NETWORKING TRAINING - MODULE 3 & 4

Akash S, embedUR Systems

Task 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

Explanation:

- Installed GNS3 & Cisco Packet Tracerand necessary packages. Created a new workspace inside the GNS3 & Cisco Packet Tracer.
- Added two PCs and a switch.
- Configured IP: **192.168.1.10** for PC1 and **192.168.1.20** for PC2

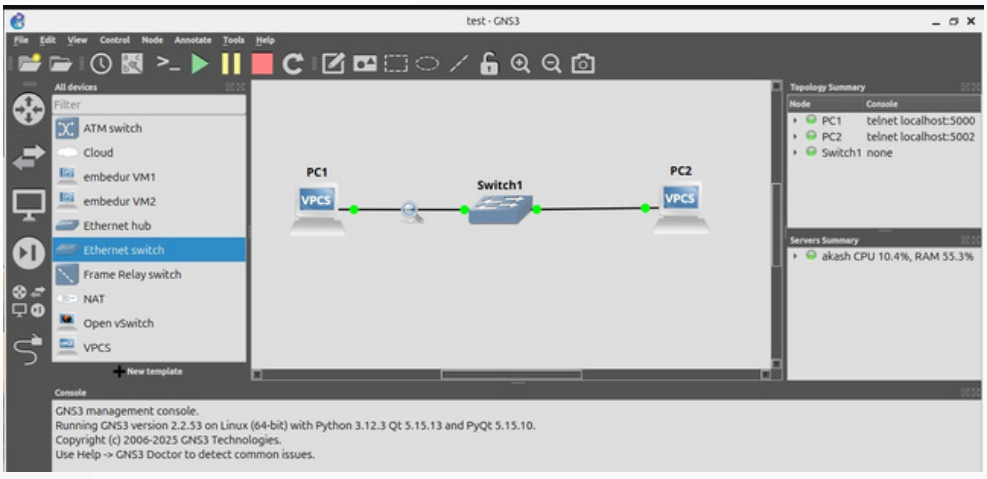
- Pinged PC2 from PC1 to check the connectivity
- Added two PCs and a switch.
- Configured IP: **192.168.1.10** for PC1 and **192.168.1.20** for PC2
- Checking the MAC address of the table using

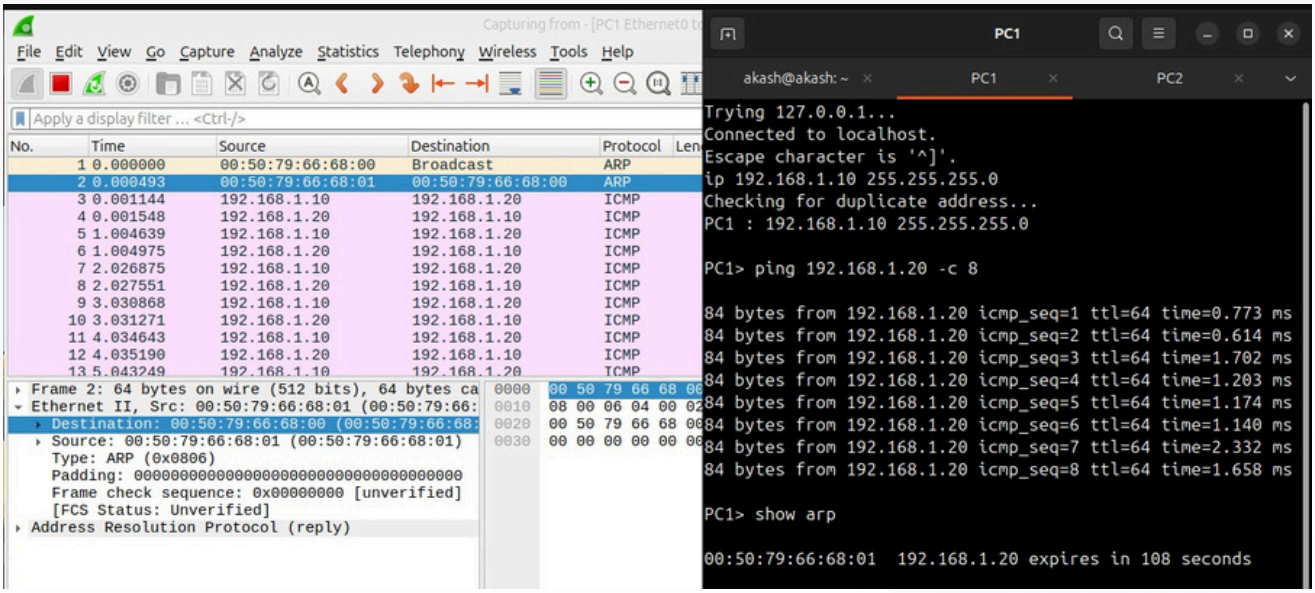
  $ **show mac address-table**

- Packets are captured using Wireshark tool and Simulation is verified

Outputs:

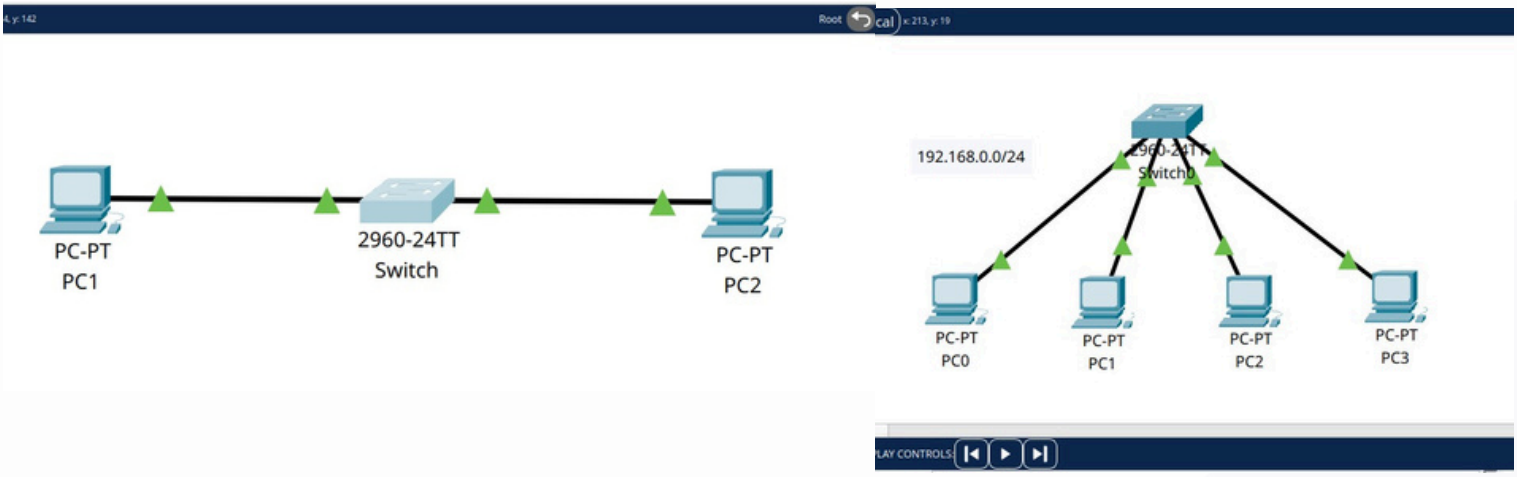
GNS3

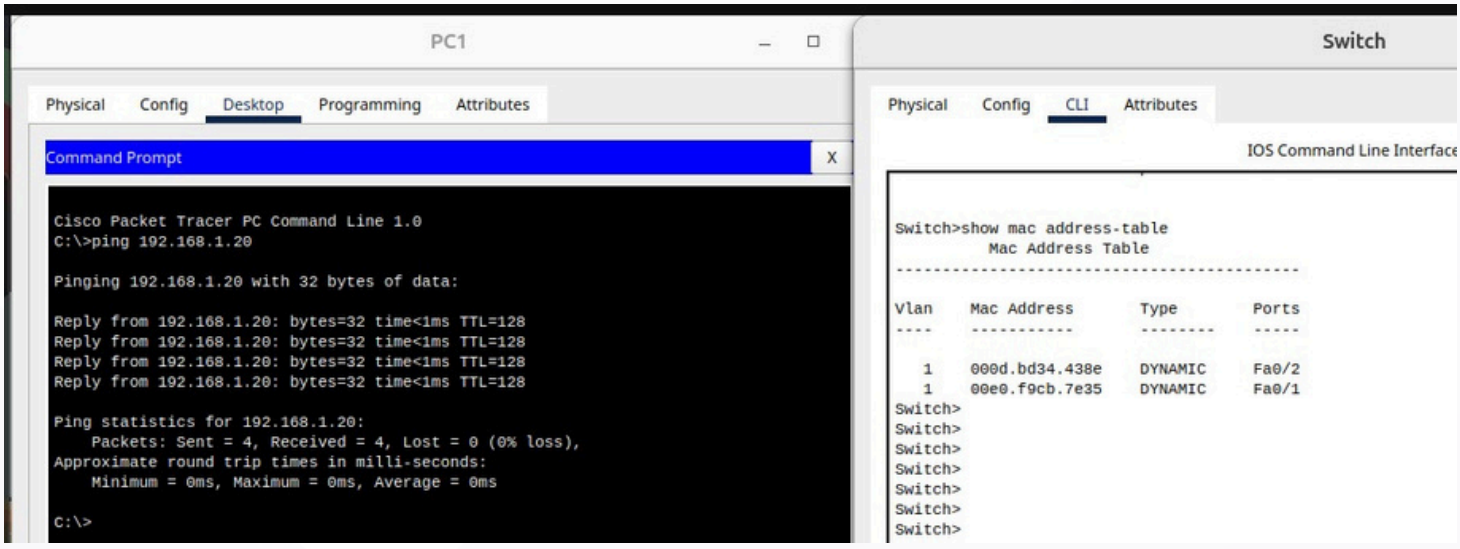




Wireshark Packet Capture - Pinging PC2 from PC1

Cisco Packet Tracer:





MAC Address table of Switch

Task 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.

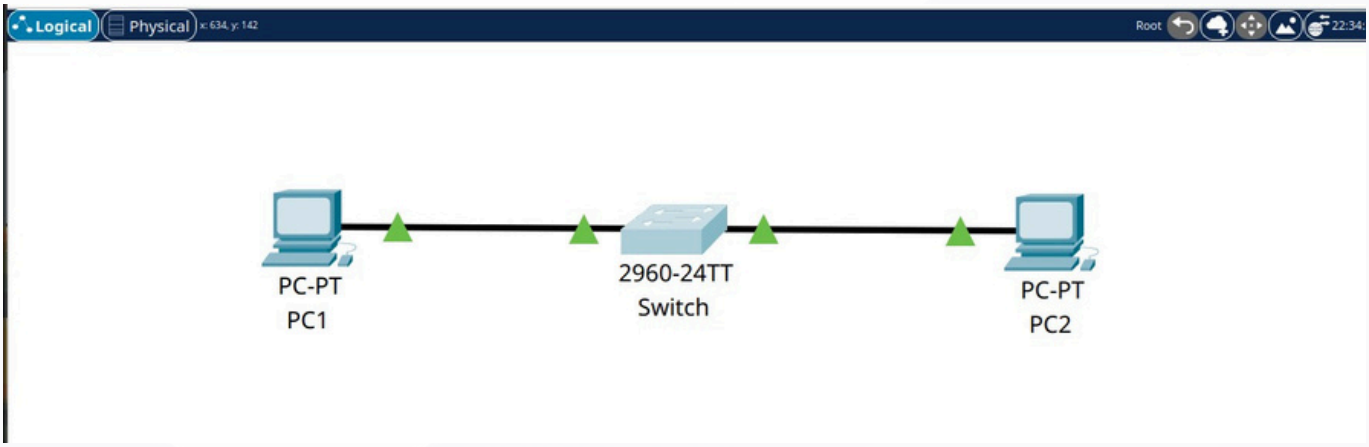
Explanation:

- In the Cisco Packet Tracer, create a network setup with two VPCs and a switch.
- Configured IP: **192.168.1.1** for PC1 and **192.168.1.2** for PC2

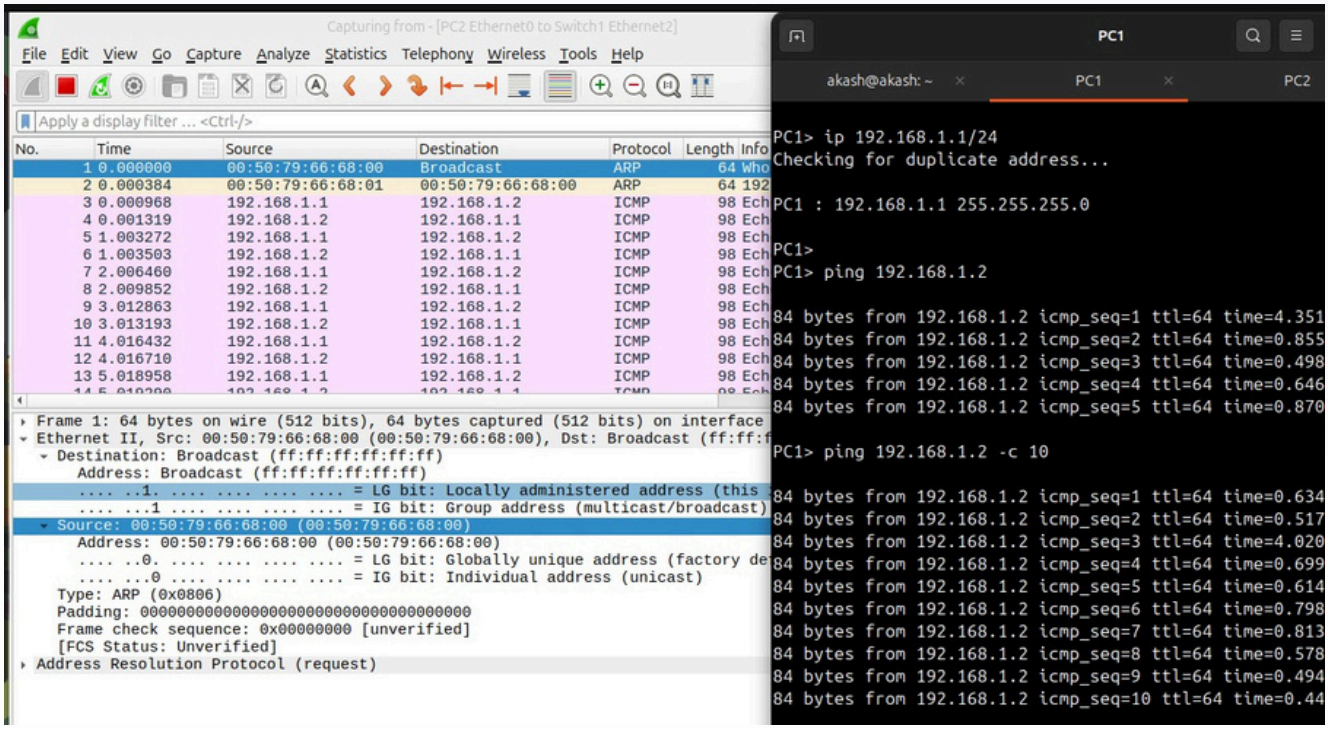
- Ping PC2 (192.168.1.2) from PC1 using Ping command

  $ **ping 192.168.1.2**
- Once ping is successful, open the simulation mode and play.
- We can now examine the movement of packets from switch to PCs
- We can capture the packets in **Wireshark** and on packets to see Detailed Ethernet Frame Format including Source MAC, Destination MAC, FCS, Padding, Payload, TTL, etc...

Outputs:

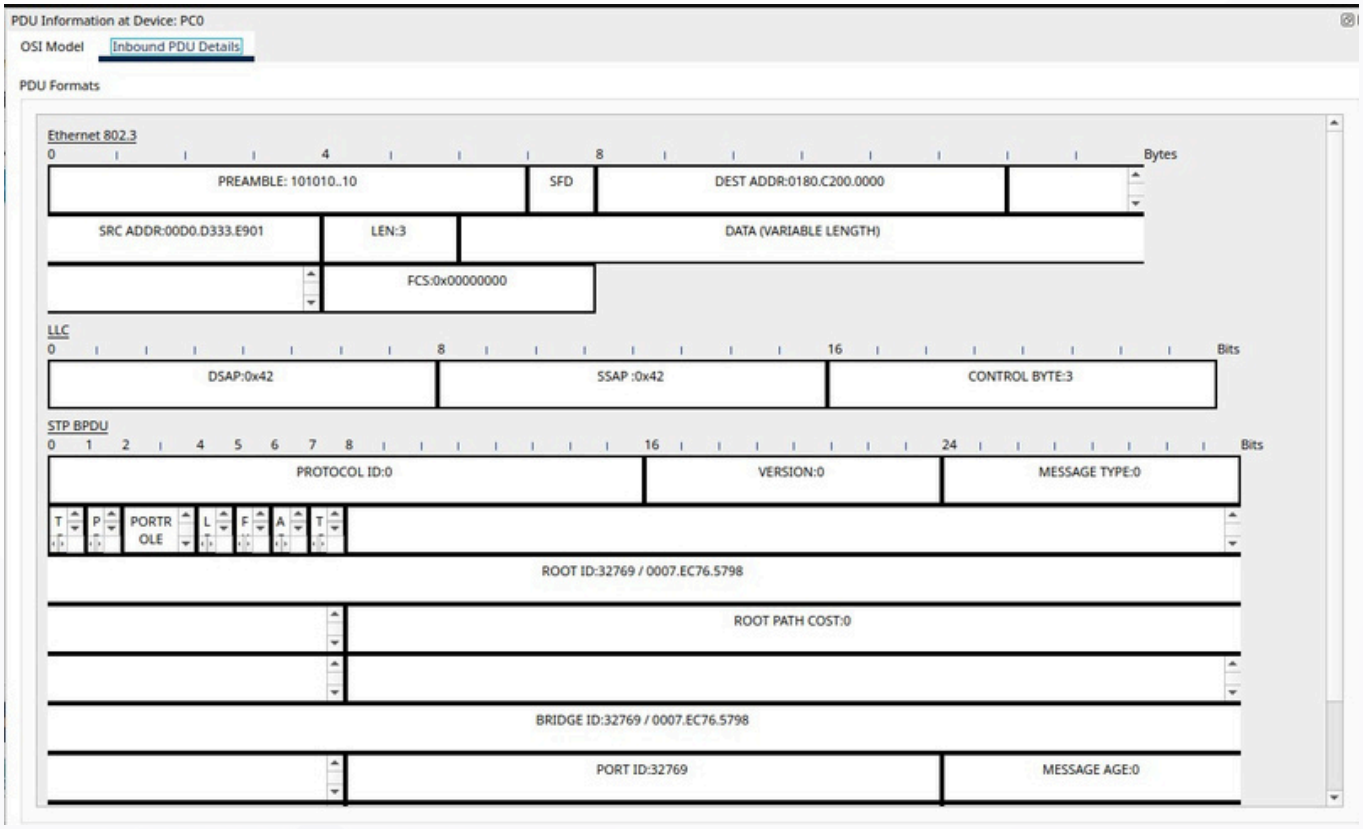


Wireshark:



Pinging PC2 from PC1

Message Frame:



Detailed Ethernet Frame Format

Task 3:

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

Explanation:

Static IP:

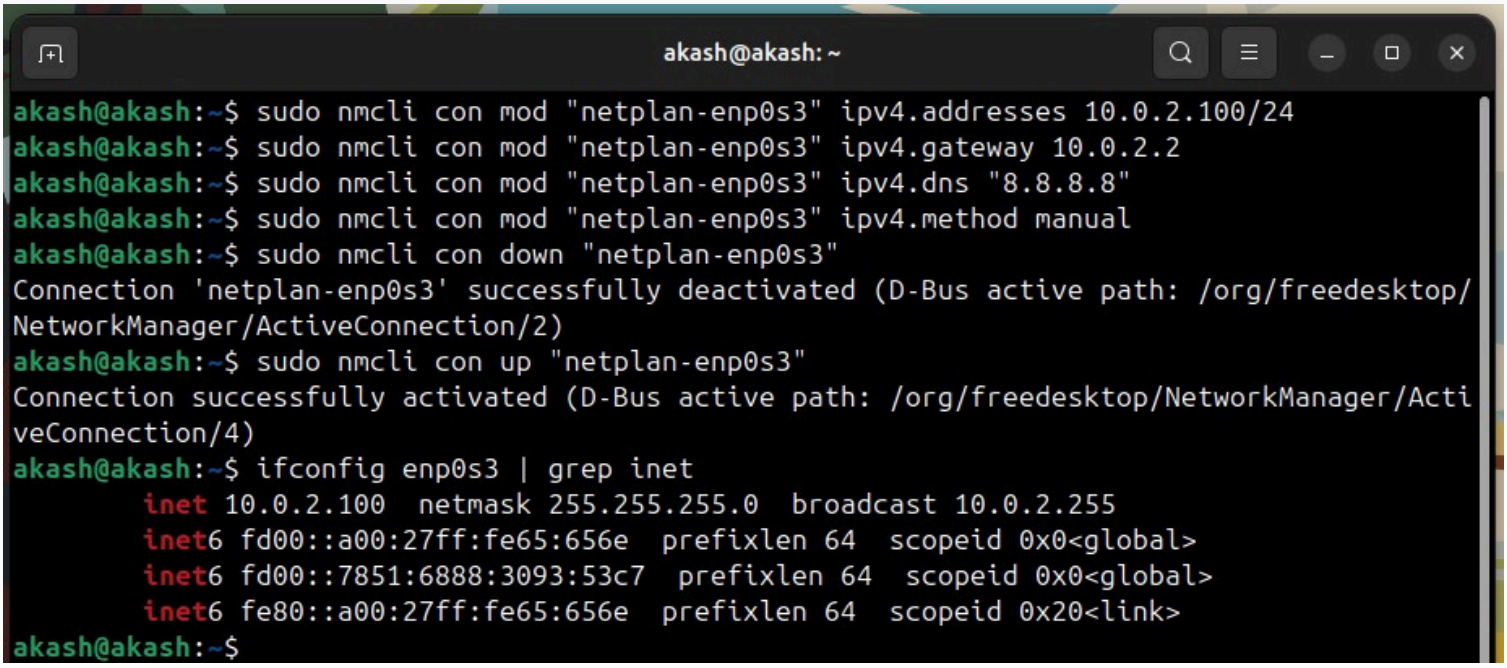
- Created a clone of a Virtual Machine for this task - **embedUR VM**
- First, the interface is found using the command
  $ **ifconfig**
- In my case, the interface is **enp0s3.**
- Next, the static IP is set using NAT/Bridged adapter.
  - ipv4 address: **10.0.2.100/24 | 192.168.29.10**
  - default gateway: **10.0.2.2 | 192.168.29.1**
- Once it is set, we can confirm the static IP using the command,
  $ **ifconfig enp0s3 | grep inet**

MAC address:

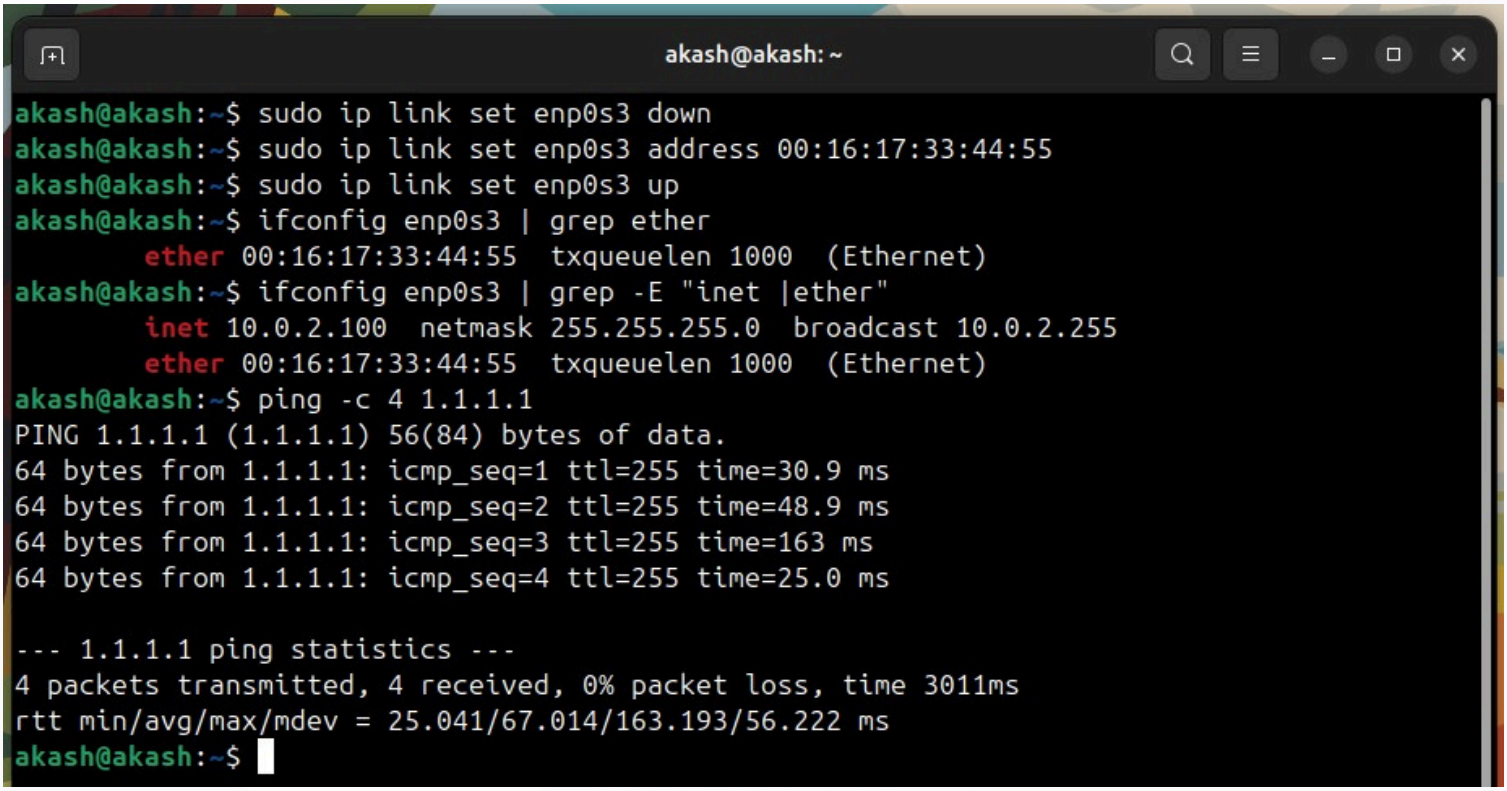
- A MAC address is the one which is set by the IEEE and the manufacturer.
- If we change the MAC address, we can ensure privacy - meaning it is hard to track the device.
- First step is to find the interface using the command,
  $ **ifconfig**
- In my case the interface is **enp0s3**
- To configure MAC address,
  - Make the interface down
    $ **sudo ifconfig enp0s3 down**
  - Set the address
    $ **sudo ifconfig enp0s3 hw ether 00:11:22:33:44:55**
  - Bring the interface up
    $ **sudo ifconfig enp0s3 up**

Outputs:

Static IP:



MAC Address:



Checking Connectivity using Ping

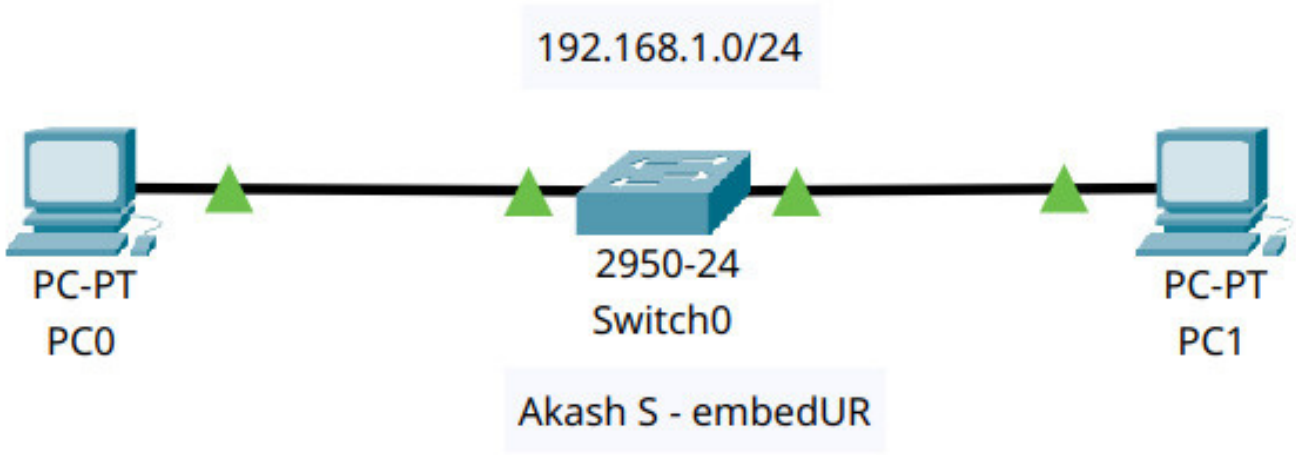
Task 4, 5, 6, 7:

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

Explanation:

LAN setup:

- Created a simple **Local Area Network** setup in Cisco Packet Tracer with Two VPCs and a Switch (2950-24).
- Configured IP:
  - PC1: **192.168.1.10**
  - PC2: **192.168.1.20**
- To check the connectivity between the end devices, I pinged from PC1 to PC2 and from PC2 to PC1.

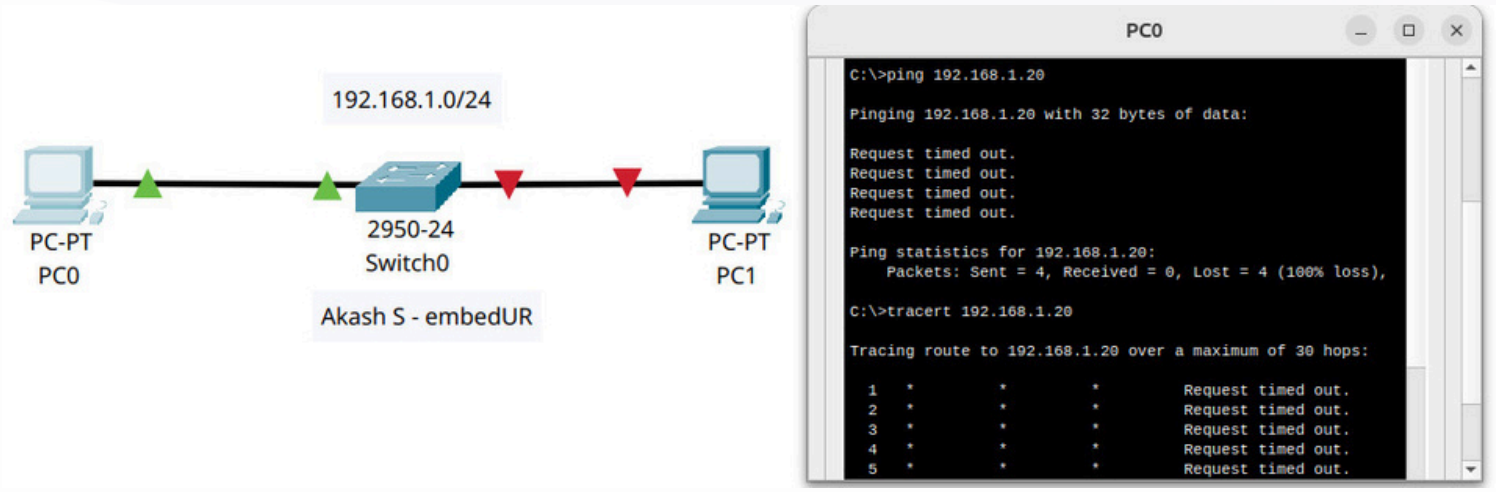


Check the configuration

- Checked whether the IPs are configured correctly using
  - **ifconfig enp0s3 | grep inet**

Troubleshooting:

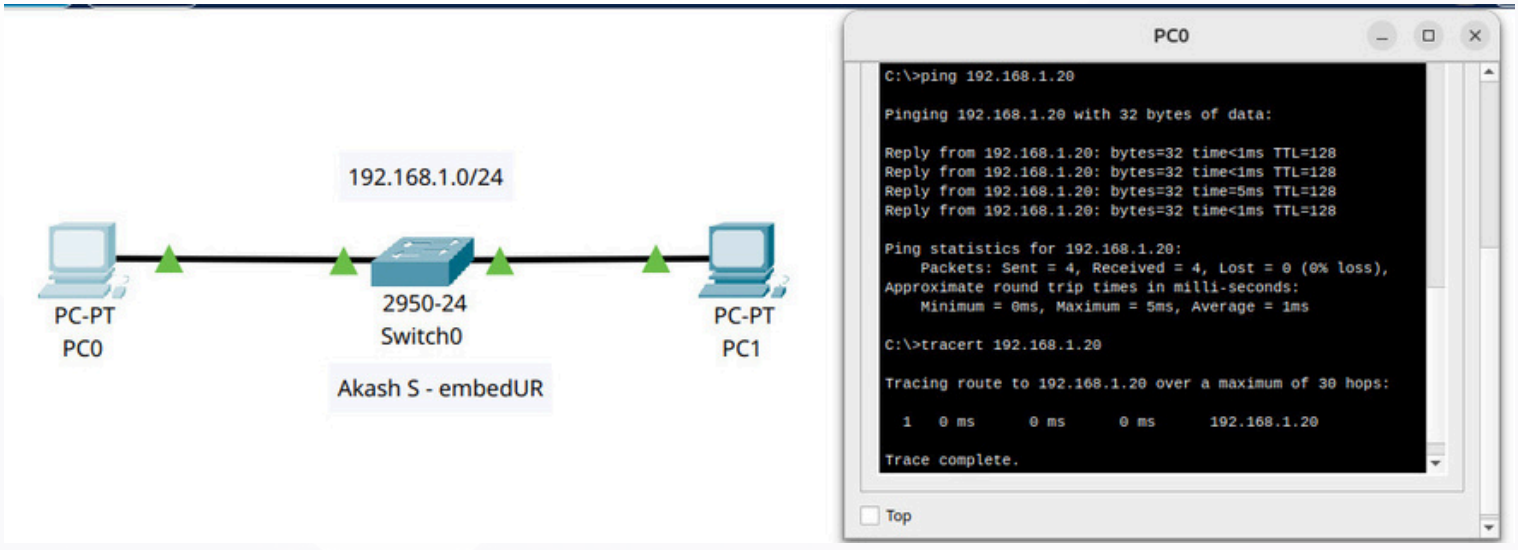
- From switch, inside CLI, disabled the link between Switch and PC2
- So now the ping from PC1 to PC2 fails as the link is down
- Since it fails, the **ping** shows *"Request timed out"*
- Also, we can't find the routes using tracert, it shows "* * *"



*Failure - Request Timed Out! and * * **

End Results:

- Upon enabling the link between the switch and the PC2, it works!
- Pinging from PC1 to PC2 to check the connectivity
- Finding the intermediate routes using **tracert**
- We can know upto 30 intermediate hops



Success - ping and tracert

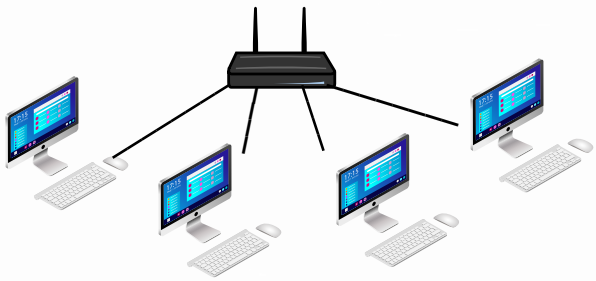
Task 8:

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).

Report:

Introduction to Ethernet and Linux:

- I found out that Ethernet is a technology that connects devices in a local network (like a LAN at home or school).
- It’s super important because it lets computers talk to each other using cables or Wi-Fi.
- The Linux kernel, which is the core of the Linux system, has special features to support Ethernet.
- I learned this is part of its “networking subsystem” that makes communication smooth.



LAN

Device Detection & Driver Loading:

- The kernel uses the PCI subsystem to spot Ethernet devices and matches them to drivers using unique PCI IDs.
- It then loads the right driver (e.g., for Intel or Realtek cards) and sets up a network interface like eth0.
- I can configure this interface with tools like ip or ifconfig to assign an IP address.

Data Handling & Management tools:

- The kernel's networking stack processes data through four layers, adding or removing headers for transmission.
- Tools like ifconfig, ip, and ethtool let me manage interfaces, check settings, or troubleshoot issues.
- This setup ensures smooth data flow and easy network control.

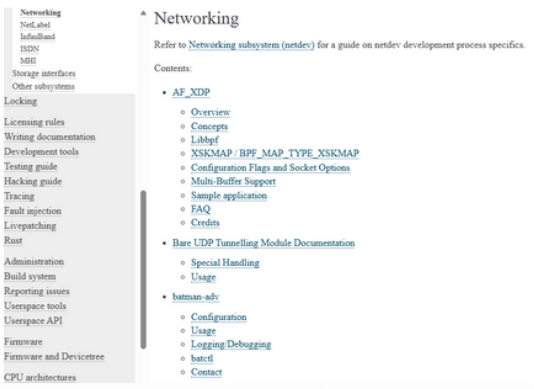
ifconfig

ip

ethtool

Kernel handling Network Data:

- I checked out kernel.org for more details:
- The Networking Documentation explains the whole system.
- The Ethernet Drivers Section talks about specific drivers.
- These pages helped me confirm what I learned and dig deeper if I want.

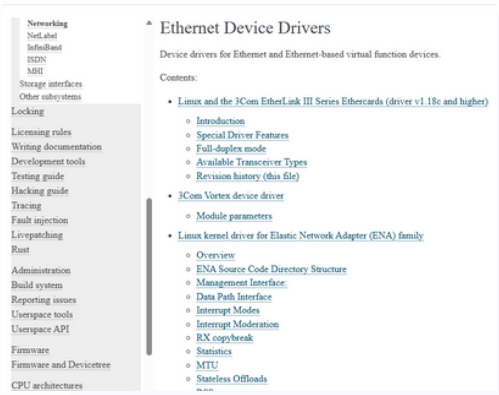


Networking

Refer to Networking subsystem (netdev) for a guide on netdev development process specifics.

Contents:

- AF_XDP
  - Overview
  - Concepts
  - Libbpf
  - XSKMAP / BPF_MAP_TYPE_XSKMAP
  - Configuration Flags and Socket Options
  - Multi-Buffer Support
  - Sample application
  - FAQ
  - Credits
- Bare UDP Tunnelling Module Documentation
  - Special Handling
  - Usage
- batman-adv
  - Configuration
  - Usage
  - Logging/Debugging
  - batctl
  - Contact



Ethernet Device Drivers

Device drivers for Ethernet and Ethernet-based virtual function devices.

Contents:

- Linux and the 3Com EtherLink III Series Ethercards (driver v1.18c and higher)
  - Introduction
  - Special Driver Features
  - Full-duplex mode
  - Available Transceiver Types
  - Revision history (this file)
- 3Com Vortex device driver
  - Module parameters
- Linux kernel driver for Elastic Network Adapter (ENA) family
  - Overview
  - ENA Source Code Directory Structure
  - Management Interface:
  - Data Path Interface
  - Interrupt Modes
  - Interrupt Moderation
  - RX copybreak
  - Statistics
  - MTU
  - Stateless Offloads

My Takeaway:

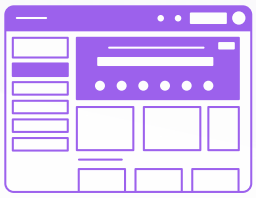
- I think it's awesome how the Linux kernel handles Ethernet so smartly.
- It finds devices, loads drivers, sets up interfaces, and moves data—all automatically!
- The tools and docs make it easier to understand and control.
- I feel like I get the basics now and could even try setting up a network myself.

Task 9:

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

Documentation:

Find Interface:



- Open a terminal and type: $ **ip link show**.
- Find the available interface.
- In my case, the interface is **enp0s3**
- Also we need to identify the status (UP or DOWN)

Bring the interface UP:

- Use the command: $ **ip link set enp0s3**
- This activates the interface so it's ready to use.
- Verify it's up with: ip link show <interface>.

Bring the interface UP:

- Set a static IP with: (e.g., **ip addr add 192.168.1.10/24 dev enp0s3**
- The /24 means a subnet mask of 255.255.255.0 (LAN)
- Check it's applied: **ip addr show enp0s3**.

Confirm Routing:

- Confirm routing with: $**ip route show**

Test the configuration:

- Ping using **ping <ip>**
- If it works
    - Successfully set
- If it fails
    - Troubleshoot with **ip** command

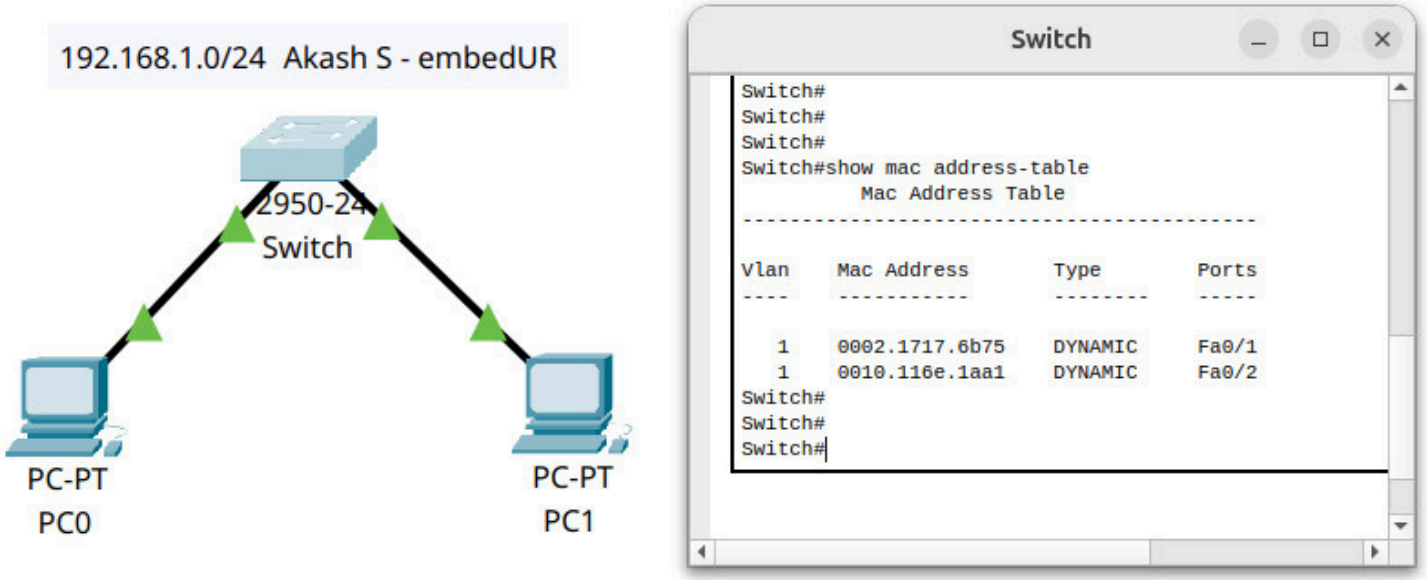
Task 10:

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.a

Explanation:

Switch MAC table:

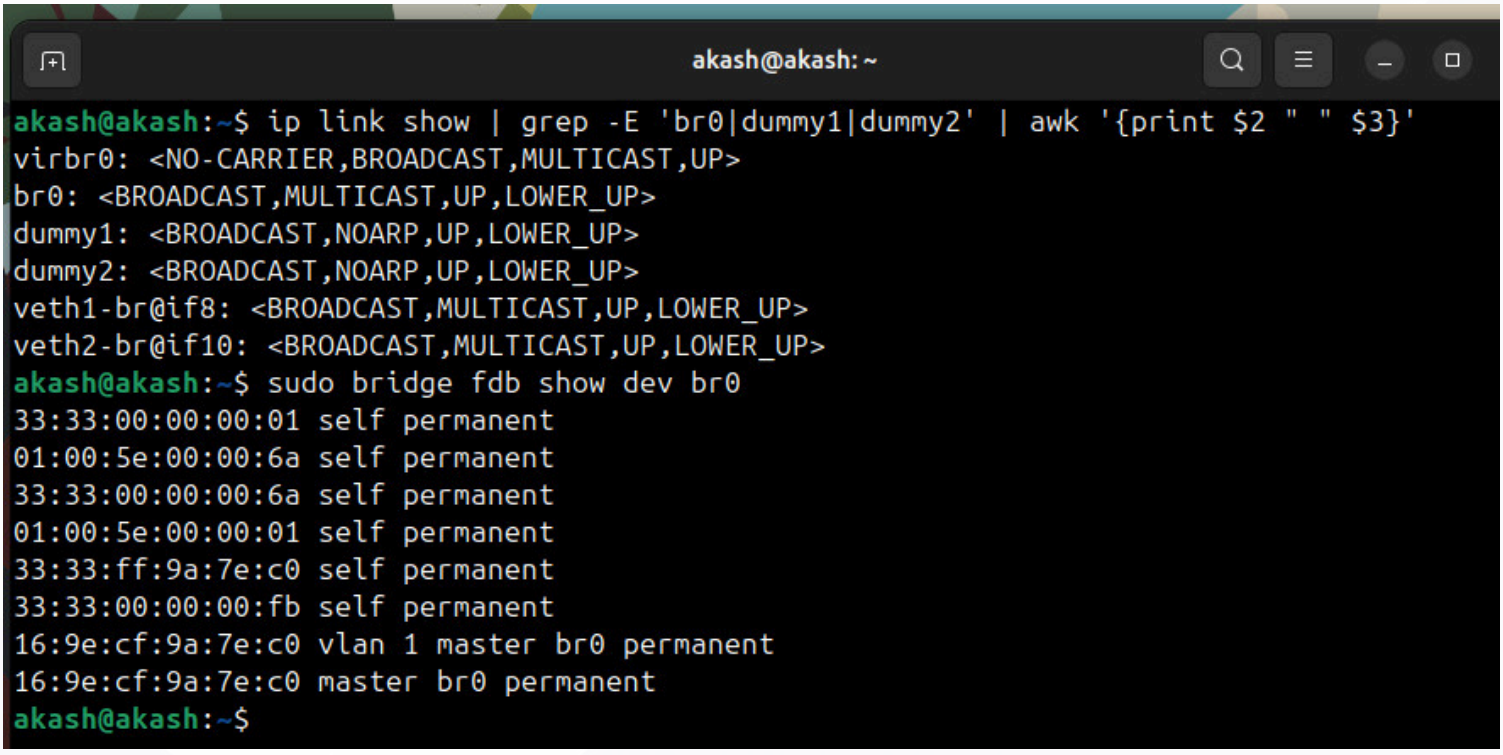
- Created a simple LAN setup in Cisco Packet Tracer with Two VPCs and a Switch (2950-24).
- Configured IP:
    - PC1: **192.168.1.10**
    - PC2: **192.168.1.20**
- To check the connectivity between the end devices, I pinged from PC1 to PC2 and from PC2 to PC1.
- Click on the switch to see the MAC address by executing the command
    - $ **show mac address-table**



Switch MAC Address table

Bridge:

- Create a bridge named br0 with **sudo ip link add name br0 type bridge** and turn it on with **sudo ip link set br0** up to act as a switch in Linux.
- Add two dummy ports with
  - dummy1
  - dummy2
- Link the ports to bridge using **master** command.
- Verify the entry in the
  - $ **ip addr show | grep -E “br0|dummy1|dummy2”**
- The bridge forwards frames using the table
  - if it knows the destination MAC, it sends to the right port
  - if not, it sends to all ports except the sender, working like a switch.



Bridge as Linux Switch