

            // Loop to send custom messages
            while (true) {
                System.out.print("Client: ");
                message = scanner.nextLine();

                // Send message to server with timestamp
                String timestamp = sdf.format(new Date());
                writer.println "[" + timestamp + " ] " + message;

                // Break the loop if the user types 'exit'
                if ("exit".equalsIgnoreCase(message)) {
                    break;
                }

                // Receive and print the response from the server
                String serverMessage = reader.readLine();
                System.out.println("Server: " + serverMessage);
            }
        }
    }
}
```

```
        // Close the socket and scanner
        socket.close();
        scanner.close();
        System.out.println("Connection closed");
    } catch (UnknownHostException ex) {
        System.out.println("Server not found: " + ex.getMessage());
    } catch (IOException ex) {
        System.out.println("I/O error: " + ex.getMessage());
    }
}
}
```

#### *Server.Java*

```
import java.io.*;
import java.net.*;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Server {
    public static void main(String[] args) {
        try {
            // Create a server socket that listens on port 12345
            ServerSocket serverSocket = new ServerSocket(12345);
            System.out.println("Server is listening on port 12345");

            // Accept a connection from the client
            Socket socket = serverSocket.accept();
            System.out.println("Client connected");

            // Set up input and output streams
            InputStream input = socket.getInputStream();
            BufferedReader reader = new BufferedReader(new InputStreamReader(input));
            OutputStream output = socket.getOutputStream();
            PrintWriter writer = new PrintWriter(output, true);

            String message;
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

            // Loop to read and respond to messages from the client
            while ((message = reader.readLine()) != null) {
                String timestamp = sdf.format(new Date());
                System.out.println "[" + timestamp + "] Received message from client: " +
message);

                // If the client sends 'exit', break the loop
                if ("exit".equalsIgnoreCase(message)) {
                    break;
                }

                // Send a response to the client
                writer.println "[" + timestamp + "] Message received: " + message);
            }

            // Close the socket
            socket.close();
            serverSocket.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

```
    }  
  }  
}
```

## Outputs:

```
Server is listening on port 12345  
Client connected  
[2025-01-19 18:30:16] Received message from client: [2025-01-19 18:30:16] hello  
[2025-01-19 18:30:22] Received message from client: [2025-01-19 18:30:22] hi  
[2025-01-19 18:30:27] Received message from client: [2025-01-19 18:30:27] how are you  
[2025-01-19 18:30:30] Received message from client: [2025-01-19 18:30:30] exit
```

Connected to the server

Type your messages below (type 'exit' to quit):

Client: hello

Server: [2025-01-19 18:30:16] Message received: [2025-01-19 18:30:16] hello

Client: hi

Server: [2025-01-19 18:30:22] Message received: [2025-01-19 18:30:22] hi

Client: how are you

Server: [2025-01-19 18:30:27] Message received: [2025-01-19 18:30:27] how are you

Client: exit

Connection closed

*MultiThreadedServer.java*

```
import java.io.*;  
import java.net.*;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class MultiThreadedServer {  
    public static void main(String[] args) {  
        try {  
            ServerSocket serverSocket = new ServerSocket(12345);  
            System.out.println("Server is listening on port 12345");  
  
            while (true) {  
                Socket socket = serverSocket.accept();  
                System.out.println("New client connected");  
  
                // Create a new thread for each client connection  
                new ClientHandler(socket).start();  
            }  
        } catch (IOException ex) {  
            ex.printStackTrace();  
        }  
    }  
}  
  
class ClientHandler extends Thread {  
    private Socket socket;  
  
    public ClientHandler(Socket socket) {  
        this.socket = socket;  
    }  
  
    public void run() {  
        try {  
            InputStream input = socket.getInputStream();
```

```

        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        OutputStream output = socket.getOutputStream();
        PrintWriter writer = new PrintWriter(output, true);

        String message;
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

        while ((message = reader.readLine()) != null) {
            String timestamp = sdf.format(new Date());
            System.out.println "[" + timestamp + "] Received from client: " +
message);
            writer.println "[" + timestamp + "] Echo: " + message);

            if ("exit".equalsIgnoreCase(message)) {
                break;
            }
        }

        socket.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
}

```

## Outputs:

Server

```

Server is listening on port 12345
New client connected
[2025-01-19 18:33:21] Received from client: [2025-01-19 18:33:21] hello
[2025-01-19 18:33:29] Received from client: [2025-01-19 18:33:29] this is the 1st client
New client connected
[2025-01-19 18:33:48] Received from client: [2025-01-19 18:33:48] hello i am the 2nd client
[2025-01-19 18:34:01] Received from client: [2025-01-19 18:34:01] bye from 2nd client
[2025-01-19 18:34:05] Received from client: [2025-01-19 18:34:05] exit

```

Client1

```

Connected to the server
Type your messages below (type 'exit' to quit):
Client: hello
Server: [2025-01-19 18:33:21] Echo: [2025-01-19 18:33:21] hello
Client: this is the 1st client
Server: [2025-01-19 18:33:29] Echo: [2025-01-19 18:33:29] this is the 1st client
Client: ...
Server: [2025-01-19 18:34:13] Echo: [2025-01-19 18:34:13] ...
Client: exit
Connection closed

```

Client2

```

Connected to the server
Type your messages below (type 'exit' to quit):
Client: hello i am the 2nd client
Server: [2025-01-19 18:33:48] Echo: [2025-01-19 18:33:48] hello i am the 2nd client
Client: bye from 2nd client
Server: [2025-01-19 18:34:01] Echo: [2025-01-19 18:34:01] bye from 2nd client
Client: exit
Connection closed

```

## Inputs :

Client.c

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <time.h>

int main() {
    int client_socket;

```



```
struct sockaddr_in server_addr;
char buffer[1024];
time_t now;
struct tm *timeinfo;
char timestamp[80];

client_socket = socket(AF_INET, SOCK_STREAM, 0);
if (client_socket == -1) {
    perror("socket");
    exit(1);
}

server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
server_addr.sin_port = htons(12345);

if (connect(client_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) == -1)
{
    perror("connect");
    close(client_socket);
    exit(1);
}

printf("Connected to the server\n");
printf("Type your messages below (type 'exit' to quit):\n");

while (1) {
    printf("Client: ");
    fgets(buffer, sizeof(buffer), stdin);
    buffer[strcspn(buffer, "\n")] = '\0'; // Remove the newline character

    time(&now);
    timeinfo = localtime(&now);
    strftime(timestamp, sizeof(timestamp), "%Y-%m-%d %H:%M:%S", timeinfo);

    snprintf(buffer, sizeof(buffer), "[%s] %s", timestamp, buffer);
    send(client_socket, buffer, strlen(buffer), 0);

    if (strcmp(buffer, "[%s] exit") == 0) {
        break;
    }

    int read_size = recv(client_socket, buffer, sizeof(buffer) - 1, 0);
    buffer[read_size] = '\0';
    printf("Server: %s\n", buffer);
}

close(client_socket);
printf("Connection closed\n");
return 0;
}
```

Server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
```

```
#include <pthread.h>
#include <time.h>

void *handle_client(void *arg);

int main() {
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr;
    socklen_t client_addr_size;
    pthread_t thread_id;

    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket == -1) {
        perror("socket");
        exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(12345);

    if (bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) == -1) {
        perror("bind");
        close(server_socket);
        exit(1);
    }

    if (listen(server_socket, 5) == -1) {
        perror("listen");
        close(server_socket);
        exit(1);
    }

    printf("Server is listening on port 12345\n");

    while (1) {
        client_addr_size = sizeof(client_addr);
        client_socket = accept(server_socket, (struct sockaddr*)&client_addr,
&client_addr_size);
        if (client_socket == -1) {
            perror("accept");
            continue;
        }
        printf("New client connected\n");

        if (pthread_create(&thread_id, NULL, handle_client, (void*)&client_socket) != 0) {
            perror("pthread_create");
            close(client_socket);
        }
        pthread_detach(thread_id);
    }

    close(server_socket);
    return 0;
}

void *handle_client(void *arg) {
    int client_socket = *(int*)arg;
    char buffer[1024];
```

```

int read_size;
time_t now;
struct tm *timeinfo;
char timestamp[80];

while ((read_size = recv(client_socket, buffer, sizeof(buffer) - 1, 0)) > 0) {
    buffer[read_size] = '\0';
    time(&now);
    timeinfo = localtime(&now);
    strftime(timestamp, sizeof(timestamp), "%Y-%m-%d %H:%M:%S", timeinfo);
    printf("[%s] Received message from client: %s\n", timestamp, buffer);

    if (strcmp(buffer, "exit") == 0) {
        break;
    }

    snprintf(buffer, sizeof(buffer), "[%s] Message received: %s", timestamp, buffer);
    send(client_socket, buffer, strlen(buffer), 0);
}

close(client_socket);
printf("Client disconnected\n");
return NULL;
}

```

## Outputs :

```

mafuj@mafuj-desktop:~/Desktop$ gcc -o server server.c
mafuj@mafuj-desktop:~/Desktop$ gcc -o client client.c
mafuj@mafuj-desktop:~/Desktop$ ./server
bash: ./ : Is a directory
mafuj@mafuj-desktop:~/Desktop$ ./server
Waiting for connections...
New connection, socket fd is 4, ip is : 127.0.0.1, port : 53506
[2025-01-21 19:12:53] Message from client: wfrefe

[2025-01-21 19:13:01] Message from client: ferwer

[2025-01-21 19:13:03] Message from client: egg

New connection, socket fd is 5, ip is : 127.0.0.1, port : 48532
[2025-01-21 19:13:13] Message from client: ere

[2025-01-21 19:13:15] Message from client: efer

[2025-01-21 19:13:16] Message from client: eg

[2025-01-21 19:13:17] Message from client: ewsafas

[2025-01-21 19:13:24] Message from client: exit

```

```

mafuj@mafuj-desktop:~/Desktop$ ./client
Enter message: ere
Server: [2025-01-21 19:13:13] Server received your message
Enter message: efer
Server: [2025-01-21 19:13:15] Server received your message
Enter message: eg
Server: [2025-01-21 19:13:16] Server received your message
Enter message: ewsafas
Server: [2025-01-21 19:13:17] Server received your message
Enter message: exit
Server: [2025-01-21 19:13:24] Server received your message
Enter message: 

```

```

mafuj@mafuj-desktop:~/Desktop$ ./client
Enter message: wfrefe
Server: [2025-01-21 19:12:53] Server received your message
Enter message: ferwer
Server: [2025-01-21 19:13:01] Server received your message
Enter message: egg
Server: [2025-01-21 19:13:03] Server received your message
Enter message: wer

```

## Experiment No : 4

# Title : Study of Different Network Topologies Using Cisco Packet Tracer

### Introduction:

The topology of a network defines how various devices like routers, switches, and computers are connected and how data flows between them. Different topologies offer distinct benefits and drawbacks in terms of performance, reliability, and cost. Cisco Packet Tracer is a simulation tool used to model various network topologies and test their functionality without the need for physical hardware. This report studies five common network topologies and aims to analyze their performance and behavior within a simulated environment.

### Objective :

- To study and simulate various network topologies using Cisco Packet Tracer.
- To analyze and compare the advantages and disadvantages of different topologies.
- To evaluate the performance and behavior of each topology under different scenarios.

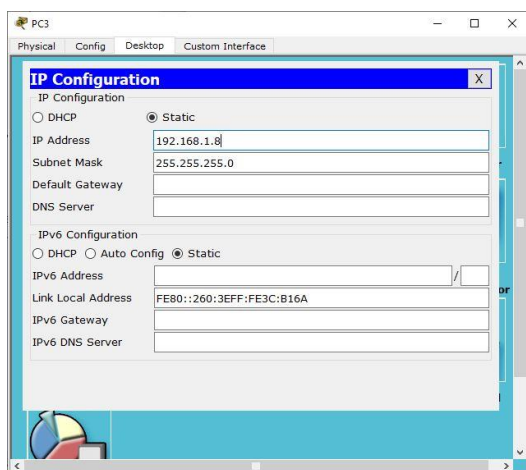
### Material and Methods :

- **Cisco Packet Tracer:** A network simulation software used to design and analyze network topologies.
- **Network Devices:** Routers, switches, PCs, and other devices used to form the network.
- **Network Topologies:** The following topologies were studied:

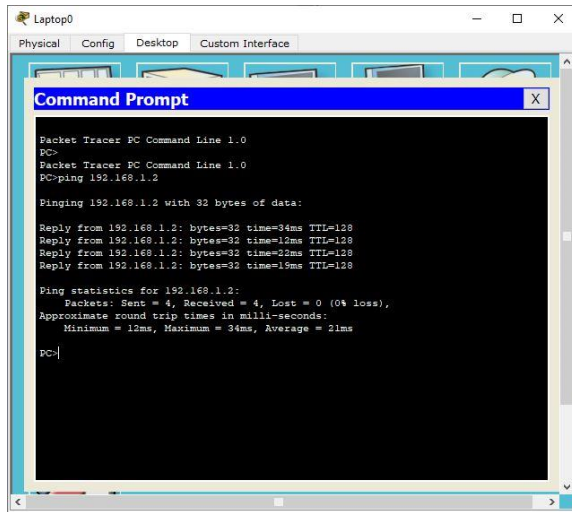
- **Bus Topology**
- **Star Topology**
- **Ring Topology**
- **Mesh Topology**
- **Hybrid Topology**

For each topology, a simulated network was created in Cisco Packet Tracer. The following steps were followed:

- Devices were connected to create the network according to the chosen topology.
- The appropriate IP configurations were applied to each device.



- Connectivity and performance were tested using ping commands and data transfers.

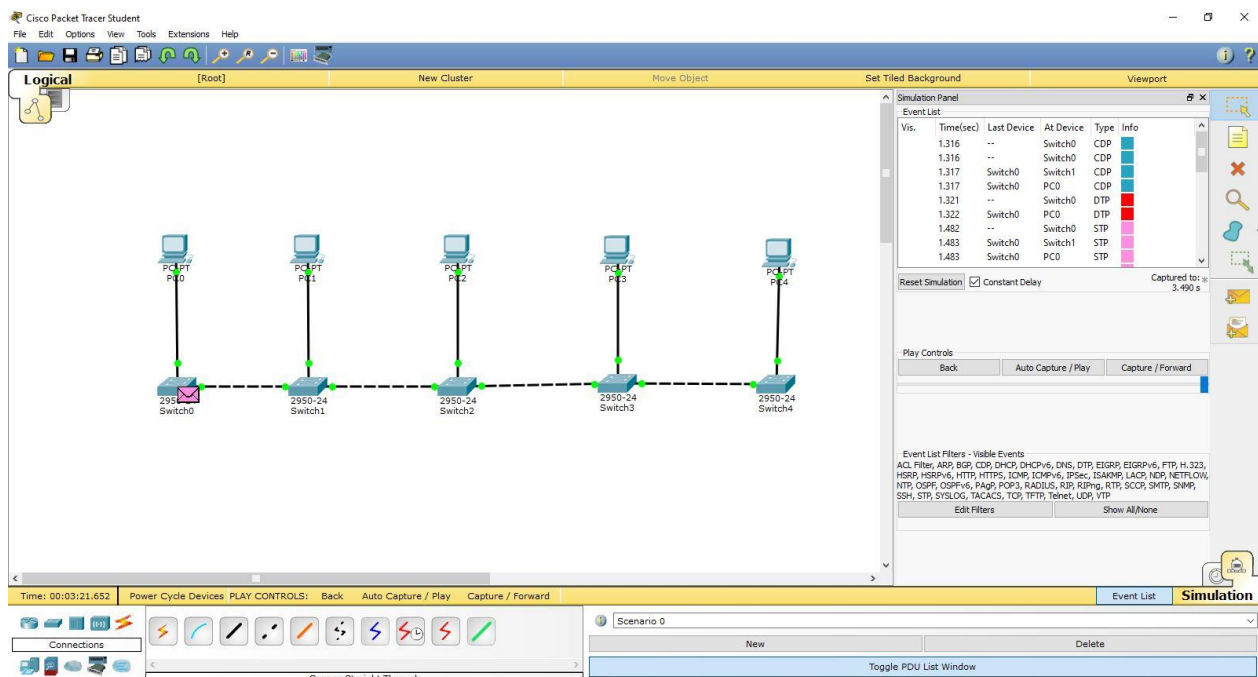


- Troubleshooting and failure scenarios were simulated to observe how the network behaved under fault conditions.

## Theory :

### Bus Topology:

- All devices are connected to a single central cable (bus).
- Data sent by any device is received by all others.
- Advantages: Easy to set up, cost-effective.
- Disadvantages: A failure in the central bus can bring down the entire network.



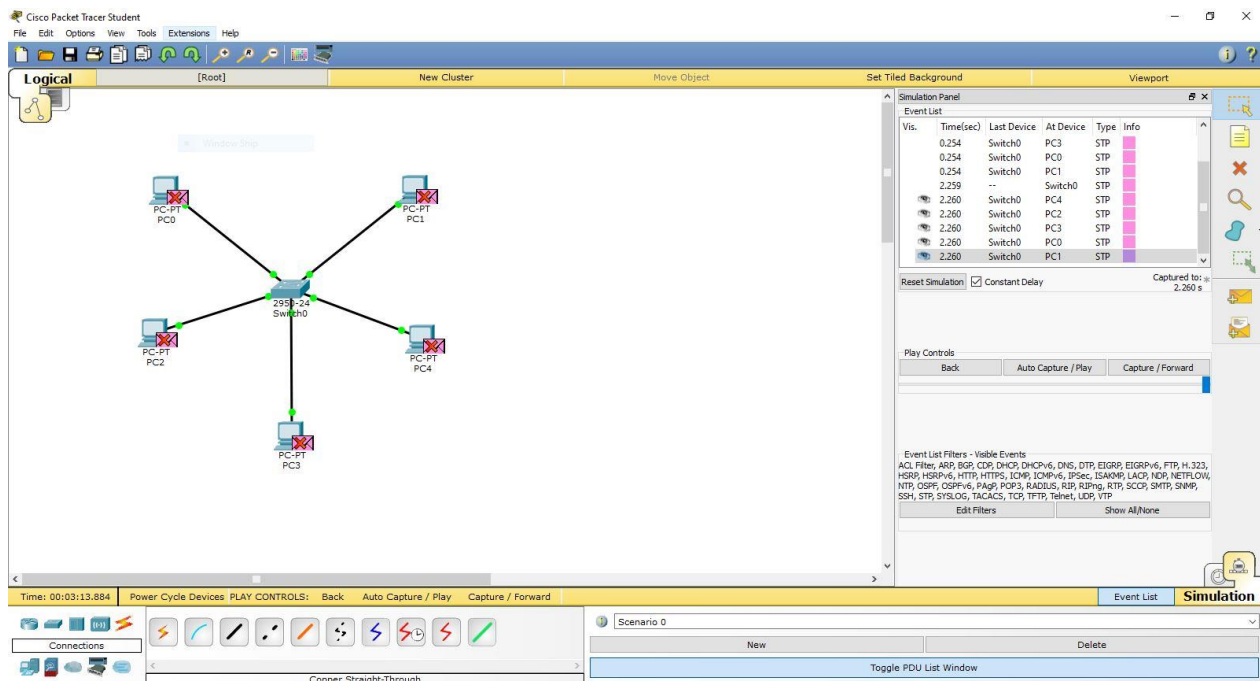
### Observation :

When testing bus topology, we observed that network performance was good with minimal devices. However, as more devices were added, the network speed decreased due to congestion on the shared bus.

### Star Topology:

- All devices are connected to a central device, usually a switch or hub.

- Advantages: Easy to manage and troubleshoot, failure in one device doesn't affect others.
- Disadvantages: High dependency on the central hub; if the hub fails, the entire network is affected.

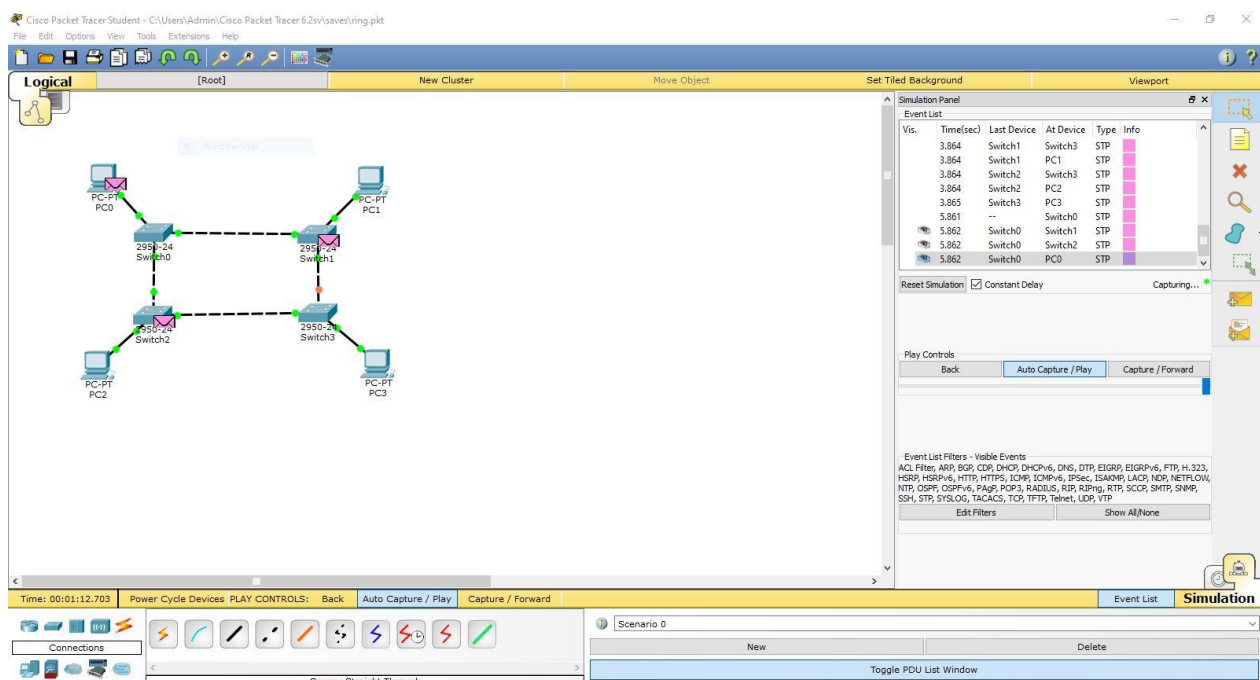


### Observation :

The star topology was very stable, and adding/removing devices did not impact the overall network performance as long as the central hub/switch remained functional. However, a failure in the hub led to a total network shutdown.

### Ring Topology:

- Devices are connected in a circular fashion.
- Data travels in one direction (or two in a dual-ring topology).
- Advantages: Minimal collision; better performance with fewer devices.
- Disadvantages: A failure in one device or connection can disrupt the entire network.



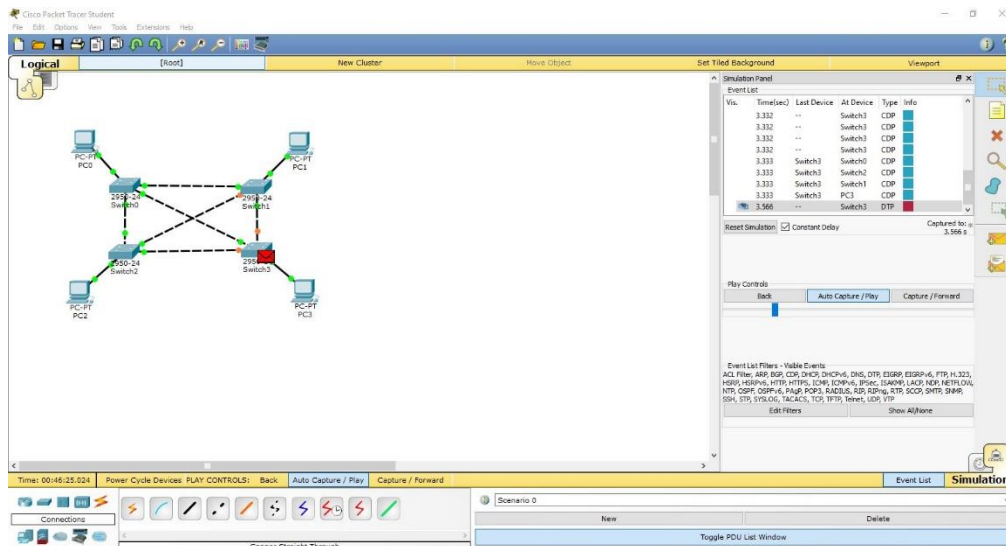
### Observation :



In the ring topology, data was transmitted smoothly without collisions. A failure in one of the connections, however, brought the entire network to a halt. In a dual-ring setup, the network remained functional even with a failed connection.

## Mesh Topology:

- Every device is connected to every other device.
- Advantages: High redundancy, highly reliable.
- Disadvantages: Expensive to implement due to the number of connections.

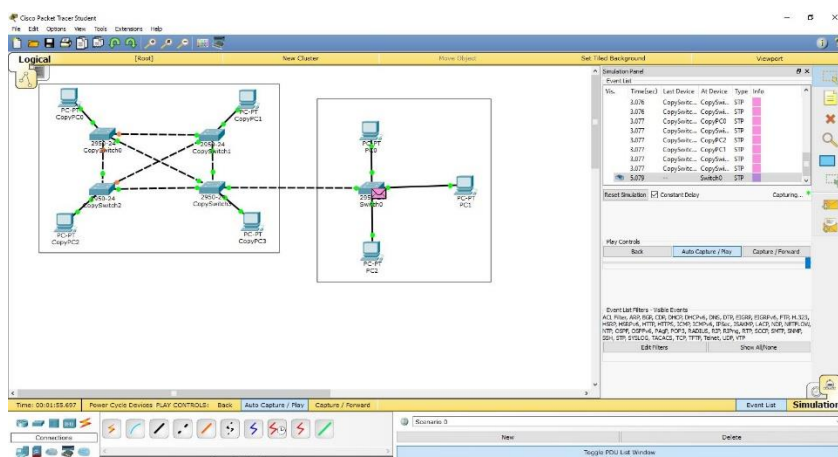


### Observation :

Mesh topology proved to be the most resilient, with multiple paths for data to travel. Even if one link failed, communication could still be maintained. However, it was also the most resource-intensive topology, requiring the most connections and devices.

## Hybrid Topology:

- A combination of two or more different topologies.
- Advantages: Flexible and scalable, can provide the benefits of different topologies.
- Disadvantages: Complexity in setup and management.



### Observation :

Combining the best features of star and bus topologies provided a flexible and scalable solution. However, it was more complex to configure and manage.



## Experiment No : 5

# Title : Configuration of Wireless LAN

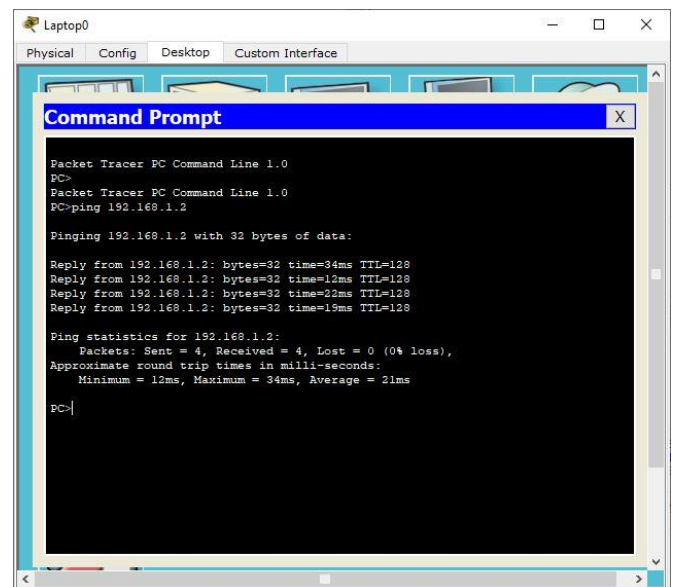
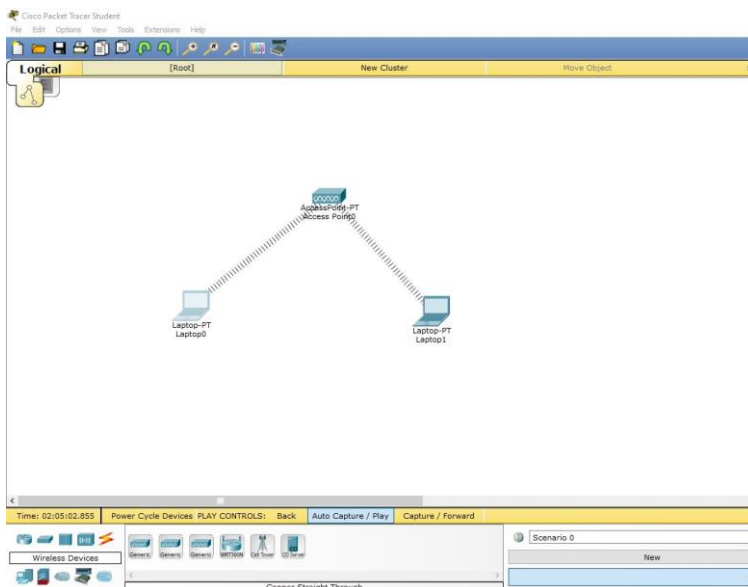
**Aim:** To construct a Wireless LAN and make the PC's communicate wirelessly

### Requirements

- Windows pc – 2 Nos
- CISCO Packet Tracer Software ( Student Version)
- 8 port switch – 1 No
- Cat-5 LAN cable

### Procedure

- Open the CISCO Packet tracer software
- Drag and drop 2 Laptop pcs using End Device Icons on the left corner
- Select Access point and server from wireless devices
- Select laptop-> physical-> OFF laptop-> remove LAN Module & replace WPC 300N Wireless module -> ON Laptop
- Observe the wireless connections between access point and laptops
- Give IP address of the PCs as per table, ping between PCs and observe the transfer of data packets in real and simulation mode.



## Experiment No : 6

# Title : Configuration and Protocol in LAN

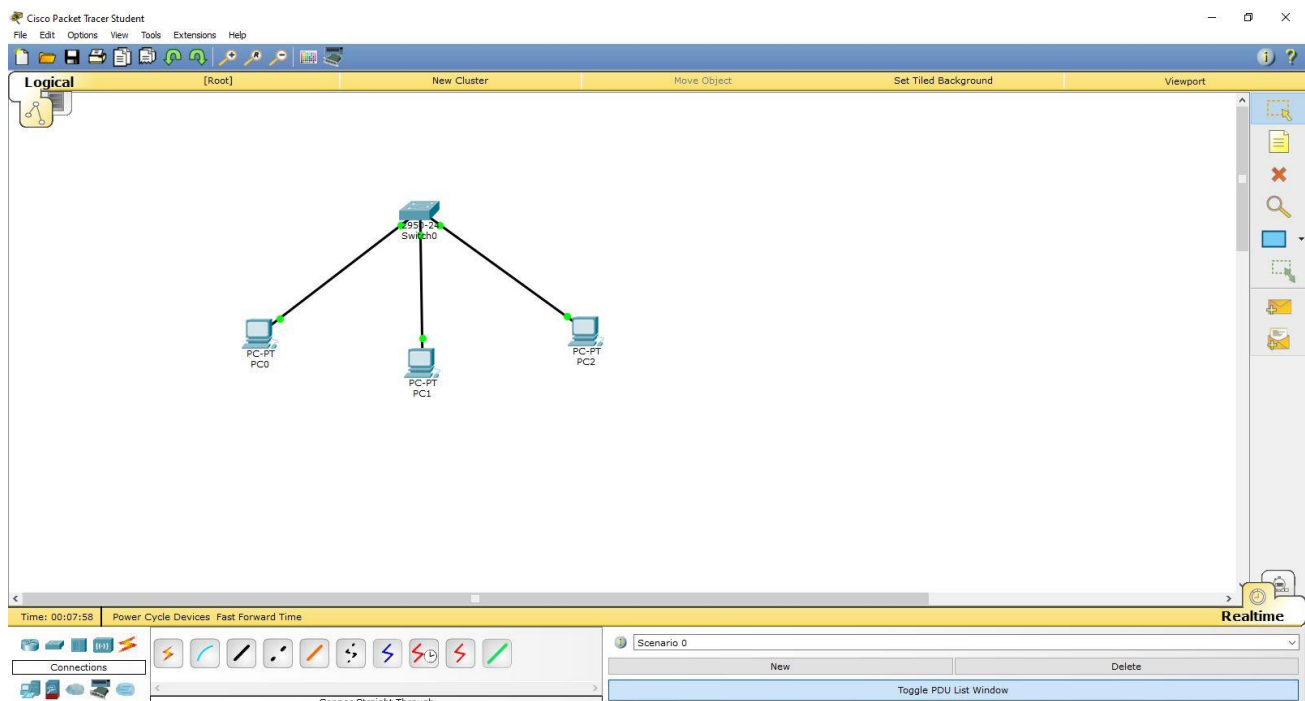
**Aim:** To analyse the performance of various configurations and protocols in LAN

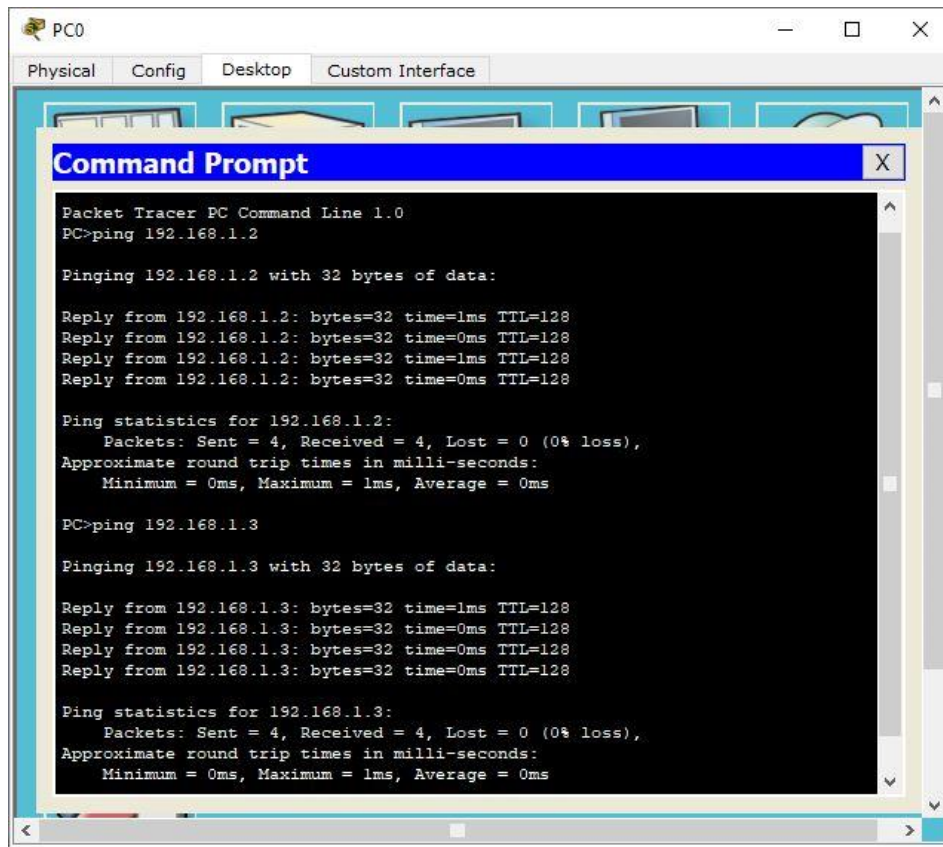
### Requirements

- Windows pc – 3 Nos
- CISCO Packet Tracer Software ( Student Version)
- 8 port switch – 1 No
- Cat-5 LAN cable

### Procedure

- Open the CISCO Packet tracer software
- Drag and drop 3 pcs using End Device Icons on the left corner
- Select 8 port switch from switch icon list in the left bottom corner
- Make the connections using Straight through Ethernet cables
- Give IP address of the PC1, PC2 and PC3 as 192.168.1.1, 192.168.1.2 and 192.168.1.3 respectively, ping between PCs and observe the transfer of data packets in real and simulation mode.





PC0	PC1	PC2
IP Address 10.0.0.1	IP Address 10.0.0.2	IP Address 10.0.0.2
Gateway 10.0.0.50	Gateway 10.0.0.50	Gateway 10.0.0.50

## Experiment No : 7

# Title: Network Simulation Using MATLAB

### Objective:

1. Simulate a simple computer network using MATLAB.
2. Analyze network performance metrics such as throughput, delay, and packet loss.
3. Visualize data transmission over the network.

### Tasks:

1. Design a Simple Network Topology Simulate a network with:
  - a. 1 Router
  - b. 2 Hosts (H1 and H2)
  - c. A communication link between H1  $\leftrightarrow$  Router  $\leftrightarrow$  H2
2. Key Parameters
  - a. Bandwidth (B): 10 Mbps
  - b. Packet size: 1 KB
  - c. Propagation delay (Tp): 10 ms
  - d. Simulation duration: 10 seconds
  - e. Traffic generation: Poisson process for packet arrival
3. Shows all Metrics Output:
  - a. Throughput: How much data is transmitted per second.
  - b. Average Delay: The average time a packet takes to travel from source to destination.
  - c. Packet Loss Rate: Percentage of dropped packets due to queue overflow.
4. Shows all Graphs:
  - a. Packet Arrivals: Visualize when packets arrive at the router.
  - b. Packet Transmissions: Show transmission times of packets.
  - c. Cumulative Packets Sent: Display the total number of packets sent over time.

### Code:

```
B = 10e6; % Bandwidth in bps (10 Mbps)
packet_size = 1e3 * 8; % Packet size in bits (1 KB)
Tp = 10e-3; % Propagation delay in seconds (10 ms)
simulation_duration = 10; % Simulation duration in seconds
lambda = 0.5; % Packet arrival rate (packets per second)

% Initialize variables
time = 0:0.01:simulation_duration; % Time vector
packet_arrivals = poissrnd(lambda, size(time)); % Poisson packet arrivals
total_packets = cumsum(packet_arrivals); % Cumulative packets sent
transmission_time = packet_size / B; % Transmission time for one packet
queue = []; % Initialize queue
packet_loss = 0; % Initialize packet loss counter
queue_size = 10; % Queue size

% Simulation loop
for t = 1:length(time)
    % Process arrivals
    for p = 1:packet_arrivals(t)
        if length(queue) < queue_size
            queue = [queue, time(t)]; % Add packet to queue
        else
            packet_loss = packet_loss + 1; % Increment packet loss
        end
    end
end
```

```

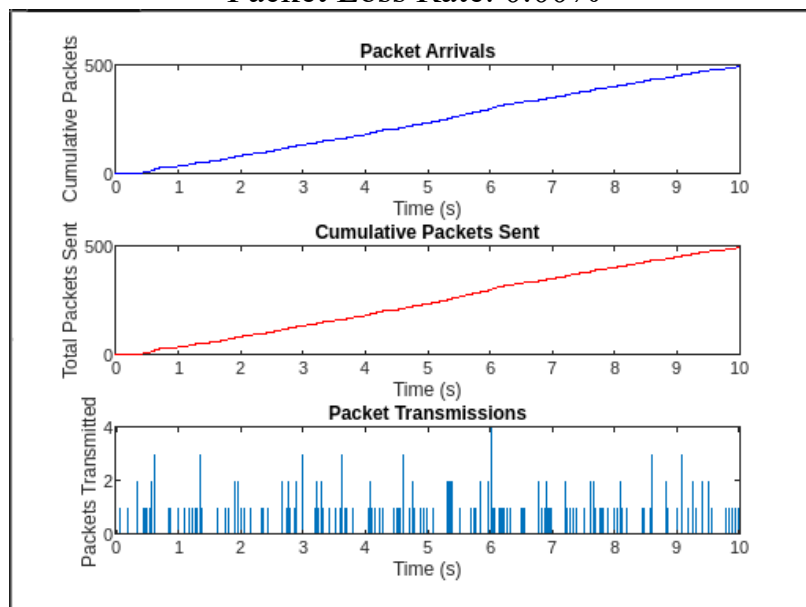
end

% Process transmission
if ~isempty(queue)
    % Check if the first packet can be transmitted
    if time(t) >= queue(1) + Tp + transmission_time
        queue(1) = []; % Remove the packet from the queue
    end
end
end

% Calculate metrics
throughput = (sum(total_packets) - packet_loss) * packet_size / simulation_duration; %
in bits/sec
average_delay = mean((queue + Tp + transmission_time) - time(1:length(queue))); %
Average delay
packet_loss_rate = (packet_loss / sum(total_packets)) * 100; % Packet loss rate in
percentage
% Display metrics
fprintf('Throughput: %.2f bps\n', throughput);
fprintf('Average Delay: %.2f seconds\n', average_delay);
fprintf('Packet Loss Rate: %.2f%%\n', packet_loss_rate);
% Visualization
figure;
subplot(3,1,1);
plot(time, cumsum(packet_arrivals), 'b');
title('Packet Arrivals');
xlabel('Time (s)');
ylabel('Cumulative Packets');
subplot(3,1,2);
plot(time, total_packets, 'r');
title('Cumulative Packets Sent');
xlabel('Time (s)');
ylabel('Total Packets Sent');
subplot(3,1,3);
bar(time, packet_arrivals);
title('Packet Transmissions');
xlabel('Time (s)');
ylabel('Packets Transmitted');

```

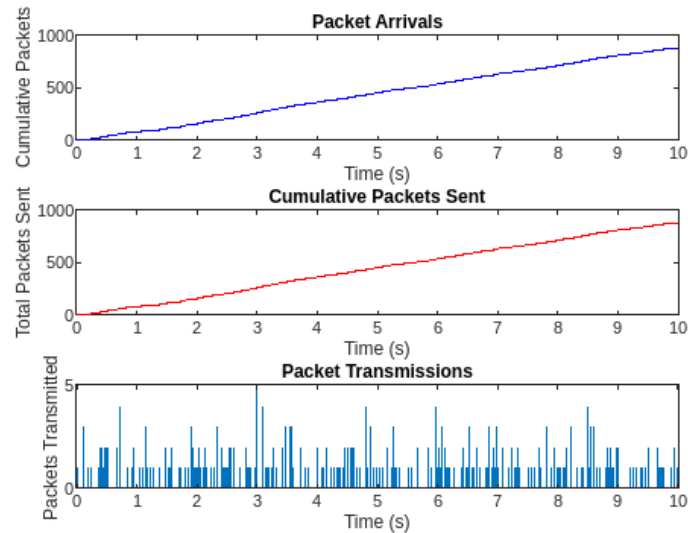
Throughput: 194945600.00 bps  
Average Delay: 10.00 seconds  
Packet Loss Rate: 0.00%



### 3. Assignment

Questions: 3.1.How does increasing the packet arrival rate ( $\lambda$ ) affect throughput and delay?

Lambda : 0.9

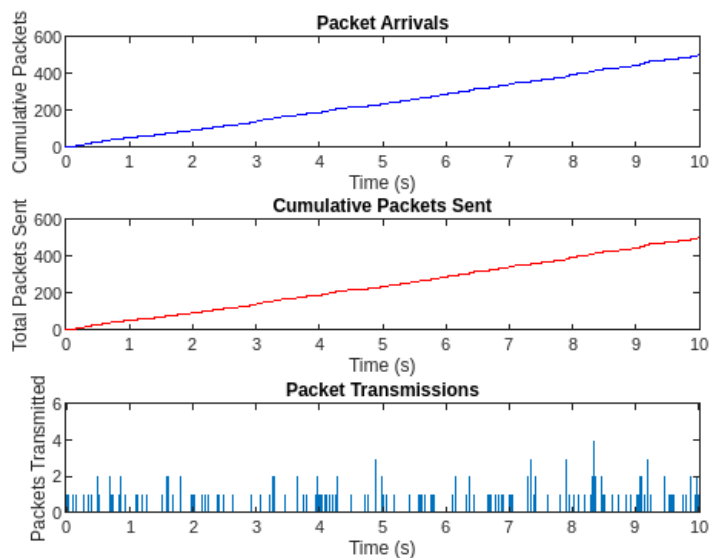


Throughput: 356957600.00 bps

Average Delay: 9.99 seconds

Packet Loss Rate: 0.00%

3.2.What happens to the packet loss rate when the queue size is reduced to 5?

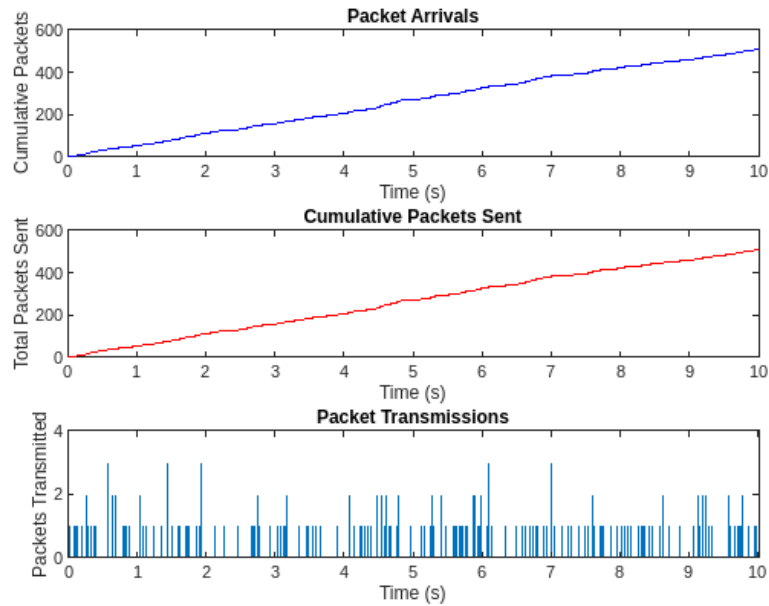


Throughput: 195692000.00 bps

Average Delay: 9.99 seconds

Packet Loss Rate: 0.00%

3.3.Modify the bandwidth to 100 Mbps. How does it impact the performance metrics?



## Part 2:

### Scenario: Multiple Links

The new network setup includes:

- 1 Router (R1) with multiple links.
- 2 Hosts (H1 and H2) connected to the router via separate links.
- A shared bandwidth and queue at the router to simulate contention.

Each link has its own propagation delay and bandwidth. The router handles packets from both hosts and forwards them according to their arrival time.

- Host 1 uses Link 1 with bandwidth B1 and propagation delay Tp1.
- Host 2 uses Link 2 with bandwidth B2 and propagation delay Tp2.

## Code

```
% Network Parameters
B1 = 10e6; % Bandwidth of Link 1 (10 Mbps)
B2 = 20e6; % Bandwidth of Link 2 (20 Mbps)
Tp1 = 5e-3; % Propagation Delay of Link 1 (5 ms)
Tp2 = 10e-3; % Propagation Delay of Link 2 (10 ms)
packet_size = 1500 * 8; % Packet size in bits (1500 bytes)
% Transmission Times
T1_transmission = packet_size / B1; % Transmission time for H1
T2_transmission = packet_size / B2; % Transmission time for H2
% Latency Calculation
latency_H1 = Tp1 + T1_transmission; % Latency for H1
latency_H2 = Tp2 + T2_transmission; % Latency for H2

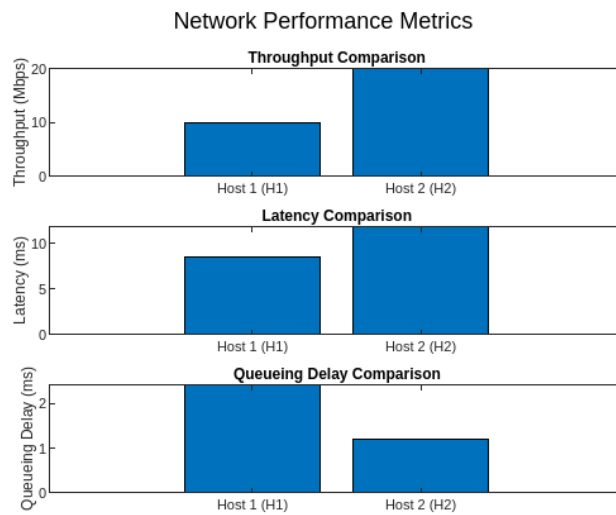
% Queueing Delay (Assuming 2 packets in queue)
queueing_delay_H1 = 2 * T1_transmission; % Queueing delay for H1
queueing_delay_H2 = 2 * T2_transmission; % Queueing delay for H2
% Total Latency
total_latency_H1 = latency_H1 + queueing_delay_H1; % Total latency for H1
total_latency_H2 = latency_H2 + queueing_delay_H2; % Total latency for H2
```



```

% Throughput
throughput_H1 = B1; % Throughput for H1
throughput_H2 = B2; % Throughput for H2
% Display Results
fprintf('Host 1 (H1):\n');
fprintf('Throughput: %.2f Mbps\n', throughput_H1 / 1e6);
fprintf('Latency: %.2f ms\n', total_latency_H1 * 1e3);
fprintf('Queueing Delay: %.2f ms\n\n', queueing_delay_H1 * 1e3);
fprintf('Host 2 (H2):\n');
fprintf('Throughput: %.2f Mbps\n', throughput_H2 / 1e6);
fprintf('Latency: %.2f ms\n', total_latency_H2 * 1e3);
fprintf('Queueing Delay: %.2f ms\n\n', queueing_delay_H2 * 1e3);
% Plotting
figure;
% Throughput Plot
subplot(3, 1, 1);
bar([throughput_H1, throughput_H2] / 1e6);
title('Throughput Comparison');
ylabel('Throughput (Mbps)');
set(gca, 'XTickLabel', {'Host 1 (H1)', 'Host 2 (H2)'});
% Latency Plot
subplot(3, 1, 2);
bar([total_latency_H1, total_latency_H2] * 1e3);
title('Latency Comparison');
ylabel('Latency (ms)');
set(gca, 'XTickLabel', {'Host 1 (H1)', 'Host 2 (H2)'});
% Queueing Delay Plot
subplot(3, 1, 3);
bar([queueing_delay_H1, queueing_delay_H2] * 1e3);
title('Queueing Delay Comparison');
ylabel('Queueing Delay (ms)');
set(gca, 'XTickLabel', {'Host 1 (H1)', 'Host 2 (H2)'});
% Adjust layout
sgtitle('Network Performance Metrics');

```



## 2. Do for private ip and public ip and do comparative analysis

```

% Network Parameters
B = 10e6; % Bandwidth (10 Mbps)
packet_size = 1e3; % 1 KB
Tp = 10e-3; % Propagation delay (10 ms)
sim_duration = 10; % Simulation duration in seconds

```

```
% For Private IPs (LAN simulation)
private_ip_bandwidth = B; % No external routing
private_ip_delay = Tp; % Minimal delay
private_ip_loss_rate = 0; % No packet loss for private IP

% For Public IPs (Internet simulation)
public_ip_bandwidth = B; % Public IP link bandwidth
public_ip_delay = Tp + 20e-3; % Higher delay due to external routing
public_ip_loss_rate = 0.05; % Simulate some packet loss

% Packet Arrival Simulation (Poisson process)
lambda_private = 0.5; % Packet arrival rate for private IPs
lambda_public = 0.5; % Packet arrival rate for public IPs

% Generate packets based on Poisson arrival process
num_packets_private = poissrnd(lambda_private * sim_duration);
num_packets_public = poissrnd(lambda_public * sim_duration);

% Simulate throughput, delay, and packet loss for Private IP
throughput_private = num_packets_private * packet_size / sim_duration;
delay_private = private_ip_delay;
packet_loss_private = private_ip_loss_rate * 100; % Convert to
percentage

% Simulate throughput, delay, and packet loss for Public IP
throughput_public = num_packets_public * packet_size / sim_duration * (1
- public_ip_loss_rate);
delay_public = public_ip_delay;
packet_loss_public = public_ip_loss_rate * 100; % Convert to percentage

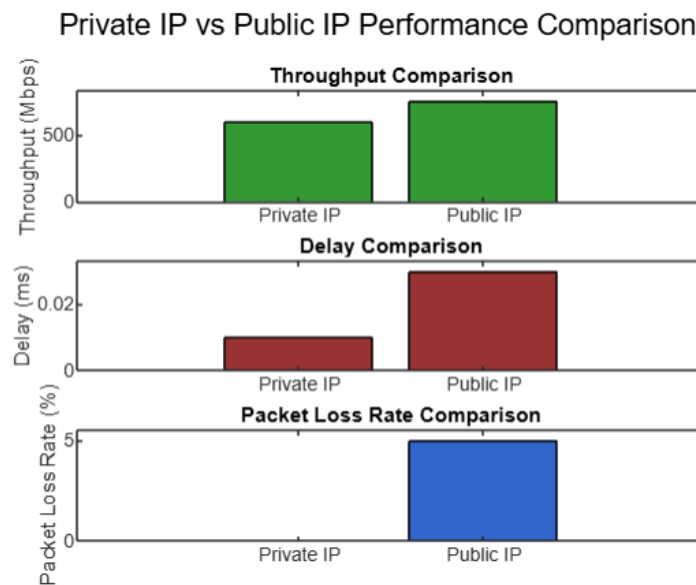
% Bar chart visualization
figure;

% Throughput comparison
subplot(3, 1, 1);
bar([1, 2], [throughput_private, throughput_public], 'FaceColor', [0.2,
0.6, 0.2]);
title('Throughput Comparison');
ylabel('Throughput (Mbps)');
xticks([1 2]);
xticklabels({'Private IP', 'Public IP'});
ylim([0 max([throughput_private, throughput_public])*1.1]);

% Delay comparison
subplot(3, 1, 2);
bar([1, 2], [delay_private, delay_public], 'FaceColor', [0.6, 0.2,
0.2]);
title('Delay Comparison');
ylabel('Delay (ms)');
xticks([1 2]);
xticklabels({'Private IP', 'Public IP'});
ylim([0 max([delay_private, delay_public])*1.1]);
```

```
% Packet Loss Rate comparison
subplot(3, 1, 3);
bar([1, 2], [packet_loss_private, packet_loss_public], 'FaceColor',
[0.2, 0.4, 0.8]);
title('Packet Loss Rate Comparison');
ylabel('Packet Loss Rate (%)');
xticks([1 2]);
xticklabels({'Private IP', 'Public IP'});
ylim([0 max([packet_loss_private, packet_loss_public])*1.1]);

% Adjust layout
sgtitle('Private IP vs Public IP Performance Comparison');
```



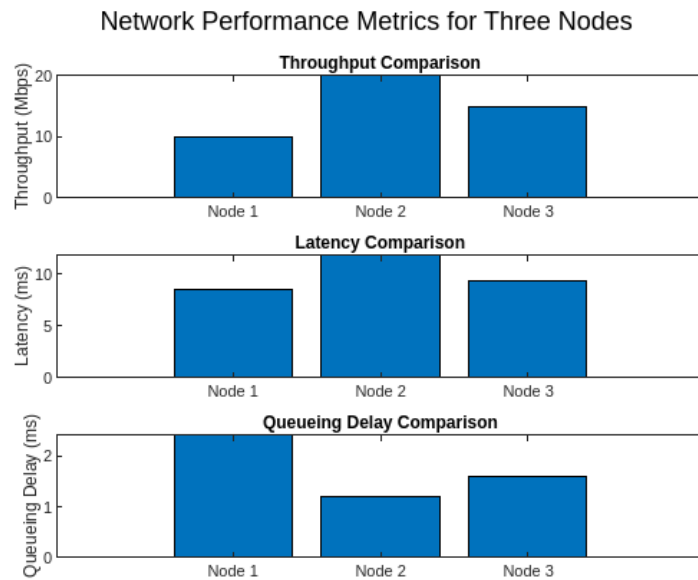
### 3. Simulate a Network with 1 switch and 3 nodes:

```
B1 = 10e6; % Bandwidth of Link 1 (10 Mbps) for Node 1
B2 = 20e6; % Bandwidth of Link 2 (20 Mbps) for Node 2
B3 = 15e6; % Bandwidth of Link 3 (15 Mbps) for Node 3
Tp1 = 5e-3; % Propagation Delay of Link 1 (5 ms)
Tp2 = 10e-3; % Propagation Delay of Link 2 (10 ms)
Tp3 = 7e-3; % Propagation Delay of Link 3 (7 ms)
packet_size = 1500 * 8; % Packet size in bits (1500 bytes)
T1_transmission = packet_size / B1; % Transmission time for Node 1
T2_transmission = packet_size / B2; % Transmission time for Node 2
T3_transmission = packet_size / B3; % Transmission time for Node 3
latency_N1 = Tp1 + T1_transmission; % Latency for Node 1
latency_N2 = Tp2 + T2_transmission; % Latency for Node 2
latency_N3 = Tp3 + T3_transmission; % Latency for Node 3
queueing_delay_N1 = 2 * T1_transmission; % Queueing delay for Node 1
queueing_delay_N2 = 2 * T2_transmission; % Queueing delay for Node 2
queueing_delay_N3 = 2 * T3_transmission; % Queueing delay for Node 3
total_latency_N1 = latency_N1 + queueing_delay_N1;
total_latency_N2 = latency_N2 + queueing_delay_N2;
total_latency_N3 = latency_N3 + queueing_delay_N3;
throughput_N1 = B1; % Throughput for Node 1
throughput_N2 = B2; % Throughput for Node 2
throughput_N3 = B3; % Throughput for Node 3
```

```

figure;
subplot(3, 1, 1);
bar([throughput_N1, throughput_N2, throughput_N3] / 1e6);
title('Throughput Comparison');
ylabel('Throughput (Mbps)');
set(gca, 'XTickLabel', {'Node 1', 'Node 2', 'Node 3'});
subplot(3, 1, 2);
bar([total_latency_N1, total_latency_N2, total_latency_N3] * 1e3);
title('Latency Comparison');
ylabel('Latency (ms)');
set(gca, 'XTickLabel', {'Node 1', 'Node 2', 'Node 3'});
subplot(3, 1, 3);
bar([queueing_delay_N1, queueing_delay_N2, queueing_delay_N3] * 1e3);
title('Queueing Delay Comparison');
ylabel('Queueing Delay (ms)');
set(gca, 'XTickLabel', {'Node 1', 'Node 2', 'Node 3'});
sgtitle('Network Performance Metrics for Three Nodes');

```



## Experiment No : 8

# Title: Pure ALOHA and Slotted ALOHA

### Objective:

1. Simulate a simple computer network using MATLAB.
2. Analyze the performance of pure ALOHA and slotted ALOHA.
3. Visualize the performance.

### Code :

```
% Simulation Parameters
numNodes = 100;           % Number of nodes
numSlots = 1000;         % Total number of time slots
numTrials = 10;           % Number of trials for averaging results
transmissionProb = 0.1;   % Probability of transmission per slot

% Run Simulations
[alohaThroughput, alohaCollisionRate] = runAlohaSimulation(numNodes, numSlots, numTrials,
transmissionProb);
[slottedThroughput, slottedCollisionRate] = runSlottedAlohaSimulation(numNodes, numSlots,
numTrials, transmissionProb);

% Display Results
fprintf('ALOHA Throughput: %.2f%%\n', alohaThroughput * 100);
fprintf('ALOHA Collision Rate: %.2f%%\n', alohaCollisionRate * 100);
fprintf('Slotted ALOHA Throughput: %.2f%%\n', slottedThroughput * 100);
fprintf('Slotted ALOHA Collision Rate: %.2f%%\n', slottedCollisionRate * 100);

% Plot Results
figure;
bar([alohaThroughput, slottedThroughput; alohaCollisionRate, slottedCollisionRate] * 100);
set(gca, 'XTickLabel', {'Throughput', 'Collision Rate'});
legend({'ALOHA', 'Slotted ALOHA'});
ylabel('Percentage');
title('Comparison of ALOHA and Slotted ALOHA');

disp('Simulation Complete');

% Function to simulate ALOHA
function [throughput, collisionRate] = runAlohaSimulation(numNodes, numSlots, numTrials,
transmissionProb)
    successfulTransmissions = 0;
    collisions = 0;
    for trial = 1:numTrials
        for t = 1:numSlots
            transmittingNodes = rand(1, numNodes) < transmissionProb;
            if sum(transmittingNodes) == 1
                successfulTransmissions = successfulTransmissions + 1;
            elseif sum(transmittingNodes) > 1
                collisions = collisions + 1;
            end
        end
    end
    throughput = successfulTransmissions / (numSlots * numTrials);
    collisionRate = collisions / (numSlots * numTrials);
end
```

```
% Function to simulate Slotted ALOHA
```

```
function [throughput, collisionRate] = runSlottedAlohaSimulation(numNodes, numSlots, numTrials,  
transmissionProb)
```

```
    successfulTransmissions = 0;
```

```
    collisions = 0;
```

```
    for trial = 1:numTrials
```

```
        for t = 1:numSlots
```

```
            transmittingNodes = rand(1, numNodes) < transmissionProb;
```

```
            if sum(transmittingNodes) == 1
```

```
                successfulTransmissions = successfulTransmissions + 1;
```

```
            elseif sum(transmittingNodes) > 1
```

```
                collisions = collisions + 1;
```

```
            end
```

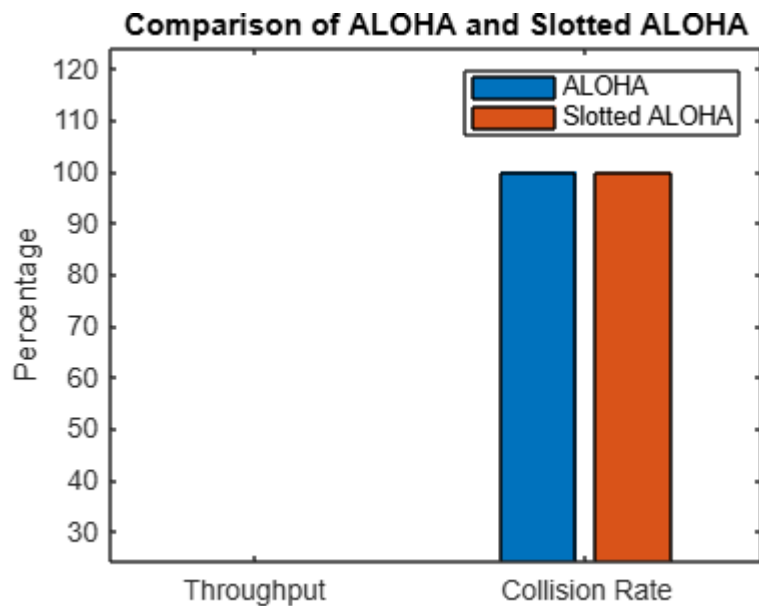
```
        end
```

```
    end
```

```
    throughput = successfulTransmissions / (numSlots * numTrials);
```

```
    collisionRate = collisions / (numSlots * numTrials);
```

```
end
```



#### Command Window

```
ALOHA Throughput: 0.02%
```

```
ALOHA Collision Rate: 99.98%
```

```
Slotted ALOHA Throughput: 0.03%
```

```
Slotted ALOHA Collision Rate: 99.97%
```

```
Simulation Complete
```

```
..
```

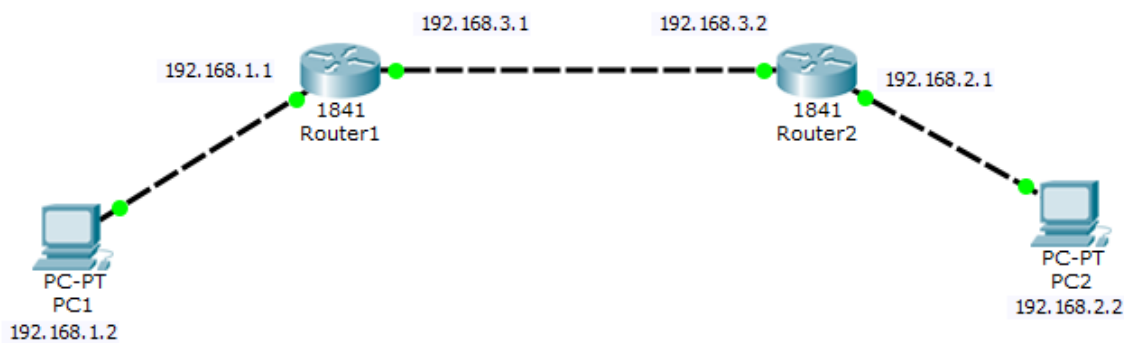
## Experiment No : 9

# Title: Static Routing

### Objective : Implement static routing in Cisco Packet Tracer.

Routing is simply a process of choosing route for delivering data to its destination. All hosts who can generate a routing table can do a routing. Routing process is needed when we are going to deliver packets of data to a network that isn't directly connected with the sender.

For easier configuration process, let's send the data between 2 PCs across just 2 routers like the following network topology.



### Step 1 : Place the routers.

First thing first, we need to place the routers on the workspace. We can pick one of them from *Network Devices > Routers > 4331* in bottom panel.

Any type of routers will do, 4331 is just the nearest to our cursor's current position.



Retrack the same step for another router, or copy paste the existing one.

### Step 2 : Place the PCs.

PC can be picked from *End Devices > End Devices > PC*. Place them on the workspace.



### Step 3 : Connect all hosts physically, and accordingly.

We can pick one of the physical media from *Connections > Connections*.



We are going to use ethernet connection, so we pick between **Straight-Through** or **Cross-Over** copper cable (choose **Straight** if unsure).



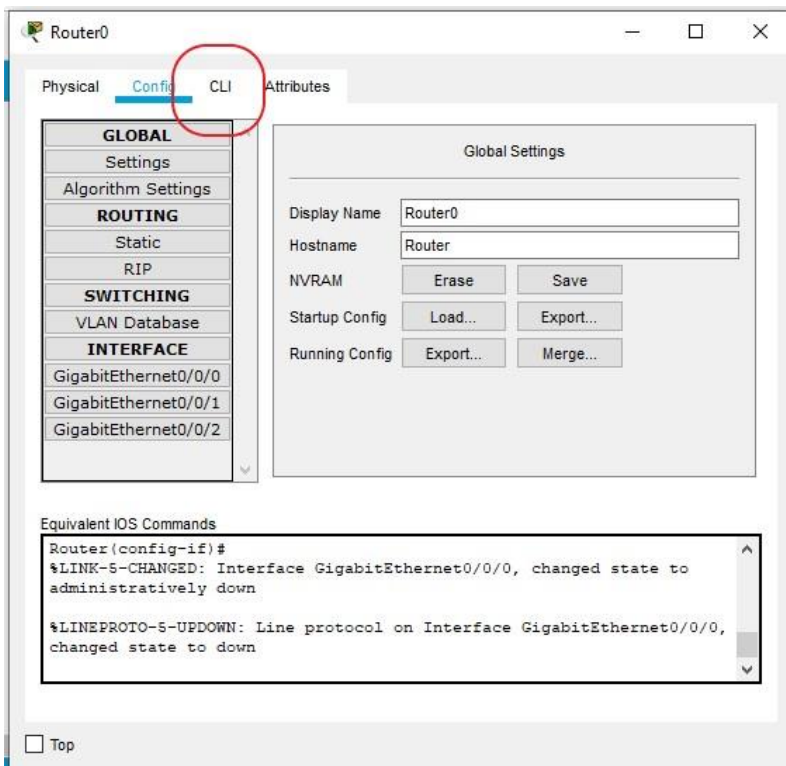
Next, click on one of the hosts, and choose one of the available ethernet ports. Then click on the another host while we see the cable being dragged.



Repeat the similar steps for connecting a PC and one of the routers.

## Step 4 : Configure the routers.

Click on a router to bring up a pop-up window. Then open CLI tab.



**In Router0 CLI, paste the following script.**

```
en
conf t
int FastEthernet0/0
ip address 192.168.1.1 255.255.255.0
no shutdown
exit
int FastEthernet0/1
ip address 192.168.3.1 255.255.255.248
no shutdown
exit
ip route 192.168.2.2 255.255.255.0 192.168.3.2
exit
```

**And the following into Router1.**

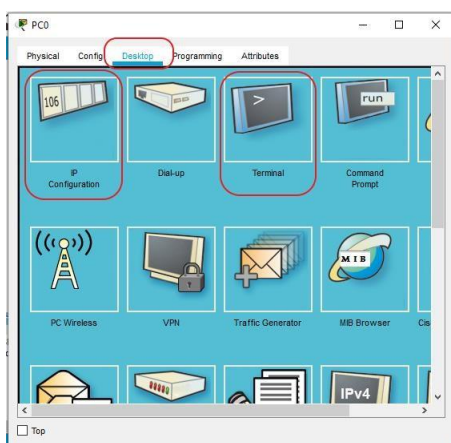
```
en
conf t
int FastEthernet0/0
ip address 192.168.3.2 255.255.255.248
no shutdown
exit
int FastEthernet0/1
ip address 192.168.2.1 255.255.255.0
no shutdown
exit
ip route 192.168.1.0 255.255.255.0 192.168.3.1
exit
```

Type “no” for answering the following question beforehand if popping up.

*Would you like to enter the initial configuration dialog? [yes/no]:*

**Step 5 : Configure the PCs.**

There is Desktop tab in each PC's pop-up window. We are going to use IP Configuration and Command Prompt application here.



Choose IP Configuration to do IP configurations, and insert the following information.

### For PC0.

IP Address : 192.168.1.2

Netmask : 255.255.255.0

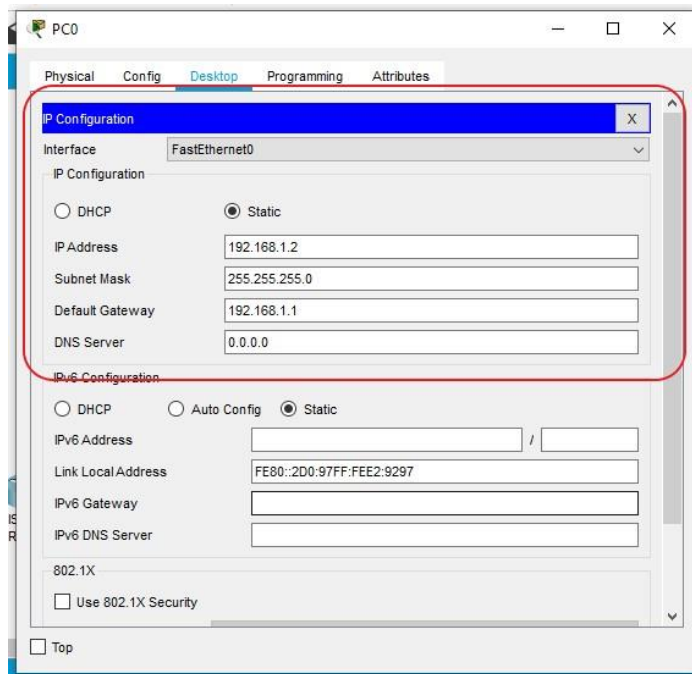
Gateway : 192.168.1.1

### For PC1.

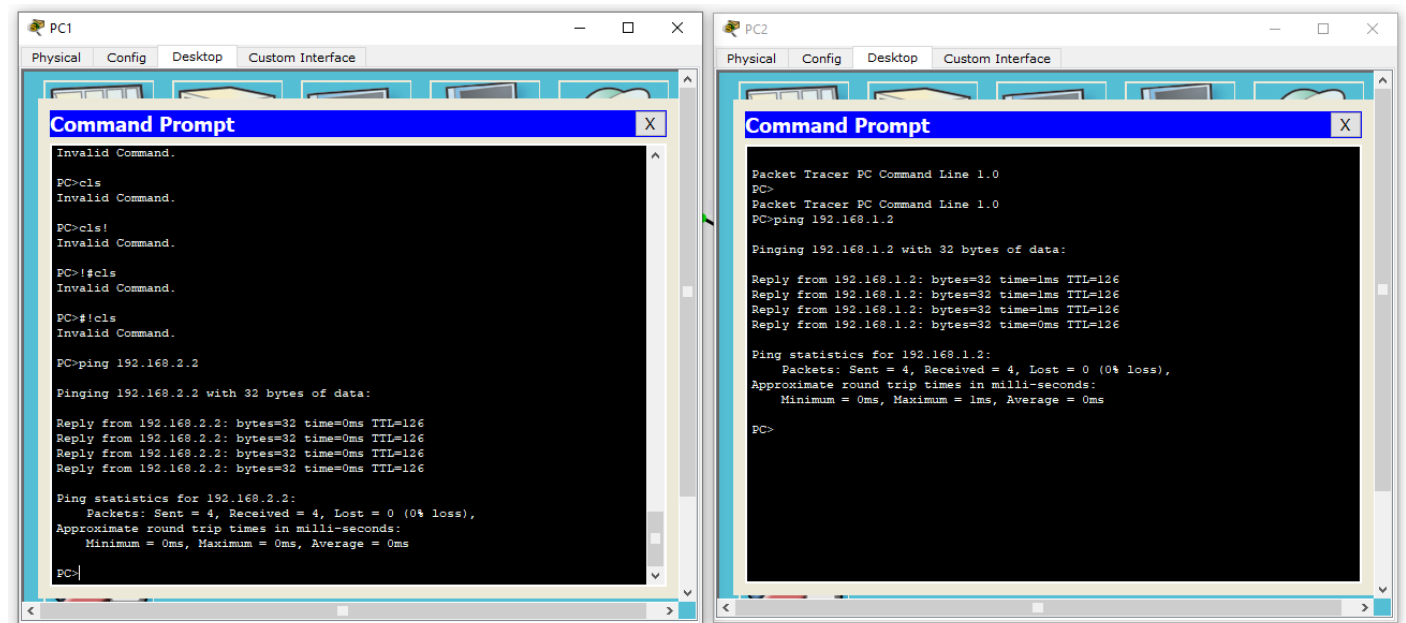
IP Address : 192.168.2.2

Netmask : 255.255.255.0

Gateway : 192.168.2.1



### Step 6 : Ping test between the two PCs.



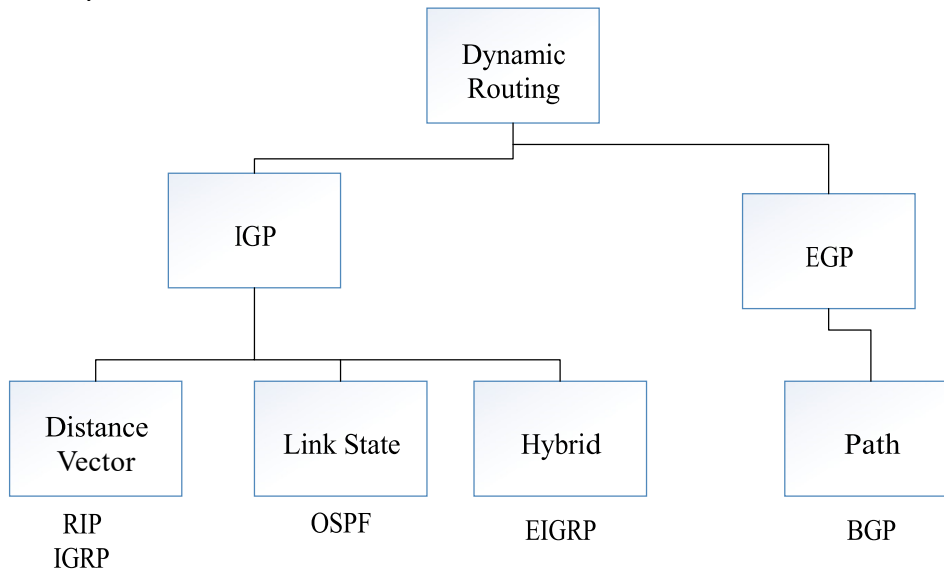
## Experiment No : 10

### Title: Dynamic Routing Using Cisco Packet Tracer: RIP, OSPF, and BGP

Dynamic routing is a networking technique that provides optimal data routing. Unlike static routing, dynamic routing enables routers to select paths according to real-time logical network layout changes.

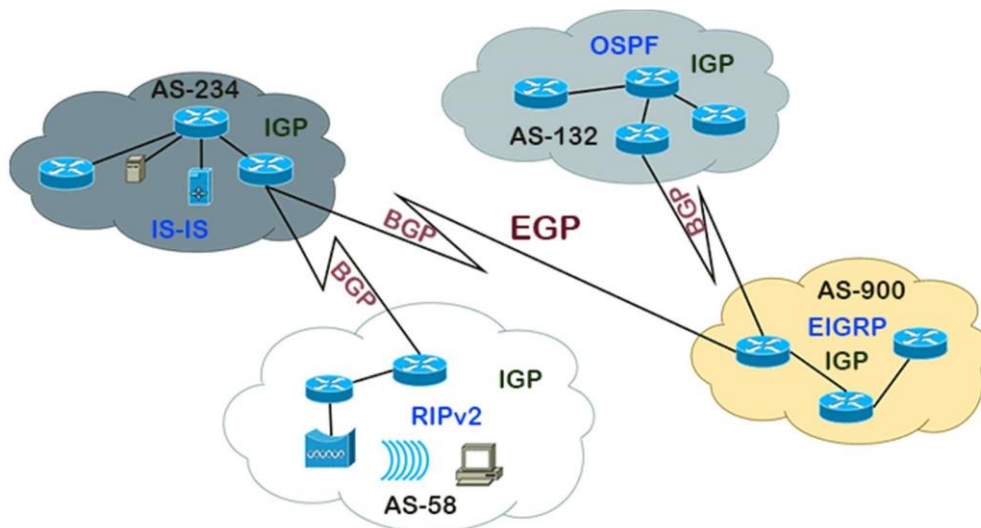
#### Advantages of Dynamic Routing

- Allows the exchange of routing information whenever the network experiences a change in topology.
- Since the routes do not have to be configured manually, there is less administrative overhead.
- Less error-prone than static routing.
- Allows scalability since there is less administrative overhead involved.



Classification of Dynamic routing

- An Autonomous System is a group of networks that is governed and controlled by a single administrative entity. For example, a network created by a single company, organization, corporation, or ISP is a single AS.
- An AS number is a unique identity of the AS on the Internet. If you want to connect your AS to the Internet, you must obtain an AS number. Internet Assigned Numbers Authority (IANA) has the worldwide right to assign AS numbers. It delegates that right to the organizations that assign public IP addresses. For example, in Asia, the Asia Pacific Network Information Centre (APNIC) assigns both IP addresses and AS numbers.
- An AS can contain multiple networks. To connect these networks, the administrator can use a routing protocol. The routing protocol the administrator uses to connect the networks within the autonomous system is known as an interior gateway routing protocol. Since all networks in an autonomous system belong to the same administrative entity, the administrator can configure any IGP protocol to connect them.
- RIPv1, IGRP, OSPF, EIGRP, RIPv2, and IS-IS are some examples of interior gateway routing protocols.
- An EGP protocol provides connectivity between different autonomous systems. Since different autonomous systems belong to different administrative entities, administrators cannot use routing protocols of their choices to connect them. They have to use a uniform routing protocol. A uniform routing protocol that connects different autonomous systems is known as an exterior gateway routing protocol.
- Nowadays, BGP is the only used exterior routing protocol. BGP connects all public autonomous systems on the Internet.



Schematic of routing protocols

### Distance Vector Routing

As the name implies, distance vector routing uses the parameters: distance and direction to calculate the best possible path to forward the packet from source to destination. Distance is calculated by counting the numbers of routers in between the source and destination. Every router counts its adjacent router as a next-hop. The minimum number of hop count between the source router and the destination router makes the best possible path for routing.

Distance vector routing uses the Bellman-Ford Algorithm or Ford-Fulkerson algorithms to choose the best path. In distance vector routing, routing advertisements are received from the directly connected or the neighbour router and update its routing information periodically. The update process continues router to router till the destination is reached. It shares entire routing table information to its immediate neighbour router.

Examples of distance vector routing protocols are RIP v1, RIPv2, and IGRP.

### Link State Routing

Unlike distance-vector, link-state routing does not rely on the neighbour router for routing information. Instead, it maps the overall state of the entire network that includes the topology, bandwidth, traffic congestion in a particular link, etc.

In link-state routing, each router retrieves information about itself, its directly connected link, and the state of that link. This information is passed from router to router, without changing the entire routing information of the adjacent router. Each router makes only a copy of the information received from its neighbour. Thus, every router has identical information and can independently calculate its best path for forwarding packets from source to destination.

Link State routing uses Dijkstra's Algorithm. It is also referred to as the Shortest Path First (SPF) Algorithm.

Although link-state is more complex than distance-vector, however, it is more reliable and minimizes the shortcomings of distance vector routing. The basic functionality of link-state routing are:

- Every router establishes an adjacency relationship with its neighbours.
- Link state advertisements (LSAs) are sent by each router to its neighbour
- Each router stores a copy of all the LSAs as a database. These databases in all routers are identical.
- Each router maintains a complete topological database, also called the link-state database.

- Each router calculates the shortest path to each network using Dijkstra's Algorithm and enters this information into the route table.

Examples of link-state routing are OSPF, IS-IS, etc.

### **Hybrid Routing**

It is a combination of both distance vector and link-state routing. Basically, hybrid routing is considered to be the advanced distance-vector, however, it also contains some of the features of the link-state routing protocol.

An example of a hybrid routing protocol is EIGRP. Enhanced Interior Gateway Routing Protocol has the features of both the distance vector and the link-state routing. It does not send Link state advertisement, but it sends traditional distance vector routing information to the neighbour router. It synchronizes with the routing updates of the neighbour router. After that, it sends a specific link-state update to its neighbour.

### **10.a**

**Objective : To understand the concept and operation of Routing Information Protocol (RIP)**

#### **Requirements**

- Windows pc – 2 Nos
- CISCO Packet Tracer Software (Student Version)
- 8 port switch – 2 No
- Router – 2 Nos
- Cat-5 LAN cable

#### **Procedure**

- Open the CISCO Packet tracer software
- Drag and drop 5 pcs using End Device Icons on the left corner
- Select 8 port switch from switch icon list in the left bottom corner
- Select Routers and Give the IP address for serial ports of router and apply clock rate as per the table.
- Make the connections using Straight through Ethernet cables
- Ping between PCs and observe the transfer of data packets in real and simulation mode.

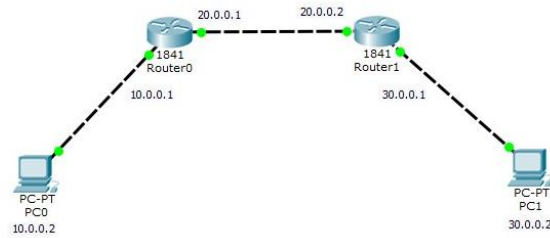
### **Theory**

RIP (Routing Information Protocol) is one of the oldest distance vector routing protocols. It is usually used on small networks because it is very simple to configure and maintain, but lacks some advanced features of routing protocols like OSPF or EIGRP. Two versions of the protocol exists: version 1 and version 2. Both versions use hop count as a metric and have the administrative distance of 120. RIP version 2 is capable of advertising subnet masks and uses multicast to send routing updates, while version 1 doesn't advertise subnet masks and uses broadcast for updates. Version 2 is backwards compatible with version 1. RIPv2 sends the entire routing table every 30 seconds, which can consume a lot of bandwidth. RIPv2 uses multicast address of 224.0.0.9 to send routing updates, supports authentication and triggered updates (updates that are sent when a change in the network occurs).

#### **Downsides of RIP:**

- RIP sends the entire routing table every 30 seconds, which can consume a lot of network bandwidth.
- It lacks some more advanced of features of the newer routing protocols like OSPF or EIGRP and is not widely used in modern networks. For example, RIP doesn't support route summarization.

## Network Topology Diagram for RIP 1. Build the network topology.



### 2. Configure IP addresses on the PCs and the routers.

Router 1

enable

configure terminal

hostname Router0

interface FastEthernet0/0

ip address 10.0.0.1 255.255.255.0

no shutdown

exit

interface Serial0/0/0

ip address 20.0.0.1 255.255.255.252

clock rate 64000

no shutdown

exit

exit

write memory

Router 2

enable

configure terminal

hostname Router1

interface FastEthernet0/0

ip address 30.0.0.1 255.255.255.0

no shutdown

exit

interface Serial0/0/0

ip address 20.0.0.2 255.255.255.252

no shutdown

exit

exit

write memory

## IP configuration on PCs

Click PC->Desktop->IP Configuration. On each PC assign these addresses:

PC1: IP address: 10.0.0.2 Subnet mask 255.0.0.0 Default Gateway 10.0.0.1

PC2: IP address: 30.0.0.2 Subnet mask 255.0.0.0 Default Gateway 30.0.0.1

### 3. Configure RIPv2 on the routers

#### Router 1

```
enable
configure terminal
router rip
version 2
network 10.0.0.0
network 20.0.0.0
no auto-summary
exit
write memory
```

#### Router 2

```
enable
configure terminal
router rip
version 2
network 30.0.0.0
network 20.0.0.0
no auto-summary
exit
write memory
```

As you can see, to configure rip on each router, we enable RIP using router rip command then advertise the networks directly connected to the router interfaces using network command.

That's all for RIP configuration.

### 4. We'll now verify RIP configuration.

To verify that RIP is indeed advertising routes, we can use the show ip route command on R1.

```
Router>show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

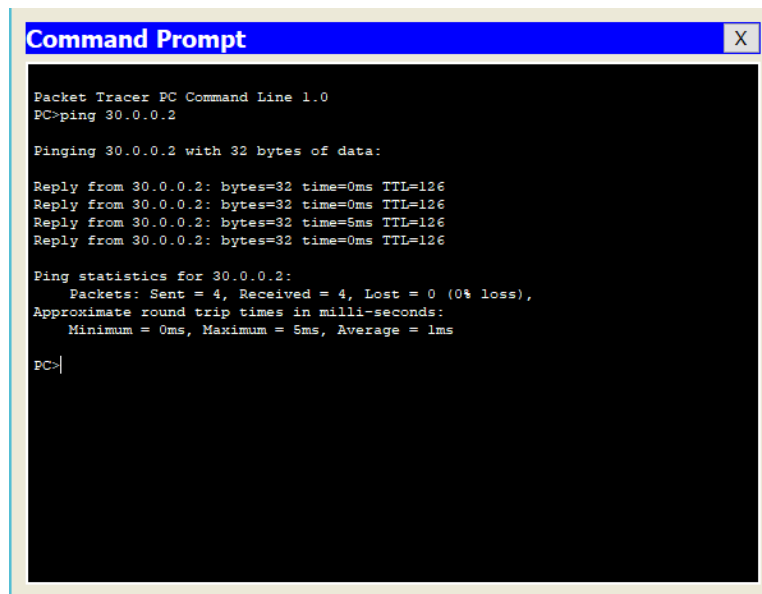
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    20.0.0.0/8 is directly connected, FastEthernet0/1
R    30.0.0.0/8 [120/1] via 20.0.0.2, 00:00:13, FastEthernet0/1
-    |
```



You can see that R1 has learned about the 30.0.0/8 network. The letter R indicates that the route was learned using RIP. Note the administrative distance of 120 and the metric of 1 in the [120/1] part.

To specifically display routes learnt through RIP use show ip route rip command on the router.

Now lets Ping PC2 from PC1 to further confirm that connectivity is really established between the two subnets.



```

Command Prompt

Packet Tracer PC Command Line 1.0
PC>ping 30.0.0.2

Pinging 30.0.0.2 with 32 bytes of data:

Reply from 30.0.0.2: bytes=32 time=0ms TTL=126
Reply from 30.0.0.2: bytes=32 time=0ms TTL=126
Reply from 30.0.0.2: bytes=32 time=5ms TTL=126
Reply from 30.0.0.2: bytes=32 time=0ms TTL=126

Ping statistics for 30.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 5ms, Average = 1ms

PC>
    
```

How do we configure rip? Specify the commands.

```

#Router rip
#network 10.0.0.0
    
```

Which command is used to check RIP routing?

```

#show ip route
#show ip protocols
    
```

## 10.b

**Configuration of Open shortest Path First (OSPF) Algorithm : To construct multiple router networks and understand the operation of OSPF Protocol.**

### Requirements

- Windows pc – 3 Nos
- CISCO Packet Tracer Software (Student Version)
- 8 port switch – 3 No
- Router – 3 Nos
- Cat-5 LAN cable

### Procedure

- Open the CISCO Packet tracer software.
- Drag and drop 5 pcs using End Device Icons on the left corner.
- Select 8 port switch from switch icon list in the left bottom corner.
- Select Routers and Give the IP address for serial ports of router and apply clock rate.
- Add HWIC -2T Peripheral to all routers, type CLI's for all routers.
- Make the connections using Straight through Ethernet cables.

- Ping between PCs and observe the transfer of data packets in real and simulation mode.

## Theory

The OSPF routing protocol has largely replaced the older Routing Information Protocol (RIP) in corporate networks. Using OSPF, a router that learns of a change to a routing table (when it is reconfigured by network staff, for example) or detects a change in the network immediately multicasts the information to all other OSPF hosts in the network so they will all have the same routing table information. Unlike RIP, which requires routers to send the entire routing table to neighbours every 30 seconds, OSPF sends only the part that has changed and only when a change has taken place. When routes change -- sometimes due to equipment failure - the time it takes OSPF routers to find a new path between endpoints with no loops (which is called "open") and that minimizes the length of the path is called the convergence time.

## Input Details for OSPF

PC0	PC1	PC2
IP Address : 192.168.1.2 Gate way : 192.168.1.1	IP Address: 192.168.2.2 Gate way : 192.168.2.1	IP Address: 192.168.3.2 Gate way : 192.168.3.1

Router 0	Router 1	Router 2
fa 0/0 IP Address: 192.168.1.1 Serial 0/0/0 : 10.0.0.1 @ 2000000 clock rate Serial 0/0/1 : -	fa 0/0 IP Address : 192.168.2.1 Serial 0/0/0 : 10.0.0.2 Serial 0/0/1 : - @ 2000000 clock rate	Fa 0/0 IP Address : 192.168.3.1 Serial 0/0/0 : 10.0.0.2 @ no clock rate Se 0/0/1 : 11.0.0.1

Commands to Configuring OSPF:

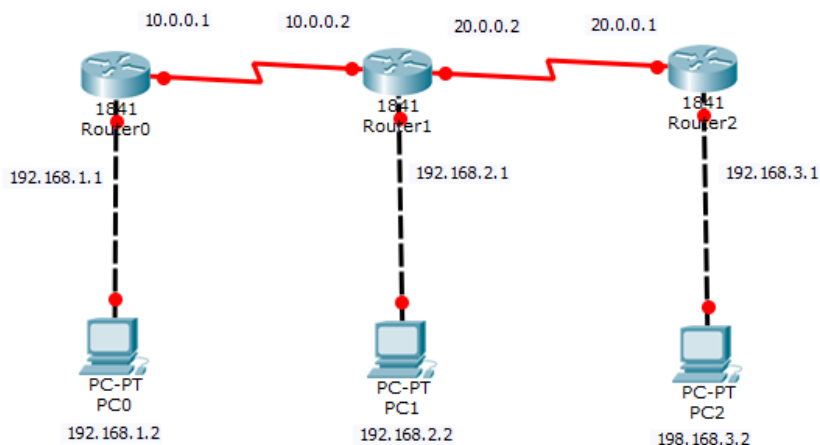
(config)# router ospf<process ID>

(config-router)# network<network ID><wildcard mask>area<area ID>

## Implement the OSPF Single Area Network:

Step 1: Initialize the interface and host with IP addresses and default gateway respectively:

- Network topology consists of 3 Host, 3 Interfaces, and 3 switches
- HOST1: IP 192.168.1.2, Default gateway: 192.168.1.1
- HOST2: IP 192.168.2.2, Default gateway: 192.168.2.1
- HOST3: IP 192.168.3.2, Default gateway: 192.168.3.1



As we can see we have configured interface 1 (Router0) with Host 1 which is PC0 and the Serial port.

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#hostname Router0
Router0(config)#interface FastEthernet0/0
Router0(config-if)#ip address 192.168.1.1 255.255.255.0
Router0(config-if)#no shutdown

Router0(config-if)#exit
Router0(config)#
Router0(config)#interface Serial0/0/0
Router0(config-if)#ip address 10.0.0.1 255.255.255.252
Router0(config-if)#no shutdown

%LINK-5-CHANGED: Interface Serial0/0/0, changed state to down
Router0(config-if)#exit
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

Router0(config)#interface FastEthernet0/0
Router0(config-if)#
Router0(config-if)#exit
Router0(config)#interface FastEthernet0/1
Router0(config-if)#
Router0(config-if)#exit
Router0(config)#interface Serial0/0/0
Router0(config-if)#
%LINK-5-CHANGED: Interface Serial0/0/0, changed state to up
```

Step 2: Configuring the Interface 2 which is router1.

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#hostname Router1
Router1(config)#
Router1(config)#interface FastEthernet0/0
Router1(config-if)#ip address 192.168.2.1 255.255.255.0
Router1(config-if)#no shutdown

Router1(config-if)#exit
Router1(config)#
Router1(config)#interface Serial0/0/0
Router1(config-if)#ip address 10.0.0.2 255.255.255.252
Router1(config-if)#no shutdown

Router1(config-if)#exit
Router1(config)#
Router1(config)#interface Serial0/0/1
Router1(config-if)#ip address 20.0.0.2 255.255.255.252
Router1(config-if)#no shutdown

%LINK-5-CHANGED: Interface Serial0/0/1, changed state to down
Router1(config-if)#exit
Router1(config)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINK-5-CHANGED: Interface Serial0/0/0, changed state to up
```

Step 3: Configuring the Interface 3 which is router2.

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname Router2
Router2(config)#
Router2(config)#interface FastEthernet0/0
Router2(config-if)#ip address 192.168.3.1 255.255.255.0
Router2(config-if)#no shutdown

Router2(config-if)#exit
Router2(config)#
Router2(config)#interface Serial0/0/0
Router2(config-if)#ip address 20.0.0.1 255.255.255.252
Router2(config-if)#no shutdown

Router2(config-if)#exit
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINK-5-CHANGED: Interface Serial0/0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0/0, changed state to up
|
```

Now comes the main part now we have to configure the OSPF implementation:

Step 1: Configure the Router0 and create router OSPF1 and then add network id with wildcard mask.

```
Router0(config)#router ospf 1
Router0(config-router)#network 192.168.1.0 0.0.0.255 area 0
Router0(config-router)#network 10.0.0.0 0.0.0.3 area 0
Router0(config-router)#exit
Router0(config)#
```

Step2: Configure the Router1 and create router OSPF 1 and then add network id with wildcard mask.

```
Router1(config-if)#router ospf 1
Router1(config-router)#network 192.168.2.0 0.0.0.255 area 0
Router1(config-router)#network 10.0.0.0 0.0.0.3 area 0
Router1(config-router)#network 20.0.0.0 0.0.0.3 area 0
Router1(config-router)#exit|
```

Step 3: Configure the Router2 and create router OSPF 1 and then add network id with wildcard mask.

```
Router2(config)#
Router2(config)#router ospf 1
Router2(config-router)#network 192.168.3.0 0.0.0.255 area 0
Router2(config-router)#network 20.0.0.0 0.0.0.3 area 0
Router2(config-router)#exit
_
```

- Verifying the OSPF routing

OSPF shares routing information only with neighbors. We use the `show ip ospf neighbor` command to verify OSPF neighbors. The following image shows the output of this command on R1.

The output includes the following fields.

Neighbor ID

This field displays the RID of the neighbor.

State

This field displays the convergency state. An OSPF router goes through seven states to reach convergency. The Full state in this field verifies the router has reached convergency with the router listed in the neighbor ID field.

### Interface

This field displays the local interface connected to the neighbor.

### Address

This field displays the IP address of the neighbor.

### Dead time

This field displays the dead interval.

- Viewing OSPF routes

The show ip route ospf command displays a list of all OSPF routes in the routing table.

```
Router1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

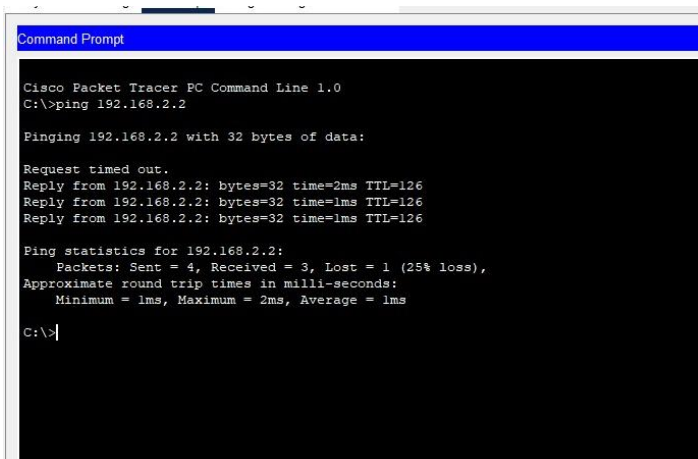
Gateway of last resort is not set

10.0.0.0/30 is subnetted, 1 subnets
C       10.0.0.0 is directly connected, Serial0/0/0
20.0.0.0/30 is subnetted, 1 subnets
C       20.0.0.0 is directly connected, Serial0/0/1
Router1#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.3.1	0	FULL/ -	00:00:34	20.0.0.1	Serial0/0/1
192.168.1.1	0	FULL/ -	00:00:34	10.0.0.1	Serial0/0/0

- Testing connectivity between end devices

We can test connectivity between end devices to verify the OSPF configuration on all routers. Send ping requests from PC to Server. If it gets replies, it verifies the OSPF configuration. We can also use the tracert command to print the path the data packets take to reach the destination.



```
Command Prompt
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.2.2: bytes=32 time=2ms TTL=126
Reply from 192.168.2.2: bytes=32 time=1ms TTL=126
Reply from 192.168.2.2: bytes=32 time=1ms TTL=126

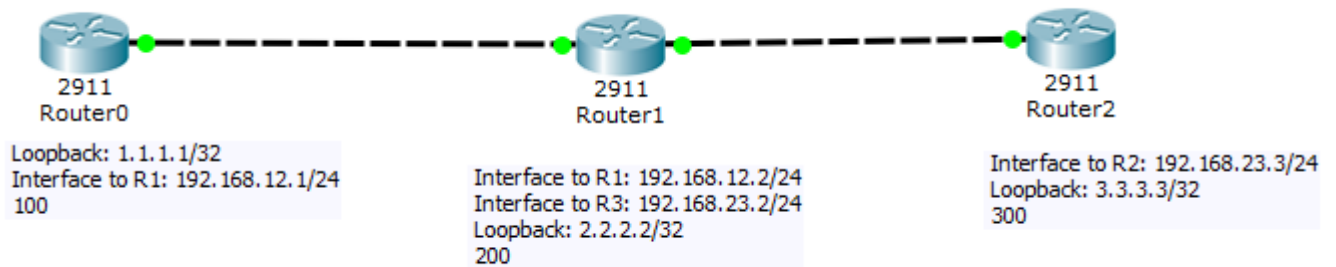
Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms

C:\>
```

**10.c****Configure BGP using packet tracer**

BGP (Border Gateway Protocol) is the core routing protocol of the Internet. It is described as a path vector protocol as BGP does not use traditional IGP (OSPF, EIGRP, RIP) metrics, but makes routing decisions based on path, network policies and/or rulesets. It maintains a table of IP networks or 'prefixes' which designate network reachability among autonomous systems (AS). Here three AS are there 1, 71 and 79 respectively. Configure accordingly using three routers.

Step 1: Draw BGP Topology Diagram.



Step 2: Assign ip address on each device as mentioned in Diagram.

**Router0**

```
Router(config)#interface gigabitethernet0/0
Router(config-if)#ip address 192.168.12.1 255.255.255.0
Router(config-if)# no shutdown

Router(config-if)#exit
Router(config)#
Router(config)#interface loopback0

Router(config-if)# ip address 1.1.1.1 255.255.255.255
Router(config-if)#exit
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up
%LINK-5-CHANGED: Interface Loopback0, changed state to up
```

**Router1**

```
Router(config)#interface gigabitethernet0/0
Router(config-if)# ip address 192.168.12.2 255.255.255.0
Router(config-if)# no shutdown

Router(config-if)#exit
Router(config)#
Router(config)#interface gigabitethernet0/1
Router(config-if)# ip address 192.168.23.2 255.255.255.0
Router(config-if)# no shutdown

Router(config-if)#exit
Router(config)#
Router(config)#interface loopback0

Router(config-if)# ip address 2.2.2.2 255.255.255.255
Router(config-if)#exit
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up
```

**Router2**



```

Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface gigabitethernet0/0
Router(config-if)# ip address 192.168.23.3 255.255.255.0
Router(config-if)# no shutdown

Router(config-if)#exit
Router(config)#
Router(config)#interface loopback0

Router(config-if)# ip address 3.3.3.3 255.255.255.255
Router(config-if)#exit
Router(config)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0, changed state
to up

%LINK-5-CHANGED: Interface Loopback0, changed state to up

```

### Step 3: BGP configuration on Router R0:

```

Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router bgp 100
Router(config-router)#network 1.1.1.1 mask 255.255.255.255
Router(config-router)#neighbor 192.168.12.2 remote-as 200
Router(config-router)#exit

```

### Step 4: bgp configuration on Router R1:

```

Router(config)#router bgp 200
Router(config-router)#network 2.2.2.2 mask 255.255.255.255
Router(config-router)#neighbor 192.168.12.1 remote-as 100
Router(config-router)#neighbor 192.168.23.3 remote-as 300
Router(config-router)#exit

```

### Step 5: bgp configuration on Router R2:

```

Router(config)#router bgp 300
Router(config-router)#network 3.3.3.3 mask 255.255.255.255
Router(config-router)#neighbor 192.168.23.2 remote-as 200
Router(config-router)#exit
Router(config)#%BGP-5-ADJCHANGE: neighbor 192.168.23.2 Up

```

### Step 6: bgp configuration Testing and troubleshooting.

```

Pinging 192.168.10.10 with 32 bytes of data:

Reply from 192.168.10.10: bytes=32 time<1ms TTL=128
Reply from 192.168.10.10: bytes=32 time<1ms TTL=128
Reply from 192.168.10.10: bytes=32 time<1ms TTL=128
Reply from 192.168.10.10: bytes=32 time=2ms TTL=128

Ping statistics for 192.168.10.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 0ms

```

### Step 7: check bgp route on router R1:

```
Router>show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route
```

Gateway of last resort is not set

```
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.1.0/24 is directly connected, GigabitEthernet0/0
L       192.168.1.1/32 is directly connected, GigabitEthernet0/0
```

Step 8: Check whether bgp protocols configure or not on Route R1:

```
Router>show ip protocols
Routing Protocol is "bgp 65001"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  IGP synchronization is disabled
  Automatic route summarization is disabled
  Neighbor(s):
    Address          FiltIn FiltOut DistIn DistOut Weight RouteMap
    10.0.0.2
    10.0.0.1
  Maximum path: 1
  Routing Information Sources:
    Gateway          Distance      Last Update
  Distance: external 20 internal 200 local 200
```

Step 9: Show BGP Status

```
Router#show ip bgp summary
BGP router identifier 0.0.0.0, local AS number 65001
BGP table version is 1, main routing table version 6
0 network entries using 0 bytes of memory
0 path entries using 0 bytes of memory
0/0 BGP path/bestpath attribute entries using 0 bytes of memory
0 BGP AS-PATH entries using 0 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 32 total bytes of memory
BGP activity 0/0 prefixes, 0/0 paths, scan interval 60 secs
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
10.0.0.2	4	6507	0	0	1	0	0	00:13:02	4
10.0.0.1	4	65071	0	0	1	0	0	00:13:02	4

Show bgp neighbors status:

```
Router>show ip bgp neighbors
BGP neighbor is 10.0.0.2, remote AS 6507, external link
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Active, up for 00:01:38
  Last read 00:01:38, last write 00:01:38, hold time is 180, keepalive interval is
60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0

      Sent      Rcvd
  Opens:         0         0
  Notifications: 0         0
  Updates:        0         0
  Keepalives:     0         0
  Route Refresh:  0         0
  Total:          0         0
Default minimum time between advertisements runs is 30 seconds
```

Similarly we check bgp route on Router R2:



```
Router>show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route
```

Gateway of last resort is not set

```
      10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.0.0.0/8 is directly connected, GigabitEthernet0/0
L       10.0.0.1/32 is directly connected, GigabitEthernet0/0
```

```
Router>show ip protocols
```

Routing Protocol is "bgp 65071"

Outgoing update filter list for all interfaces is not set

Incoming update filter list for all interfaces is not set

IGP synchronization is disabled

Automatic route summarization is disabled

Neighbor(s):

Address	FiltIn	FiltOut	DistIn	DistOut	Weight	RouteMap
172.16.0.2						
10.0.0.2						

Maximum path: 1

Routing Information Sources:

Gateway	Distance	Last Update
Distance: external 20 internal 200 local 200		

Similarly check bgp route on Router R3:

```
Router>show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route
```

Gateway of last resort is not set

```
      172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       172.16.0.0/16 is directly connected, GigabitEthernet0/0
L       172.16.0.1/32 is directly connected, GigabitEthernet0/0
```

```
Router>show ip protocols
```

Routing Protocol is "bgp 65079"

Outgoing update filter list for all interfaces is not set

Incoming update filter list for all interfaces is not set

IGP synchronization is disabled

Automatic route summarization is disabled

Neighbor(s):

Address	FiltIn	FiltOut	DistIn	DistOut	Weight	RouteMap
10.0.0.2						

Maximum path: 1

Routing Information Sources:

Gateway	Distance	Last Update
Distance: external 20 internal 200 local 200		

## Experiment No : 11

# Title: Domain Name System (DNS) Configuration and Analysis

### Objective:

- Understand how DNS works and its role in network communication.
- Set up a local DNS server in a simulated environment.
- Configure and test DNS query resolution.
- Use Wireshark to capture and analyze DNS traffic.

### Lab Requirements:

- A computer with Linux (Ubuntu/CentOS) or Windows (WSL or Virtual Machine recommended).
- BIND9 (for Linux) or Simple DNS Plus (for Windows) installed.
- Wireshark for packet capture and analysis.
- Internet access (optional, if testing external domains).

## Part 1: Setting Up a Local DNS Server

### Step 1: Install a DNS Server

#### For Linux (Ubuntu/Debian):

1. Open the terminal and install **BIND9**:

```
sudo apt update  
sudo apt install bind9 -y
```

2. Start and enable the DNS service:

```
sudo systemctl start bind9  
sudo systemctl enable bind9
```

#### For Windows (Using Simple DNS Plus or Windows DNS Server):

1. Download and install **Simple DNS Plus** or enable **Windows DNS Server** via Server Manager.
2. Configure the primary DNS zone in the **DNS Manager**.

### Step 2: Configure a Local DNS Zone (For Internal Domains)

1. Open the **BIND configuration file**:

```
sudo nano /etc/bind/named.conf.local
```

2. Add the following zone configuration for a **local domain (example.local)**:

```
zone "example.local" {  
    type master;  
    file "/etc/bind/db.example.local";  
};
```

3. Create a **zone file** for the domain:

```
sudo nano /etc/bind/db.example.local
```

4. Add the following **DNS records**:

```
$TTL 604800
@   IN  SOA  ns1.example.local. admin.example.local. (
        2      ; Serial
        604800 ; Refresh
        86400  ; Retry
        2419200 ; Expire
        604800 )      ; Negative Cache TTL

; Name Servers
@   IN  NS   ns1.example.local.

; A Records
ns1   IN  A   192.168.1.1 www      IN  A
192.168.1.2
```

5. Restart the DNS service:

```
sudo systemctl restart bind9
```

6. Verify the DNS server status:

```
sudo systemctl status bind9
```

## Part 2: DNS Query Resolution Testing

### Step 1: Configure a Client to Use the Local DNS Server

1. Edit the **resolv.conf** file on the client system:

```
sudo nano /etc/resolv.conf
```

2. Add the following:

```
nameserver 192.168.1.1 domain
example.local
```

### Step 2: Test DNS Resolution Using nslookup and dig

#### Using nslookup:

```
nslookup www.example.local
```

Expected Output:

```
Server: 192.168.1.1
Address: 192.168.1.1#53
Name: www.example.local
Address: 192.168.1.2
```

#### Using dig:

```
dig www.example.local Expected
```

Output:

```
www.example.local. 604800 IN A 192.168.1.2
```

## Part 3: DNS Packet Analysis with Wireshark

### Step 1: Capture DNS Traffic

1. Open **Wireshark** and select the **network interface** connected to the DNS server.

2. Start **packet capture** and apply the filter:

```
dns
```

3. Open a **new terminal** and run:

```
nslookup www.example.local
```

4. Observe the **DNS query and response** packets in Wireshark.

## Step 2: Analyze DNS Query Resolution

Look for:

1. **DNS Request (Standard Query A)** – Sent by the client to the DNS server.
2. **DNS Response (Standard Query Response)** – Reply from the DNS server with the IP address.
1. **Query Type (A, AAAA, MX, NS, etc.)** – Identifies the requested record type.
2. **Query Time** – Measures DNS resolution time.

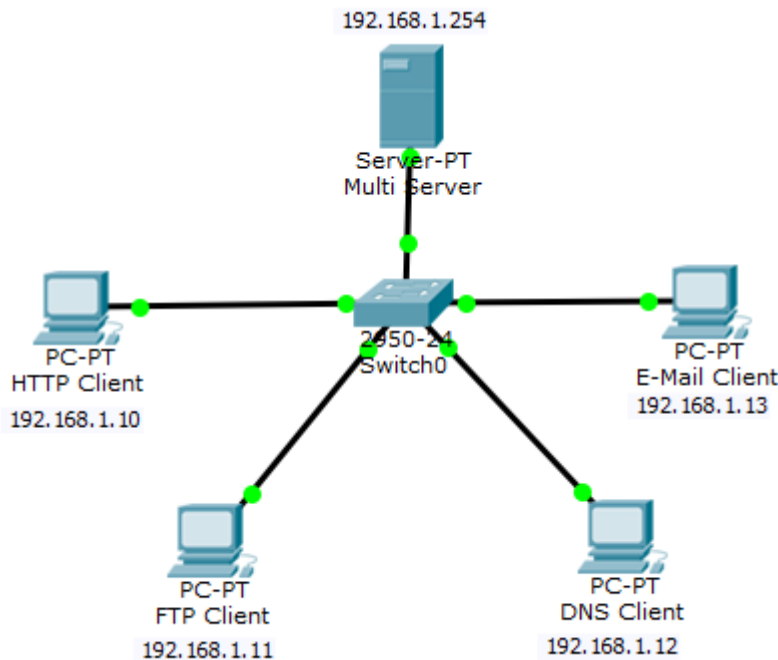
## Lab Report Submission Requirements:

- **Screenshots of:**
  - Local DNS configuration (`db.example.local` file).
  - **DNS query results using `nslookup` and `dig`.**
  - **Wireshark capture showing DNS request and response.**

## Experiment No : 12

### Title: Packet Tracer Simulation - TCP and UDP Communications

#### Topology



#### Objectives

**Part 1: Generate Network Traffic in Simulation Mode**

**Part 2: Examine the Functionality of the TCP and UDP Protocols**

#### Background

This simulation activity is intended to provide a foundation for understanding the TCP and UDP in detail. Simulation mode provides the ability to view the functionality of the different protocols.

As data moves through the network, it is broken down into smaller pieces and identified in some fashion so that the pieces can be put back together. Each of these pieces is assigned a specific name (protocol data unit [PDU]) and associated with a specific layer. Packet Tracer Simulation mode enables the user to view each of the protocols and the associated PDU. The steps outlined below lead the user through the process of requesting services using various applications available on a client PC.

This activity provides an opportunity to explore the functionality of the TCP and UDP protocols, multiplexing and the function of port numbers in determining which local application requested the data or is sending the data.

## Part 1: Generate Network Traffic in Simulation Mode

### Step 1: Generate traffic to populate Address Resolution Protocol (ARP) tables.

Perform the following tasks task to reduce the amount of network traffic viewed in the simulation.

- Click **MultiServer** and click the **Desktop** tab > **Command Prompt**.
- Enter the ping **192.168.1.255** command. This will take a few seconds as every device on the network responds to **MultiServer**.
- Close the **MultiServer** window.

```
SERVER>ping 192.168.1.255

Pinging 192.168.1.255 with 32 bytes of data:

Reply from 192.168.1.11: bytes=32 time=0ms TTL=128
Reply from 192.168.1.13: bytes=32 time=0ms TTL=128
Reply from 192.168.1.10: bytes=32 time=0ms TTL=128
Reply from 192.168.1.12: bytes=32 time=0ms TTL=128
Reply from 192.168.1.13: bytes=32 time=0ms TTL=128
Reply from 192.168.1.10: bytes=32 time=0ms TTL=128
Reply from 192.168.1.11: bytes=32 time=1ms TTL=128
Reply from 192.168.1.12: bytes=32 time=0ms TTL=128
Reply from 192.168.1.10: bytes=32 time=0ms TTL=128
Reply from 192.168.1.12: bytes=32 time=0ms TTL=128
Reply from 192.168.1.13: bytes=32 time=1ms TTL=128
Reply from 192.168.1.11: bytes=32 time=1ms TTL=128
Reply from 192.168.1.13: bytes=32 time=0ms TTL=128
Reply from 192.168.1.10: bytes=32 time=0ms TTL=128
Reply from 192.168.1.11: bytes=32 time=0ms TTL=128
Reply from 192.168.1.12: bytes=32 time=0ms TTL=128

Ping statistics for 192.168.1.255:
    Packets: Sent = 4, Received = 16, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

### Step 2: Generate web (HTTP) traffic.

- Switch to Simulation mode.
- Click **HTTP Client** and click the **Desktop** tab > **Web Browser**.
- In the URL field, enter **192.168.1.254** and click **Go**. Envelopes (PDUs) will appear in the simulation window.
- Minimize, but do not close, the **HTTP Client** configuration window.

### Step 3: Generate FTP traffic.

- Click **FTP Client** and click the **Desktop** tab > **Command Prompt**.
- Enter the **ftp 192.168.1.254** command. PDUs will appear in the simulation window.
- Minimize, but do not close, the **FTP Client** configuration window.

### Step 4: Generate DNS traffic.

- Click **DNS Client** and click the **Desktop** tab > **Command Prompt**.

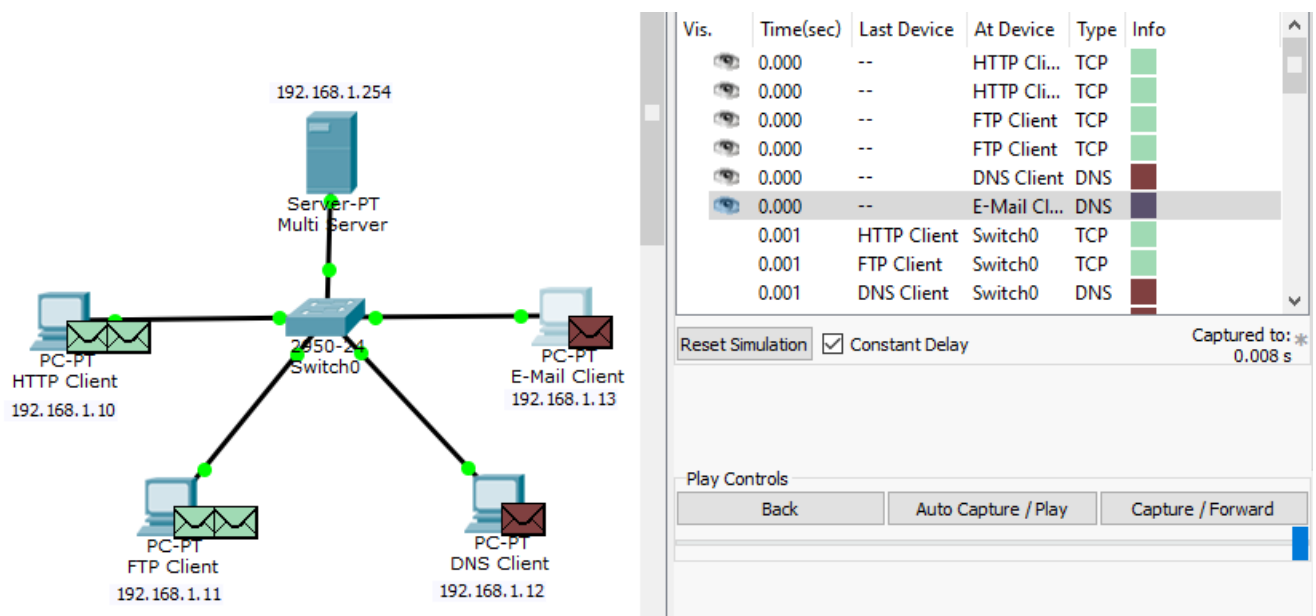
- b. Enter the **nslookup multiserver.pt.ptu** command. A PDU will appear in the simulation window.
- c. Minimize, but do not close, the **DNS Client** configuration window.

### Step 5: Generate Email traffic.

- a. Click **E-Mail Client** and click the **Desktop** tab > **E Mail** tool.
- b. Click Compose and enter the following information:
  1. **To:** user@multiserver.pt.ptu
  2. **Subject:** Personalize the subject line
  3. **E-Mail Body:** Personalize the Email
- c. Click **Send**.
- d. Minimize, but do not close, the **E-Mail Client** configuration window.

### Step 6: Verify that the traffic is generated and ready for simulation.

Every client computer should have PDUs listed in the Simulation Panel.



## Part 2: Examine Functionality of the TCP and UDP Protocols

### Step 1: Examine multiplexing as all of the traffic crosses the network.

You will now use the **Capture/Forward** button and the **Back** button in the Simulation Panel.

- a. Click **Capture/Forward** once. All of the PDUs are transferred to the switch.
- b. Click **Capture/Forward** again. Some of the PDUs disappear.

What do you think happened to them?

- They have reached their destination

- c. Click **Capture/Forward** six times. All clients should have received a reply. Note that only one PDU can cross a wire in each direction at any given time. What is this called?
- d. A variety of PDUs appears in the event list in the upper right pane of the simulation window.

Why are they so many different colors?

- Different Protocols (TCP, UDP)

- e. Click **Back** eight times. This should reset the simulation.

**Note:** Do not click **Reset Simulation** any time during this activity; if you do, you will need to repeat the steps in Part 1.

### Step 2: Examine HTTP traffic as the clients communicate with the server.

- a. Filter the traffic that is currently displayed to display only **HTTP** and **TCP** PDUs filter the traffic that is currently displayed:
  - 1. Click **Edit Filters** and toggle the **Show All/None** check box.
  - 2. Select **HTTP** and **TCP**. Click anywhere outside of the Edit Filters box to hide it. The Visible Events should now display only **HTTP** and **TCP** PDUs.
- b. Click **Capture/Forward**. Hold your mouse above each PDU until you find one that originates from **HTTP Client**. Click the PDU envelope to open it.
- c. Click the **Inbound PDU Details** tab and scroll down to the last section. What is the section labeled?

Are these communications considered to be reliable?

- Yes as it uses TCP

- d. Record the **SRC PORT**, **DEST PORT**, **SEQUENCE NUM**, and **ACK NUM** values. What is written in the field to the left of the **WINDOW** field?
  - OFF., RES., SYN
- e. Close the PDU and click **Capture/Forward** until a PDU returns to the **HTTP Client** with a checkmark.
- f. Click the PDU envelope and select **Inbound PDU Details**. How are the port and sequence numbers different than before?
- g. There is a second **PDU** of a different color, which **HTTP Client** has prepared to send to **MultiServer**. This is the beginning of the HTTP communication. Click this second PDU envelope and select **Outbound PDU Details**.
- h. What information is now listed in the TCP section? How are the port and sequence numbers different from the previous two PDUs?
  - Sequence Number is 1 and Ack Number is 1.
- i. Click **Back** until the simulation is reset.



**Step 3: Examine FTP traffic as the clients communicate with the server.**

- In the Simulation Panel, change Edit Filters to display only **FTP** and **TCP**.
- Click **Capture/Forward**. Hold your cursor above each PDU until you find one that originates from **FTP Client**. Click that PDU envelope to open it.
- Click the **Inbound PDU Details** tab and scroll down to the last section.

What is the section labeled?

Are these communications considered to be reliable?

- *Yes, it uses TCP*

- Record the **SRC PORT**, **DEST PORT**, **SEQUENCE NUM**, and **ACK NUM** values.

TCP

0		16		31		Bits
SRC PORT: 1025			DEST PORT: 80			
SEQUENCE NUM: 0						
ACK NUM: 0						
OFF.	RES.	SYN	WINDOW			
CHECKSUM: 0x0			URGENT POINTER			
OPTION				PADDING		
DATA (VARIABLE)						

What is written in the field to the left of the **WINDOW** field?

- *OFF., RES., SYN*

- Close the PDU and click **Capture/Forward** until a PDU returns to the **FTP Client** with a checkmark.
- Click the PDU envelope and select **Inbound PDU Details**. How are the port and sequence numbers different than before?
  - *Previously the src port was 1025 which is now 21 and dest port was 80 which is now 1025*
- Click the **Outbound PDU Details** tab. How are the port and sequence numbers different from the previous two results?
  - *Now the src port is 1025 and dest port is 21.*
- Close the PDU and click **Capture/Forward** until a second PDU returns to the **FTP Client**. The PDU is a different color.
- Open the PDU and select **Inbound PDU Details**. Scroll down past the TCP section. What is the message from the server?
  - *Welcome to PT Ftp server*
- Click **Back** until the simulation is reset.

**Step 4: Examine DNS traffic as the clients communicate with the server.**

- In the Simulation Panel, change **Edit Filters** to display only **DNS** and **UDP**.
- Click the PDU envelope to open it.
- Click the **Inbound PDU Details** tab and scroll down to the last section. What is the section labeled?

Are these communications considered to be reliable?

- *No, it uses UDP*

- Record the **SRC PORT** and **DEST PORT** values.

UDP

0	16	31	Bits
SRC PORT: 1025		DEST PORT: 53	
LENGTH: 0x1d		CHECKSUM: 0x0	
DATA (VARIABLE)			

Why is there no sequence and acknowledgement number?

- *UDP don't have seq and ack number.*

- Close the **PDU** and click **Capture/Forward** until a PDU returns to the **DNS Client** with a checkmark.
  - Click the PDU envelope and select **Inbound PDU Details**. How are the port and sequence numbers different than before?
  - What is the last section of the **PDU** called?
- *DATA*
- Click **Back** until the simulation is reset.

**Step 5: Examine email traffic as the clients communicate with the server.**

- In the Simulation Panel, change **Edit Filters** to display only **POP3, SMTP and TCP**.
- Click **Capture/Forward**. Hold your cursor above each PDU until you find one that originates from **E-mail Client**. Click that PDU envelope to open it.
- Click the **Inbound PDU Details** tab and scroll down to the last section. What transport layer protocol does email traffic use?

Are these communications considered to be reliable?

- *Yes, it uses TCP*

- Record the **SRC PORT**, **DEST PORT**, **SEQUENCE NUM**, and **ACK NUM** values. What is written in the field to the left of the WINDOW field?
- Close the **PDU** and click **Capture/Forward** until a PDU returns to the **E-Mail Client** with a checkmark.

- f. Click the PDU envelope and select **Inbound PDU Details**. How are the port and sequence numbers different than before?
- g. Click the **Outbound PDU Details** tab. How are the port and sequence numbers different from the previous two results?
- h. There is a second **PDU** of a different color that **HTTP Client** has prepared to send to **MultiServer**. This is the beginning of the email communication. Click this second PDU envelope and select **Outbound PDU Details**.
- i. How are the port and sequence numbers different from the previous two **PDUs**?
- j. What email protocol is associated with TCP port 25? What protocol is associated with TCP port 110?
  - *SMTP – Port 25*
  - *POP3 – Port 110*
- k. Click **Back** until the simulation is reset.

### Step 6: Examine the use of port numbers from the server.

- a. To see TCP active sessions, perform the following steps in quick succession:
  - 1. Switch back to Realtime mode.
  - 2. Click MultiServer and click the Desktop tab > Command Prompt.
- b. Enter the netstat command.  
What protocols are listed in the left column?
  - *TCP*What port numbers are being used by the server?
  - *Port 21*
- c. What states are the sessions in?
  - *ESTABLISHED*
- d. Repeat the netstat command several times until you see only one session still ESTABLISHED.  
For which service is this connection still open?
  - *TCP*Why doesn't this session close like the other three? (Hint: Check the minimized clients)
  - *TCP is active and Persistent connection*

## Experiment No : 13

### Title: Subnetting and VLAN Implementation in Cisco Packet Tracer

#### Objective

- Understand the concepts of IP subnetting and subnet mask calculation.
- Perform subnet calculations for a given network scenario.
- Implement VLANs and Subnetting in Cisco Packet Tracer.
- Improve network efficiency through proper subnet division.

#### Part 1: Subnet Calculation and IP Addressing

##### Task 1: Given Network Scenario

- Network Address: 192.168.1.0/24
- Requirement: Divide the network into 4 subnets for different departments.

##### Task 2: Subnet Calculation

1. **Determine Subnet Mask:**
  - /24 means default subnet mask: 255.255.255.0
  - 4 subnets → Need **2 extra bits** (/26)
  - New subnet mask: 255.255.255.192
  - **Each subnet has 62 usable IPs** ( $2^6 - 2 = 62$ ).

2. **Assign Subnets and Host Ranges:**

Subnet	Network Address	First Usable IP	Last Usable	IP Broadcast Address
<b>Subnet 1</b>	192.168.1.0/26	192.168.1.1	192.168.1.62	192.168.1.63
<b>Subnet 2</b>	192.168.1.64/26	192.168.1.65	192.168.1.126	192.168.1.127
<b>Subnet 3</b>	192.168.1.128/26	192.168.1.129	192.168.1.190	192.168.1.191
<b>Subnet 4</b>	192.168.1.192/26	192.168.1.193	192.168.1.254	192.168.1.255

#### Part 2: Implementing VLANs and Subnetting in Cisco Packet Tracer

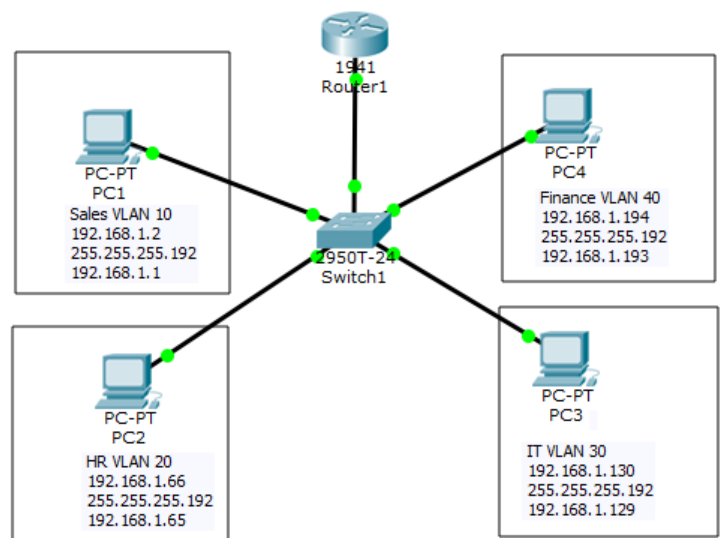
##### Task 3: Create the Network in Cisco Packet Tracer

1. **Devices Required:**
  - **1 Router** (For Inter-VLAN Routing)
  - **1 Layer 3 Switch**
  - **4 PCs** (Each in a separate VLAN)
  - **4 VLANs** (e.g., VLAN 10, 20, 30, 40)
  - **Connections:** Use Ethernet cables.

2. **VLAN Configuration on Switch:**

```
Switch(config)# vlan 10
Switch(config-vlan)# name Sales
Switch(config-vlan)# exit
```

```
Switch(config)# vlan 20
Switch(config-vlan)# name HR
Switch(config-vlan)# exit
```



```
Switch(config)# vlan 30
Switch(config-vlan)# name IT
Switch(config-vlan)# exit
```

```
Switch(config)# vlan 40
Switch(config-vlan)# name Finance
Switch(config-vlan)# exit
```

### 3. Assign Ports to VLANs:

```
Switch(config)# interface fa0/1
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# exit
```

```
Switch(config)# interface fa0/2
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 20
Switch(config-if)# exit
```

### 4. Enable Inter-VLAN Routing on Router:

```
Router(config)# interface g0/0.10
Router(config-subif)# encapsulation dot1Q 10
Router(config-subif)# ip address 192.168.1.1 255.255.255.192
Router(config-subif)# exit
```

## Outcome

- ✓ Learn **IP subnetting** concepts and perform **subnet calculations**.
- ✓ Successfully implement **VLANs and Subnets** in **Cisco Packet Tracer**.

```
Switch#show vlan brief
```

VLAN Name	Status	Ports
1 default	active	Fa0/5, Fa0/6, Fa0/7, Fa0/8 Fa0/9, Fa0/10, Fa0/11, Fa0/12 Fa0/13, Fa0/14, Fa0/15, Fa0/16 Fa0/17, Fa0/18, Fa0/19, Fa0/20 Fa0/21, Fa0/22, Fa0/23, Fa0/24 Gig0/1, Gig0/2
10 Sales	active	Fa0/1
20 HR	active	Fa0/2
30 IT	active	Fa0/3
40 Finance	active	Fa0/4
1002 fddi-default	active	
1003 token-ring-default	active	
1004 fddinet-default	active	
1005 trnet-default	active	

- ✓ Improve **network efficiency** through **structured subnet division**.