Foundations of Computer Networks

(DSE-3/GE-5)

Practical File

Submitted By

Shivam Chaurasiya
Roll No.: 23294917129 (Batch: ECE-B (B2))
B. Tech Electronics and Communication Engineering
(Semester V)

To

Dr. Sunil Kumar
(Assistant Professor)
Department of Computer Science Engineering

FACULTY OF TECHNOLOGY
UNIVERSITY OF DELHI
NEW DELHI – 110007
(2025 – 2026)

Record of Experiments

S. No.	Experiment Title	Date	Remarks
1	To establish point-to-point connectivity between two PCs using Cisco Packet Tracer, assign IP addresses, and verify communication using the ping command.		
2	To design and implement a Mesh Topology using Cisco Packet Tracer, and verify connectivity using the ping command.		
3	To design and implement a Bus Topology using Cisco Packet Tracer, configure IP addresses for all PCs, and verify connectivity using the ping command.		
4	To design and implement a Star Topology using Cisco Packet Tracer, configure IP addresses for all PCs, and verify connectivity using the ping command.		
5	To configure and test a basic firewall on a Server in Cisco Packet Tracer using security rules that block or allow ICMP traffic.		
6	To design and implement two Local Area Networks (LANs) and connect them using a router in Cisco Packet Tracer. Configure IP addresses, default gateways, and verify connectivity using the ping command.		
7	To design and implement two separate LANs connected using two routers in Cisco Packet Tracer and configure static routing to enable communication between the networks.		
8	To design and implement three LANs and connect them in series (line topology) using three routers in Cisco Packet Tracer.		
9	To design and configure a secure VPN topology using Cisco Packet Tracer and verify connectivity using the ping command.		
10	To capture and analyze different types of network traffic (ICMP, TCP, HTTP, UDP) using Wireshark and study the color-coding rules for packet identification.		

Experiment 1
Point-to-Point Connectivity between Two PCs

1.0.1 Aim

To establish point-to-point connectivity between two PCs using Cisco Packet Tracer, assign IP addresses, and verify communication using the `ping` command.

1.0.2 Apparatus / Software

- **Hardware:** Personal Computer (2 Nos.)
- **Software:** Cisco Packet Tracer

1.0.3 Theory

A point-to-point (P2P) connection is the simplest form of network topology, where a dedicated link exists between exactly two devices. In a wired Ethernet network, this can be implemented by directly connecting the Network Interface Cards (NICs) of two PCs.

Traditionally, when connecting two similar devices (such as PC-to-PC or switch-to-switch), a **crossover cable** is used. A crossover cable internally swaps the transmit (TX) and receive (RX) pairs on one end so that the TX pins of one device are connected to the RX pins of the other, and vice versa.

Modern NICs often support **Auto-MDIX** (Automatic Medium Dependent Interface Crossover), which can automatically adjust to straight-through or crossover cabling. However, in this experiment we explicitly use a crossover cable in Cisco Packet Tracer to illustrate the classical requirement for direct PC-to-PC connections.

Once the physical link is established, both PCs must be assigned IP addresses from the same subnet (for example, `192.168.1.1` and `192.168.1.2` with subnet mask `255.255.255.0`). Connectivity is verified using the `ping` command, which sends ICMP Echo Request packets from one host to another. Successful replies indicate that Layer 1 (physical), Layer 2 (data link), and Layer 3 (network) configurations are all correct.

1.0.4 Procedure

1. **Place end devices**
    - Open Cisco Packet Tracer.
    - From the *End Devices* section, drag and drop two **PC** devices onto the workspace and rename them as **PC0** and **PC1** (optional).
2. **Connect PCs using crossover cable**
    - Click on the *Connections* (lightning bolt) icon.
    - Select **Copper Cross-Over** cable.
    - Click on **PC0** $\rightarrow$ choose `FastEthernet0`.

- Click on **PC1** → choose `FastEthernet0`.
- Wait until the link lights turn green on both ends, indicating a successful physical connection.

3. **Assign IP addresses**
   - Click on **PC0** → **Desktop** tab → **IP Configuration**.
   - Set:
     IP Address = 192.168.1.1, Subnet Mask = 255.255.255.0
   - Click on **PC1** → **Desktop** tab → **IP Configuration**.
   - Set:
     IP Address = 192.168.1.2, Subnet Mask = 255.255.255.0
4. **Verify connectivity using `ping`**
   - Open **PC0** → **Desktop** tab → **Command Prompt**.
   - Type:
     `ping 192.168.1.2`
     and press Enter.
   - Observe the ICMP Echo Request and Echo Reply messages.

1.0.5 Diagram / Topology

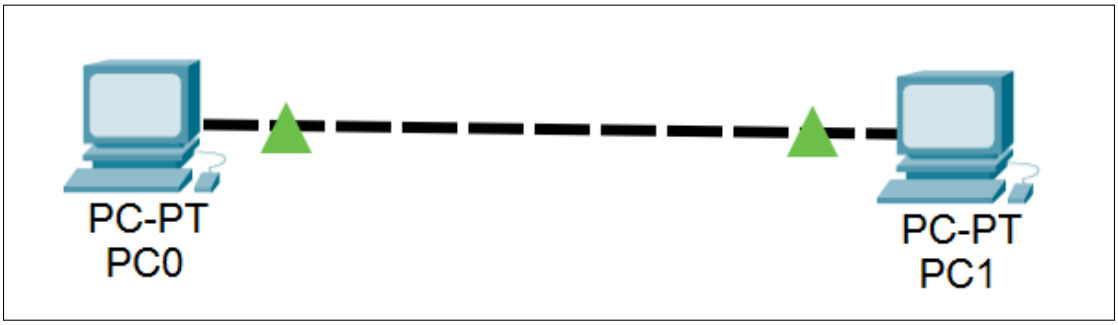

Figure 1.1: Point-to-point topology between two PCs using a crossover cable

1.0.6 Results

The connectivity test was successful. The `ping` command sent 4 ICMP Echo Request packets from `192.168.1.1` (PC0) to `192.168.1.2` (PC1), and all 4 Echo Replies were received (0% packet loss). The round-trip time was very small (of the order of milliseconds) as expected in a simulated local network. The presence of replies and the value of `TTL` confirm that PC1 is active and reachable over the configured point-to-point link.

```
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=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
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 = 1ms, Average = 0ms

C:\>
```

Figure 1.2: Successful `ping` reply from PC0 to PC1

1.0.7 Conclusion

A direct point-to-point connection between two PCs was successfully created and tested in Cisco Packet Tracer. By using a crossover cable and assigning appropriate IP addresses in the same subnet, end-to-end connectivity was established. The experiment demonstrates fundamental concepts of basic Ethernet connections, IP addressing, and connectivity verification using the `ping` utility.

Experiment 2
Mesh Topology Implementation

2.0.1 Aim

To design and implement a Mesh Topology using Cisco Packet Tracer, and verify connectivity using the `ping` command.

2.0.2 Apparatus / Software

- **Hardware:** 4 PCs, 4 Cisco 2960 Switches
- **Software:** Cisco Packet Tracer

2.0.3 Theory

A **Full Mesh Topology** is a network structure in which every device is connected to every other device through a dedicated link. In switched Ethernet networks, a mesh is implemented by interconnecting all switches with crossover cables.

This configuration provides:

- High redundancy
- Multiple alternate paths
- Automatic rerouting in case of link failure

However, redundant paths can create **Layer 2 loops** which result in:

- Broadcast storms
- MAC table instability
- Network congestion

To prevent this, switches run the **Spanning Tree Protocol (STP)**, which:

- Detects redundant links
- Places extra ports in a **Blocking** (Orange) state
- Ensures a loop-free logical topology

2.0.4 Procedure

1. **Place Devices**
   - Add 4 PCs (PC0–PC3).
   - Add 4 switches (Switch0–Switch3).
2. **Connect PCs to Switches**
   - Use **Copper Straight-Through** cables.
   - Connect each PC to its corresponding switch.
3. **Create Mesh Between Switches**
   - Use **Copper Crossover** cables.
   - Connect every switch to every other switch.
4. **Assign IP Addresses**
   - PC0 → `192.168.1.1`
   - PC1 → `192.168.1.2`
   - PC2 → `192.168.1.3`
   - PC3 → `192.168.1.4`
   - Subnet Mask for all: `255.255.255.0`
5. **Verify Connectivity**
   - Open PC0 → Command Prompt.
   - Type: `ping 192.168.1.3`

2.0.5 Diagram / Topology

2.0.6 Results

STP successfully prevented switching loops by logically blocking redundant links (shown in orange). Despite blocked ports, the network maintained full logical connectivity. PC0 successfully pinged PC2 with 0% packet loss, indicating correct IP configuration and functional Layer 2 redundancy handling.

2.0.7 Conclusion

A full mesh topology was successfully implemented. The experiment demonstrated how STP maintains a loop-free topology despite multiple redundant paths, ensuring both reliability and stability in switched networks.

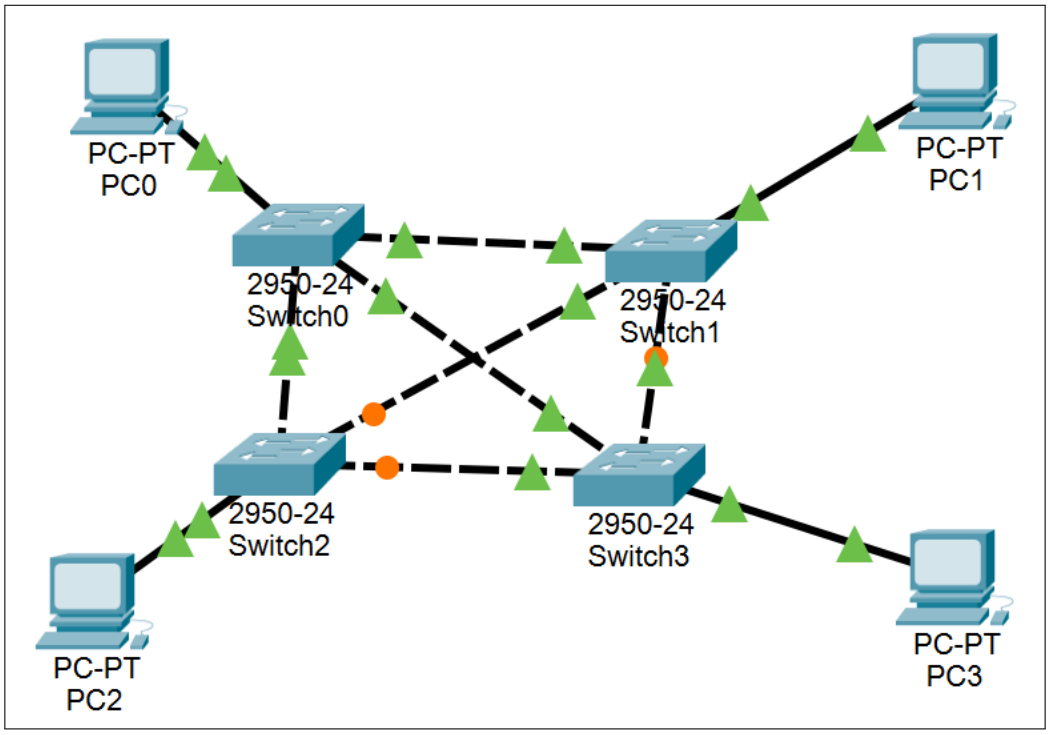

Figure 2.1: Switched Mesh Topology showing STP Blocking (Orange Ports)

```
Cisco Packet Tracer PC Command Line 1.0
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
Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
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 = 1ms, Average = 0ms

C:\>
```

Figure 2.2: Successful `ping` in Mesh Topology

Experiment 3
Bus Topology Implementation

3.0.1 Aim

To design and implement a Bus Topology using Cisco Packet Tracer, configure IP addresses for all PCs, and verify connectivity using the `ping` command.

3.0.2 Apparatus / Software

- **Software:** Cisco Packet Tracer

3.0.3 Theory

A **Bus Topology** consists of a single communication backbone to which all nodes are connected. In modern simulation tools like Cisco Packet Tracer, a bus is emulated by connecting multiple hubs in series (daisy-chaining), forming a shared medium.

Key characteristics:

- Operates at **Layer 1** (Physical Layer)
- All devices share the same bandwidth
- All incoming frames are broadcast to all ports
- Large **collision domain** and **broadcast domain**

When a PC transmits data, hubs repeat the signal to all connected devices, causing collisions if multiple systems transmit simultaneously.

3.0.4 Procedure

1. **Place Devices**
    - Add 4 PCs (PC0–PC3).
    - Add 4 hubs (Hub0–Hub3).
2. **Create Backbone**
    - Use **Copper Crossover** cables.
    - Connect Hub0 → Hub1 → Hub2 → Hub3.
3. **Connect PCs to Hubs**
    - Use **Copper Straight-Through** cables.
    - Connect each PC to a hub.
4. **Assign IP Addresses**

- Assign IPs `192.168.1.1` to `192.168.1.4`
- Subnet Mask: `255.255.255.0`

5. **Verify Connectivity**
   - Open PC0 → Command Prompt
   - Type: `ping 192.168.1.4`

3.0.5 Diagram / Topology

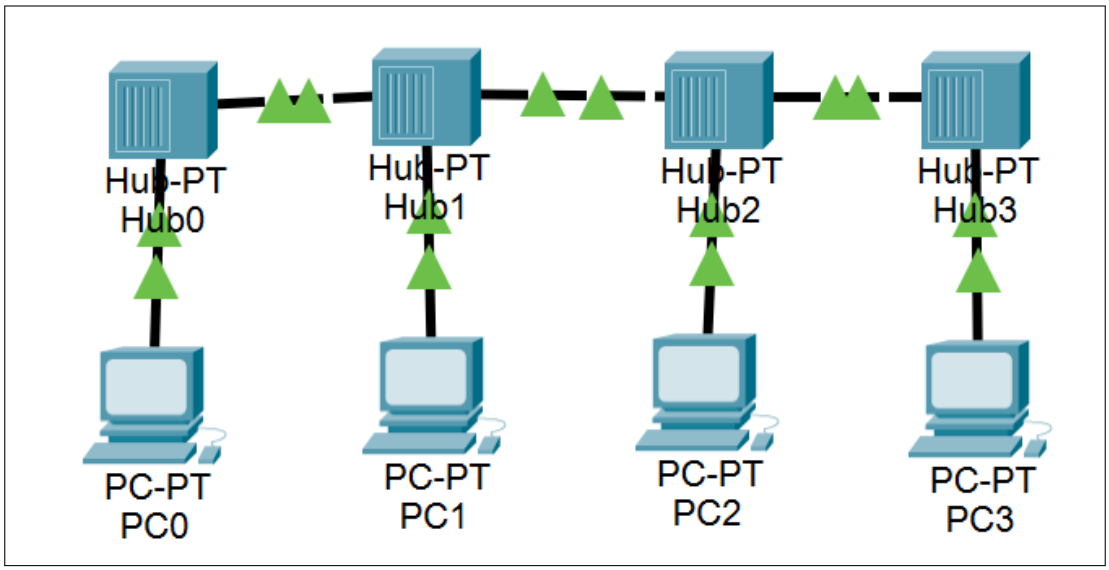

Figure 3.1: Bus Topology implemented using Daisy-Chained Hubs

3.0.6 Results

PC0 successfully pinged PC3 with 0% packet loss. The packet was forwarded by each hub in the chain, demonstrating the flooding nature of hubs. All devices operated within a single collision and broadcast domain.

3.0.7 Conclusion

The bus topology was successfully simulated using cascaded hubs. The experiment verified proper connectivity while illustrating the limitations of bus networks such as broadcasts, collisions, and shared bandwidth.

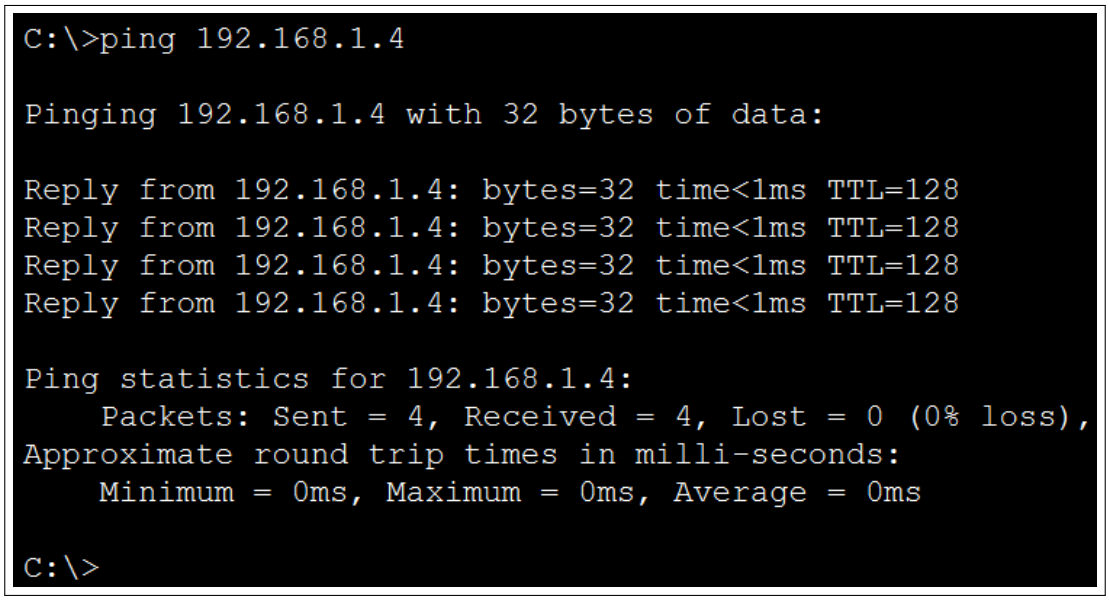

Figure 3.2: Successful Ping Across the Bus Backbone

Experiment 4
Star Topology Implementation

4.0.1 Aim

To design and implement a Star Topology using Cisco Packet Tracer, configure IP addresses for all PCs, and verify connectivity using the `ping` command.

4.0.2 Apparatus / Software

- **Software:** Cisco Packet Tracer

4.0.3 Theory

In a **Star Topology**, all end devices are connected to a single central device, usually a **switch**. The switch operates at the **Data Link Layer (Layer 2)** and maintains a MAC address table (CAM table), which maps MAC addresses to specific ports.

When a frame arrives at the switch:

- The source MAC address is learned and stored in the CAM table.
- The switch looks up the destination MAC address.
- The frame is forwarded only out of the port associated with that MAC (unicast), instead of flooding out all ports (as a hub would).

Advantages of a star topology with a switch:

- Each device has a dedicated link to the switch.
- Collisions are greatly reduced compared to a hub-based topology.
- Easy fault isolation (a failed link typically affects only one device).
- Better bandwidth utilization and security.

4.0.4 Procedure

1. **Place Devices**
    - Add 5 PCs (PC0–PC4).
    - Add 1 Cisco 2960 switch.
2. **Connect PCs to the Switch**
    - Use **Copper Straight-Through** cables.
    - Connect each PC's FastEthernet interface to a FastEthernet port on the switch (e.g., `Fa0/1` to `Fa0/5`).

3. **Assign IP Addresses**
   - PC0 → `192.168.1.1`
   - PC1 → `192.168.1.2`
   - PC2 → `192.168.1.3`
   - PC3 → `192.168.1.4`
   - PC4 → `192.168.1.5`
   - Subnet Mask (all PCs): `255.255.255.0`
4. **Verify Connectivity**
   - On PC0, open **Command Prompt**.
   - Type:

     `ping 192.168.1.5`
   - Observe the ICMP Echo Replies from PC4.

4.0.5 Diagram / Topology

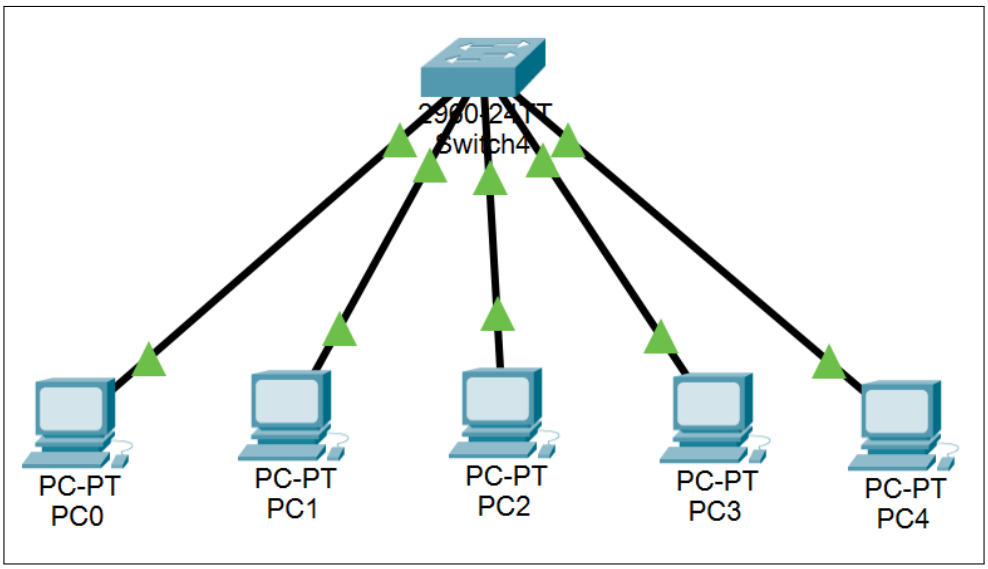

Figure 4.1: Star Topology with Central Switch

4.0.6 Results

The `ping` test from PC0 (`192.168.1.1`) to PC4 (`192.168.1.5`) showed 100% success. After the initial ARP resolution, the switch forwarded frames directly between the corresponding ports without flooding. Each PC enjoyed dedicated bandwidth on its switch port, and no collisions occurred, highlighting the efficiency of a switched star topology.

```
C:\>ping 192.168.1.5

Pinging 192.168.1.5 with 32 bytes of data:

Reply from 192.168.1.5: bytes=32 time<1ms TTL=128
Reply from 192.168.1.5: bytes=32 time<1ms TTL=128
Reply from 192.168.1.5: bytes=32 time<1ms TTL=128
Reply from 192.168.1.5: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

Figure 4.2: `ping` Output from PC0 to PC4

4.0.7 Conclusion

A star topology using a central switch was successfully implemented and tested. The experiment demonstrated that switches provide:

- Better performance than hubs,
- Reduced collision domains,
- More efficient and secure frame forwarding.

This confirms why star topology with switches is widely used in modern LANs.

Experiment 5
Basic Firewall Configuration

5.0.1 Aim

To configure and test a basic firewall on a server in Cisco Packet Tracer using security rules that block or allow ICMP traffic.

5.0.2 Apparatus / Software

- **Software:** Cisco Packet Tracer

5.0.3 Theory

A **firewall** is a network security mechanism that monitors and filters incoming and outgoing traffic based on predefined rules. It can be:

- **Network-based firewall** (deployed on routers, security appliances), or
- **Host-based firewall** (installed on individual servers or PCs).

In this experiment, we configure a **host-based firewall** on a server to:

- **Deny** ICMP (ping) traffic from a specific source IP address.
- **Allow** all other IP traffic.

This demonstrates:

- **Blacklisting**: blocking selected hosts.
- Basic access control based on protocol (ICMP) and source IP.

5.0.4 Procedure

1. **Place Devices**
    - Add 1 Server, 1 Switch, and 2 PCs (PC0 and PC1).
2. **Connect Devices**
    - Use **Copper Straight-Through** cables.
    - Connect Server, PC0, and PC1 to the switch.
3. **Assign IP Addresses**
    - Server → `192.168.1.100`
    - PC0 → `192.168.1.1`
    - PC1 → `192.168.1.2`

- Subnet Mask for all: `255.255.255.0`

4. **Configure Firewall on Server**
   - On the Server, go to: **Desktop → Firewall (IPv4)**.
   - Add a **Deny** rule:
     - Action: Deny
     - Protocol: ICMP
     - Remote IP: `192.168.1.1`
     - Wildcard: `0.0.0.0`
   - Add a general **Allow** rule:
     - Action: Allow
     - Protocol: IP
     - Remote IP: `0.0.0.0`
     - Wildcard: `255.255.255.255`
   - Turn the firewall **ON**.
5. **Verify Firewall Behavior**
   - From PC0, `ping 192.168.1.100` (should **fail**).
   - From PC1, `ping 192.168.1.100` (should **succeed**).

5.0.5 Diagram / Topology

5.0.6 Results

- **From PC0 (`192.168.1.1`):** All ICMP Echo Requests to `192.168.1.100` timed out. This indicates that the server's firewall successfully dropped packets from the blacklisted source IP.
- **From PC1 (`192.168.1.2`):** ICMP Echo Replies were received successfully from `192.168.1.100`, confirming that traffic from other IP addresses was allowed by the firewall.

5.0.7 Conclusion

The experiment successfully demonstrated host-based firewall filtering using IP and protocol-specific rules. By selectively blocking ICMP traffic from one host and allowing others, we verified how firewalls enforce security policies and protect critical resources from unauthorized or undesired access.

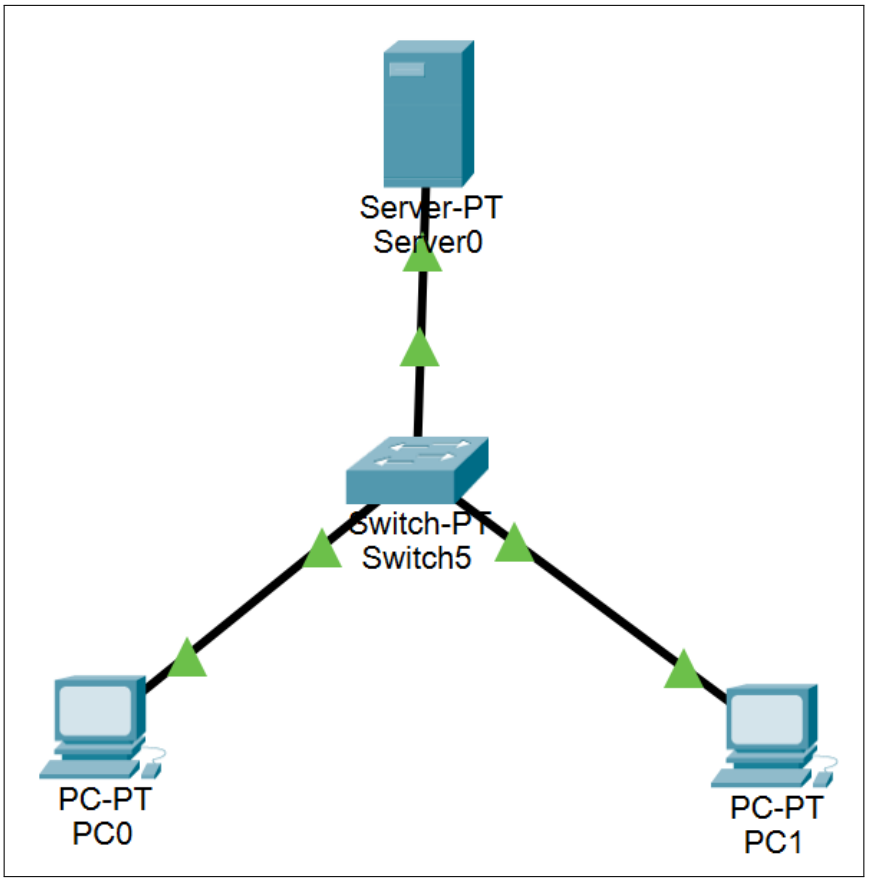

Figure 5.1: Firewall Topology: Two PCs and a Server connected via a Switch

```
C:\>ping 192.168.1.100

Pinging 192.168.1.100 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.100:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>
```

Figure 5.2: Ping Result from PC0 to Server (Blocked)

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.100

Pinging 192.168.1.100 with 32 bytes of data:

Reply from 192.168.1.100: bytes=32 time<1ms TTL=128
Reply from 192.168.1.100: bytes=32 time<1ms TTL=128
Reply from 192.168.1.100: bytes=32 time<1ms TTL=128
Reply from 192.168.1.100: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

Figure 5.3: Ping Result from PC1 to Server (Allowed)

Experiment 6
Connecting Two LANs via Router

6.0.1 Aim

To design and implement two Local Area Networks (LANs) and connect them using a router in Cisco Packet Tracer. Configure IP addresses, default gateways, and verify connectivity using the `ping` command.

6.0.2 Apparatus / Software

- **Software:** Cisco Packet Tracer

6.0.3 Theory

Switches forward frames within a **single LAN or subnet**. To communicate between **different networks** (for example, `192.168.1.0/24` and `192.168.2.0/24`), a **router** is required.

A router:

- Operates at the **Network Layer (Layer 3)**.
- Makes forwarding decisions based on **IP addresses**.
- Has interfaces that typically belong to different IP networks.

Each host uses a **Default Gateway**, which is the IP address of the router interface on its local network. Any traffic destined for an IP outside the local subnet is sent to this default gateway.

6.0.4 Procedure

1. **Place Devices**
    - Add 1 Cisco 1941 router.
    - Add 2 switches (Switch0 and Switch1).
    - Add 2 PCs: PC0 (on LAN 1) and PC1 (on LAN 2).
2. **Connect Devices**
    - Use **Copper Straight-Through** cables:
    - PC0 → Switch0, Switch0 → Router (G0/0).
    - PC1 → Switch1, Switch1 → Router (G0/1).
3. **Configure Router Interfaces**
    - On `G0/0`:

- IP Address: `192.168.1.1`
- Subnet Mask: `255.255.255.0`
- Set interface to **Up** (no shutdown).

- On `G0/1`:
  - IP Address: `192.168.2.1`
  - Subnet Mask: `255.255.255.0`
  - Set interface to **Up**.

4. **Configure PCs**

- **PC0 (LAN 1):**
  - IP Address: `192.168.1.10`
  - Subnet Mask: `255.255.255.0`
  - Default Gateway: `192.168.1.1`
- **PC1 (LAN 2):**
  - IP Address: `192.168.2.10`
  - Subnet Mask: `255.255.255.0`
  - Default Gateway: `192.168.2.1`

5. **Verify Inter-LAN Connectivity**

- From PC0, open Command Prompt.
- Type:

```
ping 192.168.2.10
```

- Observe the ping results.

6.0.5 Diagram / Topology

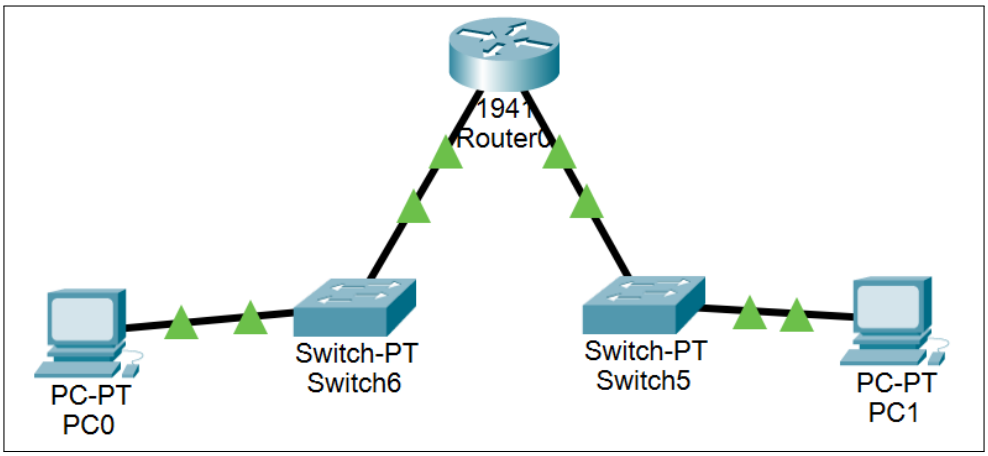

Figure 6.1: Two LANs Connected via a Router

```
C:\>ping 192.168.2.10

Pinging 192.168.2.10 with 32 bytes of data:

Reply from 192.168.2.10: bytes=32 time<1ms TTL=127
Reply from 192.168.2.10: bytes=32 time<1ms TTL=127
Reply from 192.168.2.10: bytes=32 time<1ms TTL=127
Reply from 192.168.2.10: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.2.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

Figure 6.2: `ping` Result Between Different LANs

6.0.6 Results

The first ICMP Echo Request from PC0 to PC1 timed out due to ARP resolution on the router and PCs. Subsequent `ping` attempts were successful with 0% packet loss. This confirms that:

- The router correctly routed traffic between `192.168.1.0/24` and `192.168.2.0/24`.
- Default gateways on both PCs were configured properly.

6.0.7 Conclusion

The experiment successfully demonstrated the role of a router in interconnecting two different LANs. It highlighted the importance of:

- Proper IP addressing and subnetting.
- Default gateway configuration on end devices.
- Router interfaces acting as gateways between networks.

End-to-end connectivity across different subnets was achieved and verified.

Experiment 7
Static Routing with Two Routers

7.0.1 Aim

To design and implement two separate Local Area Networks (LANs) connected together using two routers in Cisco Packet Tracer. Configure IP addresses, default gateways, and static routing to enable communication between the two networks.

7.0.2 Apparatus / Software

- **Software:** Cisco Packet Tracer

7.0.3 Theory

When multiple routers are present in a network, each router initially knows only about the networks that are **directly connected** to its own interfaces. To allow communication with remote networks, routing information must be added.

Static routing is a manual method in which the network administrator adds entries to the routing table. Each static route specifies:

To reach network N, forward packets to next-hop IP H.

In this experiment:

- Each router has a LAN on one side and a WAN link between them.
- Static routes are configured so that:
  - Router0 knows how to reach the LAN behind Router1.
  - Router1 knows how to reach the LAN behind Router0.

This allows end devices in different LANs to communicate across the WAN link.

7.0.4 Procedure

1. **Network Setup**
   - Place 2 Cisco 1941 Routers (Router0, Router1).
   - Place 2 Switches (one per LAN).
   - Place 4 PCs (two on each side).
   - Add **HWIC-2T** serial modules to both routers to enable serial interfaces.
2. **Cabling**
   - Connect Router0 `S0/1/0` to Router1 `S0/1/0` using a **Serial DCE Cable**.

- Connect PCs to their respective switches using **Copper Straight-Through** cables.
- Connect each switch to the appropriate router GigabitEthernet interface.

3. **IP Addressing**
   - **Router0:**
     - LAN interface (e.g., `G0/0`): `192.168.1.1` / `255.255.255.0`
     - WAN interface (`S0/1/0`): `10.0.0.1` / `255.255.255.252`
   - **Router1:**
     - LAN interface (e.g., `G0/0`): `192.168.2.1` / `255.255.255.0`
     - WAN interface (`S0/1/0`): `10.0.0.2` / `255.255.255.252`
   - Assign suitable IP addresses to PCs in each LAN (e.g., `192.168.1.x` on the left, `192.168.2.x` on the right) with matching subnet masks and default gateways.
4. **Static Route Configuration**
   - On **Router0**, add a route to the remote LAN `192.168.2.0/24` via next hop `10.0.0.2`.
   - On **Router1**, add a route to the remote LAN `192.168.1.0/24` via next hop `10.0.0.1`.
5. **Verification**
   - From a PC in the left LAN, open Command Prompt.
   - Run:

     ```
     ping 192.168.2.x
     ```

     where `192.168.2.x` is a PC in the right LAN.

7.0.5 Diagram / Topology

7.0.6 Results

The `ping` from the source PC in the left LAN to the destination PC in the right LAN was successful. The ICMP packets traversed:

- From the source PC to Router0 (default gateway),
- Across the serial WAN link to Router1,
- Finally to the destination PC in the remote LAN.

Initial packets may have timed out due to ARP and routing resolution, but subsequent replies were received with 0% packet loss.

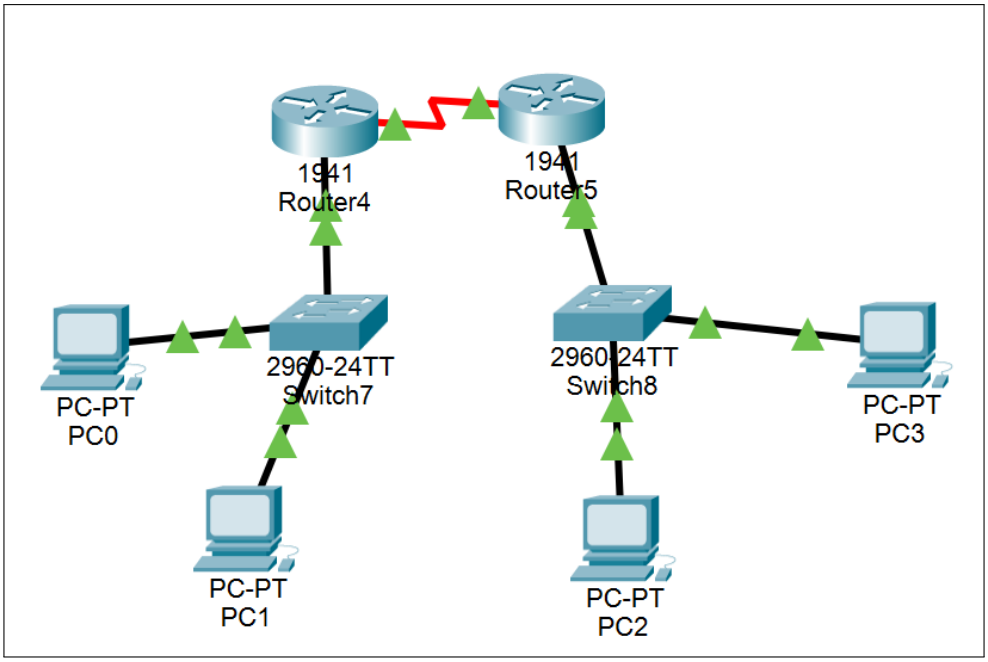

Figure 7.1: Static Routing Topology across WAN Link

```
C:\>ping 192.168.2.10

Pinging 192.168.2.10 with 32 bytes of data:

Reply from 192.168.2.10: bytes=32 time=28ms TTL=126
Reply from 192.168.2.10: bytes=32 time=1ms TTL=126
Reply from 192.168.2.10: bytes=32 time=1ms TTL=126
Reply from 192.168.2.10: bytes=32 time=2ms TTL=126

Ping statistics for 192.168.2.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 28ms, Average = 8ms

C:\>
```

Figure 7.2: Successful `ping` across the WAN Link

7.0.7 Conclusion

Static routing between two routers was successfully configured. The experiment shows that:

- Routers only know directly connected networks by default.
- Static routes must be manually defined to reach remote networks.

This forms the basis for more advanced routing using dynamic routing protocols.

Experiment 8
Multi-Hop Routing with Three Routers

8.0.1 Aim

To design and implement three Local Area Networks (LANs) and connect them in series (line topology) using three routers in Cisco Packet Tracer, and to verify end-to-end connectivity using static routing.

8.0.2 Apparatus / Software

- **Software:** Cisco Packet Tracer

8.0.3 Theory

In a **multi-hop** routed network, traffic must pass through one or more intermediate routers to reach its final destination. Consider three routers:

$$R1 \leftrightarrow R2 \leftrightarrow R3$$

- R1 connects to LAN 1.
- R3 connects to LAN 3.
- R2 acts as a **transit (intermediate) router**.

For successful end-to-end communication:

- R1 must know how to reach R3's LAN.
- R3 must know how to reach R1's LAN.
- R2 must have routes to both outer LANs so that it can forward traffic in both directions.

Using **static routing**, these paths are manually configured on each router.

8.0.4 Procedure

1. **Topology Setup**
   - Place 3 routers: R1, R2, R3.
   - Connect R1 ↔ R2 and R2 ↔ R3 using serial links (Serial DCE/DTE cables).
   - Attach a LAN with PCs to R1 and another LAN with PCs to R3.
2. **IP Addressing**
   - **WAN Link 1 (R1–R2):** Network `10.0.0.0/30`.

- **WAN Link 2 (R2–R3):** Network `20.0.0.0/30`.
- **LAN 1 (R1 side):** e.g., `192.168.1.0/24`, gateway `192.168.1.1`.
- **LAN 3 (R3 side):** e.g., `192.168.3.0/24`, gateway `192.168.3.1`.
- Assign appropriate IP addresses on all router interfaces and PCs.

3. **Static Route Configuration**

   - **On R1:**
     - Add a static route to R3's LAN (`192.168.3.0/24`) via the next-hop IP on R2 (`10.0.0.2`).
   - **On R2:**
     - Add a static route to R1's LAN (`192.168.1.0/24`) via `10.0.0.1`.
     - Add a static route to R3's LAN (`192.168.3.0/24`) via `20.0.0.2`.
   - **On R3:**
     - Add a static route to R1's LAN (`192.168.1.0/24`) via `20.0.0.1`.

4. **Verification**

   - From a PC in R1's LAN, open Command Prompt.
   - Ping a PC in R3's LAN:

     ```
     ping 192.168.3.x
     ```

   - Observe whether the ping is successful and note any initial time-outs.

8.0.5 Diagram / Topology

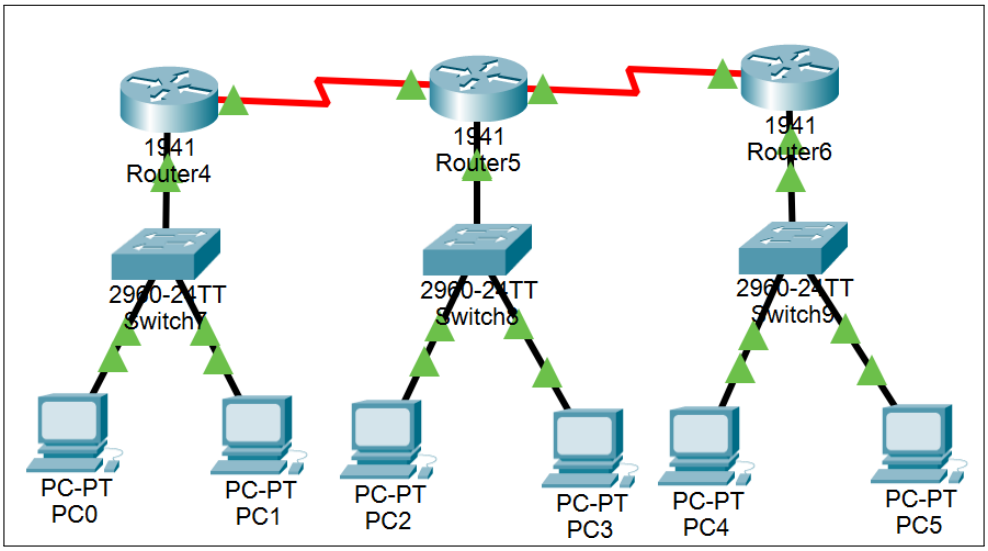

Figure 8.1: Multi-Hop Routing Topology with Three Routers

```
C:\>ping 192.168.3.11

Pinging 192.168.3.11 with 32 bytes of data:

Reply from 192.168.3.11: bytes=32 time=32ms TTL=125
Reply from 192.168.3.11: bytes=32 time=10ms TTL=125
Reply from 192.168.3.11: bytes=32 time=16ms TTL=125
Reply from 192.168.3.11: bytes=32 time=9ms TTL=125

Ping statistics for 192.168.3.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 32ms, Average = 16ms

C:\>
```

Figure 8.2: Successful `ping` across Two WAN Hops

8.0.6 Results

End-to-end connectivity between the PC in R1's LAN and the PC in R3's LAN was achieved. ICMP packets successfully traversed:

$$\text{PC (LAN 1)} \rightarrow R1 \rightarrow R2 \rightarrow R3 \rightarrow \text{PC (LAN 3)}$$

The observed round-trip time reflected the additional processing across multiple hops. The TTL value decreased accordingly, indicating traversal through intermediate routers.

8.0.7 Conclusion

Multi-hop static routing using three routers was successfully implemented. The experiment confirmed that:

- Transit routers must have correct routes for both source and destination networks.
- Each router must have a complete routing view (via static or dynamic routing) for successful two-way communication.

Experiment 9
Secure VPN Configuration

9.0.1 Aim

To design and configure a secure VPN topology using Cisco Packet Tracer and verify encrypted connectivity between two remote LANs using the `ping` command.

9.0.2 Apparatus / Software

- **Software:** Cisco Packet Tracer

9.0.3 Theory

A **Virtual Private Network (VPN)** creates a secure, encrypted tunnel over an untrusted or public network (e.g., the Internet). In enterprise networks, **IPsec** (Internet Protocol Security) is a common framework used to provide:

- Confidentiality (encryption),
- Integrity (protection against tampering),
- Authentication (verifying peers).

IPsec operation is typically divided into two phases:

1. **ISAKMP / IKE Phase 1:**
    - Establishes a secure management tunnel between VPN peers.
    - Negotiates parameters such as encryption algorithm (e.g., AES), hashing (e.g., SHA), and authentication method (e.g., pre-shared key).
2. **IPsec Phase 2:**
    - Protects user data traffic.
    - Uses the agreed transform set (e.g., ESP-AES-SHA) to encrypt and authenticate packets.

A **site-to-site** VPN connects two remote LANs via their edge routers, so that traffic between these LANs is encrypted across the WAN.

9.0.4 Procedure

1. **Base Topology**
    - Reuse or construct the 2-router topology from Experiment 7, with one LAN behind each router.

- Ensure that basic static routing is already working (plain `ping` between LANs succeeds before VPN).

2. **Enable Security Features**
   - On each router, enable the security license (in Packet Tracer):

     ```
     license boot module c1900 technology-package securityk9
     ```

   - Reload the router after enabling the license.
3. **Configure IPsec VPN**
   - Define an **ISAKMP policy** (Phase 1):
     - Encryption: AES
     - Hash: SHA
     - Authentication: Pre-shared key
     - DH Group and lifetime as required
   - Configure the **pre-shared key** on both routers using the same key string.
   - Define an **IPsec transform set** (Phase 2) using ESP with AES and SHA.
   - Create an **access-list** on each router to match interesting traffic (e.g., traffic from LAN 1 to LAN 2).
   - Create a **crypto map**, attach the transform set and ACL, and apply it to the WAN-facing interface on both routers.
4. **Verification**
   - From a PC in LAN 1, ping a PC in LAN 2.
   - During the first packets, observe that the tunnel is being negotiated and pings may initially fail.
   - Once established, use router CLI commands such as:

     ```
     show crypto isakmp sa
     ```

   - Check that the state is `QM_IDLE`, indicating an active Phase 2 tunnel.

9.0.5 Diagram / Topology

9.0.6 Results

Initially, `ping` attempts timed out while Phase 1 and Phase 2 negotiations were in progress. After the VPN tunnel was established:

- ICMP Echo Replies were received successfully between the two LANs.
- The command `show crypto isakmp sa` displayed a state of `QM_IDLE`, confirming that the IPsec tunnel was active.

All user traffic between the two LANs traversed the encrypted tunnel, protecting data over the simulated WAN.

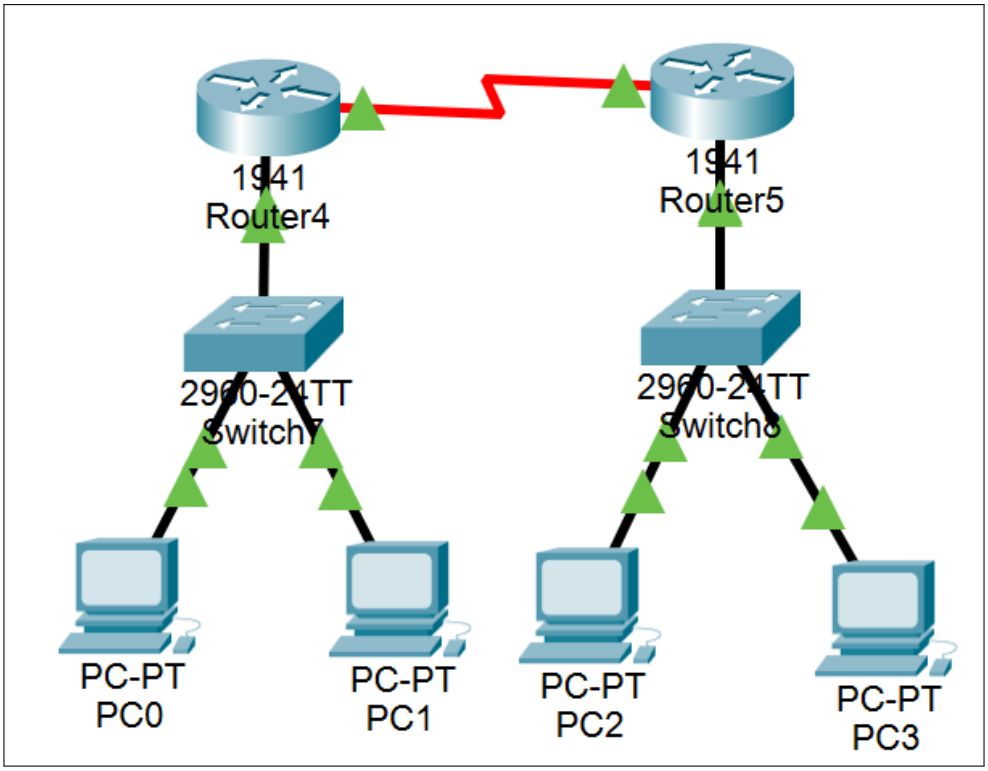

Figure 9.1: Site-to-Site Secure VPN Topology

```
C:\>ping 192.168.2.10

Pinging 192.168.2.10 with 32 bytes of data:

Reply from 192.168.2.10: bytes=32 time=31ms TTL=126
Reply from 192.168.2.10: bytes=32 time=1ms TTL=126
Reply from 192.168.2.10: bytes=32 time=1ms TTL=126
Reply from 192.168.2.10: bytes=32 time=8ms TTL=126

Ping statistics for 192.168.2.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 31ms, Average = 10ms

C:\>
```

Figure 9.2: Successful `ping` over Encrypted VPN Tunnel

9.0.7 Conclusion

A secure site-to-site IPsec VPN was successfully configured between two remote LANs. The experiment demonstrated:

- How encryption and tunneling protect data over untrusted networks.
- The role of ISAKMP Phase 1 and IPsec Phase 2 in establishing and maintaining the VPN.

This is a fundamental concept in modern enterprise WAN and remote-site connectivity.

Experiment 10
Traffic Analysis using Wireshark

10.0.1 Aim

To capture and analyze different types of network traffic (ICMP, TCP, HTTP, UDP) using Wireshark and to study the color-coding rules for packet identification.

10.0.2 Apparatus / Software

- **Software:** Wireshark Network Protocol Analyzer

10.0.3 Theory

Wireshark is a powerful network protocol analyzer used to capture and inspect packets traveling over a network. It:

- Captures frames from a network interface (Ethernet/Wi-Fi).
- Decodes protocol headers (Ethernet, IP, TCP, UDP, ICMP, HTTP, etc.).
- Displays fields in a structured, human-readable format.

Wireshark also uses **color coding** to quickly distinguish different protocol types. For example (depending on default profile):

- ICMP packets may appear in light red/pink.
- TCP packets in light green.
- UDP packets in light blue.

By applying display filters (such as `icmp`, `tcp`, `udp`, `http`), specific traffic can be isolated and analyzed in detail, including flags, sequence numbers, and payloads.

10.0.4 Procedure

1. **Start Capture**
    - Open Wireshark on the host machine.
    - Select the active network interface (e.g., `Wi-Fi` or `Ethernet`).
    - Click on **Start Capturing Packets**.
2. **Generate ICMP Traffic**
    - Open Command Prompt or Terminal.
    - Type:

```
ping google.com
```

- Observe ICMP Echo Request and Echo Reply packets appearing in Wireshark.

3. **Generate TCP/HTTP Traffic**
   - Open a web browser.
   - Visit `http://example.com` or another HTTP site.
   - This generates TCP connections and HTTP request/response packets.
4. **Filter and Analyze**
   - In Wireshark's display filter bar, type `icmp` and press Enter.
   - Inspect ICMP packet details such as Type, Code, Identifier, and Sequence Number.
   - Change the filter to `tcp` and analyze TCP segments (3-way handshake, sequence/ack numbers).
   - Use **Follow TCP Stream** on an HTTP packet to reconstruct the HTTP conversation (GET request and server response).

10.0.5 Diagram / Topology

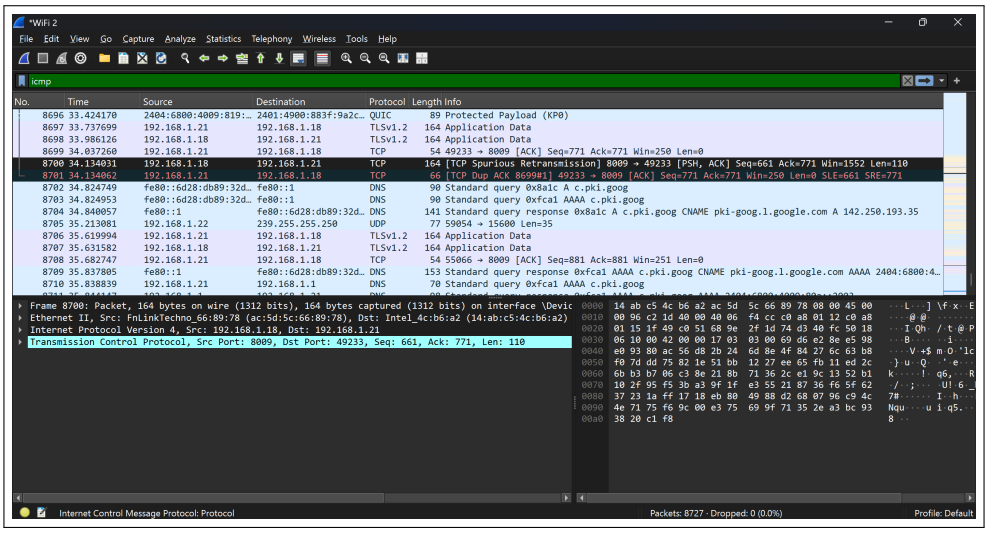

Figure 10.1: Wireshark Capturing ICMP Packets

10.0.6 Results

The following observations were made:

- **ICMP Traffic:** The `ping` command generated ICMP Echo Request (Type 8) and Echo Reply (Type 0) packets. These were highlighted with a specific color in Wireshark, making them easy to identify.
- **TCP/HTTP Traffic:** When accessing a web page, TCP packets carrying HTTP data were captured. Using "Follow TCP Stream" showed the complete HTTP GET request and the server's `200 OK` response in human-readable text.

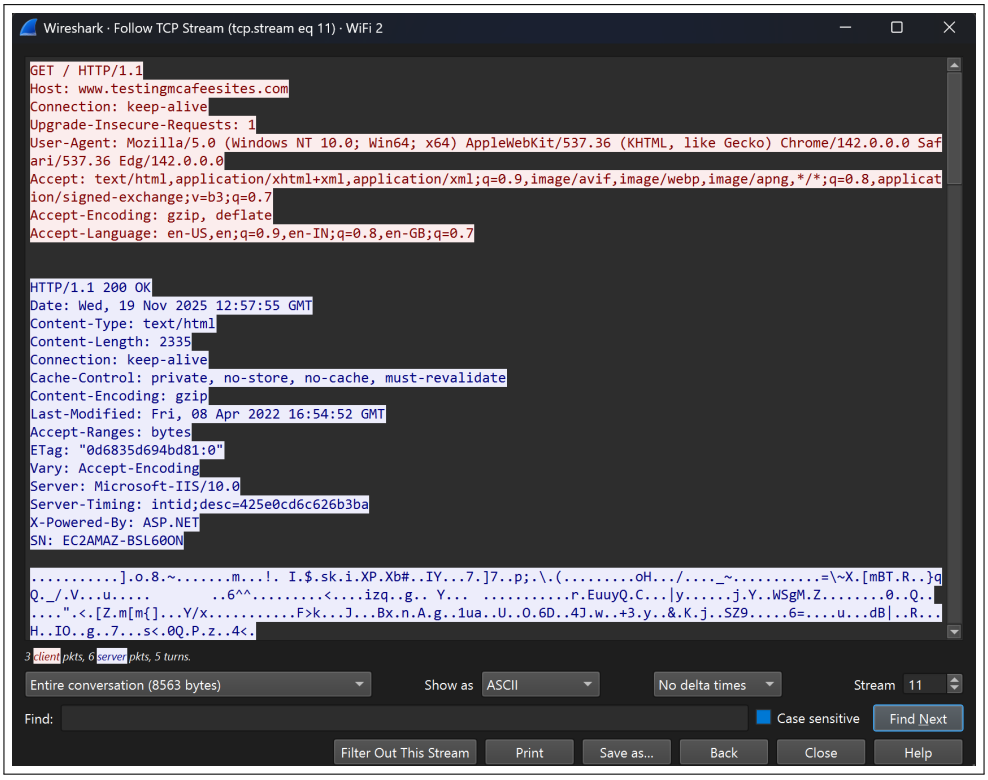

Figure 10.2: TCP Stream Analysis of HTTP Request and Response

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

Pinging www.google.com [2404:6800:4007:838::2004] with 32 bytes of data:
Reply from 2404:6800:4007:838::2004: time=43ms
Reply from 2404:6800:4007:838::2004: time=53ms
Reply from 2404:6800:4007:838::2004: time=43ms
Reply from 2404:6800:4007:838::2004: time=45ms

Ping statistics for 2404:6800:4007:838::2004:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 43ms, Maximum = 53ms, Average = 46ms

C:\Users\agarw>
```

Figure 10.3: `ping` Command Generating ICMP Traffic

10.0.7 Conclusion

Wireshark successfully captured and decoded various protocol packets (ICMP, TCP, HTTP, UDP). This experiment:

- Demonstrated how real-time network traffic can be analyzed at the packet level.
- Showed how display filters and color coding help quickly isolate and interpret specific protocols.
- Provided insight into how higher-level applications (such as web browsing and ping) are mapped onto underlying transport and network protocols.