

# **PERIYAR UNIVERSITY**

(NAAC 'A++' Grade with CGPA 3.61 (Cycle - 3)  
State University - NIRF Rank 59 - NIRF Innovation Band of 11-50)  
SALEM - 636 011, Tamil Nadu, India.

**CENTRE FOR DISTANCE AND ONLINE EDUCATION  
(CDOE)**

**MASTER OF COMPUTER APPLICATIONS**

**SEMESTER – I**



**Elective: NETWORK PROTOCOL LAB**  
(Candidates admitted from 2024 onwards)

**PERIYAR UNIVERSITY**

**CENTRE FOR DISTANCE AND ONLINE EDUCATION  
(CDOE)  
MCA 2024 admission onwards**

**Elective Course Lab  
NETWORK PROTOCOL LAB [24DLCSAE02]**

Prepared by:

**Dr. D. Arul Pon Daniel**

Deputy Principal,

Assistant Professor in Department of Computer Science,

Jayarani Arts & Science College for Women,

Salem – 636002.

## SYLLABUS

**Course Code: 24DLCSAE02**

**Credits: 1**

### **Network Protocol Lab**

#### **Course Objectives**

- To understand and implement the basic concepts of Transmission Control Protocol/Internet Protocol and associated functions.
- To acquire programming skills in Implement various technologies and services associated with network protocols along with the challenges of data transfer.
- Implement the importance and functioning of Routing Protocols over communication service.
- To acquire skills to connect two routers and any two switches.
- To comprehend related to SSH protocols and accessing the remote device.

#### **Implement the following using Linux / Windows environments**

1. Implement the following commands
  - a. ipconfig
  - b. ping
  - c. traceroute
  - d. netsat
  - e. nslookup
2. Implement the following server commands
  - a. ifconfig
  - b. ip
  - c. tracepath
  - d. ss
  - e. tcpdum
3. Connect and place the given file in the FTP server
4. Install packet tracer and connect a computer to router, switch and get a Icmp request
5. Implement the SSH protocols and accessing the remote device
6. Connect any two switches and get the status of each switches
7. Connect two routers and get packets from the routers.
8. Get the access of the router by connecting with working computer
9. Identify the route password of server and get the connection using telnet
10. Install wire shark for capture and analyse the packets (TCP /UDP)

## Course Outcomes

On the successful completion of the course, students will be able to

CO1:	Comprehend the programming skills the SSH protocols and accessing the remote device	K1-K6
CO2:	Understand and implement the various functioning of Routing Protocols over communication service	K1-K6
CO3:	Evaluate the use of FTP server	K1-K6
CO4:	Design to Connect any two switches and get the status of each switches	K1-K6
CO5:	Solve to Connect two routers and get packets from the routers	K1-K6

**K1-Remember, K2-Understand, K3-Apply, K4-Analyze, K5- Evaluate, K6- Create**

## Mapping Course outcomes with Programme outcomes

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	S	-	-	-	-	L	-	-	-	-	-	-
CO2	S	-	M	-	M	L	-	-	-	-	-	-
CO3	S	-	S	-	S	L	-	-	-	S	S	S
CO4	S	-	S	-	S	L	-	-	-	S	S	S
CO5	S	-	S	-	S	L	-	-	-	S	S	S

**S-Strong; M-Medium; L-Low**

## Extra lab tips & checklist

- Always run privileged commands as **Administrator** (Windows) or **sudo/root** (Linux).
- Use ss instead of netstat on newer Linux distributions.
- Prefer **SSH** over Telnet in real environments. Telnet is insecure (cleartext credentials).
- When using packet capture tools, get written permission for networks you do not own.
- For Cisco devices in Packet Tracer use show running-config, show interfaces, show ip route, debug ip icmp (lab-only) to inspect behavior.
- When showing sample outputs in reports, label them as example output.

## Important Safety & Access Notes

- Run commands requiring elevated privileges as Administrator (Windows) or with sudo/root (Linux).
- Use SSH (secure) instead of Telnet (insecure) for real networks. Telnet is only shown for controlled lab environments.
- Only capture traffic or attempt password recovery on devices and networks you own or have explicit permission to test.

## CONTENTS

S. No.	Title of the Program	Page No.
1.	Implement the following commands a) ipconfig b) ping c) traceroute d) netsat e) nslookup	6
2.	Implement the following server commands a) ifconfig b) ip c) tracepath d) ss e) tcpdum	12
3.	Connect and place the given file in the FTP server	17
4.	Install packet tracer and connect a computer to router, switch and get a Icmp request	23
5.	Implement the SSH protocols and accessing the remote device	27
6.	Connect any two switches and get the status of each switches	30
7.	Connect two routers and get packets from the routers.	33
8.	Get the access of the router by connecting with working computer	38
9.	Identify the route password of server and get the connection using telnet	40
10.	Install wire shark for capture and analyse the packets (TCP /UDP)	45

Ex. No. 1	Implement the following client/host commands
-----------	--

**a) ipconfig (Windows) / ifconfig (Linux) / ip addr (modern Linux)**

**Aim:**

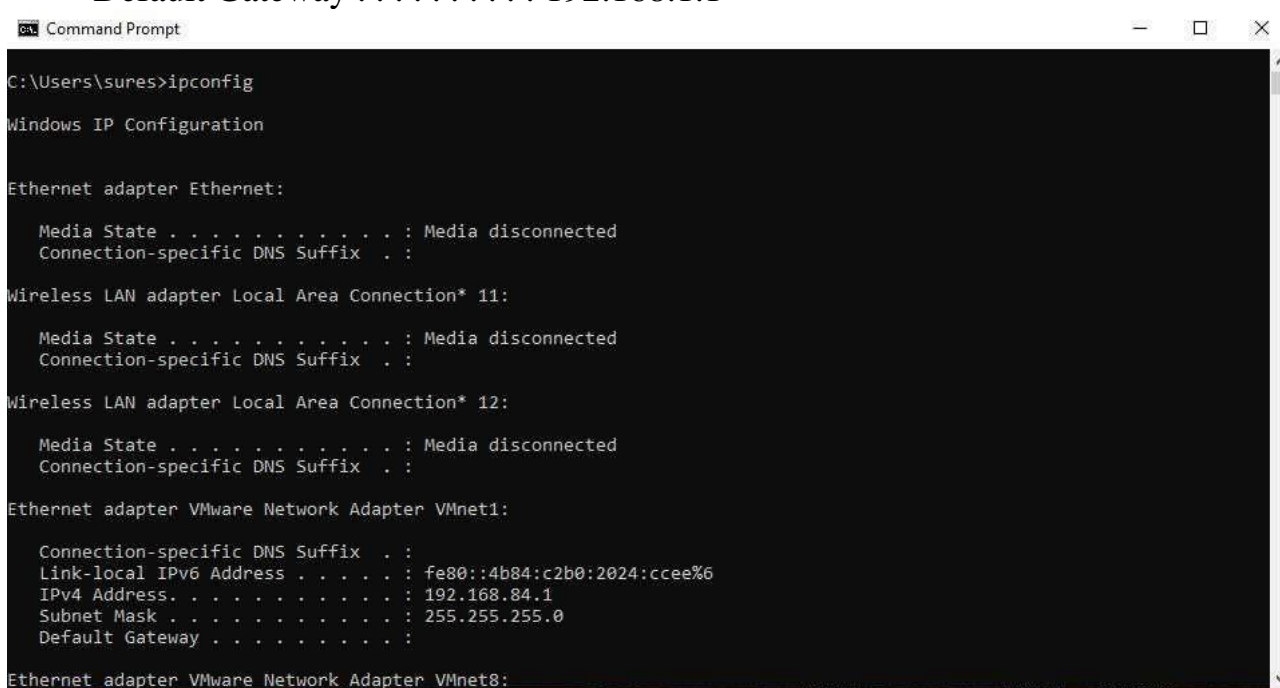
Display network interface configuration (IP, mask, gateway, MAC).

**Procedure (Windows):**

1. Open Command Prompt as Administrator.
2. Run: ipconfig /all

**Sample Input & output Screenshot (Windows ipconfig /all):**

```
Ethernet adapter Ethernet:
  Connection-specific DNS Suffix . :
  Description . . . . . : Intel(R) Ethernet
  Physical Address. . . . . : 00-1A-2B-3C-4D-5E
  DHCP Enabled. . . . . : Yes
  IPv4 Address. . . . . : 192.168.1.50(Preferred)
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.1.1
```

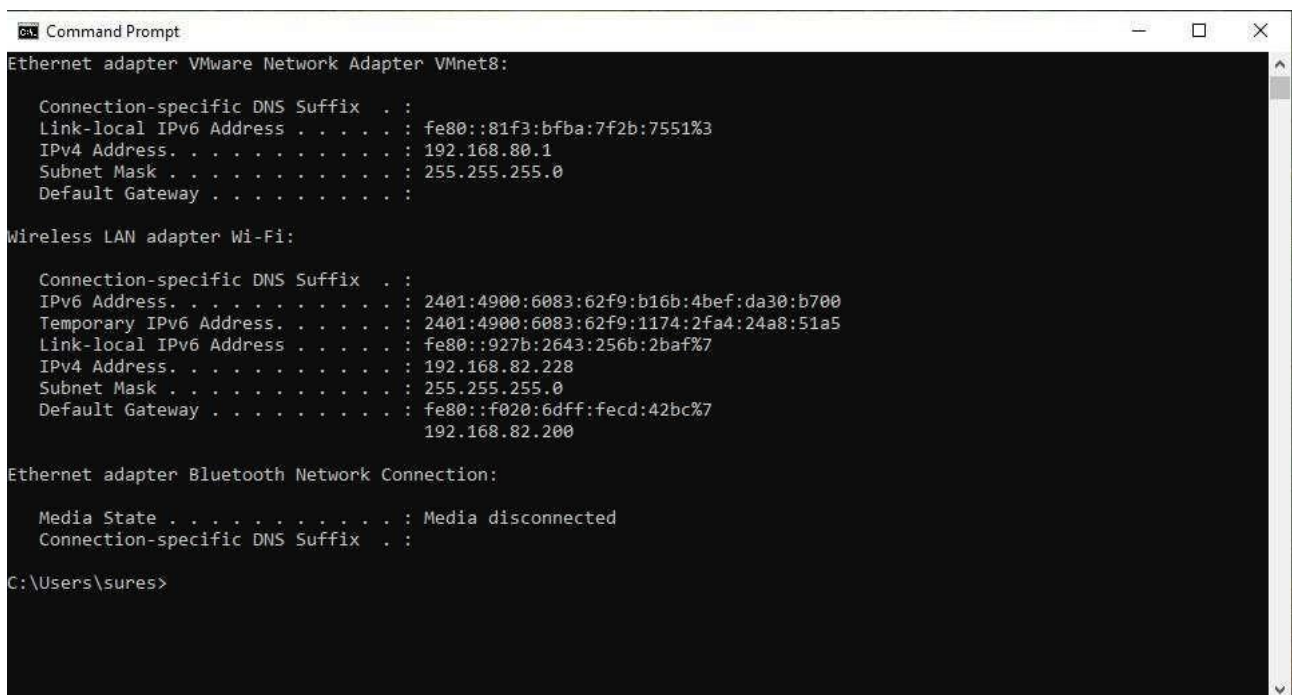


## Procedure (Linux):

1. Open terminal. For legacy: `sudo ifconfig -a`
2. Modern: `ip addr show` or `sudo ip a`

## Sample Input & output Screenshot (Linux):

```
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
link/ether 00:1a:2b:3c:4d:5e brd ff:ff:ff:ff:ff:ff
inet 192.168.1.50/24 brd 192.168.1.255 scope global dynamic enp3s0
```



```
ca Command Prompt
Ethernet adapter VMware Network Adapter VMnet8:

Connection-specific DNS Suffix . . :
Link-local IPv6 Address . . . . . : fe80::81f3:bfa:7f2b:7551%3
IPv4 Address. . . . . : 192.168.80.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . . :
IPv6 Address. . . . . : 2401:4900:6083:62f9:b16b:4bef:da30:b700
Temporary IPv6 Address. . . . . : 2401:4900:6083:62f9:1174:2fa4:24a8:51a5
Link-local IPv6 Address . . . . . : fe80::927b:2643:256b:2baf%7
IPv4 Address. . . . . : 192.168.82.228
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::f020:6dff:febd:42bc%7
192.168.82.200

Ethernet adapter Bluetooth Network Connection:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . :

C:\Users\sures>
```

## b) ping

**Aim:** Test reach ability and measure round-trip time (ICMP).

### Procedure (Windows & Linux):

1. Open terminal.
2. Run: ping <destination> (Windows defaults to 4 pings; Linux continues until stopped — use -c to set count).

Examples:

ping -n 4 8.8.8.8 (Windows)

ping -c 4 8.8.8.8 (Linux)

### Sample Input & output Screenshot:

Pinging 8.8.8.8 with 32 bytes of data:

Reply from 8.8.8.8: bytes=32 time=18ms TTL=118

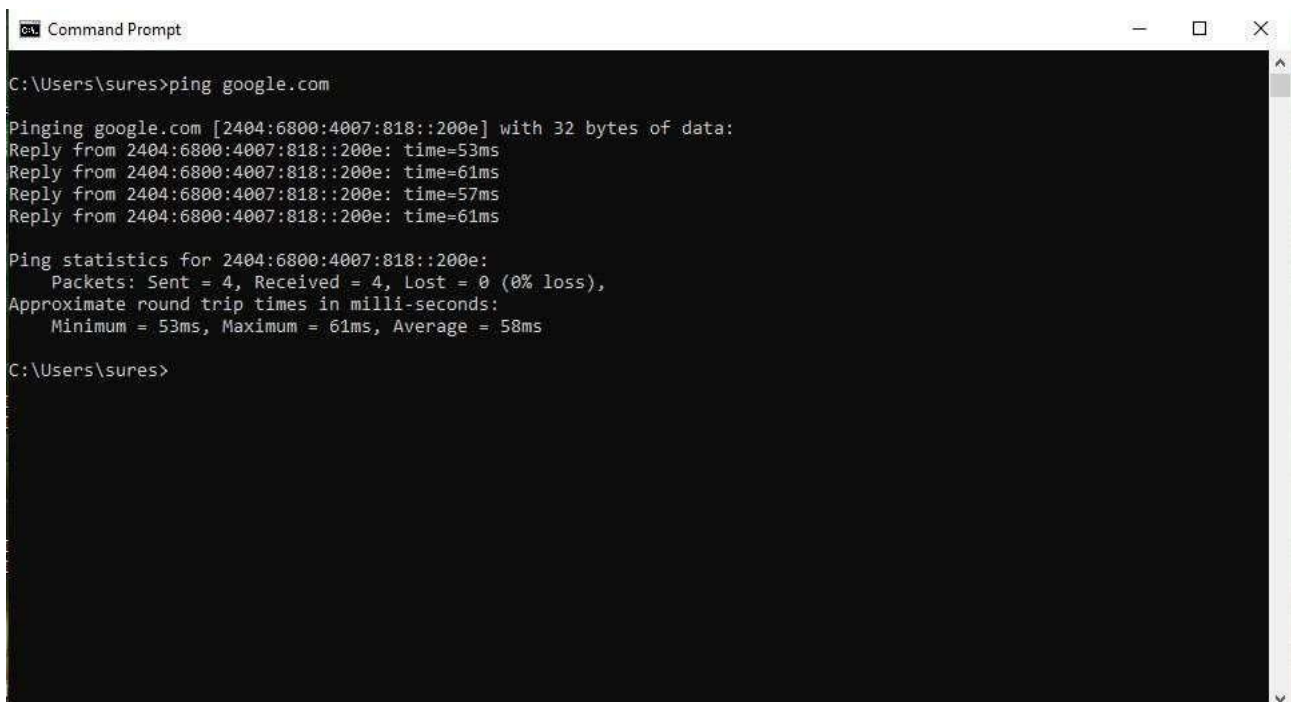
Reply from 8.8.8.8: bytes=32 time=19ms TTL=118

Ping statistics for 8.8.8.8:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)

Approximate round trip times in milli-seconds:

Minimum = 18ms, Maximum = 19ms, Average = 18ms



```
Command Prompt
C:\Users\sures>ping google.com

Pinging google.com [2404:6800:4007:818::200e] with 32 bytes of data:
Reply from 2404:6800:4007:818::200e: time=53ms
Reply from 2404:6800:4007:818::200e: time=61ms
Reply from 2404:6800:4007:818::200e: time=57ms
Reply from 2404:6800:4007:818::200e: time=61ms

Ping statistics for 2404:6800:4007:818::200e:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 53ms, Maximum = 61ms, Average = 58ms

C:\Users\sures>
```

### c) traceroute (Linux)/tracert (Windows)

**Aim:** Show the path (hops) packets take to reach a host.

#### Procedure (Windows):

- tracert example.com

#### Procedure (Linux):

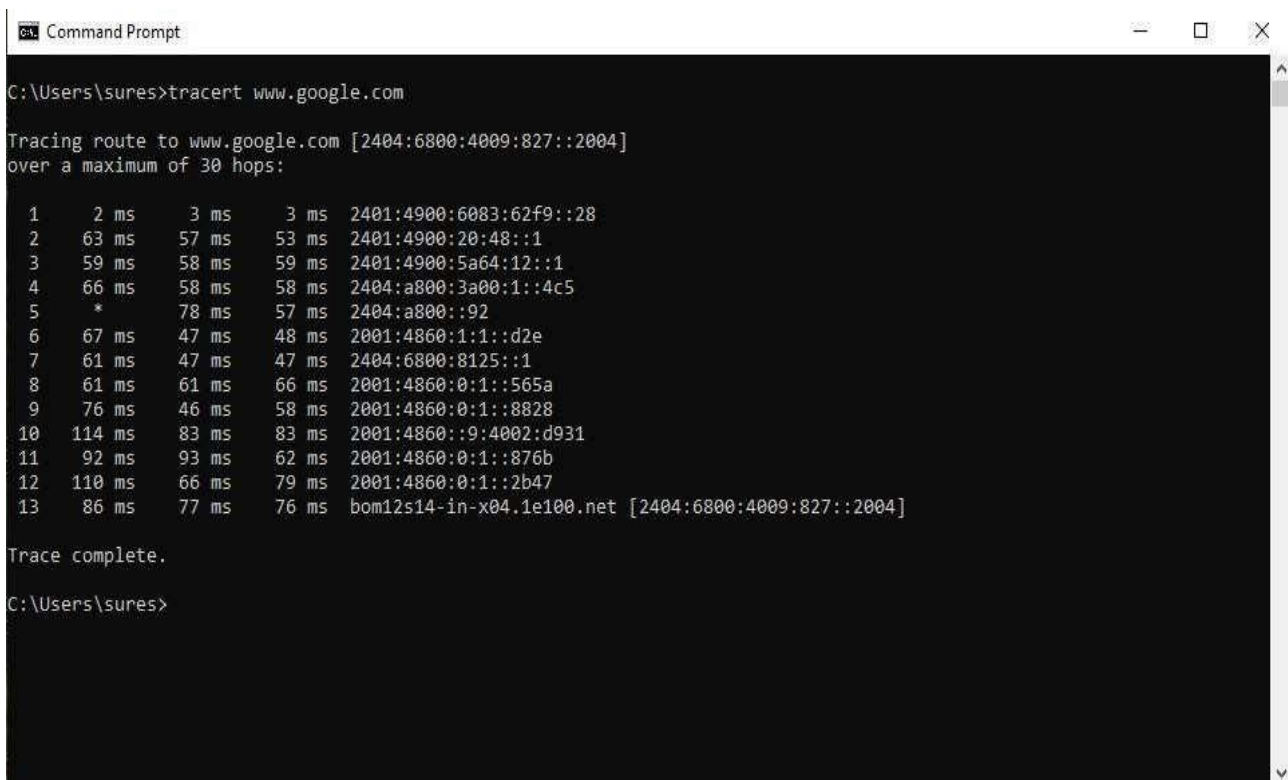
- traceroute example.com or tracepath example.com

#### Sample output Screenshot (tracert):

Tracing route to example.com [93.184.216.34]

```
1  <1 ms  <1 ms  <1 ms  192.168.1.1
2  10 ms  9 ms   9 ms   10.0.0.1
3  18 ms  17 ms  17 ms  203.0.113.1
4  21 ms  21 ms  21 ms  93.184.216.34
```

Trace complete.



```
ca. Command Prompt
C:\Users\sunes>tracert www.google.com

Tracing route to www.google.com [2404:6800:4009:827::2004]
over a maximum of 30 hops:

 1  2 ms   3 ms   3 ms   2401:4900:6083:62f9::28
 2  63 ms  57 ms  53 ms  2401:4900:20:48::1
 3  59 ms  58 ms  59 ms  2401:4900:5a64:12::1
 4  66 ms  58 ms  58 ms  2404:a800:3a00:1::4c5
 5  *      78 ms  57 ms  2404:a800::92
 6  67 ms  47 ms  48 ms  2001:4860:1:1::d2e
 7  61 ms  47 ms  47 ms  2404:6800:8125::1
 8  61 ms  61 ms  66 ms  2001:4860:0:1::565a
 9  76 ms  46 ms  58 ms  2001:4860:0:1::8828
10 114 ms  83 ms  83 ms  2001:4860::9:4002:d931
11 92 ms  93 ms  62 ms  2001:4860:0:1::876b
12 110 ms  66 ms  79 ms  2001:4860:0:1::2b47
13 86 ms  77 ms  76 ms  bom12s14-in-x04.1e100.net [2404:6800:4009:827::2004]

Trace complete.

C:\Users\sunes>
```

#### d) netsat — likely netstat (Windows/Linux)

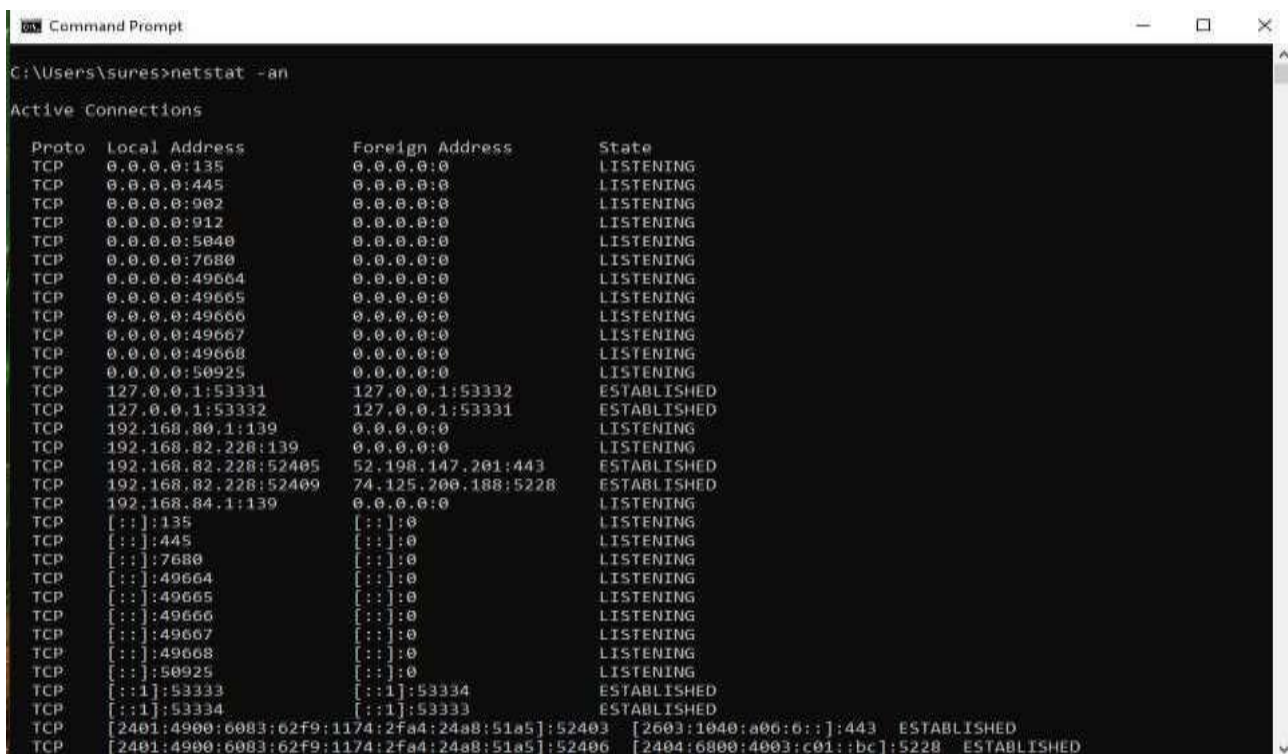
**Aim:** Display network connections, listening ports, and routing table.

#### **Procedure:**

1. Windows: netstat -ano (shows PID)
2. Linux: sudo netstat -tulpn or ss -tulpn (ss is preferred)

#### **Sample output Screenshot (netstat -ano):**

Proto	Local Address	Foreign Address	State	PID
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING	1234
TCP	192.168.1.50:5000	203.0.113.5:443	ESTABLISHED	4321



```
C:\Users\sures>netstat -an
Active Connections
Proto Local Address           Foreign Address         State
TCP   0.0.0.0:135             0.0.0.0:0               LISTENING
TCP   0.0.0.0:445             0.0.0.0:0               LISTENING
TCP   0.0.0.0:902             0.0.0.0:0               LISTENING
TCP   0.0.0.0:912             0.0.0.0:0               LISTENING
TCP   0.0.0.0:5080            0.0.0.0:0               LISTENING
TCP   0.0.0.0:7680            0.0.0.0:0               LISTENING
TCP   0.0.0.0:49664           0.0.0.0:0               LISTENING
TCP   0.0.0.0:49665           0.0.0.0:0               LISTENING
TCP   0.0.0.0:49666           0.0.0.0:0               LISTENING
TCP   0.0.0.0:49667           0.0.0.0:0               LISTENING
TCP   0.0.0.0:49668           0.0.0.0:0               LISTENING
TCP   0.0.0.0:50925           0.0.0.0:0               LISTENING
TCP   127.0.0.1:53331         127.0.0.1:53332        ESTABLISHED
TCP   127.0.0.1:53332         127.0.0.1:53331        ESTABLISHED
TCP   192.168.80.1:139        0.0.0.0:0               LISTENING
TCP   192.168.82.228:139     0.0.0.0:0               LISTENING
TCP   192.168.82.228:52405    52.198.147.201:443     ESTABLISHED
TCP   192.168.82.228:52409    74.125.200.188:5228    ESTABLISHED
TCP   192.168.84.1:139       0.0.0.0:0               LISTENING
TCP   [::]:135               [::]:0                  LISTENING
TCP   [::]:445               [::]:0                  LISTENING
TCP   [::]:7680              [::]:0                  LISTENING
TCP   [::]:49664             [::]:0                  LISTENING
TCP   [::]:49665             [::]:0                  LISTENING
TCP   [::]:49666             [::]:0                  LISTENING
TCP   [::]:49667             [::]:0                  LISTENING
TCP   [::]:49668             [::]:0                  LISTENING
TCP   [::]:50925             [::]:0                  LISTENING
TCP   [::1]:53333            [::1]:53334            ESTABLISHED
TCP   [::1]:53334            [::1]:53333            ESTABLISHED
TCP   [2401:4900:6083:62f9:1174:2fa4:24a8:51a5]:52403 [2603:1040:a00:0:]:443 ESTABLISHED
TCP   [2401:4900:6083:62f9:1174:2fa4:24a8:51a5]:52406 [2404:6800:4003:c01:bc]:5228 ESTABLISHED
```

## e) nslookup

**Aim:** Query DNS servers to resolve domain names to IPs and vice-versa.

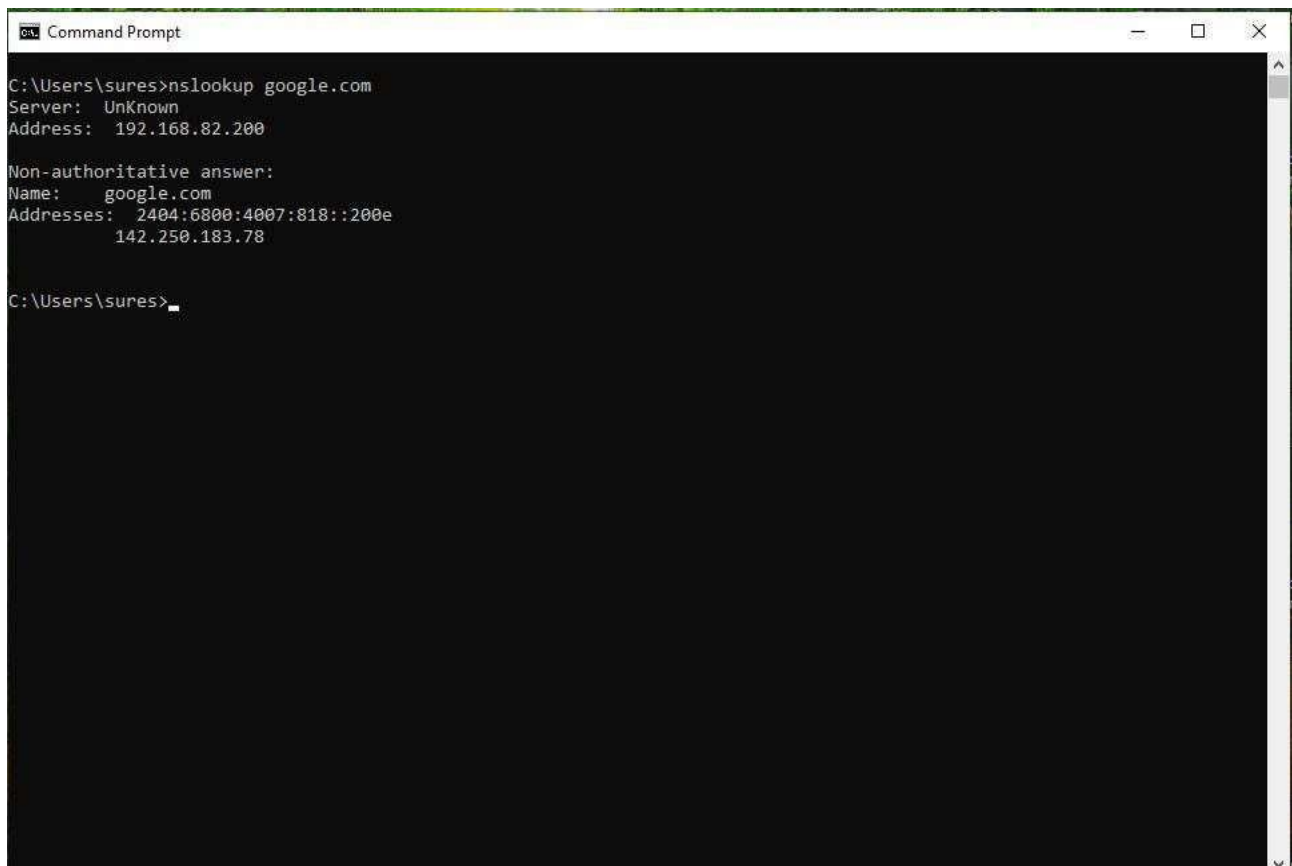
### Procedure (Windows & Linux):

1. nslookup example.com
2. (Optional) nslookup then server 8.8.8.8 then set type=mx etc.

### Sample output Screenshot:

Server: dns.google  
Address: 8.8.8.8

Non-authoritative answer:  
Name: example.com  
Address: 93.184.216.34



```
Command Prompt
C:\Users\sures>nslookup google.com
Server: UnKnown
Address: 192.168.82.200

Non-authoritative answer:
Name: google.com
Addresses: 2404:6800:4007:818::200e
          142.250.183.78

C:\Users\sures>
```

### RESULT:

Thus all the network protocol client commands are successfully tested and executed.

Ex. No. 2

Implement the server commands

**a) ifconfig (legacy) & b) ip (modern)**

**Aim:**

Configure and display network interfaces and addresses.

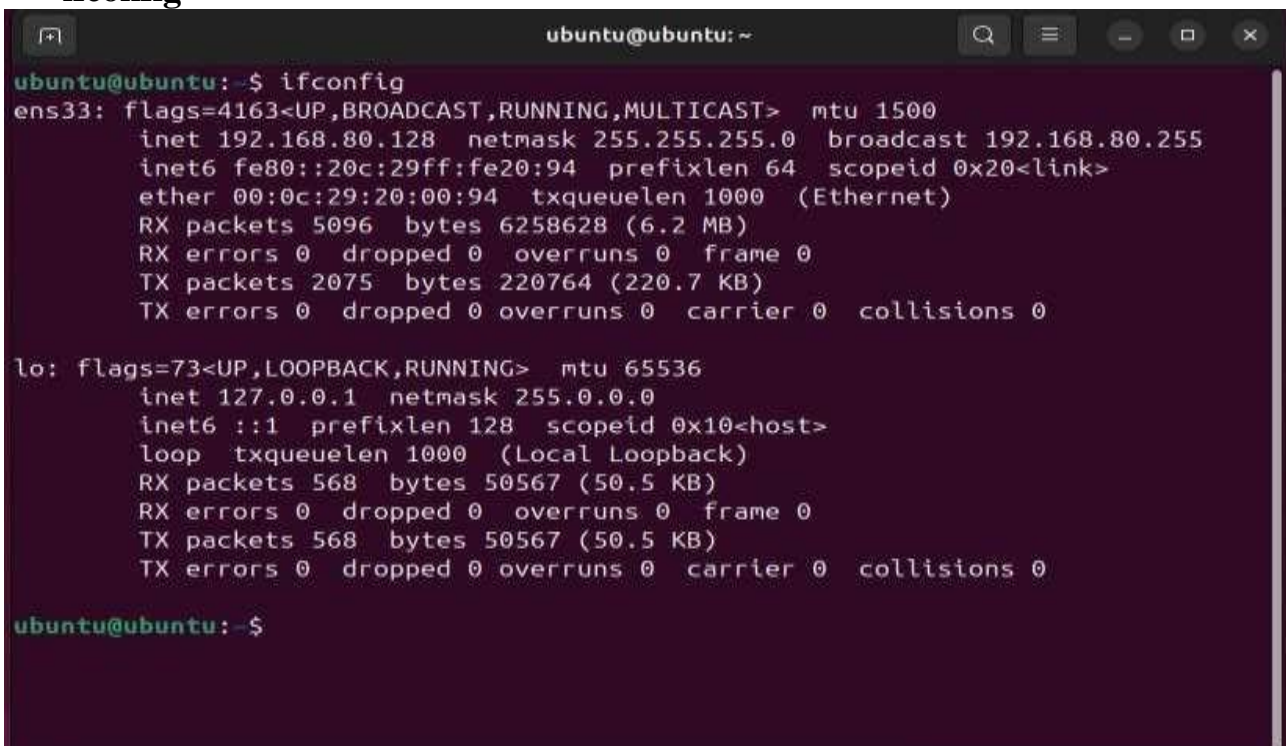
**Procedure (Linux):**

- Display: `sudo ip addr show` or `sudo ifconfig -a`
- Assign IP (example): `sudo ip addr add 192.168.10.10/24 dev eth0`
- Bring interface up/down: `sudo ip link set eth0 up / down`

**Sample output Screenshot (ip addr show):**

```
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
   inet 192.168.10.10/24 brd 192.168.10.255 scope global eth0
```

**a) ifconfig**



```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
   inet 192.168.80.128 netmask 255.255.255.0 broadcast 192.168.80.255
   inet6 fe80::20c:29ff:fe20:94 prefixlen 64 scopeid 0x20<link>
   ether 00:0c:29:20:00:94 txqueuelen 1000 (Ethernet)
   RX packets 5096 bytes 6258628 (6.2 MB)
   RX errors 0 dropped 0 overruns 0 frame 0
   TX packets 2075 bytes 220764 (220.7 KB)
   TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
   inet 127.0.0.1 netmask 255.0.0.0
   inet6 ::1 prefixlen 128 scopeid 0x10<host>
   loop txqueuelen 1000 (Local Loopback)
   RX packets 568 bytes 50567 (50.5 KB)
   RX errors 0 dropped 0 overruns 0 frame 0
   TX packets 568 bytes 50567 (50.5 KB)
   TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ubuntu@ubuntu:~$
```

b) ip

```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ ip address  
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 noprefixroute  
        valid_lft forever preferred_lft forever  
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000  
    link/ether 00:0c:29:20:00:94 brd ff:ff:ff:ff:ff:ff  
    altname enp2s1  
    inet 192.168.80.128/24 brd 192.168.80.255 scope global dynamic noprefixroute  
    ens33  
        valid_lft 1070sec preferred_lft 1070sec  
    inet6 fe80::20c:29ff:fe20:94/64 scope link  
        valid_lft forever preferred_lft forever  
ubuntu@ubuntu:~$
```

### c) **tracert**

#### **Aim:**

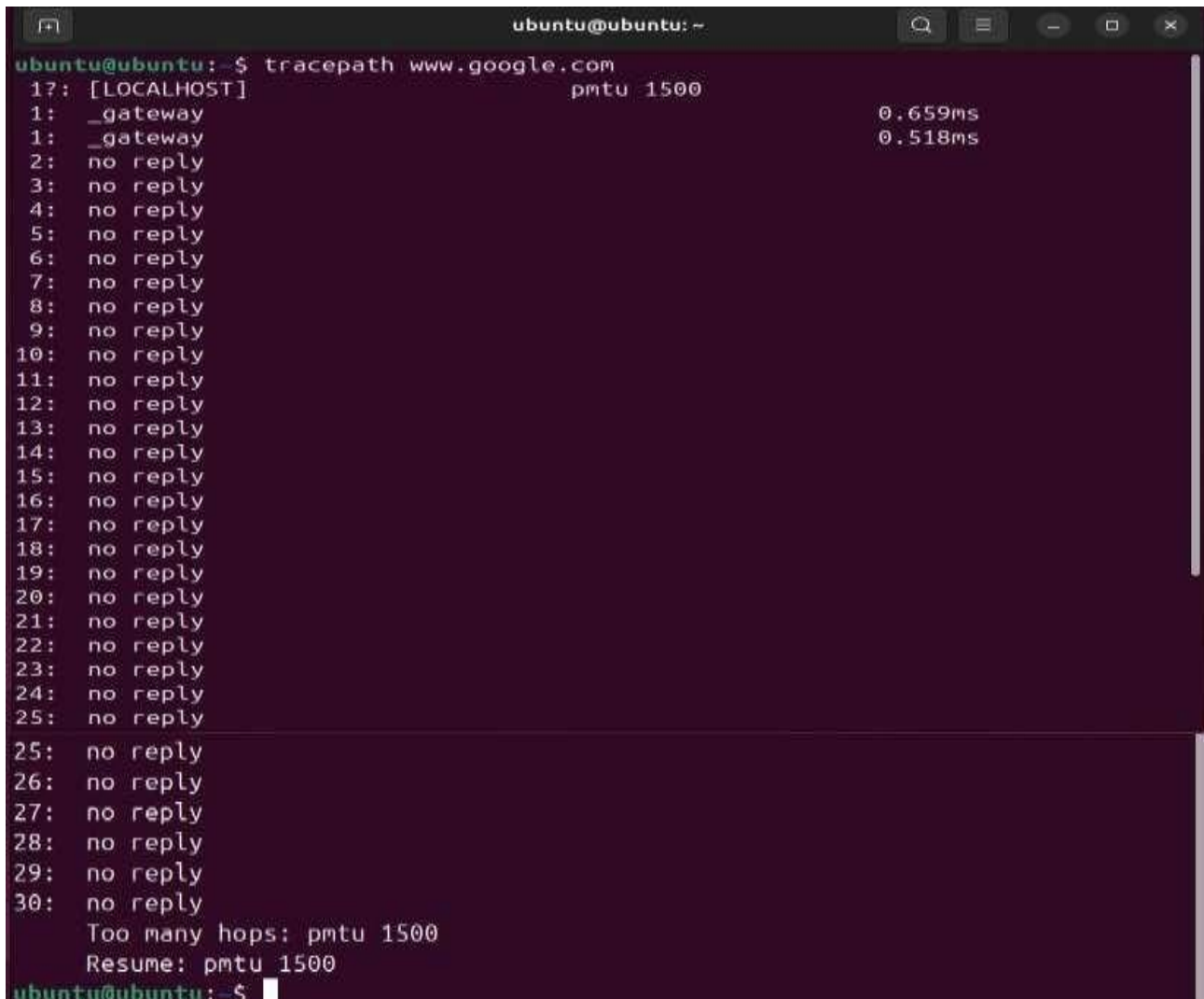
Determine path to destination and MTU along the path (Linux).

#### **Procedure:**

tracert example.com

#### **Sample output Screenshot:**

1?: [LOCALHOST]	pmtu 1500
1: 192.168.1.1	0.281ms
2: 10.0.0.1	8.760ms
3: 93.184.216.34	20.112ms reached



```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ tracert www.google.com  
1?: [LOCALHOST] pmtu 1500  
1: _gateway 0.659ms  
1: _gateway 0.518ms  
2: no reply  
3: no reply  
4: no reply  
5: no reply  
6: no reply  
7: no reply  
8: no reply  
9: no reply  
10: no reply  
11: no reply  
12: no reply  
13: no reply  
14: no reply  
15: no reply  
16: no reply  
17: no reply  
18: no reply  
19: no reply  
20: no reply  
21: no reply  
22: no reply  
23: no reply  
24: no reply  
25: no reply  
25: no reply  
26: no reply  
27: no reply  
28: no reply  
29: no reply  
30: no reply  
Too many hops: pmtu 1500  
Resume: pmtu 1500  
ubuntu@ubuntu:~$
```

## d) ss

### Aim:

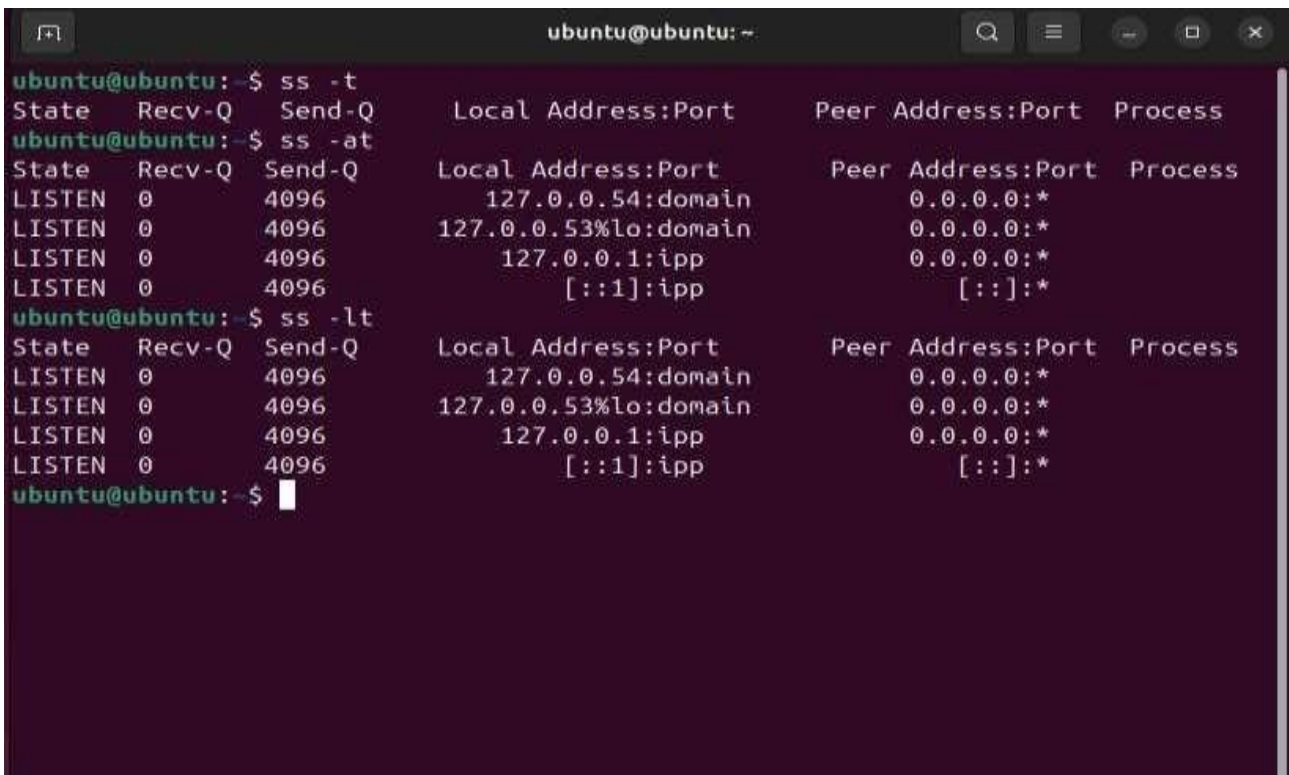
Socket statistics; view listening ports and established connections (replacement for netstat).

### Procedure:

- Enter “sudo ss -tulpn” in commend line in console

### Sample output Screenshot:

```
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
tcp LISTEN 0 128 0.0.0.0:22 *:* users:(("sshd",pid=1111,fd=3))
tcp ESTAB 0 0 192.168.10.10:22 203.0.113.5:54321
users:(("sshd",pid=2222,fd=4))
```



```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ ss -t
State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
ubuntu@ubuntu:~$ ss -at
State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
LISTEN 0 4096 127.0.0.54:domain 0.0.0.0:*
LISTEN 0 4096 127.0.0.53%lo:domain 0.0.0.0:*
LISTEN 0 4096 127.0.0.1:ipp 0.0.0.0:*
LISTEN 0 4096 [::1]:ipp [::]:*
ubuntu@ubuntu:~$ ss -lt
State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
LISTEN 0 4096 127.0.0.54:domain 0.0.0.0:*
LISTEN 0 4096 127.0.0.53%lo:domain 0.0.0.0:*
LISTEN 0 4096 127.0.0.1:ipp 0.0.0.0:*
LISTEN 0 4096 [::1]:ipp [::]:*
ubuntu@ubuntu:~$
```

## e) tcpdump

### Aim:

Capture and display network packets for analysis.

### Procedure:

1. `sudo tcpdump -i eth0` (live capture)
2. Save to file: `sudo tcpdump -i eth0 -w capture.pcap`
3. Filter e.g., `sudo tcpdump -i eth0 tcp port 80`

### Sample output Screenshot (text summary):

14:22:31.123456 IP 192.168.1.50.54321 > 93.184.216.34.80: Flags [S], seq 0, win 64240, options [mss 1460,sackOK,TS], length 0



```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ dpkg -s tcpdump  
Package: tcpdump  
Status: install ok installed  
Priority: optional  
Section: net  
Installed-Size: 1313  
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>  
Architecture: amd64  
Multi-Arch: foreign  
Version: 4.99.4-3ubuntu4  
Replaces: apparmor-profiles-extra (<< 1.12~)  
Depends: adduser, libc6 (>= 2.38), libpcap0.8t64 (>= 1.9.1), libssl3t64 (>= 3.0.0)  
Suggests: apparmor (>= 2.3)  
Breaks: apparmor-profiles-extra (<< 1.12~)  
Conffiles:  
 /etc/apparmor.d/usr.bin.tcpdump e6c5b5c0c144de30d497c7843b69ad81  
Description: command-line network traffic analyzer  
This program allows you to dump the traffic on a network. tcpdump  
is able to examine IPv4, ICMPv4, IPv6, ICMPv6, UDP, TCP, SNMP, AFS  
BGP, RIP, PIM, DVMRP, IGMP, SMB, OSPF, NFS and many other packet  
types.  
.  
It can be used to print out the headers of packets on a network  
interface, filter packets that match a certain expression. You can  
use this tool to track down network problems, to detect attacks  
or to monitor network activities.  
Homepage: https://www.tcpdump.org/  
Original-Maintainer: Romain Francoise <rfrancoise@debian.org>  
ubuntu@ubuntu:~$
```

### RESULT:

Thus all the network protocol server commands are successfully tested and executed.

Ex. No. 3	Connect and place the given file in the FTP server
-----------	--

**Aim:** Upload a file from a client to an FTP server.

**Procedure (Windows CLI ftp):**

1. ftp <server-ip>
2. Login with username/password.
3. binary (if binary file)
4. put localfile.txt remotefile.txt

**Procedure (Linux using ftp or curl/lftp):**

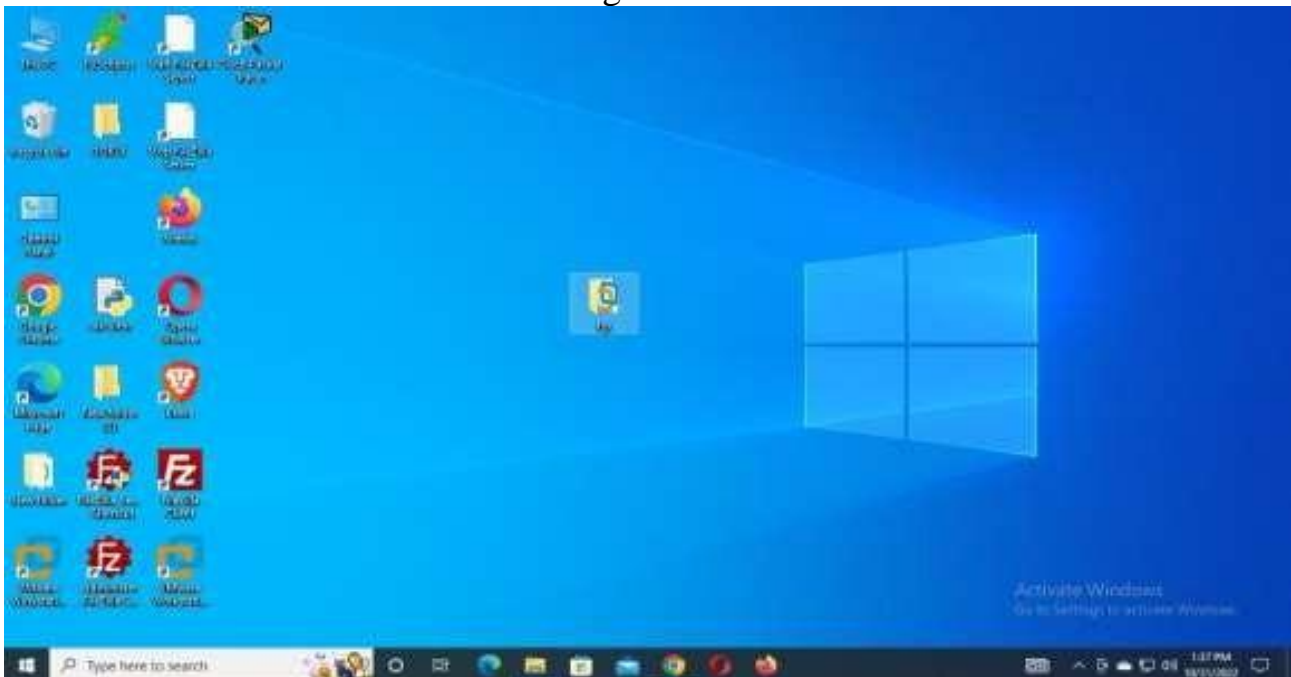
1. ftp <server-ip> or
2. curl -T localfile.txt ftp://user:pass@server-ip/remotefile.txt
3. Or use lftp -u user,password -e "put localfile.txt; bye" ftp://server-ip

**Sample FTP session:**

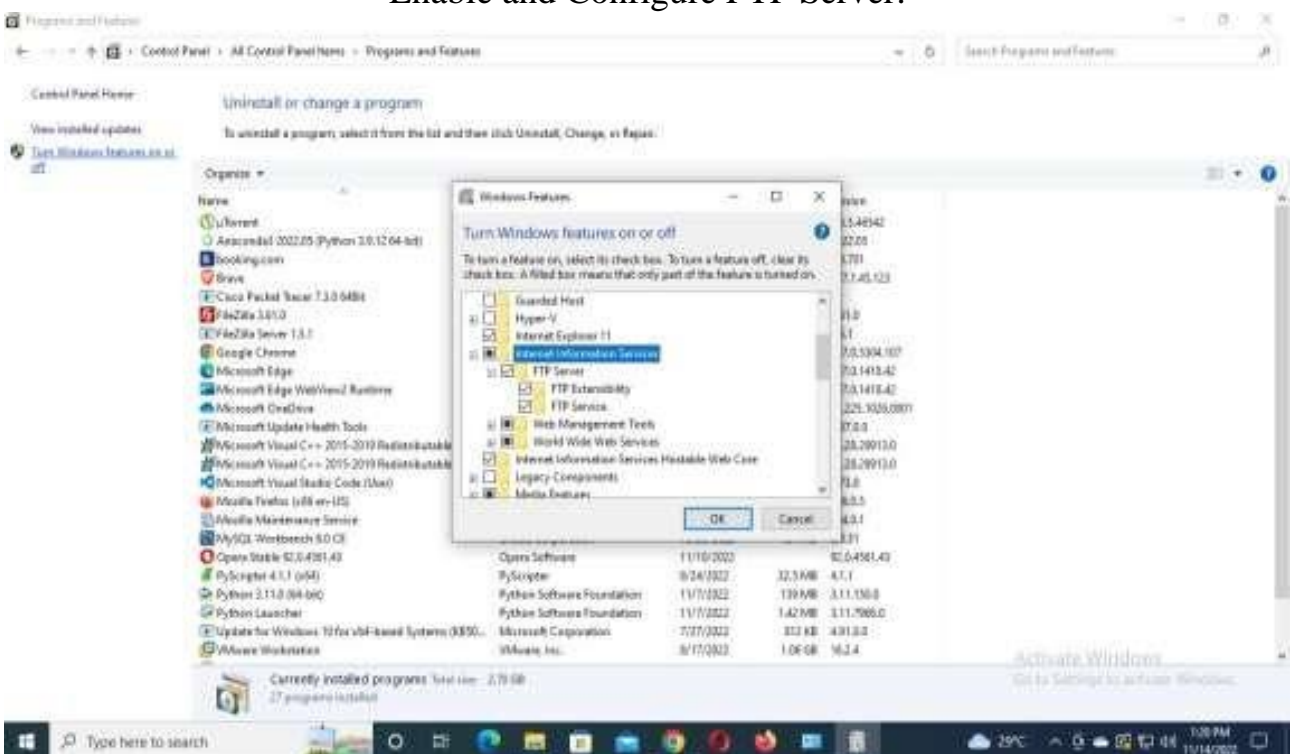
```
Connected to 192.168.10.100.
220 (vsFTPd 3.0.3)
Name (192.168.10.100:admin): user1
331 Please specify the password.
Password:
230 Login successful.
ftp> binary
ftp> put report.pdf
200 PORT command successful.
150 Ok to send data.
226 Transfer complete.
ftp> quit
```

## Screenshots:

Creating FTP folder:



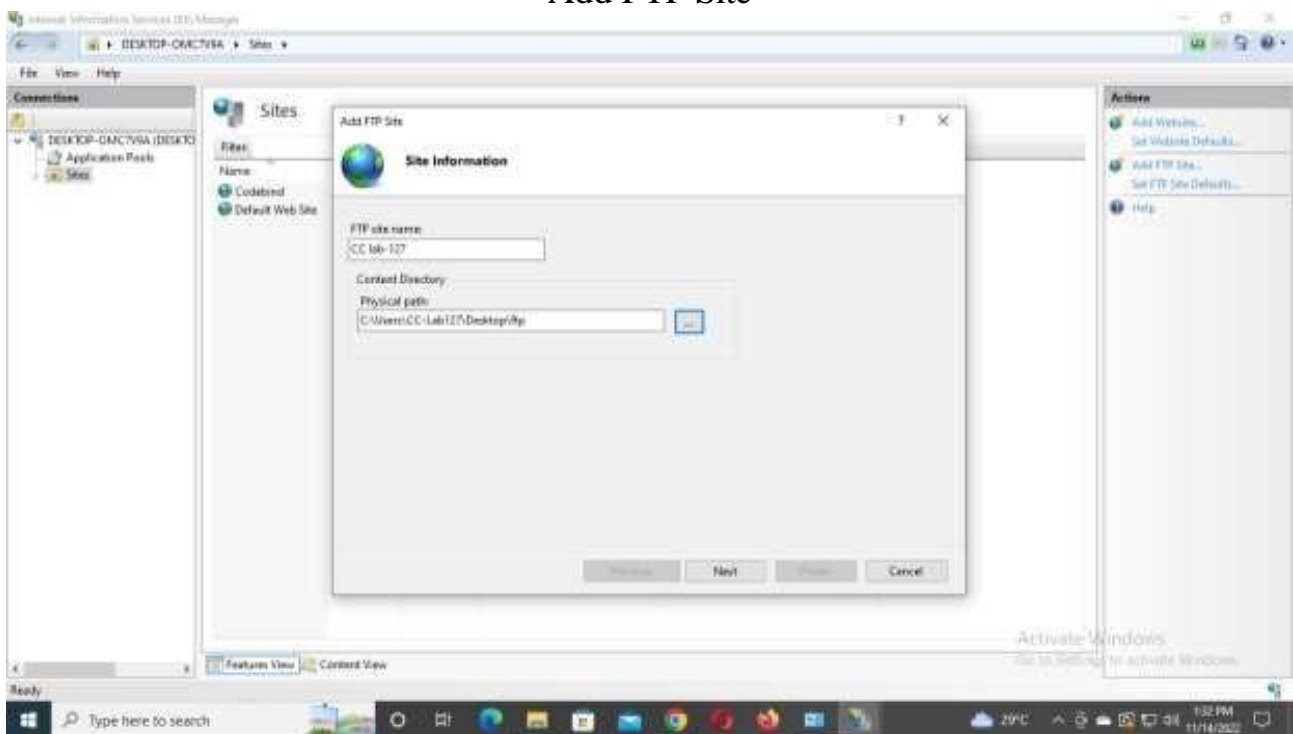
Enable and Configure FTP Server:

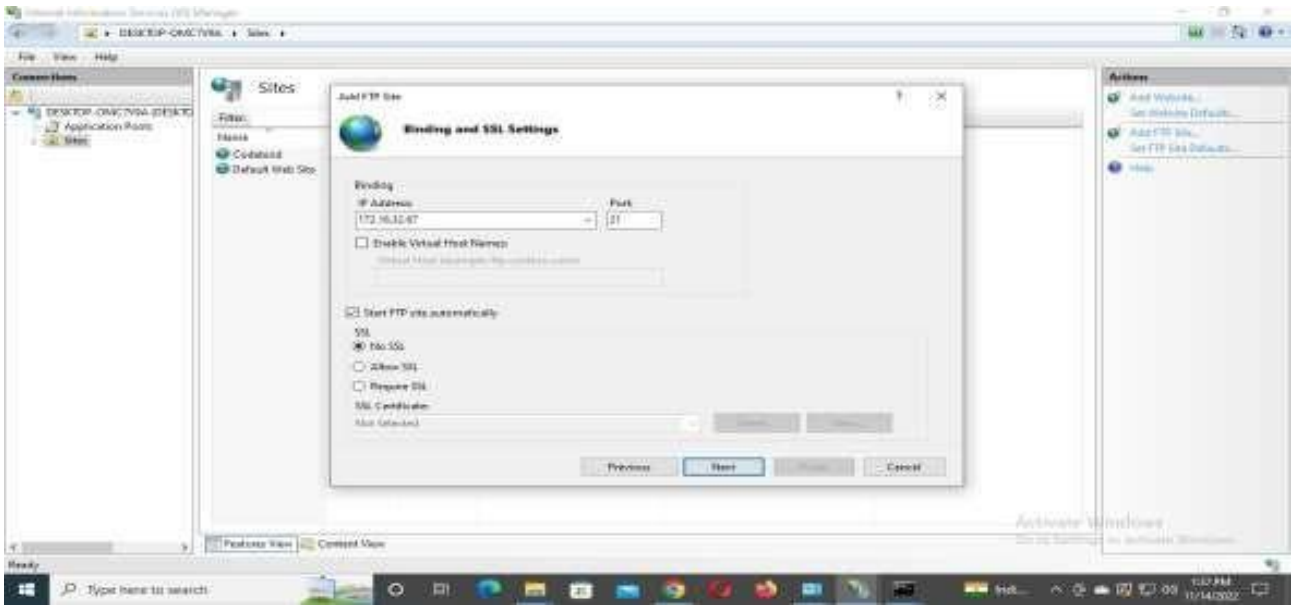


## Open Internet Information Services (IIS) Manager

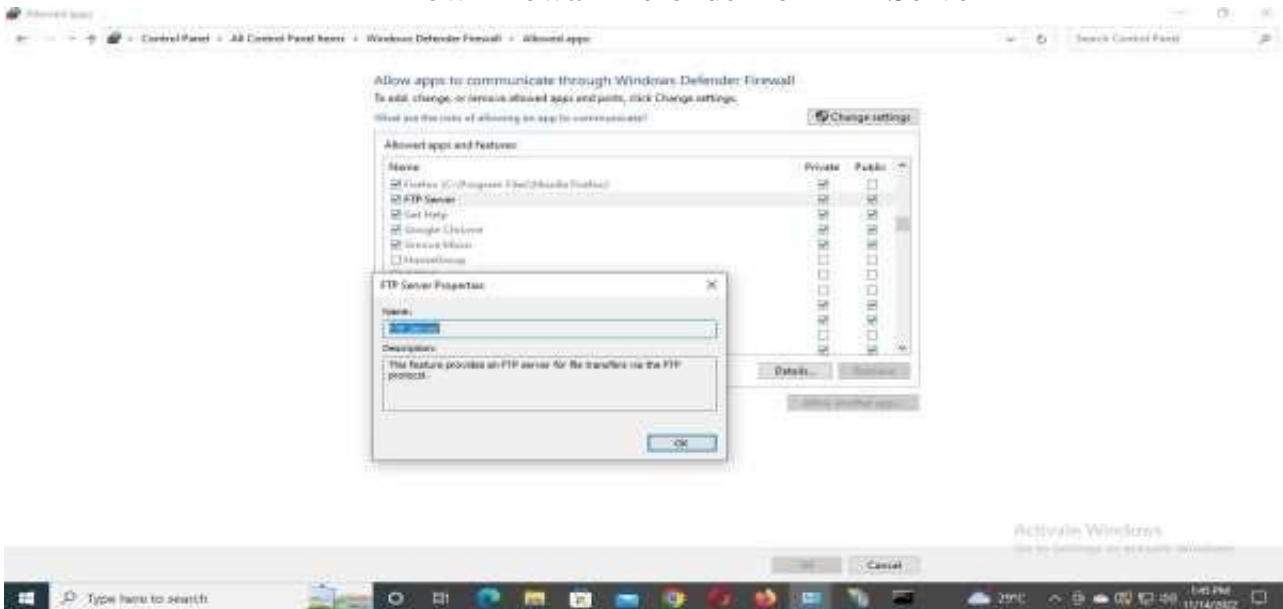


## Add FTP Site



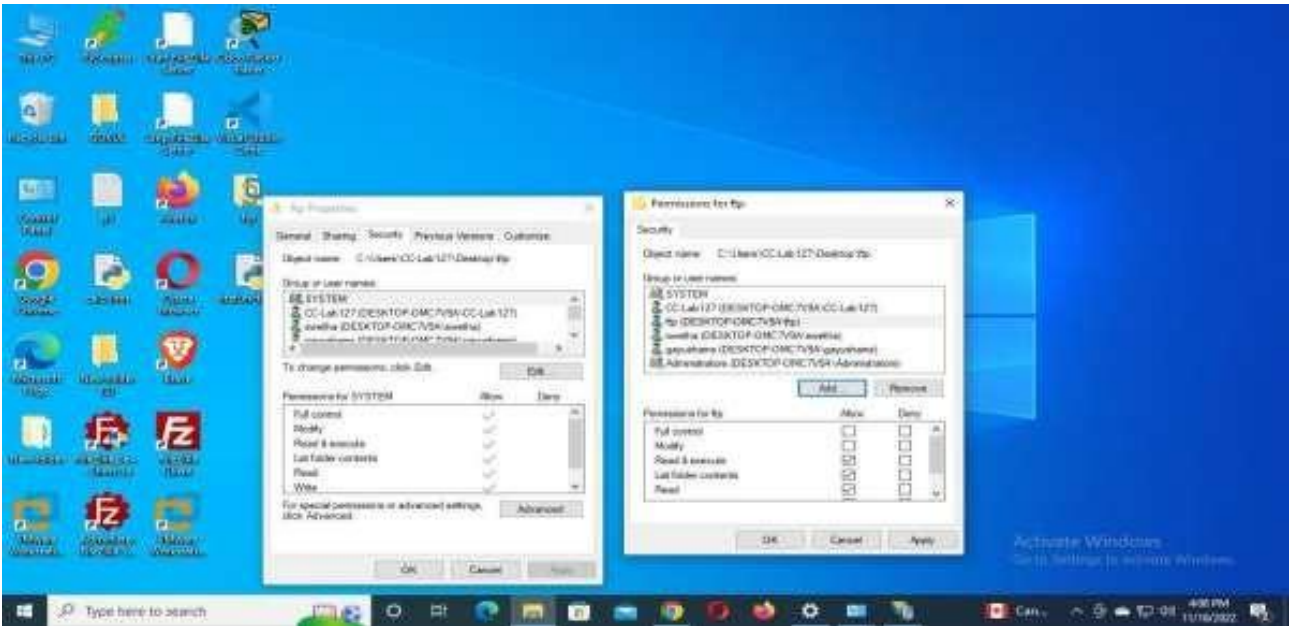


## Allow Firewall Defender for FTP Server

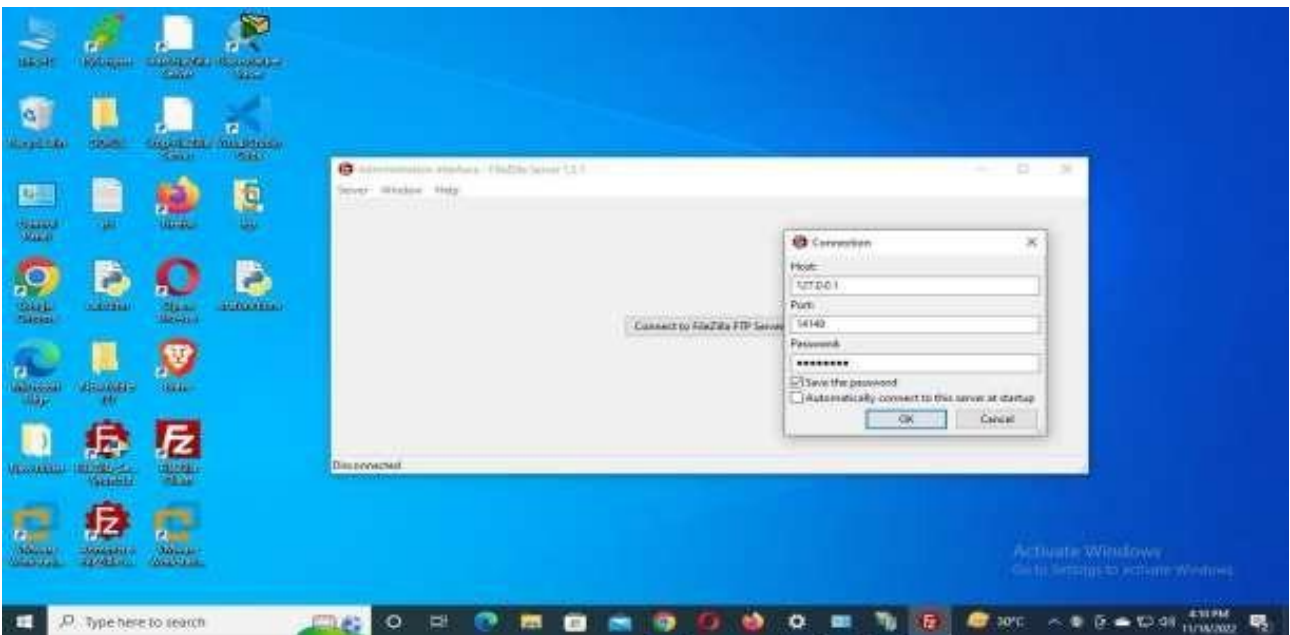


## Administrative Permission

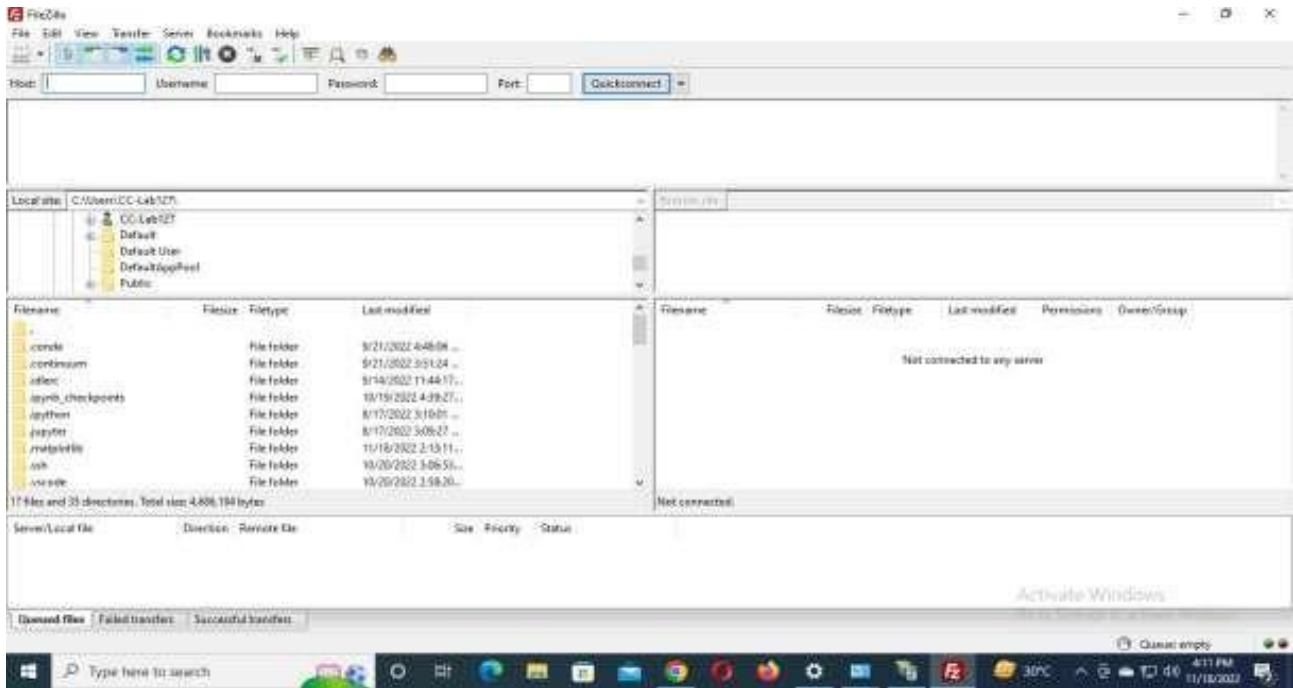
Add



### Connect the Server with FILEZILLA



### Establish Connection from Client Side using FILEZILLA CLIENT



## RESULT:

Thus all the FTP Operations are successfully tested and executed.

Ex. No. 4

Install packet tracer and connect a computer to router, switch and get a Icmp request

**Aim:**

Use Packet Tracer to simulate a small network and validate ICMP (ping) between PC and router.

**Procedure:**

1. Install Cisco Packet Tracer on Windows/Linux (follow installer).
2. Create topology: add PC — Switch — Router; connect with appropriate cables.
3. Assign IPs (PC: 192.168.1.10/24, Router interface: 192.168.1.1/24).
4. On PC, open command prompt (in Packet Tracer) and run ping 192.168.1.1.

**Sample output (Packet Tracer PC):**

Pinging 192.168.1.1 with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time<1ms TTL=255

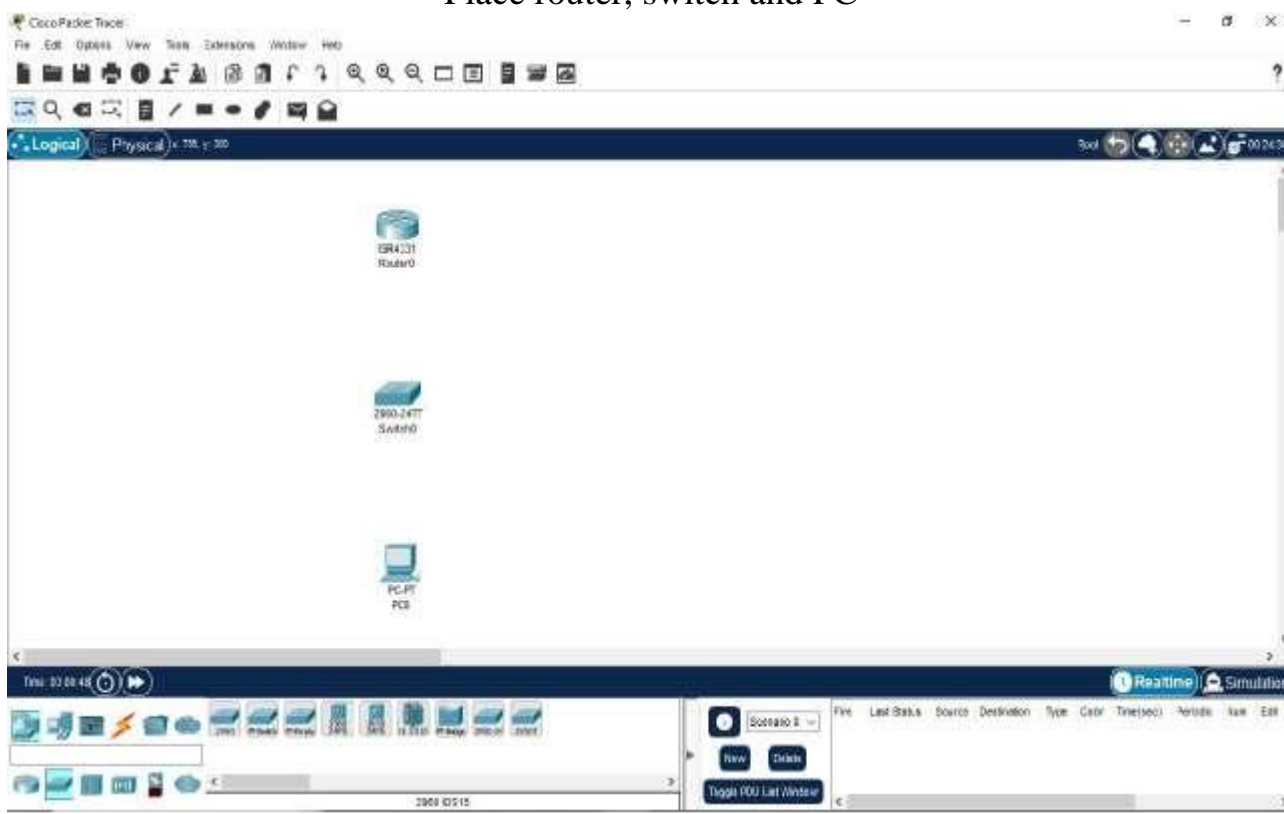
Reply from 192.168.1.1: bytes=32 time<1ms TTL=255

**Notes:**

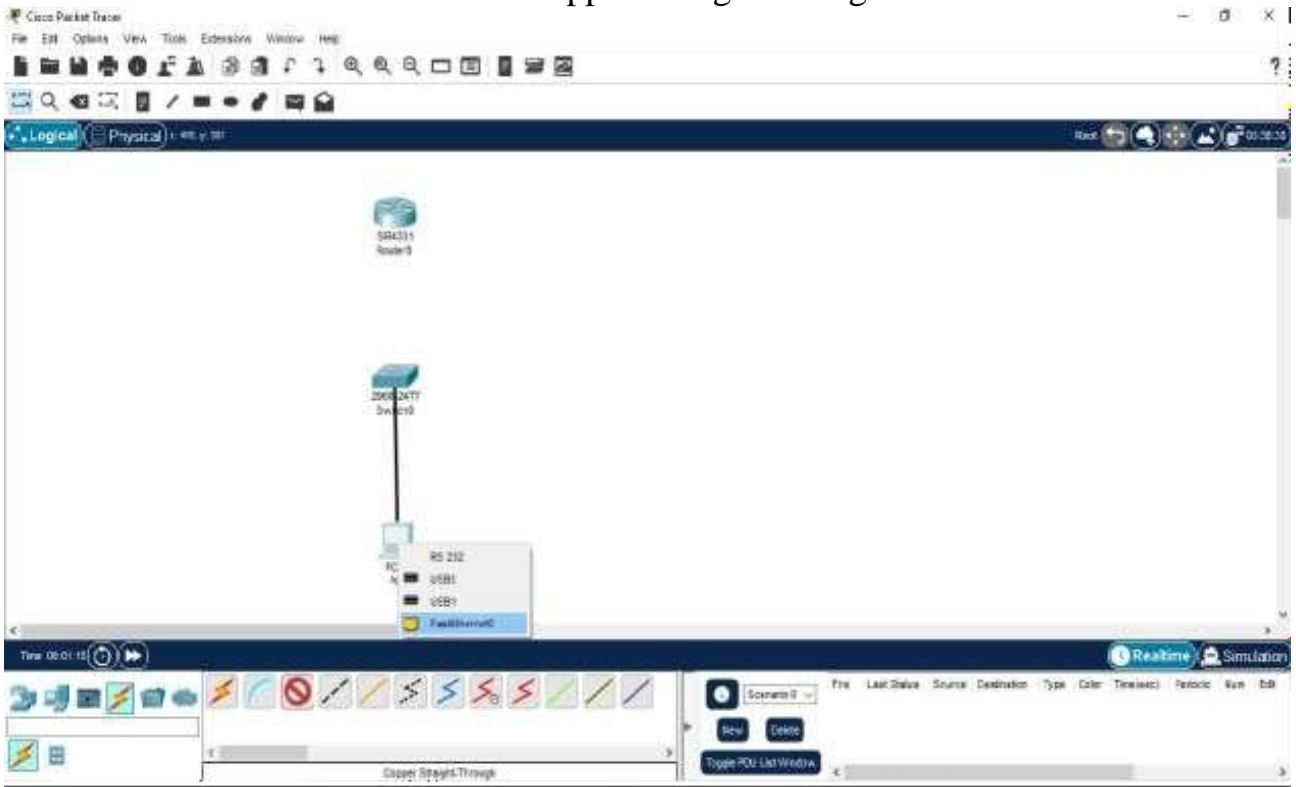
If no reply, check interface state (no shutdown) and cabling.

**Sample Input & output Screenshot:**

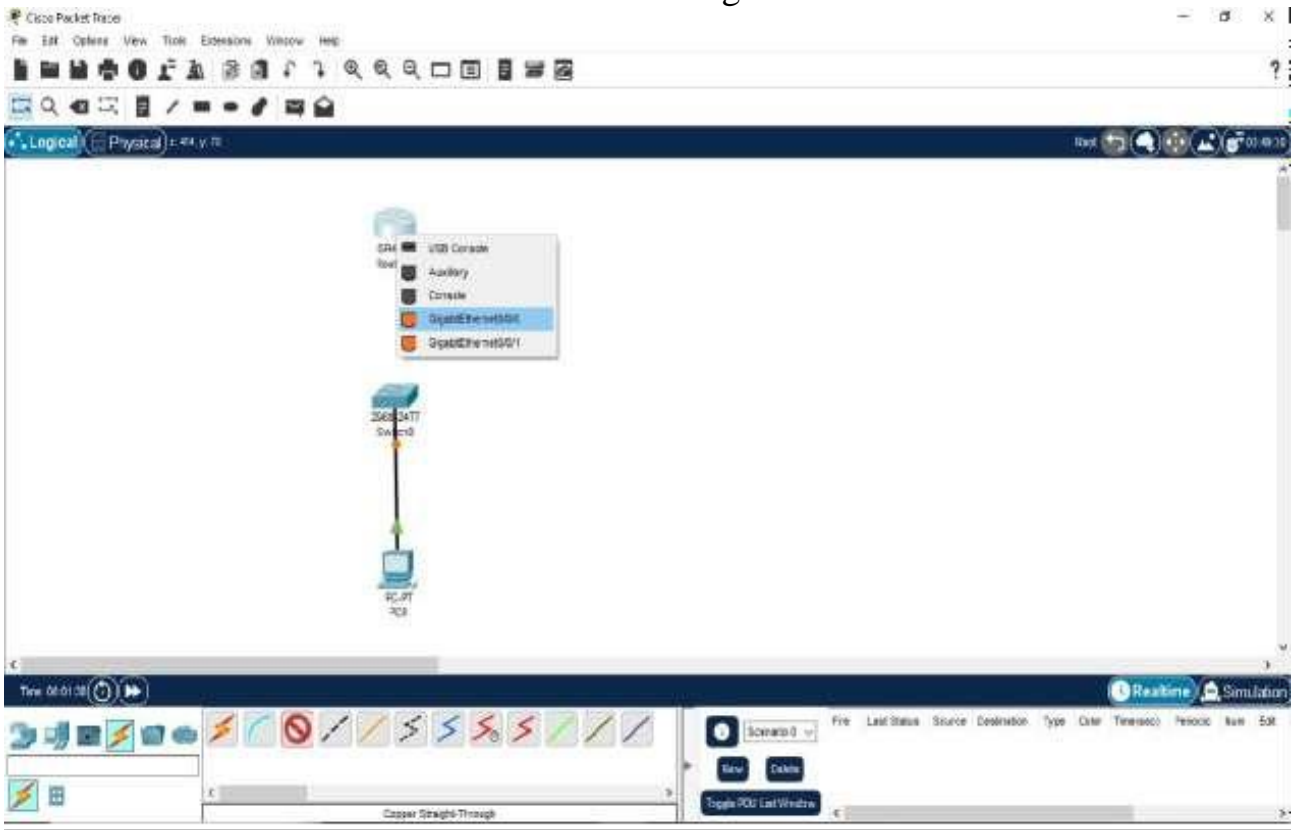
Place router, switch and PC



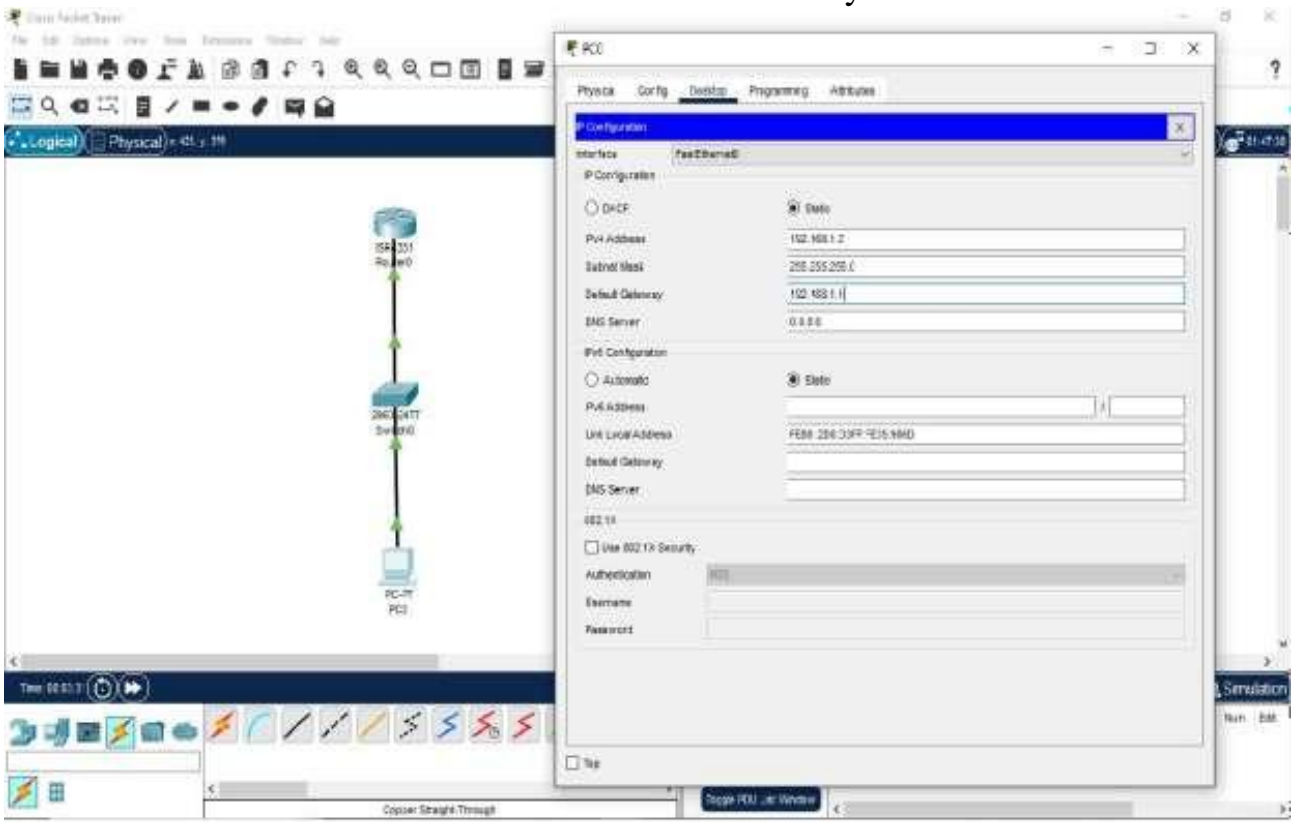
## Connect with Copper Straight through Cable



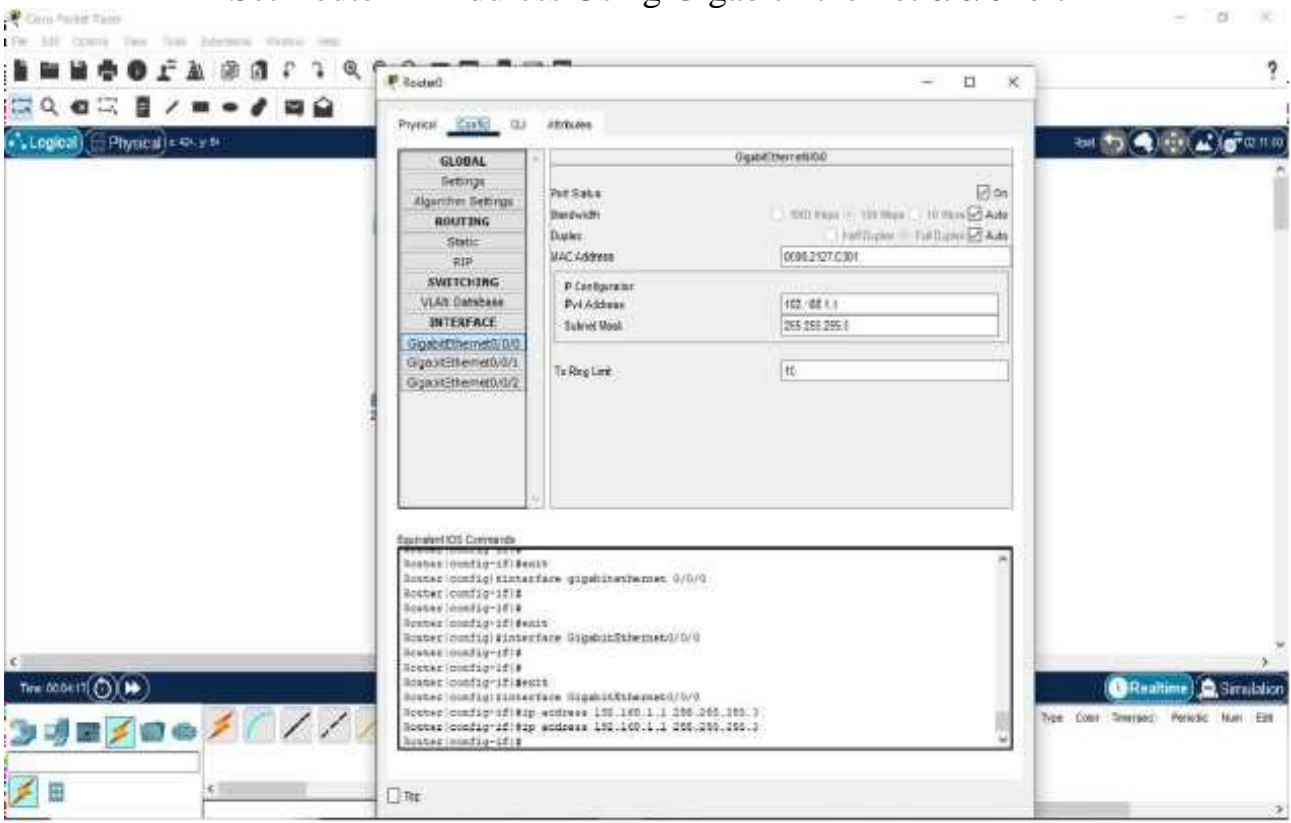
## Connect Switch and Router on Gigabit Ethernet 0/0/0Port



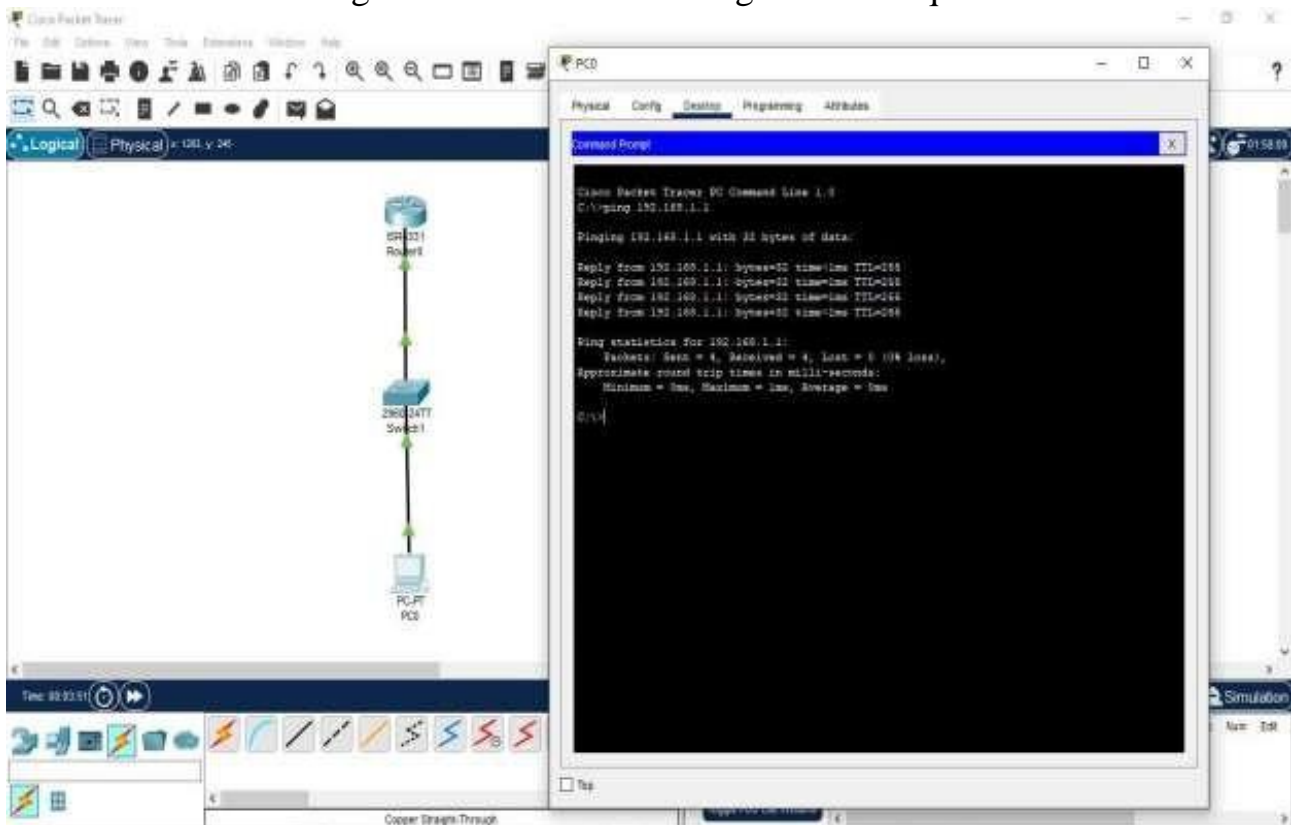
## Set IP address and Default Gateway for PC



## Set Router IP Address Using Gigabit Ethernet 0/0/0Port



## Ping the router IP address to get ICMP request



### RESULT:

Thus all the Router Operations are successfully tested and executed.

Ex. No. 5	Implement the SSH protocols and accessing the remote device
-----------	---

**Aim:** Securely connect to a remote device using SSH and verify authentication and connectivity.

**Procedure (Linux client to server):**

1. Ensure SSH server installed on target: `sudo apt install openssh-server` and `sudo systemctl start ssh`.
2. From client: `ssh user@server-ip` (or `ssh -p 2222 user@server-ip` if custom port).
3. Use key-based auth: generate key `ssh-keygen`, copy public key: `ssh-copy-id user@server-ip`.

**Sample output (client):**

```
$ ssh user@192.168.10.100
user@192.168.10.100's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-...)
user@server:~$
```

**Windows:**

- Use PowerShell `ssh user@server-ip` or PuTTY.

**Security note:**

Disable password auth if using keys, change default port, and allow only required users in `/etc/ssh/sshd_config`.

## Sample Input & output Screenshot:

```
lab4@lab4-virtual-machine:~$ sudo apt update
Hit:1 http://packages.microsoft.com/repos/code stable InRelease
Hit:2 http://ln.archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://ln.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 http://ln.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
225 packages can be upgraded. Run 'apt list --upgradable' to see them.
lab4@lab4-virtual-machine:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openssh-server is already the newest version (1:8.9p1-3).
0 upgraded, 0 newly installed, 0 to remove and 225 not upgraded.
lab4@lab4-virtual-machine:~$ sudo systemctl start sshd.service
lab4@lab4-virtual-machine:~$ sudo systemctl status sshd.service
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2022-11-24 16:43:33 IST; 26s ago
     Docs: man:ssh(8)
           man:ssh_config(5)
   Process: 5868 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Main PID: 5869 (sshd)
    Tasks: 1 (limit: 2247)
   Memory: 2.0M
      CPU: 45ms
   CGroup: /system.slice/ssh.service
           └─5868 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

   Tasks: 1 (limit: 2247)
   Memory: 2.0M
      CPU: 45ms
   CGroup: /system.slice/ssh.service
           └─5869 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Nov 24 16:43:33 lab4-virtual-machine systemd[1]: Starting OpenBSD Secure Shell server...
Nov 24 16:43:33 lab4-virtual-machine sshd[5869]: Server listening on 0.0.0.0 port 22.
Nov 24 16:43:33 lab4-virtual-machine sshd[5869]: Server listening on :: port 22.
Nov 24 16:43:33 lab4-virtual-machine systemd[1]: Started OpenBSD Secure Shell server.
lab4@lab4-virtual-machine:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.80.131 netmask 255.255.255.0 broadcast 192.168.80.255
    inet6 fe80::8fe6:5612:40ae:12d2 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:4b:80:80 txqueuelen 1000 (Ethernet)
    RX packets 7256 bytes 7736669 (7.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4284 bytes 326740 (326.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 1912 bytes 266697 (266.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1912 bytes 266697 (266.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lab4@lab4-virtual-machine:~$
```



Ex. No. 6	Connect any two switches and get the status of each switches
-----------	--

**Aim:**

Physically or virtually connect switches and verify interface and VLAN/status information.

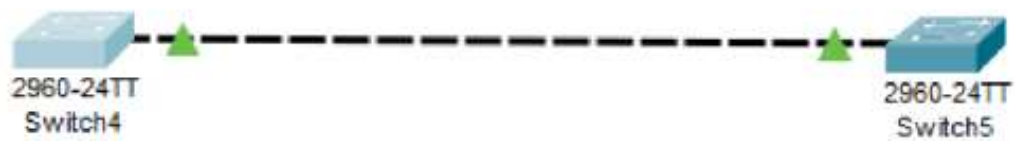
**Procedure (in Packet Tracer or real Cisco switches):**

1. Connect switch1 port Fa0/1 to switch2 Fa0/1.
2. On each switch, enable and check interfaces:  
show interfaces status  
show vlan brief  
show cdp neighbors (if CDP enabled)

**Sample outputs:**

```
Switch# show interfaces status
Port  Name  Status  Vlan  Duplex  Speed  Type
Fa0/1  connected  1  a-full a-100  10/100BaseTX
Fa0/2  notconnect  1  auto  auto
Switch# show cdp neighbors
Device ID    Local Intrfce  Holdtme  Capability  Platform  Port ID
Switch2     Fa0/1          120     S I         WS-C2960  Fa0/1
```

## Sample Input & output Screenshot

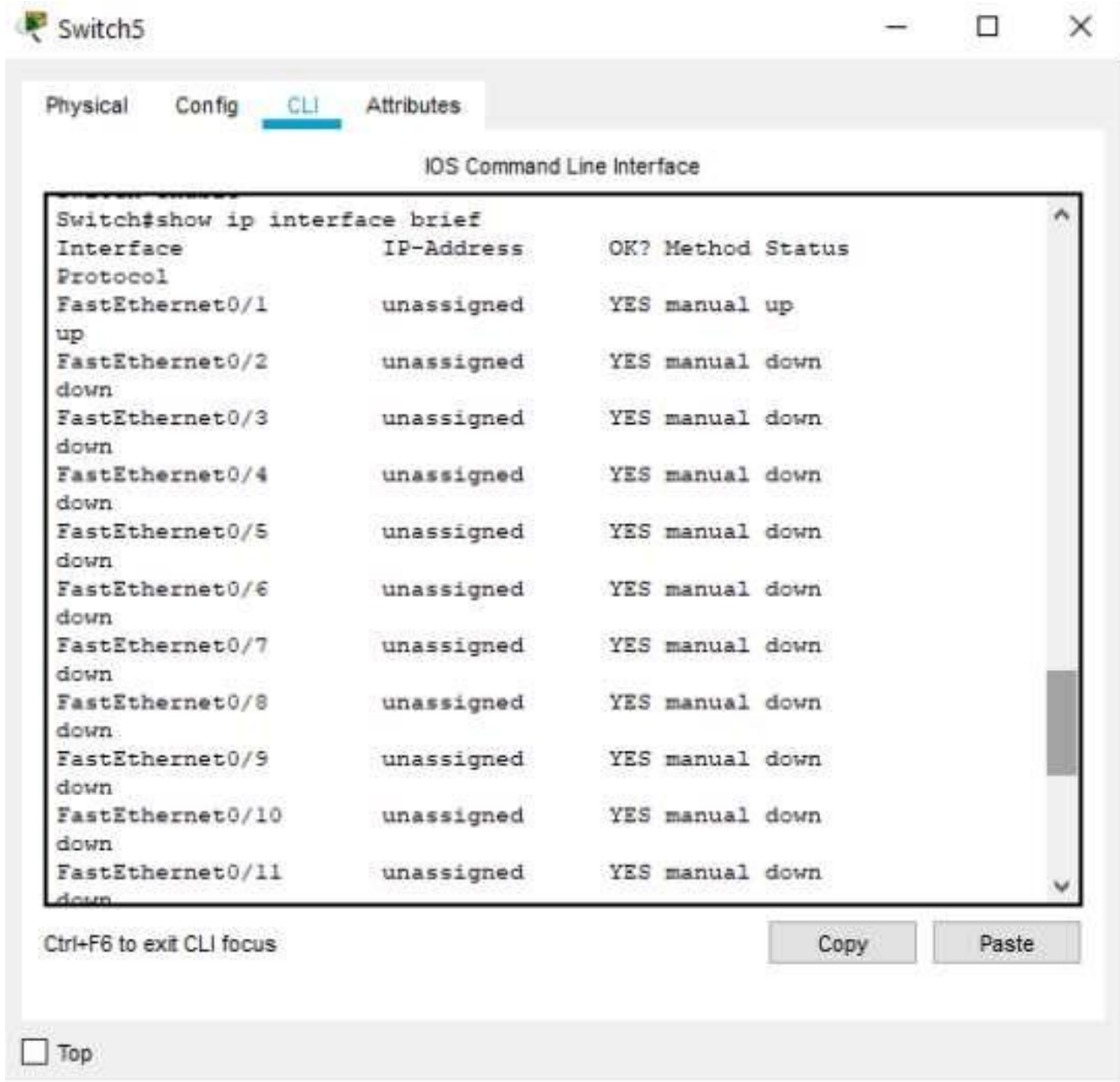


```
Switch5
Physical Config CLI Attributes
IOS Command Line Interface
Copyright (c) 1986-2005 by Cisco Systems, Inc.
Compiled Wed 13-Oct-05 22:05 by pt_team
Press RETURN to get started!
%LINK-5-CHANGED: Interface FastEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1,
changed state to up
%LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1,
changed state to down
%LINK-5-CHANGED: Interface FastEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1,
changed state to up
Switch>
Switch>enable
Switch#show ip interface brief
```

Ctrl+F6 to exit CLI focus

Copy Paste

Top



**RESULT:**

Thus the two Switches are connected successfully and tested.

Ex. No. 7	Connect two routers and get packets from the routers
-----------	--

**Aim:**

Configure routing between two routers and verify inter-router connectivity and packet flow.

**Procedure (Packet Tracer / Cisco IOS):**

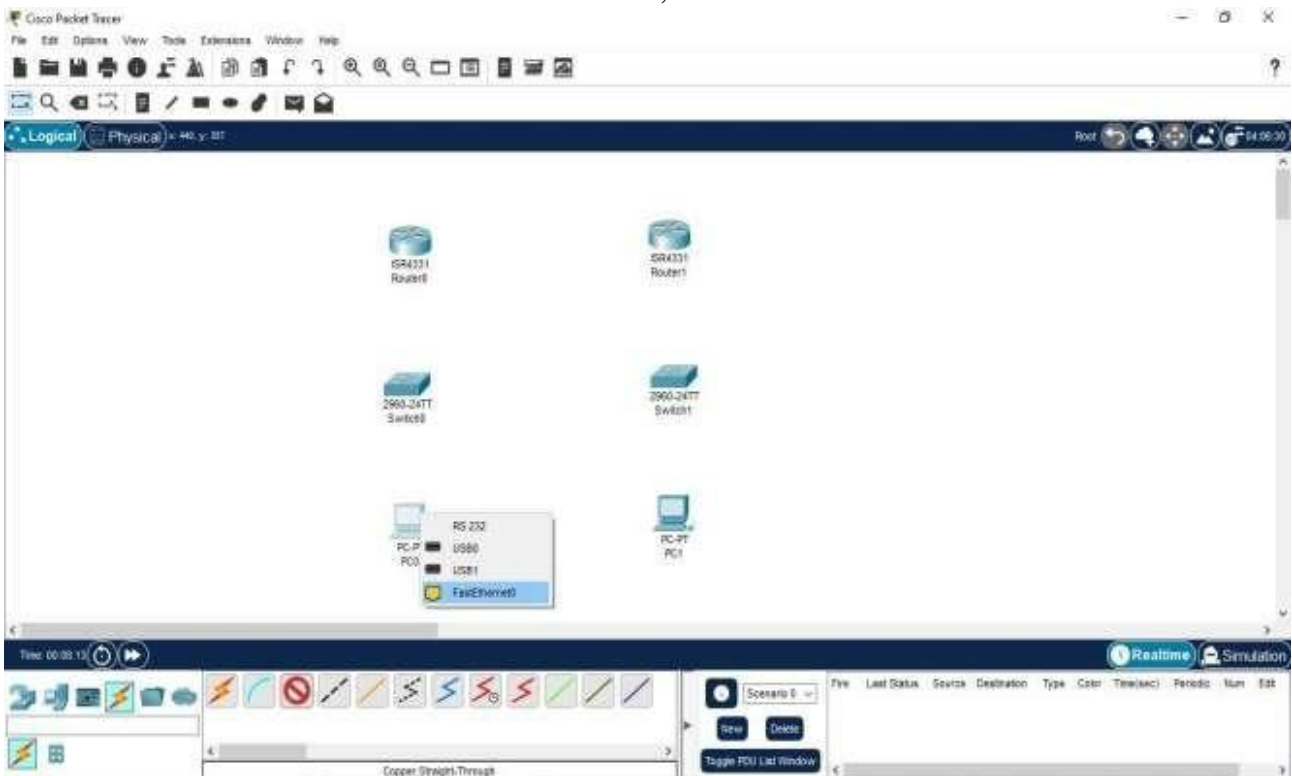
1. Connect routers R1 and R2 via serial or Ethernet.
2. Configure interfaces and IPs:
3. R1(config)# interface Gig0/0
4. R1(config-if)# ip address 10.0.0.1 255.255.255.0
5. R1(config-if)# no shutdown
6. Configure R2 accordingly (10.0.0.2/24).
7. Test: ping 10.0.0.2 from R1.
8. Optionally configure static routes or dynamic routing (e.g., router ospf 1 / network ...).

**Sample output (ping between routers):**

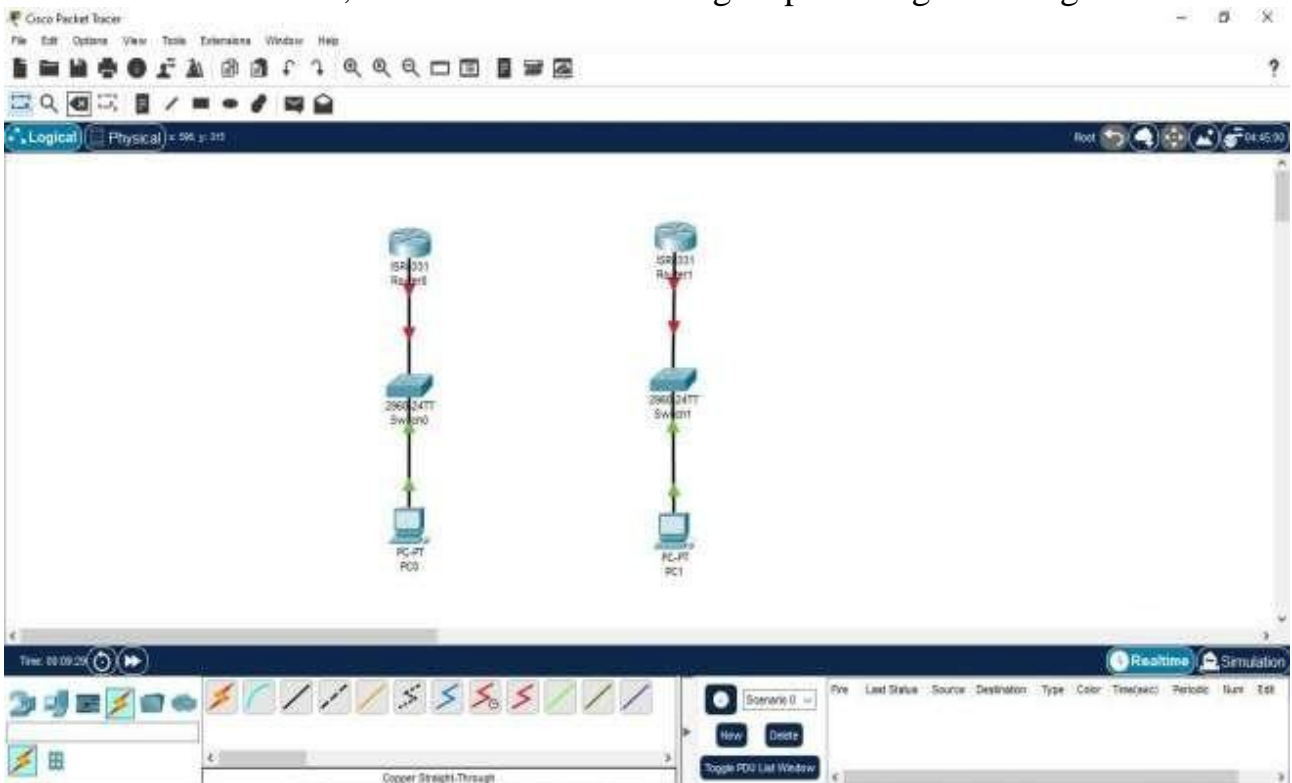
```
R1# ping 10.0.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

## Sample Input & output Screenshot

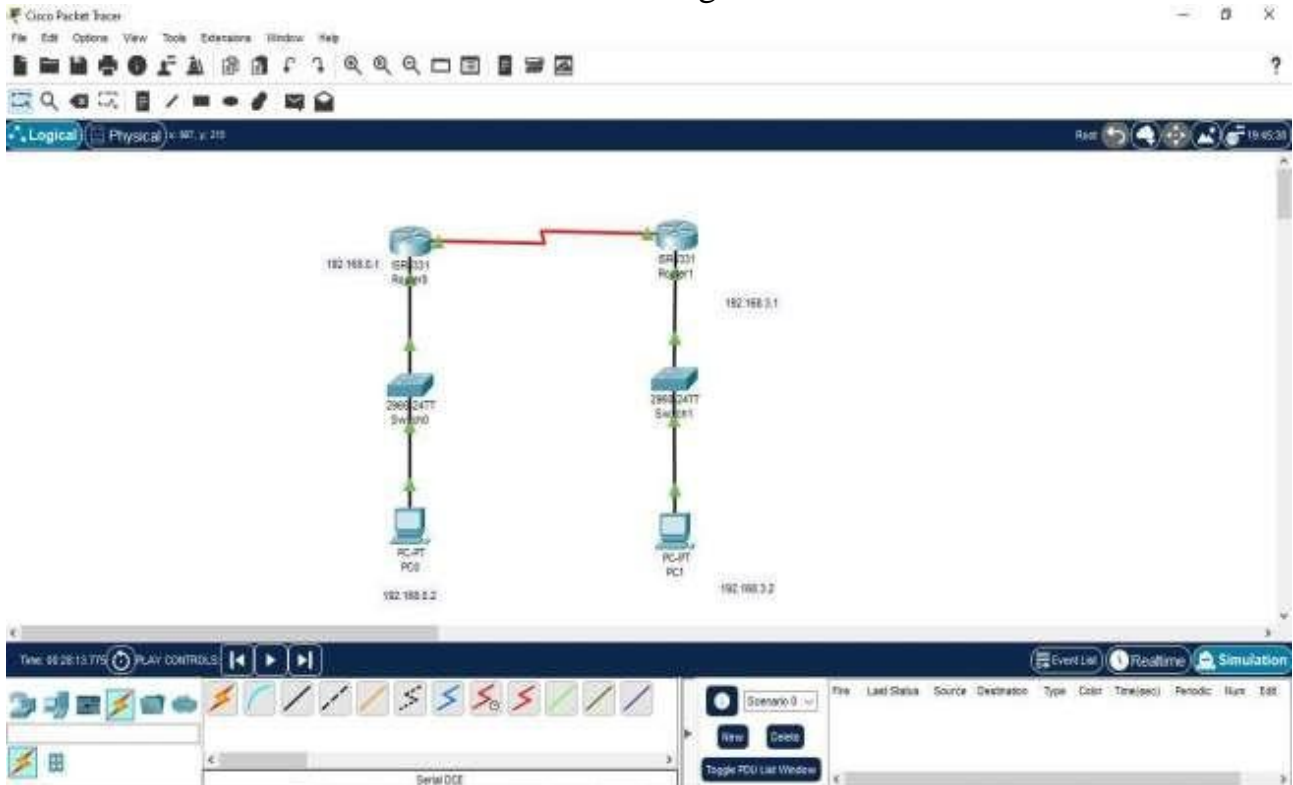
Add or Place Routers, Switches and PC's



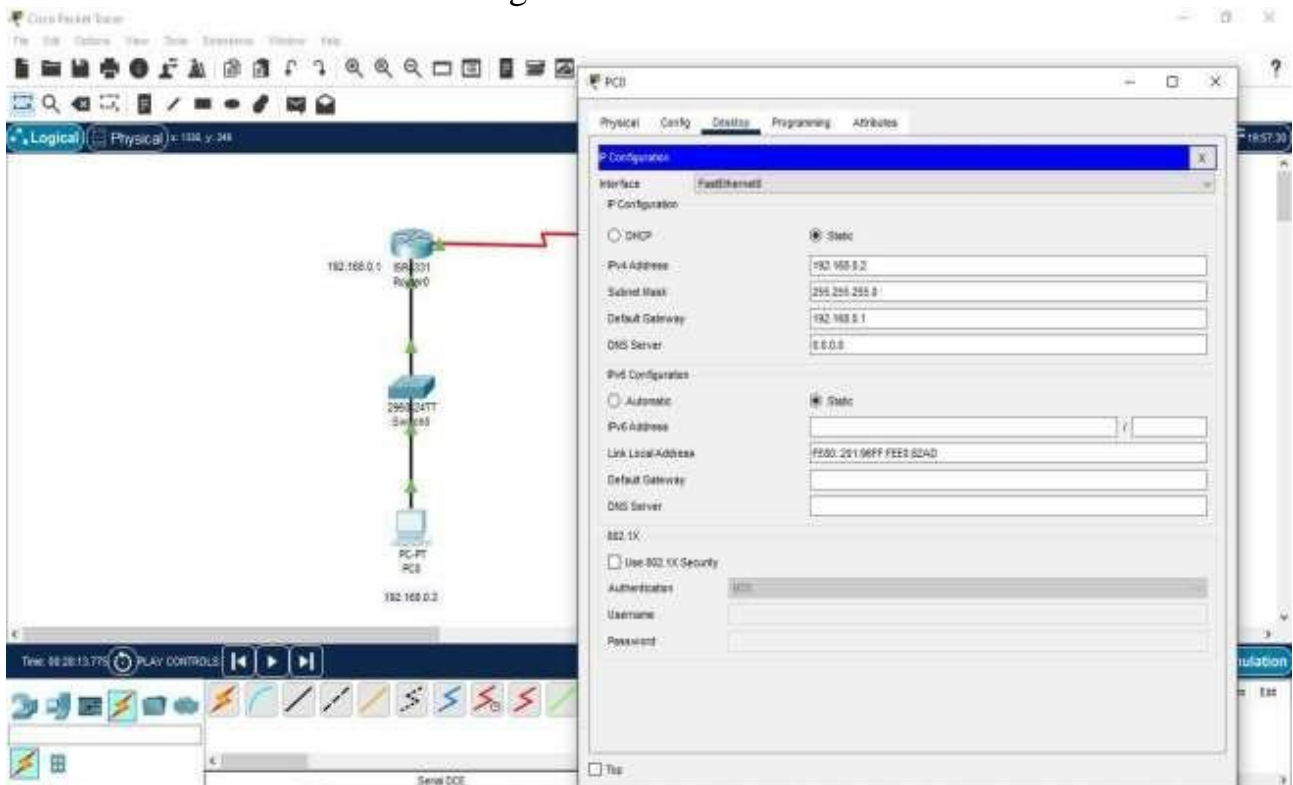
Connect Routers, Switches and PC's using Coper Straight Through Cable:



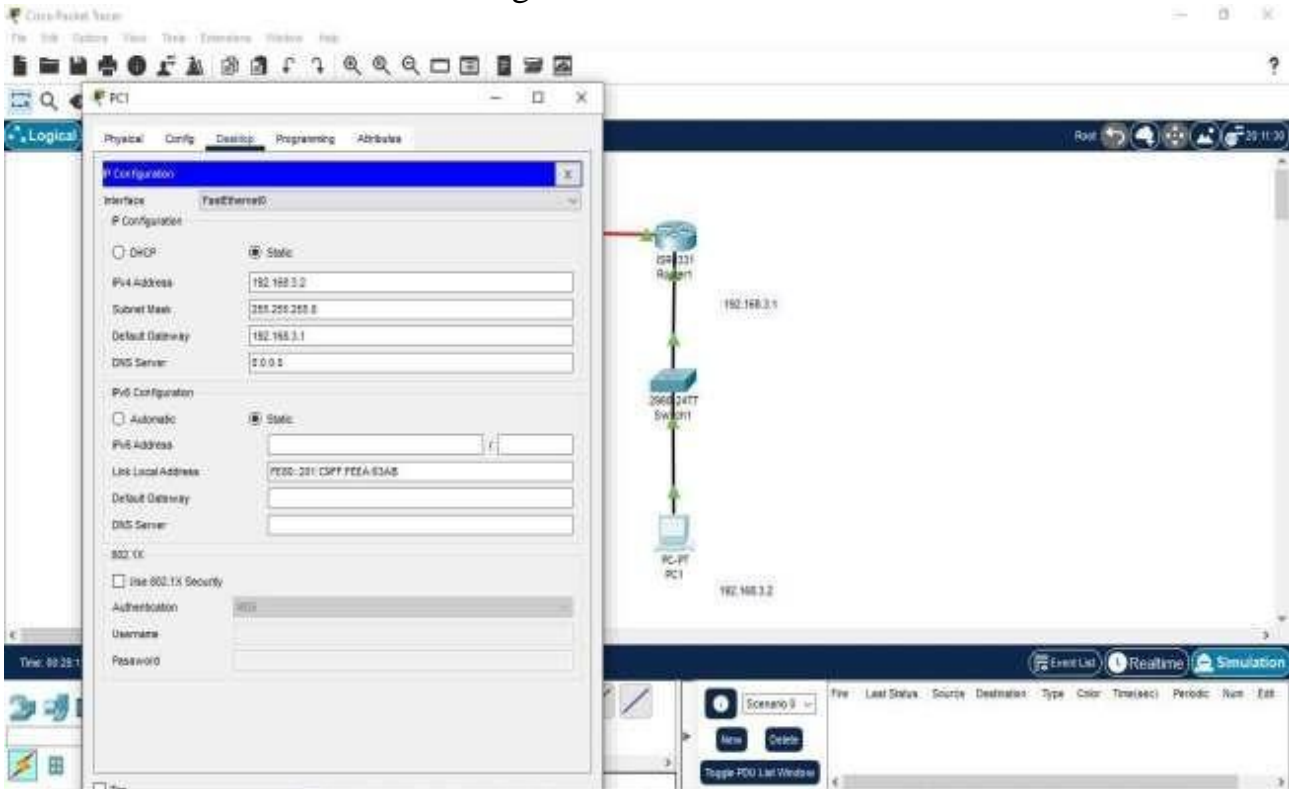
## Connect Between Two Routers using DCE Cable with Serial Port



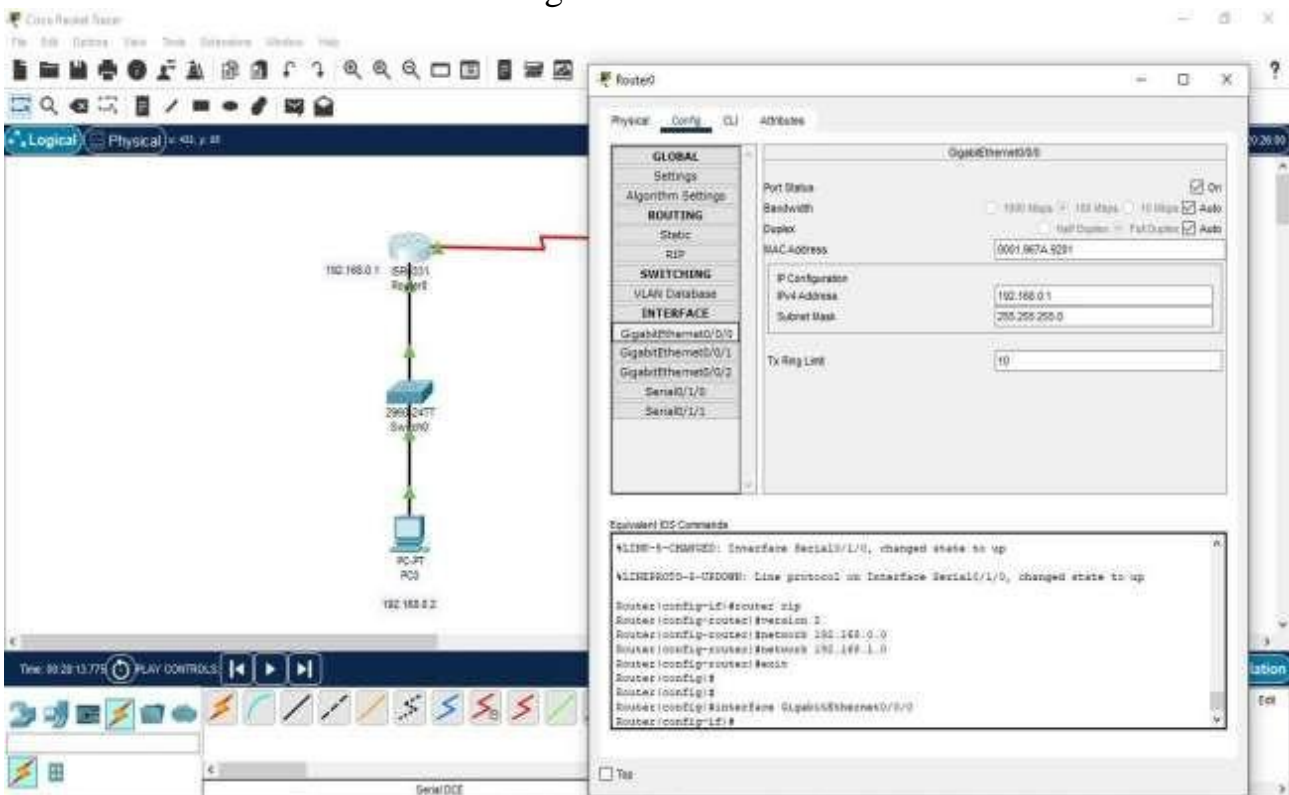
## Assign IP Address for PC0:



## Assign IP Address for PC1



## Assign IP for Router0:



## Assign IP for Router1:

The screenshot displays the Cisco Packet Tracer interface. On the left, a network diagram shows a central Router1 (IP: 192.168.0.1) connected to a 2960-DAT1 Switch, which is in turn connected to a PC-PT PC0 (IP: 192.168.0.2). The main window shows the configuration for Router1. The 'Config' tab is active, and the 'Serial0/1/0' interface is selected. The IP Configuration section shows the IP Address set to 192.168.1.1 and the Subnet Mask set to 255.255.255.0. The 'Equivalent IOS Commands' window at the bottom shows the following configuration commands:

```
Router(config)#router ospf 1
Router(config-router)#network 192.168.0.0
Router(config-router)#network 192.168.1.0
Router(config-router)#exit
Router(config)#
Router(config)#interface GigabitEthernet0/0
Router(config-if)#
Router(config-if)#ip
Router(config-if)#ip address 192.168.1.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#exit
```

## PingPC1fromthePC0UsingIPAddressofPC1:

The screenshot shows the Cisco Packet Tracer interface with a second PC-PT PC1 (IP: 192.168.0.2) added to the network. A 'Command Prompt' window is open on PC1, showing the execution of a ping command to PC0. The output of the command is as follows:

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

Reply from 192.168.0.2: bytes=32 size=64ms TTL=124
Reply from 192.168.0.2: bytes=32 size=10ms TTL=124
Reply from 192.168.0.2: bytes=32 size=10ms TTL=124
Reply from 192.168.0.2: bytes=32 size=10ms TTL=124

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 10ms, Maximum = 64ms, Average = 23ms

C:\>
```

## RESULT:

Thus the Router Protocol commands are successfully tested and executed.

Ex. No. 8	Get the access of the router by connecting with working computer
-----------	--

**Aim:**

Access router CLI from a PC via console, SSH, or Telnet.

**Procedure:**

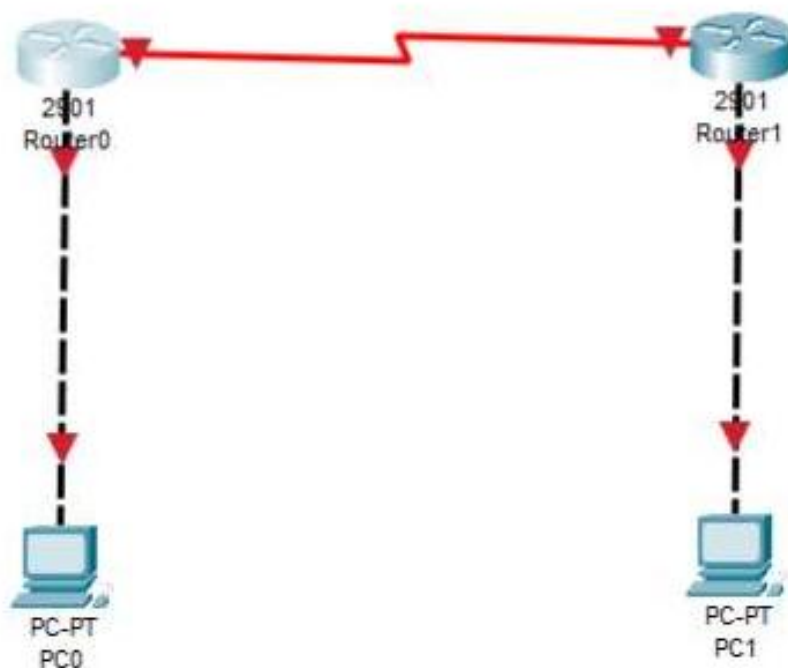
- **Console cable:** Use terminal emulator (PuTTY, minicom) on PC, set baud 9600,8,N,1.
- **SSH/Telnet (networked):** ssh admin@router-ip or telnet router-ip (Telnet insecure).

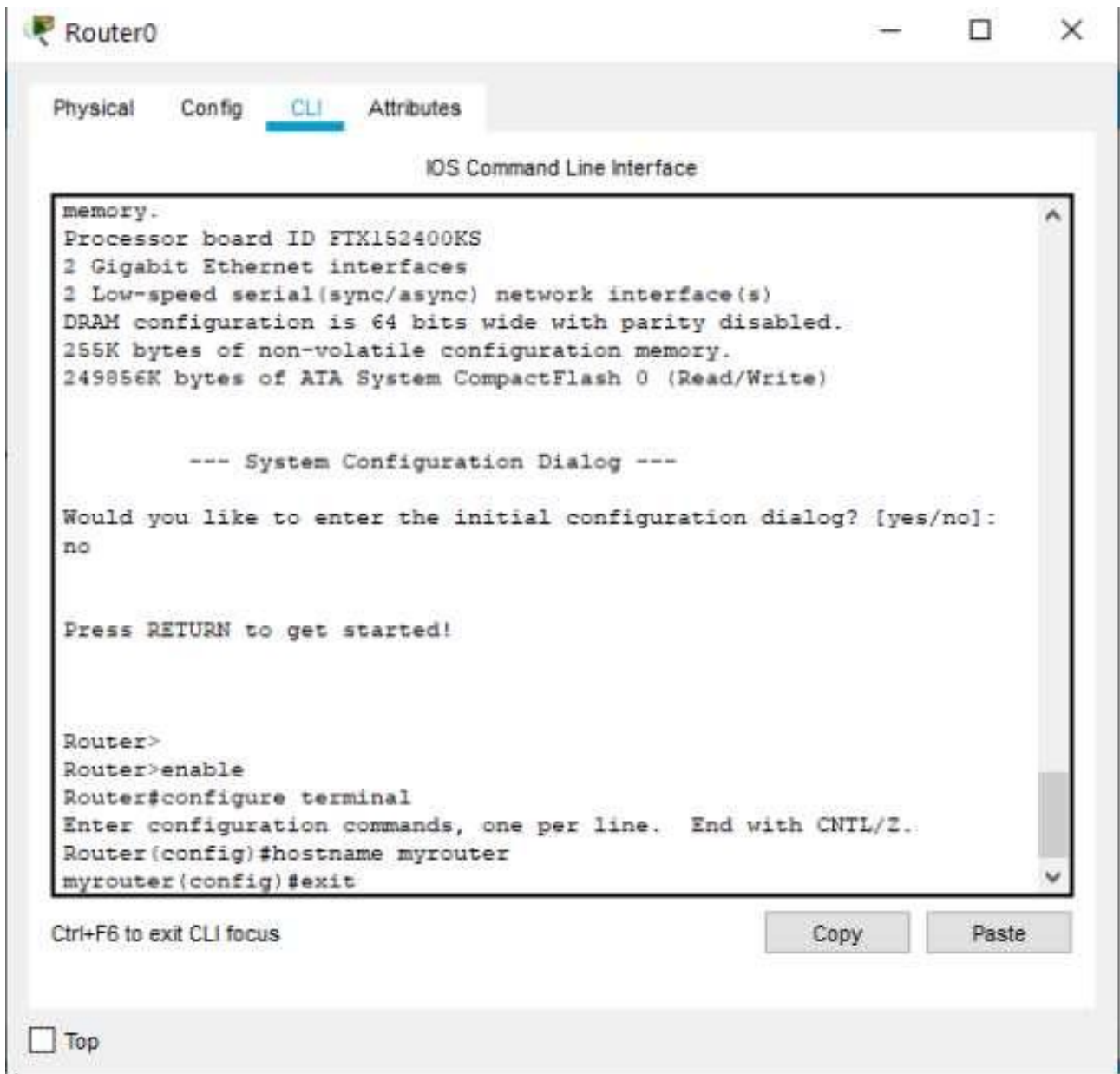
**Sample console connection (login):**

Router con0 is now available  
Press RETURN to get started!

```
Router> enable
Router# configure terminal
Router(config)#
```

**Sample Input & output Screenshot:**





## RESULT:

Thus the Router Protocol commands are successfully tested and executed with Computer System.

Ex. No. 9	Identify the route password of server and get the connection using telnet
-----------	---

**Important safety note:**

Attempting to discover or crack passwords is **unauthorized** and often illegal. No one cannot help with instructions to discover or bypass passwords on devices you do not own or have explicit permission to test. Below is the **safe, permitted** lab version.

**Aim:**

Demonstrate Telnet access to a router/server using a known username/password in a lab environment (or show how to reset/recover access on devices you own).

**Procedure (lab):**

1. Ensure Telnet server enabled on target (for lab only): `sudo apt install xinetd telnetd` (or enable on router). **Note:** Telnet is insecure — prefer SSH.
2. From client: `telnet server-ip` then login with provided lab credentials (e.g., `labuser / labpass`).
3. If you **own** the router and forgot the password, follow vendor's official password recovery procedure (console access + reload + break sequence). Consult router vendor docs.

**Sample Telnet session (lab credentials):**

```
$ telnet 192.168.10.100
Trying 192.168.10.100...
Connected to 192.168.10.100.
Escape character is '^]'.
login: labuser
Password:
Welcome to LabRouter
LabRouter> enable
LabRouter#
```

## Sample Input & output Screenshot:

The screenshot shows a web-based configuration interface for a device named "Router1". The interface has four tabs: "Physical", "Config" (selected), "CLI", and "Attributes". On the left, there is a navigation tree with categories: "GLOBAL" (Settings, Algorithm Settings), "ROUTING" (Static, RIP), "SWITCHING" (VLAN Database), and "INTERFACE" (GigabitEthernet0/0, GigabitEthernet0/1, Serial0/3/0, Serial0/3/1). The "GigabitEthernet0/0" interface is selected, and its configuration is shown in the main area. The configuration includes: Port Status (On), Bandwidth (100 Mbps), Duplex (Full Duplex), MAC Address (00E0.F7BB.CE01), IP Configuration (IP Address: 192.168.1.1, Subnet Mask: 255.255.255.0), and Tx Ring Limit (10). Below the configuration, there is a section for "Equivalent IOS Commands" showing the following commands: 

```
R1(config-if)#exit
R1(config)#interface GigabitEthernet0/0
R1(config-if)#
R1(config-if)#exit
R1(config)#interface GigabitEthernet0/0
R1(config-if)#
```

 At the bottom left, there is a "Top" button.

```
Router1
Physical Config CLI Attributes
IOS Command Line Interface

Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface gigabitEthernet0/0
Router(config-if)#ip address 192.168.1.1 255.255.255.0
Router(config-if)#exit
Router(config)#hostname R1
R1(config)#exit
R1#
%SYS-5-CONFIG_I: Configured from console by console

R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#enable password cisco
R1(config)#line vty 0 4
R1(config-line)#password class
R1(config-line)#login
R1(config-line)#exit
R1(config)#username admin password admin123
R1(config)#line vty 0 4
R1(config-line)#login local
R1(config-line)#transport input telnet
R1(config-line)#exit
R1(config)#exit
R1#
%SYS-5-CONFIG_I: Configured from console by console

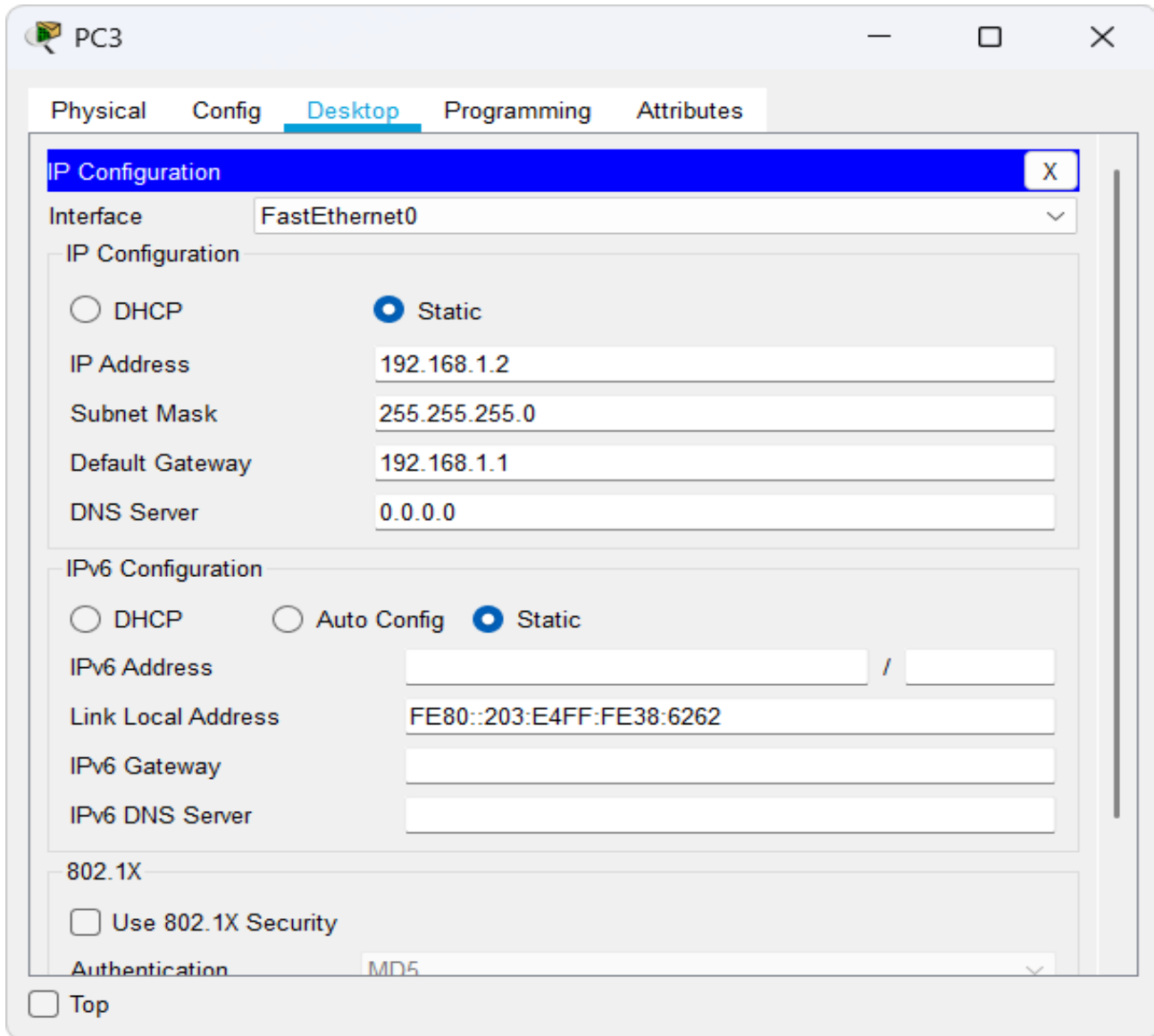
R1#write
Building configuration...
[OK]
R1#write
Building configuration...
[OK]
R1#
R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#interface GigabitEthernet0/0
R1(config-if)#no shutdown
R1(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up

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

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

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

R1(config-if)#exit
R1(config)#interface GigabitEthernet0/0
R1(config-if)#
```



The screenshot shows a Packet Tracer PC4 window with the Desktop tab selected. A Command Prompt window is open, displaying the following text:

```
Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time<lms TTL=255
Reply from 192.168.1.1: bytes=32 time<lms TTL=255
Reply from 192.168.1.1: bytes=32 time<lms TTL=255
Reply from 192.168.1.1: bytes=32 time<lms TTL=255

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>telnet 192.168.1.1
Trying 192.168.1.1 ...Open

User Access Verification

Username: adminUsername:
Password:
R1>enable
Password:
R1#
```

**RESULT:**

Thus the Telnet commends are successfully tested and executed.

Ex. No. 10	Install wireshark for capture and analyse the packets (TCP /UDP)
------------	--

**Aim:**

Capture network traffic and analyze TCP and UDP flows.

**Procedure:**

1. Install Wireshark (Windows installer / sudo apt install wireshark on Debian/Ubuntu).
2. Start Wireshark as Administrator/root or allow non-root capture per OS docs.
3. Choose interface and click Start.
4. Use capture filter (optional): tcp, udp, host 192.168.1.50, or port 80.
5. Stop capture and analyze: follow TCP stream (right-click → Follow → TCP Stream), inspect headers and flags, use display filters like tcp.port==80 && ip.addr==93.184.216.34.

**Sample Wireshark:**

Frame 124: 74 bytes on wire (592 bits), 74 bytes captured  
Ethernet II, Src: 00:1a:2b:3c:4d:5e, Dst: 00:0c:29:6d:8e:2a  
Internet Protocol Version 4, Src: 192.168.1.50, Dst: 93.184.216.34  
Transmission Control Protocol, Src Port: 54321, Dst Port: 80, Seq: 1, Ack: 1, Flags [S, ACK]

**Tip:**

Save capture as .pcap and open later or import into analysis tools.

## Sample Input & output Screenshot:

### Capturing and Analyzing WIFI signals:

The screenshot displays two windows. The left window is Wireshark, titled 'Capturing from Wi-Fi'. It shows a list of captured packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. The selected packet (No. 1) is a Transmission Control Protocol (TCP) packet with the following details:

- Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface {Device\NPF...}
- Ethernet II, Src: LiteonTechno\_41:51:e5 (8c:91:80:61:51:e5), Dst: LiteonTechno\_61:51:e5 (8c:91:80:61:51:e5)
- Internet Protocol Version 6, Src: 192.168.43.1, Dst: 2404:6800:4009:825::2004
- Transmission Control Protocol, Src Port: 443, Dst Port: 49224, Seq: 1, Ack: 1, Len: 0

The right window is a Command Prompt showing the execution of a ping command to www.google.com. The output shows the following statistics:

```

C:\Users\Sures>ping www.google.com

Pinging www.google.com [2404:6800:4009:820::2004] with 32 bytes of data:
Reply from 2404:6800:4009:820::2004: time=212ms
Reply from 2404:6800:4009:820::2004: time=172ms
Reply from 2404:6800:4009:820::2004: time=125ms
Reply from 2404:6800:4009:820::2004: time=87ms

Ping statistics for 2404:6800:4009:820::2004:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 87ms, Maximum = 212ms, Average = 149ms

C:\Users\Sures>ping www.google.com

Pinging www.google.com [2404:6800:4009:825::2004] with 32 bytes of data:
Reply from 2404:6800:4009:825::2004: time=180ms
Reply from 2404:6800:4009:825::2004: time=133ms
Reply from 2404:6800:4009:825::2004: time=87ms
Reply from 2404:6800:4009:825::2004: time=28ms

Ping statistics for 2404:6800:4009:825::2004:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 87ms, Maximum = 200ms, Average = 170ms

C:\Users\Sures>
    
```

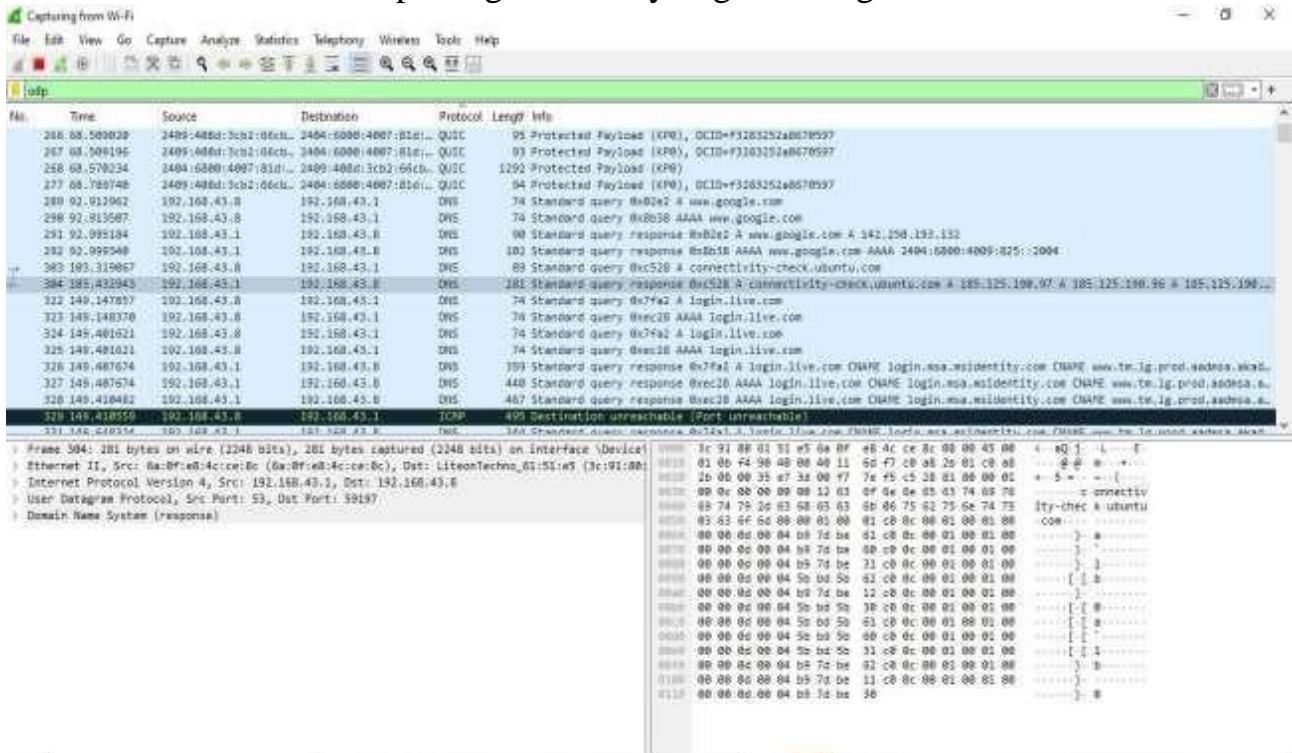
### Analyzing TCP Signals:

The screenshot shows Wireshark with a filter applied: 'tcp.port==80 | udp.port==80'. The selected packet (No. 386) is a Hypertext Transfer Protocol (HTTP) GET request. The details pane shows:

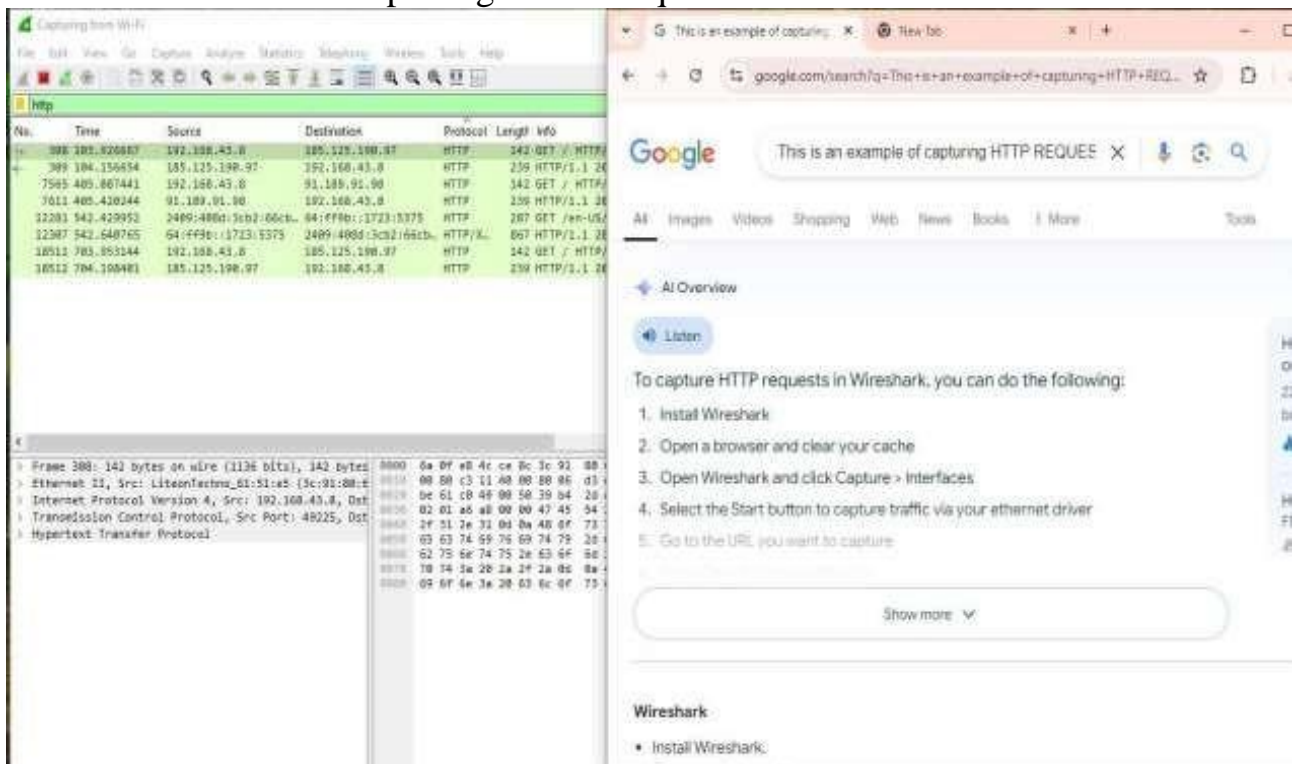
- Frame 386: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface {Device\NPF...}
- Ethernet II, Src: LiteonTechno\_41:51:e5 (8c:91:80:61:51:e5), Dst: LiteonTechno\_61:51:e5 (8c:91:80:61:51:e5)
- Internet Protocol Version 4, Src: 192.168.43.8, Dst: 185.125.190.97
- Transmission Control Protocol, Src Port: 49225, Dst Port: 80, Seq: 1, Ack: 1, Len: 88
- Hypertext Transfer Protocol

The packet bytes pane shows the raw data of the HTTP request, including the GET method and various headers like User-Agent and Accept.

## Capturing and Analyzing UDP Signals:



## Capturing HTTP Request from Browser:



## RESULT:

Thus the TCP/UDP Protocol analysis are successfully tested and executed using wireshark.

**Reference Books:**

- Nathan J. Muller, Network Manager's Handbook, 1st Edition, McGraw-Hill Professional, 2002, ISBN-10: 0071405674, ISBN-13: 978-0071405676.

**Website References:**

- Netlab: a Virtual Networking Labbing Tool. <https://netlab.tools/>