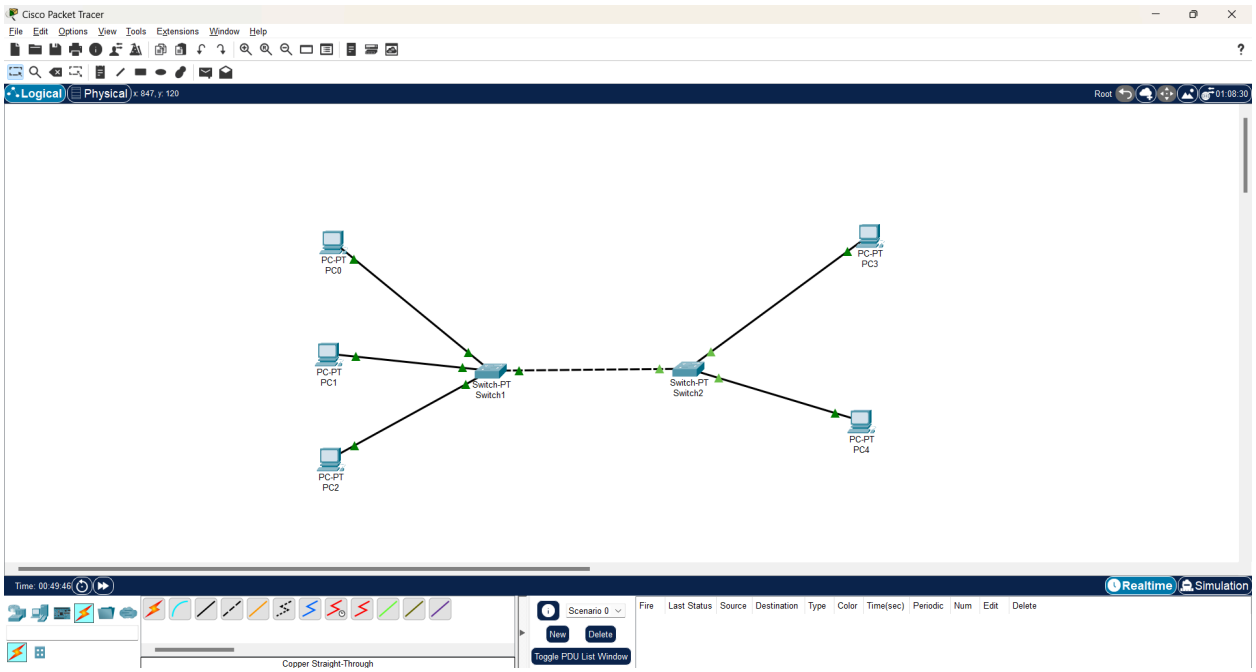
# Networking Assessment 2

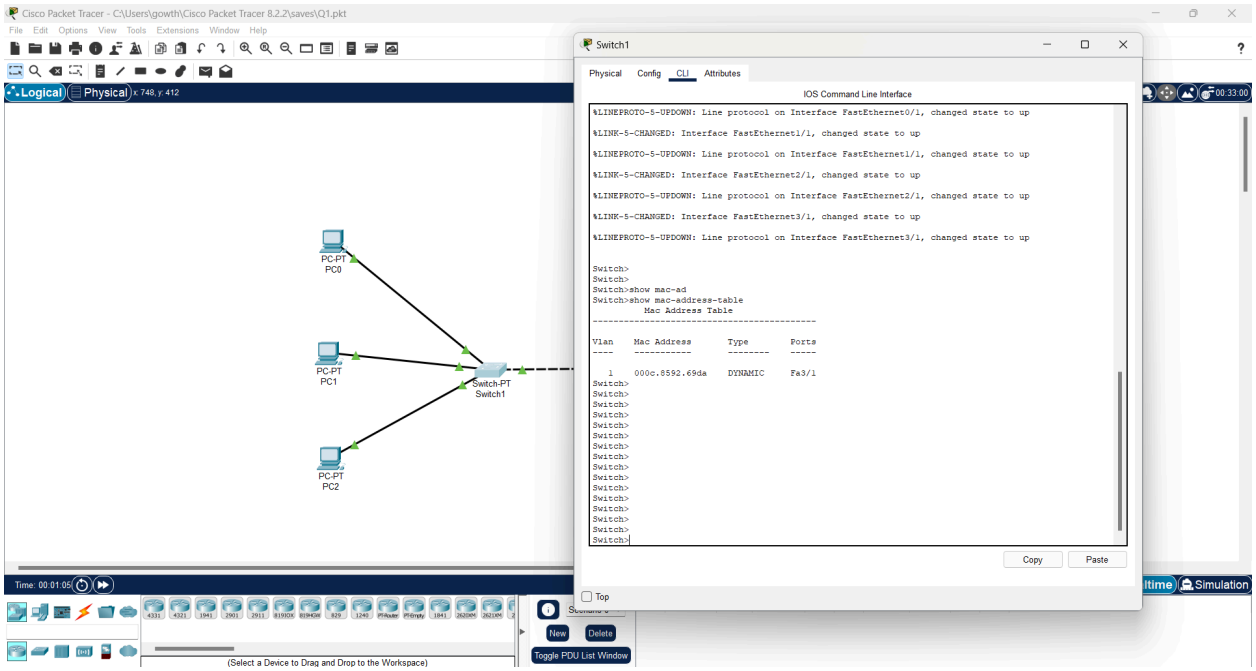
| Name | T K Gowtham |
|---|---|
| Email ID | gowthamkamalasekar@gmail.com |
| College | VIT Chennai |

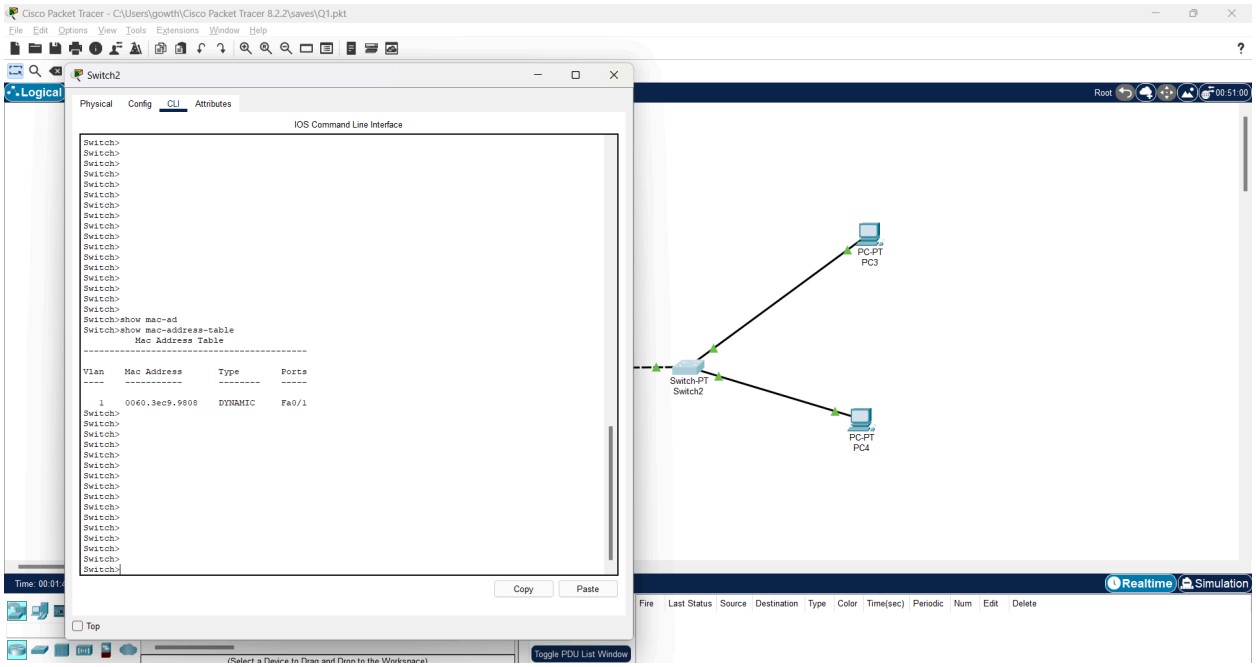
1. Simulate a small network with switches and multiple devices. Use ping to generate traffic and observe the MAC address table of the switch. Capture packets using Wireshark to analyze Ethernet frames and MAC addressing.

Creating a network :

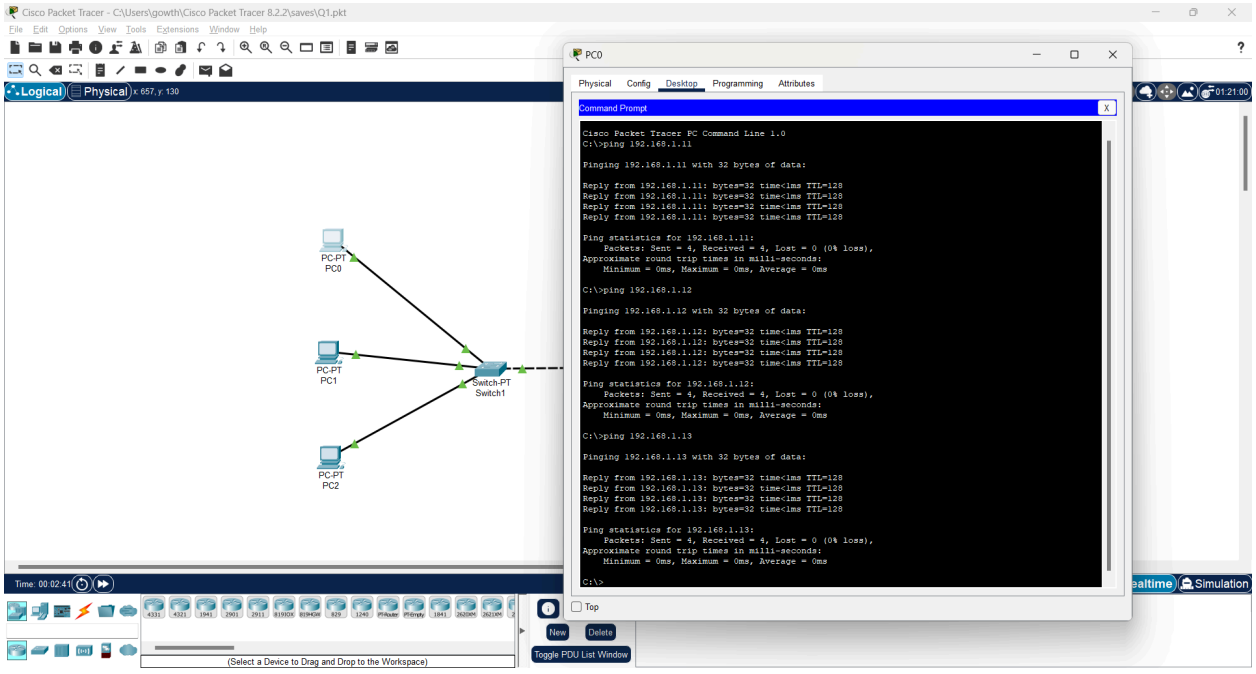


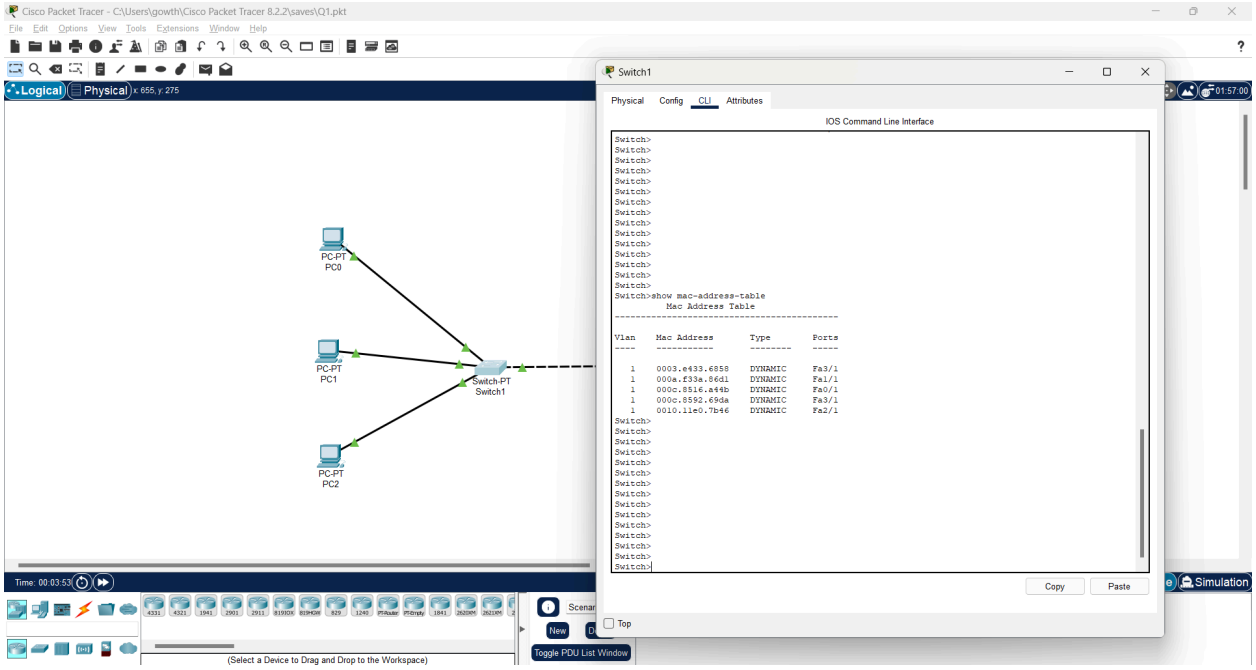
Current Mac Address Before Pinging on both routers :

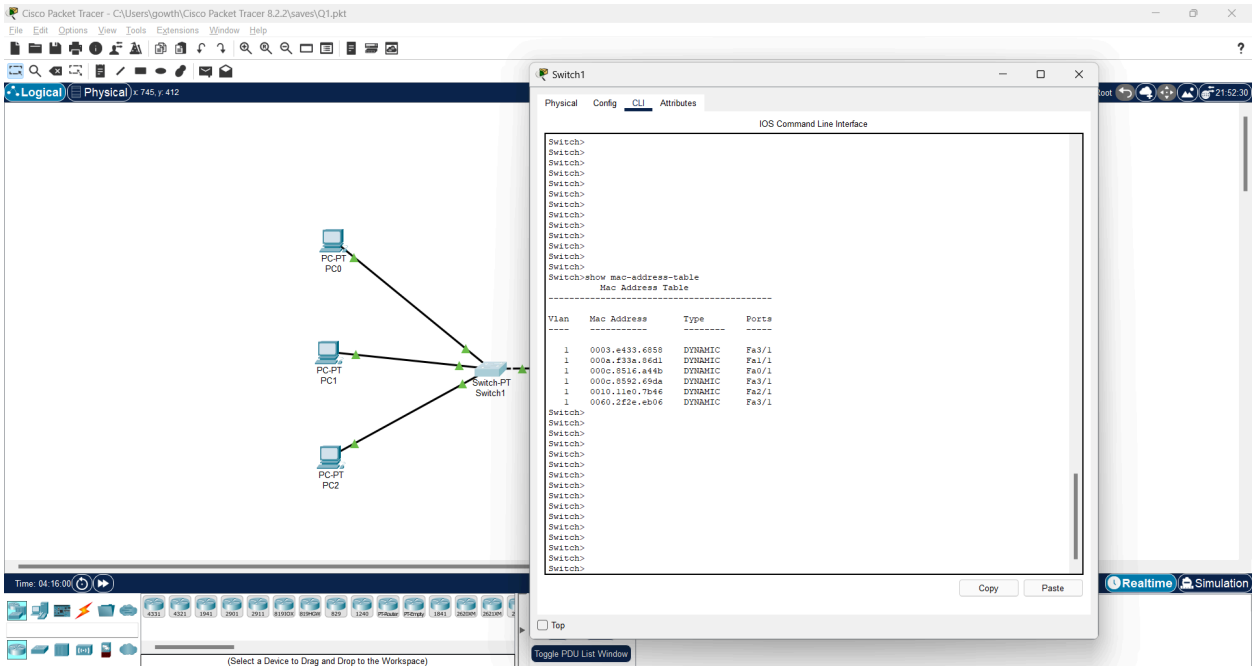


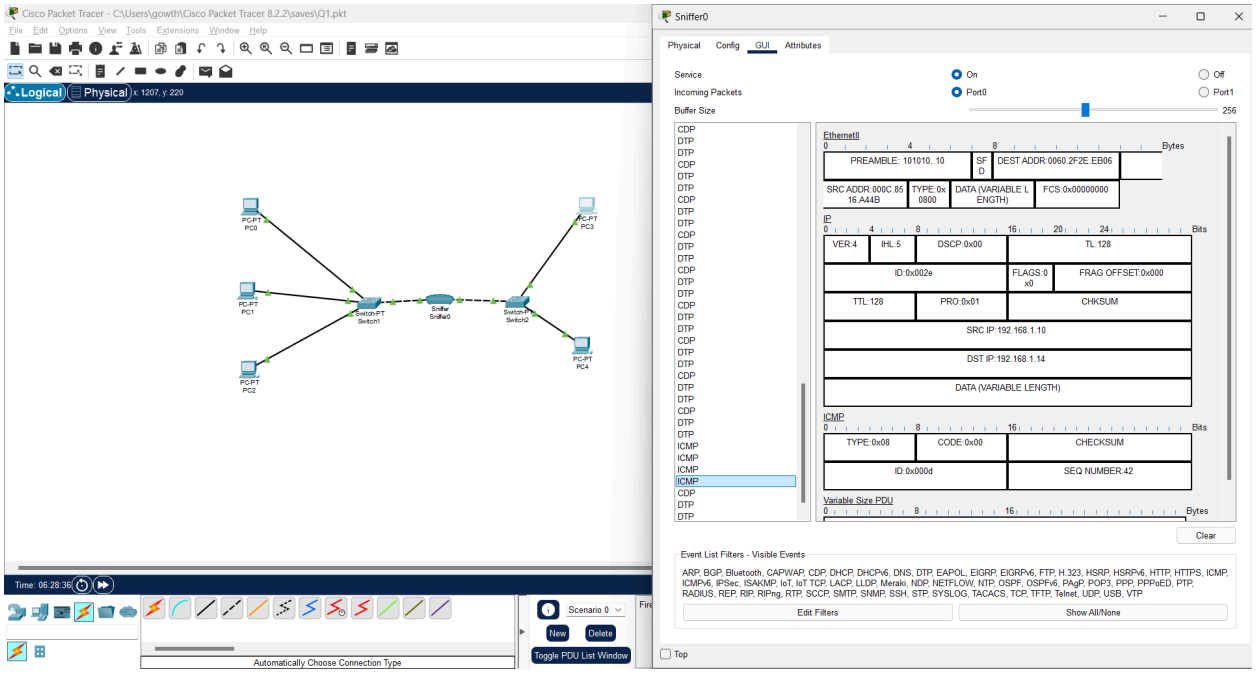
Ping from PC0 to All PC and the MAC Address gets updated all across the Switches.



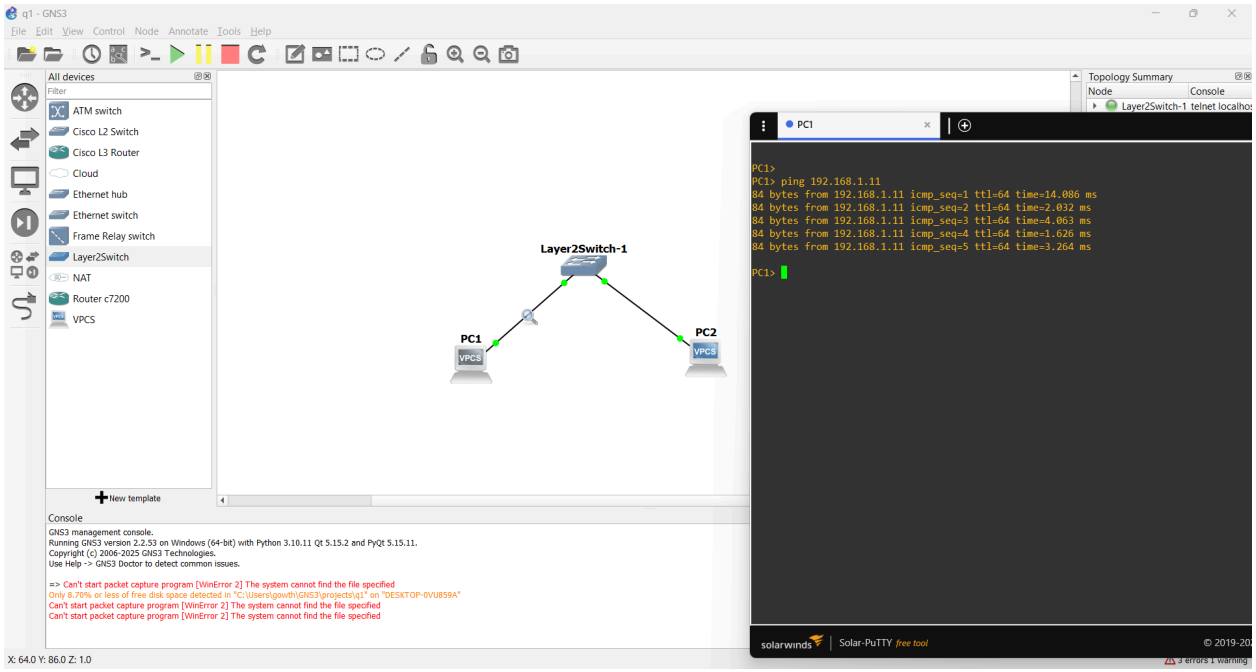



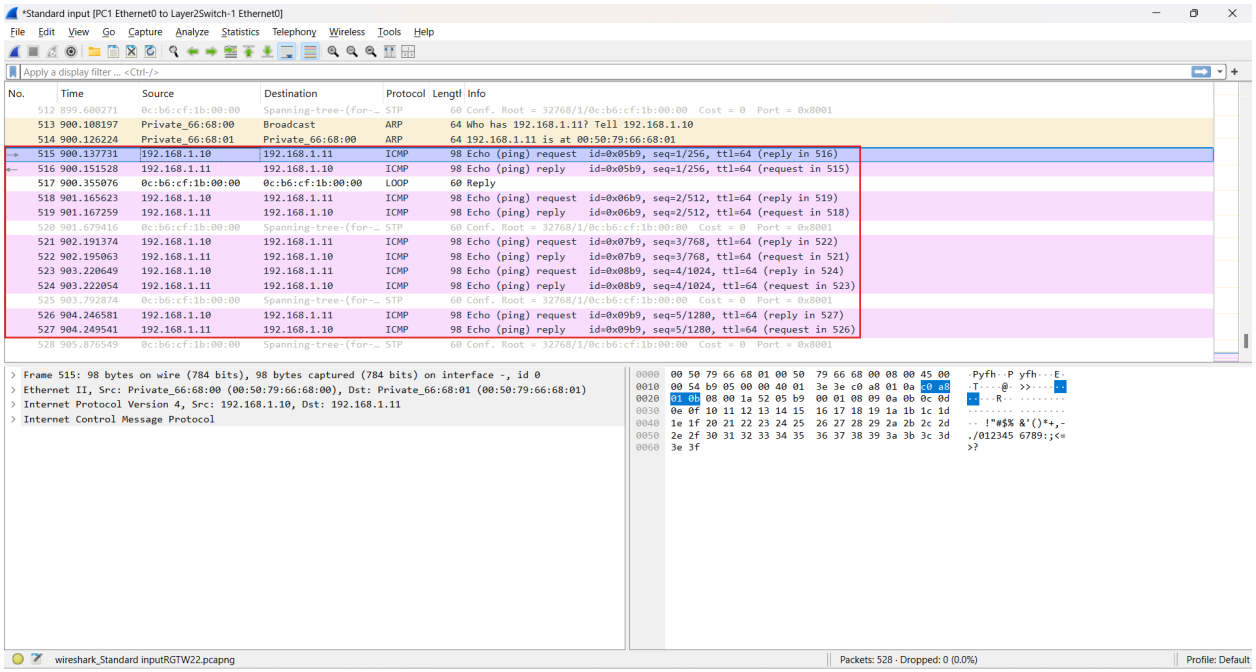



On adding a Sniffer to the network, we can examine the ICMP packet sniffed in the network :

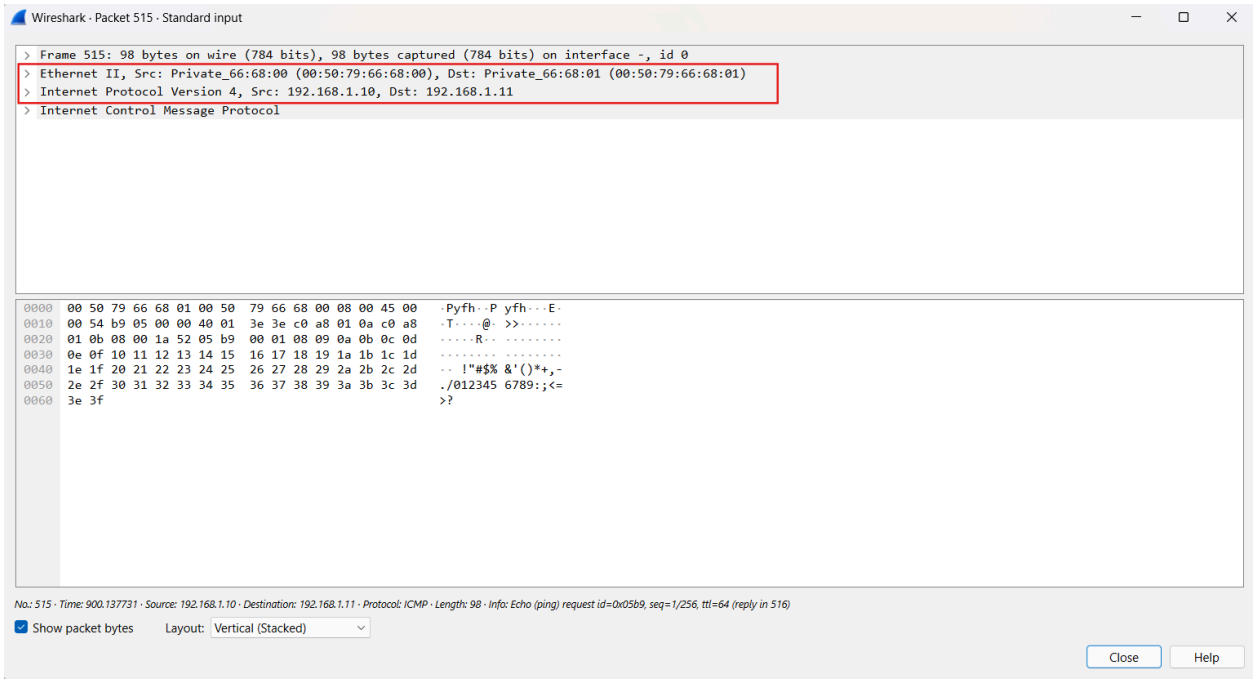


Using GNS3, with wireshark to capture packets :



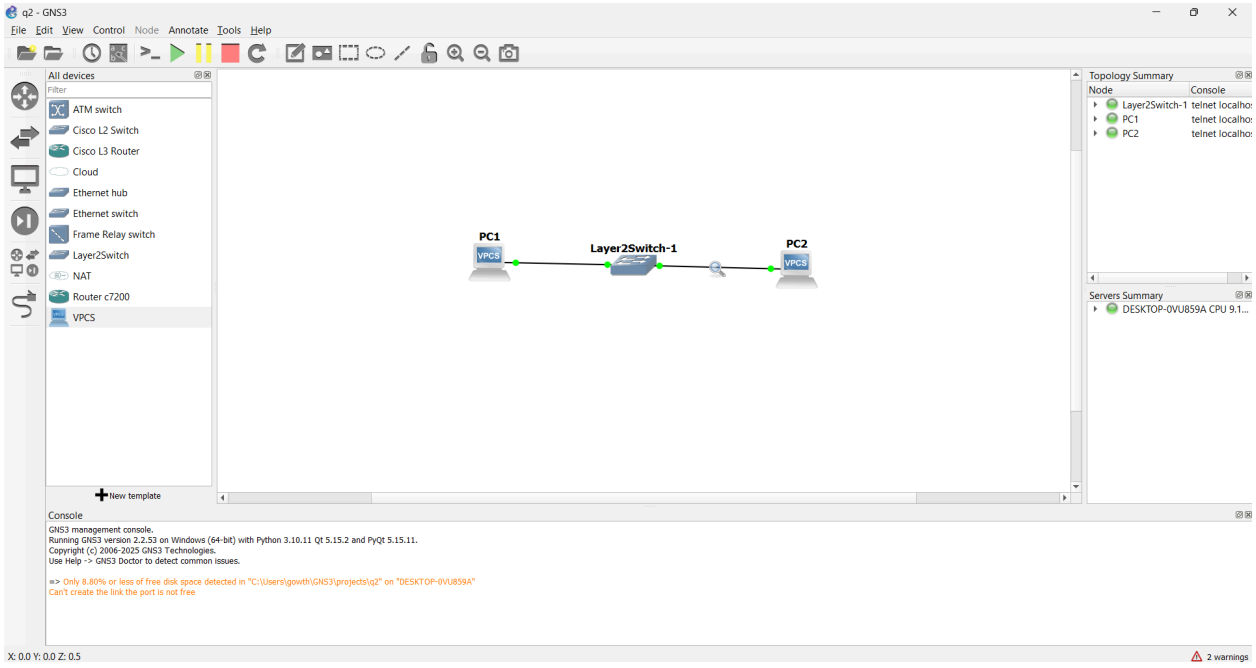



We can see the Ethernet Frame and MAC Addressing as we analyse the packet :

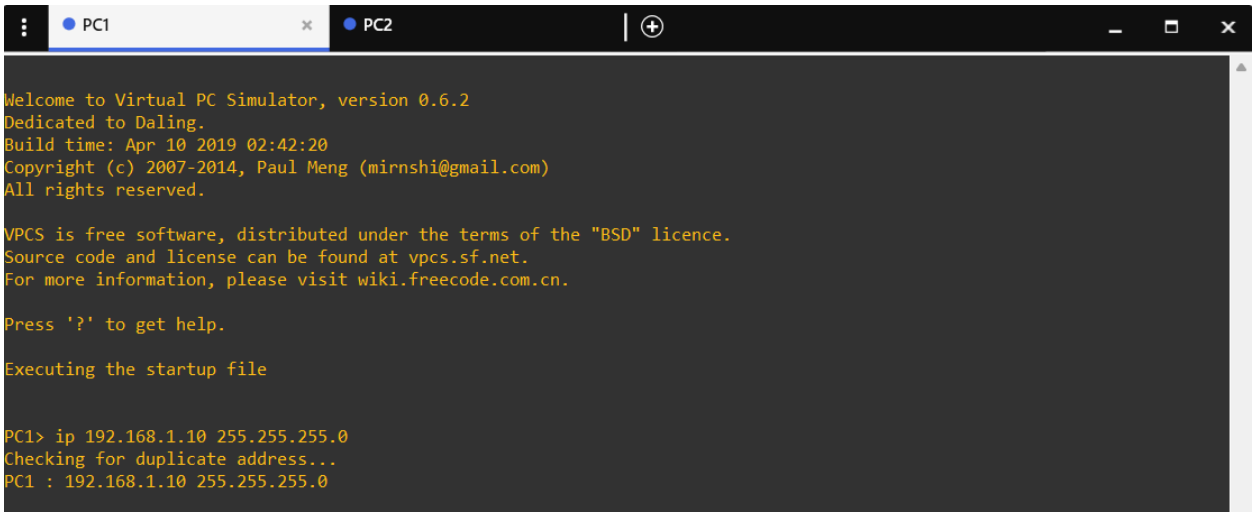


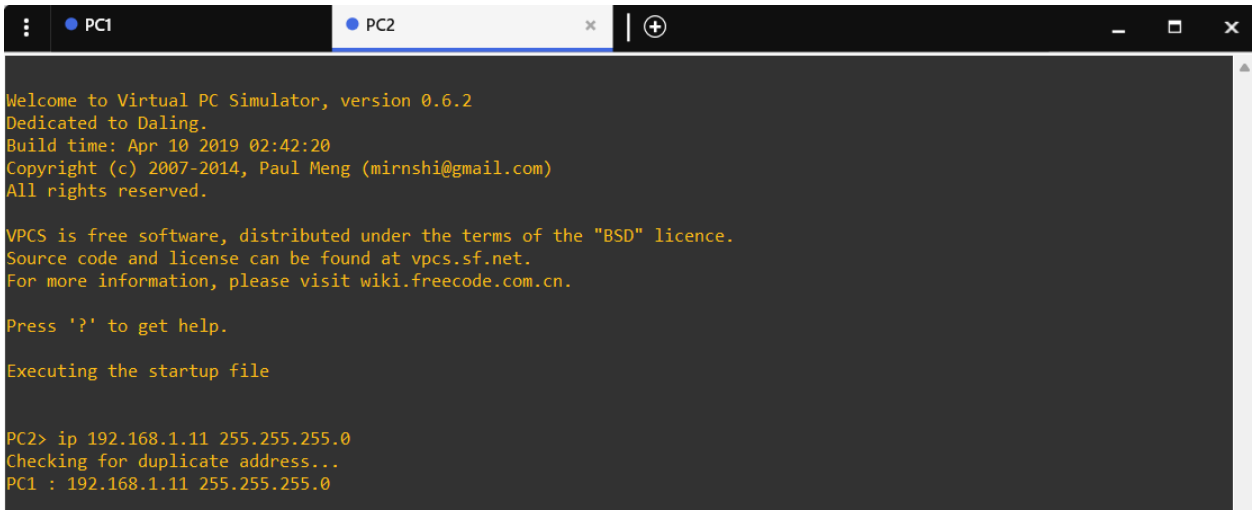

2. Capture and analyze Ethernet frames using Wireshark. Inspect the structure of the frame, including destination and source MAC addresses, Ethertype, payload, and FCS. Use GNS3 or Packet Tracer to simulate network traffic.

Creating a topology in GNS3 :

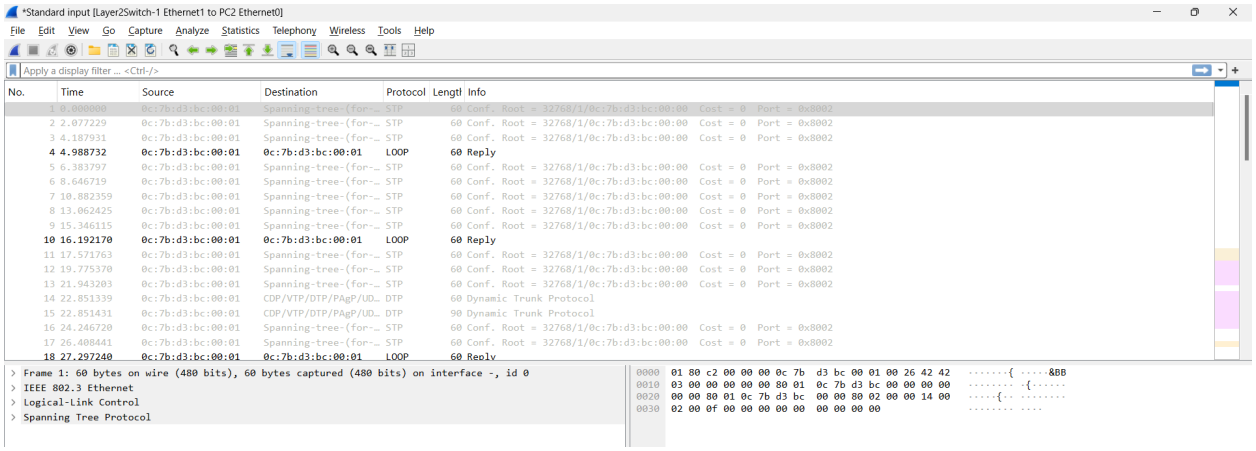


Setting IP address for PC1 and PC2 :

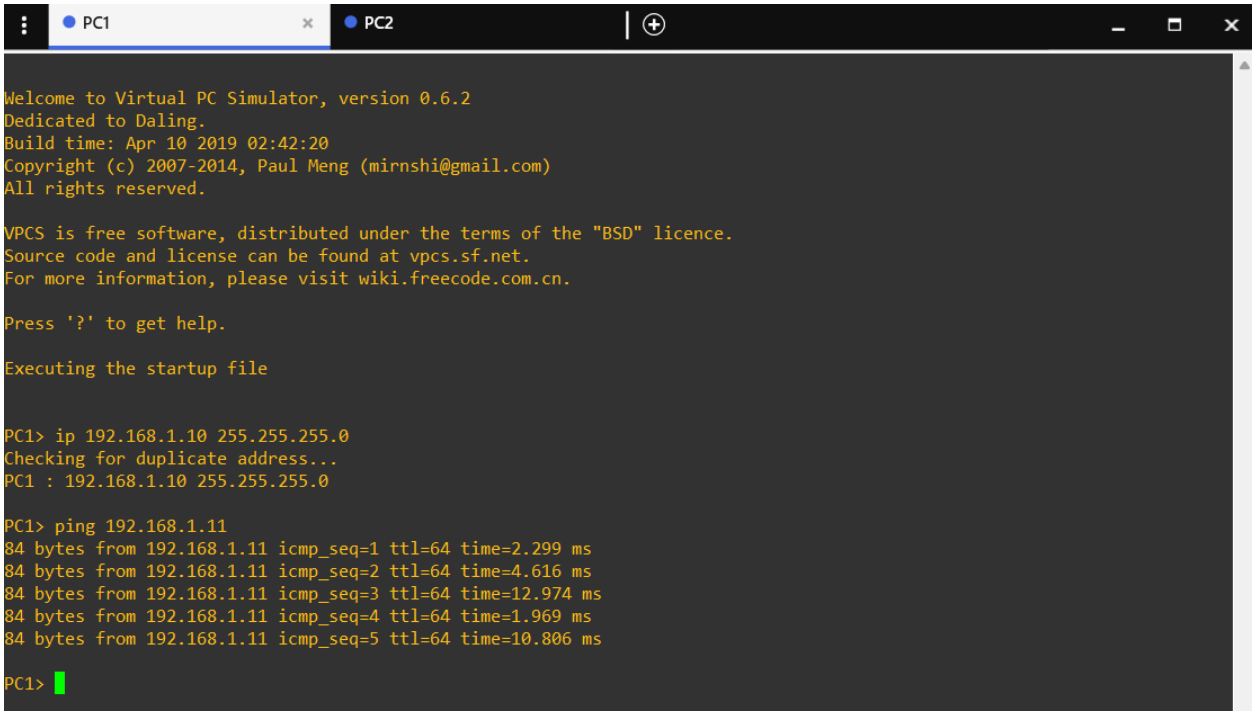


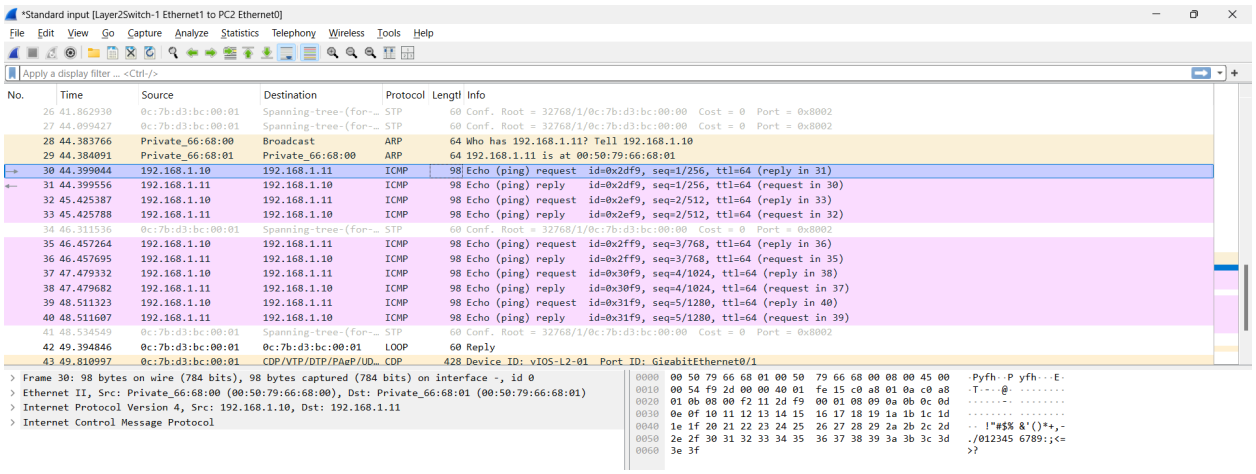

Starting Wireshark :



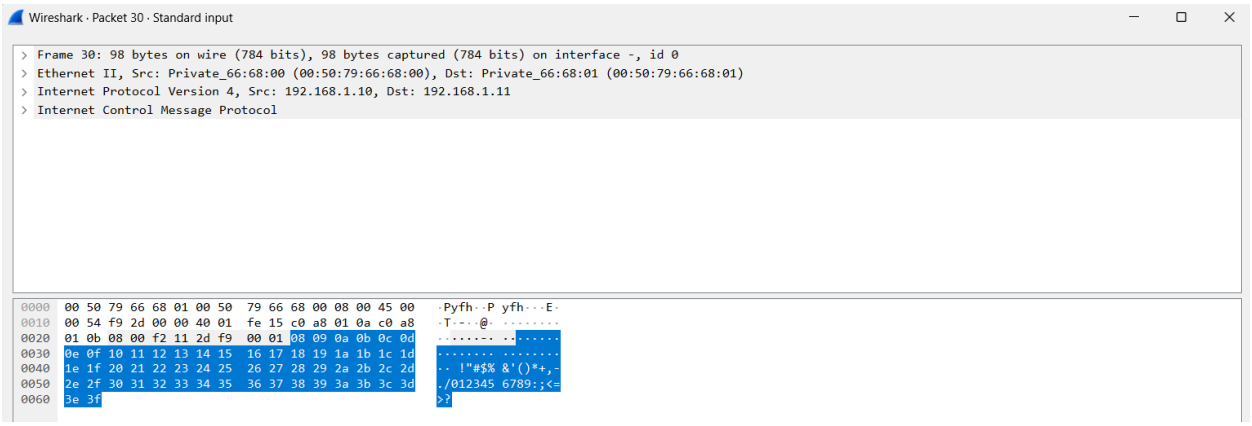
Pinging from PC1 to PC2 :




Capturing Packet :




Now let's inspect the structure :




Frame : It states the whole packet, 30 is the frame number in the captured sequence. The total size of the frame is 98 bytes long. ID 0 is the interface id. It contains interface id, encapsulation type which is ethernet, arrival time (also UTC and epoch)

Ethertype :

```
Ethernet II, Src: Private_66:68:00 (00:50:79:66:68:00), Dst: Private_66:68:01 (00:50:79:66:68:01)
  Destination: Private_66:68:01 (00:50:79:66:68:01)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Source: Private_66:68:00 (00:50:79:66:68:00)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
    [Stream index: 4]
```

Consist of the source and destination MAC address. For each of it, it also specifies the Globally unique address set by the Manufacturer like Cisco or Juniper and the Individual address which is unique and either set automatically or by the network administrator.

IPV4 :

```
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.11
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xf92d (63789)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0xfe15 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.10
```

Consist of the details needed for the networking layer such as the source and destination IP address. It also states IP version, header length, flags, fragmentation offset, TTL, protocol used, Checksum.

ICMP :

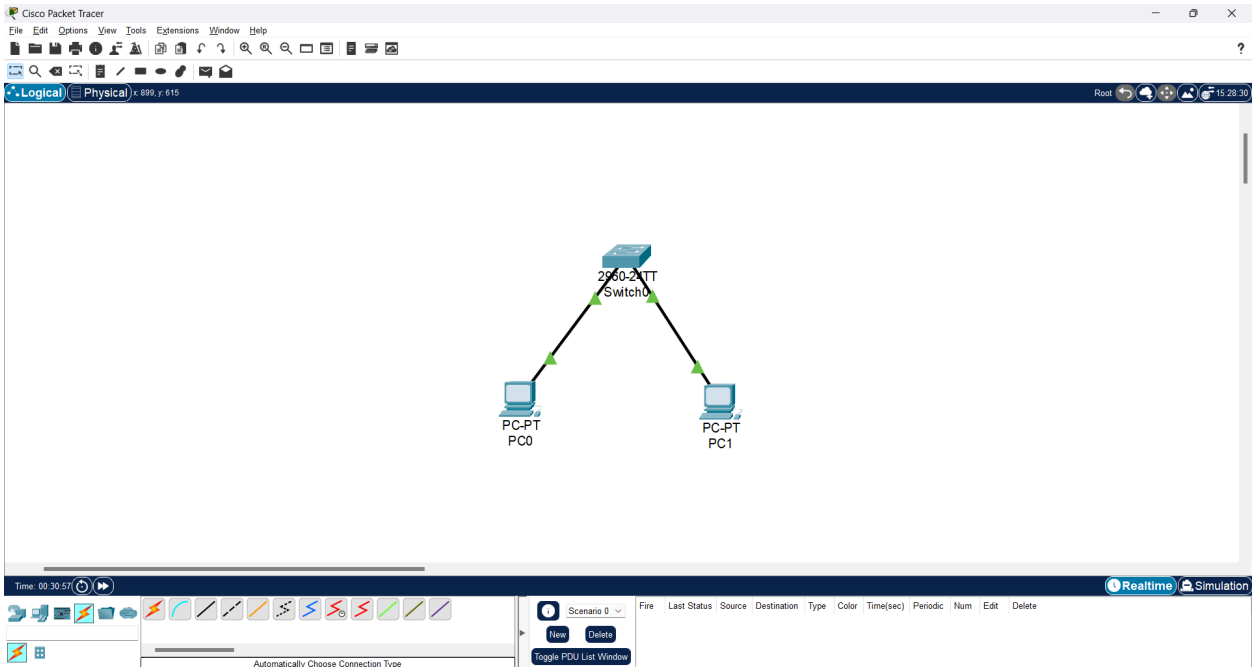
```
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xf211 [correct]
    [Checksum Status: Good]
    Identifier (BE): 11769 (0x2df9)
    Identifier (LE): 63789 (0xf92d)
    Sequence Number (BE): 1 (0x0001)
    Sequence Number (LE): 256 (0x0100)
    [Response frame: 31]
    Data (56 bytes)
```

It consists of the data Payload, checksum, identifier and sequence number for it.

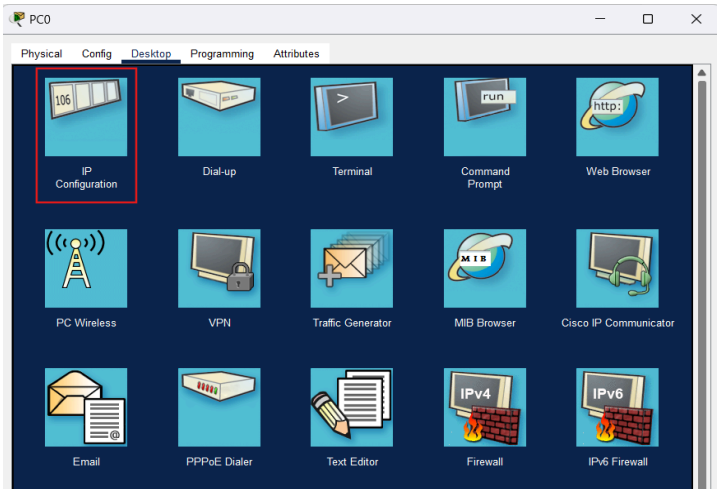
FCS : Frame Check Sequence won't be visible on Wireshark but it is checked by Wireshark. It is used for error detection in ethernet frames. It is basically the checksum which is appended to the end of the frame. It is important for data reliability and data integrity.

3. Configure static IP addresses, modify MAC addresses, and verify network connectivity using ping and ifconfig commands.

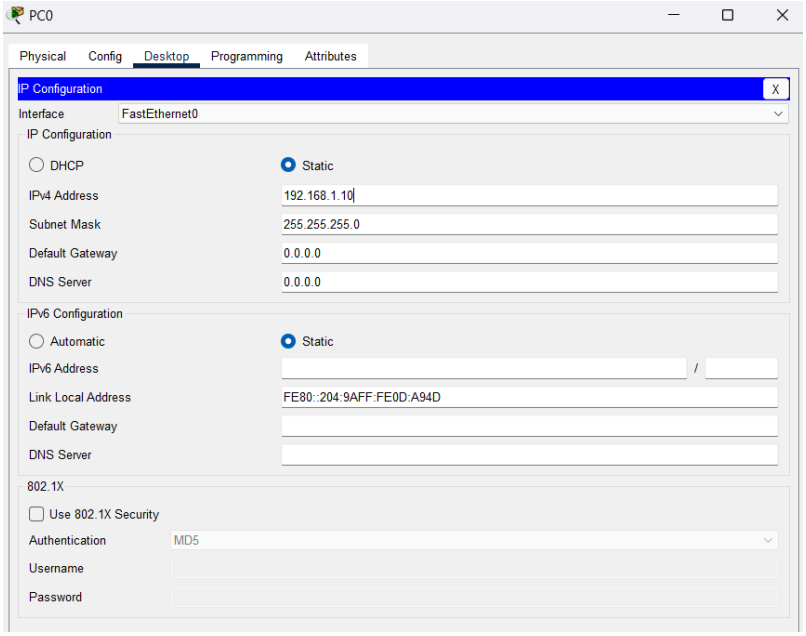
Configuring a network in cisco packet tracer with two PC.



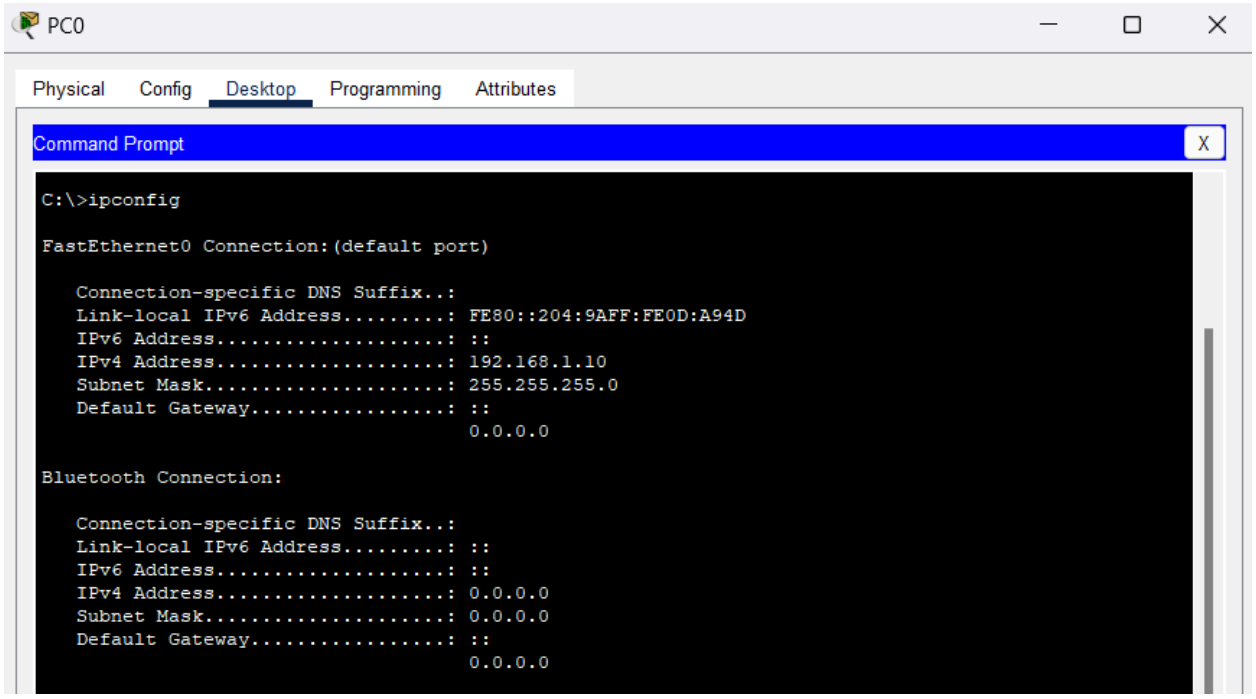
To configure static IP address, go to desktop, and click IP Configuration



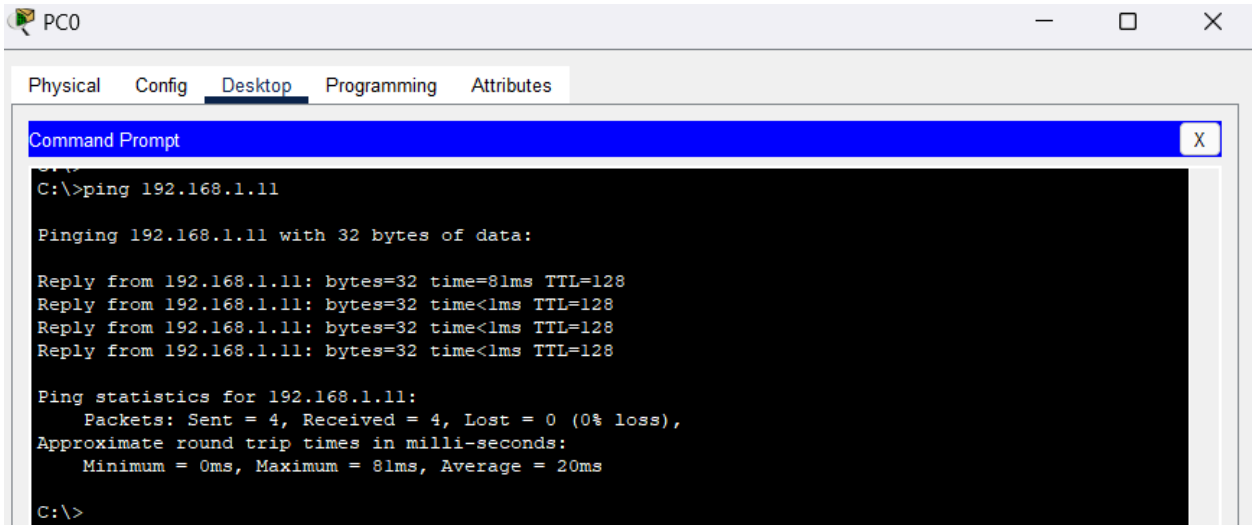

Enter Static IP address and network mask




And configure for other PC also with 192.168.1.11. Now check IP using ipconfig

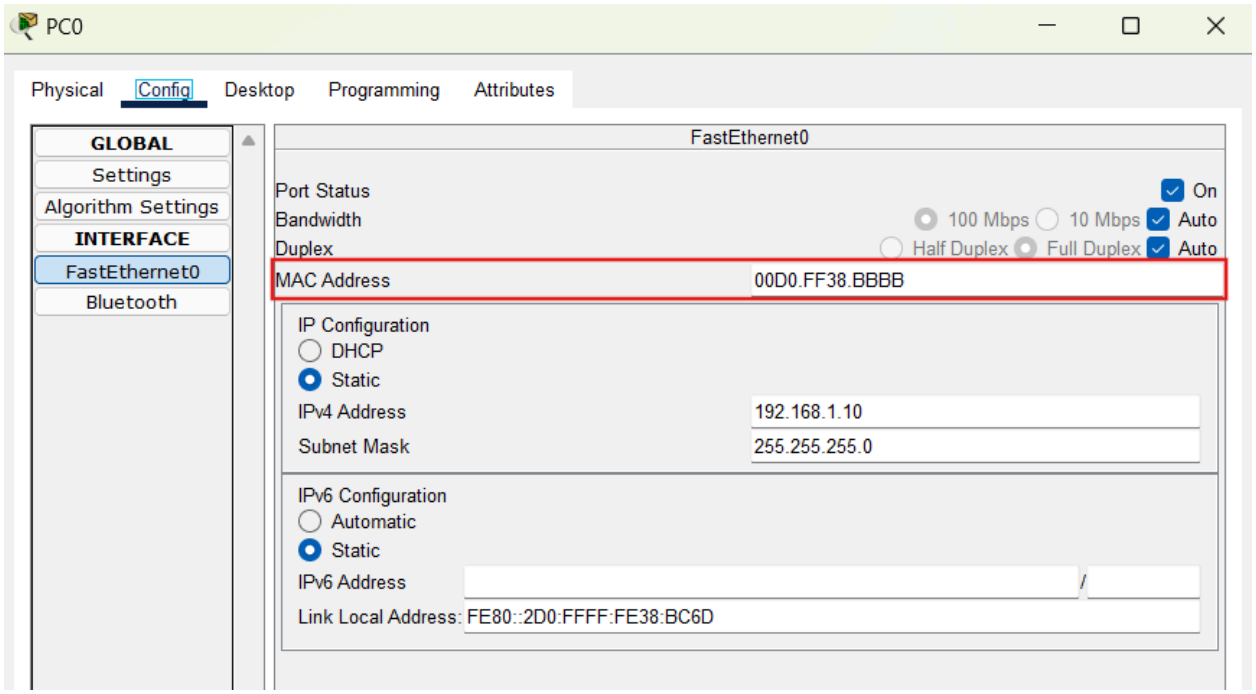



Now lets ping from PC0 to PC1 and verify

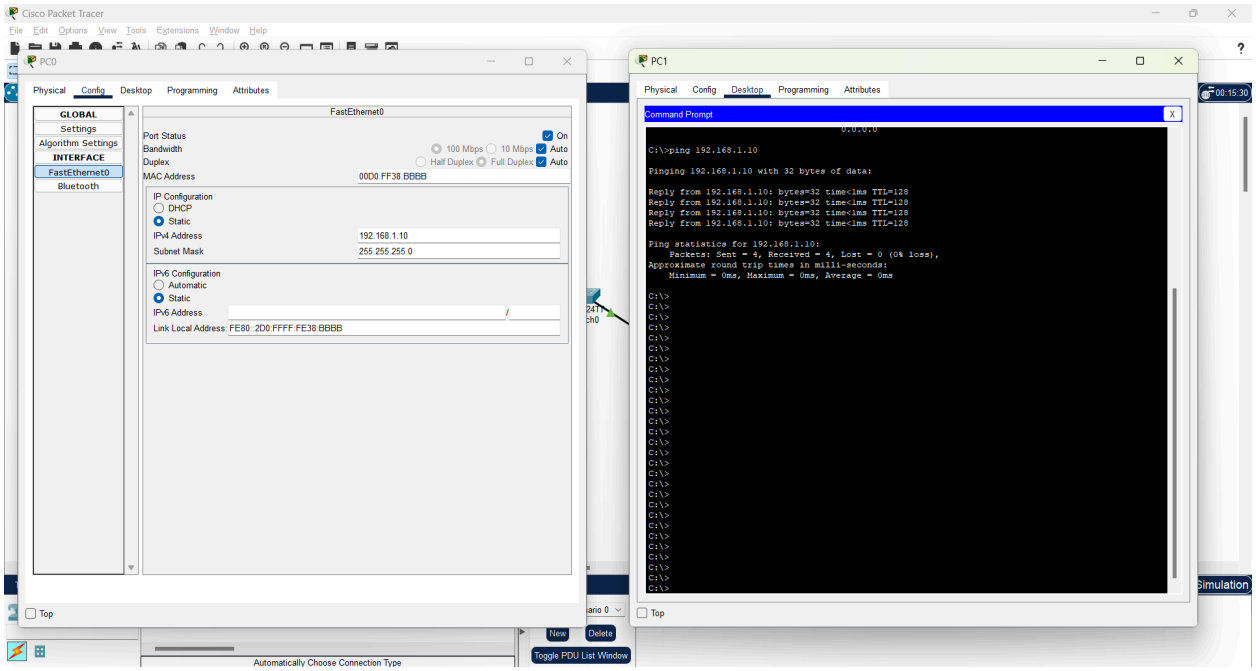



To Modify the MAC Address :

We can change the MAC address by going to the config section of the PC and change the address. After changing lets check it with ping command also.

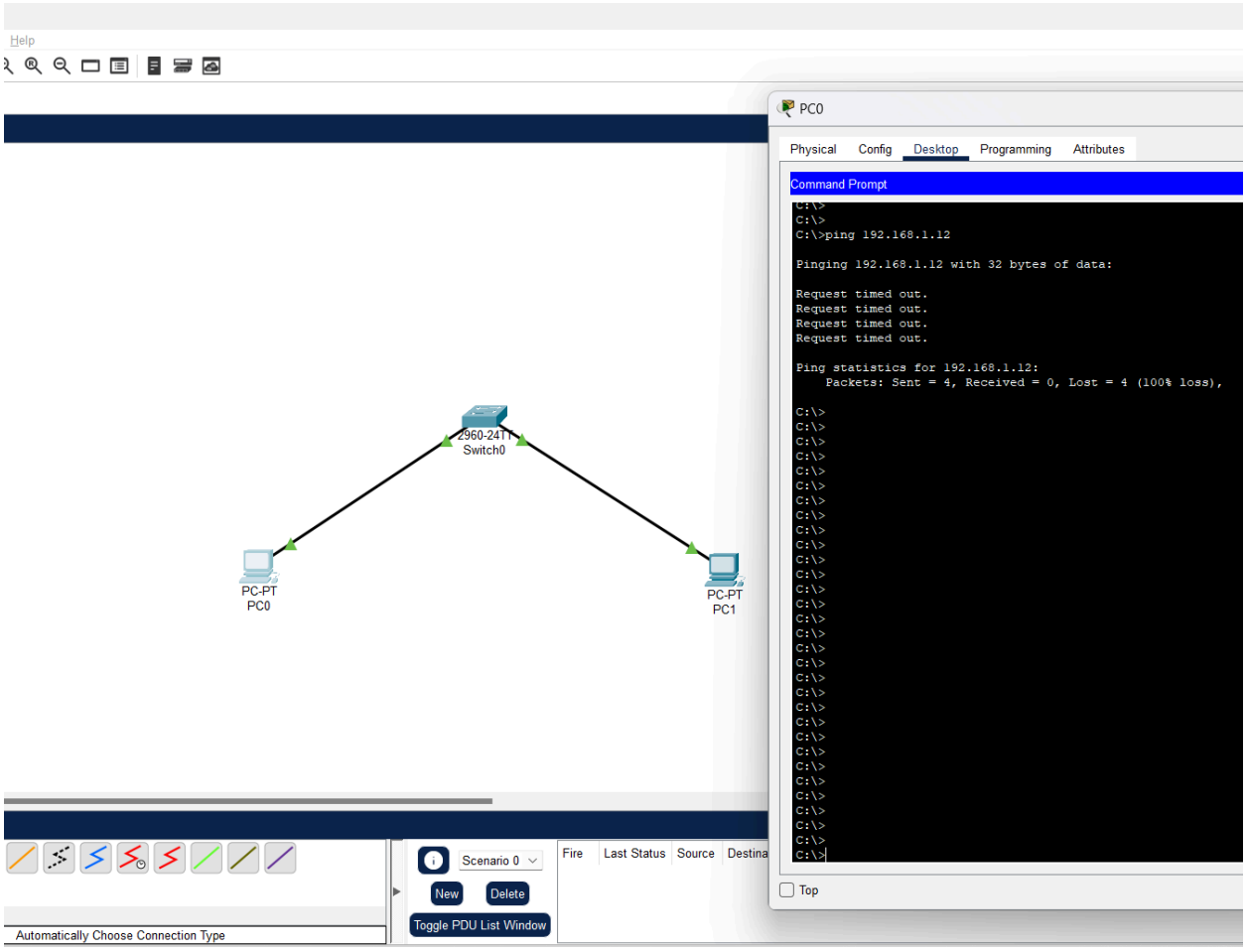



And we see that, it is working after changing the MAC address also :

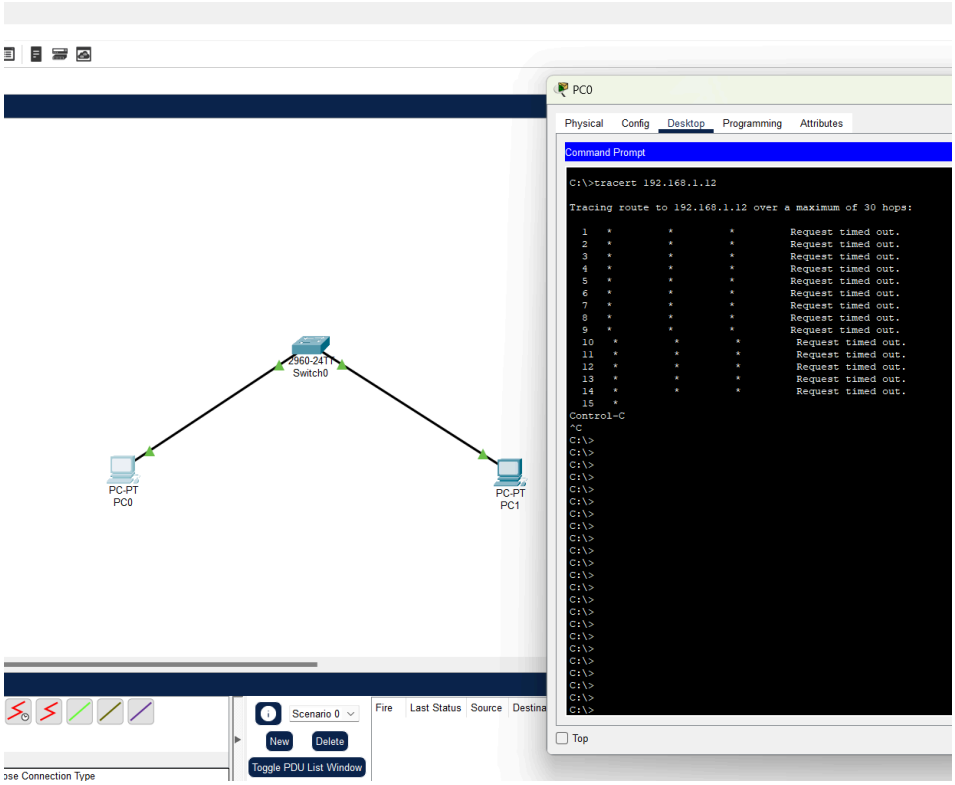


4. Troubleshoot Ethernet Communication with ping and traceroute -> Using cisco packet tracer:
    a. Create a simple LAN setup with two Linux machines connected via a switch. Ping from one machine to the other.
    b. If it fails, use ifconfig to ensure the IP addresses are configured correctly.
    c. Use traceroute to identify where the packets are being dropped if the ping fails.

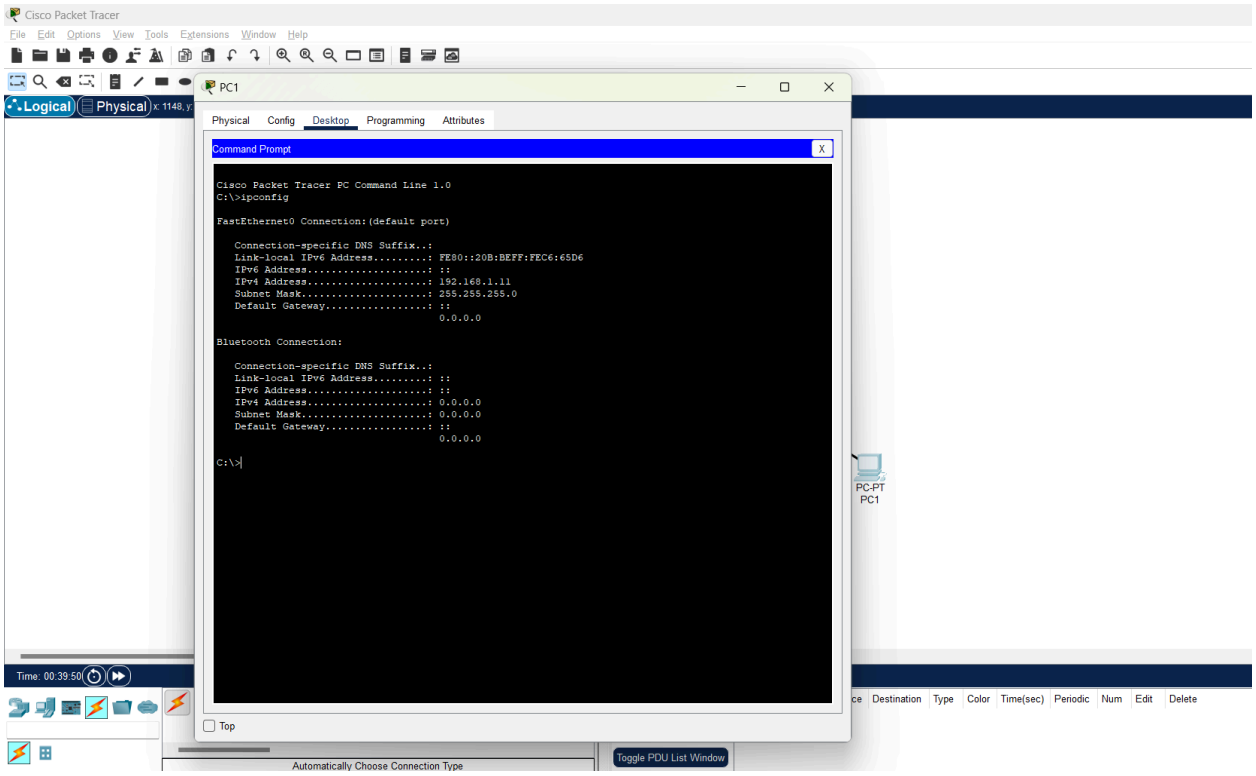
Create a simple LAN with two Machines connected via switch and trying to ping to PC1 but it is not working.



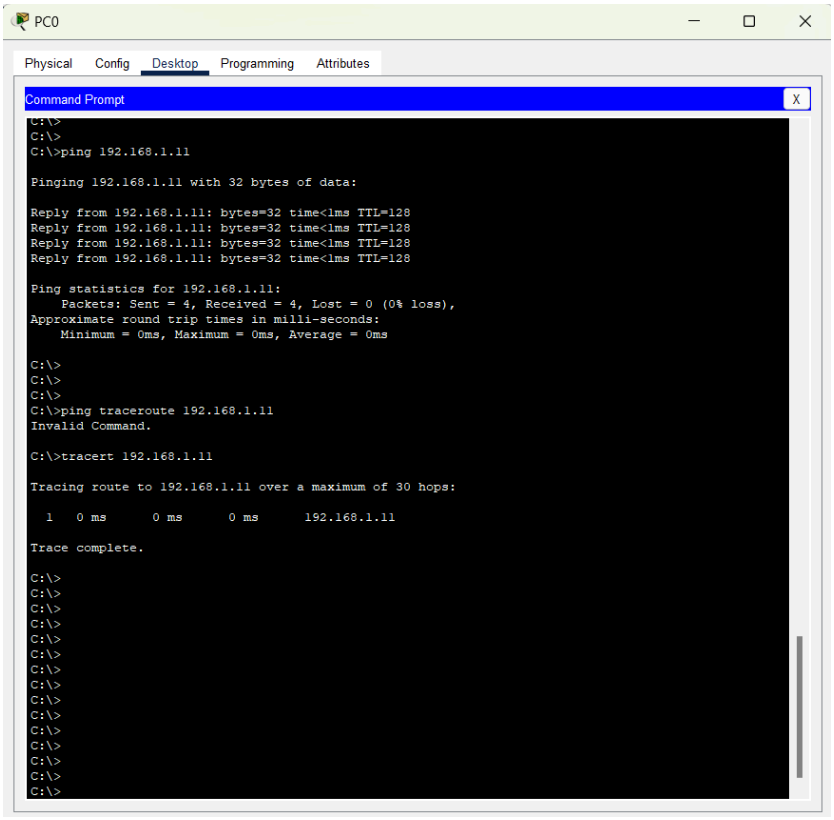
We will try with traceroute the packet and we see the packet is not going anywhere




So we will go to the CMD of the other PC and check ipconfig of it. We see that IP address 192.168.1.11



Now lets again try ping and traceroute with 192.168.1.11 and we see that it is working.




Hence we have troubleshooted the network with ping and traceroute command.

5. Research the Linux kernel's handling of Ethernet devices and network interfaces. Write a short report on how the Linux kernel supports Ethernet communication (referencing kernel.org documentation).

In the linux kernel, the ethernet devices and the network interface are handled by the netdev which is a crucial component of managing network interfaces and all network related operations.

It consists of functionalities such as device drivers for NIC, management of network interfaces, and implementation of network protocols.

Network interface objects are software abstractions that represent network interfaces in the Linux kernel.

These items serve as an essential bridge connecting a system's various network hardware components and the networking stack of the kernel.

They allow the kernel to efficiently control network traffic by storing crucial data about each interface, including its MAC address, IP address, and operational status.

The kernel can receive incoming data from the network and route outgoing data to the appropriate interface for transmission by interacting with these objects.

Administrators may easily interact with and control these network interface objects by using the `ip` command, which enables them to monitor network traffic, configure IP addresses, and bring interfaces up or down.

Every Ethernet interface in Linux is controlled by a driver and is shown as a network device object in the kernel. Important details like the MAC and IP addresses are contained in this item.

These objects are used by the kernel's network stack to manage Ethernet frame transmission and reception, guaranteeing effective network connection. These interfaces can be managed and configured by users using the ip command.

Reference :

https://web.git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/net/l2tp/l2tp_eth.c?h=v6.14-rc5

https://docs.kernel.org/process/maintainer-netdev.html#netdev-faq

6. Describe how you would configure a basic LAN interface using the ip command in Linux (kernel.org).

   To configure a LAN interface using the ip command :

   Interface names can be eth0, eth1, en0 or anything similar to that.

   Followed by which you can add or delete an LAN interface or a network interface using the IP command with the following commands :
   sudo ip addr add 192.168.1.100/24 dev eth0

   To delete an interface :
   sudo ip route delete 10.0.0.0/24 via 192.168.1.1 dev eth0

   To modify the default gateway :
   sudo ip route add default via 192.168.1.254 dev eth0

7. Use Linux to view the MAC address table of a switch (if using a Linux-based network switch). Use the bridge or ip link commands to inspect the MAC table and demonstrate a basic switch's operation.

To view the MAC address of the machine use :

```
tkgowtham@tkgowtham-VirtualBox:~$ ip link show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:d8:25:4b brd ff:ff:ff:ff:ff:ff
tkgowtham@tkgowtham-VirtualBox:~$
```

To view the MAC table we will use the neighbour option with ip command :

```
tkgowtham@tkgowtham-VirtualBox:~$ ip neigh show
192.168.1.1 dev enp0s3 lladdr f8:c4:f3:c3:f3:d0 DELAY
fe80::1 dev enp0s3 lladdr f8:c4:f3:c3:f3:d0 router DELAY
tkgowtham@tkgowtham-VirtualBox:~$
```

Using bridge command to view the MAC table :

```
tkgowtham@tkgowtham-VirtualBox:~$ bridge fdb show dev enp0s3
01:00:5e:00:00:01 self permanent
33:33:00:00:00:01 self permanent
33:33:ff:d6:33:7f self permanent
01:00:5e:00:00:fb self permanent
33:33:ff:cd:80:1f self permanent
33:33:00:00:00:fb self permanent
33:33:ff:f7:83:e1 self permanent
tkgowtham@tkgowtham-VirtualBox:~$
```

Some switch operations that can be performed using ip command are :

Add a bridge : sudo ip link add name <name> type bridge
Add port to bridge : sudo ip link set dev enp0s3 master <name>
Delete a bridge : sudo ip link delete <name>
View a bridge : sudo ip link show <name>