

Experiment No : 01

.....

Roll No : TE-43

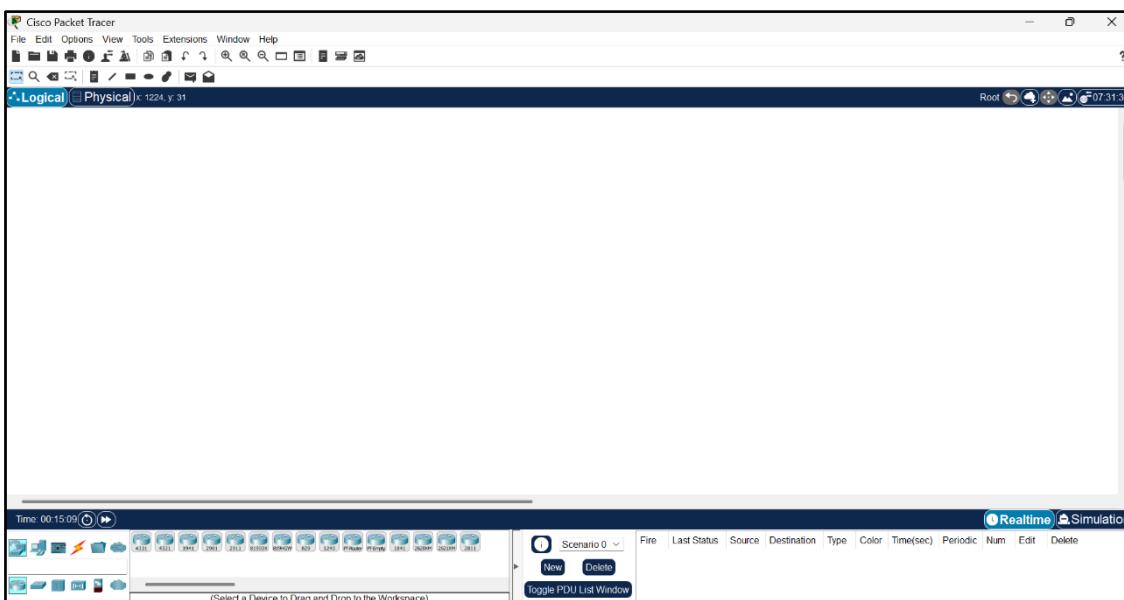
Name : Sanika Nilesh Patil

Title : Demonstrate the different types of topologies and types of transmission media by using a packet tracer tool.

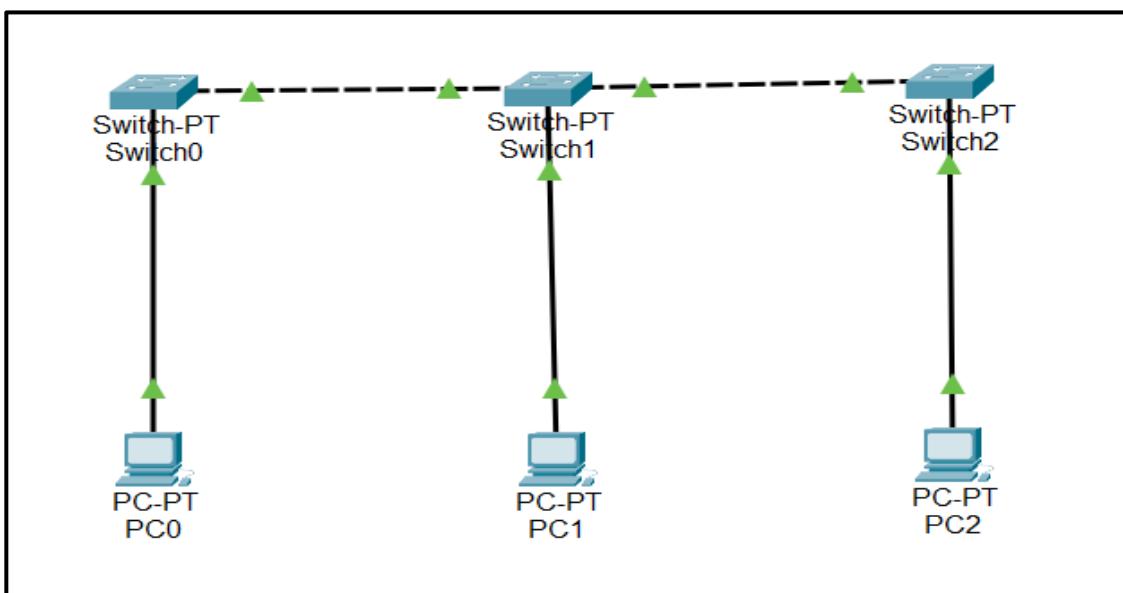
.....

Bus Topology :

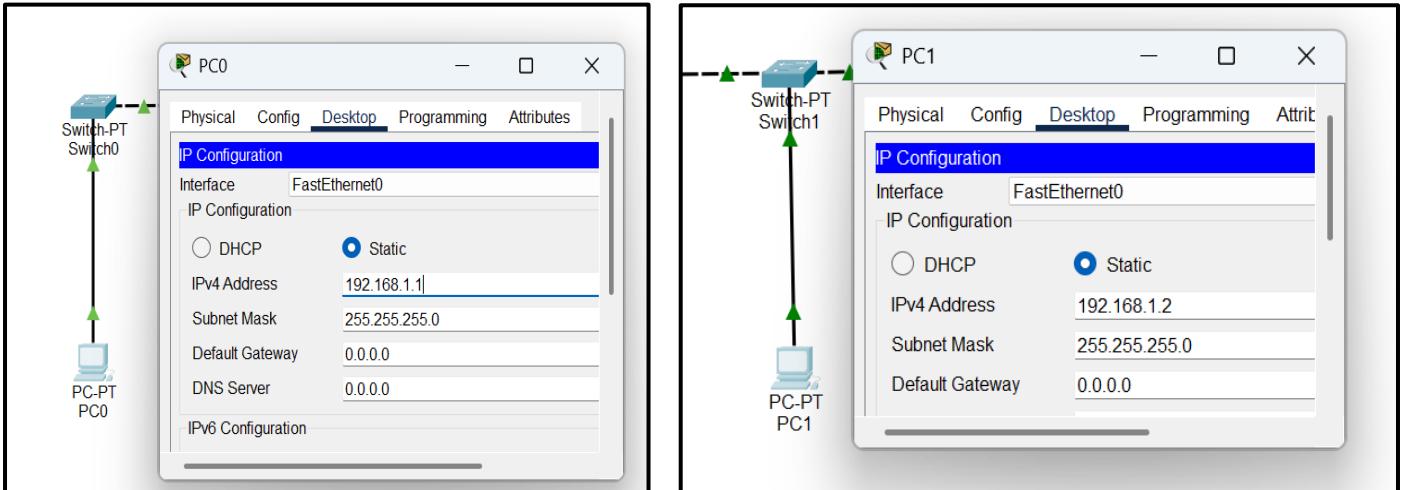
Step 1: Launch Packet Tracer.



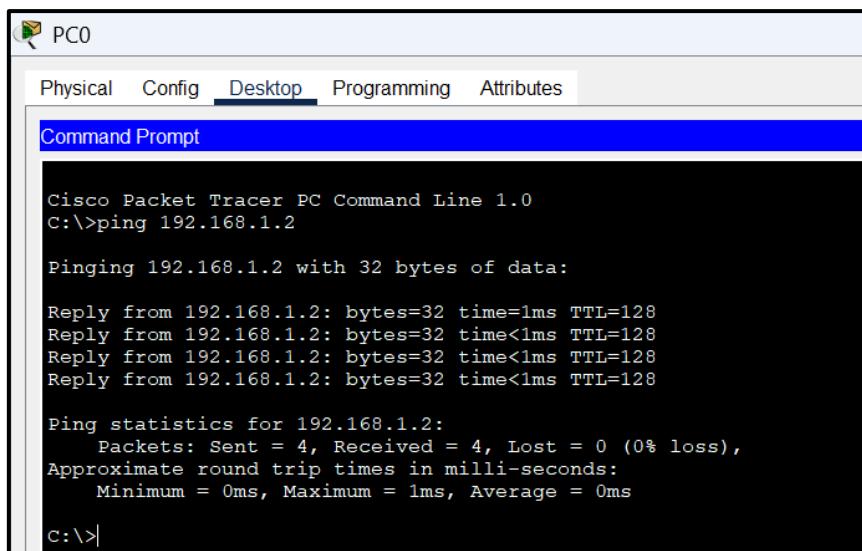
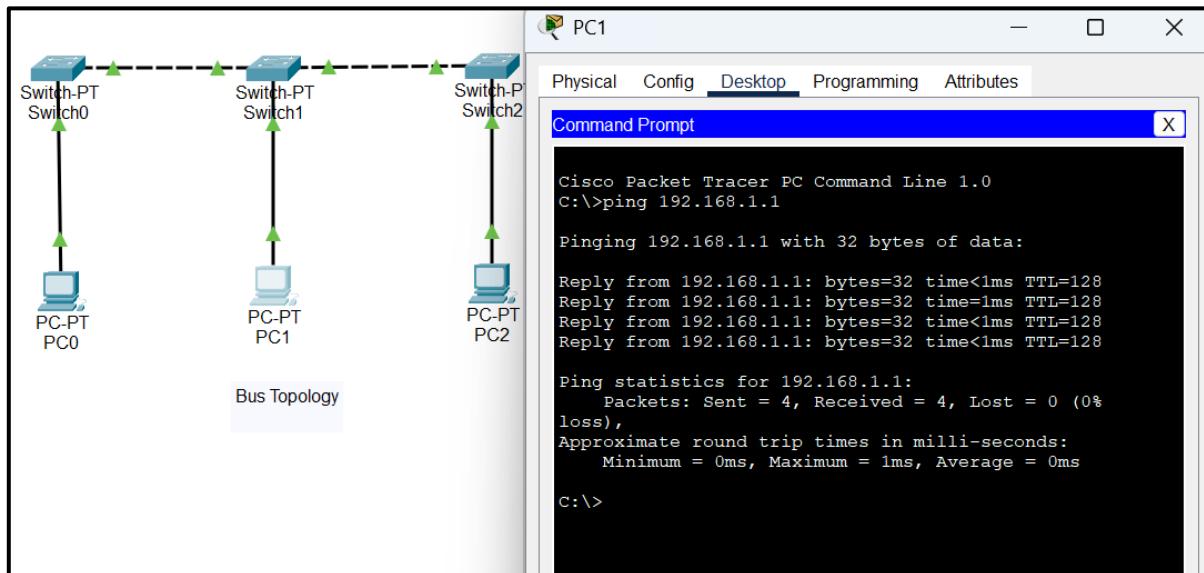
Step 2: Build bus topology.



Step 3: Configure IP addresses and subnet masks on all PCs.

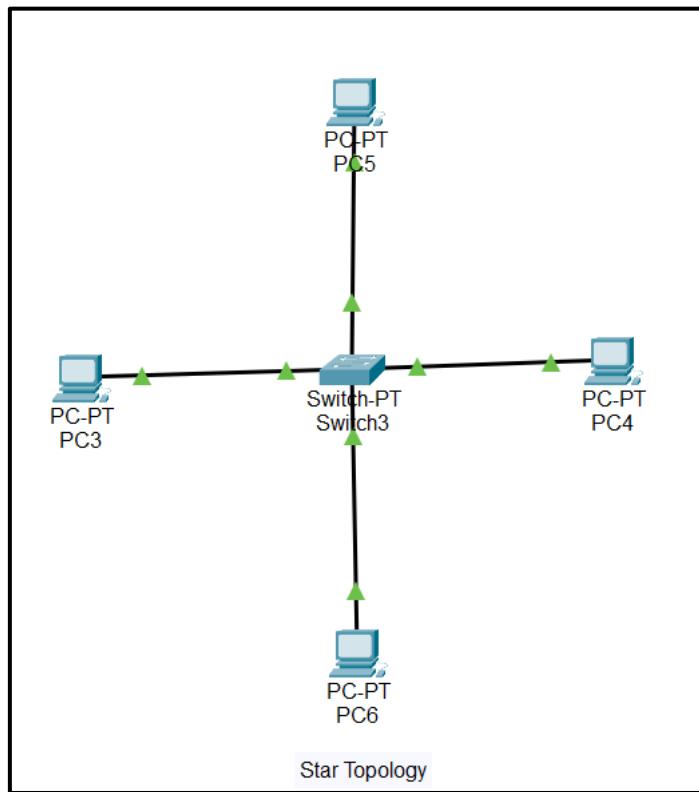


Step 4: Test network connectivity using the PING command between PCs.

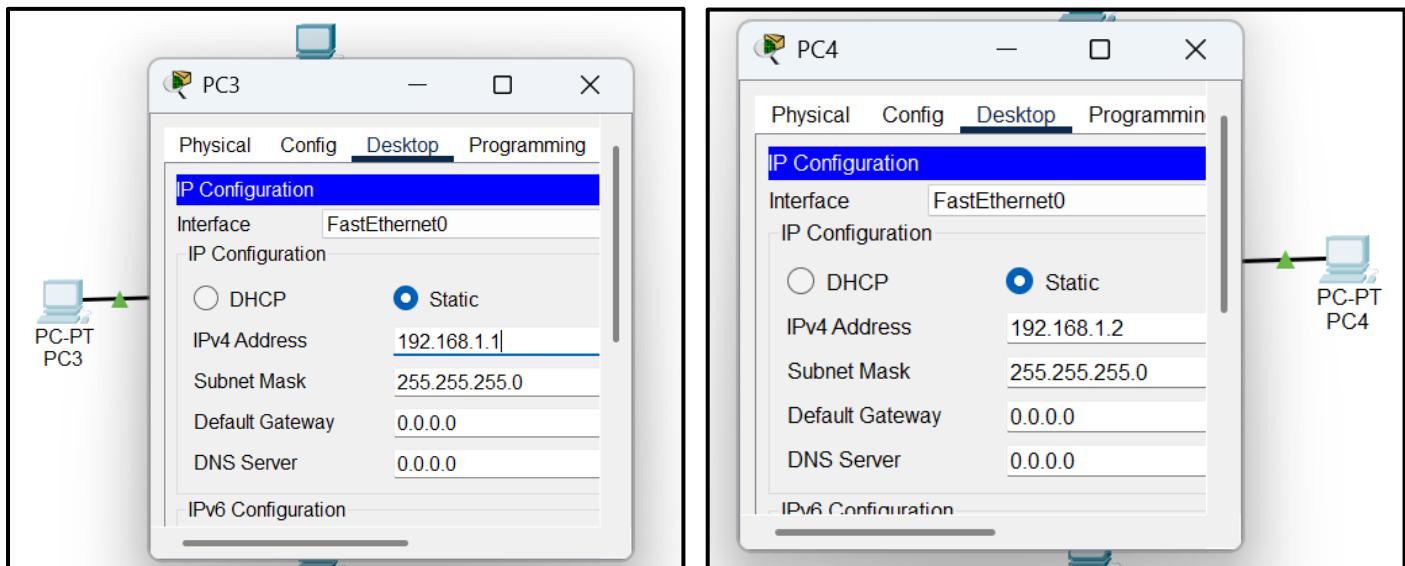


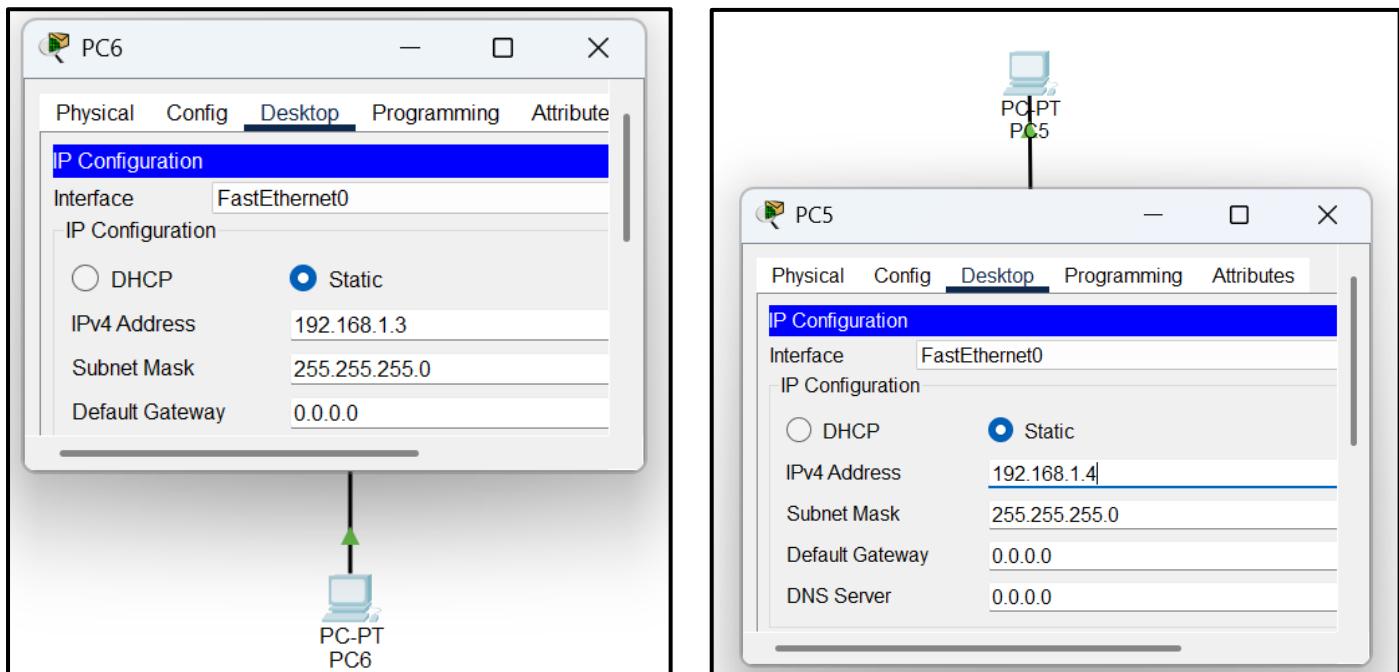
Star Topology :

Step 1: Build star topology.



Step 2: Configure IP addresses and subnet masks on all PCs.





Step 3: Test network connectivity using the PING command between PCs.

```

Cisco Packet Tracer PC Command Line 1.0
C:>ping 192.168.1.4

Pinging 192.168.1.4 with 32 bytes of data:

Reply from 192.168.1.4: bytes=32 time=4ms TTL=128
Reply from 192.168.1.4: bytes=32 time=5ms TTL=128
Reply from 192.168.1.4: bytes=32 time=8ms TTL=128
Reply from 192.168.1.4: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 8ms, Average = 4ms

C:>192.168.1.3
Invalid Command.

C:>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:

Reply from 192.168.1.3: bytes=32 time<1ms TTL=128

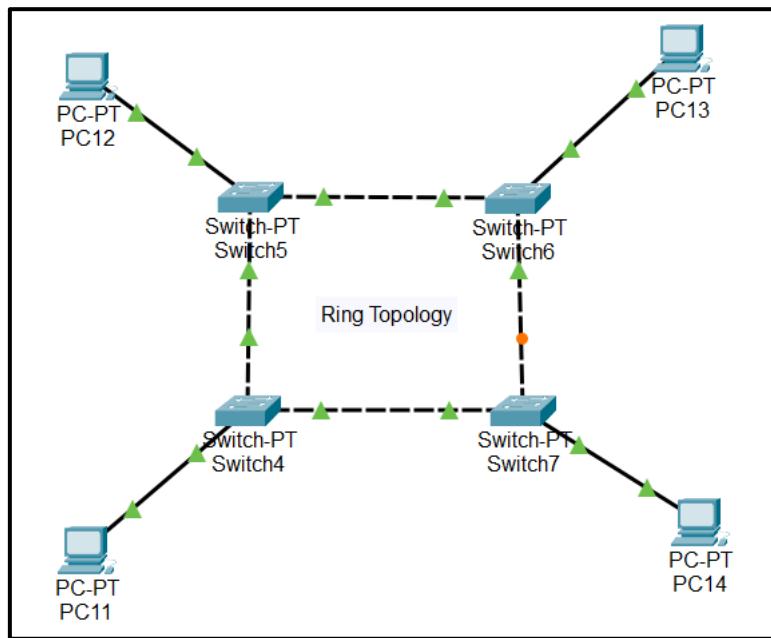
Ping statistics for 192.168.1.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:>

```

Ring Topology :

Step 1: Build Ring topology.



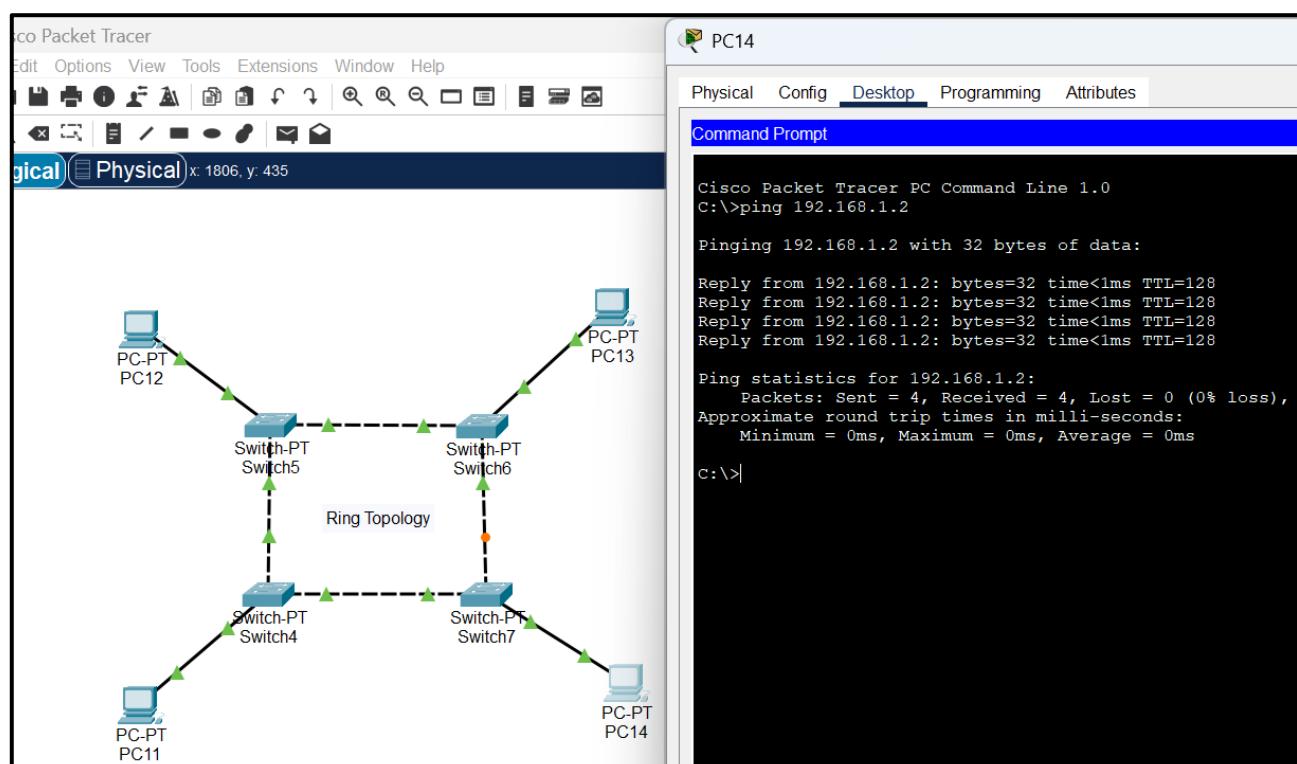
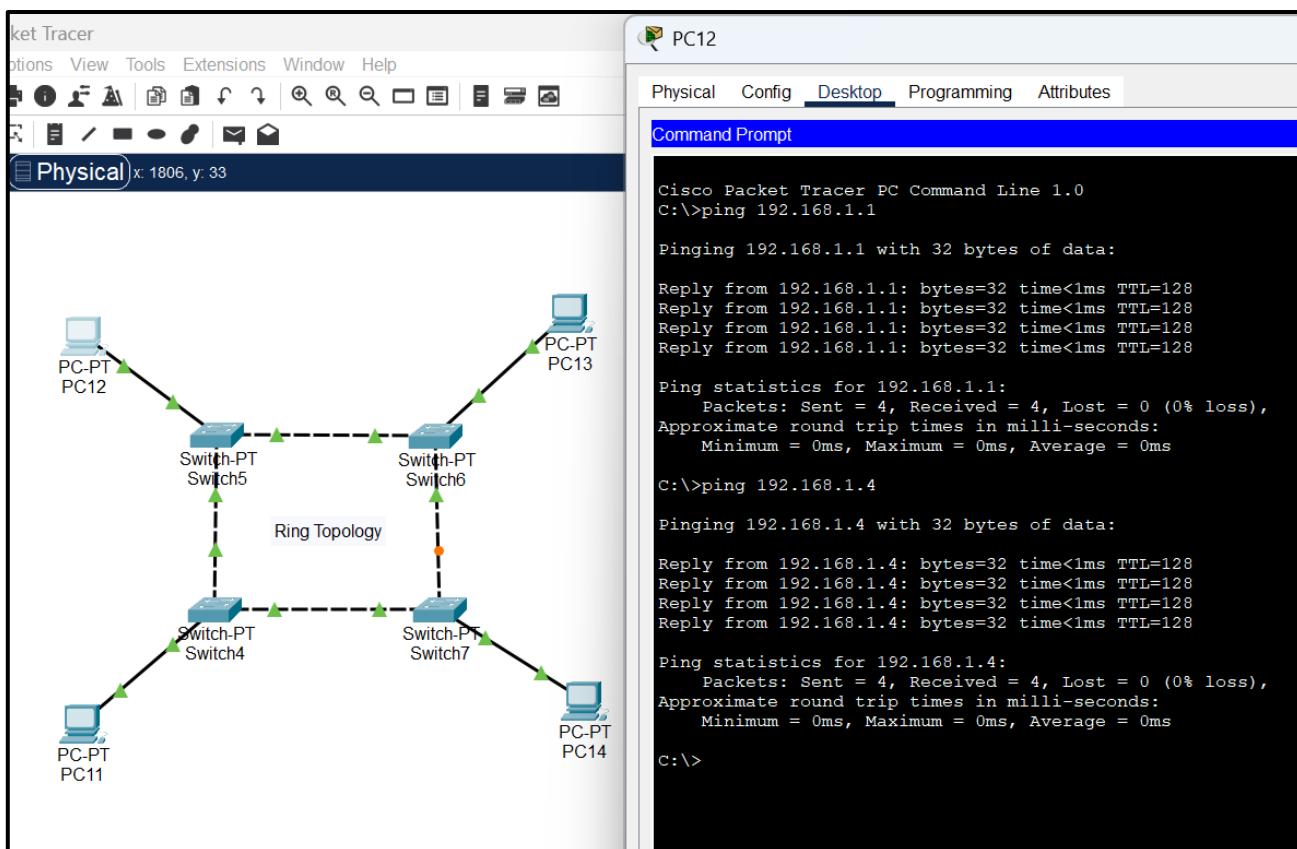
Step 2: Configure IP addresses and subnet masks on all PCs.

The image displays four separate windows, each representing a different PC in the ring topology. Each window shows the "IP Configuration" tab of a software interface, likely a network configuration tool. The PCs are labeled PC12, PC13, PC14, and PC11 from top-left to bottom-right.

- PC12:** IP Address: 192.168.1.2, Subnet Mask: 255.255.255.0
- PC13:** IP Address: 192.168.1.1, Subnet Mask: 255.255.255.0
- PC14:** IP Address: 192.168.1.3, Subnet Mask: 255.255.255.0
- PC11:** IP Address: 192.168.1.4, Subnet Mask: 255.255.255.0

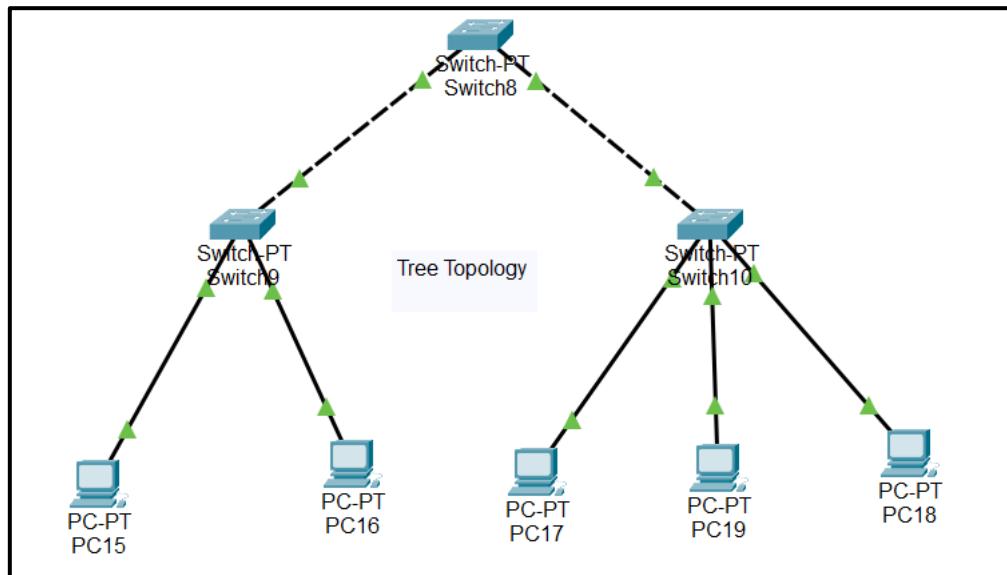
In all four configurations, the "Interface" is set to "FastEthernet0" and the "IP Configuration" method is set to "Static". The "Default Gateway" field is set to "0.0.0.0" and the "DNS Server" field is also set to "0.0.0.0". The "Config" tab is selected in all windows, while the other tabs (Physical, Desktop, Programming, Attr) are visible but not active.

Step 3: Test network connectivity using the PING command between PCs.

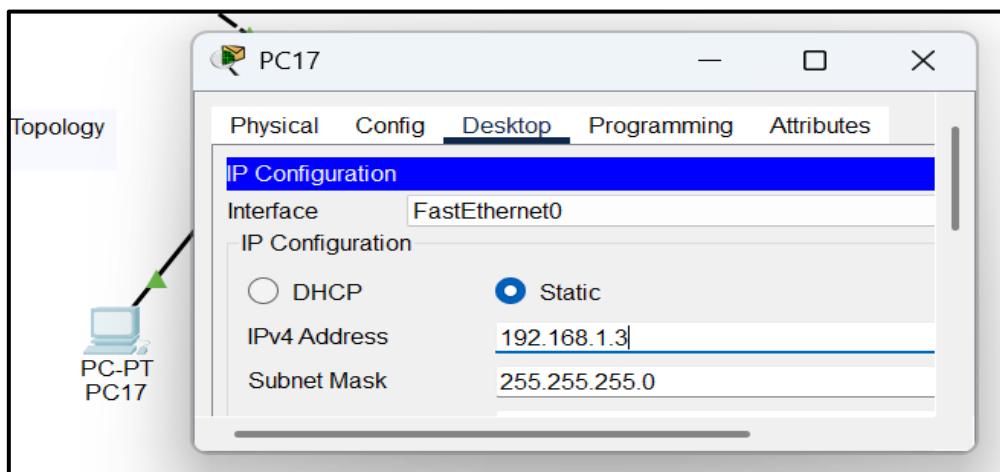
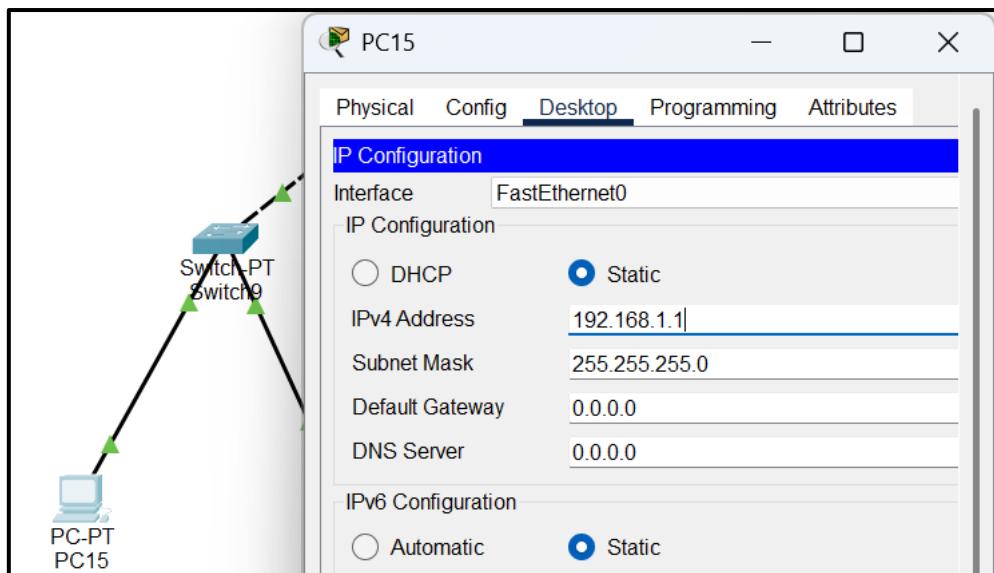


Tree Topology :

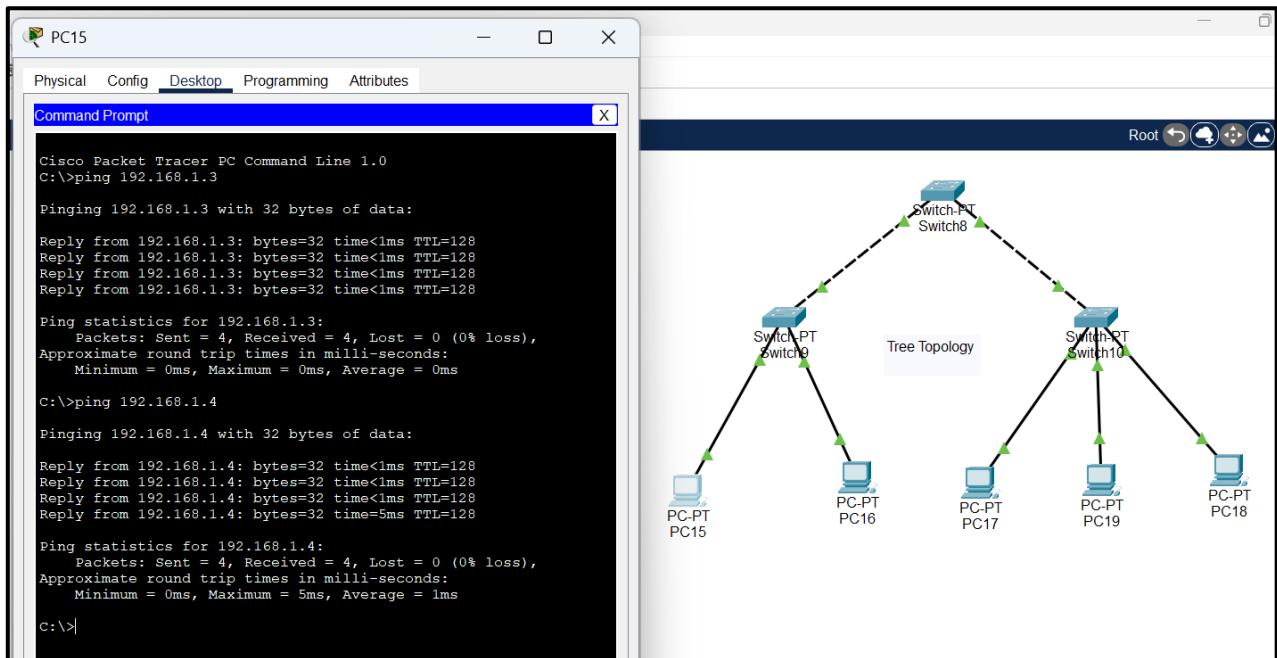
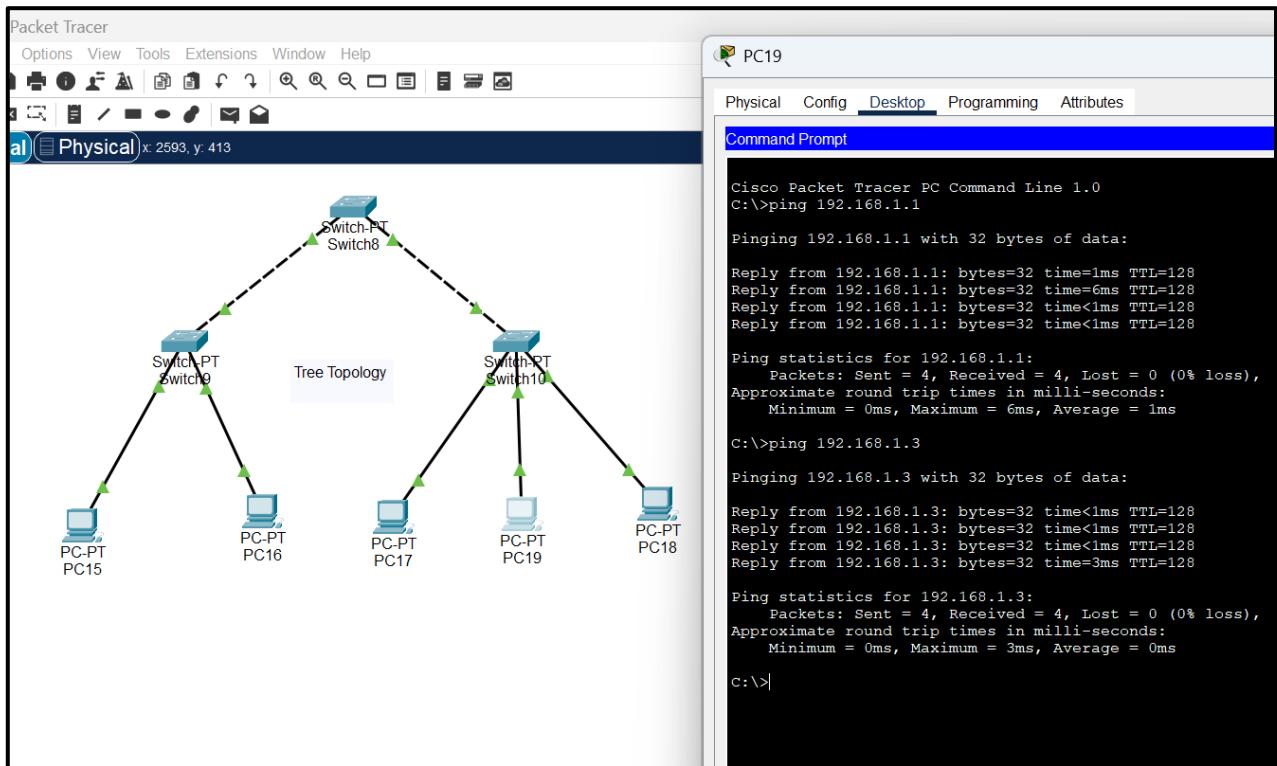
Step 1: Build Tree topology.



Step 2: Configure IP addresses and subnet masks on all PCs.

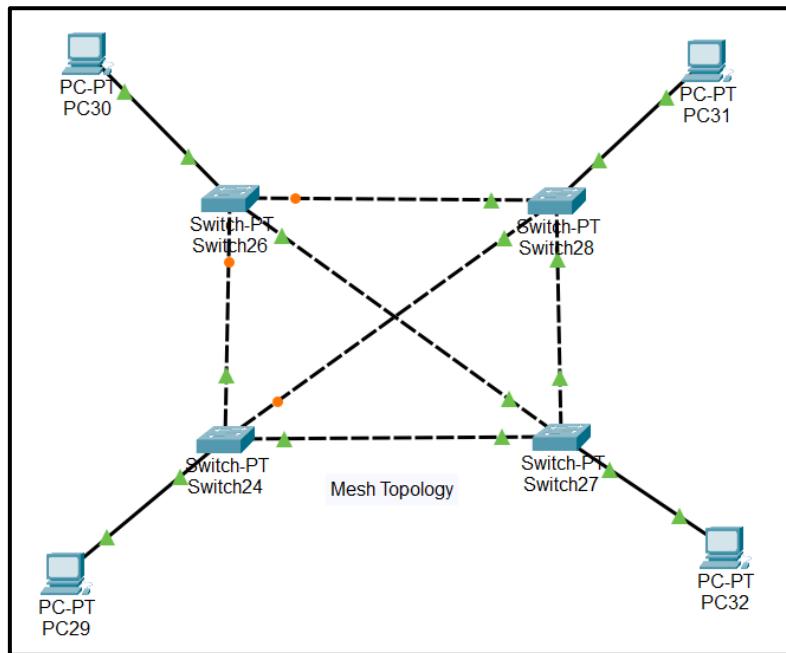


Step 3: Test network connectivity using the PING command between PCs.



Mesh Topology :

Step 1: Build Mesh topology.



Step 2: Configure IP addresses and subnet masks on all PCs.

The image shows two windows from a network configuration tool. The top window is for PC30 and the bottom window is for PC32. Both windows have tabs for Physical, Config, Desktop, Programming, and Attributes. The Desktop tab is selected, showing the IP Configuration for FastEthernet0 interface. For PC30, the configuration is:

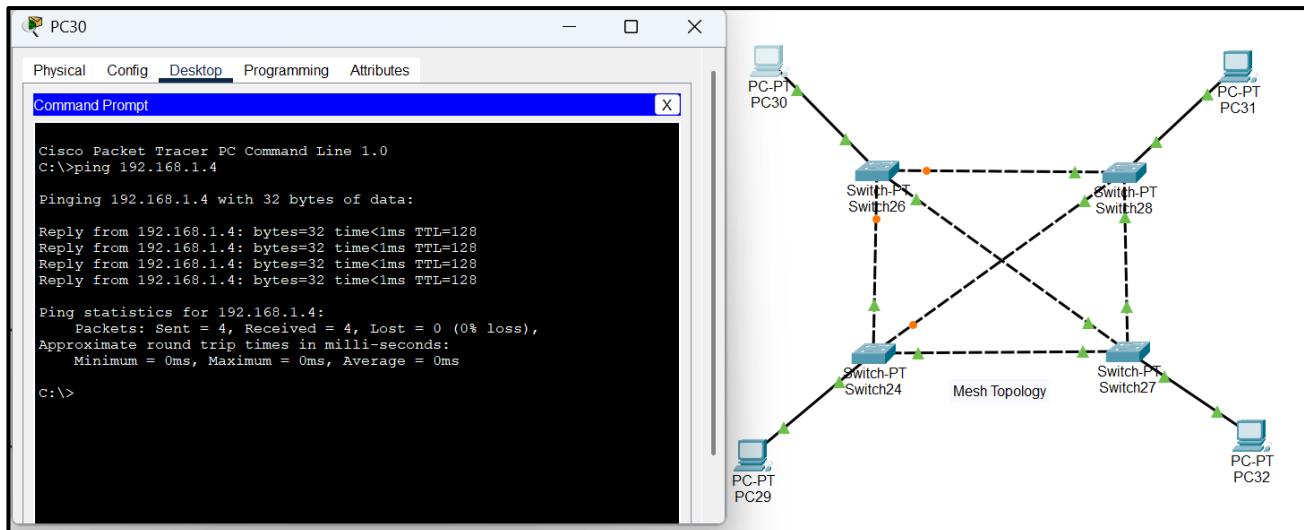
Interface	FastEthernet0
IP Configuration	<input type="radio"/> DHCP <input checked="" type="radio"/> Static
IPv4 Address	192.168.1.1
Subnet Mask	255.255.255.0
Default Gateway	0.0.0.0
DNS Server	0.0.0.0

For PC32, the configuration is:

Interface	FastEthernet0
IP Configuration	<input type="radio"/> DHCP <input checked="" type="radio"/> Static
IPv4 Address	192.168.1.4
Subnet Mask	255.255.255.0
Default Gateway	0.0.0.0
DNS Server	0.0.0.0

The background of the windows shows the network topology from the previous diagram, with the respective PCs highlighted.

Step 3: Test network connectivity using the PING command between PCs.



The screenshot shows the Cisco Packet Tracer interface. On the left, a window titled 'PC32' displays a Command Prompt session. The user has run two ping commands: one to 192.168.1.1 and one to 192.168.1.2. Both pings were successful with 0% loss. The topology on the right is identical to the first screenshot, showing a mesh network with six switches and four hosts. The network is labeled 'Mesh Topology'. A legend indicates that solid lines represent bidirectional links and dashed lines represent unidirectional links.

```
Cisco 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<1ms TTL=128

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:\>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

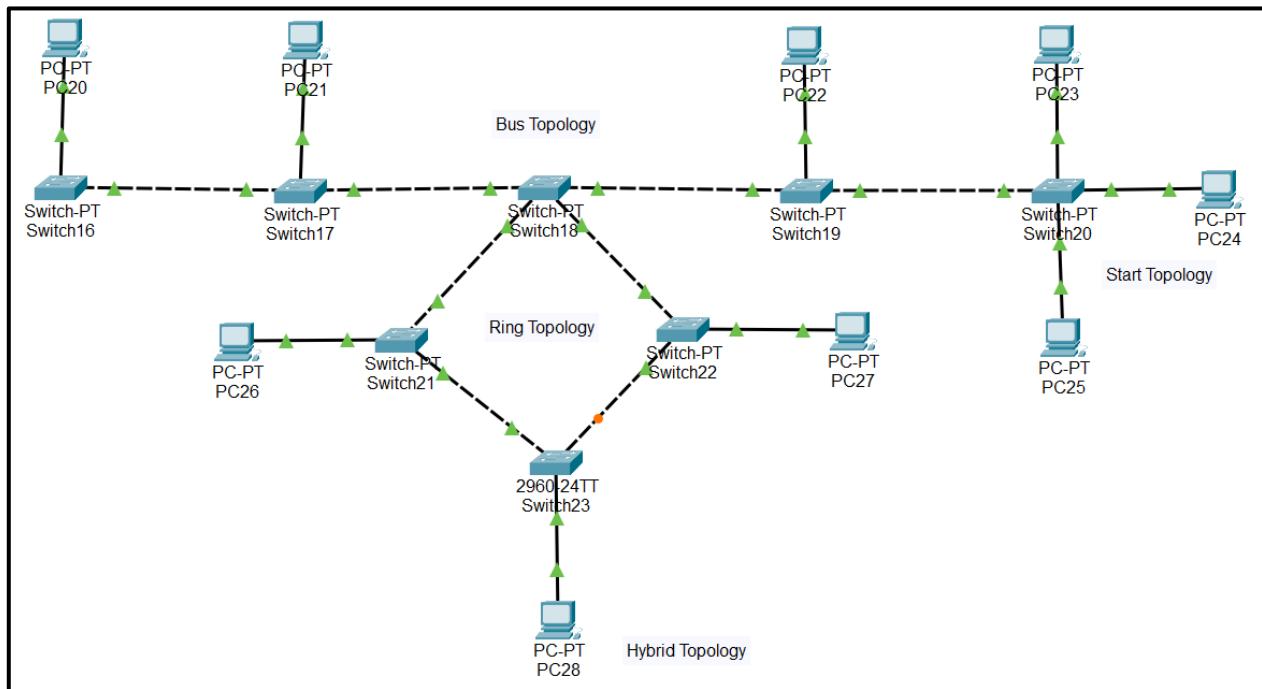
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

Hybrid Topology :

Step 1: Build Hybrid topology.



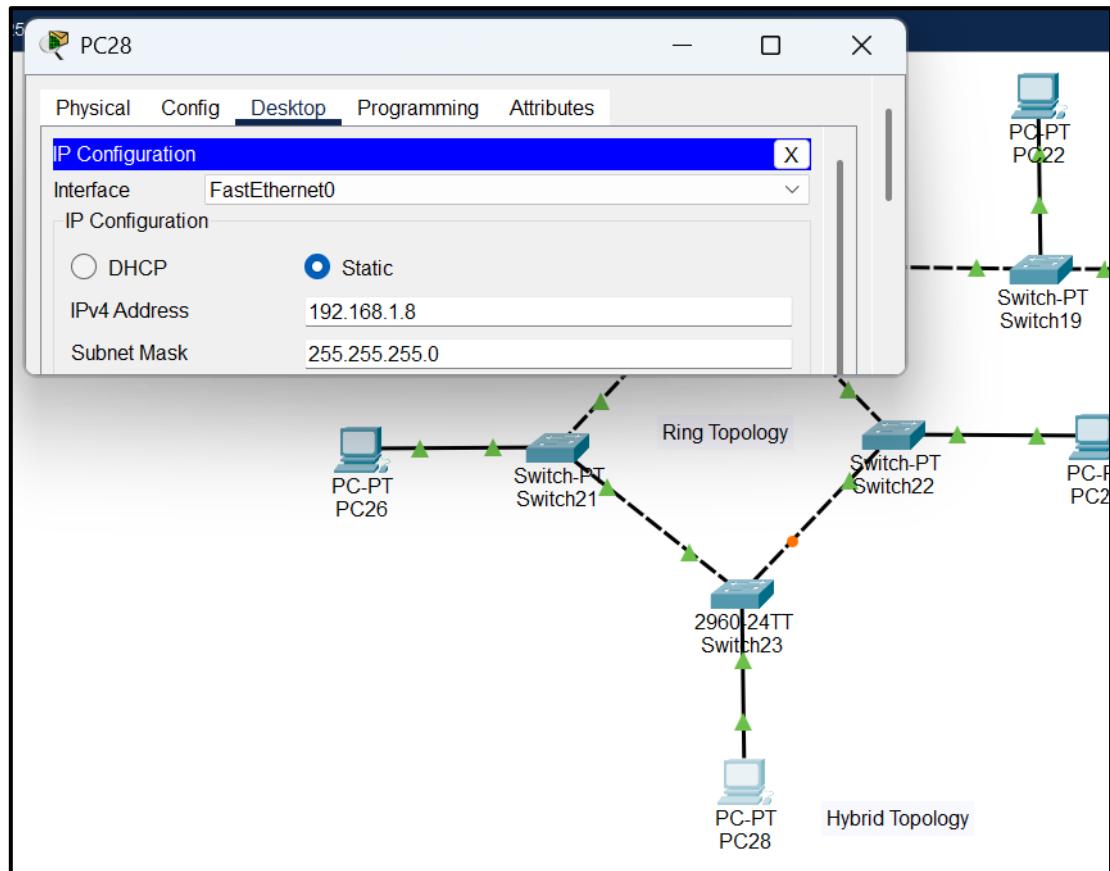
Step 2: Configure IP addresses and subnet masks on all PCs.

PC20

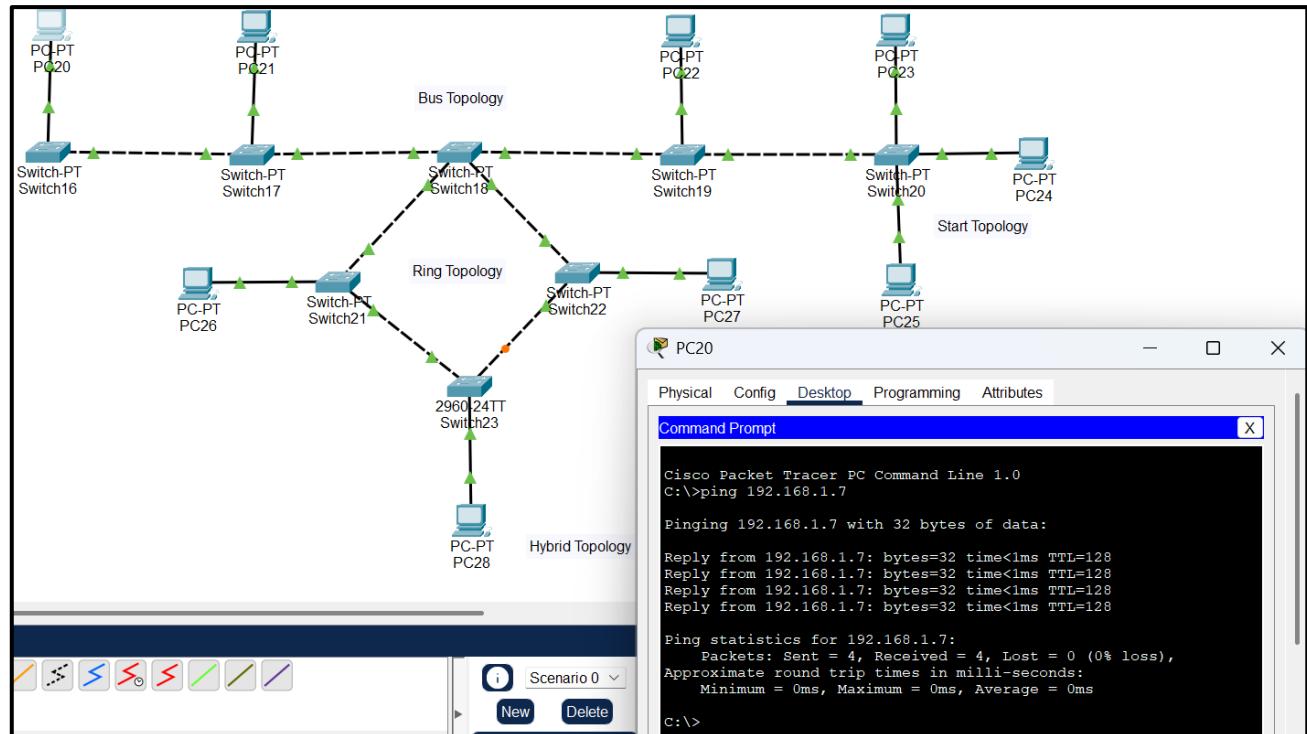
IP Configuration	
Interface	FastEthernet0
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static
IPv4 Address	192.168.1.1
Subnet Mask	255.255.255.0
Default Gateway	0.0.0.0
DNS Server	0.0.0.0

PC25

IP Configuration	
Interface	FastEthernet0
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static
IPv4 Address	192.168.1.6
Subnet Mask	255.255.255.0
Default Gateway	0.0.0.0
DNS Server	0.0.0.0



Step 3: Test network connectivity using the PING command between PCs.



Experiment No : 02

.....

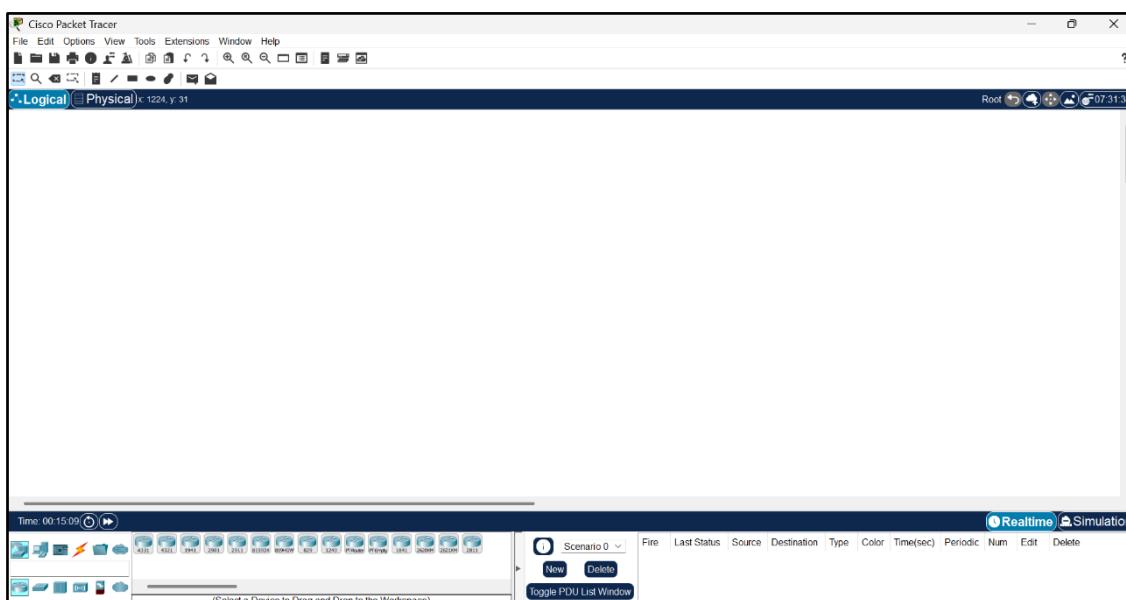
Roll No : TE-43

Name : Sanika Nilesh Patil

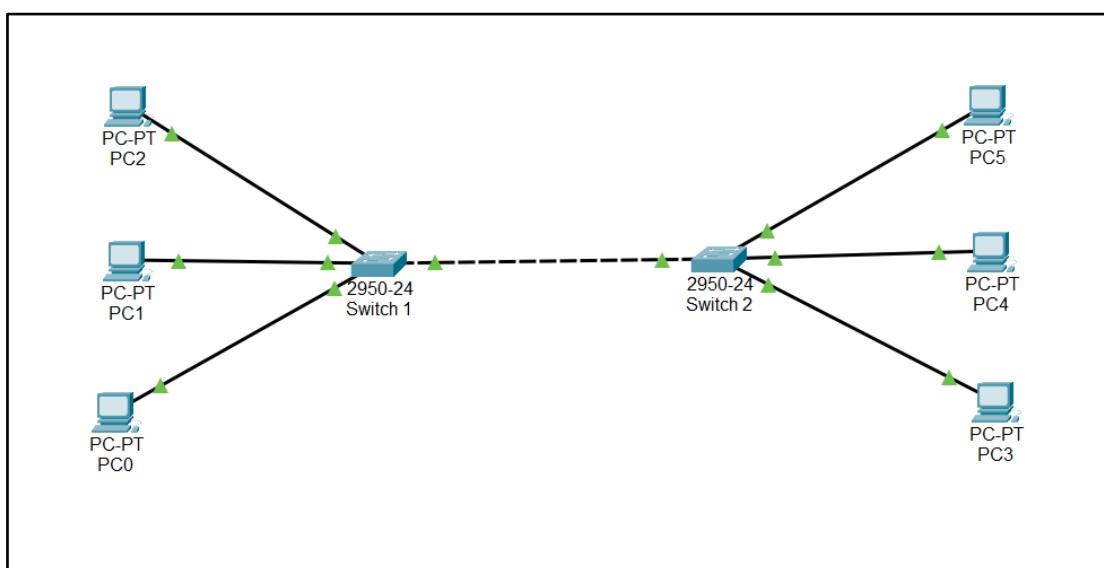
Title : Setup a wired LAN using Layer 2 Switch. It includes preparation of cable, testing of cable using line tester, configuration machine using IP addresses, testing using PING utility and demonstrating the PING packets captured traces using Wireshark Packet Analyzer Tool.

.....

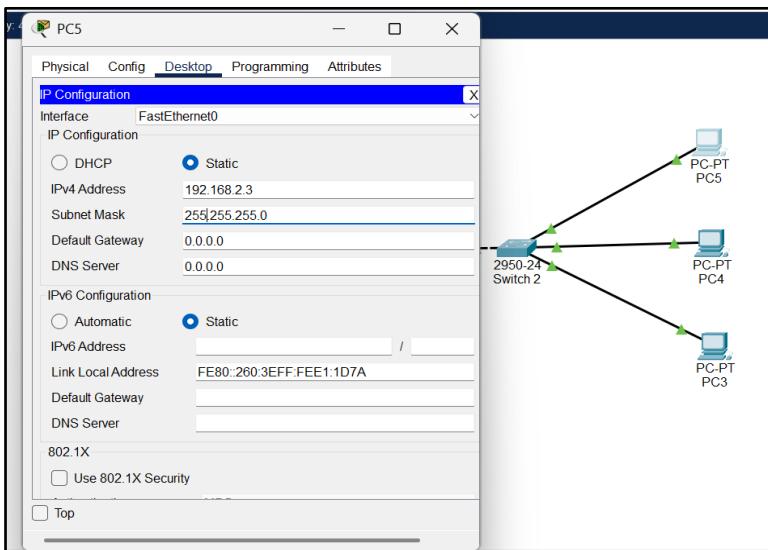
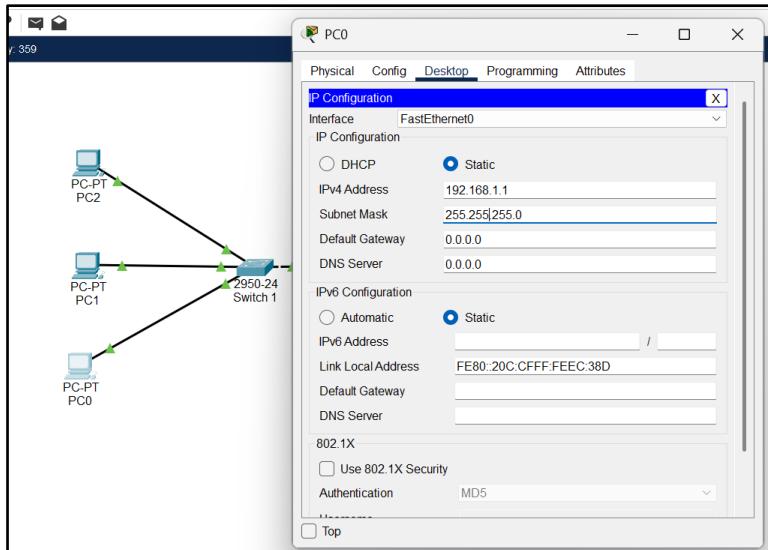
Step 1: Launch Packet Tracer.



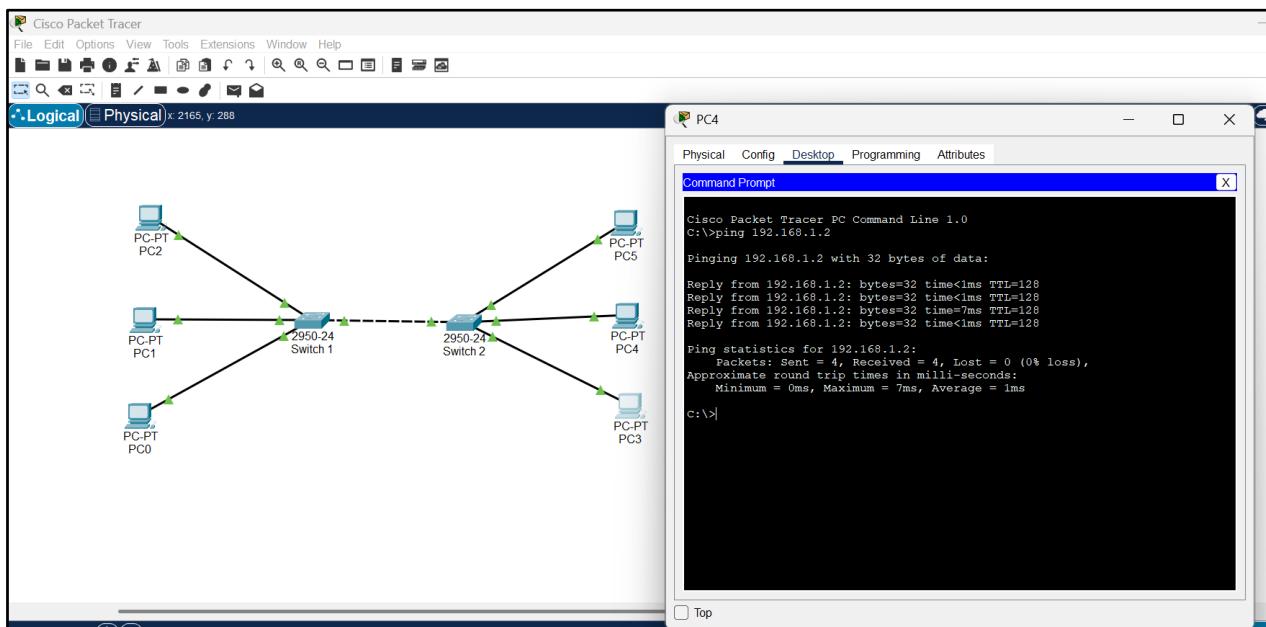
Step 2: Build the topology.



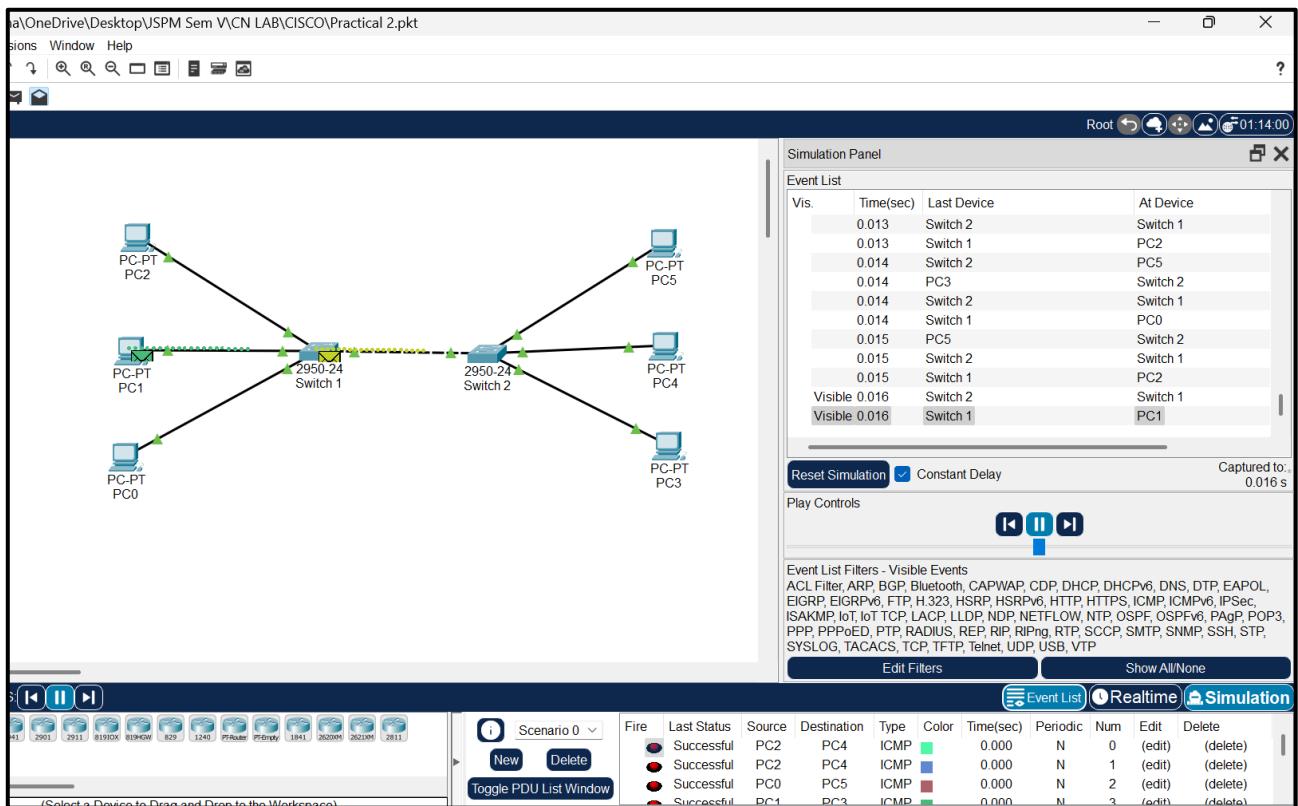
Step 3: Configure IP addresses and subnet masks on all PCs.



Step 4: Test network connectivity using the PING command between PCs.



Step 5: Capture and analyze the PING packets using Simulation Mode in Cisco Packet Tracer to demonstrate packet flow between the PCs.



Experiment No : 03

.....

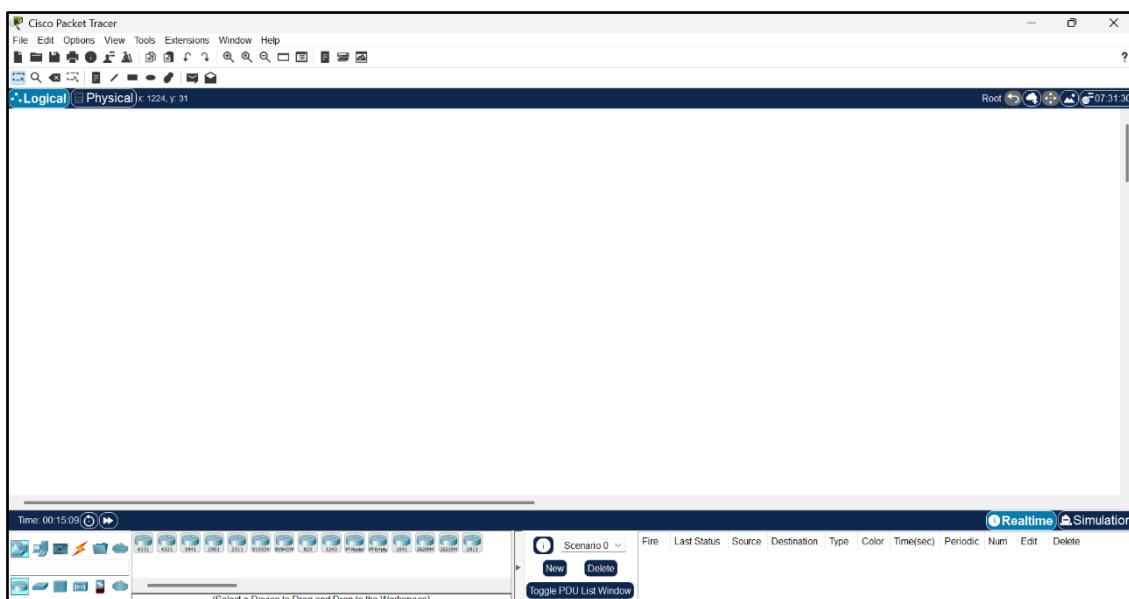
Roll No : TE-43

Name : Sanika Nilesh Patil

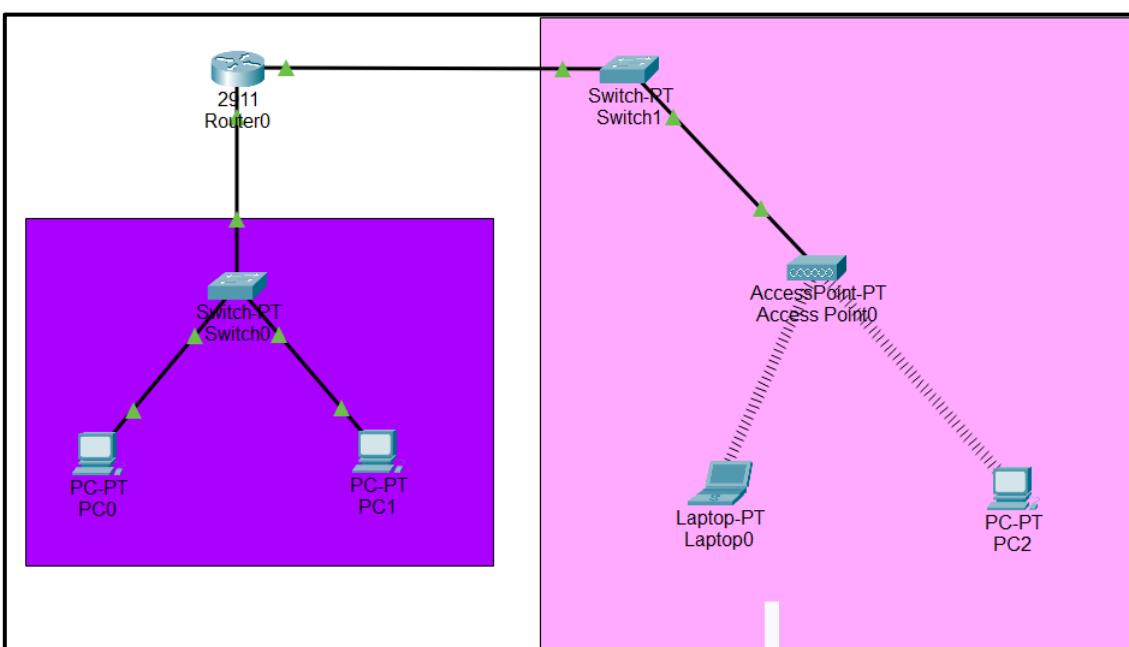
Title : Setup a WAN which contains wired as well as wireless LAN by using a packet tracer tool.
Demonstrate transfer of a packet from LAN 1 (wired LAN) to LAN2 (Wireless LAN).

.....

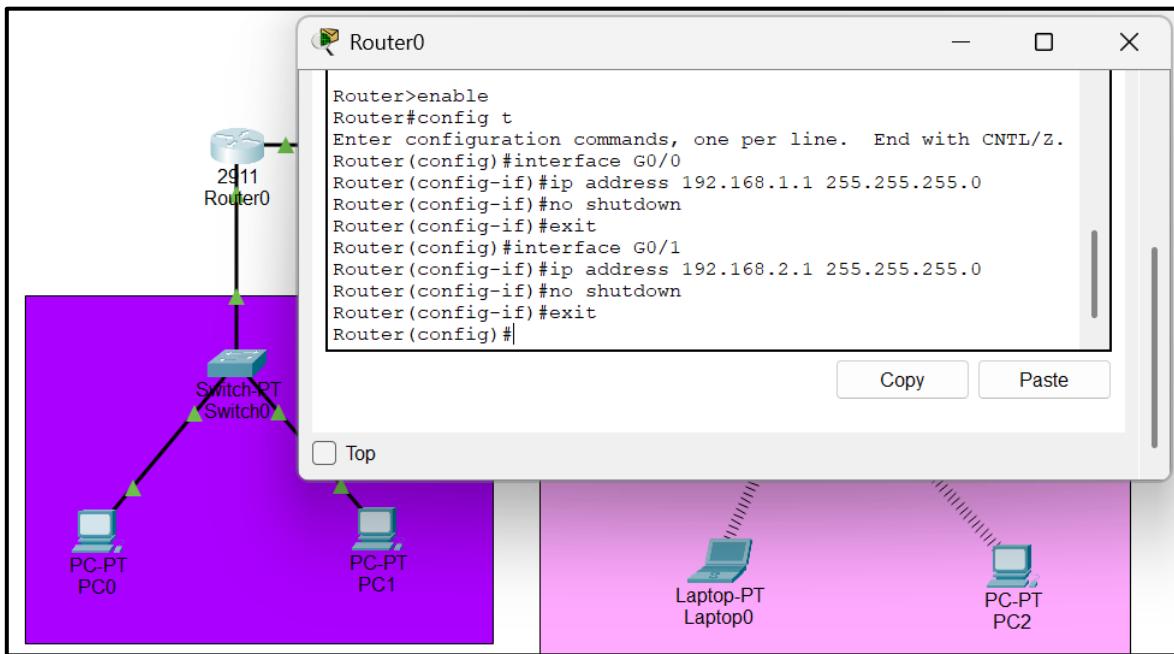
Step 1: Launch Packet Tracer.



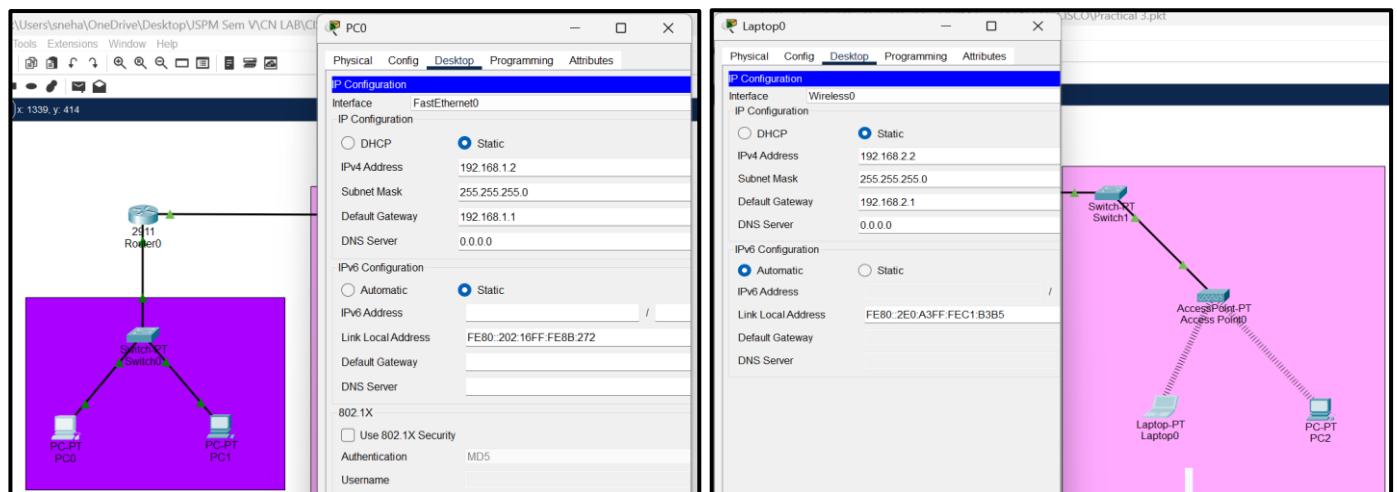
Step 2: Build the topology.



Step 3: Configure router.

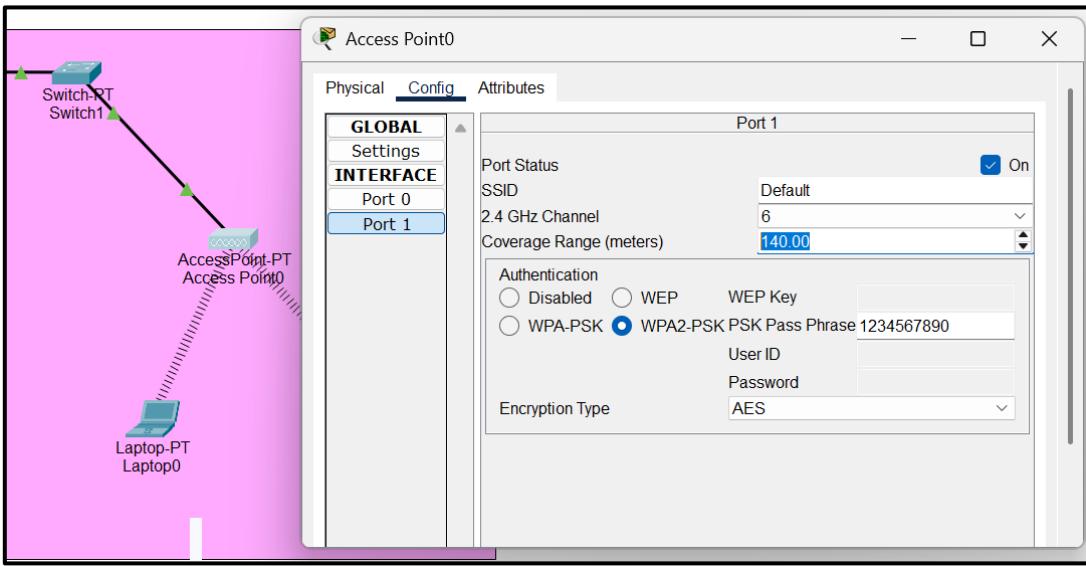


Step 4: Configure IP addresses and subnet masks on all PCs.

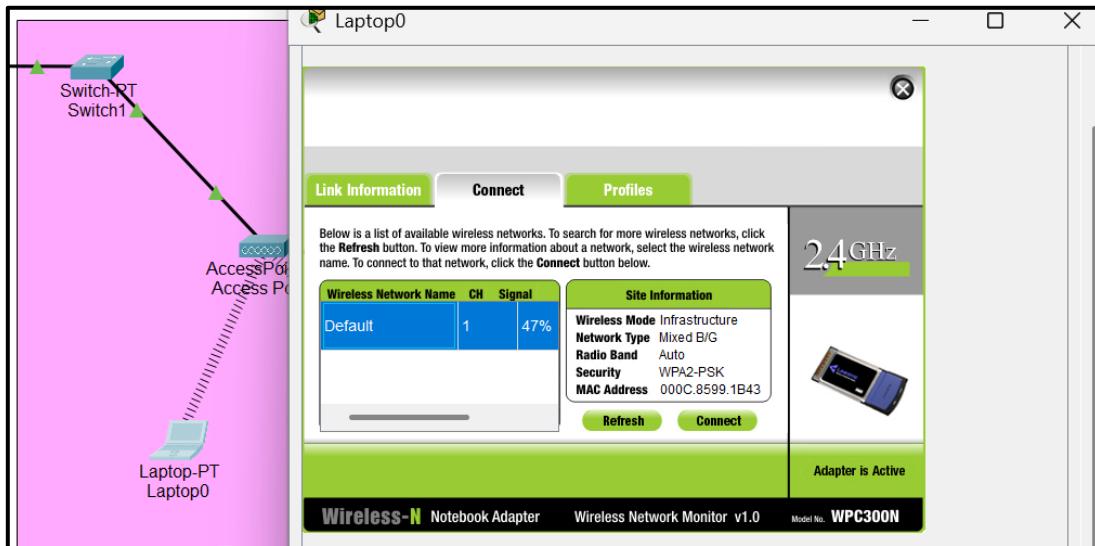


Step 5: Configure the wireless network.

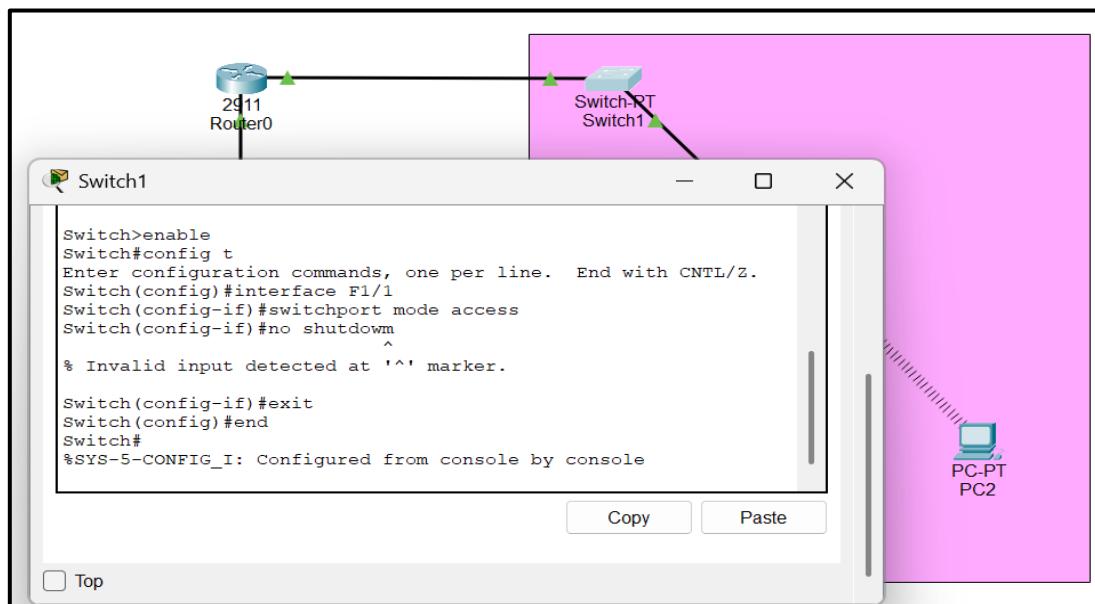
1. Click the Access Point → Config Tab → Port 1
2. Set SSID (network name) → Default
3. Set Wi-Fi password → 1234567890



Step 6: Connect Wireless laptop and PC.



Step 7: Configure switch (Wireless network) for wireless connection .



Step 8: Verify network connectivity between wired and wireless network.

PC0

Physical Config Desktop Programming Attributes

Command Prompt

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

Pinging 192.168.2.2 with 32 bytes of data:

Reply from 192.168.2.2: bytes=32 time=35ms TTL=127
Reply from 192.168.2.2: bytes=32 time=39ms TTL=127
Reply from 192.168.2.2: bytes=32 time=33ms TTL=127
Reply from 192.168.2.2: bytes=32 time=7ms TTL=127

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 7ms, Maximum = 39ms, Average = 28ms

C:\>
```

Laptop0

Physical Config Desktop Programming Attributes

Command Prompt

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:

Request timed out.
Reply from 192.168.1.3: bytes=32 time=33ms TTL=127
Reply from 192.168.1.3: bytes=32 time=45ms TTL=127
Reply from 192.168.1.3: bytes=32 time=29ms TTL=127

Ping statistics for 192.168.1.3:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 29ms, Maximum = 45ms, Average = 35ms

C:\>|
```

Realtime Simulation

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC1	PC2	ICMP	■	0.000	N	12	(edit)	(delete)
●	Successful	PC0	Laptop0	ICMP	■	0.000	N	13	(edit)	(delete)
●	Successful	PC0	PC2	ICMP	■	0.000	N	14	(edit)	(delete)
●	Successful	PC1	Laptop0	ICMP	■	0.000	N	15	(edit)	(delete)

Experiment No : 04

.....

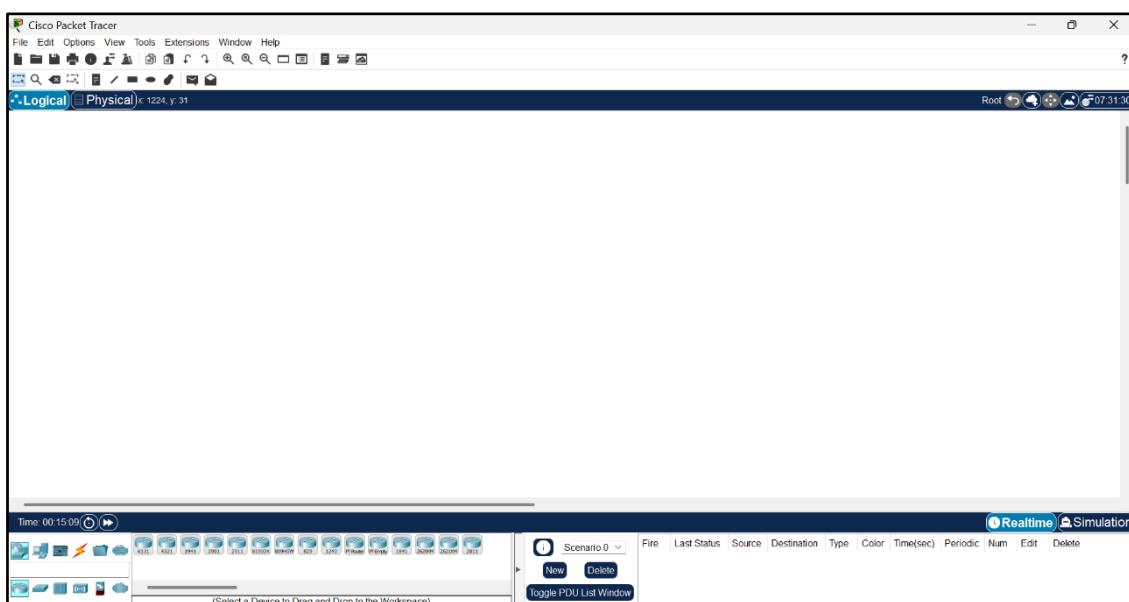
Roll No : TE- 43

Name : Sanika Nilesh Patil

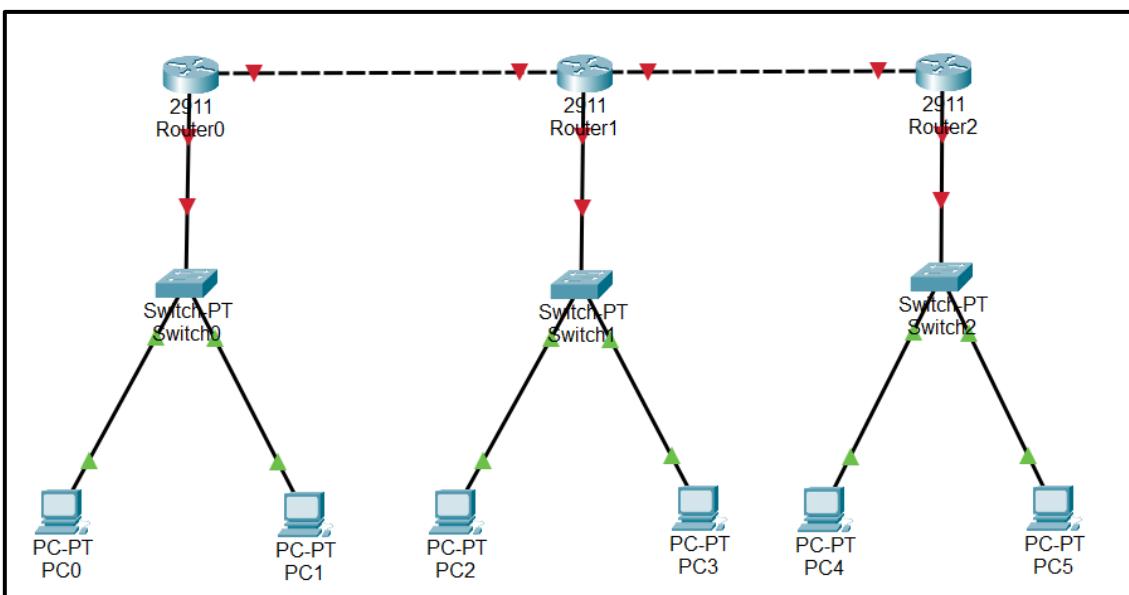
Title : Use packet Tracer tool for configuration of 3 router network using one of the following protocol
RIP/OSPF/BGP.

.....

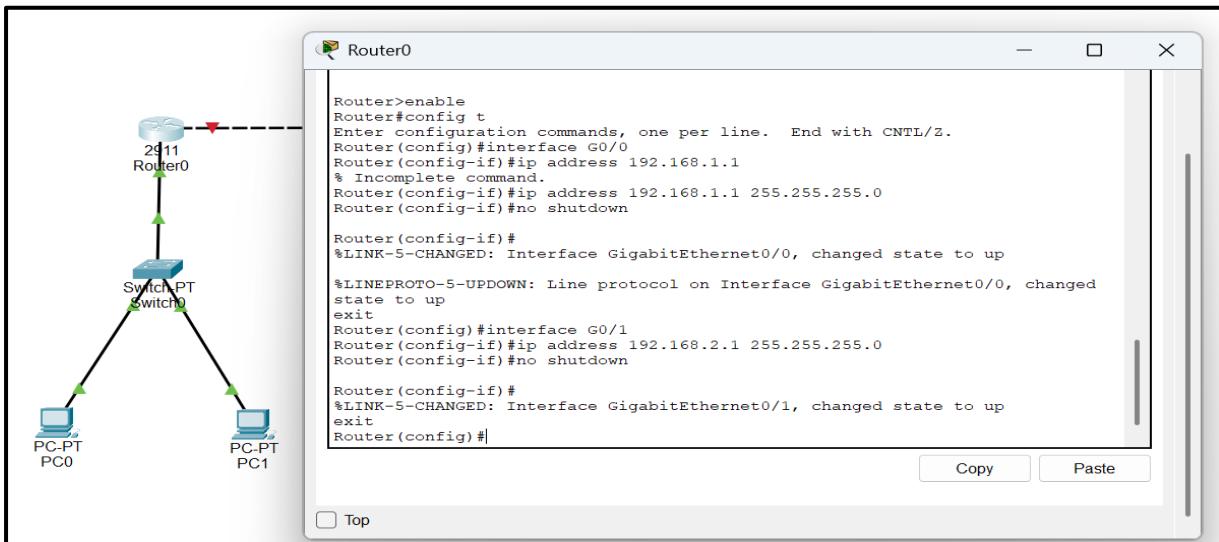
Step 1: Launch Packet Tracer.



Step 2: Build the topology.



Step 3: Configure routers.



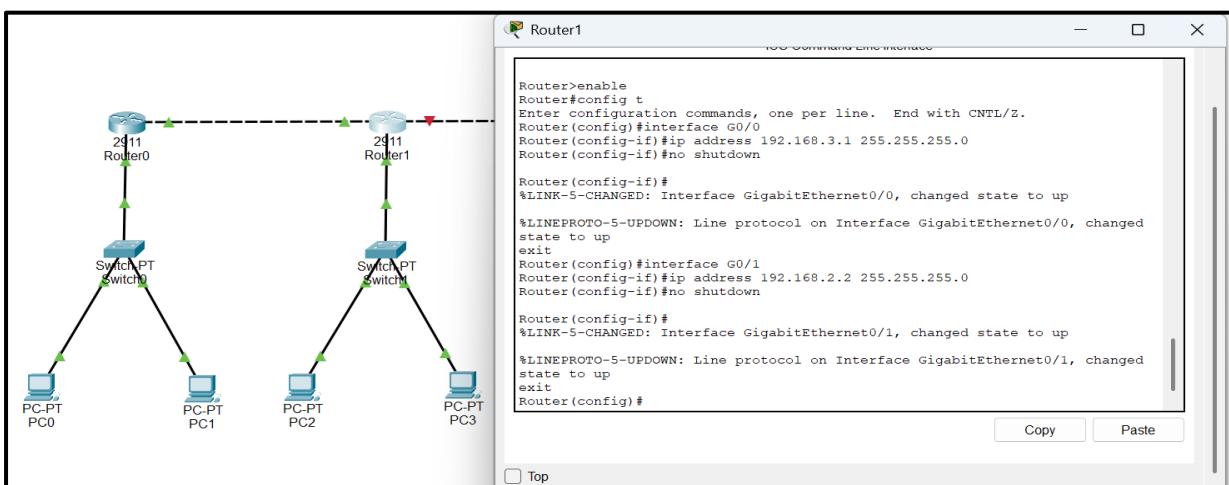
```
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface G0/0
Router(config-if)#ip address 192.168.1.0 255.255.255.0
Router(config-if)#no shutdown

Router(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
exit
Router(config)#interface G0/1
Router(config-if)#ip address 192.168.2.1 255.255.255.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up
exit
Router(config)#


```

Copy Paste



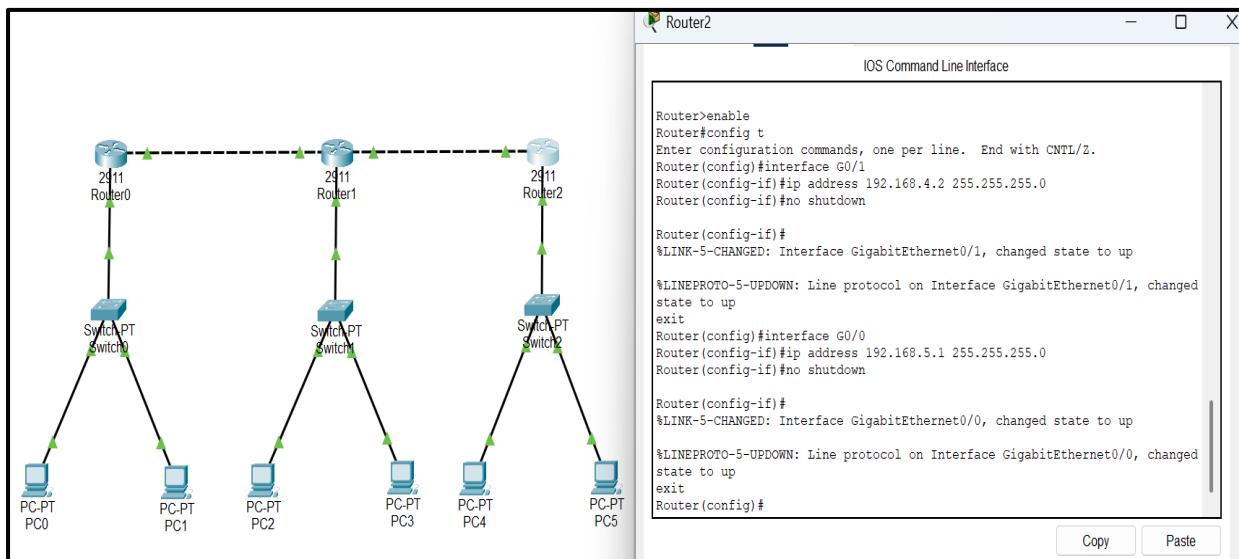
```
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface G0/0
Router(config-if)#ip address 192.168.3.1 255.255.255.0
Router(config-if)#no shutdown

Router(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
exit
Router(config)#interface G0/1
Router(config-if)#ip address 192.168.2.2 255.255.255.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed
state to up
exit
Router(config)#


```

Copy Paste



```
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface G0/1
Router(config-if)#ip address 192.168.4.2 255.255.255.0
Router(config-if)#no shutdown

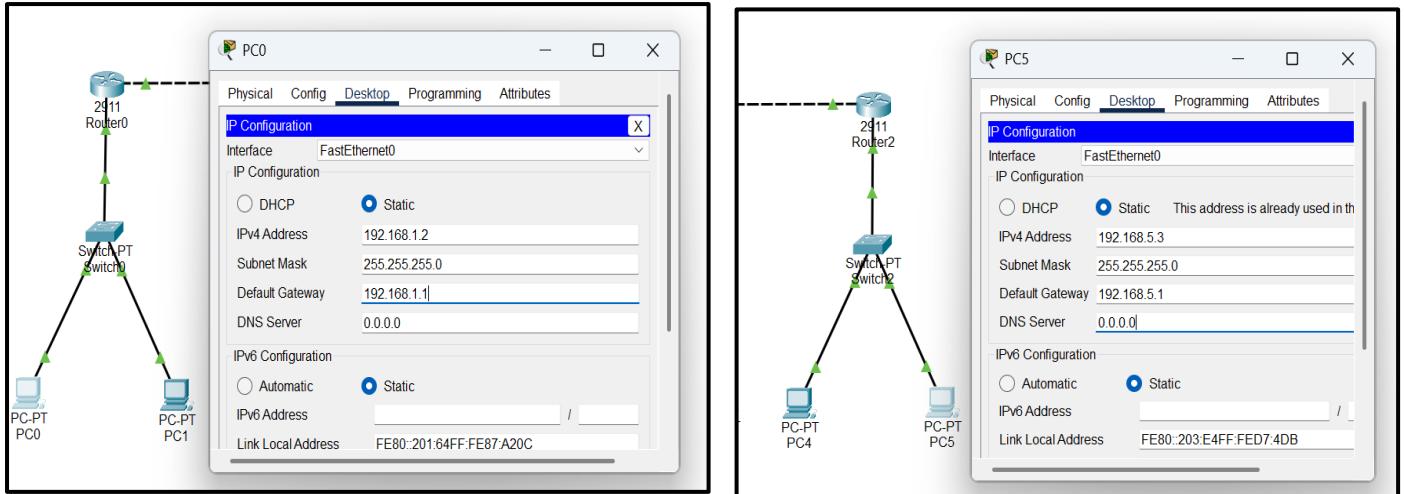
Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed
state to up
exit
Router(config)#interface G0/0
Router(config-if)#ip address 192.168.5.1 255.255.255.0
Router(config-if)#no shutdown

Router(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
exit
Router(config)#

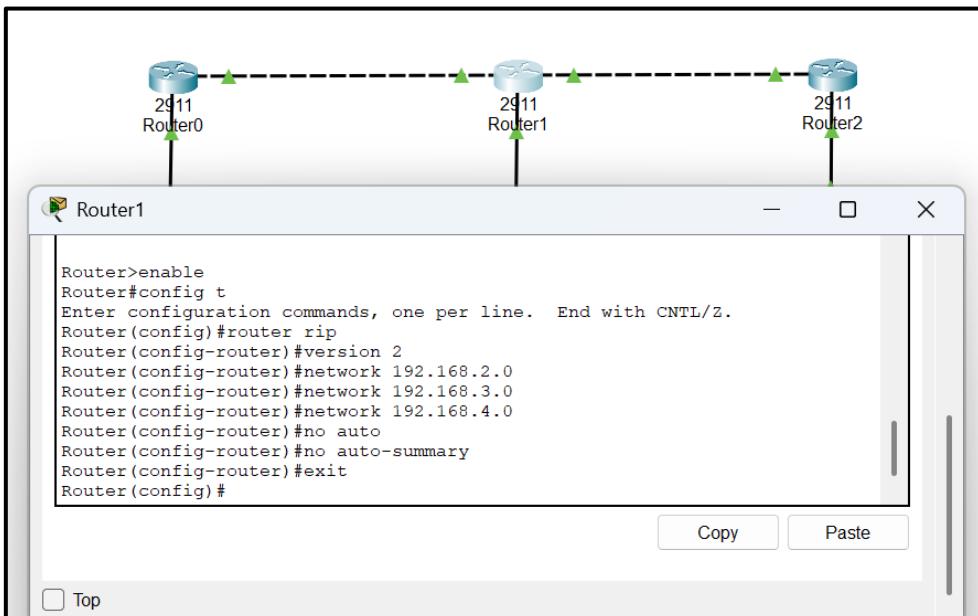
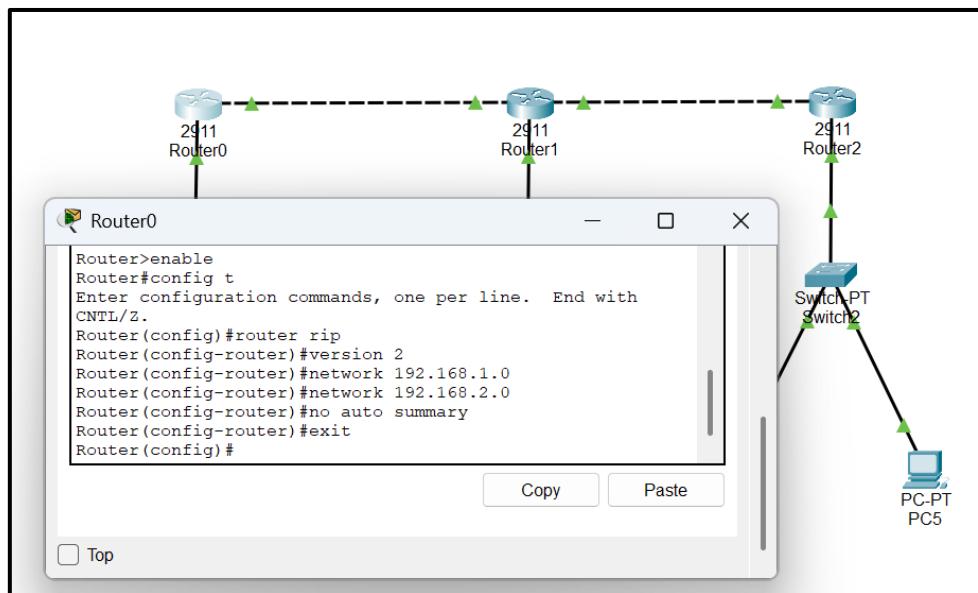

```

Copy Paste

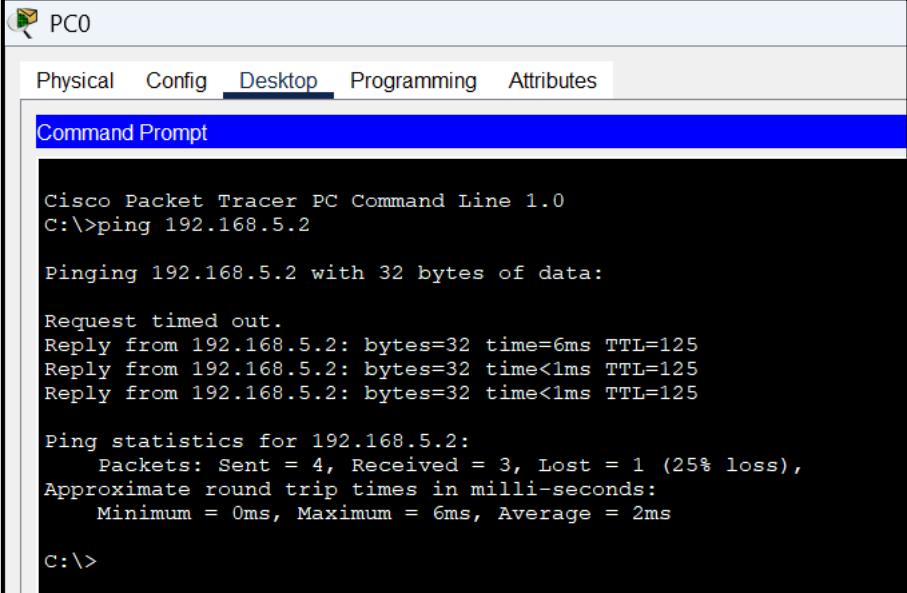
Step 4: Configure IP addresses and subnet masks and Default gateway on all PCs.



Step 5: Configuration of routers using RIP protocol.



Step 6: Verify Connectivity (Use ping command from PC0, PC1 to PC3, PC4, etc).



PC0

Physical Config Desktop Programming Attributes

Command Prompt

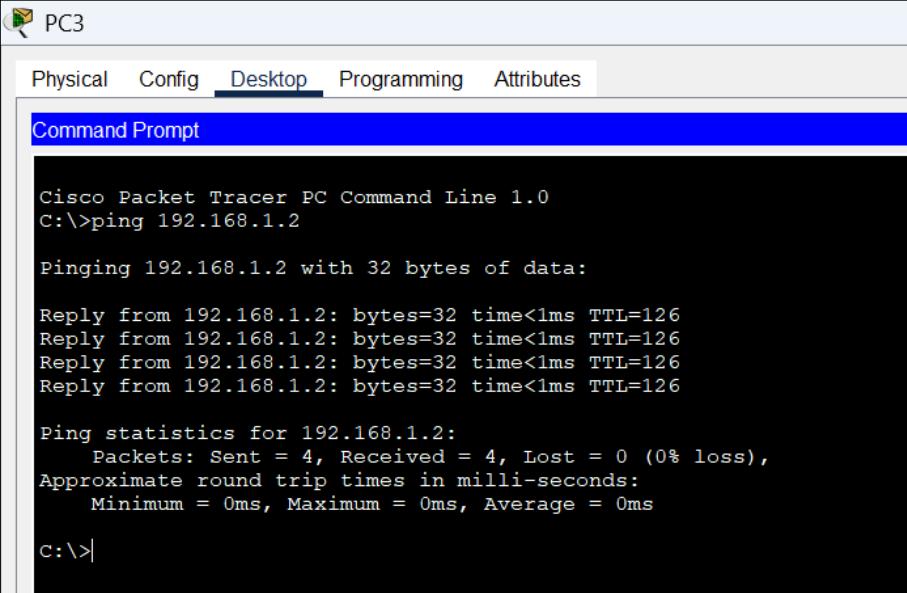
```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.5.2

Pinging 192.168.5.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.5.2: bytes=32 time=6ms TTL=125
Reply from 192.168.5.2: bytes=32 time<1ms TTL=125
Reply from 192.168.5.2: bytes=32 time<1ms TTL=125

Ping statistics for 192.168.5.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 6ms, Average = 2ms

C:\>
```



PC3

Physical Config Desktop Programming Attributes

Command Prompt

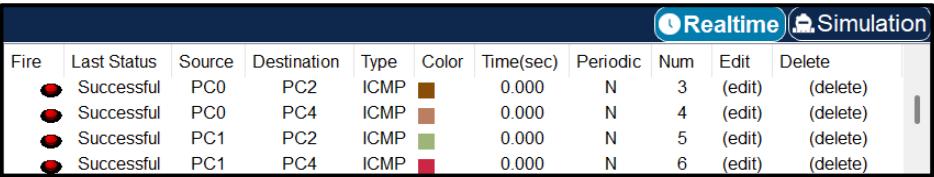
```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=126

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```



Realtime Simulation

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC2	ICMP	■	0.000	N	3	(edit)	(delete)
●	Successful	PC0	PC4	ICMP	■	0.000	N	4	(edit)	(delete)
●	Successful	PC1	PC2	ICMP	■	0.000	N	5	(edit)	(delete)
●	Successful	PC1	PC4	ICMP	■	0.000	N	6	(edit)	(delete)

Experiment No : 07

"""

Roll No : TE-43

Name : Sanika Nilesh Patil

Title : Implement Socket Programming using Python:

- a) TCP Client and TCP Server
- b) UDP Client and UDP Server

"""

Source Code :

```
import socket
import threading
import time

# -----
# TCP Server
# -----

def tcp_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('localhost', 12345))
    server_socket.listen(1)
    print("[TCP Server] Waiting for connection...")

    conn, addr = server_socket.accept()
    print(f"[TCP Server] Connected by {addr}")
    data = conn.recv(1024).decode()
    print(f"[TCP Server] Client says: {data}")
    conn.sendall("Hello from TCP Server".encode())
    conn.close()
    print("[TCP Server] Connection closed.")

# -----
# TCP Client
# -----
```

```
def tcp_client():
    time.sleep(1) # wait for server to start
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 12345))
    client_socket.sendall("Hello from TCP Client".encode())
    data = client_socket.recv(1024).decode()
    print(f"[TCP Client] Server says: {data}")
    client_socket.close()
    print("[TCP Client] Connection closed.")

# -----
# UDP Server
# -----
def udp_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind(('localhost', 12346)) # use different port
    print("[UDP Server] Waiting for message...")
    data, addr = server_socket.recvfrom(1024)
    print(f"[UDP Server] Received from {addr}: {data.decode()}")
    server_socket.sendto("Hello from UDP Server".encode(), addr)
    print("[UDP Server] Response sent.")

# -----
# UDP Client
# -----
def udp_client():
    time.sleep(1) # wait for server to start
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_address = ('localhost', 12346)
    client_socket.sendto("Hello from UDP Client".encode(), server_address)
    data, addr = client_socket.recvfrom(1024)
    print(f"[UDP Client] Server says: {data.decode()}")
    client_socket.close()
    print("[UDP Client] Connection closed.")
```

```

# -----
# Run all using threads
# -----
threads = []

# TCP
threads.append(threading.Thread(target=tcp_server))
threads.append(threading.Thread(target=tcp_client))

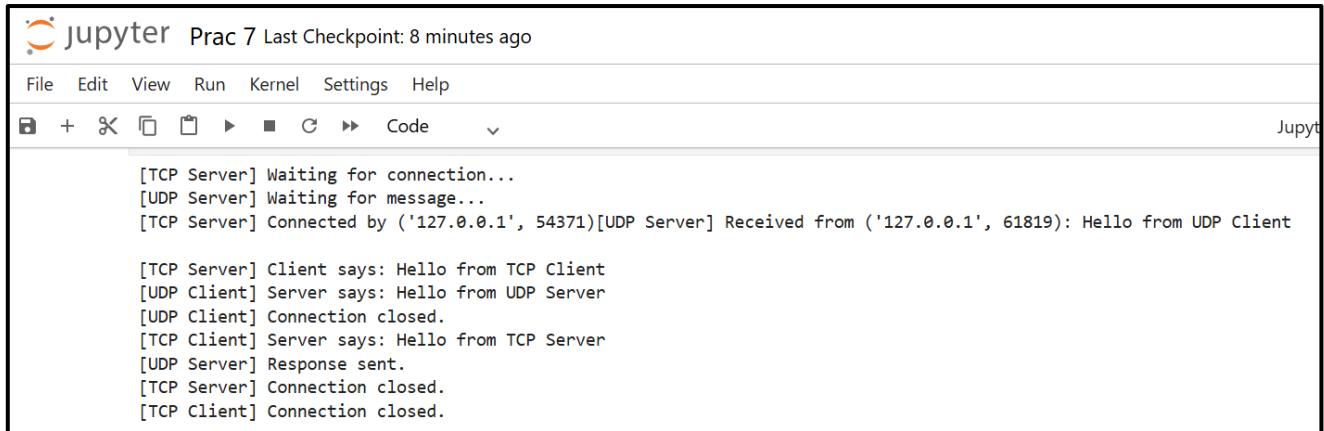
# UDP
threads.append(threading.Thread(target=udp_server))
threads.append(threading.Thread(target=udp_client))

# Start all threads
for t in threads:
    t.start()

# Wait for all threads to finish
for t in threads:
    t.join()

```

Output :



```

jupyter Prac 7 Last Checkpoint: 8 minutes ago
File Edit View Run Kernel Settings Help
+ X □ ▶ ■ ⌂ ▶ Code Jupyter
[TCP Server] Waiting for connection...
[UDP Server] Waiting for message...
[TCP Server] Connected by ('127.0.0.1', 54371)[UDP Server] Received from ('127.0.0.1', 61819): Hello from UDP Client
[TCP Server] Client says: Hello from TCP Client
[UDP Client] Server says: Hello from UDP Server
[UDP Client] Connection closed.
[TCP Client] Server says: Hello from TCP Server
[UDP Server] Response sent.
[TCP Server] Connection closed.
[TCP Client] Connection closed.

```

Experiment No : 08

"""

Roll No : TE-43

Name : Sanika Nilesh Patil

Title : Write a program using TCP socket for wired network for following

1. Say Hello to each other
2. File transfer
3. Calculator.

"""

Source Code :

❖ **Server Side Program :**

```
import socket
```

```
# Step 1: Create socket
```

```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
# Step 2: Bind server to host and port
```

```
server_host = '0.0.0.0' # Listen on all interfaces
```

```
server_port = 5001
```

```
server_socket.bind((server_host, server_port))
```

```
print(f"Server listening on {server_host}:{server_port}")
```

```
# Step 3: Listen and accept connection
```

```
server_socket.listen(1)
```

```
conn, addr = server_socket.accept()
```

```
print(f"Connected by {addr}")
```

```
# Step 4: Exchange hello messages
```

```
conn.sendall(b"Hello Client")
```

```
client_msg = conn.recv(1024).decode()
```

```
print("Client says:", client_msg)
```

```
# Step 5: Receive file
```

```
filename = "received_file.txt"
```

```
with open(filename, "wb") as f:  
    print("Receiving file...")  
    while True:  
        data = conn.recv(1024)  
        if not data:  
            break  
        f.write(data)  
  
print(f'File {filename} received successfully.')
```

```
# Step 6: Close connection  
conn.close()  
server_socket.close()  
print("Server closed.")
```

❖ Client Side Program :

```
import socket  
  
# Step 1: Create socket  
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
  
# Step 2: Connect to the server  
server_ip = '127.0.0.1' # Use 'localhost' if on same machine, else server's IP  
server_port = 5001  
client_socket.connect((server_ip, server_port))  
print("Connected to server.")  
  
# Step 3: Exchange hello messages  
server_msg = client_socket.recv(1024).decode()  
print("Server says:", server_msg)  
client_socket.sendall(b"Hello Server")  
  
# Step 4: Send a file  
file_to_send = "sample.txt"  
with open(file_to_send, "rb") as f:
```

```
print("Sending file...")
data = f.read(1024)
while data:
    client_socket.sendall(data)
    data = f.read(1024)
```

```
print("File sent successfully.")
```

```
# Step 5: Close connection
client_socket.close()
print("Client closed.")
```

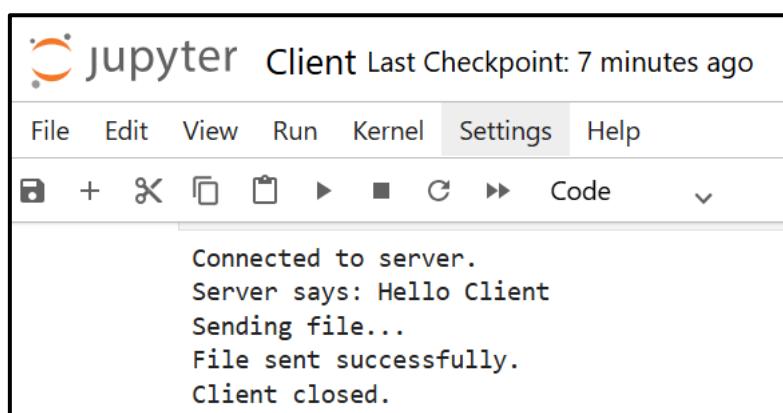
❖ Sample Text File :

```
with open("sample.txt", "w") as f:
```

```
    f.write("This is a sample file for TCP file transfer test.\nHello from Client!")
    print("Sample file created.")
```

❖ Output :

❖ Client Side :



The screenshot shows a Jupyter Client window with the following details:

- Title Bar:** jupyter Client Last Checkpoint: 7 minutes ago
- Menu Bar:** File Edit View Run Kernel Settings Help
- Toolbar:** Includes icons for file operations like Open, Save, and Print.
- Output Area:** Displays the following text:

```
Connected to server.
Server says: Hello Client
Sending file...
File sent successfully.
Client closed.
```

❖ Server Side :

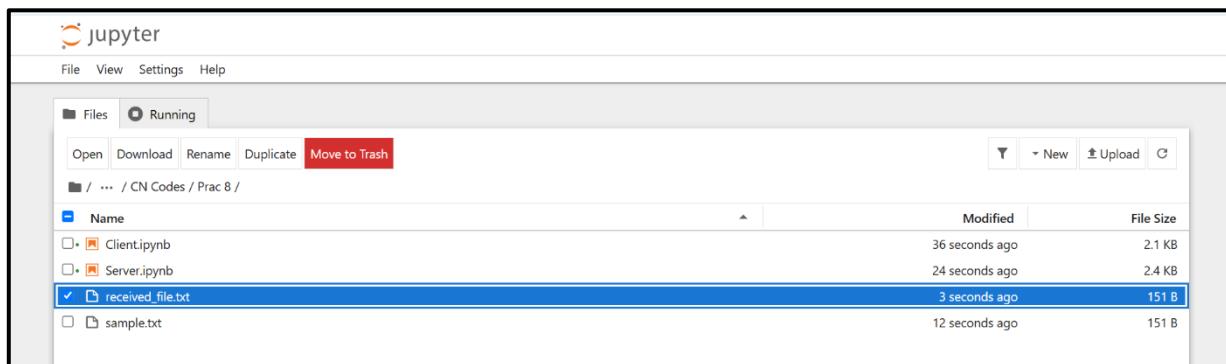
jupyter Server Last Checkpoint: 8 minutes ago

File Edit View Run Kernel Settings Help

print("Server closed.")

```
Server listening on 0.0.0.0:5001
Connected by ('127.0.0.1', 63974)
Client says: Hello Server
Receiving file...
File 'received_file.txt' received successfully.
Server closed.
```

❖ Received text file :



jupyter received_file.txt Last Checkpoint: 8 minutes ago

File Edit View Settings Help

```
1 with open("sample.txt", "w") as f:
2     f.write("This is a sample file for TCP file transfer test.\nHello from Client!")
3 print("Sample file created.")
```

Experiment No : 10

.....

Roll No : TE-43

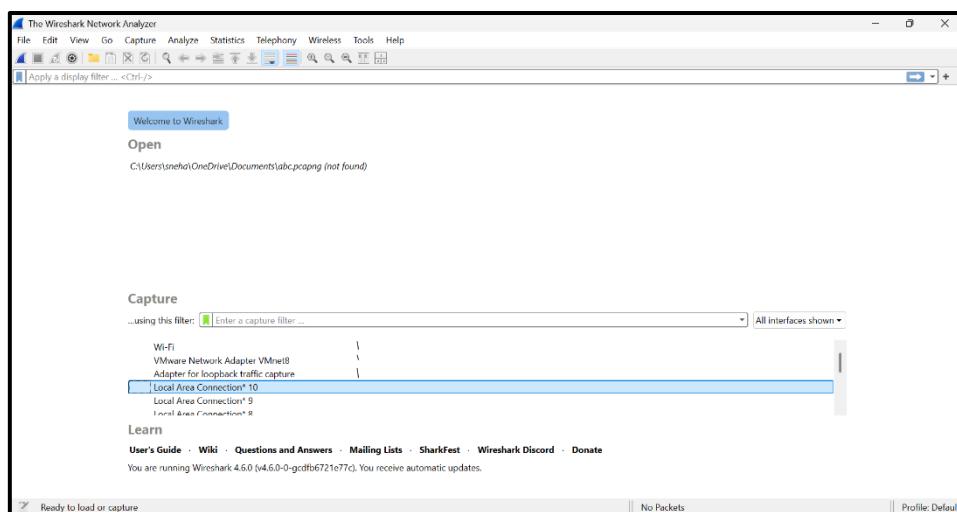
Name : Sanika Nilesh Patil

Title : Capture packets using Wireshark, write the exact packet capture filter expressions to accomplish the following and save the output in file:

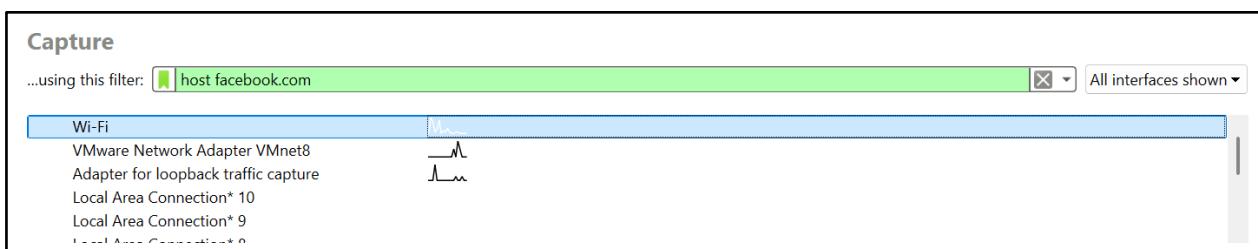
1. Capture all TCP traffic to/from Facebook, during the time when you log in to your Facebook account.
2. Capture all HTTP traffic to/from Facebook, when you log in to your Facebook account
3. Write a DISPLAY filter expression to count all TCP packets (captured under item #1) that have the flags SYN, PSH, and RST set. Show the fraction of packets that had each flag set.
4. Count how many TCP packets you received from / sent to Facebook, and how many of each were also HTTP packets.

.....

Step 1: Launch Wireshark.



Step 2: Select active network interface and start capturing filters.



Capturing from Wi-Fi (tcp port 443)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter... <Ctrl-/>

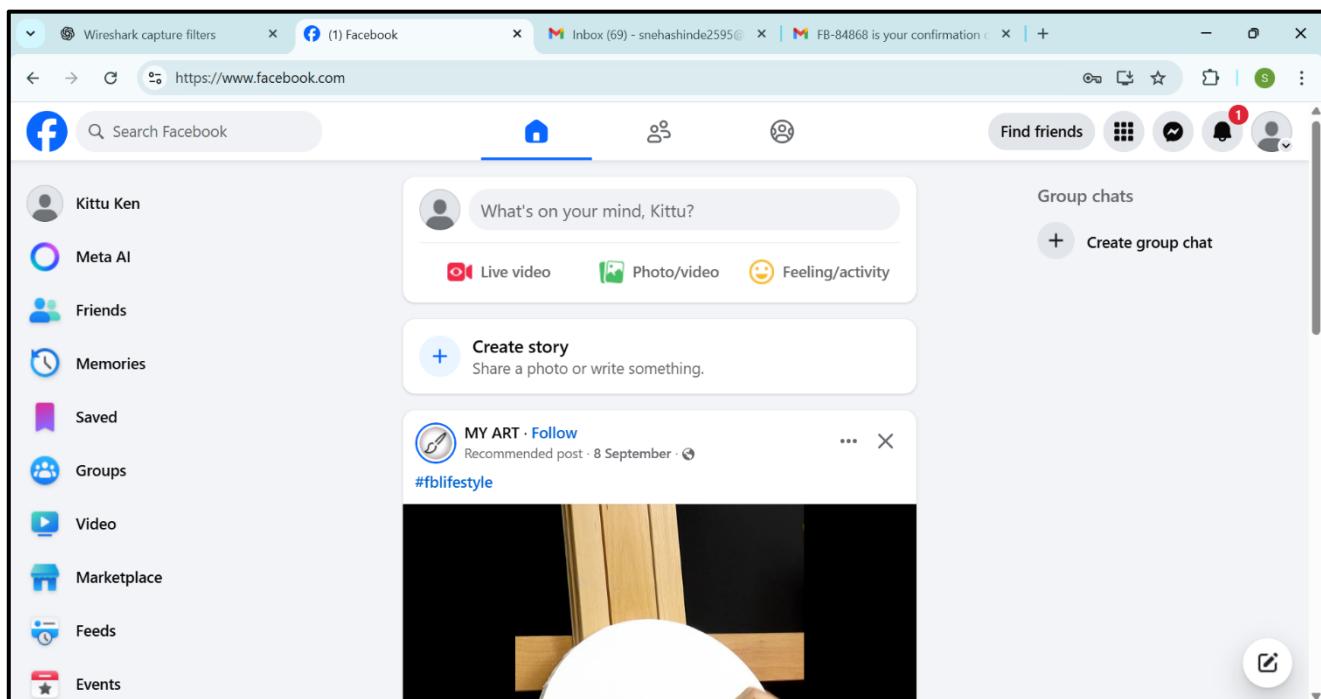
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	2401:4900:c0bb:f60d...	2606:4700:90d0:70fa...	TCP	75	55372 → 443 [ACK] Seq=1 Ack=1 Win=255 Len=1
2	0.155961	2606:4700:90d0:70fa...	2401:4900:c0bb:f60d...	TCP	86	443 → 55372 [ACK] Seq=1 Ack=2 Win=18 Len=0 SLE=1 SRE=2
3	2.161630	2606:4700:90d1:d8b7...	2401:4900:c0bb:f60d...	TLSV1.2	98	Application Data
4	2.162375	2401:4900:c0bb:f60d...	2606:4700:90d1:d8b7...	TLSV1.2	102	Application Data
5	2.228690	2606:4700:90d1:d8b7...	2401:4900:c0bb:f60d...	TCP	74	443 → 50802 [ACK] Seq=25 Ack=29 Win=19 Len=0
6	7.880288	2401:4900:c0bb:f60d...	2620:1ec:bdf:72	TCP	75	58248 → 443 [ACK] Seq=1 Ack=1 Win=251 Len=1
7	7.836047	2620:1ec:bdf:72	2401:4900:c0bb:f60d...	TLSV1.2	113	Application Data
8	7.836047	2620:1ec:bdf:72	2401:4900:c0bb:f60d...	TLSV1.2	98	Application Data
9	7.836047	2620:1ec:bdf:72	2401:4900:c0bb:f60d...	TCP	98	[TCP Retransmission] 443 → 58248 [FIN, PSH, ACK] Seq=40 Ack=2 Win=97 Len=24
10	7.836308	2401:4900:c0bb:f60d...	2620:1ec:bdf:72	TCP	86	58248 → 443 [ACK] Seq=2 Ack=65 Win=251 Len=0 SLE=40 SRE=64
11	7.836885	2401:4900:c0bb:f60d...	2620:1ec:bdf:72	TCP	74	58248 → 443 [FIN, ACK] Seq=2 Ack=65 Win=251 Len=0

```

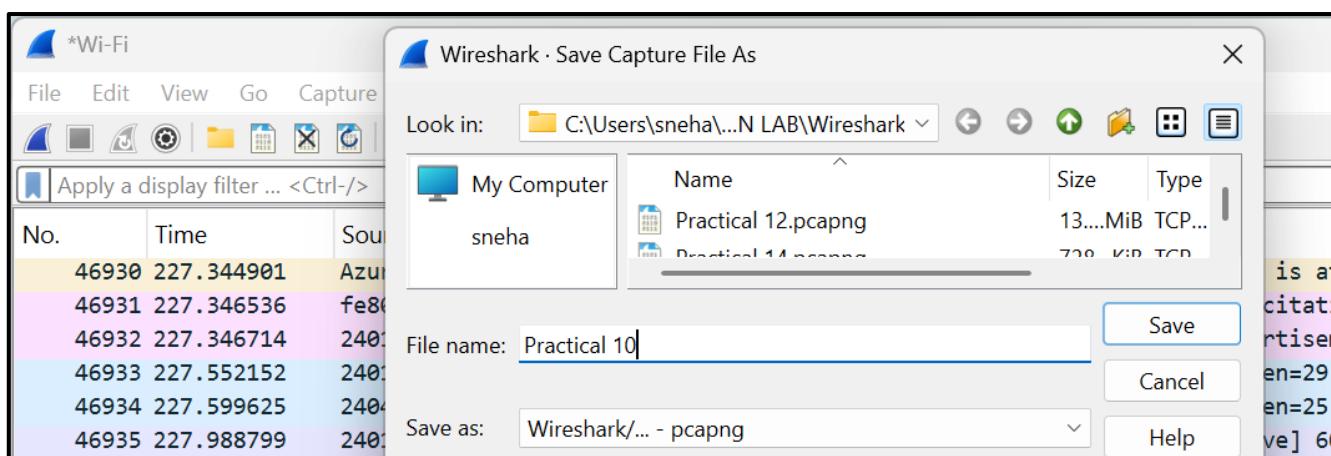
> Frame 1: Packet, 75 bytes on wire (600 bits), 75 bytes captured
> Ethernet II, Src: AzureWave_2d:db:47 (34:6f:24:2d:db:47), Dst: b...
> Internet Protocol Version 6, Src: 2401:4900:c0bb:f60d:f40f:3c13
> Transmission Control Protocol, Src Port: 55372, Dst Port: 443, ...
0000: be dd 7b 75 76 9a 34 6f 24 2d db 47 86 dd 60 0a ...{uv-4o $-G-`-
0010: 46 b5 00 15 06 3f 24 01 49 00 c0 bb f6 0d f4 0f F-----?$. I-----
0020: 3c 13 82 00 bc c6 26 06 47 00 90 d0 70 fa ce ca <-----&: G---p---
0030: 05 ca b6 d3 4c d4 d8 4c 01 bb 8c 7c 1c 30 f1 e4 -----L-L---|0--#
0040: 23 a0 50 10 00 ff 3a d8 00 00 00 # P-----.

```

Step 3: Login to your Facebook account.



Step 4: Stop capturing and save file.



Step 5: Apply display filters :

1. TCP Traffic :

Practical 10.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

No.	Time	Source	Destination	Protocol	Length	Info
372	23.229568	10.222.209.22	10.222.209.207	DNS	88	Standard query 0x39d1 HTTPS www.facebook.com
373	23.229852	10.222.209.22	10.222.209.207	TCP	56	54257 → 53 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=2 [TCP PDU reassembled]
374	23.230058	10.222.209.22	10.222.209.207	DNS	88	Standard query 0x2091 AAAA www.facebook.com
375	23.230369	10.222.209.22	10.222.209.207	TCP	56	59558 → 53 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=2 [TCP PDU reassembled]
376	23.230629	10.222.209.22	10.222.209.207	DNS	88	Standard query 0x4905 A www.facebook.com
377	23.235018	10.222.209.207	10.222.209.22	TCP	54	53 → 54257 [ACK] Seq=1 Ack=3 Win=65536 Len=0
378	23.235018	10.222.209.207	10.222.209.22	TCP	54	53 → 54884 [ACK] Seq=1 Ack=3 Win=65536 Len=0
379	23.235018	10.222.209.207	10.222.209.22	TCP	54	53 → 54257 [ACK] Seq=1 Ack=37 Win=65536 Len=0
380	23.235018	10.222.209.207	10.222.209.22	TCP	54	53 → 54884 [ACK] Seq=1 Ack=37 Win=65536 Len=0
381	23.235018	10.222.209.207	10.222.209.22	TCP	54	53 → 59558 [ACK] Seq=1 Ack=3 Win=65536 Len=0
382	23.235018	10.222.209.207	10.222.209.22	TCP	54	53 → 59558 [ACK] Seq=1 Ack=37 Win=65536 Len=0

```
> Frame 372: Packet, 88 bytes on wire (784 bits), 88 bytes captured
> Ethernet II, Src: AzureWav_2d:db:47 (34:6f:24:2d:db:47), Dst: 
> Internet Protocol Version 4, Src: 10.222.209.22, Dst: 10.222.209.207
> Transmission Control Protocol, Src Port: 54884, Dst Port: 53, 
> [2 Reassembled TCP Segments (36 bytes): #371(2), #372(34)]
> Domain Name System (query)
```

```
0000 d6 11 1b fb 18 38 34 6f 24 2d db 47 08 00 45 00 ... 84o $-G-E-
0010 00 4a 42 26 40 00 80 06 ff e5 0a de d1 16 0a de JBB@...
0020 d1 cf d6 64 00 35 6f ac ec 5b b2 d5 5e 7b 50 18 ... d5o .[.^P-
0030 00 ff 82 05 00 00 39 d1 01 00 00 01 00 00 00 00 ..... 9.
0040 00 00 03 77 77 77 08 66 61 63 65 62 6f 6b 03 ... www-f acebook-
0050 63 6f 6d 00 00 41 00 01 com .A..
```

2. HTTP Traffic :

Practical 10.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

No.	Time	Source	Destination	Protocol	Length	Info
17956	159.960052	10.222.209.22	23.193.165.65	HTTP	178	GET /ncsi.txt HTTP/1.1
17960	160.037223	23.193.165.65	10.222.209.22	HTTP	233	HTTP/1.1 200 OK (text/plain)
18201	174.947549	10.222.209.22	23.193.165.65	HTTP	178	GET /ncsi.txt HTTP/1.1
18205	174.995027	23.193.165.65	10.222.209.22	HTTP	233	HTTP/1.1 200 OK (text/plain)
18377	190.097995	10.222.209.22	23.193.165.43	HTTP	178	GET /ncsi.txt HTTP/1.1
18381	190.151425	23.193.165.43	10.222.209.22	HTTP	233	HTTP/1.1 200 OK (text/plain)
37059	204.956216	10.222.209.22	23.193.165.43	HTTP	178	GET /ncsi.txt HTTP/1.1
37064	204.993384	23.193.165.43	10.222.209.22	HTTP	233	HTTP/1.1 200 OK (text/plain)
+ 46851	220.080090	10.222.209.22	23.193.165.65	HTTP	178	GET /ncsi.txt HTTP/1.1
+ 46855	220.114576	23.193.165.65	10.222.209.22	HTTP	233	HTTP/1.1 200 OK (text/plain)

```
> Frame 46855: Packet, 233 bytes on wire (1864 bits), 233 bytes captured
> Ethernet II, Src: d6:11:1b:fb:18:38 (d6:11:1b:fb:18:38), Dst: 
> Internet Protocol Version 4, Src: 23.193.165.65, Dst: 10.222.209.22
> Transmission Control Protocol, Src Port: 80, Dst Port: 60987,
> Hypertext Transfer Protocol
> Line-based text data: text/plain (1 lines)
```

```
0000 34 6f 24 2d db 47 d6 11 1b fb 18 38 08 00 45 00 4o$-G...-8-E-
0010 00 db e2 f2 40 00 37 06 c7 33 17 c1 a5 41 0a de ...@.7...-A-
0020 d1 16 00 50 ee 3b 26 42 f3 04 24 3d 5c d7 50 18 ...P;&B-$\P-
0030 01 f5 90 ac 00 00 48 54 54 50 2f 31 2e 31 20 32 ...HT TP/1.1 2
0040 30 30 20 4f 4b 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 00 OK--content-L
0050 65 6e 67 74 68 3a 20 31 31 34 0d 0a 44 61 74 65 3a enghth: 1 4--Date:
0060 20 54 68 75 2c 20 31 36 20 4f 63 74 20 32 30 32 Thu, 16 Oct 202
0070 35 20 30 39 3a 34 35 3a 31 31 20 47 4d 54 0d 0a 5 09:45: 11 GMT-
0080 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 63 6c 6f 73 Connecti on: clos
0090 65 0d 0a 43 6f 6e 74 65 66 74 2d 54 79 70 65 3a e-Content-Type:
00a0 20 74 65 78 74 2f 70 6c 61 69 6e 0d 0a 43 61 63 text/plain-Cac
00b0 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6d 61 78 2d he-Contr ol: max
00c0 61 67 65 3d 33 30 2c 20 6d 75 73 74 2d 72 65 76 age=30, must-rev
00d0 61 6c 69 64 61 74 65 0d 0a 0d 0a 4d 69 63 72 6f alidate. ...Micro
00e0 73 6f 66 74 20 4e 43 53 49 soft NCS I
```

3. TCP SYN Packets :

Practical 10.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.flags.syn == 1

No.	Time	Source	Destination	Protocol	Length	Info
60	9.960377	10.222.209.22	23.193.165.65	TCP	66	57258 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
61	9.992565	23.193.165.65	10.222.209.22	TCP	66	80 → 57258 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1400 SACK_PERM WS=128
88	13.485144	10.222.209.22	10.222.209.207	TCP	66	56608 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
89	13.486035	10.222.209.22	10.222.209.207	TCP	66	49973 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
90	13.487498	10.222.209.22	10.222.209.207	TCP	66	59530 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
92	13.519436	10.222.209.207	10.222.209.22	TCP	66	53 → 56608 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM WS=512
93	13.519436	10.222.209.207	10.222.209.22	TCP	66	53 → 49973 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM WS=512
94	13.519436	10.222.209.207	10.222.209.22	TCP	66	53 → 59530 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM WS=512
113	13.665900	2401:4900:5628:2990...	2401:4900:5628:2990...	TCP	86	60724 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
114	13.666724	2401:4900:5628:2990...	2401:4900:5628:2990...	TCP	86	57221 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM
115	13.667296	2401:4900:5628:2990...	2401:4900:5628:2990...	TCP	86	56998 → 53 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM

```
> Frame 395: Packet, 86 bytes on wire (688 bits), 86 bytes captured
> Ethernet II, Src: AzureWav_2d:db:47 (34:6f:24:2d:db:47), Dst: 
> Internet Protocol Version 6, Src: 2401:4900:5628:2990:fc49:e35
> Transmission Control Protocol, Src Port: 62361, Dst Port: 53,
```

```
0000 d6 11 1b fb 18 38 34 6f 24 2d db 47 86 dd 60 09 ... 84o $-G...
0010 2c 48 00 20 06 3f 24 01 49 00 56 28 29 90 fc 49 .H-?$.I-V()-.I
0020 e3 94 f4 01 93 76 24 01 49 00 56 28 29 90 00 00 ...v$-I-V()-.I
0030 00 00 00 00 63 f3 99 00 35 c8 e1 7e 00 00 00 ...c-5~...
0040 00 00 02 ff f2 c8 00 00 02 04 05 a0 01 03 ...R-.....
0050 03 08 01 01 04 02
```

4. TCP PSH Packets :

This Wireshark capture shows several TCP PSH packets sent to www.facebook.com. The packets are highlighted in blue, indicating they have the PSH flag set. The details pane shows the raw hex and ASCII data for one of the selected packets.

No.	Time	Source	Destination	Protocol	Length	Info
374	23.230058	10.222.209.22	10.222.209.207	DNS	88	Standard query 0x2091 AAAA www.facebook.com
375	23.230369	10.222.209.22	10.222.209.207	TCP	56	59558 → 53 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=2 [TCP PDU reassembled in 376]
376	23.230629	10.222.209.22	10.222.209.207	DNS	88	Standard query 0x4985 A www.facebook.com
387	23.352322	10.222.209.207	10.222.209.22	TCP	55	53 → 54257 [PSH, ACK] Seq=1 Ack=37 Win=65536 Len=1 [TCP PDU reassembled in 396]
388	23.352322	10.222.209.207	10.222.209.22	TCP	55	53 → 59558 [PSH, ACK] Seq=1 Ack=37 Win=65536 Len=1 [TCP PDU reassembled in 397]
389	23.363406	10.222.209.207	10.222.209.22	TCP	55	53 → 54884 [PSH, ACK] Seq=1 Ack=37 Win=65536 Len=1 [TCP PDU reassembled in 398]
396	23.408252	10.222.209.207	10.222.209.22	DNS	146	Standard query response 0x2091 AAAA www.facebook.com CNAME star-mini.c10r.facebook.com A
397	23.408252	10.222.209.207	10.222.209.22	DNS	134	Standard query response 0x4905 A www.facebook.com CNAME star-mini.c10r.facebook.com A 15
398	23.408252	10.222.209.207	10.222.209.22	DNS	205	Standard query response 0x39d1 HTTPS www.facebook.com CNAME star-mini.c10r.facebook.com
417	23.455886	2a03:2880:f076:f:fa...	2401:4900:5628:2990...	TLSv1.2	102	Application Data
418	23.459237	57.144.176.145	10.222.209.22	TLSv1.2	79	Application Data

```

> Frame 389: Packet, 55 bytes on wire (440 bits), 55 bytes captured
> Ethernet II, Src: d6:11:1b:fb:18:38 (d6:11:1b:fb:18:38), Dst: 
> Internet Protocol Version 4, Src: 10.222.209.207, Dst: 10.222.209.22
> Transmission Control Protocol, Src Port: 53, Dst Port: 54884,

```

5. TCP RST Packets :

This Wireshark capture shows multiple TCP RST packets. The packets are highlighted in red, indicating they have the RST flag set. The details pane shows the raw hex and ASCII data for one of the selected packets.

No.	Time	Source	Destination	Protocol	Length	Info
25350	198.320480	2401:4900:5628:2990...	2401:4900:5628:2990...	TCP	74	62702 → 53 [RST] Seq=52 Win=0 Len=0
25352	198.321910	10.222.209.22	10.222.209.207	TCP	54	64650 → 53 [RST] Seq=52 Win=0 Len=0
25375	198.322350	2401:4900:5628:2990...	2401:4900:5628:2990...	TCP	74	62702 → 53 [RST] Seq=52 Win=0 Len=0
25378	198.322447	2401:4900:5628:2990...	2401:4900:5628:2990...	TCP	74	51520 → 53 [RST, ACK] Seq=52 Ack=2 Win=0 Len=0
25379	198.322545	2401:4900:5628:2990...	2401:4900:5628:2990...	TCP	74	53848 → 53 [RST, ACK] Seq=52 Ack=2 Win=0 Len=0
25380	198.322624	2401:4900:5628:2990...	2401:4900:5628:2990...	TCP	74	51520 → 53 [RST] Seq=52 Win=0 Len=0
25381	198.324449	10.222.209.22	10.222.209.207	TCP	54	56497 → 53 [RST, ACK] Seq=52 Ack=2 Win=0 Len=0
25383	198.324790	10.222.209.22	10.222.209.207	TCP	54	56497 → 53 [RST] Seq=52 Win=0 Len=0
25386	198.325506	10.222.209.22	10.222.209.207	TCP	54	64650 → 53 [RST] Seq=52 Win=0 Len=0
25411	198.326166	2401:4900:5628:2990...	2401:4900:5628:2990...	TCP	74	53848 → 53 [RST] Seq=52 Win=0 Len=0
25416	198.327318	10.222.209.22	10.222.209.207	TCP	54	56497 → 53 [RST] Seq=52 Win=0 Len=0

```

> Frame 25352: Packet, 54 bytes on wire (432 bits), 54 bytes captured
> Ethernet II, Src: AzureWay_2d:db:47 (34:6f:24:2d:db:47), Dst: 
> Internet Protocol Version 4, Src: 10.222.209.22, Dst: 10.222.209.207
> Transmission Control Protocol, Src Port: 64650, Dst Port: 53,

```

6. Count TCP Packets Sent to Facebook.

This Wireshark capture shows TCP packets sent from 10.222.209.22 to 10.222.209.207. The packets are highlighted in red, indicating they were sent to Facebook. The details pane shows the raw hex and ASCII data for one of the selected packets.

No.	Time	Source	Destination	Protocol	Length	Info
24264	197.381779	10.222.209.22	10.222.209.207	DNS	102	Standard query 0x95e9 AAAA scontent.fpnq7-3.fna.fbcn.net
24265	197.382006	10.222.209.22	10.222.209.207	TCP	56	64650 → 53 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=2 [TCP PDU reassembled in 24266]
24266	197.382113	10.222.209.22	10.222.209.207	DNS	102	Standard query 0x53a0 A scontent.fpnq7-3.fna.fbcn.net
24816	197.808761	10.222.209.22	10.222.209.207	TCP	54	56497 → 53 [FIN, ACK] Seq=51 Ack=1 Win=65280 Len=0
24821	197.810539	10.222.209.22	10.222.209.207	TCP	54	64650 → 53 [FIN, ACK] Seq=51 Ack=1 Win=65280 Len=0
25352	198.321910	10.222.209.22	10.222.209.207	TCP	54	64650 → 53 [RST] Seq=52 Win=0 Len=0
25381	198.324449	10.222.209.22	10.222.209.207	TCP	54	56497 → 53 [RST, ACK] Seq=52 Ack=2 Win=0 Len=0
25383	198.324790	10.222.209.22	10.222.209.207	TCP	54	56497 → 53 [RST] Seq=52 Win=0 Len=0
25386	198.325506	10.222.209.22	10.222.209.207	TCP	54	64650 → 53 [RST] Seq=52 Win=0 Len=0
25416	198.327318	10.222.209.22	10.222.209.207	TCP	54	56497 → 53 [RST] Seq=52 Win=0 Len=0

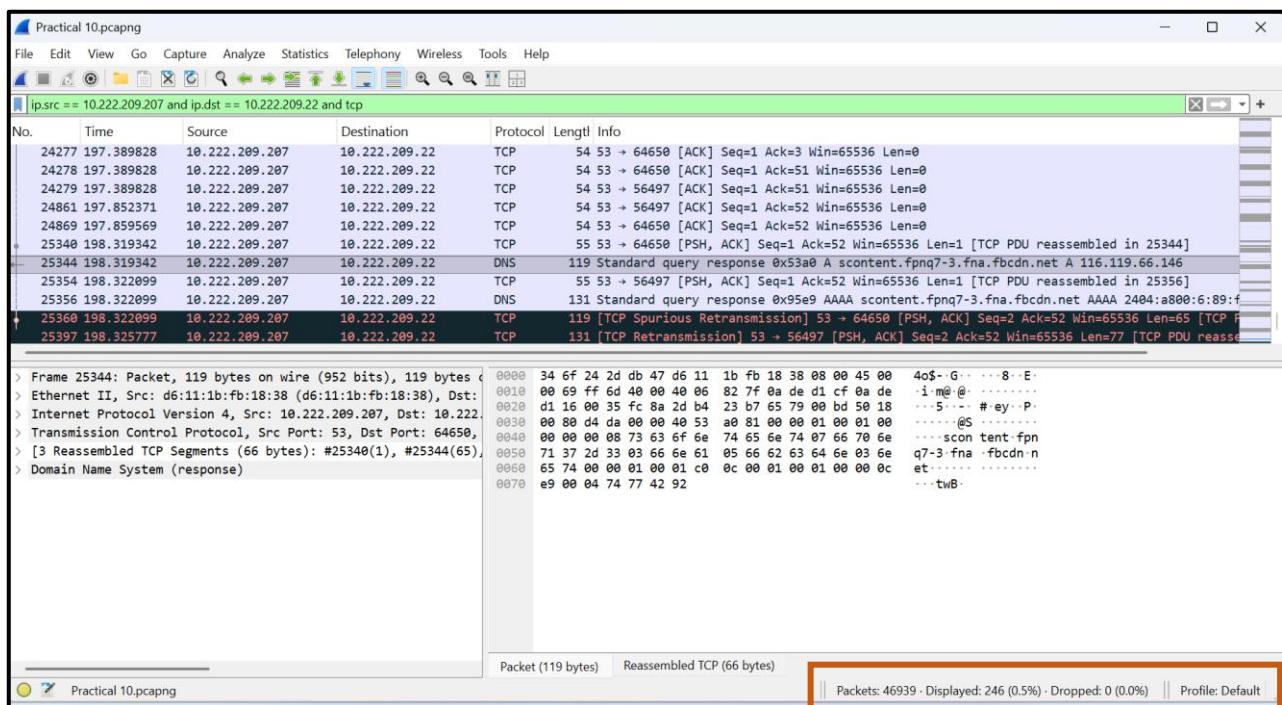
```

> Frame 25352: Packet, 54 bytes on wire (432 bits), 54 bytes captured
> Ethernet II, Src: AzureWay_2d:db:47 (34:6f:24:2d:db:47), Dst: 
> Internet Protocol Version 4, Src: 10.222.209.22, Dst: 10.222.209.207
> Transmission Control Protocol, Src Port: 64650, Dst Port: 53,

```

Packets: 46939 - Displayed: 281 (0.6%) - Dropped: 0 (0.0%) || Profile: Default

7. Count TCP Packets Received from Facebook.



Experiment No : 11

.....

Roll No : TE-43

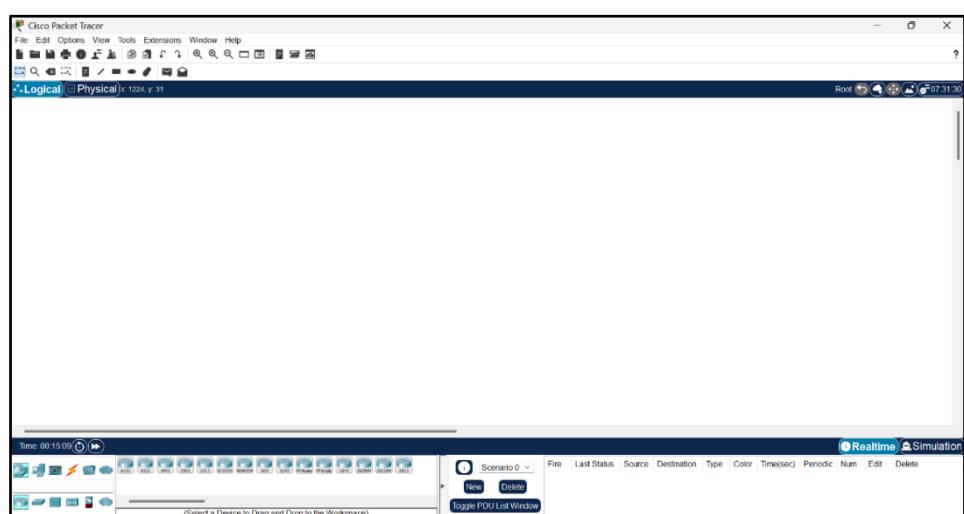
Name : Sanika Nilesh Patil

Title : Study and Analyze the performance of HTTP, HTTPS and FTP protocol using Packet tracer tool.

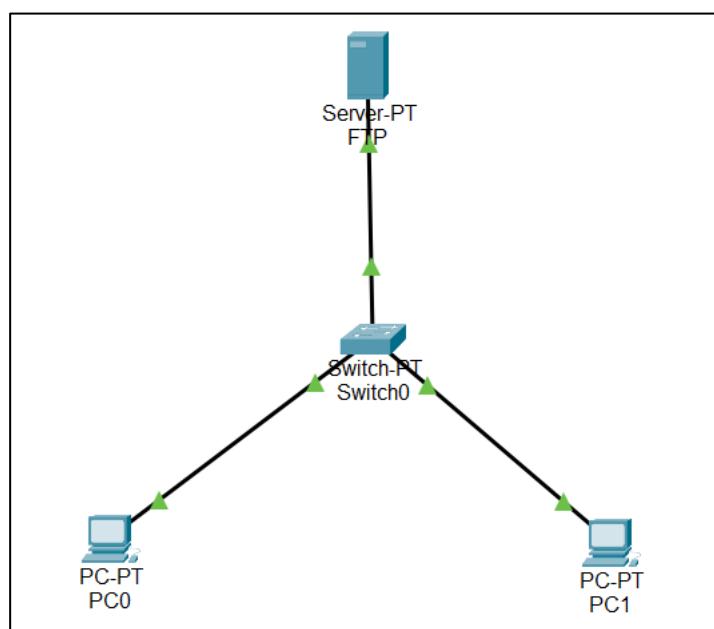
.....

FTP Protocol :

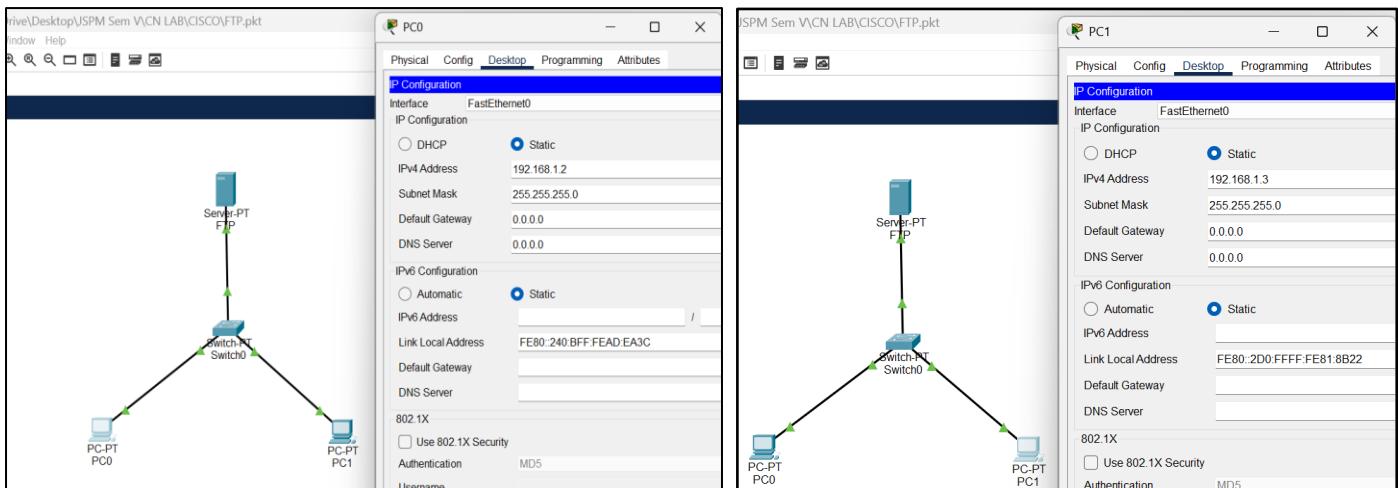
Step 1: Launch Packet Tracer.



Step 2: Build the topology (1 Server, 1 Switch and 2 PCs) .

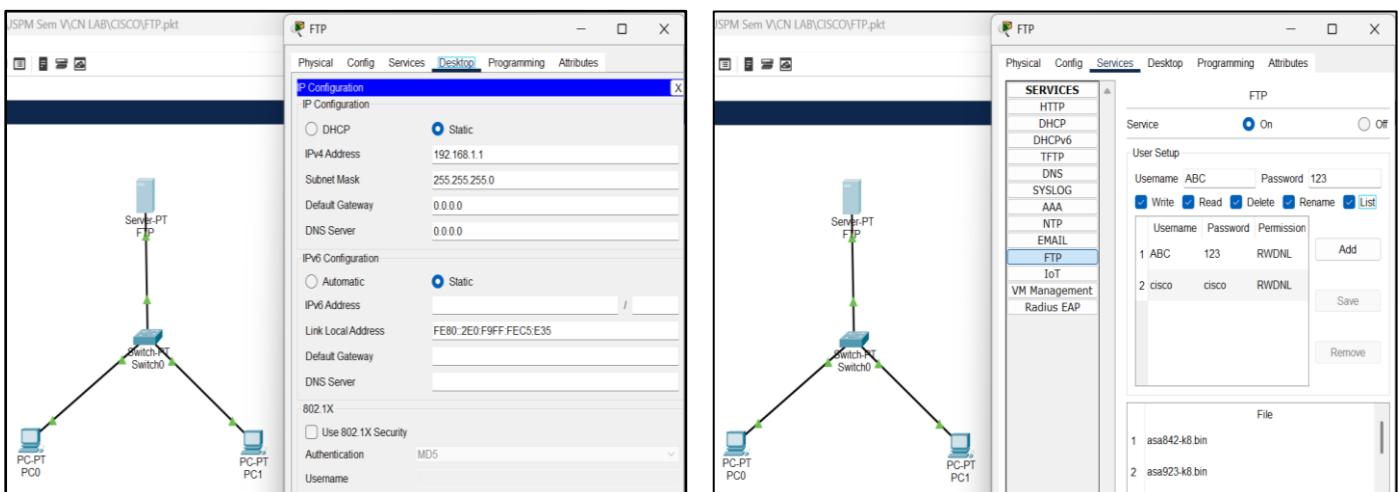


Step 3: Configure IP addresses and subnet masks on all PCs.

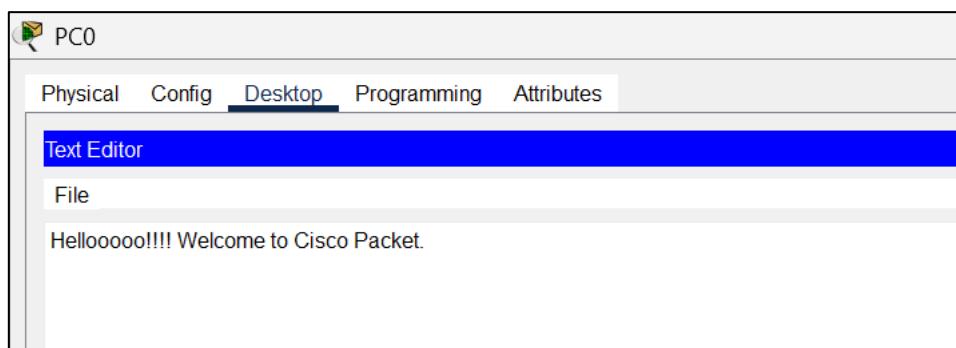


Step 4: Configure IP addresses and subnet masks on Server.

- Go to Services tab → select **FTP**.
- Turn **FTP ON** (enable the service).
- In the FTP settings you can **create users**. Add a user:
 - Username: ABC
 - Password: 123

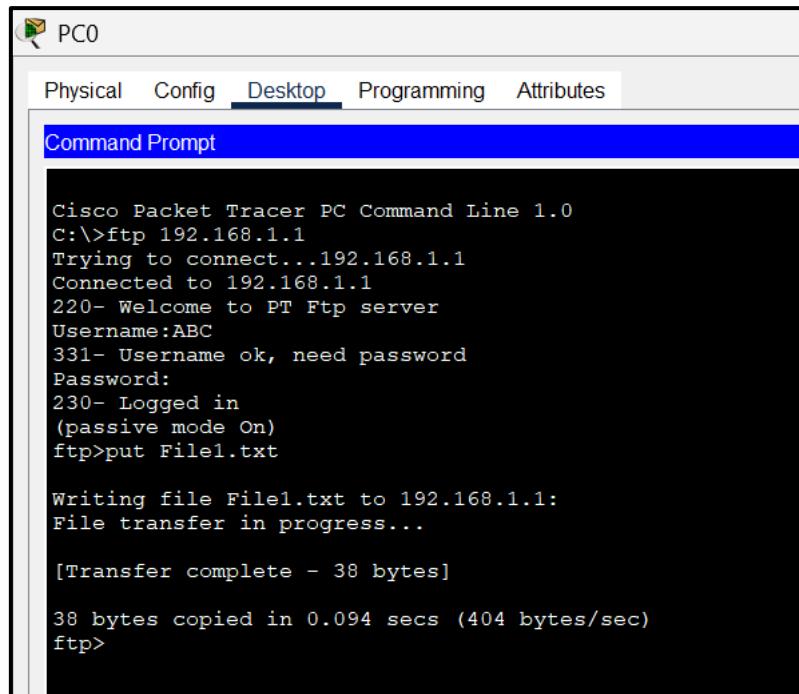


Step 5: Create text file on any PC.



Step 6 : Using FTP from PC.

Open PC0 → Desktop → Command Prompt and run:



The screenshot shows a window titled "PC0" with a menu bar containing "Physical", "Config", "Desktop" (which is underlined), "Programming", and "Attributes". Below the menu is a blue header bar with the text "Command Prompt". The main area of the window displays the following command-line session:

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ftp 192.168.1.1
Trying to connect...192.168.1.1
Connected to 192.168.1.1
220- Welcome to PT Ftp server
Username:ABC
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>put File1.txt

Writing file File1.txt to 192.168.1.1:
File transfer in progress...

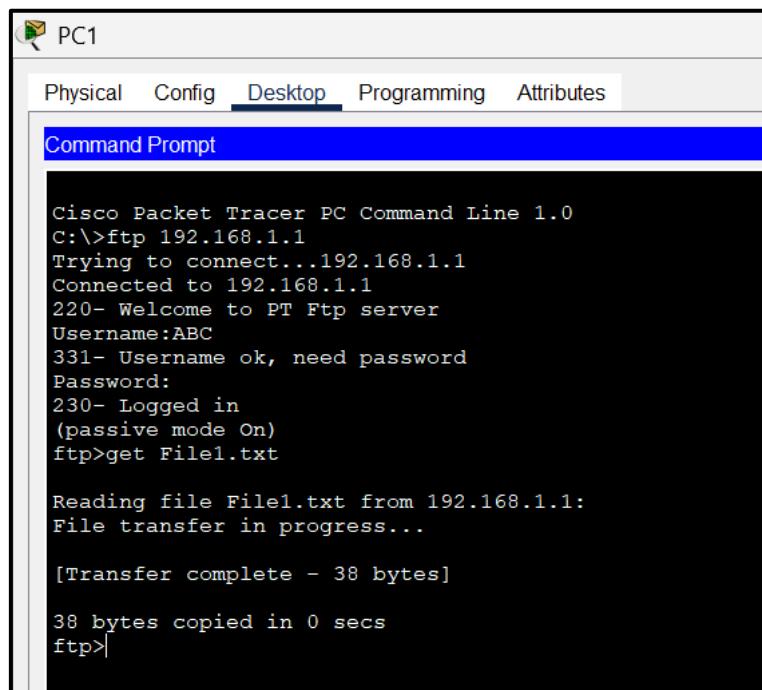
[Transfer complete - 38 bytes]

38 bytes copied in 0.094 secs (404 bytes/sec)
ftp>
```

Step 7 : Verify FTP Connection from PC1

Use FTP commands from another PC to ensure the connection and file transfer are working correctly.

Open PC1 → Desktop → Command Prompt and run :



The screenshot shows a window titled "PC1" with a menu bar containing "Physical", "Config", "Desktop" (which is underlined), "Programming", and "Attributes". Below the menu is a blue header bar with the text "Command Prompt". The main area of the window displays the following command-line session:

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ftp 192.168.1.1
Trying to connect...192.168.1.1
Connected to 192.168.1.1
220- Welcome to PT Ftp server
Username:ABC
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>get File1.txt

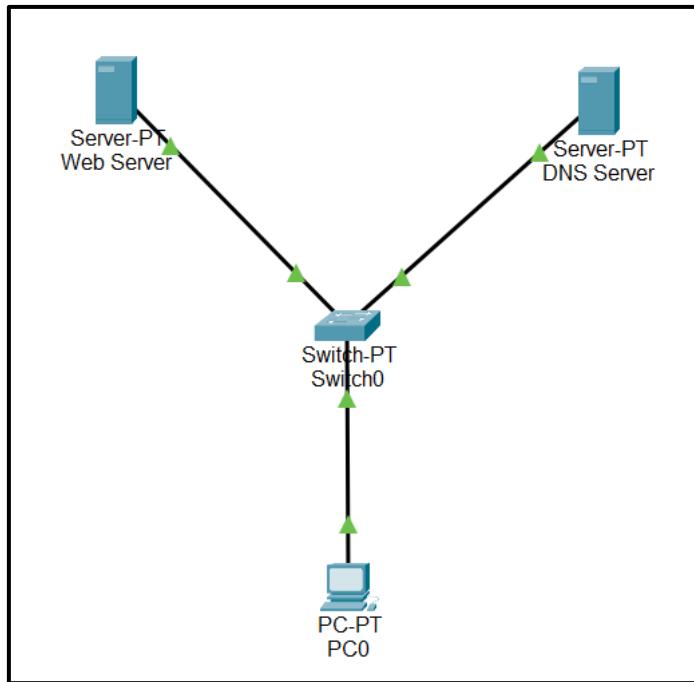
Reading file File1.txt from 192.168.1.1:
File transfer in progress...

[Transfer complete - 38 bytes]

38 bytes copied in 0 secs
ftp>
```

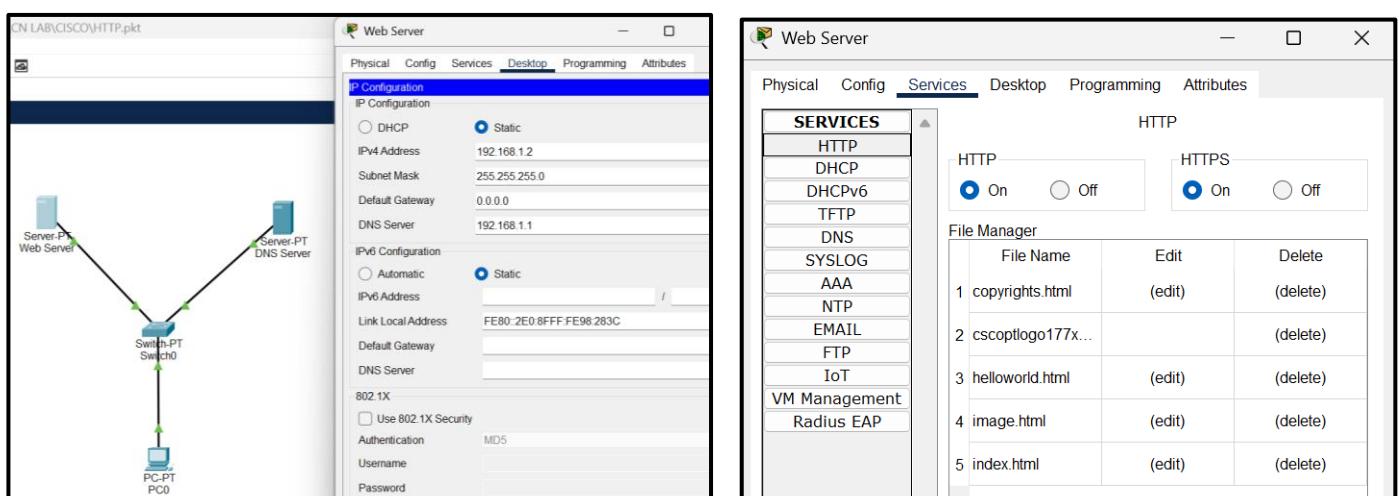
HTTP and HTTPS Protocol :

Step 1: Build the topology (1 Server, 1 Switch and 2 PCs) .



Step 2: Click Web Server → Desktop → IP Configuration

- IP Address: 192.168.1.2
 - Subnet Mask: 255.255.255.0
1. Go to Services → HTTP → Turn HTTP ON
 2. Go to Services → HTTPS → Turn HTTPS ON and
 3. Edit index.html page content under HTTP service



The screenshot shows two windows of the Cisco Web Server interface. The left window displays the network topology. The right window has two tabs: "IP Configuration" and "Services".

IP Configuration Tab (Left Window):

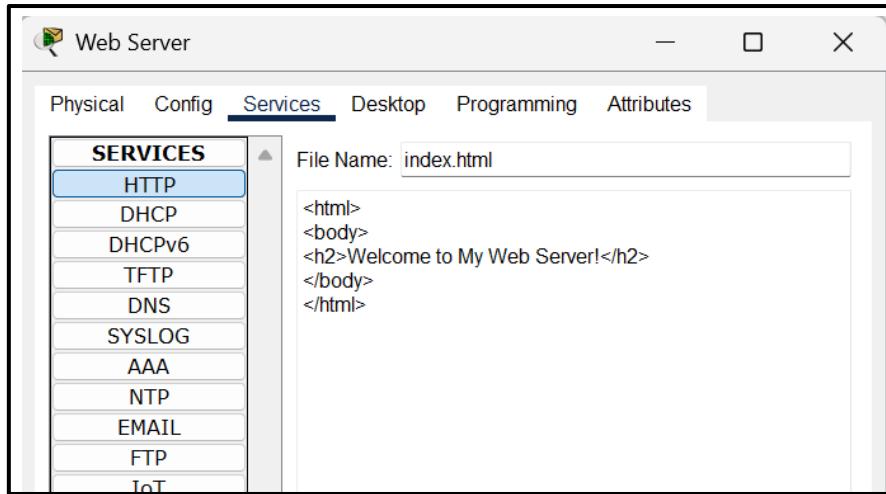
IP Configuration	
IP Configuration	
<input type="radio"/> DHCP	<input checked="" type="radio"/> Static
IPv4 Address	192.168.1.2
Subnet Mask	255.255.255.0
Default Gateway	0.0.0.0
DNS Server	192.168.1.1

Services Tab (Right Window):

SERVICES	
HTTP	<input checked="" type="radio"/> On <input type="radio"/> Off
DHCP	<input type="radio"/> On <input checked="" type="radio"/> Off
DHCIPv6	<input type="radio"/> On <input checked="" type="radio"/> Off
TFTP	<input type="radio"/> On <input checked="" type="radio"/> Off
DNS	<input type="radio"/> On <input checked="" type="radio"/> Off
SYSLOG	<input type="radio"/> On <input checked="" type="radio"/> Off
AAA	<input type="radio"/> On <input checked="" type="radio"/> Off
NTP	<input type="radio"/> On <input checked="" type="radio"/> Off
EMAIL	<input type="radio"/> On <input checked="" type="radio"/> Off
FTP	<input type="radio"/> On <input checked="" type="radio"/> Off
IoT	<input type="radio"/> On <input checked="" type="radio"/> Off
VM Management	<input type="radio"/> On <input checked="" type="radio"/> Off
Radius EAP	<input type="radio"/> On <input checked="" type="radio"/> Off

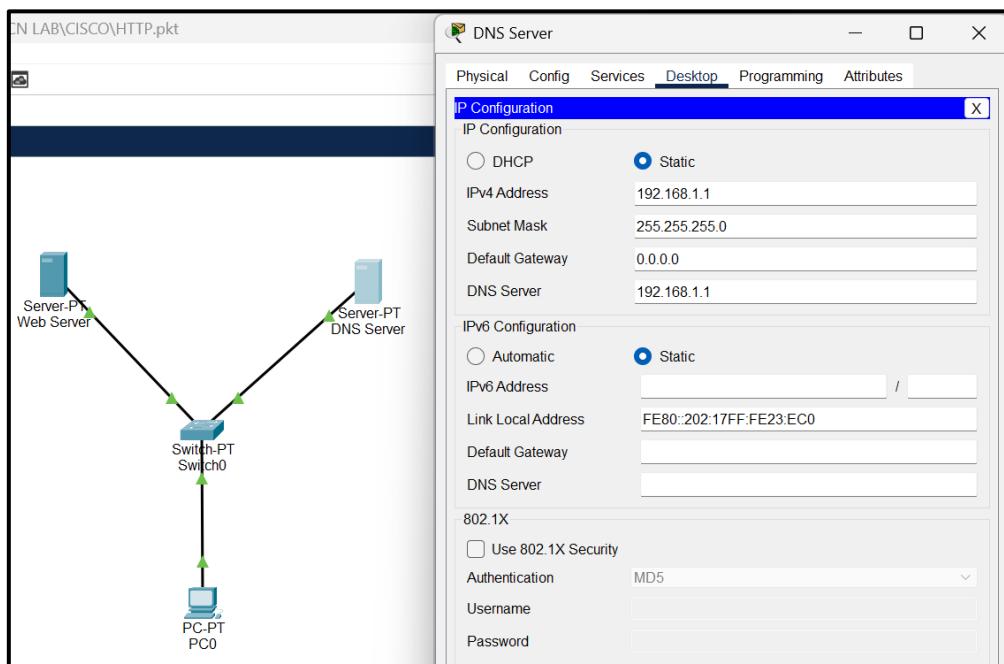
File Manager (Right Window):

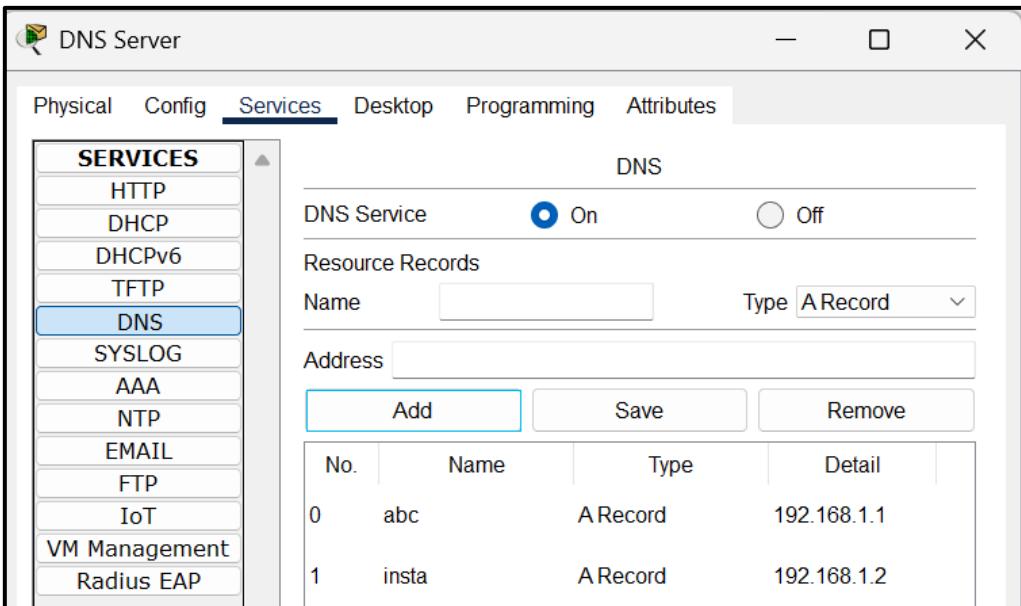
File Name	Edit	Delete
1 copyrights.html	(edit)	(delete)
2 cscoptlogo177x...		
3 helloworld.html	(edit)	(delete)
4 image.html	(edit)	(delete)
5 index.html	(edit)	(delete)



Step 4: Configure DNS Server

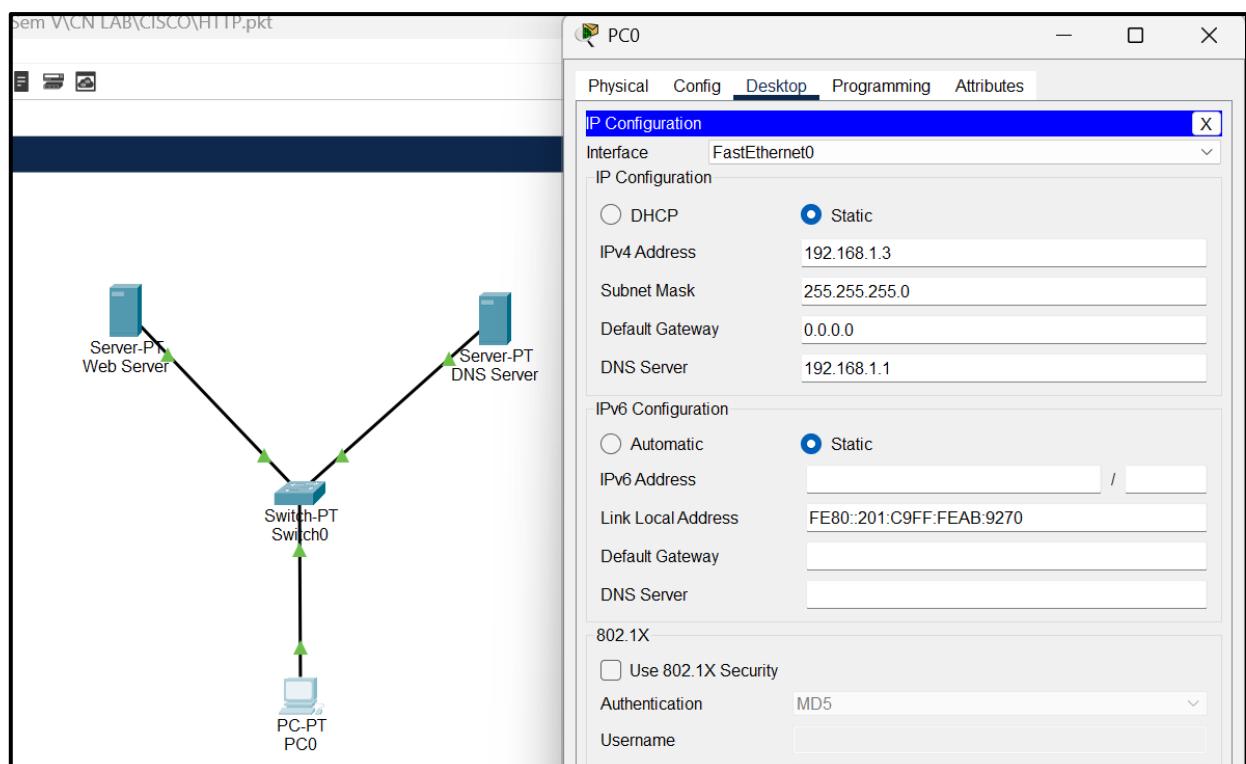
1. Click Server1 → Desktop → IP Configuration.
 - IP Address: 192.168.1.1
 - Subnet Mask: 255.255.255.0
2. Go to Services → DNS
 - Turn DNS ON
 - Under “Name” enter: Insta
 - Under “Address” enter: 192.168.1.2
 - Click Add





Step 5: Click PC0 → Desktop → IP Configuration

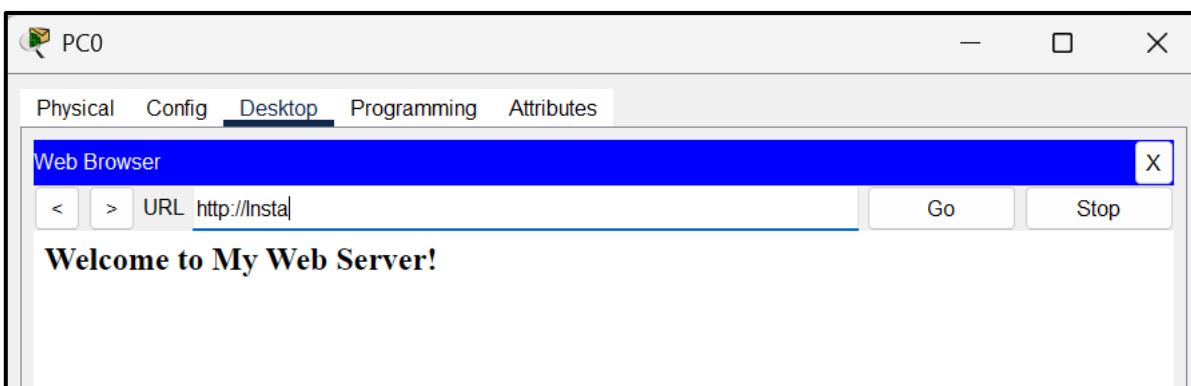
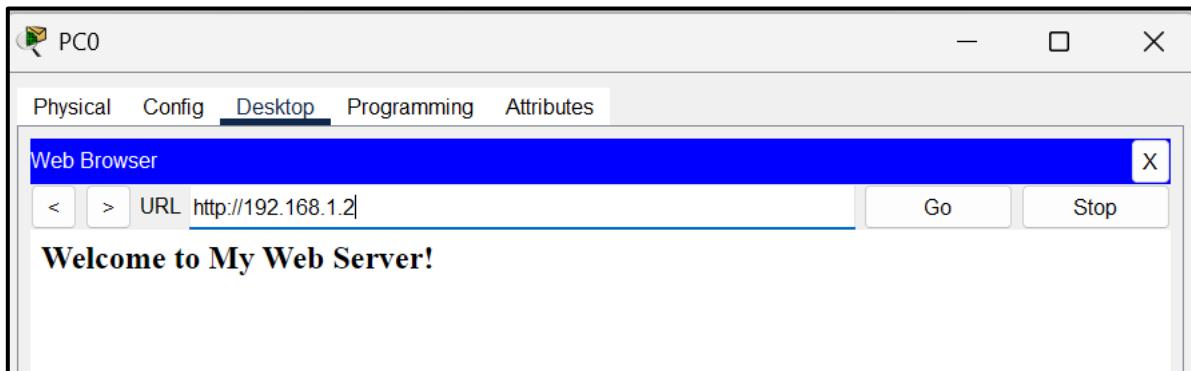
- IP Address: 192.168.1.3
- Subnet Mask: 255.255.255.0
- DNS Server: 192.168.1.1



Step 6 : Verify HTTPS Connection from PC0

Use the Web Browser to test the secure connection.

1. On PC0 → Desktop → Web Browser
2. Type:



Experiment No : 12

.....

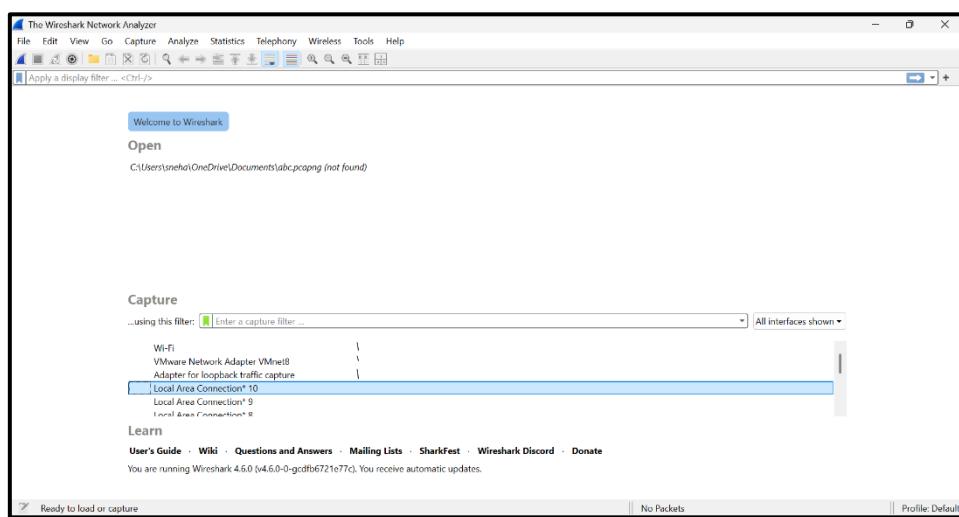
Roll No : TE-43

Name : Sanika Nilesh Patil

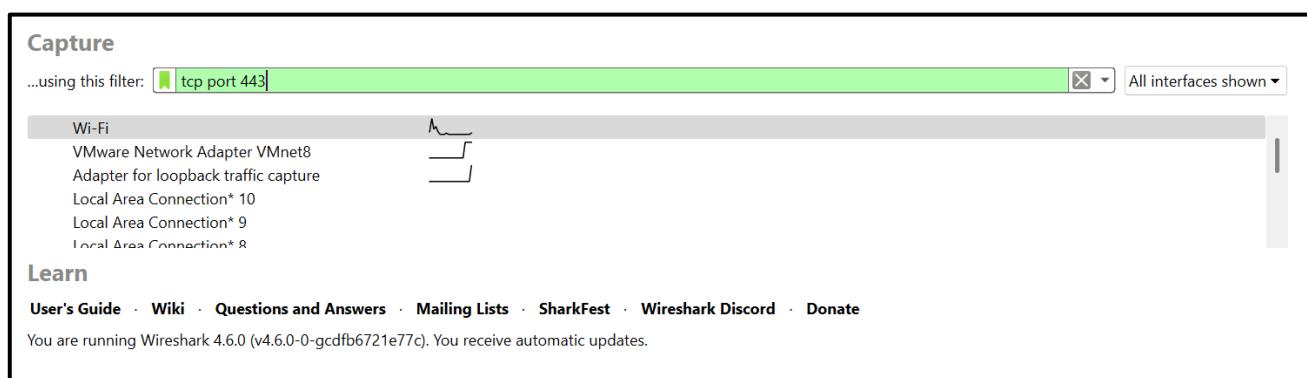
Title : To study the SSL (Secure Sockets Layer) protocol by capturing packets using Wireshark while visiting any SSL-secured website (e.g., online banking, e-commerce, or secure login pages).

.....

Step 1: Launch Wireshark.



Step 2: Capture filter → TCP port 443.



Capturing from Wi-Fi (tcp port 443)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	2401:4900:c0bb:f60d..	2606:4700:90d0:70fa..	TCP	75	55372 → 443 [ACK] Seq=1 Ack=1 Win=255 Len=1
2	0.155961	2606:4700:90d0:70fa..	2401:4900:c0bb:f60d..	TCP	86	443 → 55372 [ACK] Seq=1 Ack=2 Win=18 Len=0 SLE=1 SRE=2
3	2.161630	2606:4700:90d1:d8b7..	2401:4900:c0bb:f60d..	TLSv1.2	98	Application Data
4	2.162375	2401:4900:c0bb:f60d..	2606:4700:90d1:d8b7..	TLSv1.2	102	Application Data
5	2.228690	2606:4700:90d1:d8b7..	2401:4900:c0bb:f60d..	TCP	74	443 → 50802 [ACK] Seq=25 Ack=29 Win=19 Len=0
6	7.880288	2401:4900:c0bb:f60d..	2620:1ec:bdf:72	TCP	75	58248 → 443 [ACK] Seq=1 Ack=1 Win=251 Len=1
7	7.836047	2620:1ec:bdf:72	2401:4900:c0bb:f60d..	TLSv1.2	113	Application Data
8	7.835647	2620:1ec:bdf:72	2401:4900:c0bb:f60d..	TLSv1.2	98	Application Data
9	7.835647	2620:1ec:bdf:72	2401:4900:c0bb:f60d..	TCP	98	[TCP Retransmission] 443 → 58248 [FIN, PSH, ACK] Seq=40 Ack=2 Win=97 Len=24
10	7.836308	2401:4900:c0bb:f60d..	2620:1ec:bdf:72	TCP	86	58248 → 443 [ACK] Seq=2 Ack=65 Win=251 Len=0 SLE=40 SRE=64
11	7.835685	2401:4900:c0bb:f60d..	2620:1ec:bdf:72	TCP	74	58248 → 443 [FIN, ACK] Seq=2 Ack=65 Win=251 Len=0

```
> Frame 1: Packet, 75 bytes on wire (600 bits), 75 bytes captured
> Ethernet II, Src: AzureWave_2:db:47 (34:6f:24:2d:db:47), Dst: b:  

> Internet Protocol Version 6, Src: 2401:4900:c0bb:f60d:f40f:3c13  

> Transmission Control Protocol, Src Port: 55372, Dst Port: 443, :  

0000  be dd 7b 75 69 a3 34 6f 24 2d db 47 86 dd 60 0a  ..{uv-4o $-G`.  

0010 46 b5 00 15 06 3f 24 01 49 00 c0 bb f6 0d f4 0f F.....?$. I.....  

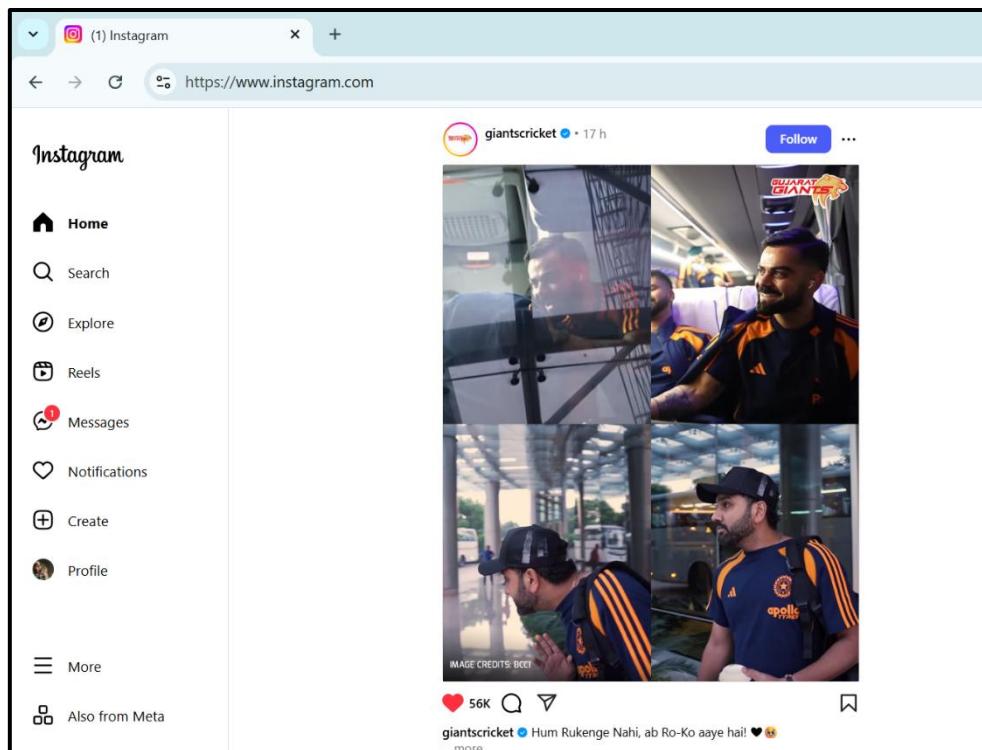
0020 3c 13 82 00 bc c6 26 06 47 00 90 d0 70 fa ce ca <.....& G..p...  

0030 05 ca b6 d3 4c d4 d8 4c 01 bb 8c 7c 1c 30 f1 e4 .....L-L ...].0..  

0040 23 a0 50 10 00 ff 3a d8 00 00 00 # P.....
```

Step 3: Open web browser and visit site HTTPs.

Wait to load page and perform some activity on page.



Step 4: Go to Wireshark and type TLS to view TLS/SSL packets.

Capturing from Wi-Fi (tcp port 443)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter... <Ctrl-/>

tls

No.	Time	Source	Destination	Protocol	Length	Info
3	2.161630	2606:4700:90d1:d8b7..	2401:4900:c0bb:f60d..	TLSv1.2	98	Application Data
4	2.162375	2401:4900:c0bb:f60d..	2606:4700:90d1:d8b7..	TLSv1.2	102	Application Data
7	7.836047	2620:1ec:bdf:72	2401:4900:c0bb:f60d..	TLSv1.2	113	Application Data
8	7.836047	2620:1ec:bdf:72	2401:4900:c0bb:f60d..	TLSv1.2	98	Application Data
14	22.334954	2606:4700:90d1:d8b7..	2401:4900:c0bb:f60d..	TLSv1.2	98	Application Data
15	22.335977	2401:4900:c0bb:f60d..	2606:4700:90d1:d8b7..	TLSv1.2	102	Application Data
18	23.457251	10.222.209.22	4.195.15.137	TLSv1.2	112	Application Data
19	23.645971	4.195.15.137	10.222.209.22	TLSv1.2	101	Application Data
24	27.874792	10.222.209.22	52.137.106.217	TLSv1.2	268	Client Hello (SNI=settings-win.data.microsoft.com)
27	28.133273	52.137.106.217	10.222.209.22	TLSv1.2	1092	Server Hello, Certificate, Server Key Exchange, Server Hello Done
29	28.140180	10.222.209.22	52.137.106.217	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

Step 5: Observe Client Hello, Server Hello and Certificate message in packet list.

Click on packet to see details :

1. TLS Version
2. Cipher suits
3. Certificate Info

Capturing from Wi-Fi (tcp port 443)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tls

No.	Time	Source	Destination	Protocol	Length	Info
7	7.836047	2620:1ec:bdf::72	2401:4900:c0bb:f60d...	TLSv1.2	113	Application Data
8	7.836047	2620:1ec:bdf::72	2401:4900:c0bb:f60d...	TLSv1.2	98	Application Data
14	22.334954	2606:4700:90d1:d8b7...	2401:4900:c0bb:f60d...	TLSv1.2	98	Application Data
15	22.335977	2401:4900:c0bb:f60d...	2606:4700:90d1:d8b7...	TLSv1.2	102	Application Data
18	23.457251	10.222.209.22	4.195.15.137	TLSv1.2	112	Application Data
19	23.645971	4.195.15.137	10.222.209.22	TLSv1.2	101	Application Data
24	27.874792	10.222.209.22	52.137.106.217	TLSv1.2	268	Client Hello (SNI=Settings-win.data.microsoft.com)
27	28.133273	52.137.106.217	10.222.209.22	TLSv1.2	1092	Server Hello, Certificate, Server Key Exchange, Server Hello Done
29	28.140180	10.222.209.22	52.137.106.217	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
30	28.378217	52.137.106.217	10.222.209.22	TLSv1.2	185	Change Cipher Spec, Encrypted Handshake Message
31	28.378217	52.137.106.217	10.222.209.22	TLSv1.2	123	Application Data

> Frame 24: Packet, 268 bytes on wire (2144 bits), 268 bytes captured
> Ethernet II, Src: AzureWav_2d:db:47 (34:6f:24:2d:db:47), Dst: b...
> Internet Protocol Version 4, Src: 10.222.209.22, Dst: 52.137.10...
> Transmission Control Protocol, Src Port: 58249, Dst Port: 443, ...
> Transport Layer Security

Hex	Dec	ASCII
0000	be dd 7b 75 76 9a 34 6f	..{uv\4o \$- G-E-
0010	00 fe 30 4d 40 00 80 06	..0M@... NV...4-
0020	6a d9 e3 89 01 bb 9b e9	j.....<[P-
0030	00 ff b0 b4 00 00 16 03
0040	03 68 f0 6a 51 e0 f2 75	a8 d5 4a fb 97 d4 24 dd
0050	98 e6 dd 4c 57 30 5f 05	h-jQ-u ..J-\$
0060	ec 00 00 24 c0 2c c0 2b	LW0_ fe &
0070	c0 28 c0 27 c0 0a c0 09	\$, + 0 / \$ #
0080	00 3d 00 3c 00 35 00 2f	(..,.....
0090	00 22 00 00 1f 73 65 74	=< 5 /\$
00a0	6e 2e 64 61 74 61 2e 6d	"...set tings-wi
00b0	2e 63 6f 6d 00 05 00 05	n.data.m icrosoft
00c0	08 00 06 00 1d 00 17 00	.com.....
00d0	0d 00 1a 00 18 08 04 08
00e0	01 04 03 05 03 02 03 02	#.
00f0	00 00 10 00 0e 00 9c 02	h2 http/
0100	31 2e 31 00 17 00 00 ff	1.1.....

Experiment No : 14

.....

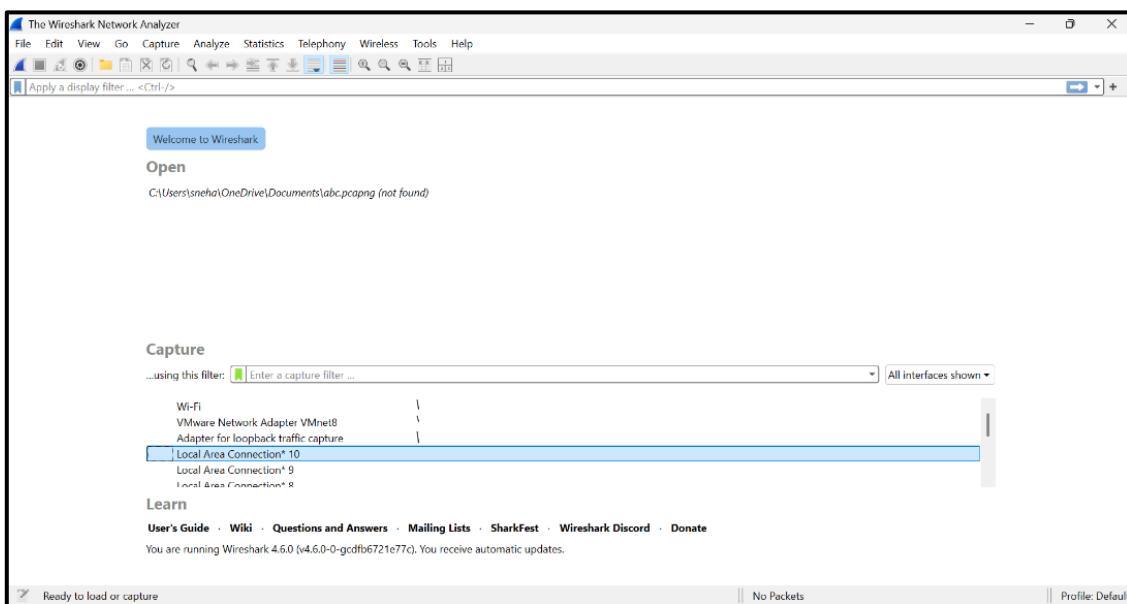
Roll No : TE-43

Name : Sanika Nilesh Patil

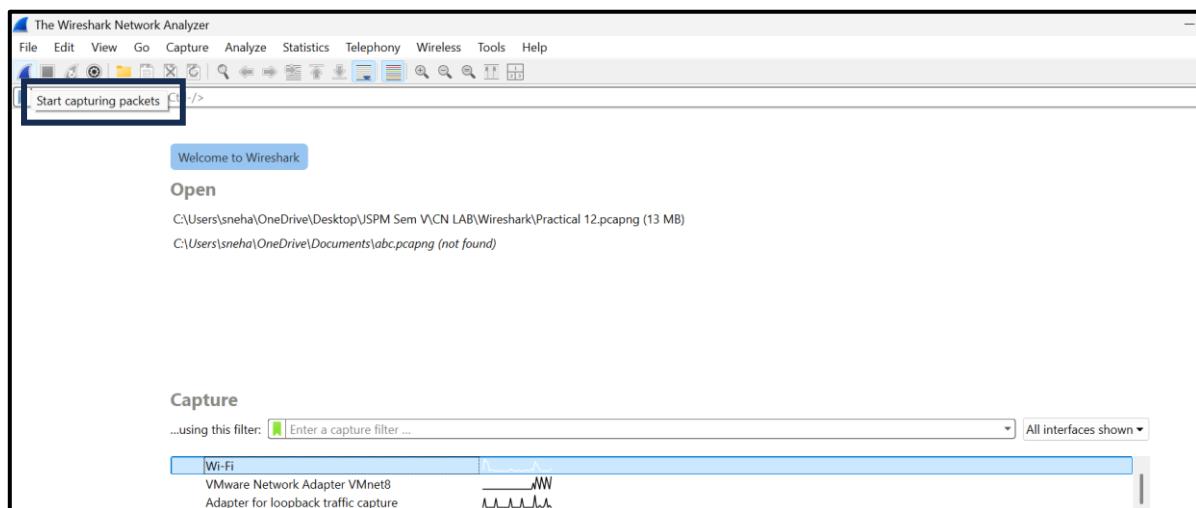
Title : To study the IPsec (Internet Protocol Security) protocol by capturing packets using Wireshark while observing ESP (Encapsulating Security Payload) and AH (Authentication Header) headers during secure communication.

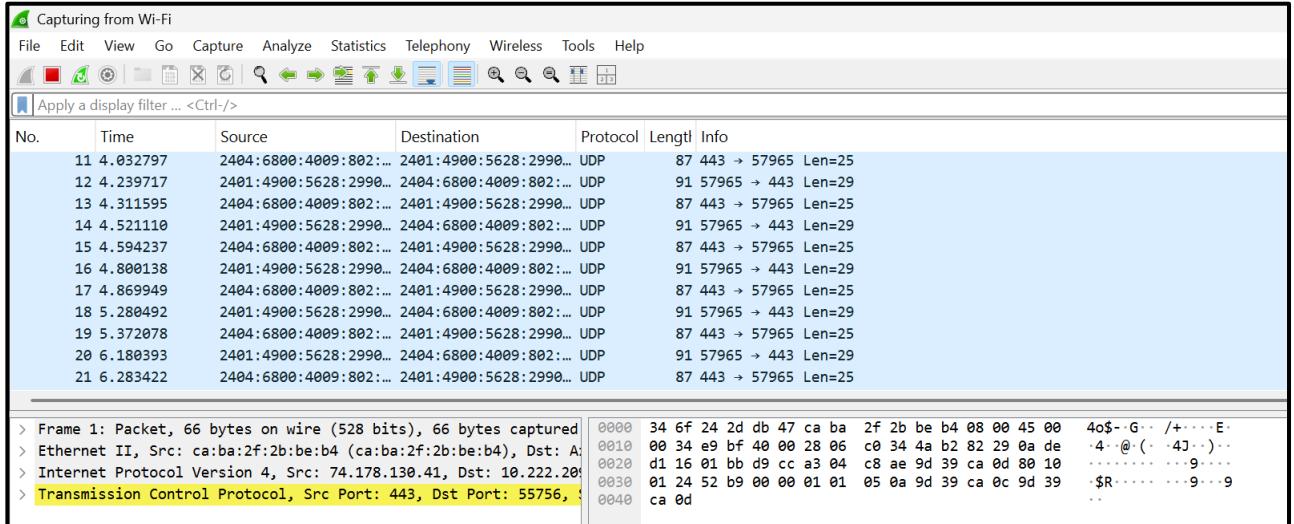
.....

Step 1: Launch Wireshark.



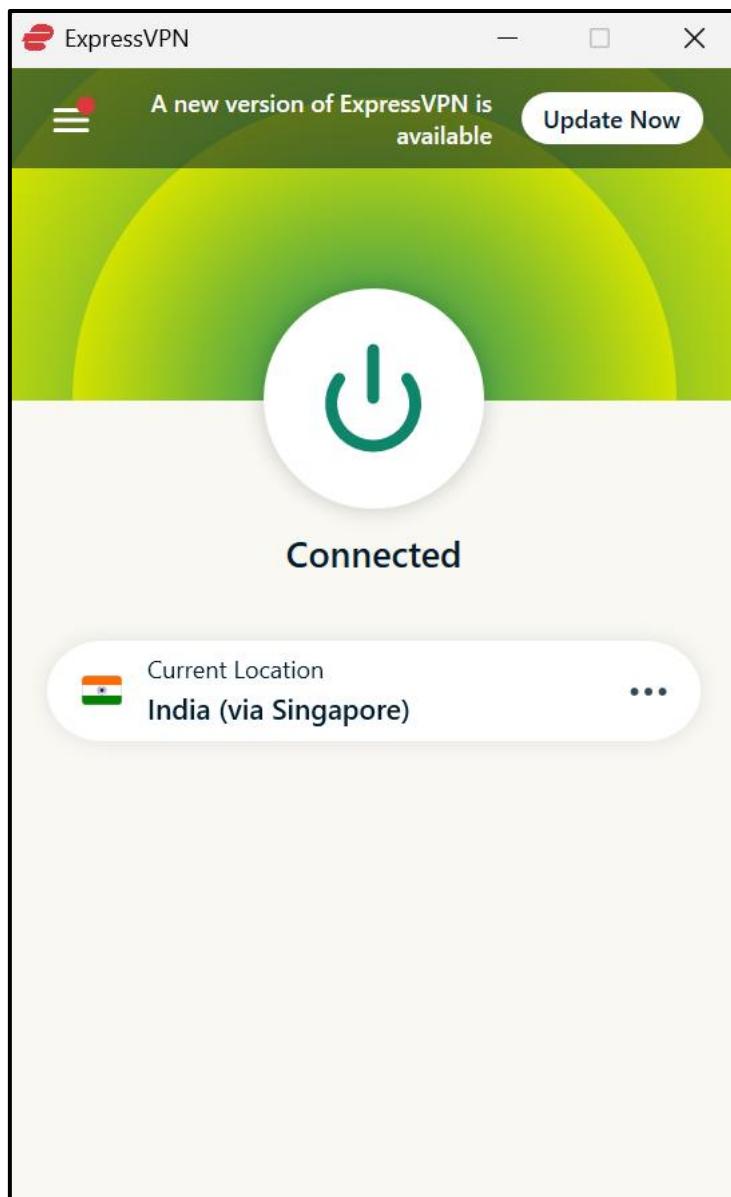
Step 2: Select network interface (Wi-Fi) and click on start capturing packets.





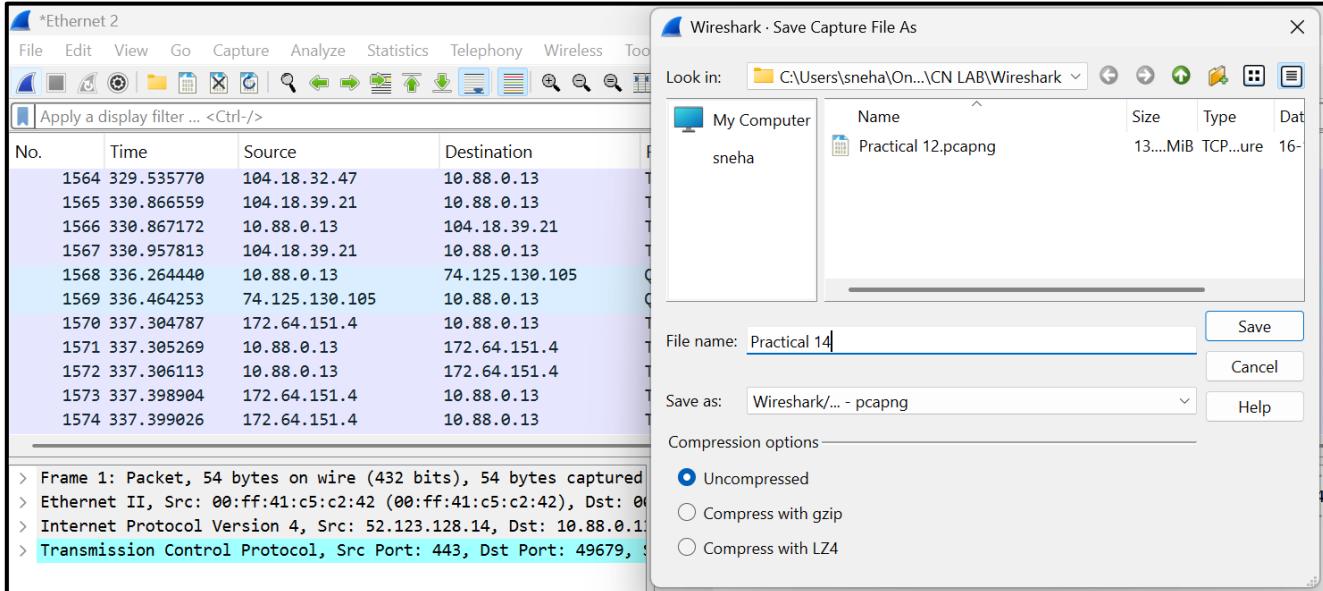
Step 3: Establish IPSec Connection (Connect to ExpressVPN).

1. Open ExpressVPN and choose a server(any server location).
2. While the VPN is connecting , Wireshark will automatically capture all network packets.



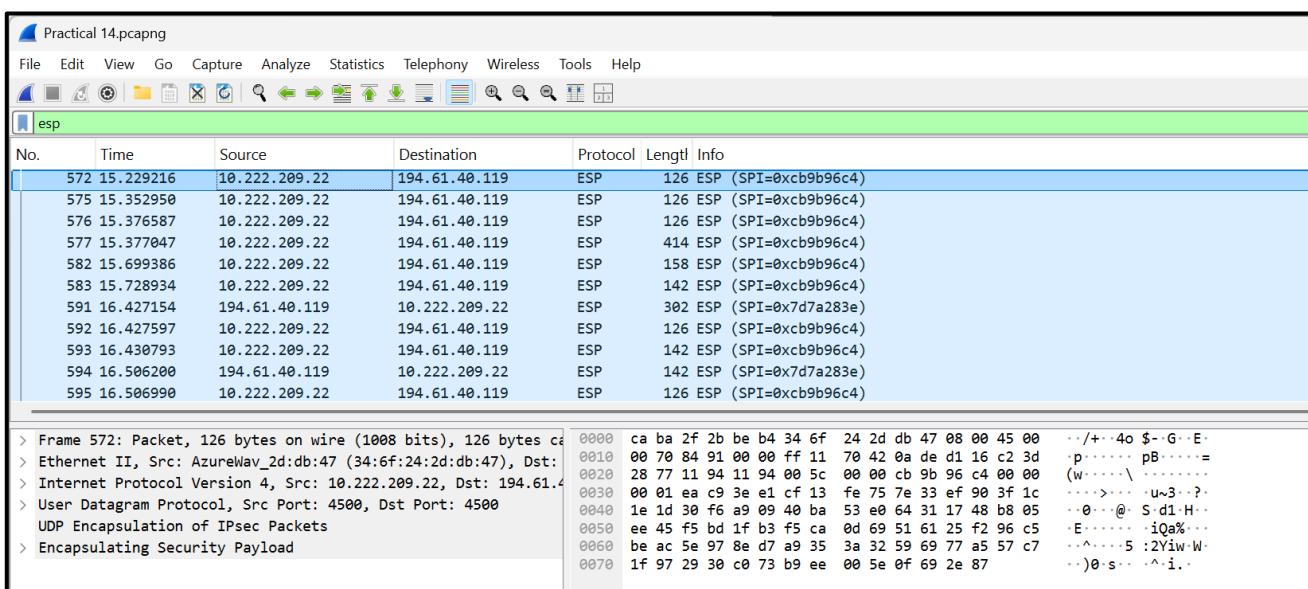
Step 4: Stop capture packets.

1. After 30–60 seconds of connection (or once traffic appears in Wireshark), click the Red Stop button.
2. Save the capture:
File → Save As → practical_14.pcapng



Step 5: Apply display filters in Wireshark.

1. For all IPSec traffic – esp or ah.



2. VPN negotiation packets : udp.port == 500 or udp.port == 4500

Practical 14.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

udp.port == 500 or udp.port == 4500

No.	Time	Source	Destination	Protocol	Length	Info
558	14.247781	194.61.40.119	10.222.209.22	ISAKMP	1282	IKE_AUTH MID=01 Responder Response (fragment 1/3)
559	14.247781	194.61.40.119	10.222.209.22	ISAKMP	1282	IKE_AUTH MID=01 Responder Response (fragment 2/3)
560	14.247781	194.61.40.119	10.222.209.22	ISAKMP	1010	IKE_AUTH MID=01 Responder Response (fragment 3/3)
561	14.307896	10.222.209.22	194.61.40.119	ISAKMP	142	IKE_AUTH MID=02 Initiator Request
562	14.472047	194.61.40.119	10.222.209.22	ISAKMP	158	IKE_AUTH MID=02 Responder Response
563	14.478268	10.222.209.22	194.61.40.119	ISAKMP	190	IKE_AUTH MID=03 Initiator Request
564	14.762954	194.61.40.119	10.222.209.22	ISAKMP	174	IKE_AUTH MID=03 Responder Response
565	14.767206	10.222.209.22	194.61.40.119	ISAKMP	126	IKE_AUTH MID=04 Initiator Request
568	15.035504	194.61.40.119	10.222.209.22	ISAKMP	126	IKE_AUTH MID=04 Responder Response
569	15.039333	10.222.209.22	194.61.40.119	ISAKMP	158	IKE_AUTH MID=05 Initiator Request
570	15.157366	194.61.40.119	10.222.209.22	ISAKMP	398	IKE_AUTH MID=05 Responder Response

```
> Frame 572: Packet, 126 bytes on wire (1008 bits), 126 bytes captured
> Ethernet II, Src: AzureWav_2d:db:47 (34:6f:24:2d:db:47), Dst: 
> Internet Protocol Version 4, Src: 10.222.209.22, Dst: 194.61.40.119
> User Datagram Protocol, Src Port: 4500, Dst Port: 4500
  UDP Encapsulation of IPsec Packets
> Encapsulating Security Payload
0000 ca ba 2f 2b be b4 34 6f 24 2d db 47 08 00 45 00 .. /+.. 4o $-·G··E·
0010 00 70 84 91 00 00 ff 11 70 42 0a de d1 16 c2 3d ·p..... pB.....=
0020 28 77 11 94 11 94 00 5c 00 00 cb 9b 96 c4 00 00 (w.....\.....·
0030 00 01 ea c9 3e e1 cf 13 fe 75 7e 33 ef 90 3f 1c .....>... u~3..?·
0040 1e 1d 30 f6 a9 09 40 ba 53 e0 64 31 17 48 b8 05 ..0.. @ S d1 H··
0050 ee 45 f5 bd 1f b3 f5 ca 0d 69 51 61 25 f2 96 c5 ·E..... iq@%··
0060 be ac 5e 97 8e d7 a9 35 3a 32 59 69 77 a5 57 c7 ..^.... 5 :2Yin·W··
0070 1f 97 29 30 c0 73 b9 ee 00 5e 0f 69 2e 87 ..)0·s.. ^·i..
```

Experiment No : 15

"""

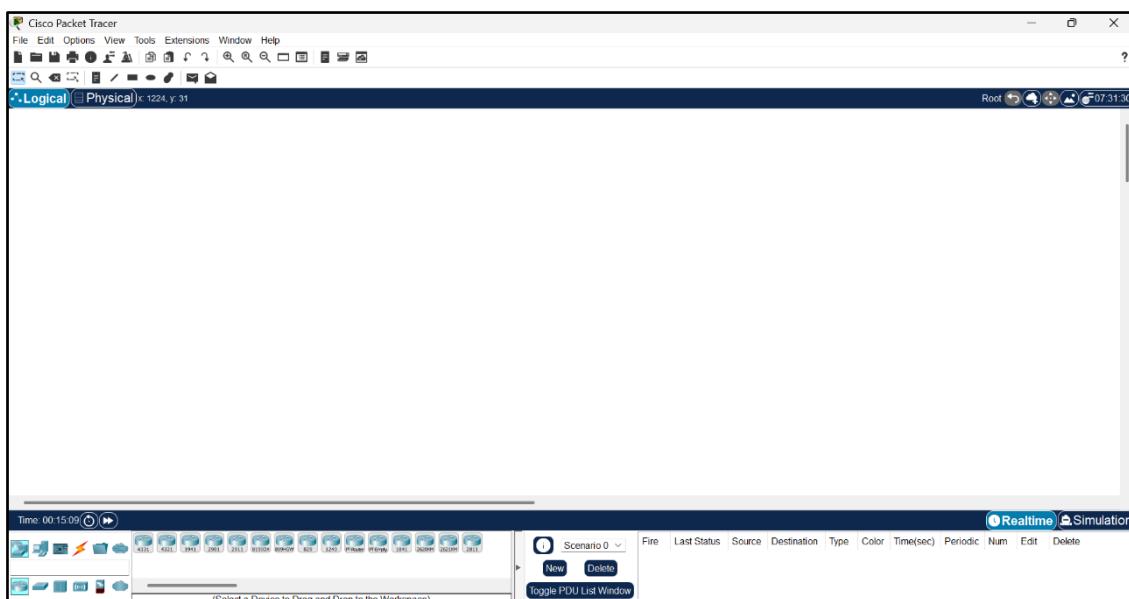
Roll No : TE-43

Name : Sanika Nilesh Patil

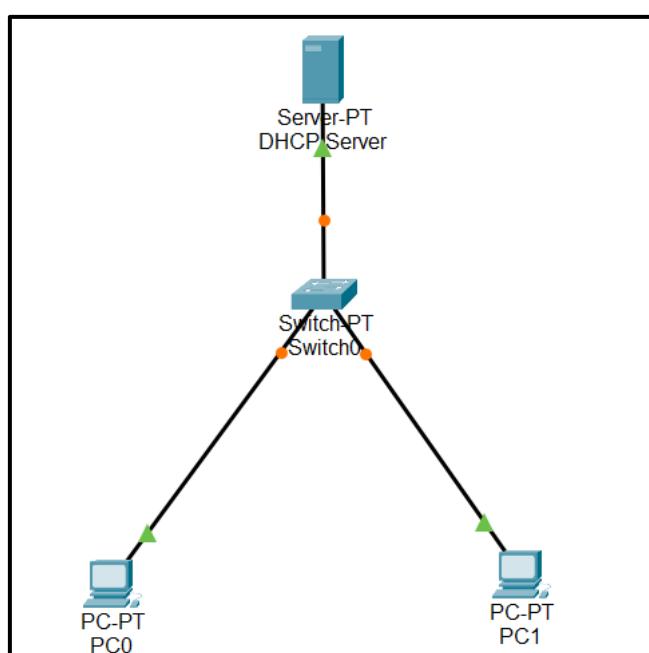
Title : To install and configure a DHCP (Dynamic Host Configuration Protocol) Server and assign IP addresses automatically to client machines connected in a network.

"""

Step 1: Launch Packet Tracer.

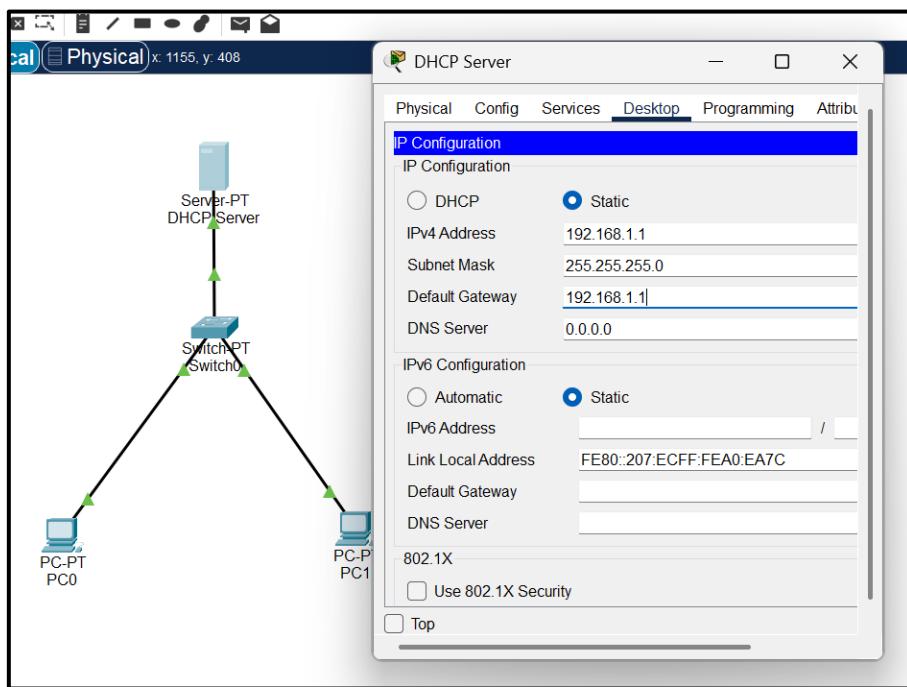


Step 2: Build the topology.



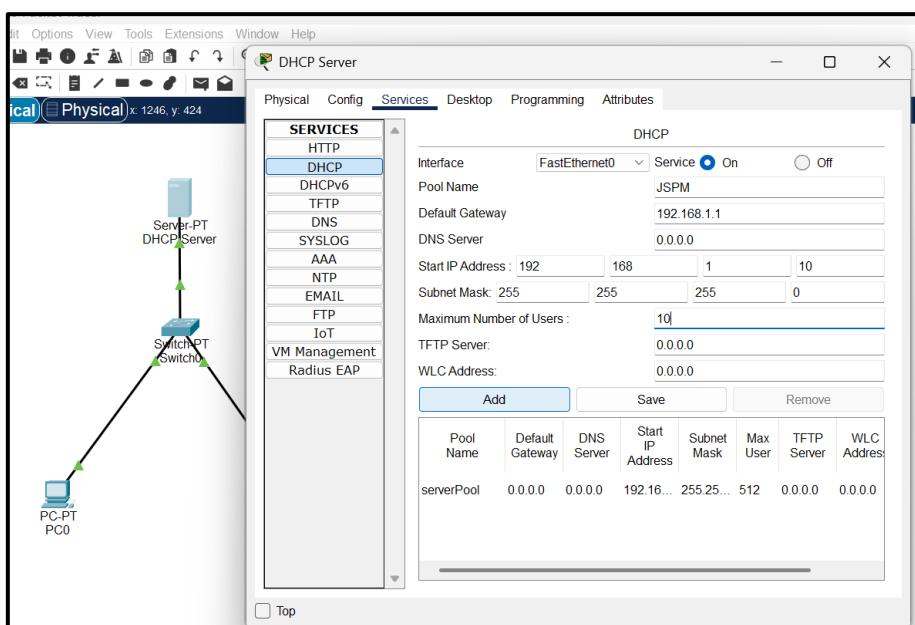
Step 3: Configure DHCP Server.

1. Assign IP address to DHCP server.



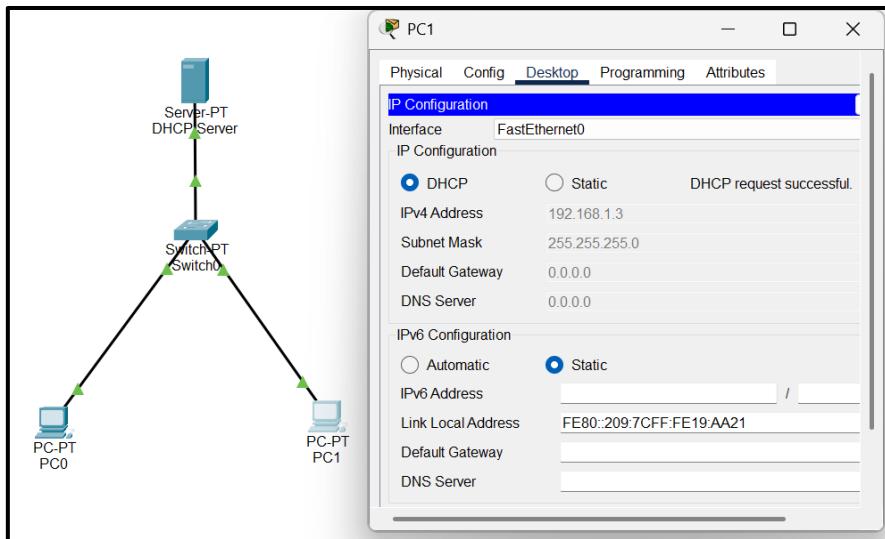
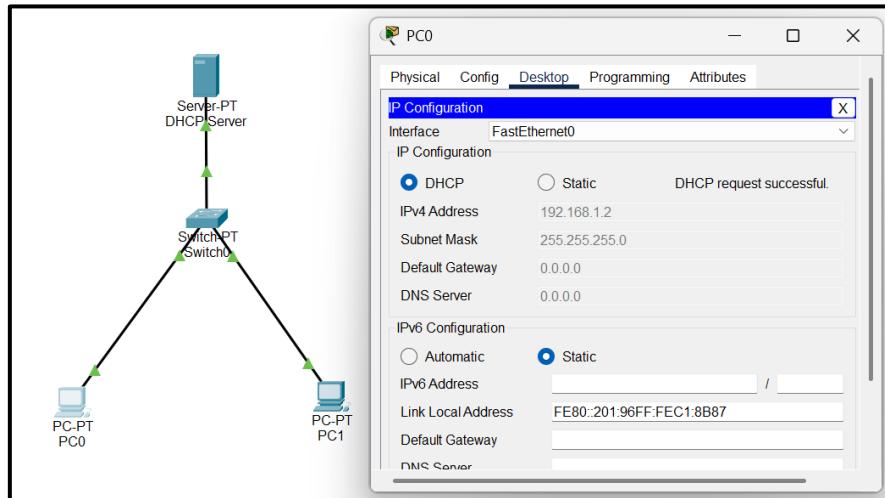
2. Activate DHCP service from service tab.

- Pool name : JSPM
- Default Gateway : 192.168.1.1
- Start IP address : 192.168.1.10
- Maximum Number of users : 10



Step 4: Configure IP addresses on all PCs.

Click on PC → Desktop → IP configuration → DHCP



Step 5: Check connectivity using PING command.

The screenshot shows the Command Prompt window for PC0. The user has entered the command "ping 192.168.1.1". The output shows four successful replies from the server, followed by ping statistics: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Approximate round trip times in milli-seconds: Minimum = 0ms, Maximum = 1ms, Average = 0ms.

```
Cisco 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=1ms TTL=128
Reply from 192.168.1.1: bytes=32 time<1ms TTL=128
Reply from 192.168.1.1: bytes=32 time<1ms TTL=128
Reply from 192.168.1.1: bytes=32 time<1ms TTL=128

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 = 1ms, Average = 0ms

C:>|
```

Experiment No : 16

"""

Roll No : TE-43

Name : Sanika Nilesh Patil

Title : Write a program for DNS lookup. Given an IP address input, it should return URL and vice versa.

"""

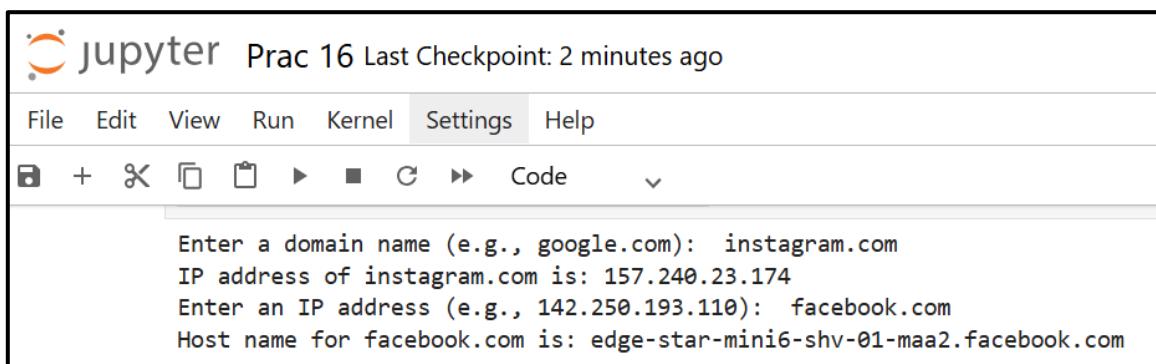
Source Code :

```
import socket

# Forward lookup (URL to IP)
url = input("Enter a domain name (e.g., google.com): ")
try:
    ip = socket.gethostbyname(url)
    print(f"IP address of {url} is: {ip}")
except socket.gaierror:
    print("Invalid domain name")

# Reverse lookup (IP to URL)
ip_input = input("Enter an IP address (e.g., 142.250.193.110): ")
try:
    host = socket.gethostbyaddr(ip_input)
    print(f"Host name for {ip_input} is: {host[0]}")
except socket.herror:
    print("Host not found for given IP")
```

Output :



The screenshot shows a Jupyter Notebook interface with the title "jupyter Prac 16 Last Checkpoint: 2 minutes ago". The menu bar includes File, Edit, View, Run, Kernel, Settings, and Help. Below the menu is a toolbar with icons for file operations like new, open, save, and run. The main notebook area contains the following text:

```
Enter a domain name (e.g., google.com): instagram.com
IP address of instagram.com is: 157.240.23.174
Enter an IP address (e.g., 142.250.193.110): facebook.com
Host name for facebook.com is: edge-star-mini6-shv-01-maa2.facebook.com
```
