# RAJALAKSHMI ENGINEERING COLLEGE

**An Autonomous Institution**
**Affiliated to Anna University, Chennai,**
**Rajalakshmi Nagar, Thandalam – 602 105**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

| |
|---|
| **CS19541 - COMPUTER NETWORKS** |
| **Laboratory Record Note Book** |

**Name :** TAMILARASAN D

**Register No. :** 2116221501157

**Year / Branch / Section :** $3^{rd}$ Year / AIML / C

**Semester :** V

**AcademicYear:** 2024-2025

# RAJALAKSHMI ENGINEERING COLLEGE

**An Autonomous Institution**
**Affiliated to Anna University, Chennai,**
**Rajalakshmi Nagar, Thandalam – 602 105**

## BONAFIDE CERTIFICATE

**Name:** TAMILARASAN D

**Academic Year:** 2024-25 **Semester:** V **Branch:** AIML - C

**Register No.** 2116221501157

***Certified that this is the bonafide record of work done by the above student in the Computer Networks (CS19341) Laboratory during the academic year 2024-2025***

**Signature of Faculty in-charge**

**Submitted for the Practical Examination held on 20.11.2024**

**Internal Examiner** **External Examiner**

| | |
|---|---|
| **EX NO : 1** | **STUDY OF VARIOUS NETWORKING COMMANDS** |
| **DATE:** | |

**AIM: -**

Study of various Network commands used in Linux and Windows:

**BASIC NETWORKING COMMANDS:**

**Arp -a :**

ARP is short form of address resolution protocol, It will show the IP address of your computer along with the IP address and MAC address of your router.

```
PS C:\Users\Lenovo> arp -a

Interface: 172.16.10.187 --- 0x8
  Internet Address      Physical Address      Type
  172.16.8.1            7c-5a-1c-cf-be-45     dynamic
  172.16.11.255         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
```

**Hostname :**

This is the simplest of all TCP/IP commands. It simply displays the name of your computer.

```
PS C:\Users\Lenovo> hostname
HDC0422264
```

**Ifconfig /all**

This command displays detailed configuration information about your TCP/IP connection including Router, Gateway, DNS, DHCP, and type of Ethernet adapter in your system.

```
PS C:\Users\Lenovo> ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : HDC0422264
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Mixed
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Realtek PCIe GbE Family Controller
   Physical Address. . . . . . . . . : 88-AE-DD-15-EE-C4
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::6285:9236:849a:e25a%8(Preferred)
   IPv4 Address. . . . . . . . . . . : 172.16.10.187(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.252.0
   Default Gateway . . . . . . . . . : 172.16.8.1
   DHCPv6 IAID . . . . . . . . . . . : 143175389
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2A-27-EC-94-88-AE-DD-15-EE-C4
   DNS Servers . . . . . . . . . . . : 172.16.8.1
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

**Netstat**

(network statistics) netstat displays a variety of statistics about a computers active TCP/IP connections. It is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc.

e.g.:- netstat -r

```
[exam@fedora ~]$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 fedora:57688            201.181.244.35.bc:https ESTABLISHED
tcp        0      0 fedora:39528            93.243.107.34.bc.:https ESTABLISHED
tcp        0      0 fedora:60110            152.195.38.76:http      ESTABLISHED
tcp        0      0 fedora:50882            76.237.120.34.bc.:https ESTABLISHED
tcp        0      0 fedora:43484            server-52-222-149:https ESTABLISHED
tcp        0      0 fedora:37128            123.208.120.34.bc:https ESTABLISHED
tcp        0      0 fedora:41700            201.181.244.35.bc:https TIME_WAIT
tcp        0      0 fedora:59508            233.90.160.34.bc.:https ESTABLISHED
tcp        0      0 fedora:49664            191.144.160.34.bc:https TIME_WAIT
tcp        0      0 fedora:39662            maa05s28-in-f5.1e:https ESTABLISHED
udp        0      0 fedora:bootpc           _gateway:bootps         ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  3      [ ]         STREAM     CONNECTED     26847
unix  3      [ ]         SEQPACKET  CONNECTED     26845
unix  3      [ ]         STREAM     CONNECTED     28033
unix  3      [ ]         STREAM     CONNECTED     26741
unix  3      [ ]         STREAM     CONNECTED     24080    /run/user/1000/bus
```

**Ping**

(Packet INternet Groper) command is the best way to test connectivity between two nodes. Ping uses ICMP (Internet Control Message Protocol) to communicate to other devices.

```
[exam@fedora ~]$ ping www.google.com
PING www.google.com (142.250.183.228) 56(84) bytes of data.
64 bytes from maa05s23-in-f4.1e100.net (142.250.183.228): icmp_seq=1 ttl=120 time=2.98 ms
64 bytes from maa05s23-in-f4.1e100.net (142.250.183.228): icmp_seq=2 ttl=120 time=3.74 ms
64 bytes from maa05s23-in-f4.1e100.net (142.250.183.228): icmp_seq=3 ttl=120 time=4.17 ms
64 bytes from maa05s23-in-f4.1e100.net (142.250.183.228): icmp_seq=4 ttl=120 time=2.95 ms
64 bytes from maa05s23-in-f4.1e100.net (142.250.183.228): icmp_seq=5 ttl=120 time=3.60 ms
^C
--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 2.947/3.488/4.170/0.467 ms
```

**Nslookup**

(name server lookup) is a tool used to perform DNS lookups in Linux. It is used to display DNS details, such as the IP address of a particular computer, the MX records for a domain or the NS servers of a domain. nslookup can operate in two modes: interactive and Non-interactive.

```
[exam@fedora ~]$ nslookup www.google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.250.183.228
Name:   www.google.com
Address: 2404:6800:4007:81e::2004
```

**Ping**

Ping is a tool that verifies IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP) Echo Request messages. The receipt of corresponding Echo Reply messages are displayed, along with round-trip times. Ping is the primary TCP/IP command used to troubleshoot connectivity, reachability, and name resolution.

```
➜  ~ ping www.instagram.com
PING z-p42-instagram.c10r.instagram.com (163.70.138.174) 56(84) bytes of data.
64 bytes from instagram-p42-shv-01-tir3.fbcdn.net (163.70.138.174): icmp_seq=5 ttl=45 time=64.9 ms
64 bytes from instagram-p42-shv-01-tir3.fbcdn.net (163.70.138.174): icmp_seq=7 ttl=45 time=65.0 ms
64 bytes from instagram-p42-shv-01-tir3.fbcdn.net (163.70.138.174): icmp_seq=8 ttl=45 time=65.0 ms
64 bytes from instagram-p42-shv-01-tir3.fbcdn.net (163.70.138.174): icmp_seq=9 ttl=45 time=65.1 ms
64 bytes from instagram-p42-shv-01-tir3.fbcdn.net (163.70.138.174): icmp_seq=10 ttl=45 time=65.3 ms
^C
--- z-p42-instagram.c10r.instagram.com ping statistics ---
10 packets transmitted, 5 received, 50% packet loss, time 9150ms
rtt min/avg/max/mdev = 64.939/65.051/65.304/0.132 ms
```

Configuring an Ethernet connection by using nmcli :

If you connect a host to the network over Ethernet, you can manage the connection’s settings on the command line by using the nmcli utility.

**Mtr**

MTR (Matt's traceroute) is a program with a command-line interface that serves as a network diagnostic and troubleshooting tool. This command combines the functionality of the ping and traceroute commands.

```
mtr google.com

                                 My traceroute  [v0.95]
fedora (172.16.11.197) -> google.com (142.250.182.78)         2024-07-23T10:42:13+0530
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                             Packets               Pings
 Host                                       Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. _gateway                                28.6%    14    0.2   0.3   0.2   0.5   0.1
 2. static-41.229.249.49-tataidc.co.in      21.4%    14    2.9   3.5   2.8   5.7   0.8
 3. 142.250.171.162                         21.4%    14    2.6   2.9   2.5   4.4   0.6
 4. 142.251.227.215                         21.4%    14    3.2   3.4   3.1   3.7   0.2
 5. 142.251.55.245                          21.4%    14    3.3   3.3   3.1   3.7   0.2
 6. maa05s20-in-f14.1e100.net               21.4%    14    3.7   3.8   3.6   4.3   0.2
```

```
mtr -b google.com

                                 My traceroute  [v0.95]
fedora (172.16.11.197) -> google.com (142.250.182.78)         2024-07-23T10:46:25+0530
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                             Packets               Pings
 Host                                       Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. _gateway (172.16.8.1)                   81.8%    12    0.4   0.5   0.4   0.6   0.2
 2. static-41.229.249.49-tataidc.co.in (49.249. 63.6%    12   41.9  12.9   2.9  41.9  19.3
 3. 142.250.171.162 (142.250.171.162)       63.6%    12    2.9   2.9   2.6   3.2   0.2
 4. 142.251.227.215 (142.251.227.215)       81.8%    12    3.7   3.5   3.3   3.7   0.3
 5. 142.251.55.245 (142.251.55.245)         80.0%    11    3.3   3.5   3.3   3.7   0.3
 6. maa05s20-in-f14.1e100.net (142.250.182.78)  90.0%    11    4.2   4.2   4.2   4.2   0.0
```

**Pathping**

```
PS C:\Users\Lenovo> pathping

Usage: pathping [-g host-list] [-h maximum_hops] [-i address] [-n]
                [-p period] [-q num_queries] [-w timeout]
                [-4] [-6] target_name

Options:
    -g host-list     Loose source route along host-list.
    -h maximum_hops  Maximum number of hops to search for target.
    -i address       Use the specified source address.
    -n               Do not resolve addresses to hostnames.
    -p period        Wait period milliseconds between pings.
    -q num_queries   Number of queries per hop.
    -w timeout       Wait timeout milliseconds for each reply.
    -4               Force using IPv4.
    -6               Force using IPv6.
```

**Route**

route command is used to show/manipulate the IP routing table. It is primarily used to set up static routes to specific host or networks via an interface.

```
IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    331 ::1/128                  On-link
  8    291 fe80::/64                On-link
  8    291 fe80::6285:9236:849a:e25a/128
                                    On-link
  1    331 ff00::/8                 On-link
  8    291 ff00::/8                 On-link
===========================================================================
```

**Ip**

```
[exam@fedora ~]$ ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { address | addrlabel | fou | help | ila | ioam | l2tp | link |
                   macsec | maddress | monitor | mptcp | mroute | mrule |
                   neighbor | neighbour | netconf | netns | nexthop | ntable |
                   ntbl | route | rule | sr | tap | tcpmetrics |
                   token | tunnel | tuntap | vrf | xfrm }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                    -h[uman-readable] | -iec | -j[son] | -p[retty] |
                    -f[amily] { inet | inet6 | mpls | bridge | link } |
                    -4 | -6 | -M | -B | -0 |
                    -l[oops] { maximum-addr-flush-attempts } | -br[ief] |
                    -o[neline] | -t[imestamp] | -ts[hort] | -b[atch] [filename] |
                    -rc[vbuf] [size] | -n[etns] name | -N[umeric] | -a[ll] |
                    -c[olor]}
```

**ip address show**

```
[exam@fedora ~]$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 88:ae:dd:15:ee:c4 brd ff:ff:ff:ff:ff:ff
    inet 172.16.10.188/23 brd 172.16.11.255 scope global noprefixroute enp2s0
       valid_lft forever preferred_lft forever
    inet6 fe80::68ce:a1c1:255d:34a6/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

**Tcpdump**

The tcpdump command is designed for capturing and displaying packets.

```
→  ~ tcpdump -D
1.enp2s0 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.usbmon2 (Raw USB traffic, bus number 2)
6.usbmon1 (Raw USB traffic, bus number 1)
7.usbmon0 (Raw USB traffic, all USB buses) [none]
8.nflog (Linux netfilter log (NFLOG) interface) [none]
9.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
→  ~ |
```

**Nmcli**

List the NetworkManager connection profiles:

```
→  ~ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  fdc1bf0b-0357-35c4-90c5-829cf413c9c7  ethernet  enp2s0
lo                  5c0b1041-d4e7-44b9-af93-270f1dc86a82  loopback  lo
→  ~ |
```

**Nmcli verification :**

Display the IP settings of the NIC:

```
→  ~ ip address show enp2s0
2: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
 qlen 1000
    link/ether 88:ae:dd:14:72:3e brd ff:ff:ff:ff:ff:ff
    inet 172.16.11.197/22 brd 172.16.11.255 scope global dynamic noprefixroute enp2s0
       valid_lft 3324sec preferred_lft 3324sec
    inet6 fe80::f20b:b3c6:bb33:40ad/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
→  ~ |
```

Display the IPv4 default gateway:

```
→  ~ ip route show default
default via 172.16.8.1 dev enp2s0 proto dhcp src 172.16.11.197 metric 100
→  ~ |
```

Display the DNS settings:

```
# operation for /etc/resolv.conf.

nameserver 127.0.0.53
options edns0 trust-ad
search .
→  ~ |
```

Use the ping utility to verify that this host can send packets to other hosts:

```
→  ~ ping fedora
PING fedora(fedora (fe80::f20b:b3c6:bb33:40ad%enp2s0)) 56 data bytes
64 bytes from fedora (fe80::f20b:b3c6:bb33:40ad%enp2s0): icmp_seq=1 ttl=64 time=0.085 ms
64 bytes from fedora (fe80::f20b:b3c6:bb33:40ad%enp2s0): icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from fedora (fe80::f20b:b3c6:bb33:40ad%enp2s0): icmp_seq=3 ttl=64 time=0.076 ms
64 bytes from fedora (fe80::f20b:b3c6:bb33:40ad%enp2s0): icmp_seq=4 ttl=64 time=0.072 ms
64 bytes from fedora (fe80::f20b:b3c6:bb33:40ad%enp2s0): icmp_seq=5 ttl=64 time=0.102 ms
^C
--- fedora ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4136ms
rtt min/avg/max/mdev = 0.072/0.082/0.102/0.010 ms
→  ~
```

**RESULT :**

Thus the basic networking commands used in Windows and Linux are studied successfully.

# Ex2: Study of Cables used in networking

## 1. 12-Core Fiber Optic Cable:

A 12 core fiber optic cable is a type of optical cable that contains 12 individual strands or cores of optical fibers within a protective outer jacket. Each core is capable of carrying a separate communication signal, allowing for multiple data streams to be transmitted simultaneously.

Fiber optic cables are made up of thin strands of glass or plastic fibers that transmit data using light pulses. These cables offer several advantages over traditional copper cables, including higher bandwidth, faster data transmission speeds, and longer transmission distances.

## Advantages of the 12-Core Fiber Optic Cable:

1. **Balanced Capacity and Cost**
   - Provides a good balance between data transmission capacity and cost efficiency.
2. **High Data Capacity**
   - Supports the transmission of multiple signals simultaneously, making it ideal for high-data-demand applications.
3. **Versatility in Applications**
   - Suitable for various use cases including telecommunications networks, data centers, and internet service providers.
4. **Scalability**
   - Allows for future expansion and upgrades without requiring a complete overhaul of the fiber infrastructure.
5. **Efficient Use of Space**
   - Compact configuration helps in optimizing the use of physical space within conduit and cable trays.
6. **Reduced Latency**
   - Provides faster data transmission rates compared to traditional copper cables, leading to lower latency.
7. **Improved Reliability**
   - Fiber optic cables are less susceptible to electromagnetic interference, resulting in more reliable data transmission.
8. **Long-Distance Transmission**
   - Capable of transmitting data over long distances with minimal signal loss, making it suitable for extensive network coverage.
9. **Future-Proofing**
   - Supports evolving data requirements and technological advancements, ensuring long-term usability and investment protection.
10. **Cost-Effective Upgrades**

    - Easier and more economical to upgrade compared to higher-core configurations, due to its moderate core count.

11. **Enhanced Bandwidth**
    - Offers substantial bandwidth, which is essential for modern high-speed internet and network services.

## 2. 2 core fiber optic cable:

A 2 core fiber optic cable is a type of optical fiber cable that consists of two individual fiber strands enclosed within a protective outer sheath. Each fiber strand, also known as a core, is made of high-quality glass or plastic and is capable of transmitting large amounts of data using light signals. These cables are widely used in telecommunications, data centers, and networking applications to provide high-speed and reliable data transmission.

## Advantages of Using 2 Core Fiber Optic Cable:

**1. High Bandwidth:**
One of the key advantages of using 2 core fiber optic cable is its ability to provide high bandwidth. With the ever-increasing demand for faster data transmission, this cable can support the transmission of large amounts of data at high speeds, making it ideal for applications that require high-speed internet connectivity.

**2. Long Transmission Distances:**
Fiber optic cables have the advantage of being able to transmit data over long distances without any significant loss in signal quality. This is particularly important in scenarios where data needs to be transmitted over long distances, such as in submarine communications or connecting remote locations.

**3. Immunity to Electromagnetic Interference:**
Unlike traditional copper cables, fiber optic cables are immune to electromagnetic interference (EMI). This means that they are not affected by nearby electrical equipment, power lines, or other sources of electromagnetic radiation. This makes fiber optic cables more reliable and less prone to signal degradation.

**4. Security:**
Fiber optic cables provide a high level of security for data transmission. Since they use light signals to transmit data, it is extremely difficult to tap into the cable and intercept the data being transmitted. This makes fiber optic cables highly secure for transmitting sensitive information.

**5. Future-Proof Technology:**
Fiber optic cables are considered to be a future-proof technology. As data demands continue to increase, fiber optic cables have the ability to support higher speeds and greater bandwidths.

This means that investing in 2 core fiber optic cables ensures that your network infrastructure is ready to handle future advancements in technology.

## 2. Limitations of 12-Core Fiber Optic Cable:

**1. Cost:**
Fiber optic cables, including 12-core variants, can be more expensive to install and maintain compared to traditional copper cables. The specialized equipment and expertise required for installation and repair contribute to the higher costs.

**2. Fragility:**
Fiber optic cables are delicate and can be easily damaged if mishandled during installation or maintenance. Extra care is needed to prevent accidental breaks or bends that could disrupt signal transmission.

**3. Limited Availability:**
While fiber optic networks are becoming more widespread, they may not be available in all areas. The availability of 12-core fiber optic cables can be limited to certain regions or specific network providers.

## 3. Twisted Pair Cables:

Twisted-pair are generally made of copper, and a pair of wires are twisted together to decrease interference by adjacent wires. A twisted pair includes two conductors (copper), each with its plastic insulation, twisted together. One of the wires can transfer signals to the receiver, and the difference is used just as a ground reference. The receiver helps the difference between the two.

This means, if two wires are correlated to each other, the noise or crosstalk can affect one wire, and the difference between the two levels would vary. When these wires are twisted, both wires have a similar effect of noise. This way, the receiver receives the correct signal. The number of a twist on the cable defines the quality of signals carried by them.

### Advantages of Twisted Pair Cables :

- It is easiest to install, manpower to repair, and service are readily available.
- In the mobile system, the signal can traverse various kilometres without amplification.

- It can be the least costly for short distances.
- If part of a twisted-pair cable is broken, the whole network is not shut down.
- It is flexible and easy to connect.
- It has a low weight.

## Disadvantages of Twisted Pair Cables:

- It is higher error rates when the line length is more than 100 meters because it is easily affected by more signals.
- It has low bandwidth.
- It only supports a data transfer rate of up to 10 MBPS (Megabytes per second).

## 4. Cross Over Cable:

A straight through cable is a type of twisted pair cable that is used in local area networks to connect a computer to a network hub such as a router. This type of cable is also sometimes called a patch cable and is an alternative to wireless connections where one or more computers access a router through a wireless signal. On a straight through cable, the wired pins match. Straight through cable use one wiring standard: both ends use T568A wiring standard or both ends use T568B wiring standard. The following figure shows a straight through cable of which both ends are wired as the T568B standard.

Use crossover cables for the following cabling:

- Switch to switch
- Switch to hub
- Hub to hub
- Router to router
- Router Ethernet port to PC NIC
- PC to PC

## 5. Straight Through Cable:

A straight through cable is a type of twisted pair cable that is used in local area networks to connect a computer to a network hub such as a router. This type of cable is also sometimes called a patch cable and is an alternative to wireless connections where one or more computers access a router through a wireless signal. On a straight through cable, the wired pins match. Straight through cable use one wiring standard: both ends use T568A wiring standard or both ends use T568B wiring standard. The following figure shows a straight through cable of which both ends are wired as the T568B standard.

Use straight through Ethernet cable for the following cabling:

- Switch to router
- Switch to PC or server
- Hub to PC or server

**AIM: Write a program to implement error detection and correction using HAMMING code concept. Make a test run to input data stream and verify error correction feature.**

**Error Correction at Data Link Layer:**
Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is transmitted from the sender to the receiver. It is a technique developed by R.W. Hamming for error correction.

**Create sender program with below features.**

1. Input to sender file should be a text of any length. Program should convert the text to binary.
2. Apply hamming code concept on the binary data and add redundant bits to it.
3. Save this output in a file called channel.

**Create a receiver program with below features**

1. Receiver program should read the input from Channel file.
2. Apply hamming code on the binary data to check for errors.
3. If there is an error, display the position of the error.
4. Else remove the redundant bits and convert the binary data to ascii and display the output.

## Student observation:-

Write the code here:

```
import numpy as np

# Function to convert text to binary
def text_to_binary(text):
return ''.join(format(ord(char), '08b') for char in text)

# Function to convert binary to text
def binary_to_text(binary_data):
chars = [chr(int(binary_data[i:i+8], 2)) for i in range(0, len(binary_data), 8)]
return ''.join(chars)

# Function to calculate redundant bits needed for Hamming Code
def calculate_redundant_bits(m):
r = 0
while (2 ** r) < (m + r + 1):
r += 1
return r

# Function to apply Hamming Code to add redundant bits to binary data
def apply_hamming_code(data):
m = len(data)
r = calculate_redundant_bits(m)
```

```
hamming_code = ['0'] * (m + r)
j = 0
k = 1
for i in range(1, m + r + 1):
if i == 2 ** j:
j += 1
else:
hamming_code[i - 1] = data[-k]
k += 1

for i in range(r):
x = 2 ** i
one_count = 0
for j in range(1, m + r + 1):
if j & x and hamming_code[j - 1] == '1':
one_count += 1
hamming_code[x - 1] = '1' if one_count % 2 != 0 else '0'

return ''.join(hamming_code[::-1])

# Function to detect and correct errors in Hamming Code
def detect_and_correct_hamming_code(data):
n = len(data)
r = calculate_redundant_bits(n)
error_position = 0
for i in range(r):
x = 2 ** i
one_count = 0
for j in range(1, n + 1):
if j & x and data[-j] == '1':
one_count += 1
if one_count % 2 != 0:
error_position += x

if error_position > 0:
print(f"Error detected at position: {error_position}")
data = list(data)
data[-error_position] = '1' if data[-error_position] == '0' else '0'
data = ''.join(data)
else:
print("No error detected.")

# Removing redundant bits
corrected_data = []
j = 0
for i in range(1, n + 1):
if i != 2 ** j:
corrected_data.append(data[-i])
else:
```

```
j += 1

return ''.join(corrected_data[::-1])

# Sender Program
def sender(text):
binary_data = text_to_binary(text)
encoded_data = apply_hamming_code(binary_data)
with open("channel.txt", "w") as f:
f.write(encoded_data)
print("Data has been sent to channel.txt.")

# Receiver Program
def receiver():
with open("channel.txt", "r") as f:
encoded_data = f.read()
print(f"Received encoded data: {encoded_data}")
corrected_data = detect_and_correct_hamming_code(encoded_data)
decoded_text = binary_to_text(corrected_data)
print(f"Decoded text after error correction: {decoded_text}")

#  Example Usage
# Enter text to send
text_to_send = "Hello" # Modify the text to test
sender(text_to_send)
receiver()
```

Input:-
**Text to send**: "Hello"

Output:

Received encoded data: 0100100001100110101101100011010100011001110100
No error detected.
Decoded text after error correction: Hello

# CS19541-COMPUTER NETWORKS

**AIM: Write a program to implement flow control at data link layer using SLIDING WINDOW PROTOCOL. Simulate the flow of frames from one node to another.**

Program should achieve at least below given requirements. You can make it a bidirectional program wherein receiver is sending its data frames with acknowledgement (Piggybacking).

**Create a sender program with following features:-**

1. Input Window size from the user.
2. Input a Text message from the user.
3. Consider 1 character per frame.
4. Create a frame with following fields [Frame no., DATA].
5. Send the frames. [Print the output on screen and save it in a file called Sender_Buffer.]
6. Wait for the acknowledgement from the Receiver. [Induce delay in the program]
7. Reader a file called Receiver_Buffer.
8. Check ACK field for the Acknowledgement number.
9. If the Acknowledgement number is as expected, send new set of frames accordingly, [overwrite the Sender_Buffer file with new frames] Else if NACK is received, resend the frames accordingly. [Overwrite the Sender_Buffer with old frame].

**Create a receiver file with following features**

1. Reader a file called Sender_Buffer.
2. Check the Frame no.
3. If the Fame no. are as expected, write the appropriate ACK no. in the Receiver_ Buffer file. Else write NACK no. in the Receiver_Buffer file.

**NOTE: Induce error and verify the behaviour of the program. Manually Change the Frame no and Ack no in the files].**

## Student observation:

Write the code here:

```
import random

# Constants for ACK and NACK
ACK = "ACK"
NACK = "NACK"

# Sliding window protocol implementation
def sliding_window_protocol(window_size, message):
# Split the message into frames (1 character per frame)
frames = [(i, message[i]) for i in range(len(message))]
current_window_start = 0
current_window_end = min(window_size, len(frames))
expected_ack = current_window_start

while current_window_start < len(frames):
print("\n--- SENDING WINDOW ---")
```

```
# Send frames in the current window
for i in range(current_window_start, current_window_end):
frame_no, data = frames[i]
print(f"Sent frame {frame_no} with data '{data}'.")

# Simulate Receiver side: Acknowledge frames
print("\n--- RECEIVER WINDOW ---")
for i in range(current_window_start, current_window_end):
frame_no, data = frames[i]

# Randomly decide to send ACK or NACK (to simulate error)
ack_type = ACK if random.random() > 0.2 else NACK  # 20% chance of NACK
if ack_type == ACK:
print(f"Frame {frame_no} received correctly. Sending {ACK}.")
else:
print(f"Error in frame {frame_no}. Sending {NACK}.")
expected_ack = frame_no  # Reset to the frame needing retransmission
break
else:
# If no errors in the window, move to the next set of frames
expected_ack = current_window_end

# Update window based on the acknowledgments received
if expected_ack == current_window_end:
print("All frames in the window acknowledged correctly. Moving window forward.")
current_window_start += window_size
current_window_end = min(current_window_start + window_size, len(frames))
else:
print(f"Resending frames from frame {expected_ack} due to error.")
current_window_start = expected_ack
current_window_end = min(current_window_start + window_size, len(frames))

print("\nAll frames sent and acknowledged successfully.")

# User input for window size and message
window_size = int(input("Enter window size: "))
message = input("Enter the message to send: ")

# Run the sliding window protocol
sliding_window_protocol(window_size, message)
```

Input:

Enter window size: 3
Enter the message to send: HELLO

Output:

--- SENDING WINDOW ---
Sent frame 0 with data 'H'.
Sent frame 1 with data 'E'.
Sent frame 2 with data 'L'.
--- RECEIVER WINDOW ---
Error in frame 0.
Sending NACK.
Resending frames from frame 0 due to error.
--- SENDING WINDOW ---
Sent frame 0 with data 'H'.
Sent frame 1 with data 'E'.
Sent frame 2 with data 'L'.
--- RECEIVER WINDOW ---
Frame 0 received correctly.
Sending ACK.
Frame 1 received correctly.
Sending ACK.
Frame 2 received correctly.
Sending ACK.
All frames in the window acknowledged correctly.
Moving window forward.
--- SENDING WINDOW ---
Sent frame 3 with data 'L'.
Sent frame 4 with data 'O'.
--- RECEIVER WINDOW ---
Error in frame 3.
Sending NACK.
Resending frames from frame 3 due to error.
--- SENDING WINDOW ---
Sent frame 3 with data 'L'.
Sent frame 4 with data 'O'.
--- RECEIVER WINDOW ---
Frame 3 received correctly.
Sending ACK. Error in frame 4.
Sending NACK.
Resending frames from frame 4 due to error.

--- SENDING WINDOW ---
Sent frame 4 with data 'O'.
--- RECEIVER WINDOW ---
Frame 4 received correctly.
Sending ACK.
All frames in the window acknowledged correctly.
Moving window forward.
All frames sent and acknowledged successfully.

## Practical-5

## AIM Experiments on Packet capture tool: Wireshark

**Packet Sniffer**

- Sniffs messages being sent/received from/by your computer
- Store and display the contents of the various protocol fields in the messages
- Passive program
  - never sends packets itself
  - no packets addressed to it
  - receives a copy of all packets (sent/received)

**Packet Sniffer Structure Diagnostic Tools**

- Tcpdump
  - E.g. tcpdump -enx host 10.129.41.2 -w exe3.out
- Wireshark
  - wireshark -r exe3.out

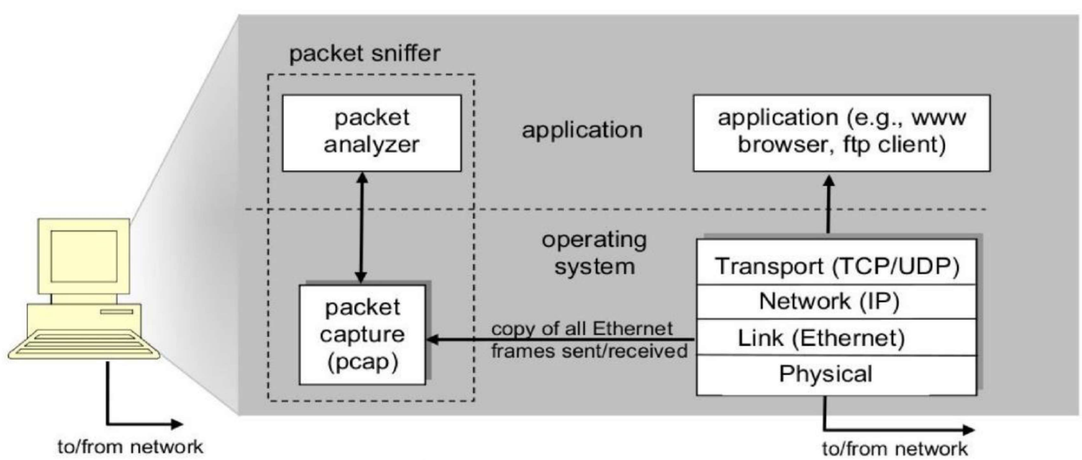



**Figure 1:** Packet sniffer structure

**DESCRIPTION:**

**WIRESHARK**

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets. You can use Wireshark to inspect a suspicious program's network traffic, analyze the traffic flow on your network, or troubleshoot network problems.

**What we can do with Wireshark:**

- Capture network traffic
- Decode packet protocols using dissectors
- Define filters – capture and display
- Watch smart statistics
- Analyze problems
- Interactively browse that traffic

**Wireshark used for:**

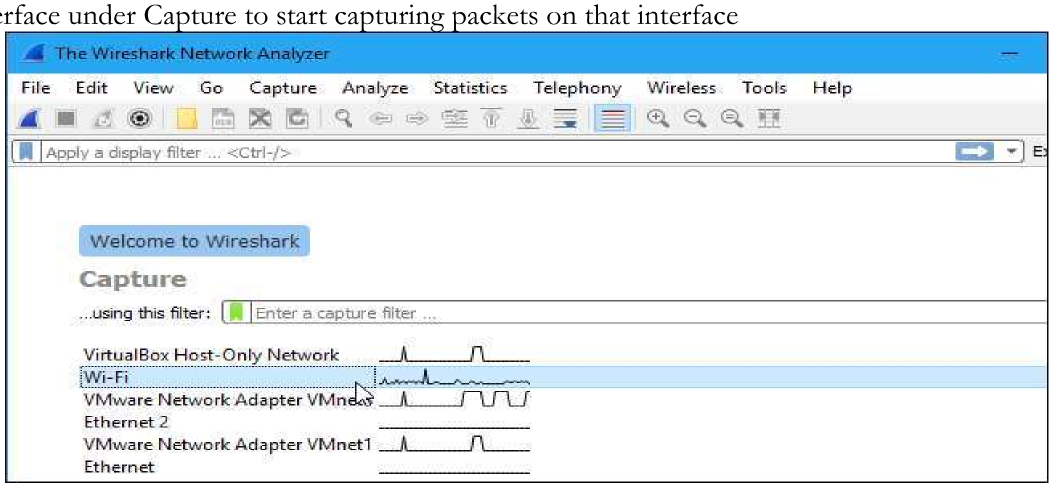
- Network administrators: troubleshoot network problems
- Network security engineers: examine security problems
- Developers: debug protocol implementations
- People: learn **network protocol internals**

**Getting Wireshark**

Wireshark can be downloaded for Windows or macOS from its official website. For Linux or another UNIX-like system, Wireshark will be found in its package repositories. For Ubuntu, Wireshark will be found in the Ubuntu Software Center.

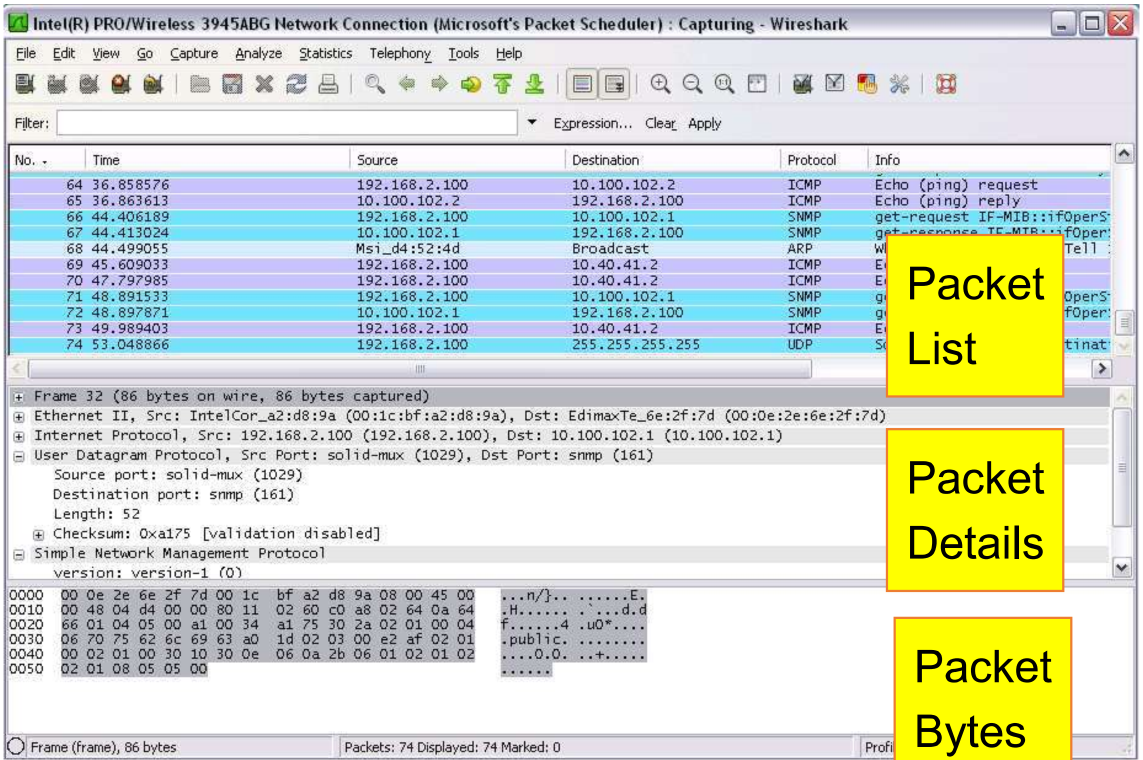
**Capturing Packets**

After downloading and installing Wireshark, launch it and double-click the name of a network interface under Capture to start capturing packets on that interface




As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.

If you have promiscuous mode enabled—it's enabled by default—you'll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click Capture > Options and verify the "Enable promiscuous mode on all interfaces" checkbox is activated at the bottom of this window.




Click the red “Stop” button near the top left corner of the window when you want to stop capturing traffic.

**The “Packet List” Pane**

The packet list pane displays all the packets in the current capture file. The “Packet List” pane Each line in the packet list corresponds to one packet in the capture file. If you select a line in this pane, more details will be displayed in the “Packet Details” and “Packet Bytes” panes.

**The “Packet Details” Pane**

The packet details pane shows the current packet (selected in the “Packet List” pane) in a more detailed form. This pane shows the protocols and protocol fields of the packet selected in the “Packet List” pane. The protocols and fields of the packet shown in a tree which can be expanded and collapsed.

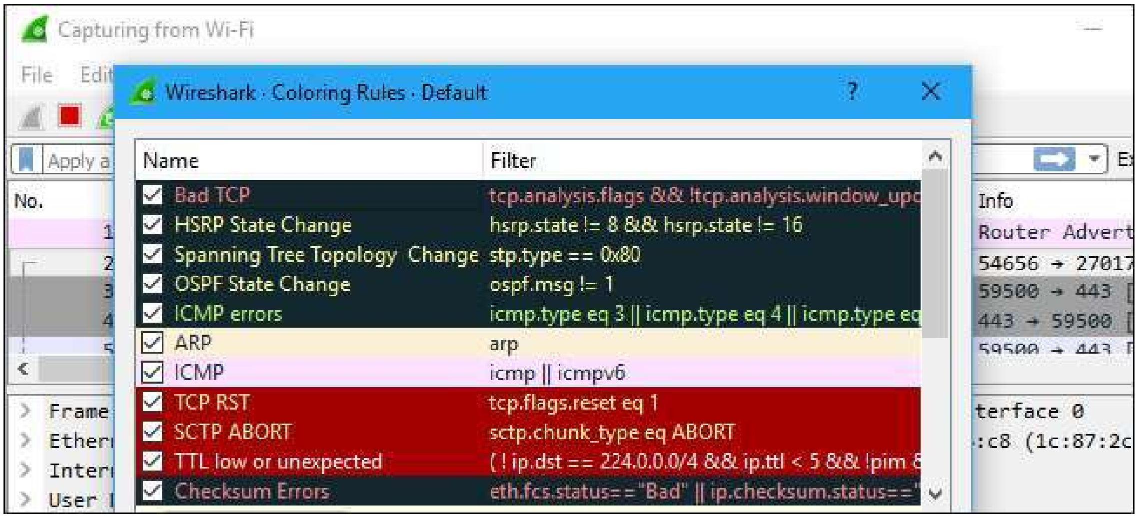
**The “Packet Bytes” Pane**

The packet bytes pane shows the data of the current packet (selected in the “Packet List” pane) in a hexdump style.

**Color Coding**

You’ll probably see packets highlighted in a variety of different colors. Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors—for example, they could have been delivered out of order.

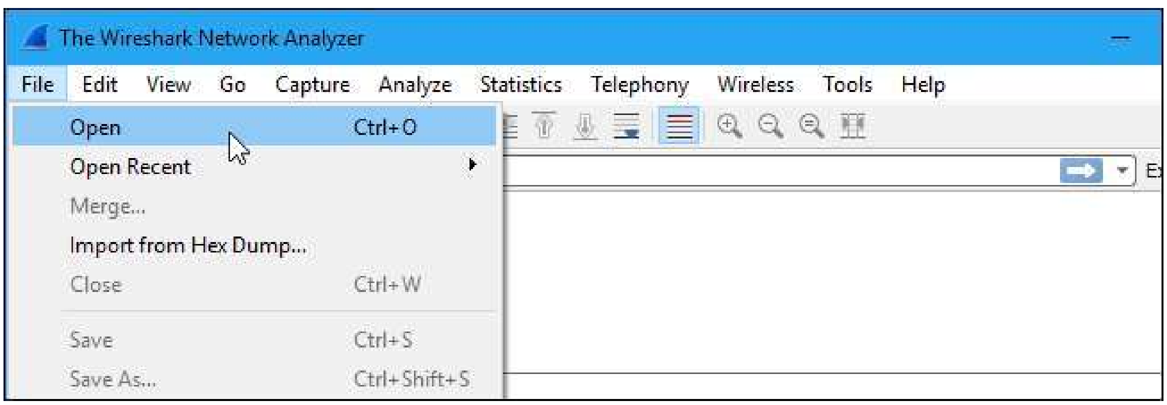
To view exactly what the color codes mean, click View > Coloring Rules. You can also customize and modify the coloring rules from here, if you like.




**Sample Captures**

If there's nothing interesting on your own network to inspect, Wireshark's wiki has you covered. The wiki contains a page of sample capture files that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one.

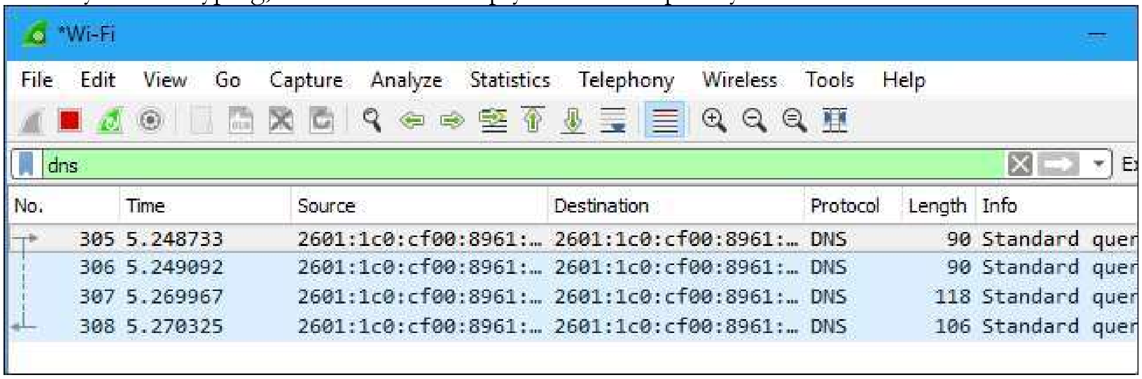
You can also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.




**Filtering Packets**

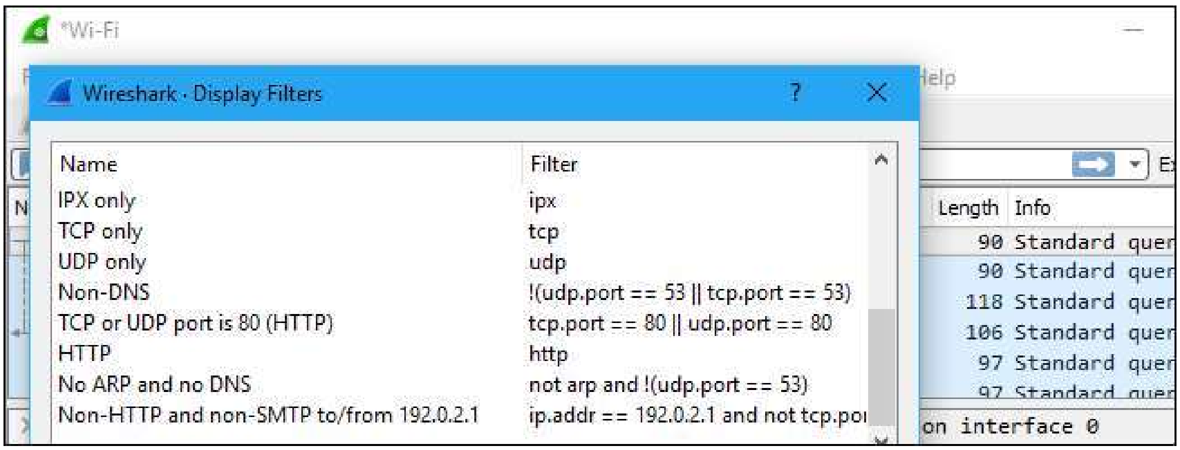
If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you'll likely have a large amount of packets to sift through. That's where Wireshark's filters come in.

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type "dns" and you'll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.

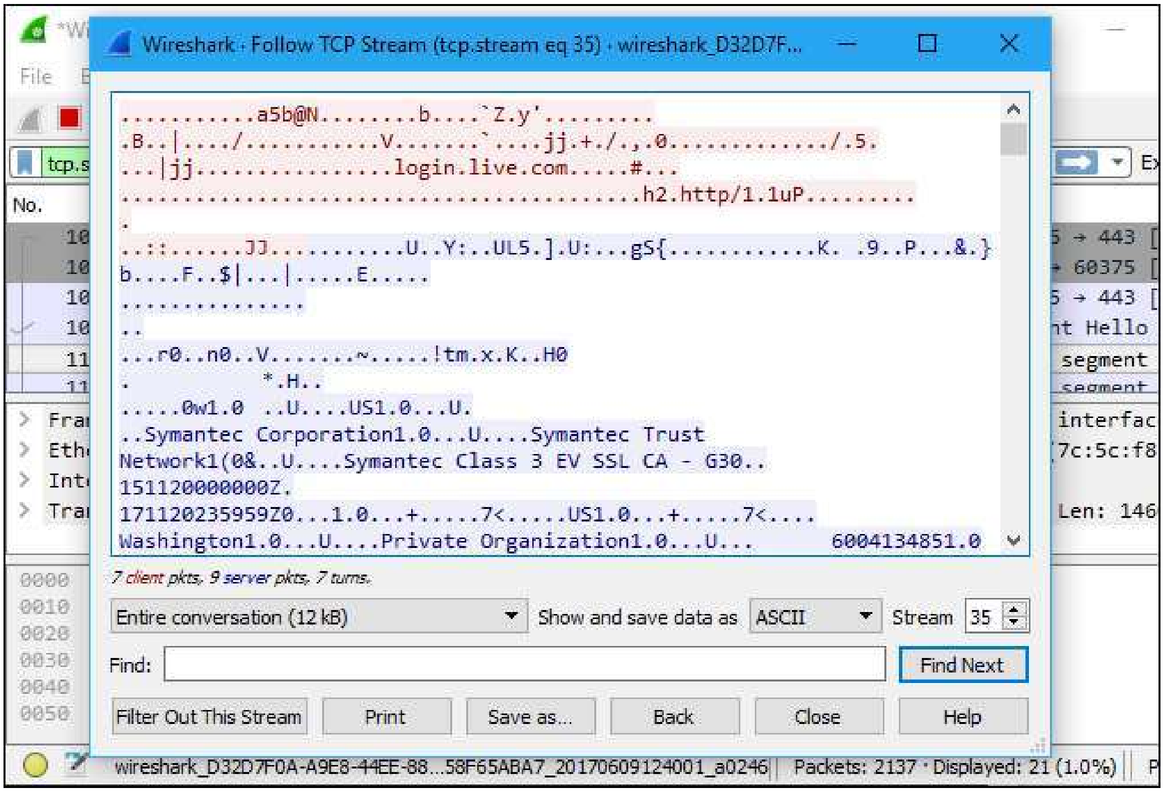



You can also click Analyze > Display Filters to choose a filter from among the default filters included in Wireshark. From here, you can add your own custom filters and save them to easily access them in the future.

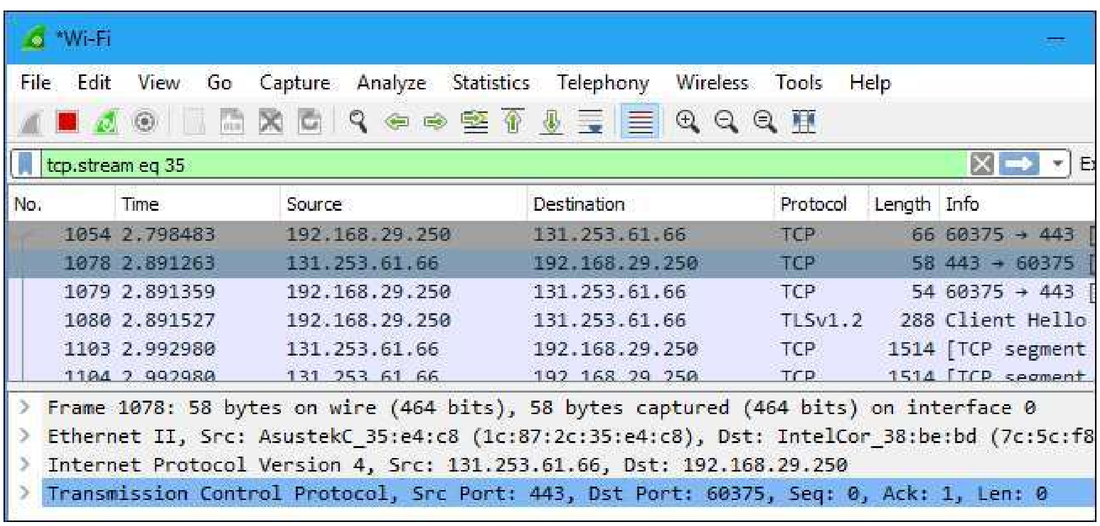
For more information on Wireshark's display filtering language, read the Building display filter expressions page in the official Wireshark documentation.




Another interesting thing you can do is right-click a packet and select Follow > TCP Stream. You'll see the full TCP conversation between the client and the server. You can also click other protocols in the Follow menu to see the full conversations for other protocols, if applicable.

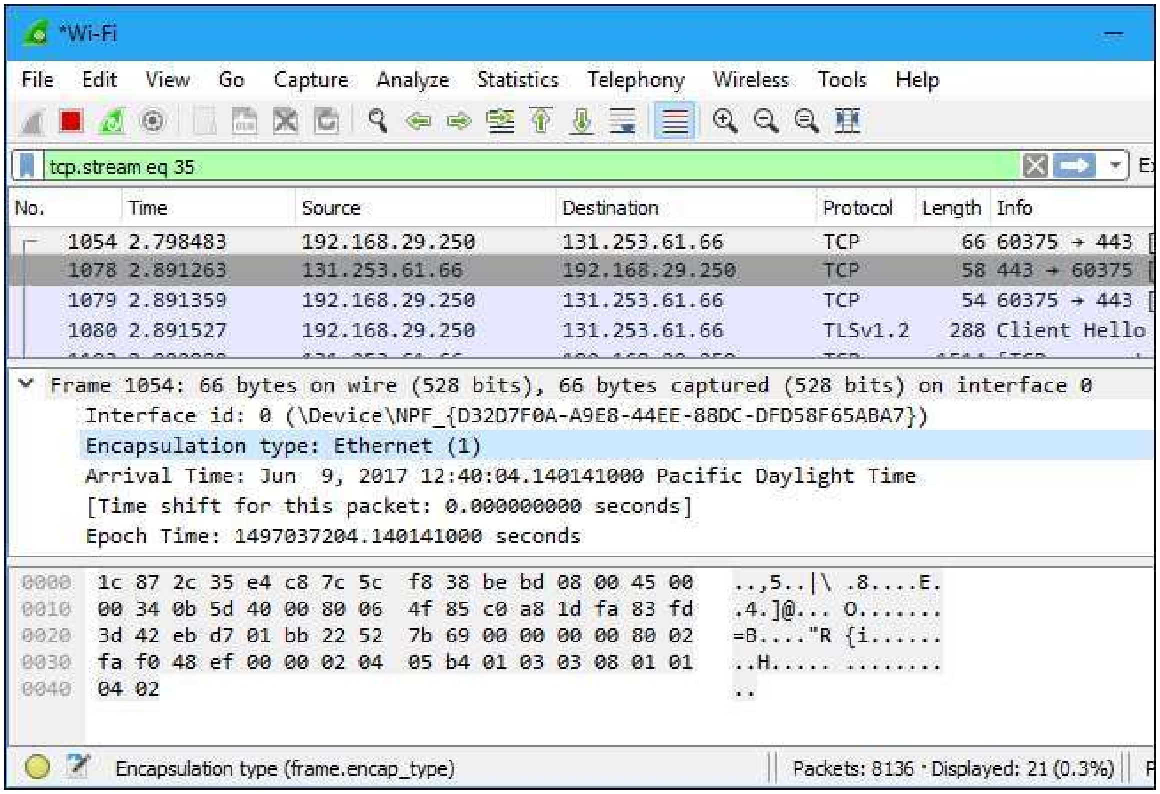


Close the window and you'll find a filter has been applied automatically. Wireshark is showing you the packets that make up the conversation.

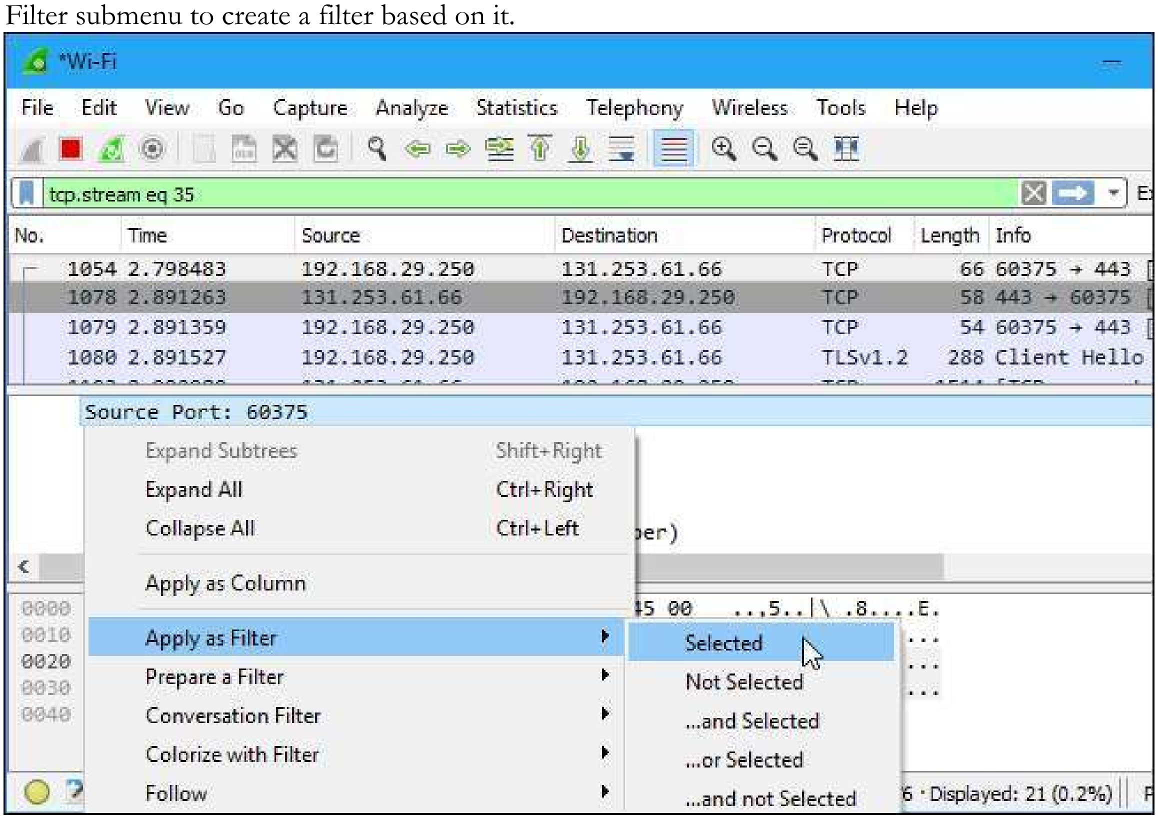


**Inspecting Packets**

Click a packet to select it and you can dig down to view its details.

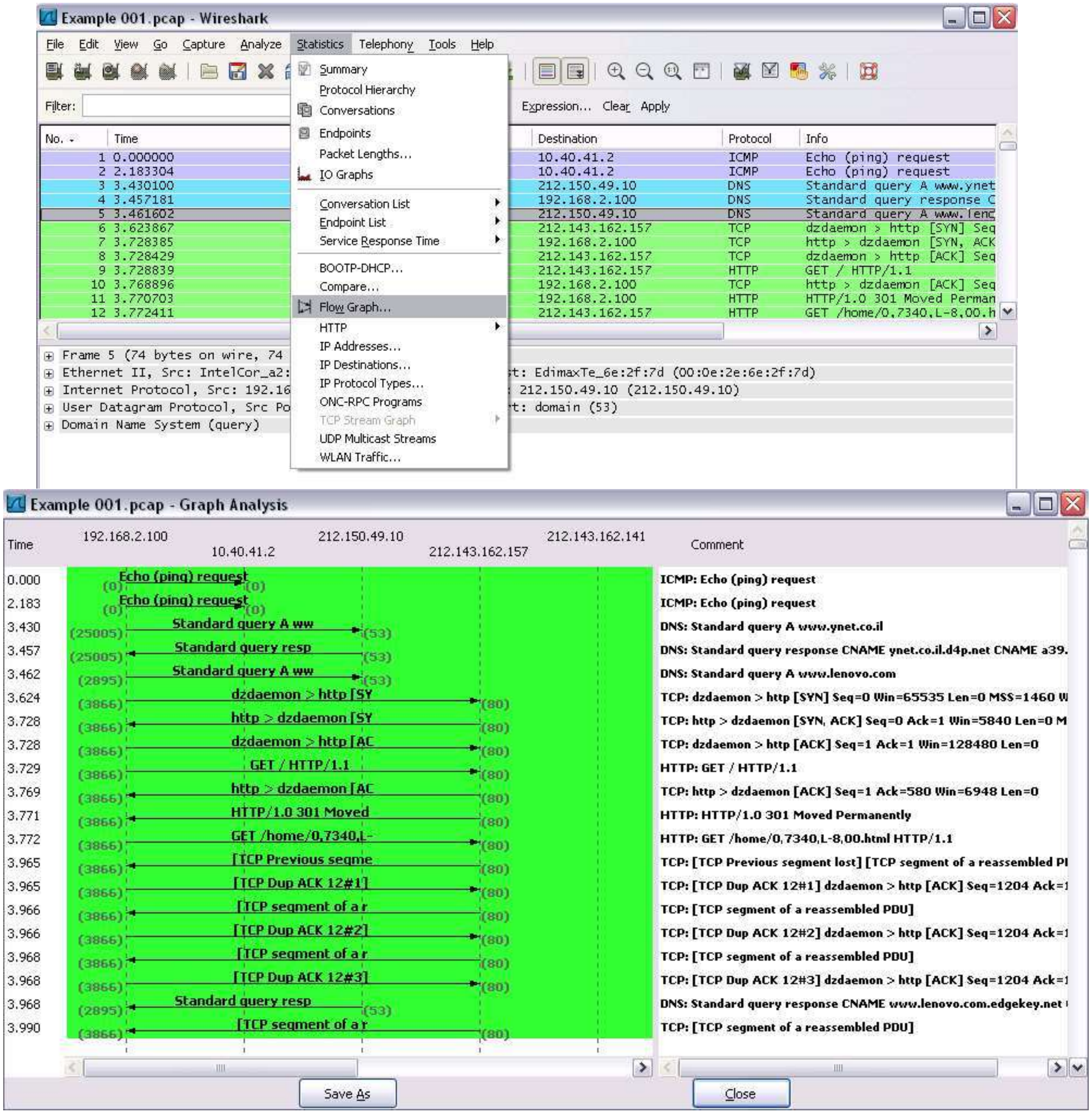



You can also create filters from here — just right-click one of the details and use the Apply as Filter submenu to create a filter based on it.



Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

**Flow Graph: Gives a better understanding of what we see.**



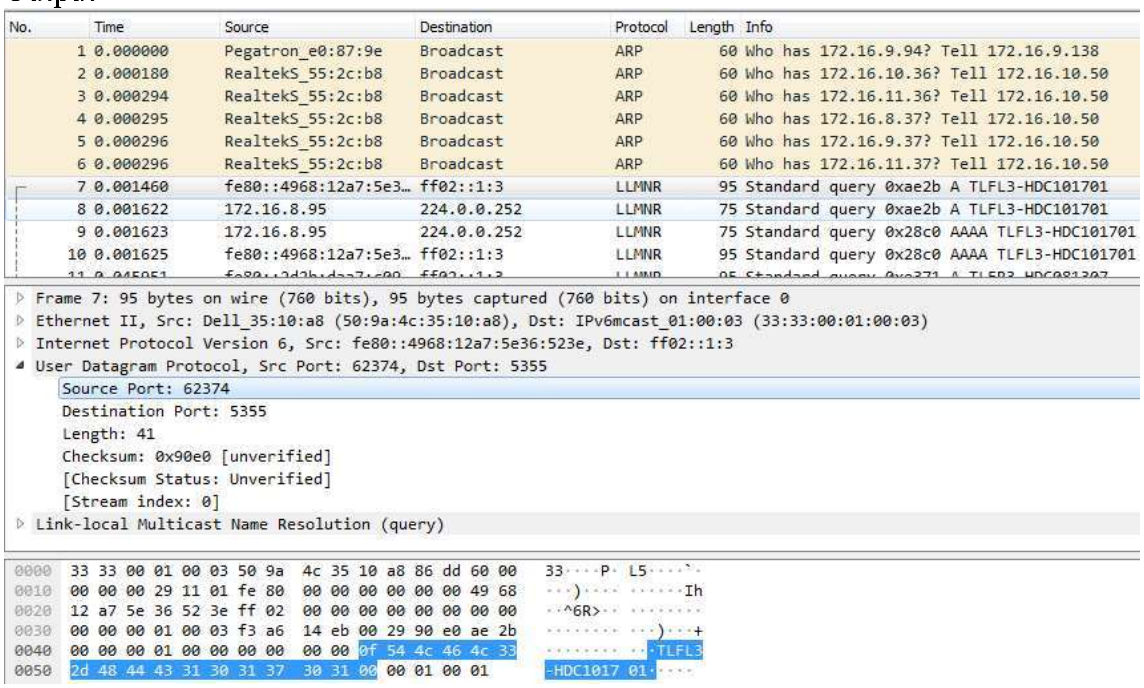
## CAPTURING AND ANALYSING PACKETS USING WIRESHARK TOOL

To filter, capture, view, packets in Wireshark Tool.
Capture 100 packets from the Ethernet: IEEE 802.3 LAN Interface and save it.

**Procedure**

- Select Local Area Connection in Wireshark.
- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Save the packets.

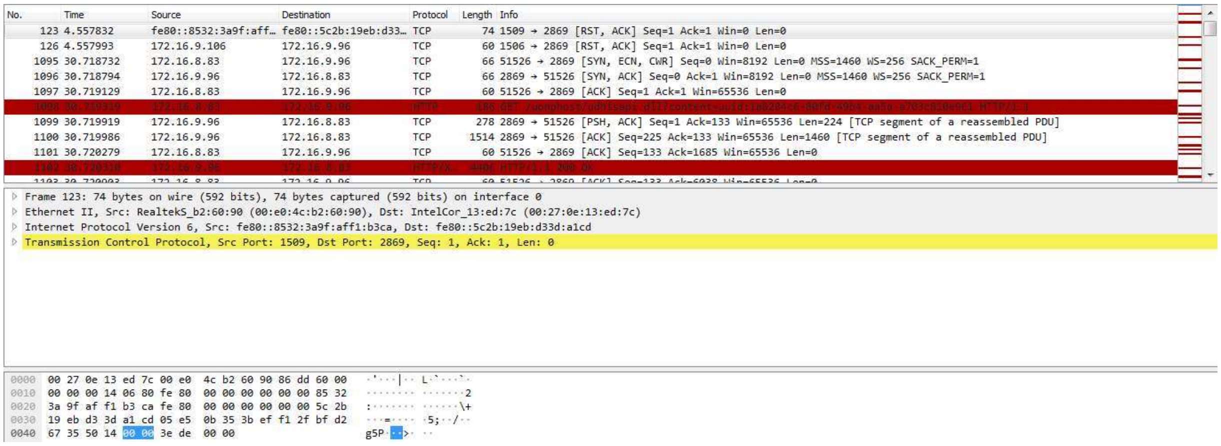
**Output**



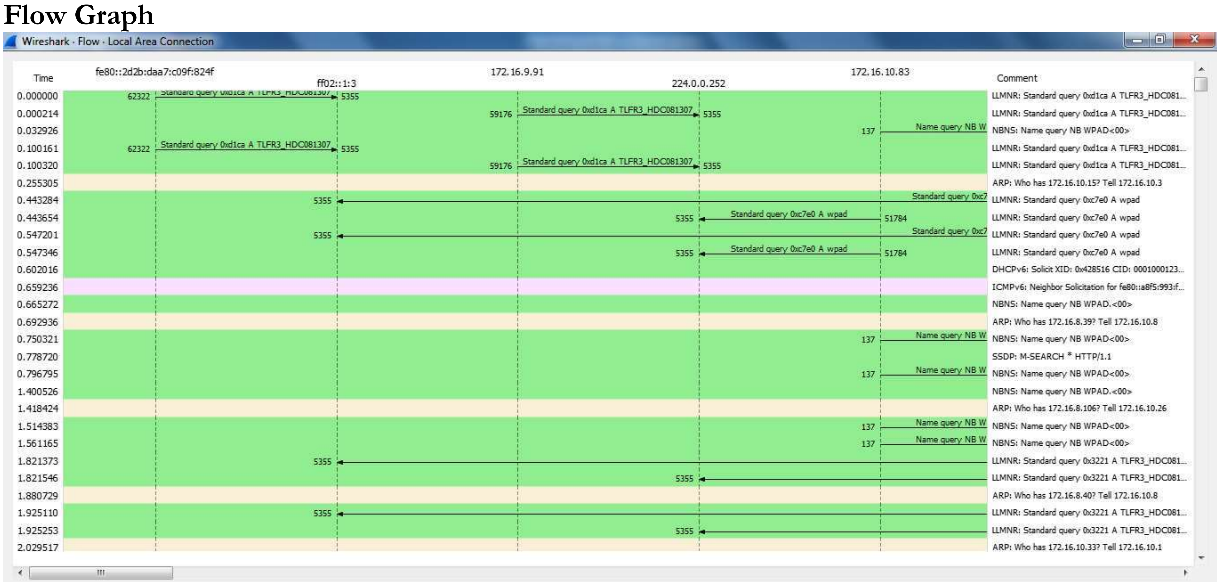

1. **Create a Filter to display only TCP/UDP packets, inspect the packets and provide the flow graph**

**Procedure**

- Select Local Area Connection in Wireshark.
- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search TCP packets in search bar.
- To see flow graph click Statistics→Flow graph.
- Save the packets.



## Flow Graph

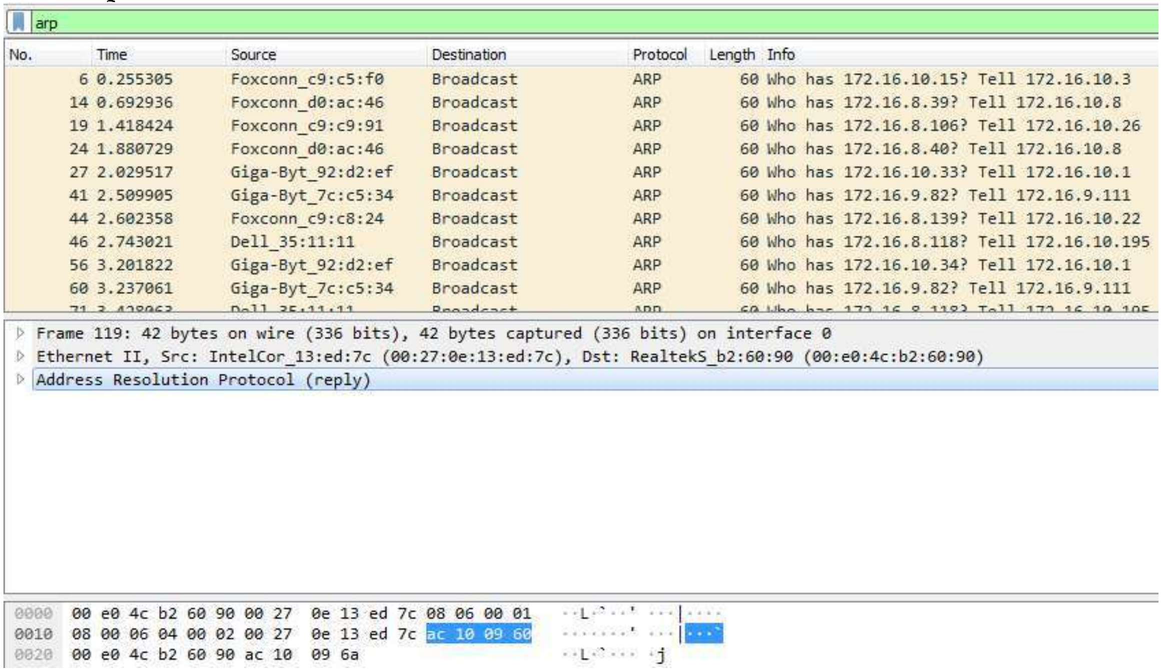


**2. Create a Filter to display only ARP packets and inspect the packets.**

**Procedure**

- ➢ Go to capture → option
- ➢ Select stop capture automatically after 100 packets.
- ➢ Then click Start capture.
- ➢ Search ARP packets in search bar.
- ➢ Save the packets.

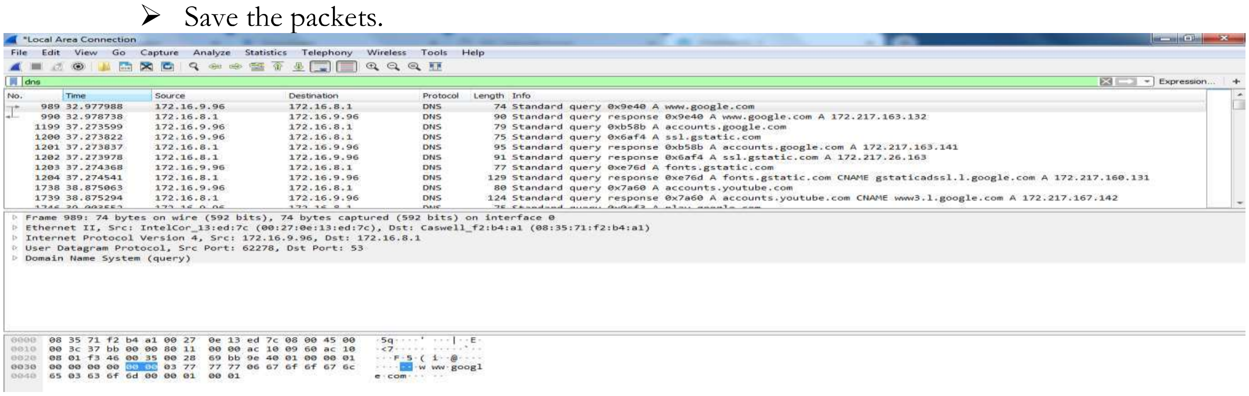
**Output**

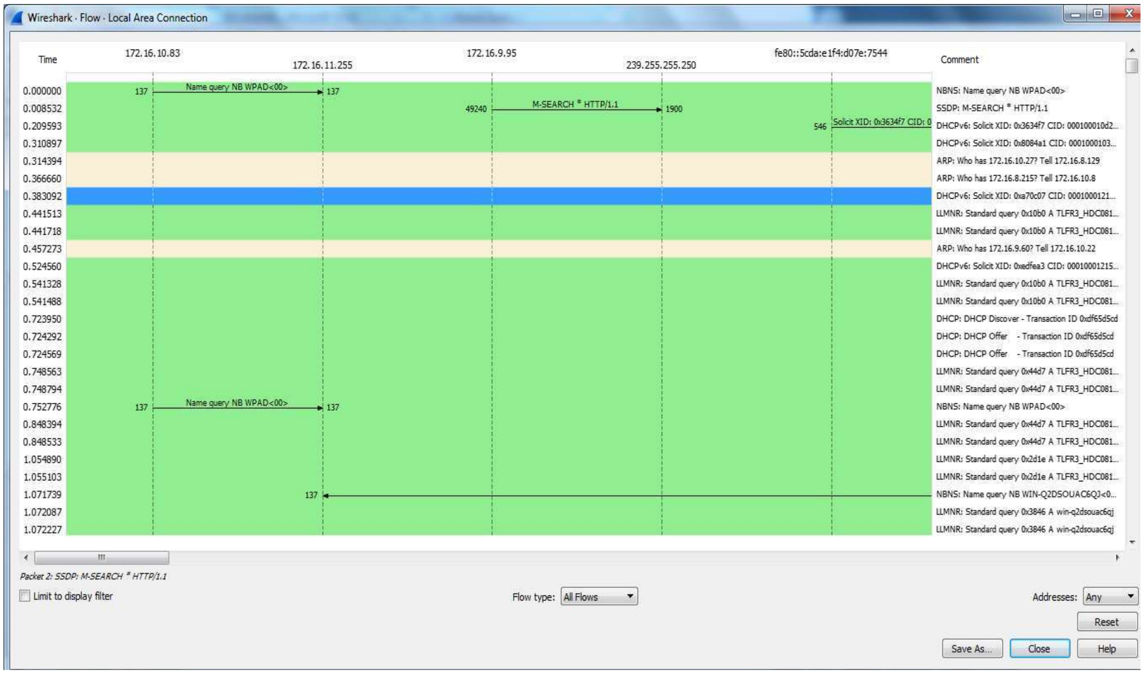



**3. Create a Filter to display only DNS packets and provide the flow graph.**

**Procedure**

- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search DNS packets in search bar.
- To see flow graph click Statistics→Flow graph.
- Save the packets.

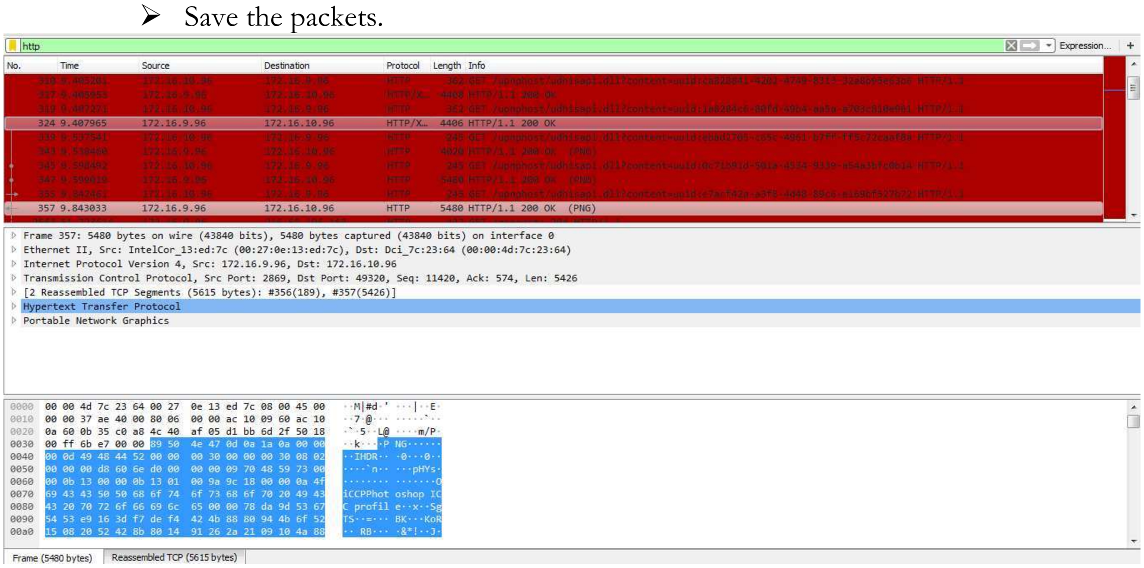





### 4. Create a Filter to display only HTTP packets and inspect the packets

**Procedure**

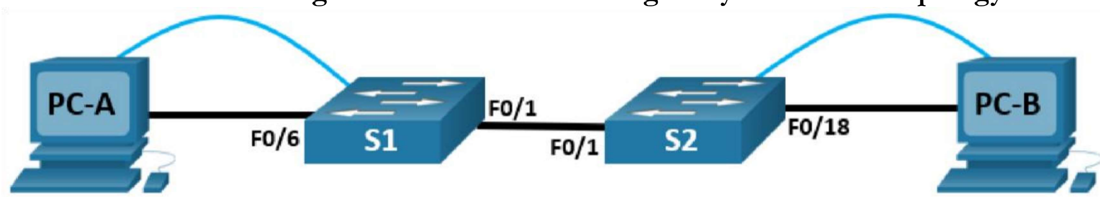
- ➢ Select Local Area Connection in Wireshark.
- ➢ Go to capture → option
- ➢ Select stop capture automatically after 100 packets.
- ➢ Then click Start capture.
- ➢ Search HTTP packets in search bar.
- ➢ Save the packets.



## Practical-8

**AIM: - a) Simulate Virtual LAN configuration using CISCO Packet Tracer Simulation.**

**Packet Tracer - Configure VLANs and Trunking - Physical Mode Topology**




**Addressing Table**

| Device | Interface | IP Address | Subnet Mask | Default Gateway |
|---|---|---|---|---|
| S1 | VLAN 1 | 192.168.1.11 | 255.255.255.0 | N/A |
| S2 | VLAN 1 | 192.168.1.12 | 255.255.255.0 | N/A |
| PC-A | NIC | 192.168.10.3 | 255.255.255.0 | 192.168.10.1 |
| PC-B | NIC | 192.168.10.4 | 255.255.255.0 | 192.168.10.1 |

*Blank Line - no additional information*

**Objectives**

**Part 1: Build the Network and Configure Basic Device Settings**
**Part 2: Create VLANs and Assign Switch Ports**
**Part 3: Maintain VLAN Port Assignments and the VLAN Database Part 4: Configure an 802.1Q Trunk between the Switches**

**Background / Scenario**

Modern switches use virtual local-area networks (VLANs) to improve network performance by separating large Layer 2 broadcast domains into smaller ones. VLANs can also be used as a security measure by controlling which hosts can communicate. In general, VLANs make it easier to design a network to support the goals of an organization.
VLAN trunks are used to span VLANs across multiple devices. Trunks allow the traffic from multiple VLANS to travel over a single link, while keeping the VLAN identification and segmentation intact.
In this Packet Tracer Physical Mode (PTPM) activity, you will create VLANs on both switches in the topology, assign VLANs to switch access ports, and verify that VLANs are working as expected. You will then create a VLAN trunk between the two switches to allow hosts in the same VLAN to communicate through the trunk, regardless of which switch to which the host is attached.

**Instructions**

### Part 1: Build the Network and Configure Basic Device Settings

**Step 1: Build the network as shown in the topology.**

Attach the devices as shown in the topology diagram, and cable as necessary. a. Click and drag both switch **S1** and **S2** to the **Rack**.

b. Click and drag both **PC-A** and **PC-B** to the **Table** and use the power button to turn them on.

c. Provide network connectivity by connecting **Copper Straight-through** cables, as shown in the topology.
d. Connect **Console Cable** from device **PC-A** to **S1** and from device **PC-B** to **S2**.

**Step 2: Configure basic settings for each switch.**

a. From the **Desktop Tab** on each PC, use the **Terminal** to console into each switch and enable privileged EXEC mode.
   *Open configuration window*
b. Enter configuration mode.
c. Assign a device name to each switch.
d. Assign **class** as the privileged EXEC encrypted password.
e. Assign **cisco** as the console password and enable login.
f. Assign **cisco** as the vty password and enable login.
g. Encrypt the plaintext passwords.
h. Create a banner that warns anyone accessing the device that unauthorized access is prohibited.
i. Configure the IP address listed in the Addressing Table for VLAN 1 on the switch.
   **Note**: The VLAN 1 address is not grade because you will remove it later in the activity. However, you will need VLAN 1 to test connectivity later in this Part.
j. Shut down all interfaces that will not be used.
k. Set the clock on each switch.
   **Note**: The clock setting cannot be graded in Packet Tracer.
l. Save the running configuration to the startup configuration file.
   *Close configuration window*

**Step 3: Configure PC hosts.**

From the **Desktop** tab on each **PC**, click IP Configuration and enter the addressing information as displayed in the Addressing Table.

**Step 4: Test connectivity.**

Test network connectivity by attempting to ping between each of the cabled devices.

Questions:

Can PC-A ping PC-B?
Can PC-A ping S1?
Can PC-B ping S2?
Can S1 ping S2?

*Close configuration window*

## Part 2: Create VLANs and Assign Switch Ports

In Part 2, you will create Management, Operations, Parking Lot, and Native VLANs on both switches. You will then assign the VLANs to the appropriate interface. The **show vlan** command is used to verify your configuration settings.

**Step 1: Create VLANs on the switches.**

From the **Desktop Tab** on each **PC**, use Terminal to continue configuring both network switches.

*Open configuration window*

a. Create the VLANs on **S1**.

S1(config)# **vlan 10**
S1(config-vlan)# **name Operations**
S1(config-vlan)# **vlan 20**
S1(config-vlan)# **name Parking_Lot**
S1(config-vlan)# **vlan 99**
S1(config-vlan)# **name Management**
S1(config-vlan)# **vlan 1000**
S1(config-vlan)# **name Native**
S1(config-vlan)# **end**

b. Create the same VLANs on S2.

c. Issue the **show vlan brief** command to view the list of VLANs on **S1**.

S1# **show vlan brief**

```
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                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
                                                Gi0/1, Gi0/2
10   Operations                       active
20   Parking_Lot                      active
99   Management                       active
1000 Native                           active
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
```

Questions:

What is the default VLAN?

What ports are assigned to the default VLAN?

**Step 2: Assign VLANs to the correct switch interfaces.**

a. Assign VLANs to the interfaces on **S1**.
   1) Assign PC-A to the Operation VLAN.
      S1(config)# **interface f0/6**
      S1(config-if)# **switchport mode access**
      S1(config-if)# **switchport access vlan 10**
   2) From VLAN 1, remove the management IP address and configure it on VLAN 99.
      S1(config)# **interface vlan 1**
      S1(config-if)# **no ip address**
      S1(config-if)# **interface vlan 99**
      S1(config-if)# **ip address 192.168.1.11 255.255.255.0**
      S1(config-if)# **end**

b. Issue the **show vlan brief** command and verify that the VLANs are assigned to the correct interfaces. c. Issue the **show ip interface brief** command.
   Question:
   What is the status of VLAN 99? Explain.

d. Assign **PC-B** to the Operations VLAN on **S2**.
e. From VLAN 1, remove the management IP address and configure it on VLAN 99 according to the Addressing Table.
f. Use the **show vlan brief** command to verify that the VLANs are assigned to the correct interfaces.
   Questions:
   Is S1 able to ping S2? Explain.
   Is PC-A able to ping PC-B? Explain.

## Part 3: Maintain VLAN Port Assignments and the VLAN Database

In Part 3, you will change port VLAN assignments and remove VLANs from the VLAN database.

**Step 1: Assign a VLAN to multiple interfaces.**

From the **Desktop Tab** on each **PC**, use **Terminal** to continue configuring both network switches.

*Open configuration window*

a. On S1, assign interfaces F0/11 – 24 to VLAN99.
   S1(config)# **interface range f0/11-24**
   S1(config-if-range)# **switchport mode access**
   S1(config-if-range)# **switchport access vlan 99**
   S1(config-if-range)# **end**
b. Issue the **show vlan brief** command to verify VLAN assignments.
c. Reassign F0/11 and F0/21 to VLAN 10.
d. Verify that VLAN assignments are correct.

**Step 2: Remove a VLAN assignment from an interface.**

a. Use the **no switchport access vlan** command to remove the VLAN 99 assignment to F0/24.
   S1(config)# **interface f0/24**
   S1(config-if)# **no switchport access vlan**
   S1(config-if)# **end**
b. Verify that the VLAN change was made.
   Question:

Which VLAN is F0/24 now associated with?

**Step 3: Remove a VLAN ID from the VLAN database.**

a. Add VLAN 30 to interface F0/24 without issuing the global VLAN command.
S1(config)# **interface f0/24**
S1(config-if)# **switchport access vlan 30**
% Access VLAN does not exist. Creating vlan 30
**Note**: Current switch technology no longer requires that the **vlan** command be issued to add a VLAN to the database. By assigning an unknown VLAN to a port, the VLAN will be created and added to the VLAN database.

b. Verify that the new VLAN is displayed in the VLAN table.
Question:
What is the default name of VLAN 30?

c. Use the **no vlan 30** command to remove VLAN 30 from the VLAN database.
S1(config)# **no vlan 30**
S1(config)# **end**

d. Issue the **show vlan brief** command. F0/24 was assigned to VLAN 30.
Question:
After deleting VLAN 30 from the VLAN database, why is F0/24 no longer displayed in the output of the **show vlan brief** command? What VLAN is port F0/24 now assigned to? What happens to the traffic destined to the host that is attached to F0/24?

e. On interface F0/24, issue the **no switchport access vlan** command.

f. Issue the **show vlan brief** command to determine the VLAN assignment for F0/24.
Questions:
To which VLAN is F0/24 assigned?

**Note**: Before removing a VLAN from the database, it is recommended that you reassign all the ports assigned to that VLAN.

Why should you reassign a port to another VLAN before removing the VLAN from the VLAN database?
*Close configuration window.*

## Part 4: Configure an 802.1Q Trunk Between the Switches

In Part 4, you will configure interface F0/1 to use the Dynamic Trunking Protocol (DTP) to allow it to negotiate the trunk mode. After this has been accomplished and verified, you will disable DTP on interface F0/1 and manually configure it as a trunk.

**Step 1: Use DTP to initiate trunking on F0/1.**

The default DTP mode of a 2960 switch port is dynamic auto. This allows the interface to convert the link to a trunk if the neighboring interface is set to trunk or dynamic desirable mode.

*Open configuration window*

a. On **S1**, set F0/1 to negotiate trunk mode.
S1(config)# **interface f0/1**
S1(config-if)# **switchport mode dynamic desirable**

Sep 19 02:51:47.257: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up
Sep 19 02:51:47.291: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan99, changed state to up
You should also receive link status messages on S2.
S2#
Sep 19 02:42:19.424: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
Sep 19 02:42:21.454: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan99, changed state to up
Sep 19 02:42:22.419: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up

b. On **S1** and **S2**, issue the **show vlan brief** command. Interface F0/1 is no longer assigned to VLAN 1. Trunked interfaces are not listed in the VLAN table.
c. Issue the **show interfaces trunk** command to view trunked interfaces. Notice that the mode on **S1** is set to desirable, and the mode on **S2** is set to auto.
S1# **show interfaces trunk**

S2# **show interfaces trunk**
**Note**: By default, all VLANs are allowed on a trunk. The **switchport trunk** command allows you to control what VLANs have access to the trunk. For this activity, keep the default settings. This allows all VLANs to traverse F0/1.

*Close configuration window*

d. Verify that VLAN traffic is traveling over trunk interface F0/1.
Questions:
Can S1 ping S2?
Can PC-A ping PC-B?
Can PC-A ping S1?
Can PC-B ping S2?

**Step 2: Manually configure trunk interface F0/1.**

The **switchport mode trunk** command is used to manually configure a port as a trunk. This command should be issued on both ends of the link.

a. On interface F0/1, change the switchport mode to force trunking. Make sure to do this on both switches.

*Open configuration window*

S1(config)# **interface f0/1**
S1(config-if)# **switchport mode trunk**

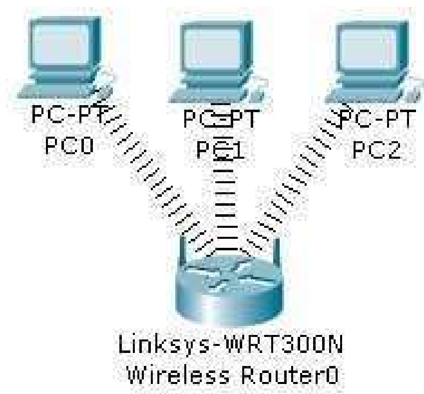
b. Issue the **show interfaces trunk** command to view the trunk mode. Notice that the mode changed from **desirable** to **on**.
S1# **show interfaces trunk**
c. Modify the trunk configuration on both switches by changing the native VLAN from VLAN 1 to VLAN 1000.
S1(config)# **interface f0/1**
S1(config-if)# **switchport trunk native vlan 1000**
d. Issue the **show interfaces trunk** command to view the trunk. Notice the Native VLAN information is updated.
S2# **show interfaces trunk**
Questions:
Why might you want to manually configure an interface to trunk mode instead of using DTP?
Why might you want to change the native VLAN on a trunk?

*Close configuration window*

## Practical-8

## AIM:-b) Configuration of Wireless LAN using CISCO Packet Tracer.

Design a topology with three PCs connected from Linksys Wireless routers.




Perform following configuration:-

- Configure Static IP on PC and Wireless Router
- Set SSID to MotherNetwork
- Set IP address of router to 192.168.0.1, PC0 to 192.168.0.2, PC1 to 192.168.0.3 and PC2 to 192.168.0.4.
- Secure your network by configuring WAP key on Router
- Connect PC by using WAP key

To complete these tasks follow these step by step instructions:-

Step1:- Click on wireless router,

- Select Administration tab from top Menu, set username and password to admin and click on Save Setting.

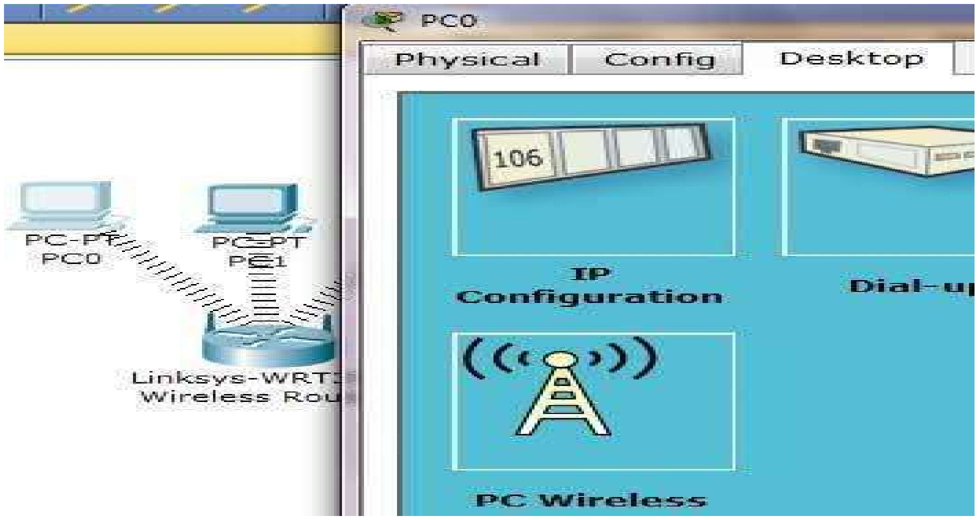
- Next click on wireless tab and set default SSID to MotherNetwork.
- Now Select wireless security and change Security Mode to WEP


- Set Key1 to 0123456789

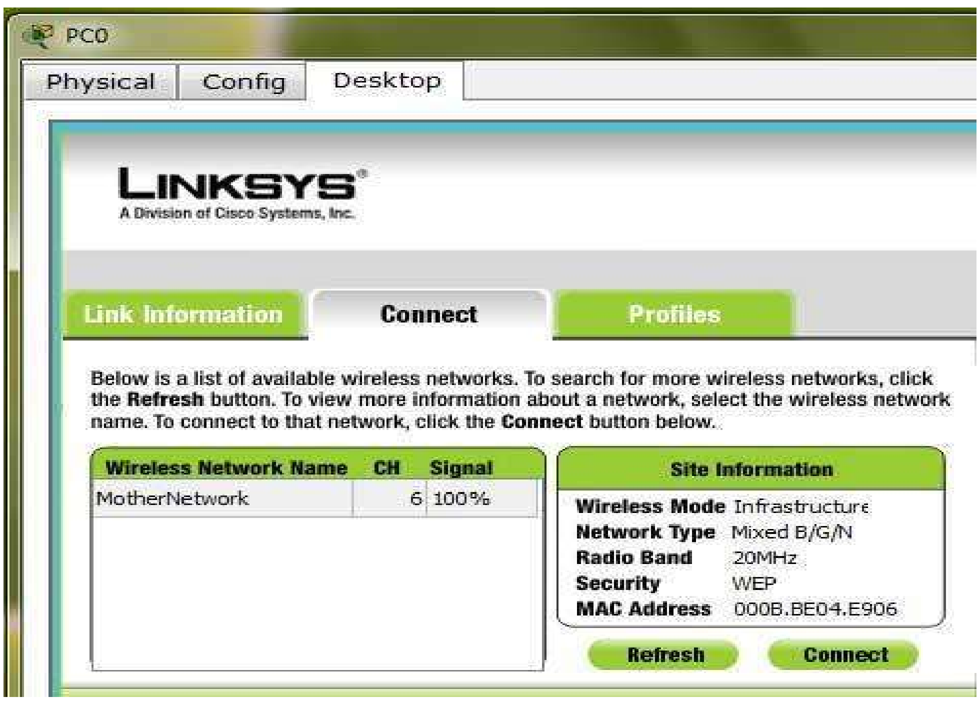

- Again go in the end of page and Click on Save Setting
- Now we have completed all given task on Wireless router. Now configure the static IP on all three PC's
- Double click on pc select Desktop tab click on IP configuration select Static IP and set IP as given below

| PC | IP | Subnet Mask | Default Gateway |
|---|---|---|---|
| PC0 | 192.168.0.2 | 255.255.255.0 | 192.168.0.1 |
| PC1 | 192.168.0.3 | 255.255.255.0 | 192.168.0.1 |
| PC2 | 192.168.0.4 | 255.255.255.0 | 192.168.0.1 |

- Now it's time to connect PC's from Wireless router. To do so click PC select Desktop click on PC Wireless

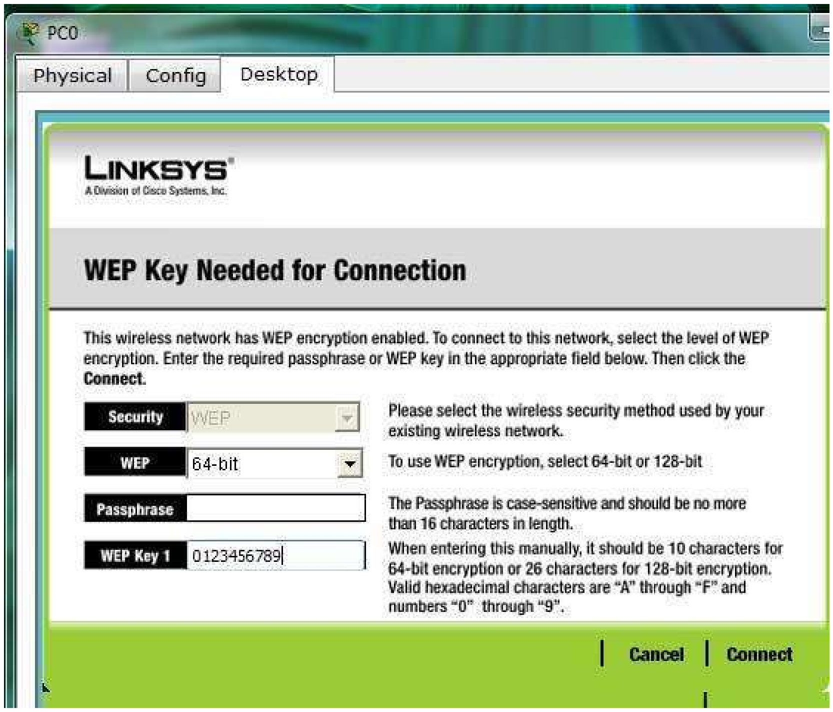



- Click on connect tab and click on Refresh button




As you can see in image that Wireless device is accessing MotherNetwork on CH 6 and signal strength is 100%. In left side you can see that WEP security is configured in network. Click on connect button to connect MotherNetwork

- It will ask for WAP key insert 0123456789 and click connect




It will connect you with wireless router.

As you can see in image below that system is connected. And PCI card is active.


- Repeat same process on PC1 and PC2.

## Practical-9

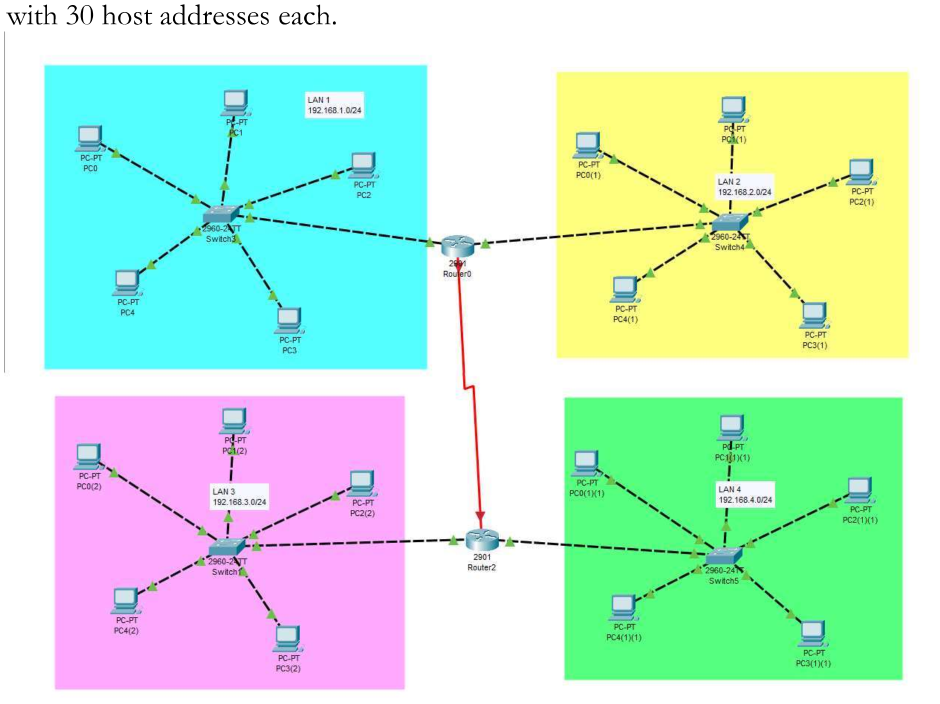
### AIM:-Implementation of SUBNETTING in CISCO PACKET TRACER simulator.

Classless IP subnetting is a technique that allows for more efficient use of IP addresses by allowing for subnet masks that are not just the default masks for each IP class. This means that we can divide our IP address space into smaller subnets, which can be useful when we have a limited number of IP addresses but need to create multiple networks.

CREATING A NETWORK TOPOLOGY:
The first step in implementing classless IP subnetting is to create a network topology in Packet Tracer. To create a network topology in Packet Tracer, select the "New" button in the top left corner, then select "Network" and "Generic". This will create a blank network topology that we can use to add devices.

ADDING THE DEVICES:
Once we have created our network topology, we can add devices to it. Here, we will be adding routers, witches, and PCs. To add a device, select the device from the bottom left corner and drag it onto the network topology. Then, connect the devices by dragging a cable from one device's port to another device's port.

SUBNETTING:
To subnet the network address of 192.168.1.0/24 to provide enough space for at least 5 addresses for end devices, the switch, and the router, we can use a /27 subnet mask. This will give us 8 subnets with 30 host addresses each.



The IP addressing for the network shown in the topology can be as follows:

- Router R1:
- GigabitEthernet0/0: 192.168.1.1
- GigabitEthernet0/1: 192.168.2.1
- Switch S1:
- FastEthernet0/1: 192.168.1.0/27
- PC1: 192.168.1.11
- PC2: 192.168.1.12
- PC3: 192.168.1.13
- PC4: 192.168.1.14
- PC5: 192.168.1.15
- FastEthernet0/2: 192.168.2.0/27
- PC1: 192.168.2.11
- PC2: 192.168.2.12
- PC3: 192.168.2.13
- PC4: 192.168.2.14
- PC5: 192.168.2.15
- Router R2:
- FastEthernet0/0: 192.168.3.1
- FastEthernet0/1: 192.168.4.1
- Switch S2:
- FastEthernet0/1: 192.168.3.0/27
- PC1: 192.168.3.11
- PC2: 192.168.3.12
- PC3: 192.168.3.13
- PC4: 192.168.3.14
- PC5: 192.168.3.15
- FastEthernet0/2: 192.168.4.0/27
- PC1: 192.168.4.11
- PC2: 192.168.4.12
- PC3: 192.168.4.13
- PC4: 192.168.4.14
- PC5: 192.168.4.15

**CONFIGURING THE DEVICES:**

Now that we have added our devices and connected them, we can start configuring them. We will start by configuring the router. Right-click on the router and select "CLI". This will open the command-line interface (CLI) for the router. In the CLI, enter the following commands:

```
#enable
#configure terminal
#interface FastEthernet0/0
#ip address {IP address} {subnet mask}
#no shutdown
#exit

interface FastEthernet0/1
ip address {IP address} {subnet mask}

no shutdown
exit
```

Replace "{IP address}" and "{subnet mask}" with your desired IP address and subnet mask. The first interface, FastEthernet0/0, will be connected to the switch, while the second interface, FastEthernet0/1, will be connected to one of the PCs. These commands configure the router's interfaces with IP addresses and subnet masks.
Next, we will configure the switch. Right-click on the switch and select "CLI". In the CLI, enter the following commands:

```
enable
configure terminal
interface FastEthernet0/1
switchport mode access
exit

interface FastEthernet0/2
switchport mode access
exit
```

These commands configure the switch to operate in access mode on its two ports, which are connected to the two PCs.
Finally, we will configure the PCs. Right-click on each PC and select "Config". In the configuration window, enter the IP address, subnet mask, default gateway, and DNS server information. The IP address and subnet mask should be within the same subnet as the router's FastEthernet0/1 interface.

To configure the GigabitEthernet interface on the router, you can follow these steps:

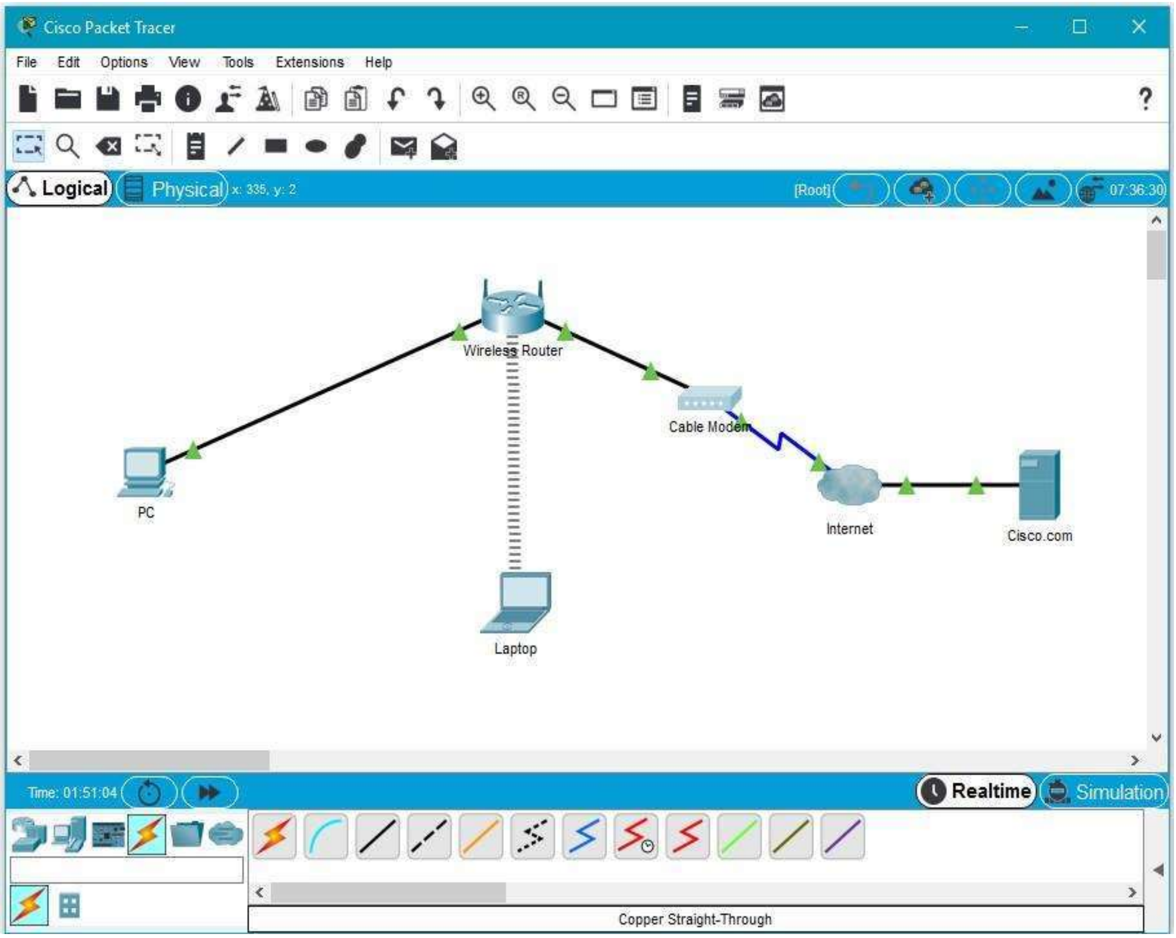
1. Right-click on the router and select "CLI".
2. Enter the following commands:

```
enable
configure terminal
interface GigabitEthernet0/0
ip address {IP address} {subnet mask}
no shutdown
exit
```

Replace "{IP address}" and "{subnet mask}" with your desired IP address and subnet mask. These commands configure the GigabitEthernet interface with an IP address and subnet mask, and enable the interface.

## Practical 10

**AIM:- b) Design and configure an internetwork using wireless router, DHCP server and internet cloud.**




**Addressing Table**

| Device | Interface | IP Address | Subnet Mask | Default Gateway |
|---|---|---|---|---|
| PC | Ethernet0 | DHCP | | 192.168.0.1 |
| Wireless Router | LAN | 192.168.0.1 | 255.255.255.0 | |
| Wireless Router | Internet | DHCP | | |
| Cisco.com Server | Ethernet0 | 208.67.220.220 | 255.255.255.0 | |
| Laptop | Wireless0 | DHCP | | |

**Objectives**

**Part 1: Build a Simple Network in the Logical Topology Workspace**

**Part 2: Configure the Network Devices**
**Part 3: Test Connectivity between Network Devices Part 4: Save the File and Close Packet Tracer**

**Part 1: Build a Simple Network in the Logical Topology Workspace**

**Step 1: Launch Packet Tracer.**

**Step 2: Build the topology**

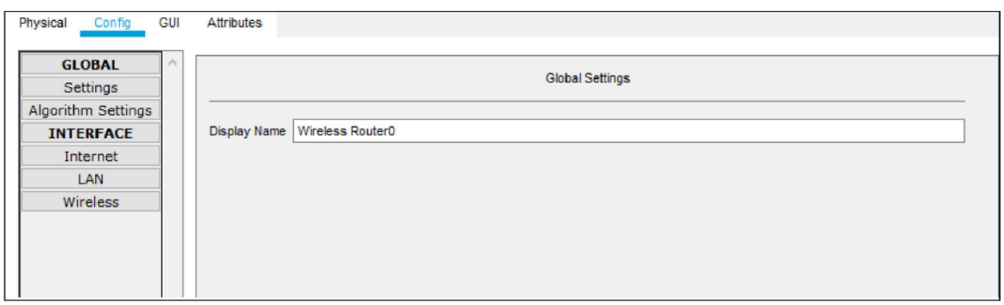
a. Add network devices to the workspace.

Using the device selection box, add the network devices to the workspace as shown in the topology diagram.

To place a device onto the workspace, first choose a device type from the **Device-Type Selection** box. Then, click on the desired device model from the **Device-Specific Selection** box. Finally, click on a location in the workspace to put your device in that location. If you want to cancel your selection, click the **Cancel** icon for that device. Alternatively, you can click and drag a device from the **Device-Specific Selection** box onto the workspace.

b. Change display names of the network devices.

To change the display names of the network devices click on the device icon on the Packet Tracer **Logical** workspace, then click on the **Config** tab in the device configuration window. Type the new name of the device into the **Display Name** box as show in the figure below.




c. Add the physical cabling between devices on the workspace

Using the device selection box, add the physical cabling between devices on the workspace as shown in the topology diagram.

The PC will need a copper straight-through cable to connect to the wireless router. Select the copper straight-through cable in the device selection box and attach it to the FastEthernet0 interface of the PC and the Ethernet 1 interface of the wireless router.

The wireless router will need a copper straight-through cable to connect to the cable modem. Select the copper straight-through cable in the device-selection box and attach it to the Internet interface of the wireless router and the Port 1 interface of the cable modem.

The cable modem will need a coaxial cable to connect to the Internet cloud. Select the coaxial cable in the device-selection box and attach it to the Port 0 interface of the cable modem and the coaxial interface of the Internet cloud.

The Interne cloud will need copper straight-through cable to connect to the Cisco.com server. Select the copper straight-through cable in the device-selection box and attach it to the Ethernet interface of the Internet cloud and the FastEthernet0 interface of the Cisco.com server.

**Part 2: Configure the Network Devices**

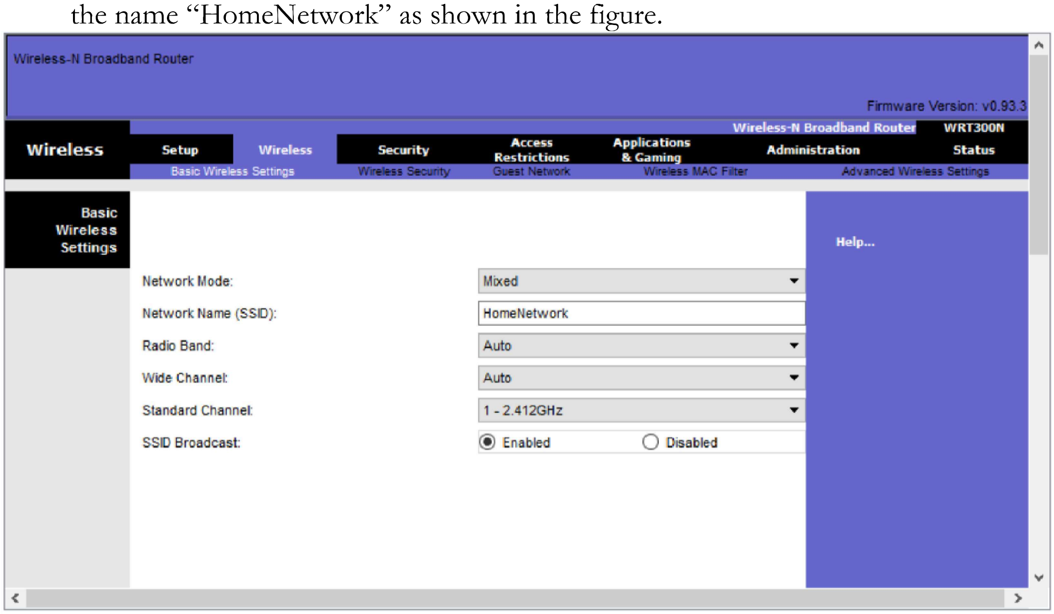
**Step 1: Configure the wireless router**

a. Create the wireless network on the wireless router

Click on the **Wireless Router** icon on the Packet Tracer **Logical** workspace to open the device configuration window.

In the wireless router configuration window, click on the **GUI** tab to view configuration options for the wireless router.

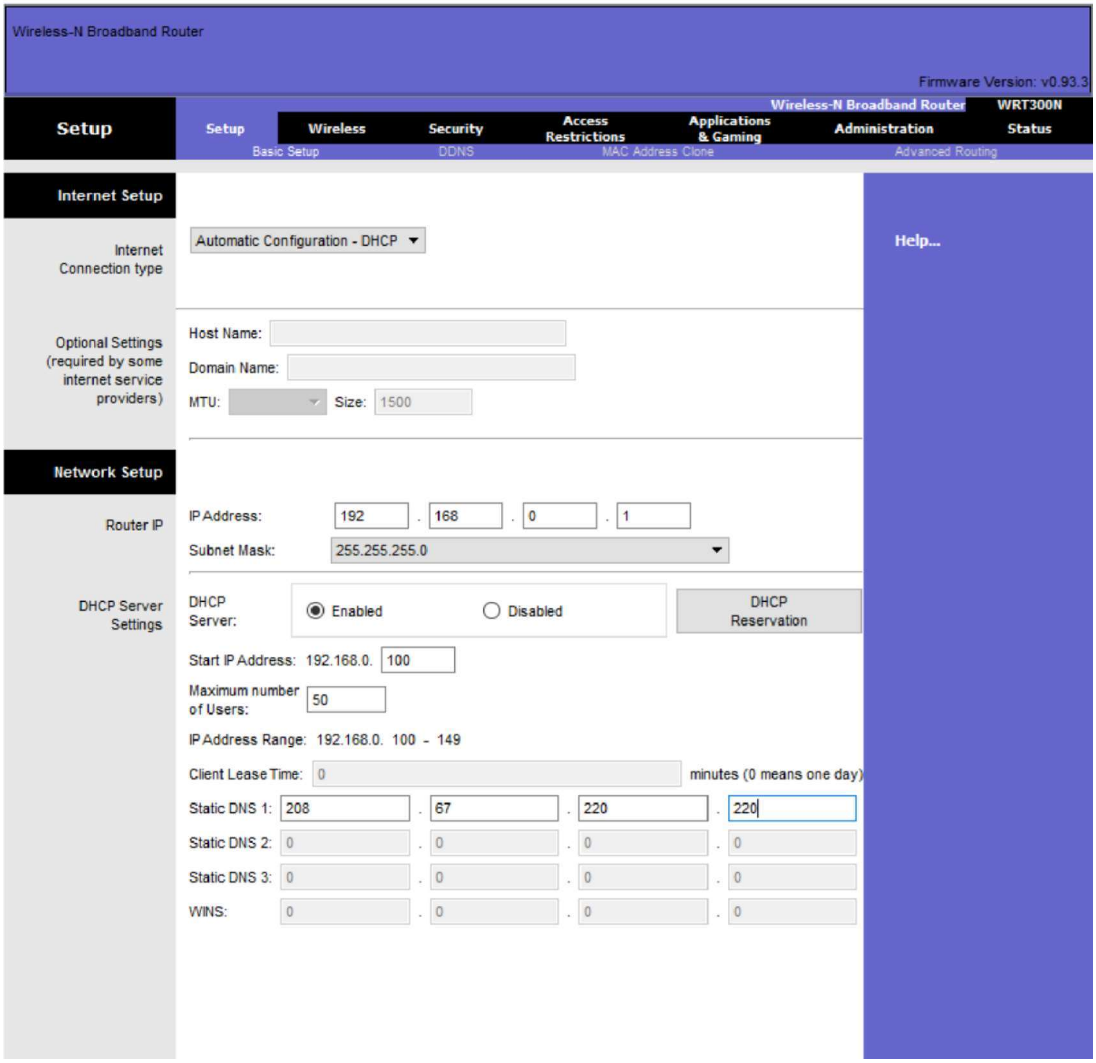
Next, click on the **Wireless** tab in the GUI to view the wireless settings. The only setting that needs to be changed from the defaults is the **Network Name (SSID)**. Here, type the name "HomeNetwork" as shown in the figure.



Configure the Internet connection on the wireless router
Click on the **Setup** tab in the wireless router GUI.

In the **DHCP Server** settings verify that the **Enabled** button is selected and configure the static IP address of the DNS server as 208.67.220.220 as shown in the figure.

b. Click on the **Save Settings** tab.




## Step 2: Configure the laptop

a. Configure the Laptop to access the wireless network

Click on the Laptop icon on the Packet Tracer **Logical** workspace and in the laptop configuration windows select the **Physical** tab.

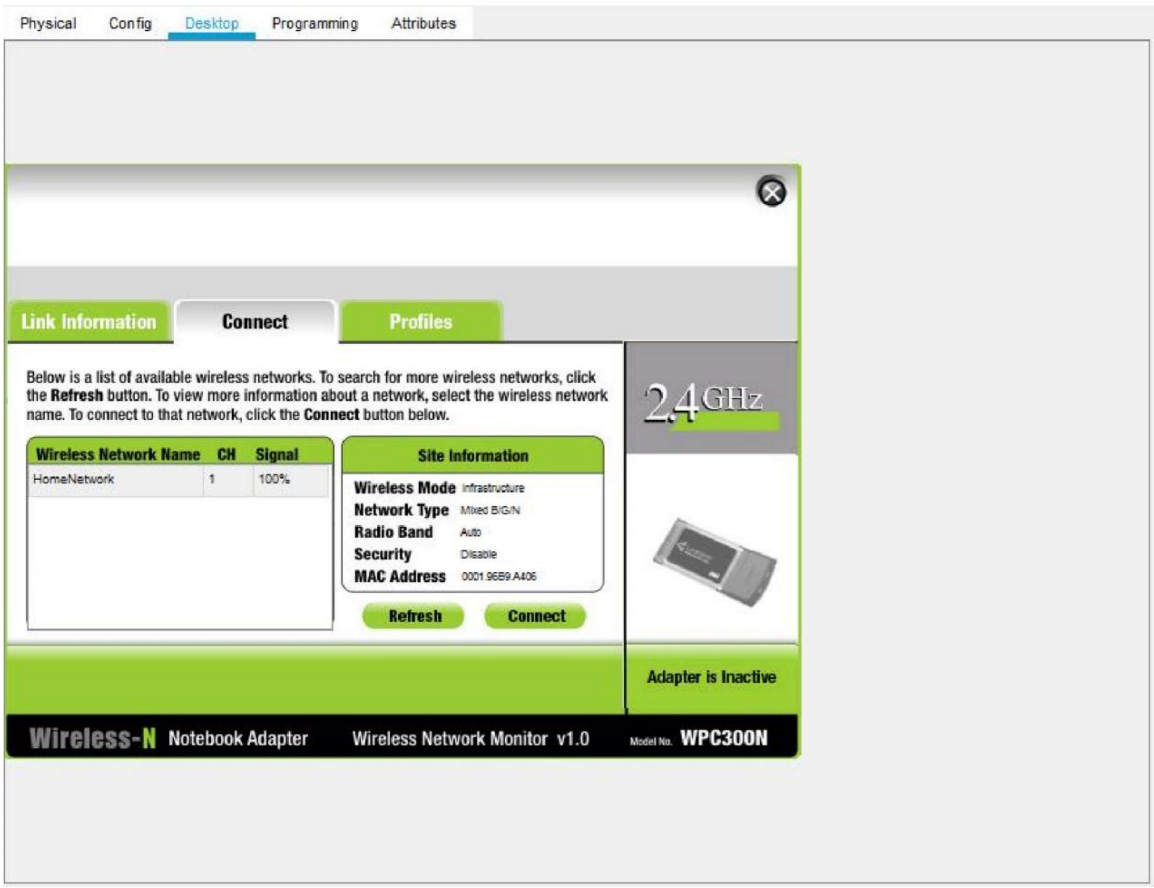
In the **Physical** tab you will need to remove the Ethernet copper module and replace it with the Wireless WPC300N module.

To do this, you first power the Laptop off by clicking the power button on the side of the laptop. Then remove the currently installed Ethernet copper module by clicking on the module on the side of the laptop and dragging it to the **MODULES** pane on the left of the laptop window. Then install the Wireless WPC300N module by clicking on it in the **MODULES** pane and dragging it to the empty module port on the side of the laptop. Power the laptop back on by clicking on the Laptop power button again.

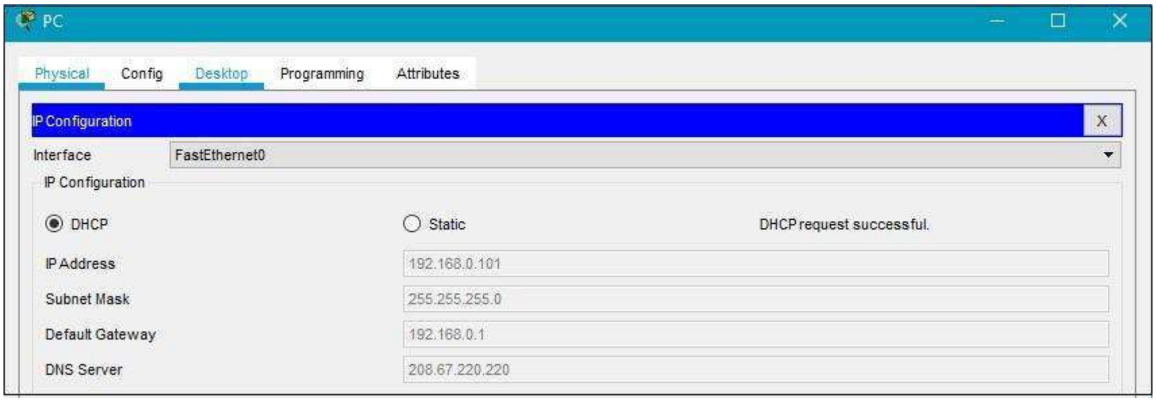
With the wireless module installed, the next task is to connect the laptop to the wireless network.

Click on the **Desktop** tab at the top of the Laptop configuration window and select the **PC Wireless** icon.

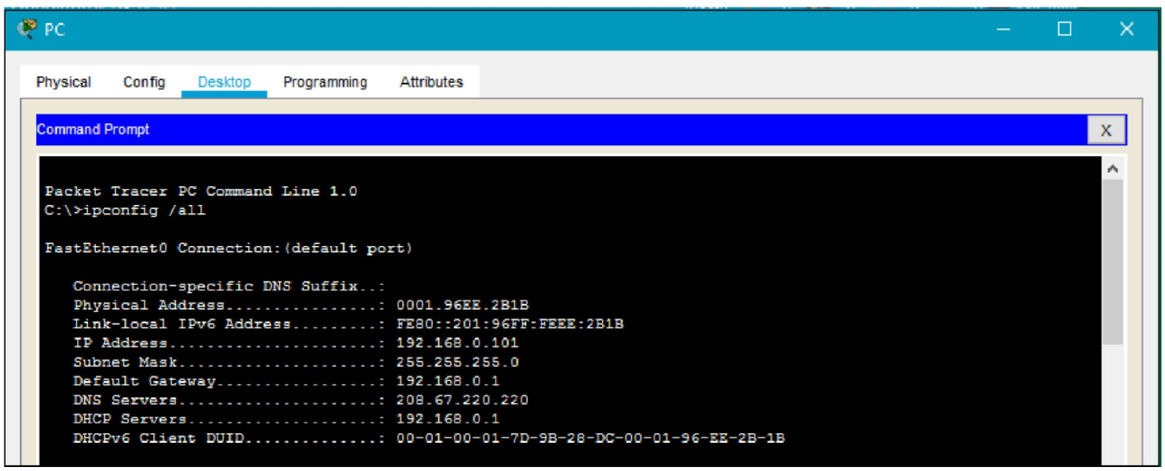
Once the Wireless-N Notebook Adapter settings are visible, select the **Connect** tab. The wireless network "HomeNetwork" should be visible in the list of wireless networks as shown in the figure.
Select the network, and click on the **Connect** tab found below the **Site Information pane.**




**Step 3: Configure the PC**
a. Configure the PC for the wired network
Click on the **PC** icon on the Packet Tracer **Logical** workspace and select the **Desktop** tab and then the **IP Configuration** icon.
In the IP Configuration window, select the **DCHP** radio button as shown in the figure so that the PC will use DCHP to receive an IPv4 address from the wireless router. Close the IP Configuration window.



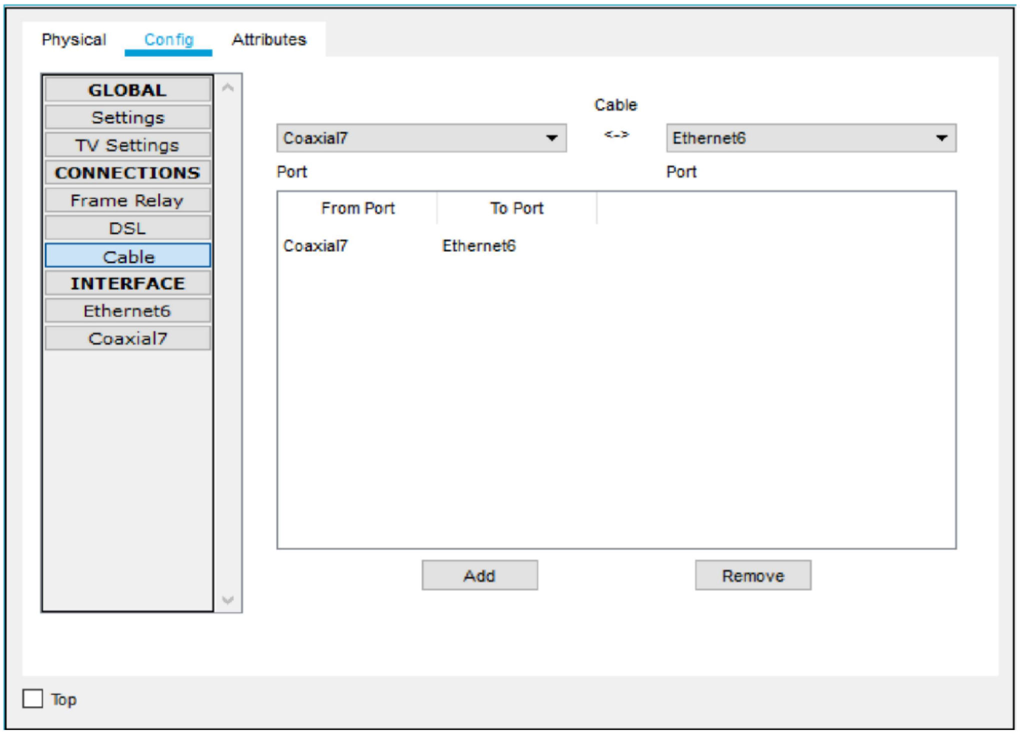

Click on the Command Prompt icon. Verify that the PC has received an IPv4 address by issuing the **ipconfig /all** command from the command prompt as shown in the figure. The PC should receive an IPv4 address in the 192.168.0.x range.



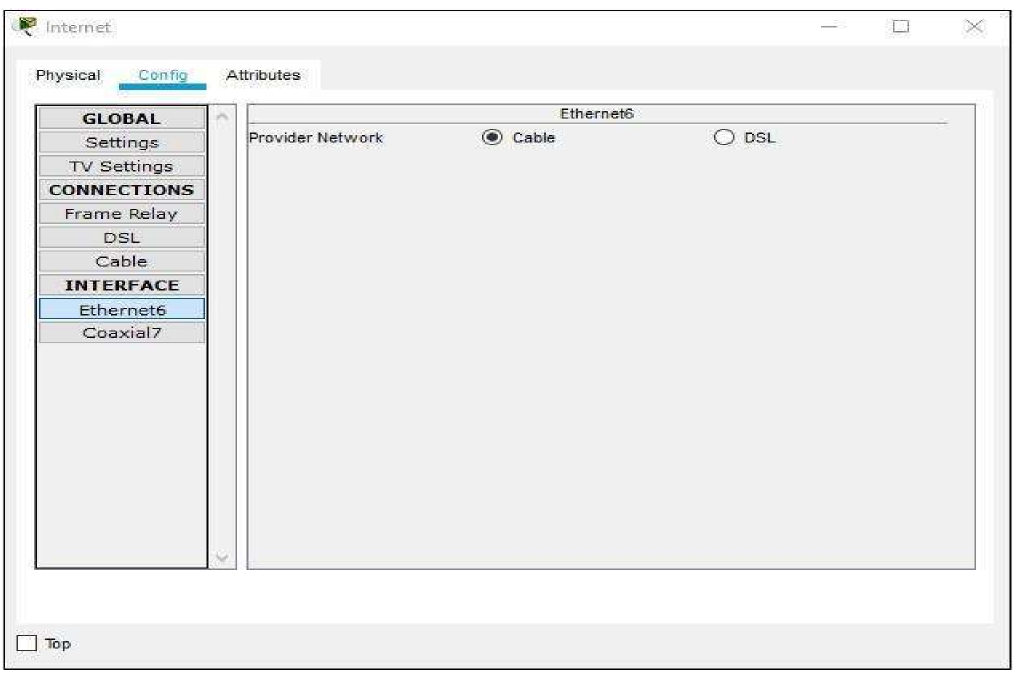

## Step 4: Configure the Internet cloud

a. Install network modules if necessary
   Click on the **Internet Cloud** icon on the Packet Tracer **Logical** workspace and then click on the **Physical** tab. The cloud device will need two modules if they are not already installed. The PT-CLOUD-NM-1CX which is for the cable modem service connection and the PT-CLOUD-NM-1CFE which is for a copper Ethernet cable connection. If these modules are missing, power off the physical cloud devices by clicking on the power button and drag each module to an empty module port on the device and then power the device back on.

b. Identify the From and To Ports
   Click on the **Config** tab in the Cloud device window. In the left pane click on **Cable** under **CONNECTIONS**. In the first drop down box choose Coaxial and in the second drop down box choose
   Ethernet then click the **Add** button to add these as the **From Port** and **To Port** as shown in the figure.




c. Identify the type of provider
While still in the **Config** tab click Ethernet under **INTERFACE** in the left pane. In the Ethernet configuration window select **Cable** as the Provider Network as shown in the figure.



**Step 5: Configure the Cisco.com server**

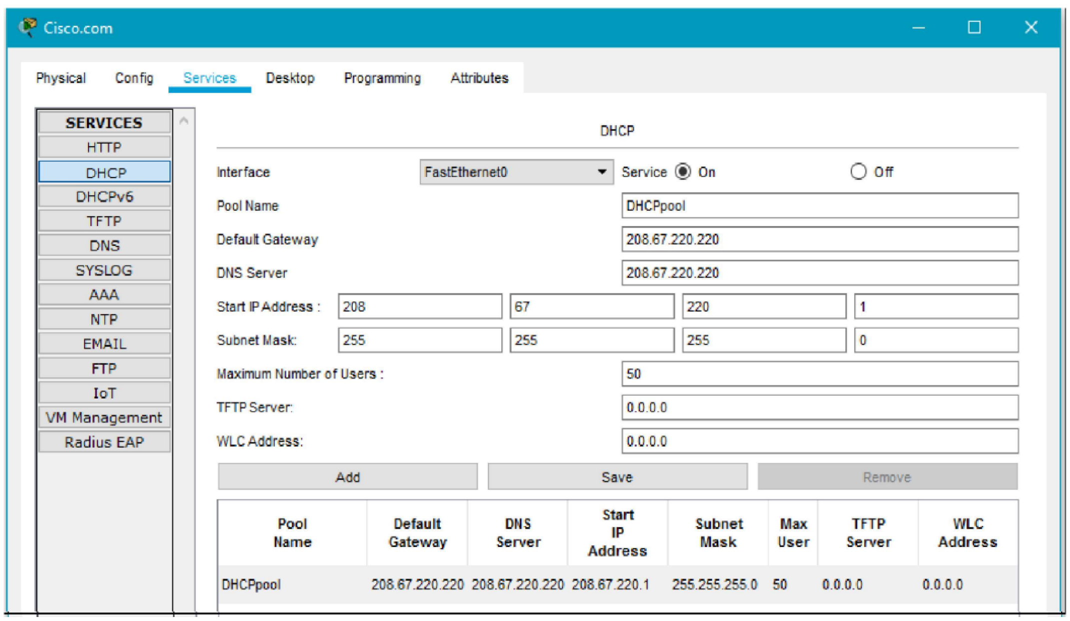
a. Configure the Cisco.com server as a DHCP server

Click on the Cisco.com server icon on the Packet Tracer **Logical** workspace and select the **Services** tab. Select **DHCP** from the **SERVICES** list in the left pane.

In the DHCP configuration window, configure a DHCP as shown in the figure with the following settings.

- Click **On** to turn the DCHP service on
- Pool name: DHCPpool
- Default Gateway: 208.67.220.220
- DNS Server: 208.67.220.220
- Starting IP Address: 208.67.220.1
- Subnet Mask 255.255.255.0
- Maximum number of Users: 50

Click **Add** to add the pool



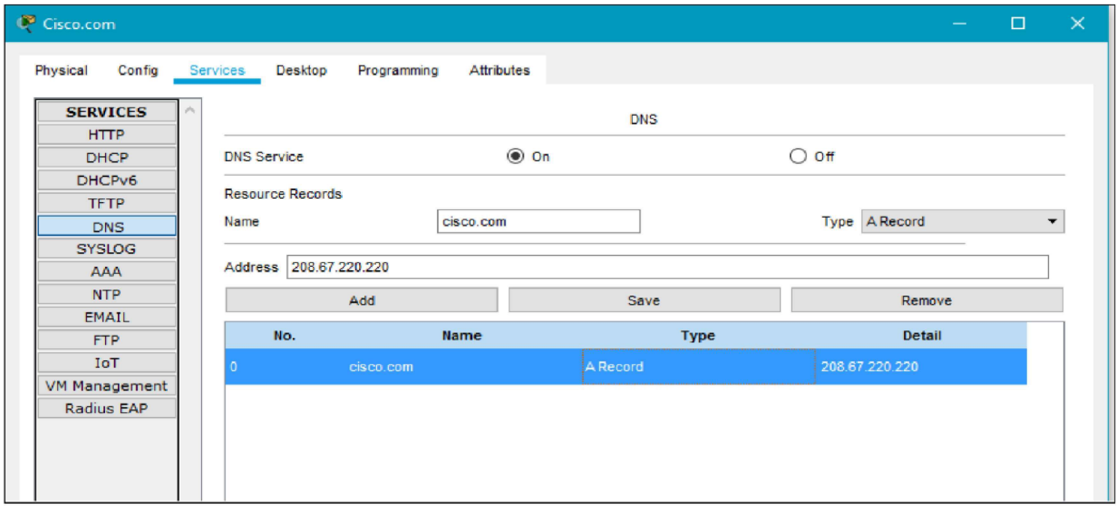

b. Configure the Cisco.com server as a DNS server to provide domain name to IPv4 address resolution.

While still in the **Services** tab, select **DNS** from the **SERVICES** listed in the left pane.

Configure the DNS service using the following settings as shown in the figure.

- Click **On** to turn the DNS service on
- Name: Cisco.com
- Type: A Record
- Address: 208.67.220.220

Click **Add** to add the DNS service settings



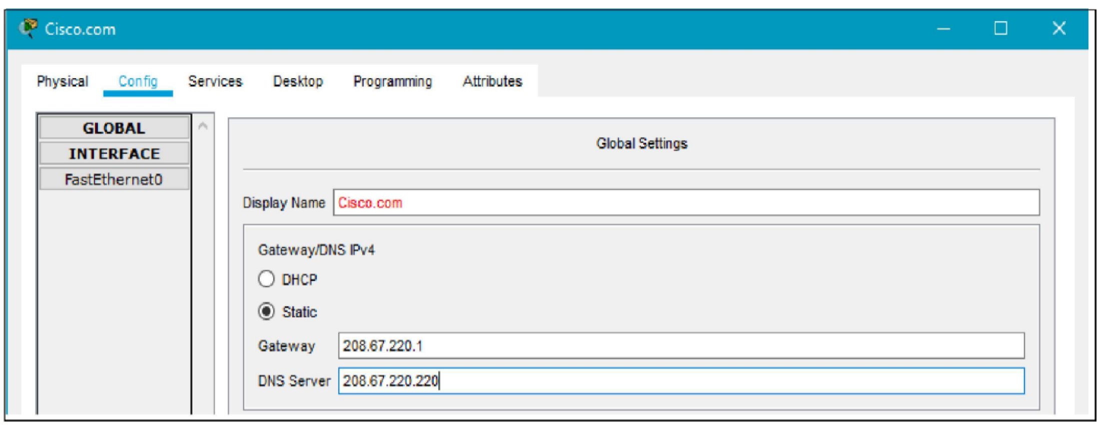

c. Configure the Cisco.com server Global settings.

Select the **Config** tab.

Click on **Settings** in left pane.

Configure the Global settings of the server as follows:

- Select **Static**
- Gateway: 208.67.220.1
- DNS Server: 208.67.220.220

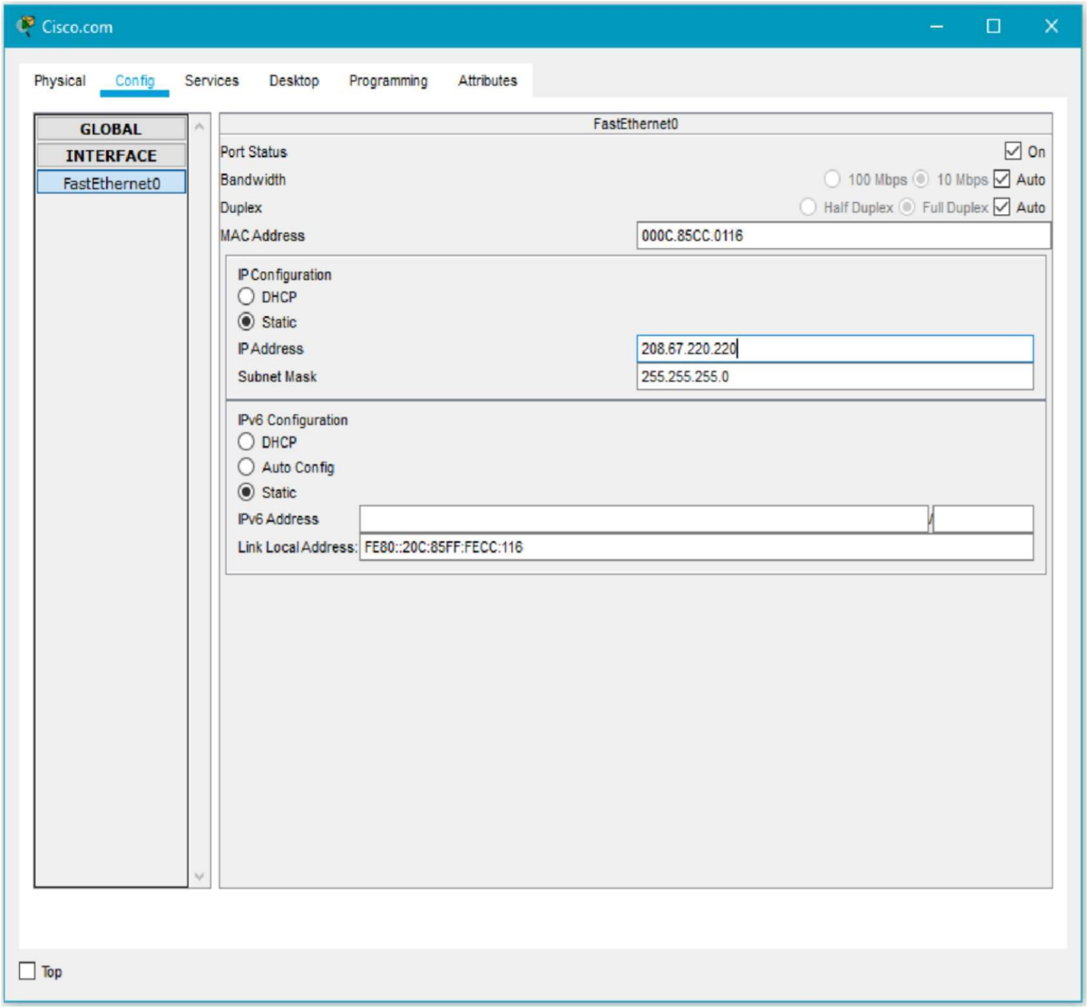



d. Configure the Cisco.com server FastEthernet0 Interface settings.

Click on **Fast Ethernet** in left pane of the **Config** tab

Configure the Fast Ethernet Interface settings of the server as follows:

- Select **Static** under IP Configuration
- IP Address: 208.67.220.220
- Subnet Mask: 255.255.255.0




**Part 3: Verify Connectivity**

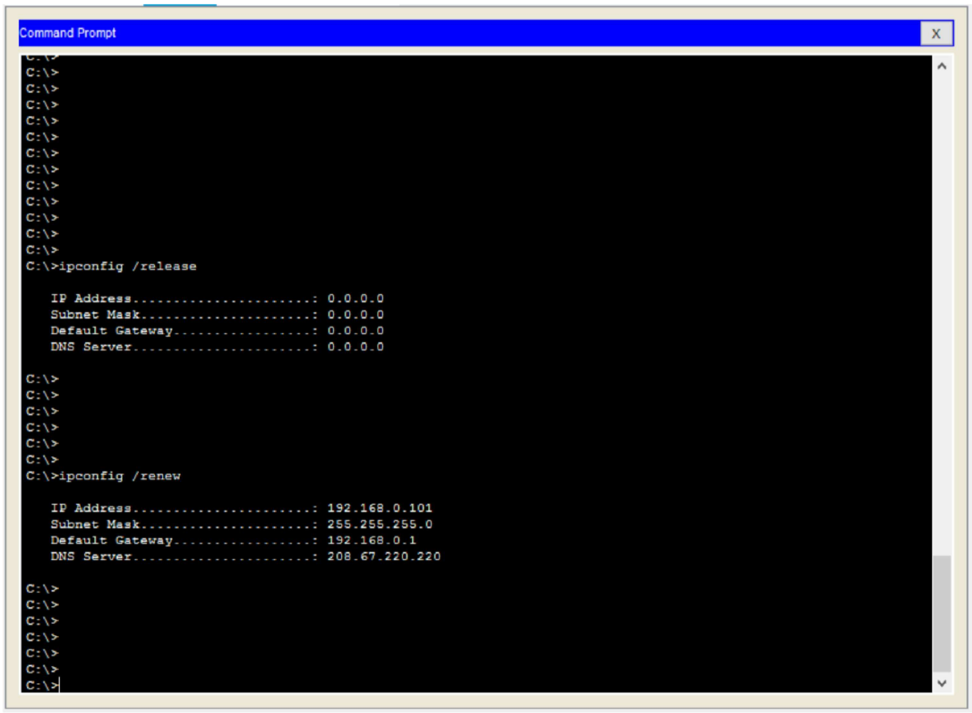
**Step 1: Refresh the IPv4 settings on the PC**

a) Verify that the PC is receiving IPv4 configuration information from DHCP.

Click on the **PC** on the Packet Tracer **Logical** workspace and then the select the **Desktop** tab of the PC configuration window.

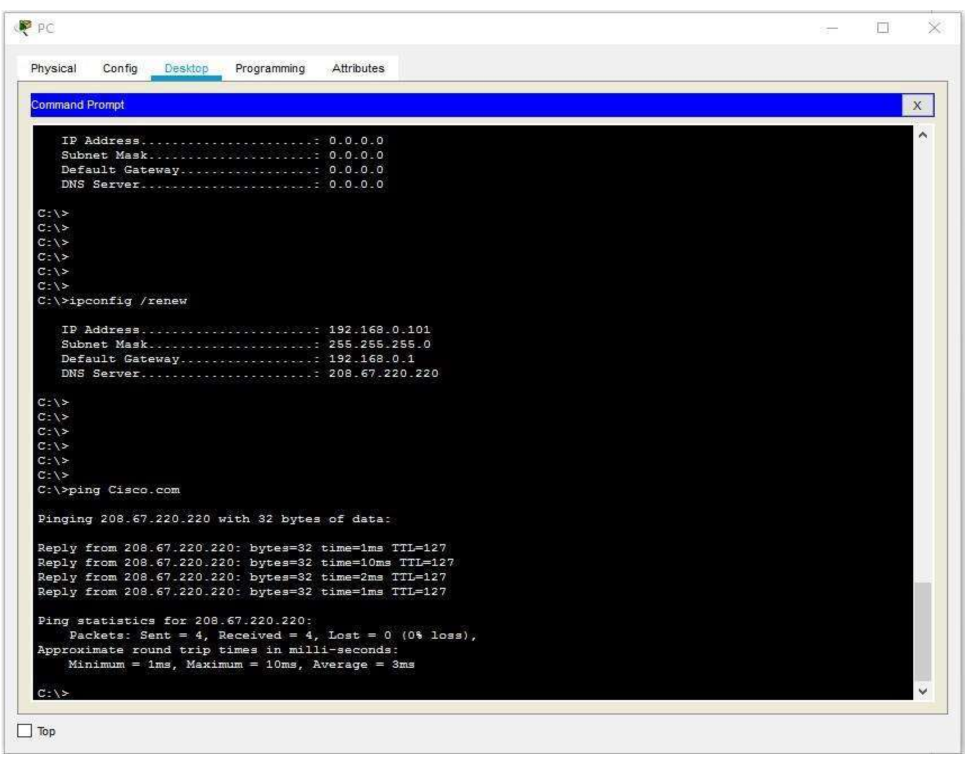
Click on the **Command Prompt** icon

In the command prompt refresh the IP settings by issuing the commands **ipconfig /release** and then **ipconfig /renew.** The output should show that the PC has an IP address in the 192.168.0.x range, a subnet mask, a default gateway, and DNS server address as shown in the figure.




b) Test connectivity to the Cisco.com server from the PC

From the command prompt, issue the command **ping Cisco.com.** It may take a few seconds for the ping to return. Four replies should be received as shown in the figure.



## Practical 11

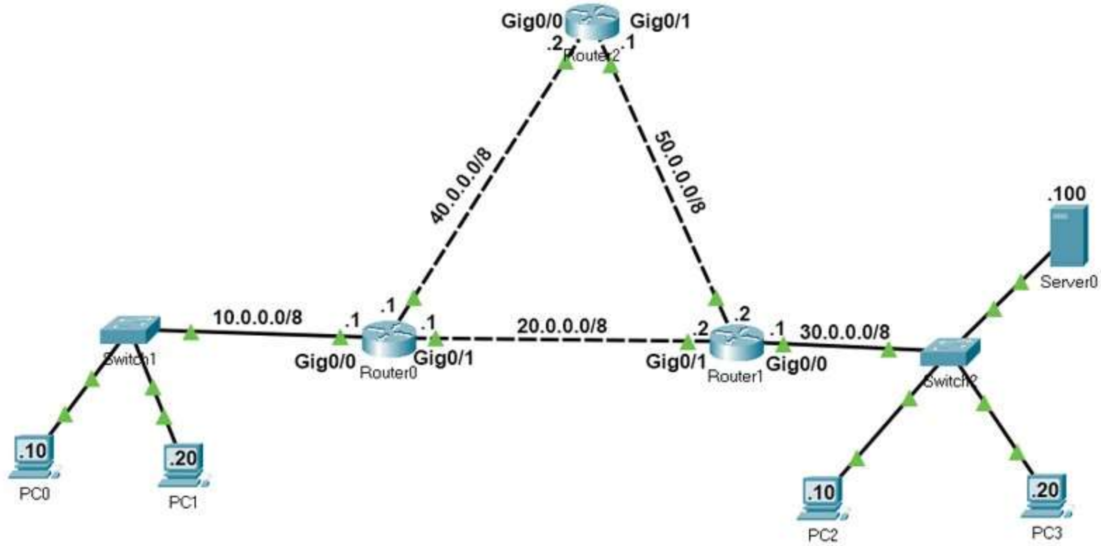
### AIM:- a)Simulate Static Routing Configuration using CISCO Packet Tracer

Static routes are the routes you manually add to the router's routing table. The process of adding static routes to the routing table is known as static routing. Let's take a packet tracer example to understand how to use static routing to create and add a static route to the routing table.

**Setting up a practice lab**

Create a packet tracer lab as shown in the following image or download the following pre-created lab and load it on Packet Tracer.

Packet Tracer Lab with Initial IP Configuration




In this lab, each network has two routes to reach. We will configure one route as the main route and another route as the backup route. If the link bandwidth of all routes is the same, we use the route that has the least number of routers as the main route. If the link bandwidth and the number of routers are the same, we can use any route as the main route and another route as the backup route.

If we specify two routes for the same destination, the router automatically selects the best route for the destination and adds the route to the routing table. If you manually want to select a route that the router should add to the routing table, you have to set the AD value of the route lower than other routes. For example, if you use the following commands to create two static routes for network 30.0.0/8, the route will place the first route to the routing table.

```
#ip route 30.0.0.0 255.0.0.0 20.0.0.2 10
#ip route 30.0.0.0 255.0.0.0 40.0.0.2 20
```

If the first route fails, the router automatically adds the second route to the routing table.

**Creating, adding, verifying static routes**

Routers automatically learn their connected networks. We only need to add routes for the networks that are not available on the router's interfaces. For example, network 10.0.0.0/8, 20.0.0.0/8 and 40.0.0.0/8 are directly connected to Router0. Thus, we don't need to configure routes for these networks. Network 30.0.0.0/8 and network 50.0.0.0/8 are not available on Router0. We have to create and add routes only for these networks.

The following table lists the connected networks of each router.

| Router | Available networks on local interfaces | Networks available on other routers' interfaces |
|---|---|---|
| Router0 | 10.0.0.0/8, 20.0.0.0/8, 40.0.0.0/8 | 30.0.0.0/8, 50.0.0.0/8 |
| Router1 | 20.0.0.0/8, 30.0.0.0/8, 50.0.0.0/8 | 10.0.0.0/8, 40.0.0.0/8 |
| Router2 | 40.0.0.0/8, 50.0.0.0/8 | 10.0.0.0/8, 20.0.0.0/8, 30.0.0.0/8 |

Let's create static routes on each router for networks that are not available on the router.

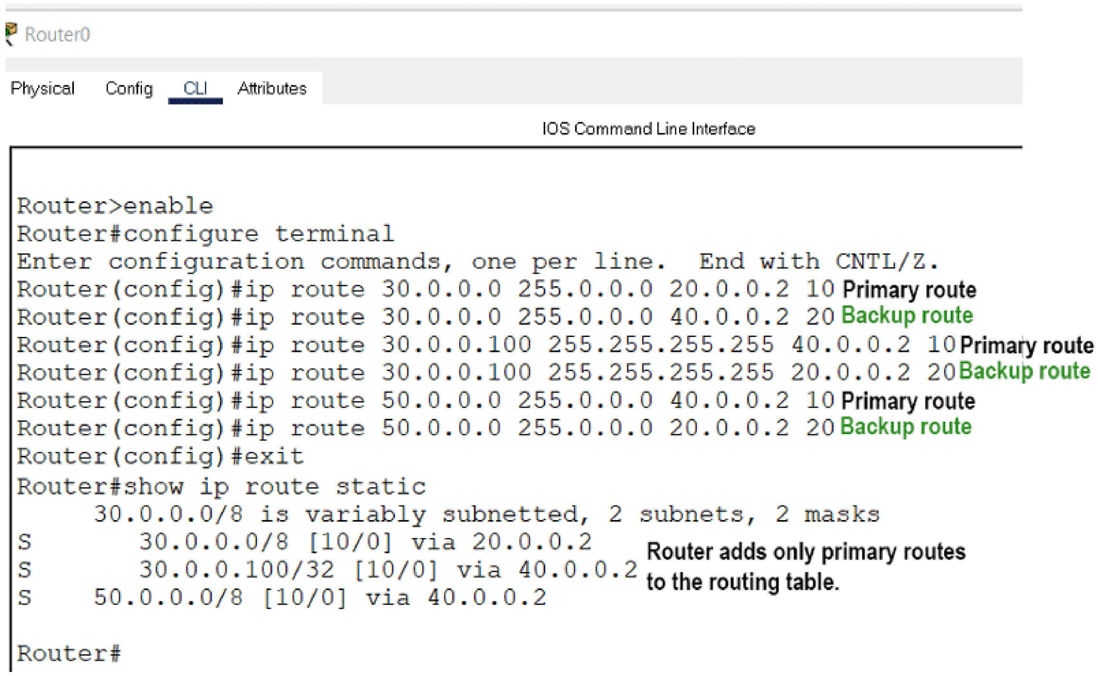
***Router0 requirements***

- Create two routes for network 30.0.0.0/8 and configure the first route (via -Router1) as the main route and the second route (via-Router2) as a backup route.
- Create two routes for the host 30.0.0.100/8 and configure the first route (via -Router2) as the main route and the second route (via-Router1) as a backup route.
- Create two routes for network 50.0.0.0/8 and configure the first route (via -Router2) as the main route and the second route (via-Router1) as a backup route.
- Verify the router adds only main routes to the routing table.

***Router0 configuration***

Access the CLI prompt of Router0 and run the following commands.

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 30.0.0.0 255.0.0.0 20.0.0.2 10
Router(config)#ip route 30.0.0.0 255.0.0.0 40.0.0.2 20
Router(config)#ip route 30.0.0.100 255.255.255.255 40.0.0.2 10
Router(config)#ip route 30.0.0.100 255.255.255.255 20.0.0.2 20
Router(config)#ip route 50.0.0.0 255.0.0.0 40.0.0.2 10
Router(config)#ip route 50.0.0.0 255.0.0.0 20.0.0.2 20
Router(config)#exit
Router#show ip route static
30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S 30.0.0.0/8 [10/0] via 20.0.0.2
S 30.0.0.100/32 [10/0] via 40.0.0.2
S 50.0.0.0/8 [10/0] via 40.0.0.2
Router#
```

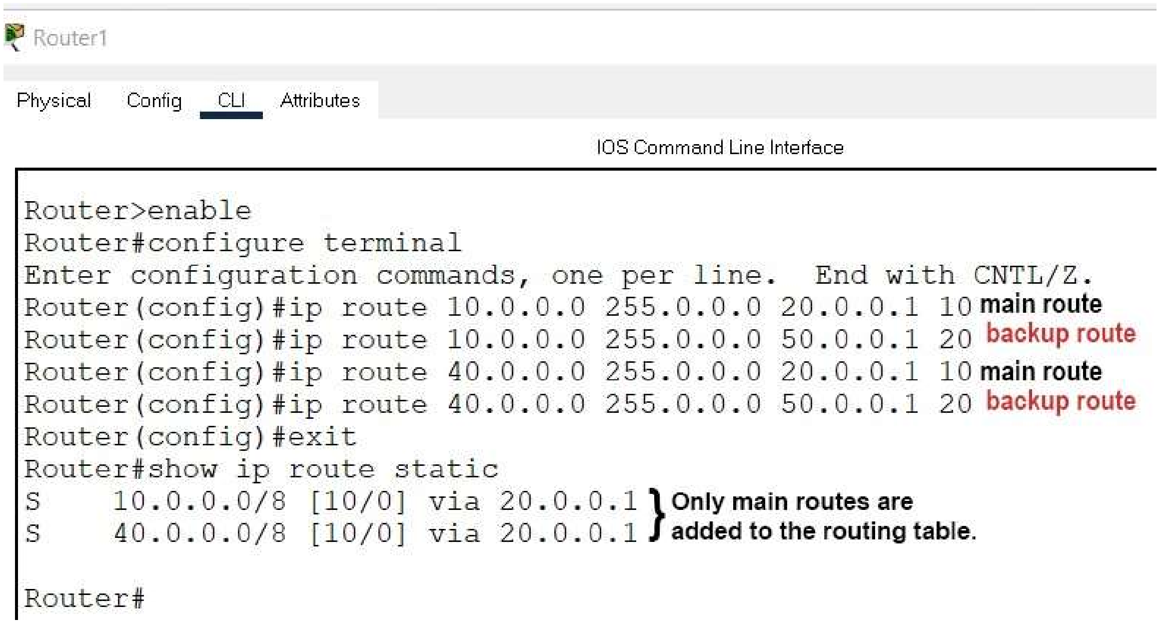



***Router1 requirements***

- Create two routes for network 10.0.0.0/8 and configure the first route (via -Router0) as the main route and the second route (via-Router1) as a backup route.
- Create two routes for network 40.0.0.0/8 and configure the first route (via -Router0) as the main route and the second route (via-Router2) as a backup route.
- Verify the router adds only main routes to the routing table.

***Router1 configuration***

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 10.0.0.0 255.0.0.0 20.0.0.1 10
Router(config)#ip route 10.0.0.0 255.0.0.0 50.0.0.1 20
Router(config)#ip route 40.0.0.0 255.0.0.0 20.0.0.1 10
Router(config)#ip route 40.0.0.0 255.0.0.0 50.0.0.1 20
Router(config)#exit
Router#show ip route static
S 10.0.0.0/8 [10/0] via 20.0.0.1
S 40.0.0.0/8 [10/0] via 20.0.0.1
Router#
```

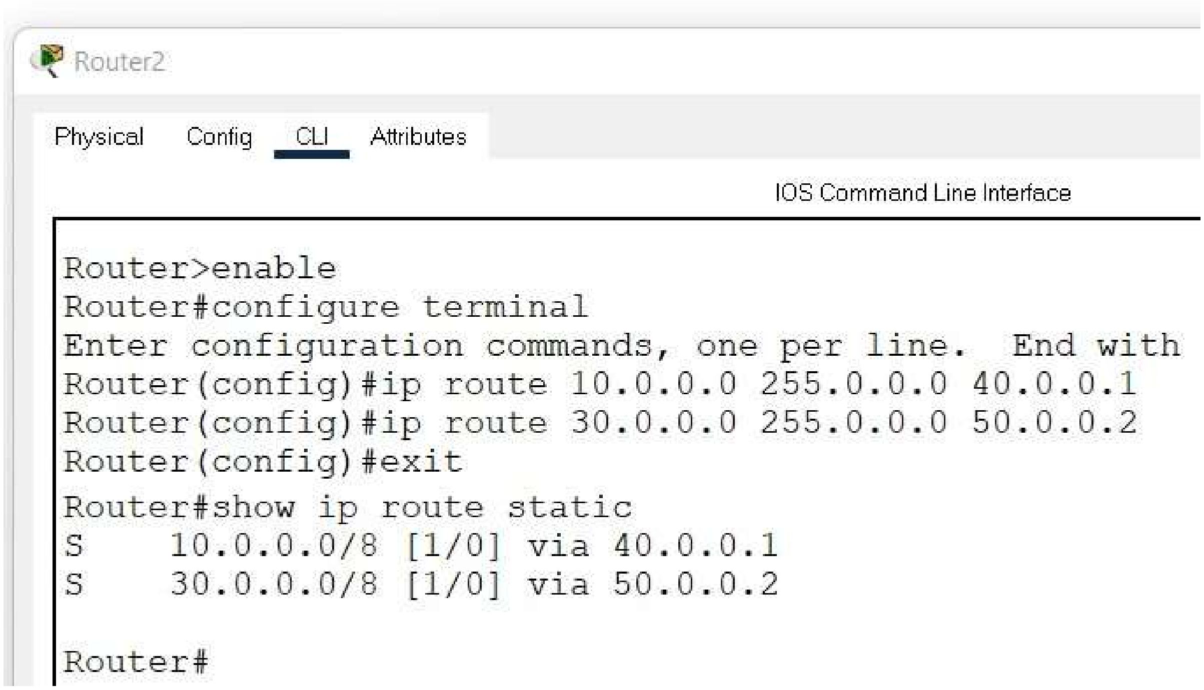



***Router2 requirements***

Create static routes for network 10.0.0.0/8 and network 30.0.0.0/8 and verify the router adds both routes to the routing table.

***Router2 configuration***

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 10.0.0.0 255.0.0.0 40.0.0.1
Router(config)#ip route 30.0.0.0 255.0.0.0 50.0.0.2
Router(config)#exit
Router#show ip route static
S 10.0.0.0/8 [1/0] via 40.0.0.1
S 30.0.0.0/8 [1/0] via 50.0.0.2
Router#
```



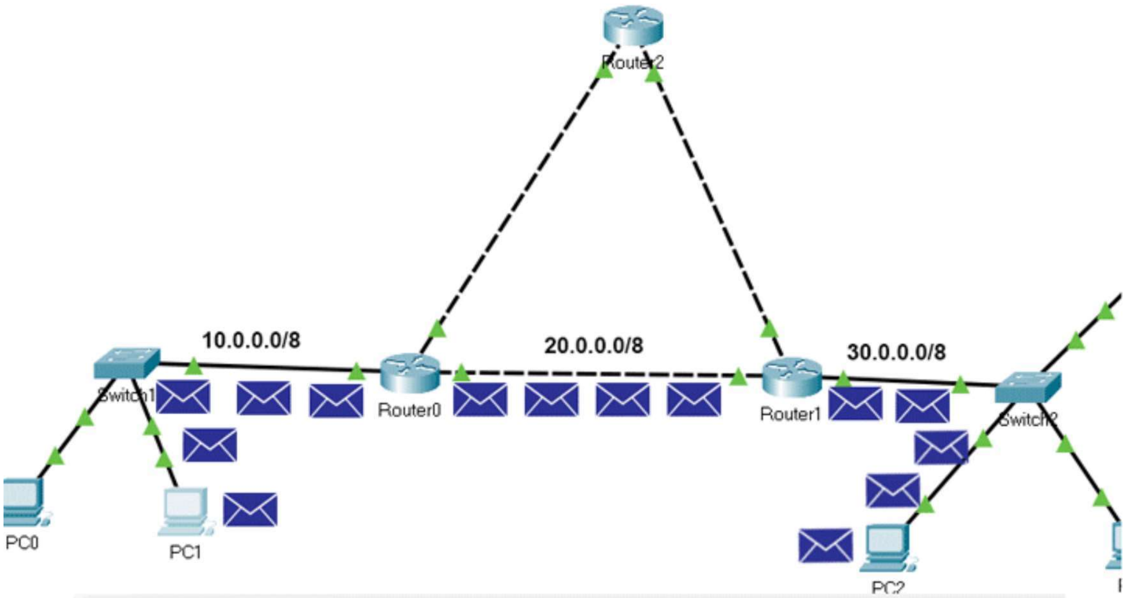

**Verifying static routing**

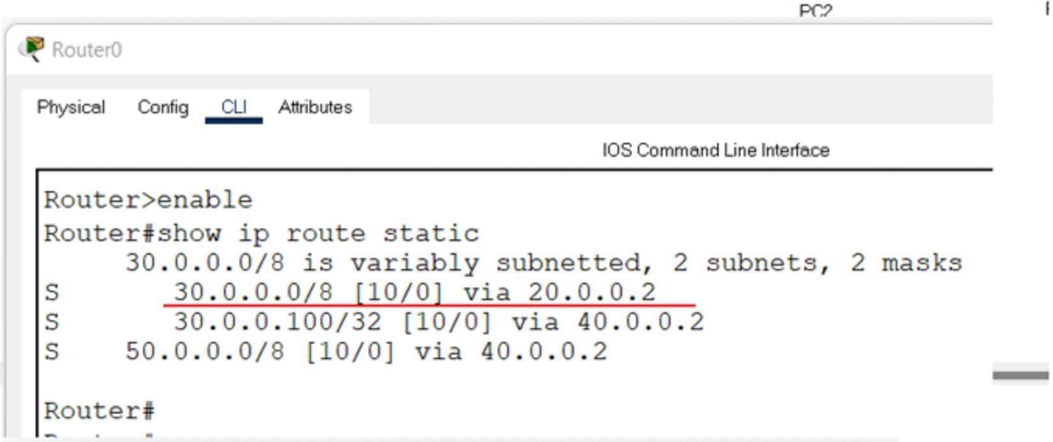
On Router0, we configured two routes for network 30.0.0.0/8. These routes are via Router1 and via Router2. We set the first route (via-Router1) as the main route and the second route as the backup route. We can verify this configuration in two ways.

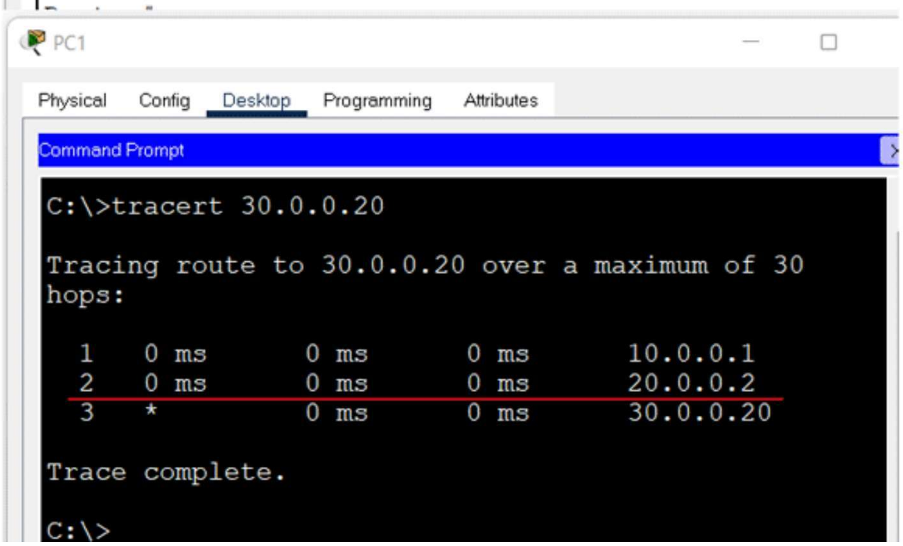
By sending ping requests to a PC of network 30.0.0.0/8 and tracing the path they take to reach the network 30.0.0.0/8. For this, you can use **'tracert'** command on a PC of network 10.0.0.0/8. The **'tracert'** command sends ping requests to the destination host and tracks the path they take to reach the destination.

By listing the routing table entries on Router0. Since a router uses the routing table to forward data packets, you can check the routing table to figure out the route the router uses to forward data packets for each destination.

The following image shows the above testing.

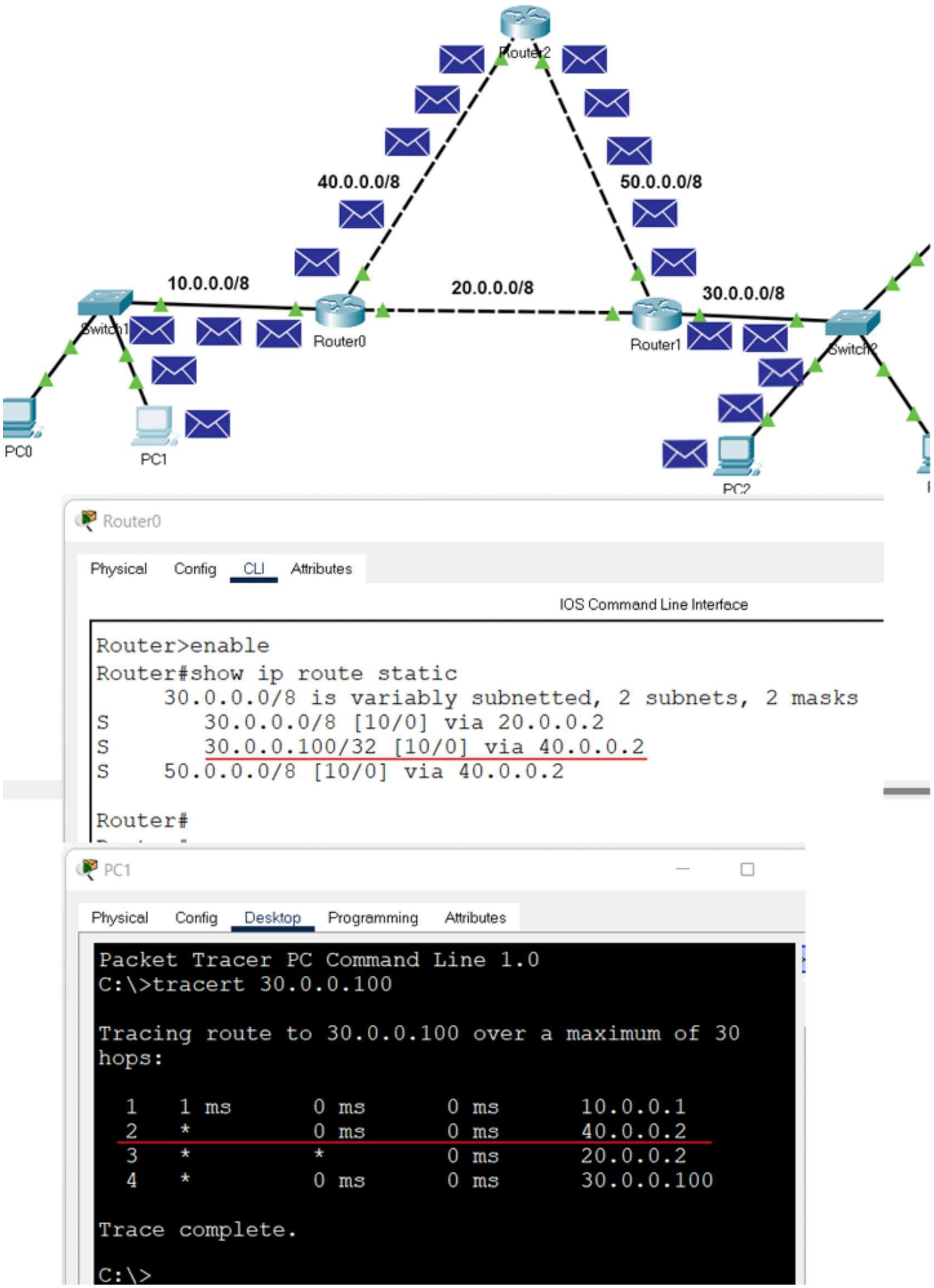


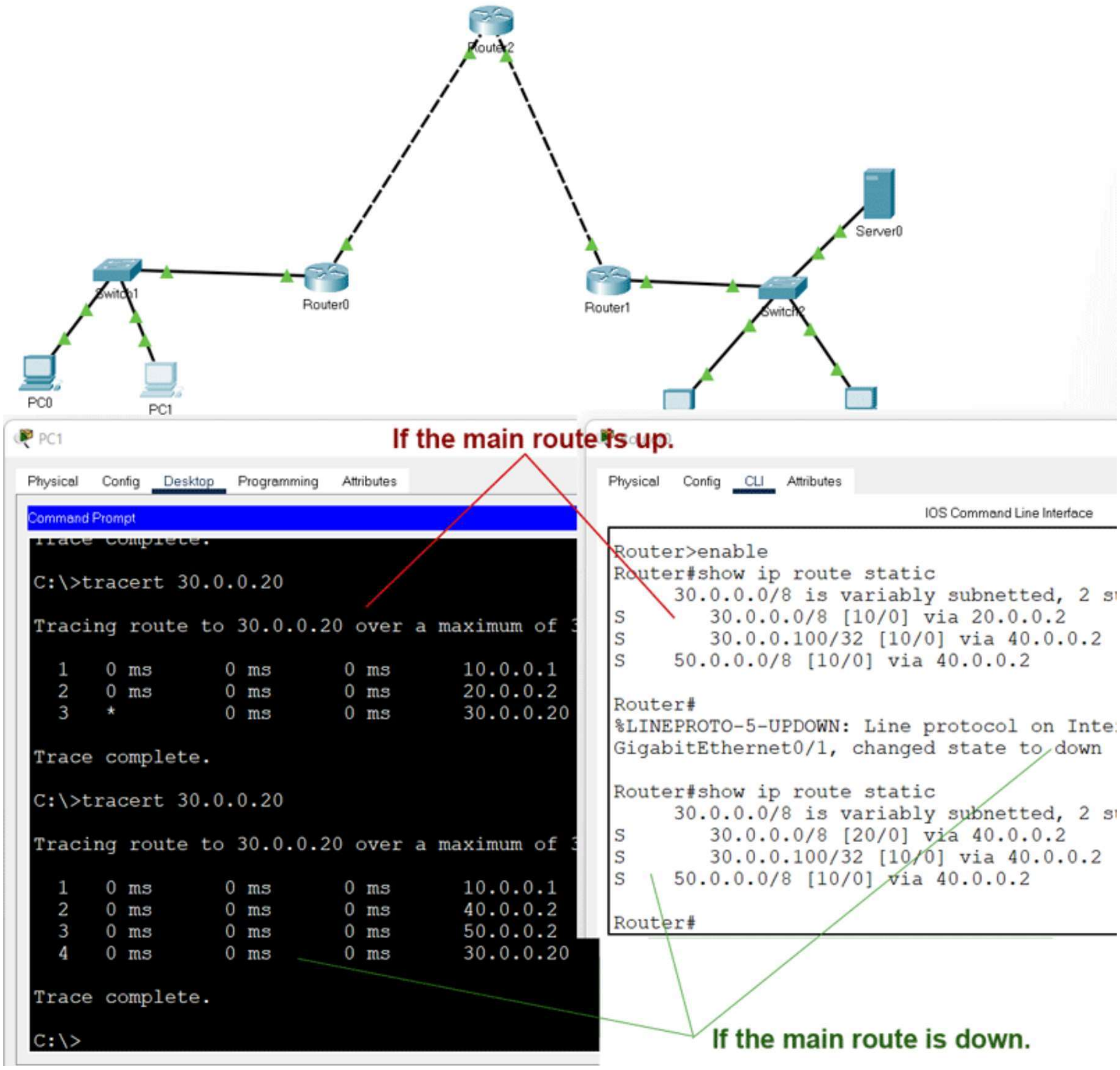
We also configured a separate static host route for the host 30.0.0.100/8. The router must use this route to forward data packets to the host 30.0.0.100/8. To verify this, you can do the same testing for the host 30.0.0.100/8.

The following image shows this testing.



We also configured a backup route for network 30.0.0.0/8. The router must put the backup route to the routing table and use it to forward data packets to network 30.0.0.0/8 when the main route fails. To verify this, we have to simulate the failure of the main route.

To simulate the failure of the main route, you can delete the link between Router0 and Router1. After deleting the link, do the same testing again for the network 30.0.0.0/8.




The following link provides the configured packet tracer lab of the above example.
Packet Tracer Lab with Static Routing Configuration

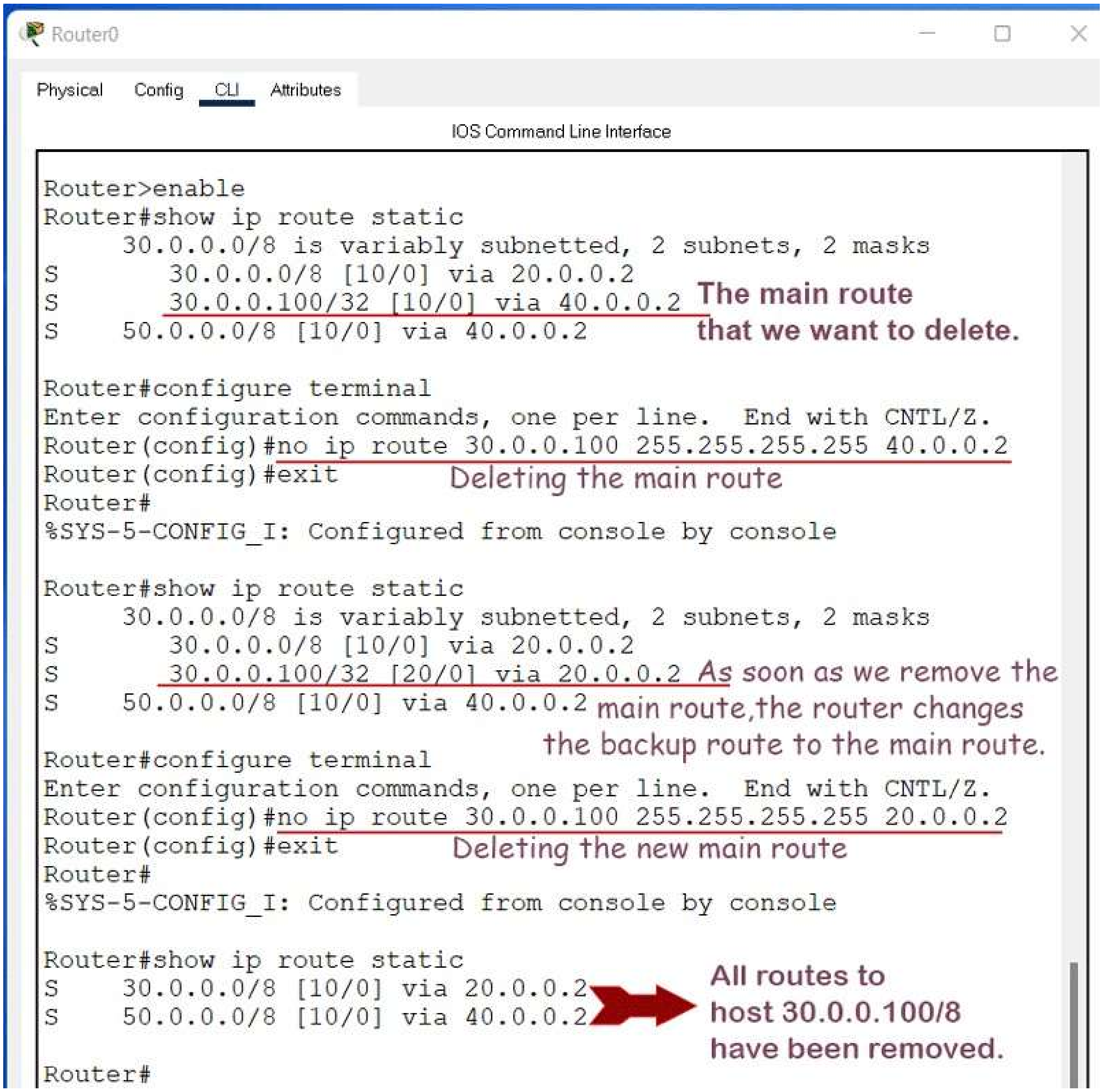
**Deleting a static route**

To delete a static route, use the following steps.

- Use the **'show ip route static'** command to print all static routes.
- Note down the route you want to delete.
- Use the **'no ip route'** command to delete the route.

If you have a backup route, the backup route becomes the main route when you delete the main route.

In our example, we have a backup route and a main route for the host 30.0.0.100/8. The following image shows how to delete both routes.


Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router>enable
Router#show ip route static
30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S 30.0.0.0/8 [10/0] via 20.0.0.2
S 30.0.0.100/32 [10/0] via 40.0.0.2
S 50.0.0.0/8 [10/0] via 40.0.0.2
The main route that we want to delete.
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#no ip route 30.0.0.100 255.255.255.255 40.0.0.2
Router(config)#exit
Deleting the main route
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#show ip route static
30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S 30.0.0.0/8 [10/0] via 20.0.0.2
S 30.0.0.100/32 [20/0] via 20.0.0.2
S 50.0.0.0/8 [10/0] via 40.0.0.2
As soon as we remove the main route,the router changes the backup route to the main route.
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#no ip route 30.0.0.100 255.255.255.255 20.0.0.2
Router(config)#exit
Deleting the new main route
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#show ip route static
S 30.0.0.0/8 [10/0] via 20.0.0.2
S 50.0.0.0/8 [10/0] via 40.0.0.2
All routes to host 30.0.0.100/8 have been removed.
Router#

## Practical 11

## AIM:- b)Simulate RIP using CISCO Packet Tracer

**Initial IP configuration**

| Device | Interface | IP Configuration | Connected with |
|---|---|---|---|
| PC0 | Fast Ethernet | 10.0.0.2/8 | Router0's Fa0/1 |
| Router0 | Fa0/1 | 10.0.0.1/8 | PC0's Fast Ethernet |
| Router0 | S0/0/1 | 192.168.1.254/30 | Router2's S0/0/1 |
| Router0 | S0/0/0 | 192.168.1.249/30 | Router1's S0/0/0 |
| Router1 | S0/0/0 | 192.168.1.250/30 | Router0's S0/0/0 |
| Router1 | S0/0/1 | 192.168.1.246/30 | Router2's S0/0/0 |
| Router2 | S0/0/0 | 192.168.1.245/30 | Router1's S0/0/1 |
| Router2 | S0/0/1 | 192.168.1.253/30 | Router0's S0/0/1 |
| Router2 | Fa0/1 | 20.0.0.1/30 | PC1's Fast Ethernet |
| PC1 | Fast Ethernet | 20.0.0.2/30 | Router2's Fa0/1 |

**Assign IP address to PCs**

Double click **PCs** and click **Desktop** menu item and click **IP Configuration**. Assign IP address referring the above table.

**Assign IP address to interfaces of routers**
Double click **Router0** and click **CLI** and press **Enter key** to access the command prompt of **Router0**.

We need to configure IP address and other parameters on interfaces before we could actually use them for routing. Interface mode is used to assign IP address and other parameters. Interface mode can be accessed from global configuration mode. Following commands are used to access the global configuration mode.

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#
```

From global configuration mode we can enter in interface mode. From there we can configure the interface. Following commands will assign IP address on FastEthernet0/0.

```
Router(config)#interface fastEthernet 0/0
Router(config-if)#ip address 10.0.0.1 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#
```

**interface *fastEthernet 0/0*** command is used to enter in interface mode.
**ip address *10.0.0.1 255.0.0.0*** command will assign IP address to interface.

**no shutdown** command will bring the interface up.
**exit** command is used to return in global configuration mode.
Serial interface needs two additional parameters **clock rate** and **bandwidth**. Every serial cable has two ends DTE and DCE. These parameters are always configured at DCE end.
We can use **show controllers interface** command from privilege mode to check the cable's end.

```
Router#show controllers serial 0/0/0
Interface Serial0/0/0
Hardware is PowerQUICC MPC860
DCE V.35, clock rate 2000000
[Output omitted]
```

Fourth line of output confirms that DCE end of serial cable is attached. If you see DTE here instead of DCE skip these parameters.
Now we have necessary information let's assign IP address to serial interface.

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface serial 0/0/0
Router(config-if)#ip address 192.168.1.249 255.255.255.252
Router(config-if)#clock rate 64000
Router(config-if)#bandwidth 64
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface serial 0/0/1
Router(config-if)#ip address 192.168.1.254 255.255.255.252
Router(config-if)#clock rate 64000
Router(config-if)#bandwidth 64
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#
```

**Router#configure terminal** Command is used to enter in global configuration mode.
**Router(config)#interface *serial 0/0/0*** Command is used to enter in interface mode.
**Router(config-if)#ip address *192.168.1.249 255.255.255.252*** Command assigns IP address to interface. For serial link we usually use IP address from /30 subnet.
**Router(config-if)#clock rate *64000*** And **Router(config-if)#bandwidth *64*** In real life environment these parameters control the data flow between serial links and need to be set at service providers end. In lab environment we need not to worry about these values. We can use these values.
**Router(config-if)#no shutdown** Command brings interface up.
**Router(config-if)#exit** Command is used to return in global configuration mode.
We will use same commands to assign IP addresses on interfaces of remaining routers. We need to provided clock rate and bandwidth only on DCE side of serial interface. Following command will assign IP addresses on interface of Router1.

**Router1**

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface serial 0/0/0
Router(config-if)#ip address 192.168.1.250 255.255.255.252
```

```
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface serial 0/0/1
Router(config-if)#ip address 192.168.1.246 255.255.255.252
Router(config-if)#clock rate 64000
Router(config-if)#bandwidth 64
Router(config-if)#no shutdown
Router(config-if)#exit
```

Use same commands to assign IP addresses on interfaces of Router2.

**Router2**

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface fastEthernet 0/0
Router(config-if)#ip address 20.0.0.1 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface serial 0/0/0
Router(config-if)#ip address 192.168.1.245 255.255.255.252
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface serial 0/0/1
Router(config-if)#ip address 192.168.1.253 255.255.255.252
Router(config-if)#no shutdown
Router(config-if)#exit
```

Now routers have information about the networks that they have on their own interfaces. Routers will not exchange this information between them on their own. We need to implement RIP routing protocol that will insist them to share this information.

**Configure RIP routing protocol**

Configuration of RIP protocol is much easier than you think. It requires only two steps to configure the RIP routing.

- Enable RIP routing protocol from global configuration mode.
- Tell RIP routing protocol which networks you want to advertise.

Let’s configure it in Router0

**Router0**

```
Router0(config)#router rip
Router0(config-router)# network 10.0.0.0
Router0(config-router)# network 192.168.1.252
Router0(config-router)# network 192.168.1.248
```

**router rip** command tell router to enable the RIP routing protocol.

**network** command allows us to specify the networks which we want to advertise. We only need to specify the networks which are directly connected with the router.

That’s all we need to configure the RIP. Follow same steps on remaining routers.

**Router1**

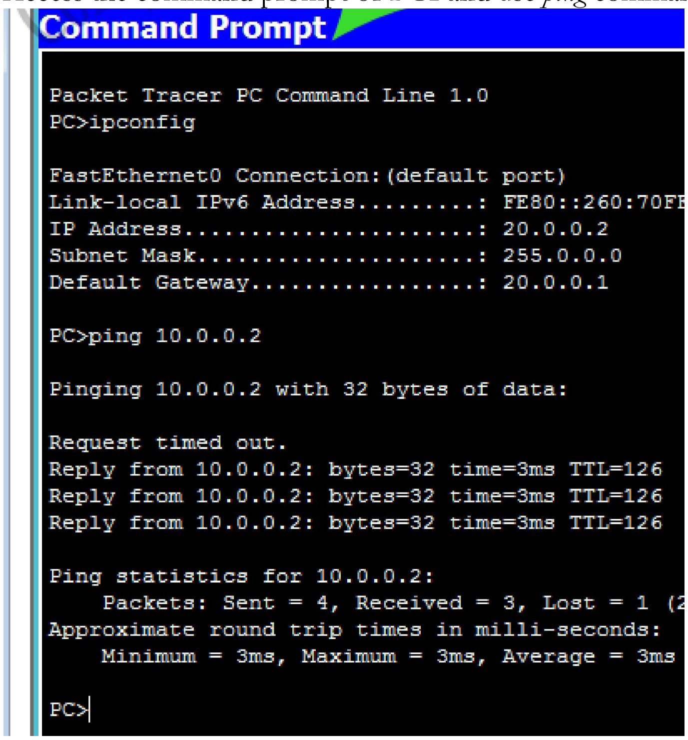
```
Router1(config)#router rip
Router1(config-router)# network 192.168.1.244
Router1(config-router)# network 192.168.1.248
```

**Router2**

```
Router2(config)#router rip
```

```
Router2(config-router)# network 20.0.0.0
Router2(config-router)# network 192.168.1.252
Router2(config-router)# network 192.168.1.244
```

That's it. Our network is ready to take the advantage of RIP routing. To verify the setup we will use ping command. ping command is used to test the connectivity between two devices.
Access the command prompt of **PC1** and use *ping* command to test the connectivity from **PC0**.



```
Command Prompt

Packet Tracer PC Command Line 1.0
PC>ipconfig

FastEthernet0 Connection:(default port)
Link-local IPv6 Address.........: FE80::260:70FF
IP Address......................: 20.0.0.2
Subnet Mask.....................: 255.0.0.0
Default Gateway.................: 20.0.0.1

PC>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Request timed out.
Reply from 10.0.0.2: bytes=32 time=3ms TTL=126
Reply from 10.0.0.2: bytes=32 time=3ms TTL=126
Reply from 10.0.0.2: bytes=32 time=3ms TTL=126

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (2
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 3ms, Average = 3ms

PC>
```

RIP protocol automatically manage all routes for us. If one route goes down, it automatically switches to another available. To explain this process more clearly we have added one more route in our network.
Currently there are two routes between PC0 and PC1.

**Route 1**

PC0 [Source / destination – 10.0.0.2] <==> Router0 [FastEthernet0/1 – 10.0.0.1] <==> Router0 [Serial0/0/1 – 192.168.1.254] <==> Router2 [Serial 0/0/1 – 192.168.1.253] <==> Router2 [FastEthernet0/0 – 20.0.0.1] <==> PC1 [Destination /source – 20.0.0.2]

**Route 2**

PC0 [Source / destination – 10.0.0.2] <==> Router0 [FastEthernet0/1 – 10.0.0.1] <==> Router0 [Serial0/0/0 – 192.168.1.249] <==> Router1 [Serial 0/0/0 – 192.168.1.250] <==> Router1 [Serial 0/0/1 – 192.168.1.246] <==> Router2 [Serial 0/0/0 – 192.168.1.245] <==> Router2 [FastEthernet0/0 – 20.0.0.1] <==> PC1 [Destination /source – 20.0.0.2]

By default RIP will use the route that has low hops counts between source and destination. In our network route1 has low hops counts, so it will be selected. We can use *tracert* command to verify it.
Now suppose route1 is down. We can simulate this situation by removing the cable attached between **Router0 [s0/0/1]** and **Router2 [s0/0/1]**.

What will happen now? There is no need to worry. RIP will automatically reroute the traffic.
Use ***tracert*** command again to see the magic of dynamic routing.