DATA COMMUNICATION AND COMPUTER NETWORKING

LAB ASSIGNMENT SET III

NAME: DEBNIL SARKAR
ROLL NO: 002210503005

MCA 2ND YEAR 1ST SEMESTER

Assignment VI

1.***Problem Statement:***

ARP Poisoning Detection
ARP (Address Resolution Protocol) poisoning, also known as ARP spoofing, is a type of attack where an attacker sends falsified ARP reply messages over a local area network to link the attacker's MAC address with the IP address of another host (usually the default gateway). This allows the attacker to intercept, modify, or redirect network traffic intended for the target host.
In this exercise, you need to write a Python program to detect ARP poisoning attacks on the local network using scapy library. The program will continuously sniff ARP packets and compare the MAC addresses of the sender's IP with the one obtained from the system's ARP cache (ARP table). If a mismatch is found, it indicates the possibility of an ARP poisoning attack.

Design of the solution:

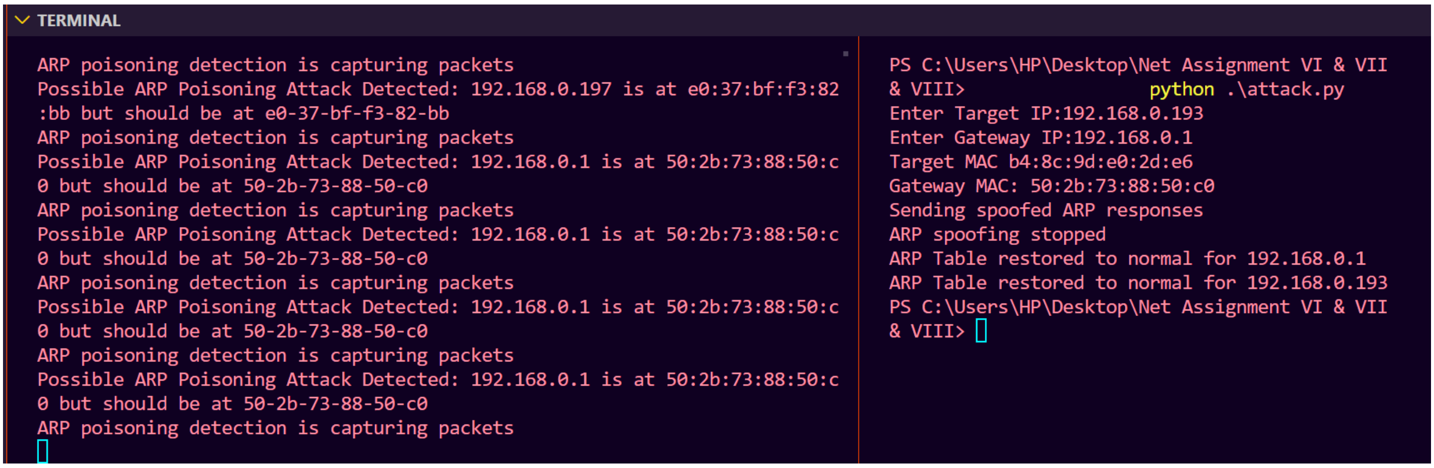
The given Python script uses Scapy to continuously sniff ARP packets and detect potential ARP poisoning attacks. It defines a function get_mac(ip) to retrieve the MAC address associated with a given IP using the ARP cache, and the sniff_arp_packets() function captures ARP packets, compares the sender's MAC address with the actual MAC address obtained from the ARP cache, and prints a warning if a potential ARP poisoning attack is detected. The script runs indefinitely, continuously monitoring the network for suspicious ARP activity.

Source Code:

```
import scapy.all as scapy
import os
import platform

# Function to get the MAC address associated with an IP from the ARP cache
def get_mac(ip):
    try:
        if platform.system() == "Windows":
            # Use the 'arp' command on Windows to fetch the MAC address
            result = os.popen(f"arp -a {ip}").read()
            mac = result.split()[10] # original mac address entry corresponding to the
given IP address
        return mac

    except Exception as e:
        print("Error: " + str(e))

# continuously sniff ARP packets and detect ARP poisoning attacks
def sniff_arp_packets():
    while True:
```

```
        print("ARP poisoning detection is capturing packets")

arp_packet = scapy.sniff(filter="arp", count=1, store=1) #sniff on ARP packets
        arp_request = arp_packet[0]
        target_ip = arp_request.psrc # get the IP address for which ARP reply was received
        sender_mac = arp_request.hwsrc # get the MAC address for which ARP reply was
received
        actual_mac = get_mac(target_ip)
      # if sender's mac is not matching with stored mac address of cache
        if sender_mac != actual_mac:
            print(f"Possible ARP Poisoning Attack Detected: {target_ip} is at {sender_mac}
but should be at {actual_mac}")

# Start ARP packet sniffing
sniff_arp_packets()
```

Sample Run:



Assignment VII

1.***Problem Statement***

Write a Python program that implements the traceroute functionality using Scapy.
• The program should take a destination IP address as input and send a series of ICMP packets with varying Time-to-Live (TTL) values to trace the route to the destination.
• Display the IP addresses of the routers along the path.
In your code, define a function traceroute () that takes the destination IP address and the maximum number of hops as inputs. Run a loop from TTL 1 to max hops, creating ICMP echo request packets with the corresponding TTL values and sending them using sr1() (send and receive in one function) from Scapy. Consider a timeout period of 1 second for the response.
• If you receive no response within the timeout, we print * to indicate no response from that hop.
• If you receive an ICMP Echo Reply, it means we have reached the destination, and we print the destination IP address.
• If you receive an ICMP Time Exceeded, it indicates that the packet has reached an intermediate router, and we print the router's IP address.
Please note that the actual number of hops may be less than max hops, depending on the network topology and firewall configurations. Also, some routers might be configured to not respond to ICMP Time Exceeded messages, which can result in incomplete traceroute information.

Design of the solution:

The provided Python script implements a basic traceroute functionality using Scapy. The traceroute function sends ICMP packets with increasing time-to-live (TTL) values to a specified destination IP address, capturing and analyzing the responses. If an ICMP Echo Reply is received, it indicates reaching the destination; if an ICMP Time Exceeded message is received, it prints the intermediate router's IP address. The script takes user input for the destination IP address and the maximum number of hops and then executes the traceroute.

Source Code

```
from scapy.all import *

def traceroute(destination, max_hops):
    for ttl in range(1, max_hops + 1):
        packet = IP(dst=destination, ttl=ttl) / ICMP()
        reply = sr1(packet, verbose=0, timeout=1)

        if reply is None:
            # No response received within the timeout

print(f"{ttl}: *")
        elif reply.haslayer(ICMP):
if reply.getlayer(ICMP).type == 0:
```

```
# ICMP Echo Reply received, we reached the destination
                print(f"{ttl}: {reply.src} (Destination Reached)")
                break
            elif reply.getlayer(ICMP).type == 11:
                # ICMP Time Exceeded, print the intermediate router's IP address
                print(f"{ttl}: {reply.src}")
        else:
            # Unhandled packet type
            print(f"{ttl}: Unknown packet type")

if __name__ == "__main__":
    destination = input("Enter the destination IP address: ")
    max_hops = int(input("Enter the maximum number of hops: "))

    traceroute(destination, max_hops)
```

Sample Run

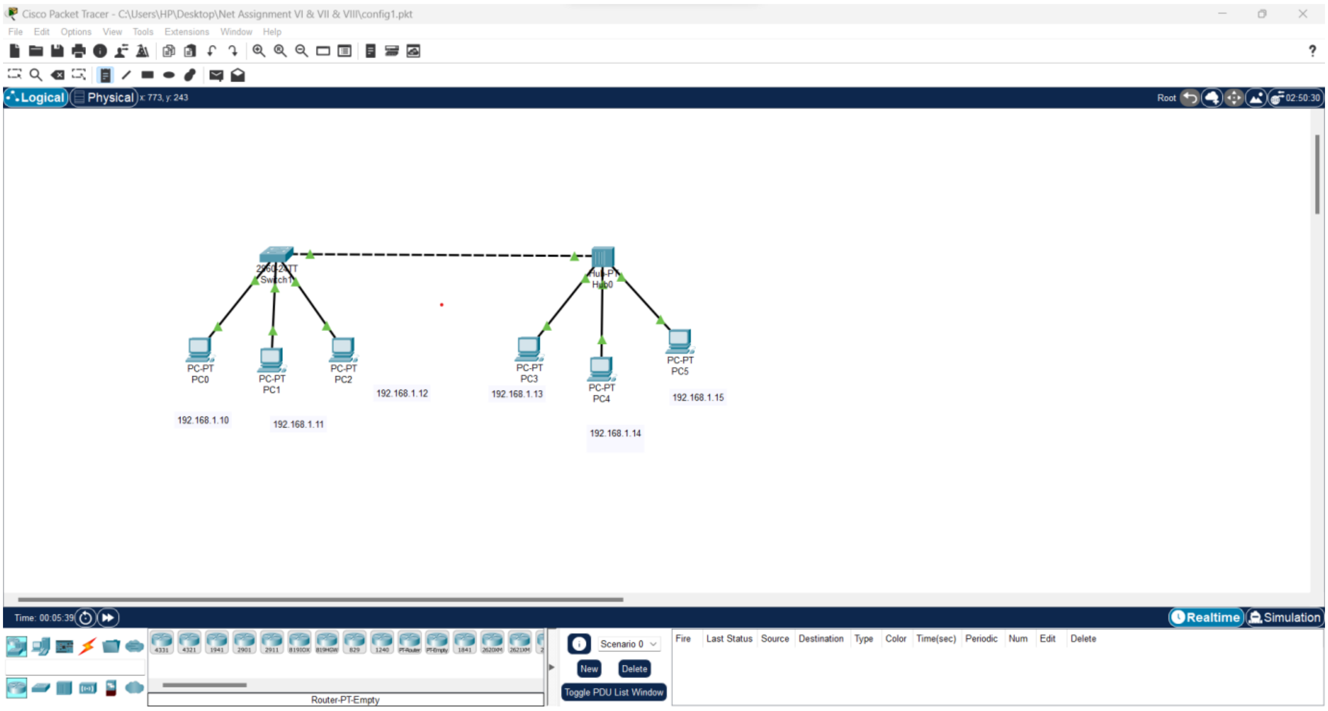
```
TERMINAL
PS C:\Users\HP\Desktop\Net Assignment VI & VII & VIII> py .\traceroute.py
Enter the destination IP address: 142.250.77.110
Enter the maximum number of hops: 20
1: 192.168.0.1
2: 10.10.84.129
3: 150.107.176.1
4: 192.168.199.170
5: 202.78.239.62
6: 142.251.227.211
7: 142.251.55.231
8: *
9: *
10: *
11: *
12: *
13: *
14: *
15: *
16: *
17: *
18: *
19: *
20: *
```

Assignment VIII

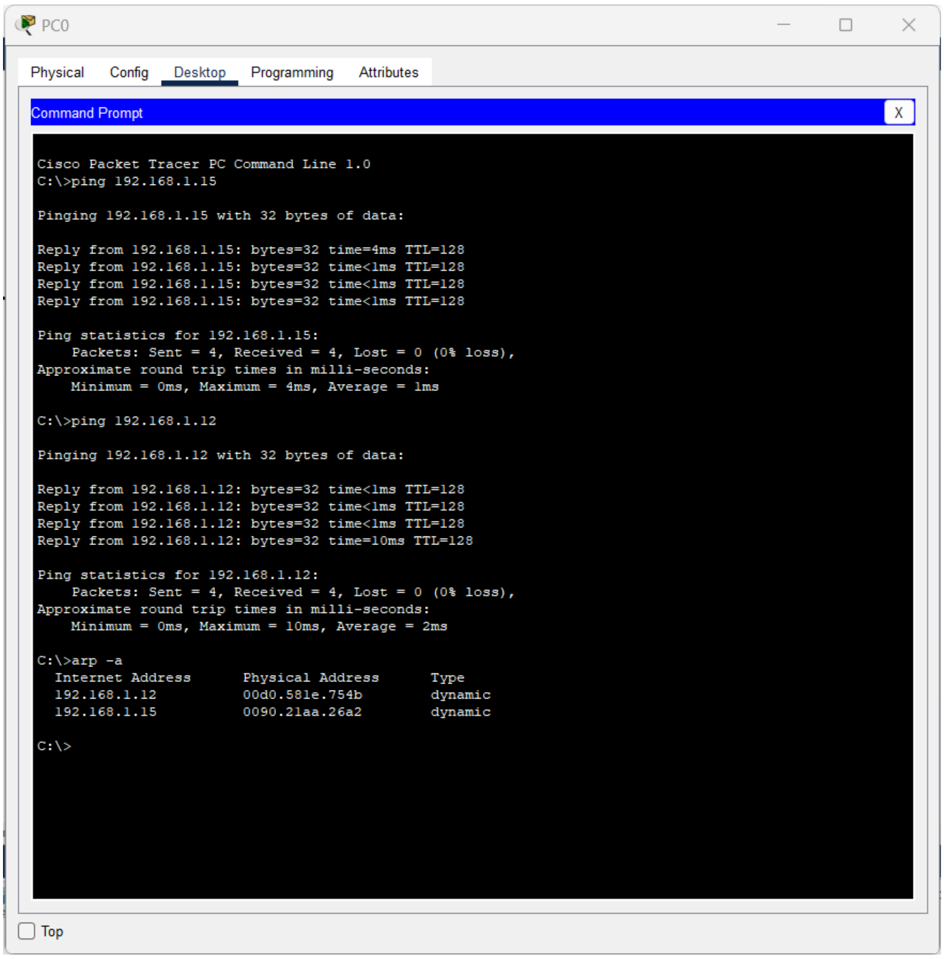
1.***Problem Statement***

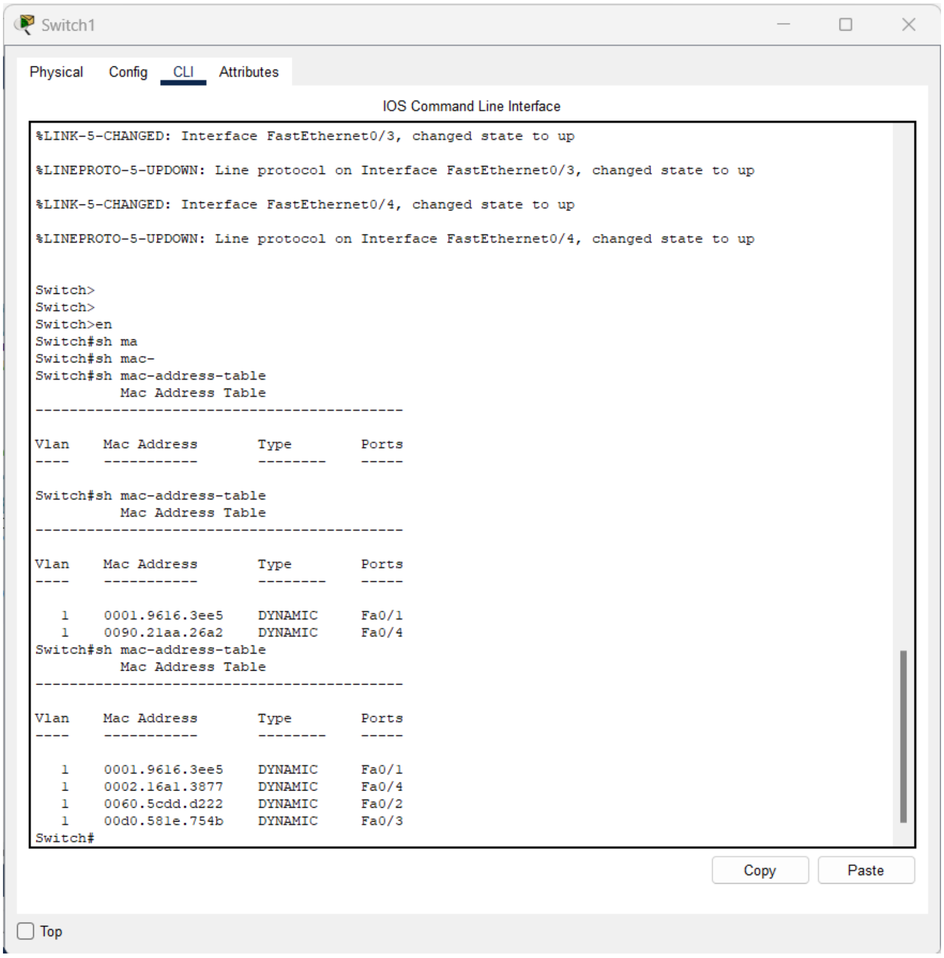
Create basic LAN topologies
(a) Connect two hosts back-to-back with a cross over cable. Assign IP addresses, and see whether they are able to ping each other.
(b) Create a LAN (named LAN-A) with 3 hosts using a hub.
(c) Create a LAN (named LAN-B) with 3 hosts using a switch. Record contents of the ARP Table of end hosts and the MAC Forwarding Table of the switch. Ping each pair of nodes. Now record the contents of the ARP Table of end hosts and the MAC Forwarding Table of the switch again.
(d) Connect LAN-A and LAN-B by connecting the hub and switch using a cross-over cable. Ping between each pair of hosts of LAN-A and LAN-B. Now record the contents of the ARP Table of end hosts and
the MAC Forwarding Table of the switch again.

Design of the network

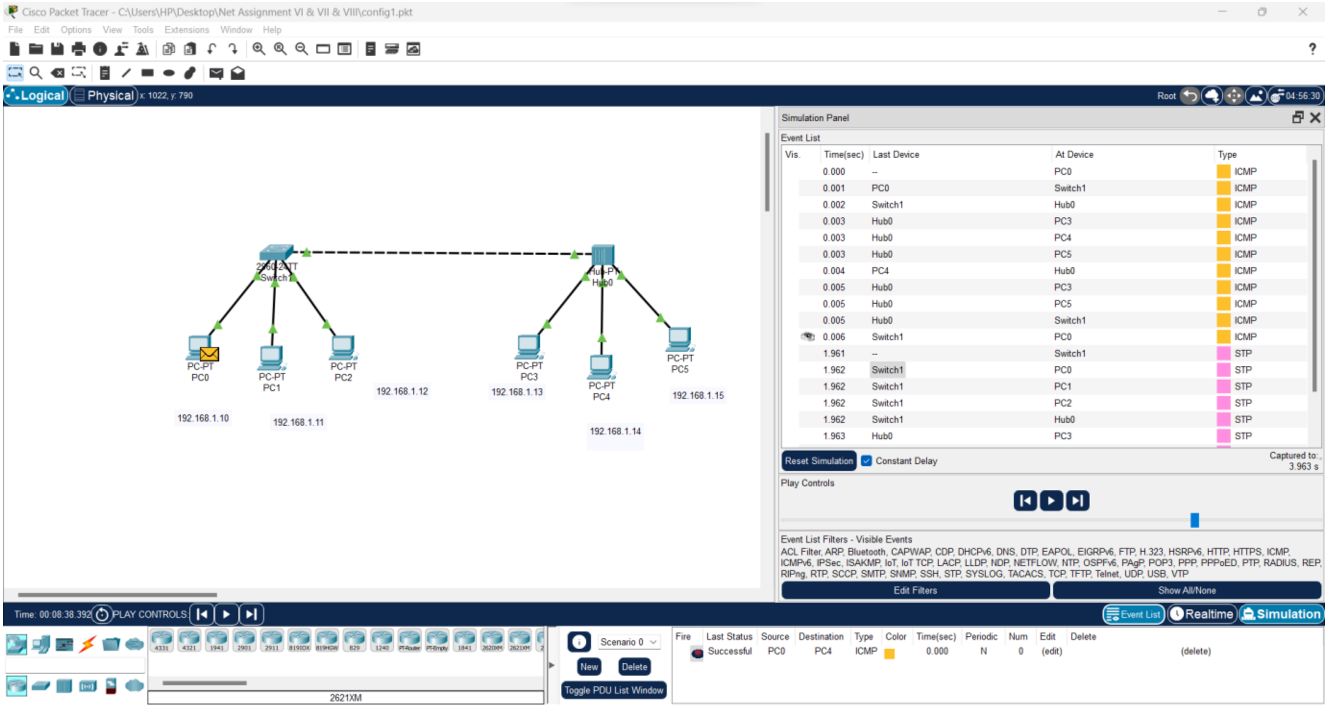


Configuration of individual hosts/switches/routers etc.





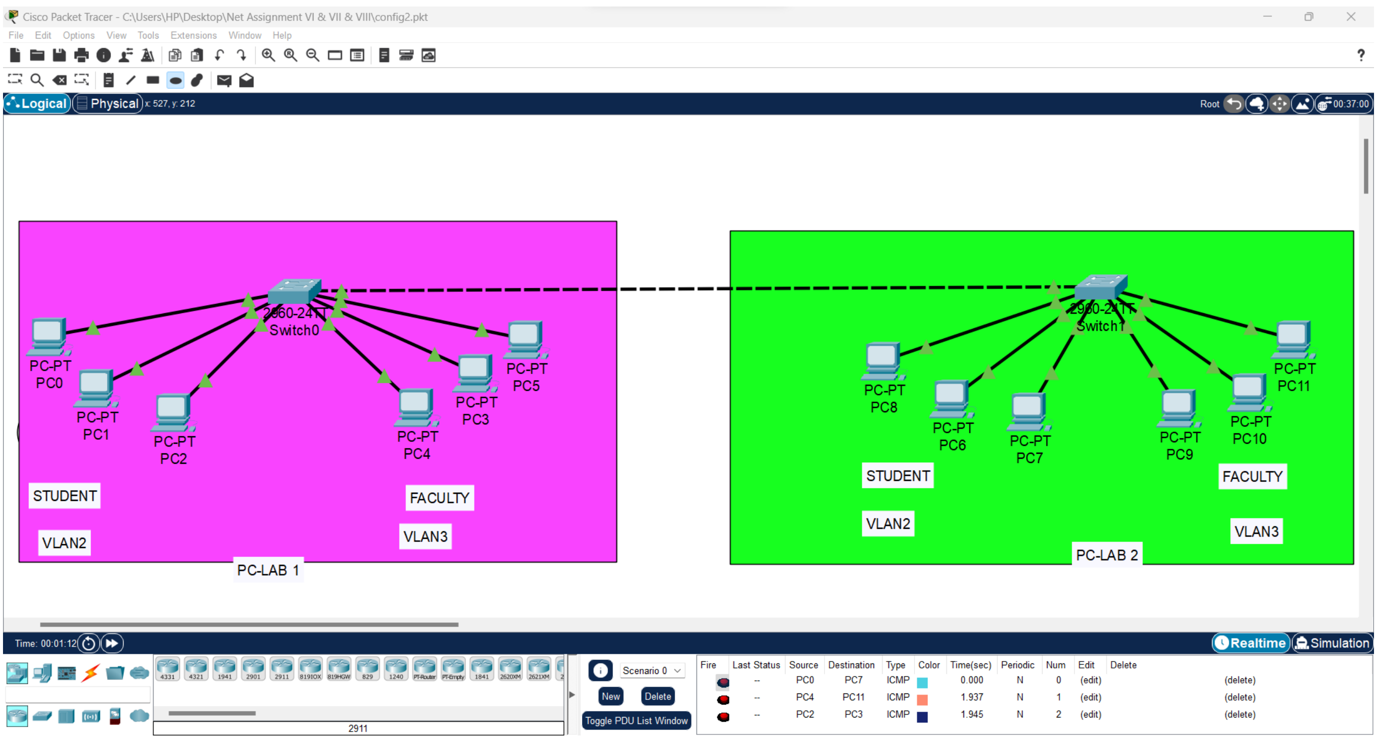
Screen shots of successful run



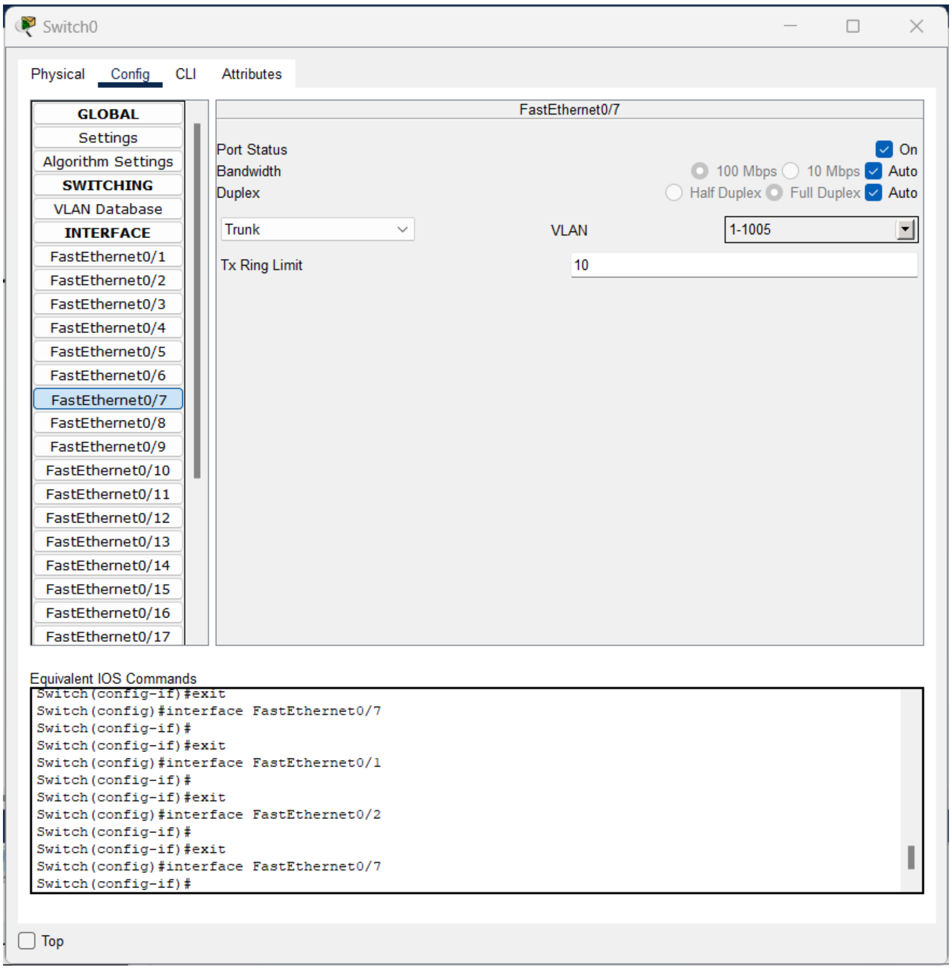
2.***Problem Statement***

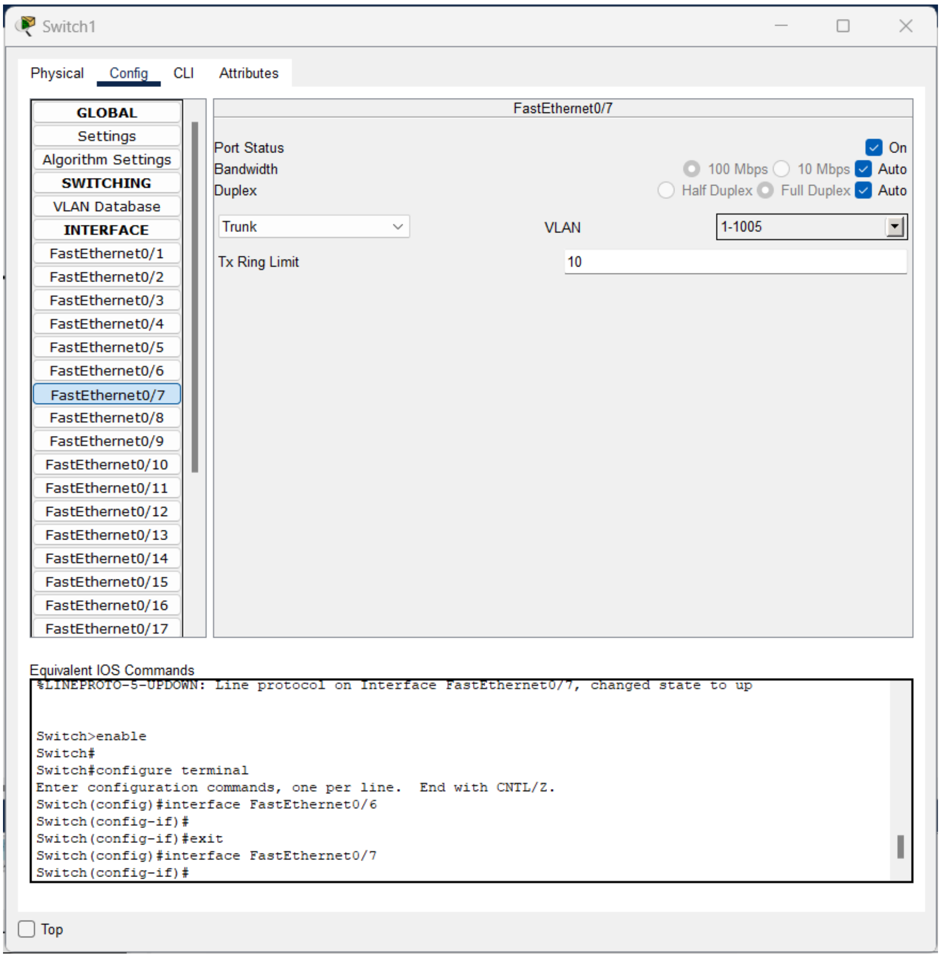
Set up VLANs and inter-VLAN routing
(a) Create a LAN (named PC-LAB1) with six hosts connected via a layer-2 switch (named PC-LAB1-Switch).
(b) Create two VLANs named "student" and "faculty". Put any three hosts into VLAN "student" and other three into VLAN "faculty"
(c) Create another LAN (named PC-LAB2) with six hosts connected via a layer-2 switch (named PC-LAB2-Switch).
(d) Repeat Experiment 2(b) for PC-LAB2.
(e) Connect the two switches via trunk ports and configure such that students/faculty in PC-LAB1 are able to communicate with students/faculty in PC-LAB2 and vice versa.

Design of the network

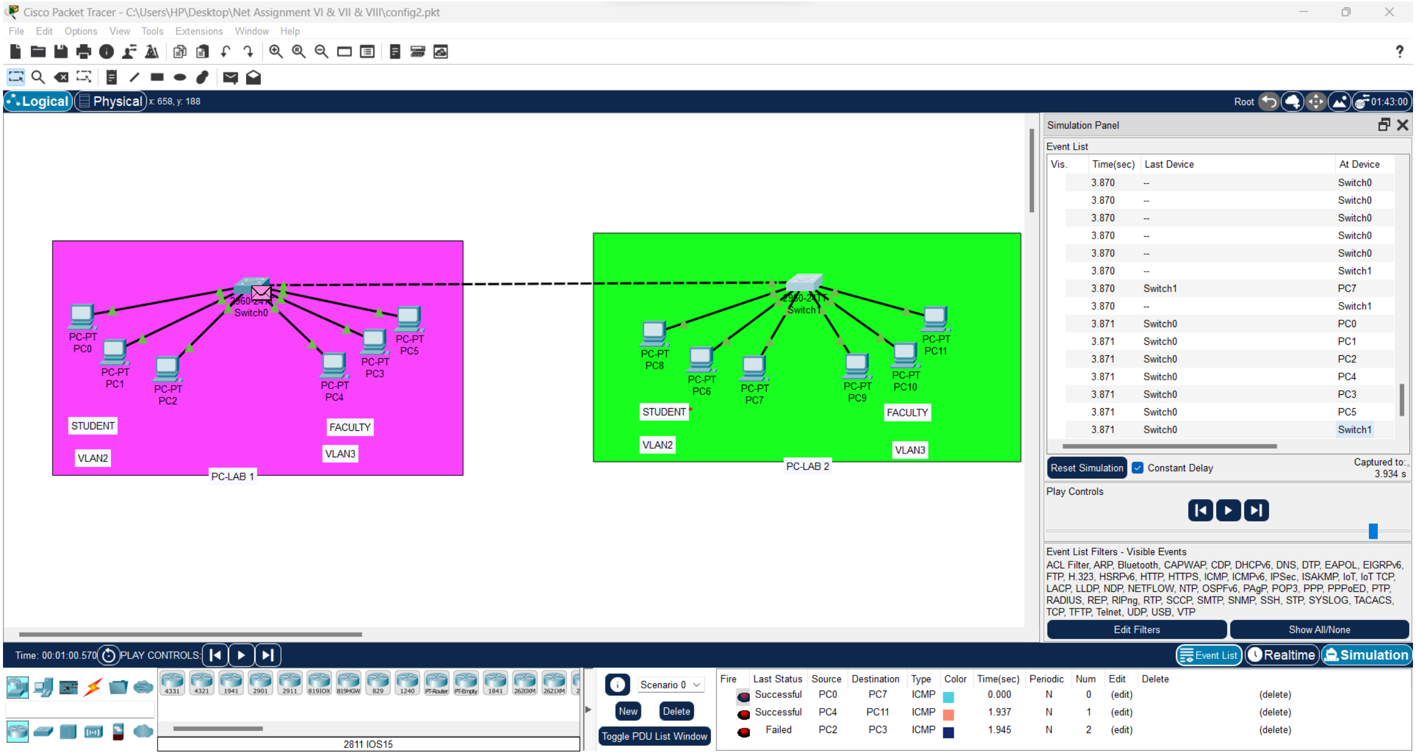


Configuration of individual hosts/switches/routers etc.





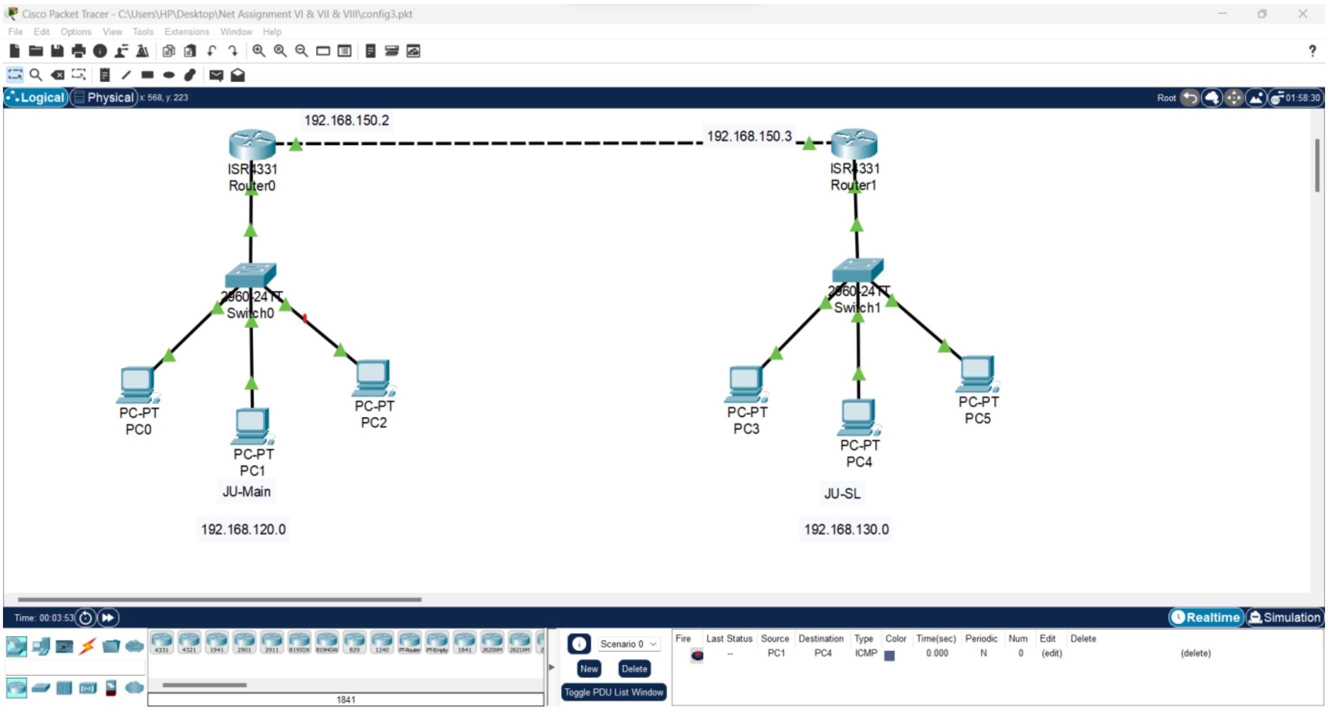
Screen shots of successful run



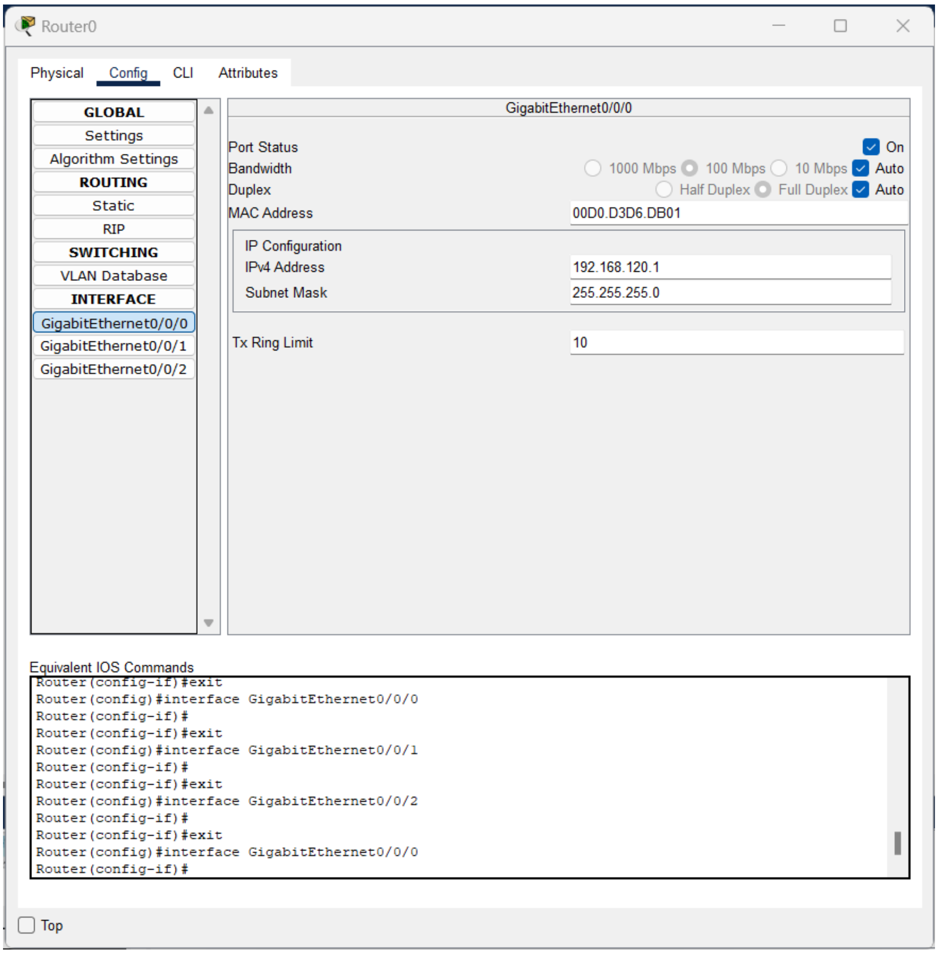
3.*Problem Statement*

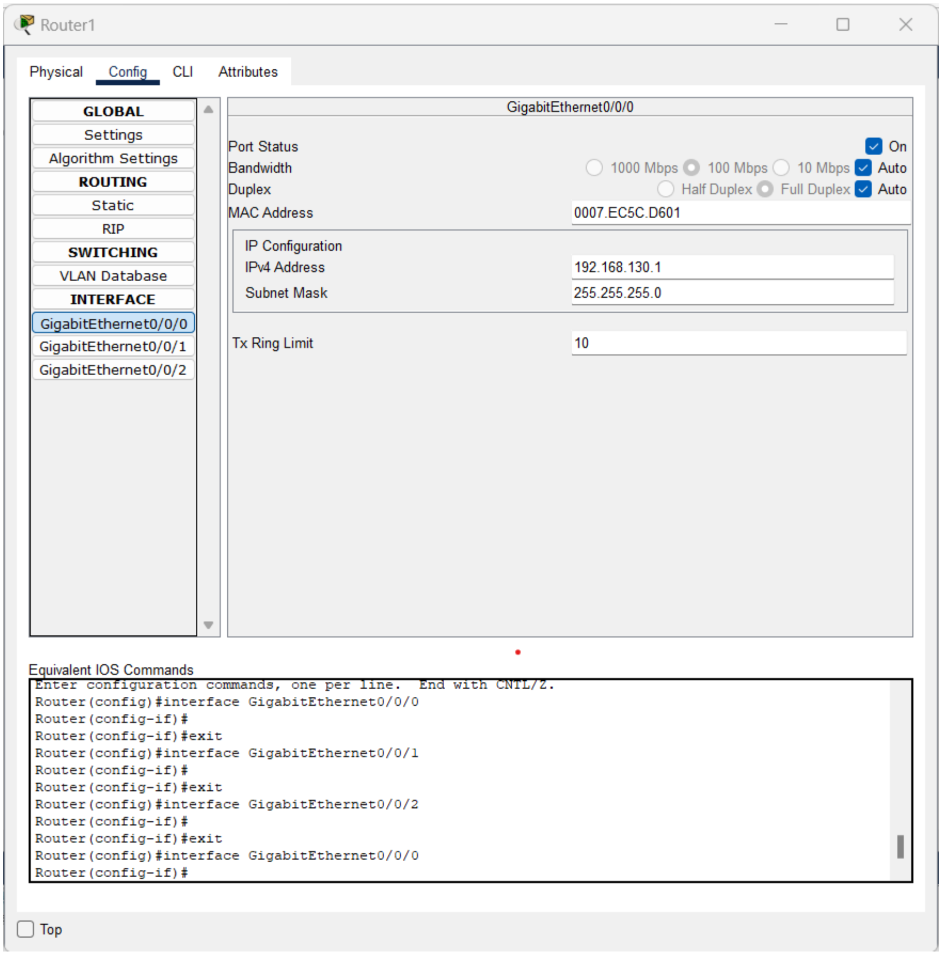
Create two LANs and connect them via a router
(a) Create a LAN (named JU-Main) with three hosts connected via a layer-2 switch. Connect the switch to a router. Assign IP addresses to all the hosts and the router interface connected to this LAN from network address 192.168.120.0/24. Configure default gateway of each host as the IP address of the interface of the router, which is connected to the LAN.
(b) Create another LAN (named JU-SL) with three hosts connected via a layer-2 switch. Connect this switch to another router. Assign IP addresses to all the hosts and the router interface connected to this LAN
from network address 192.168.130.0/24. Configure default gateway of each host as the IP address of the interface of the router which is connected to the LAN.
(c) Connect the two routers through appropriate WAN interfaces. Assign IP addresses to the WAN interfaces from network 192.168.150.0/24.
(d) Add static route in both of the routers to route packets between two LANs.

Design of the network

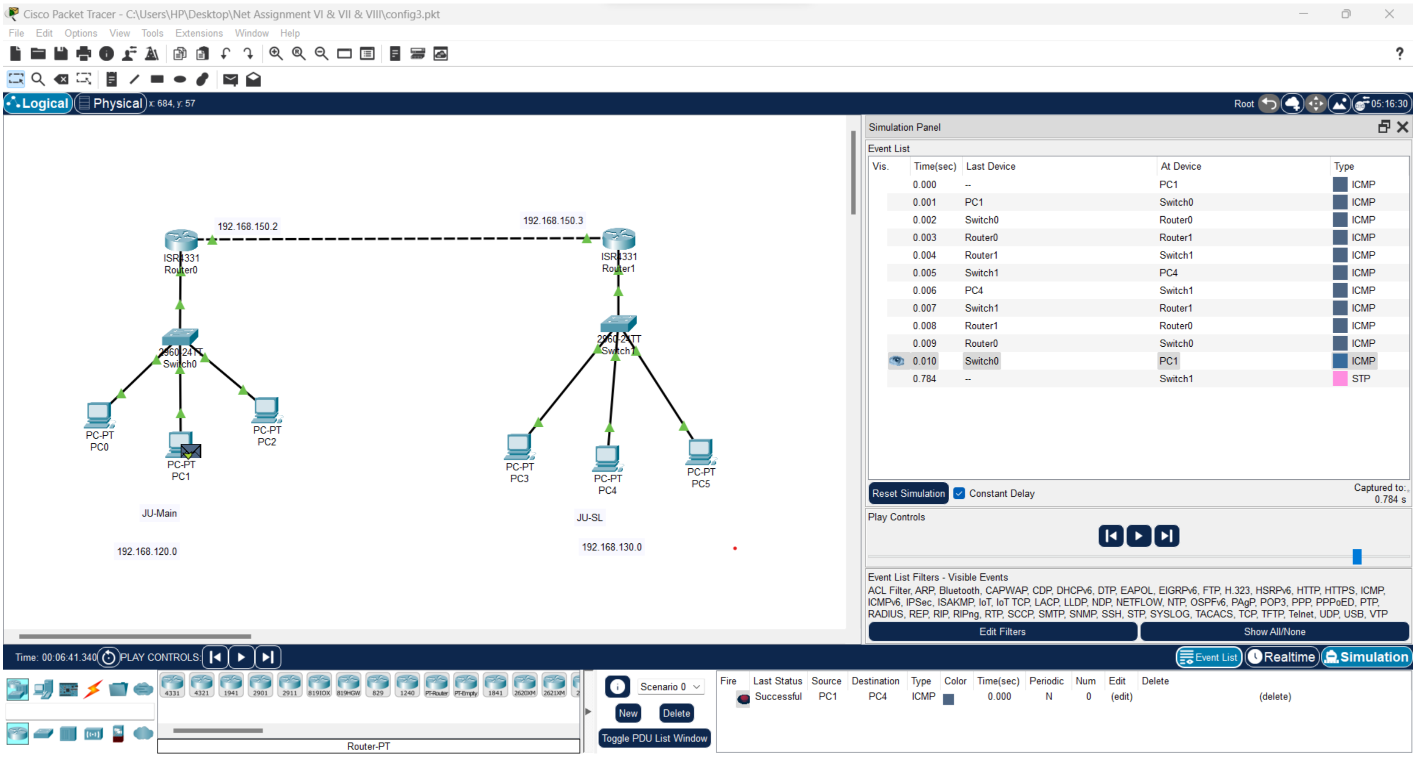


Configuration of individual hosts/switches/routers etc.



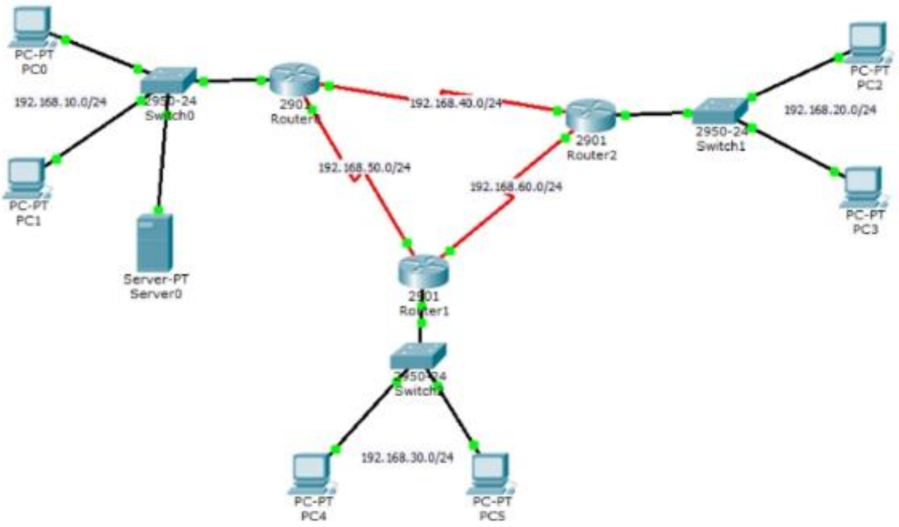


Screen shots of successful run



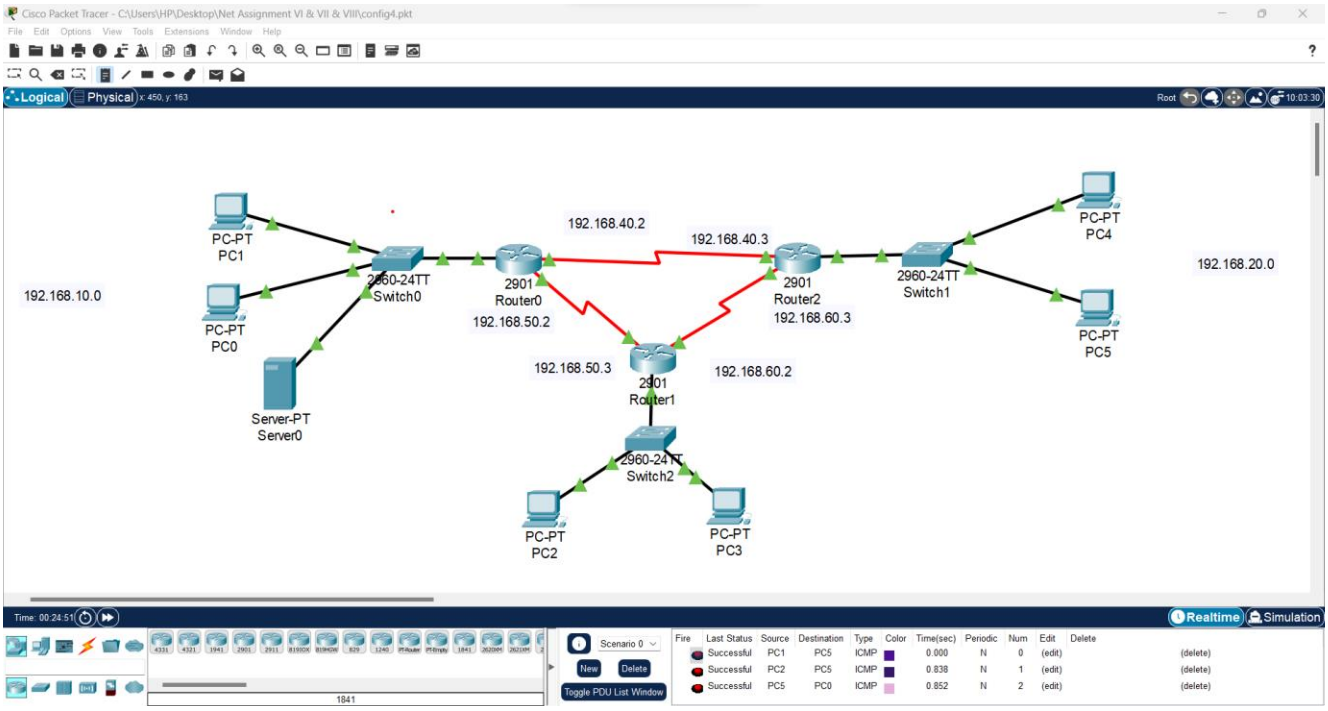
4.***Problem Statement***

Configure dynamic routing using RIP

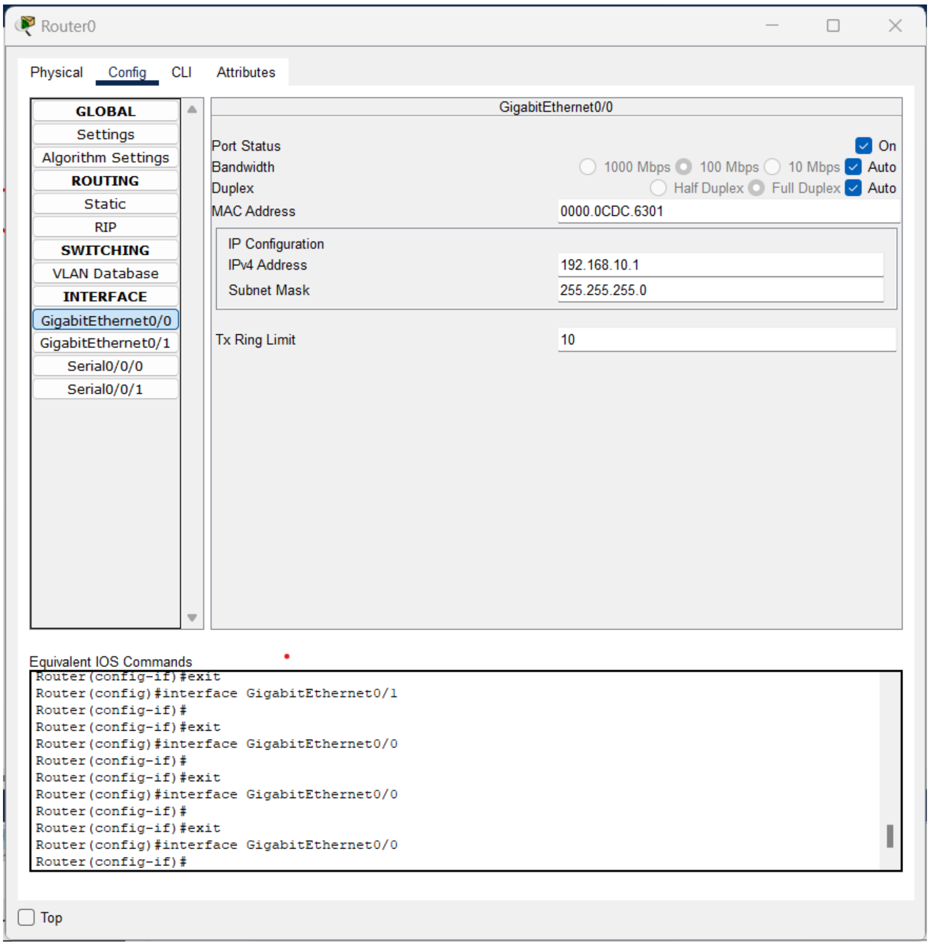


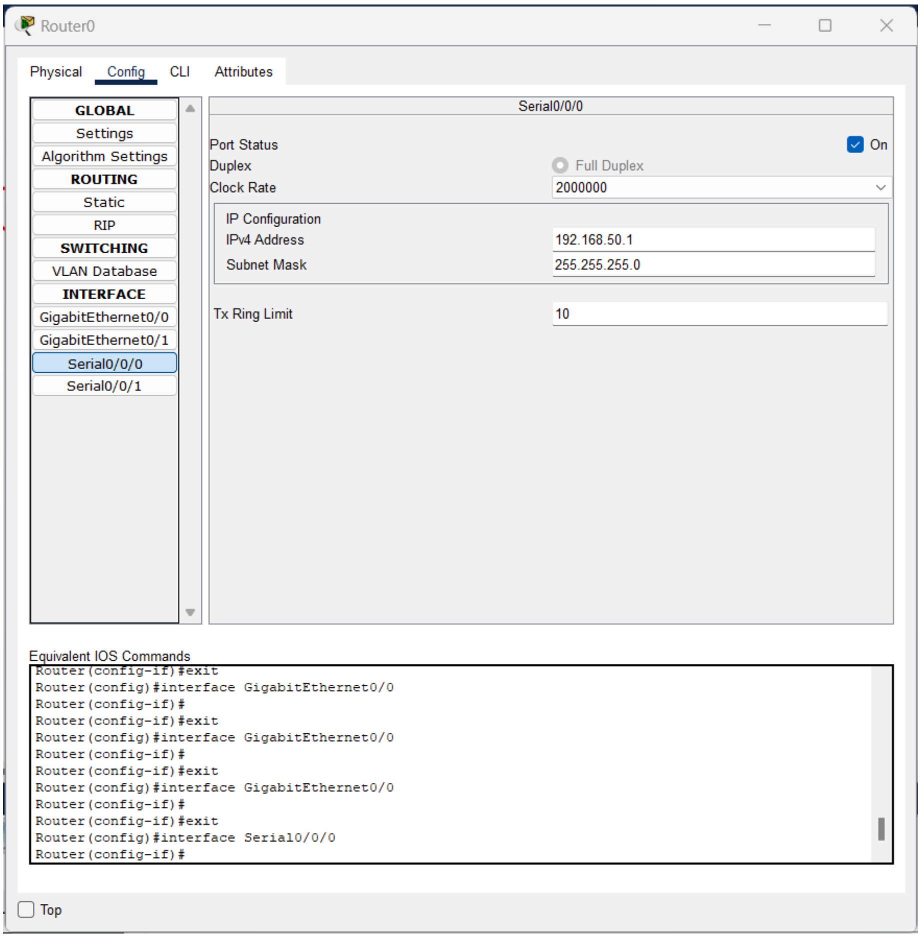
(a) Create a network topology as shown above.
(b) Configure all the routers to use dynamic routing protocol RIP.
(c) Test your configuration by pinging each pair of hosts.

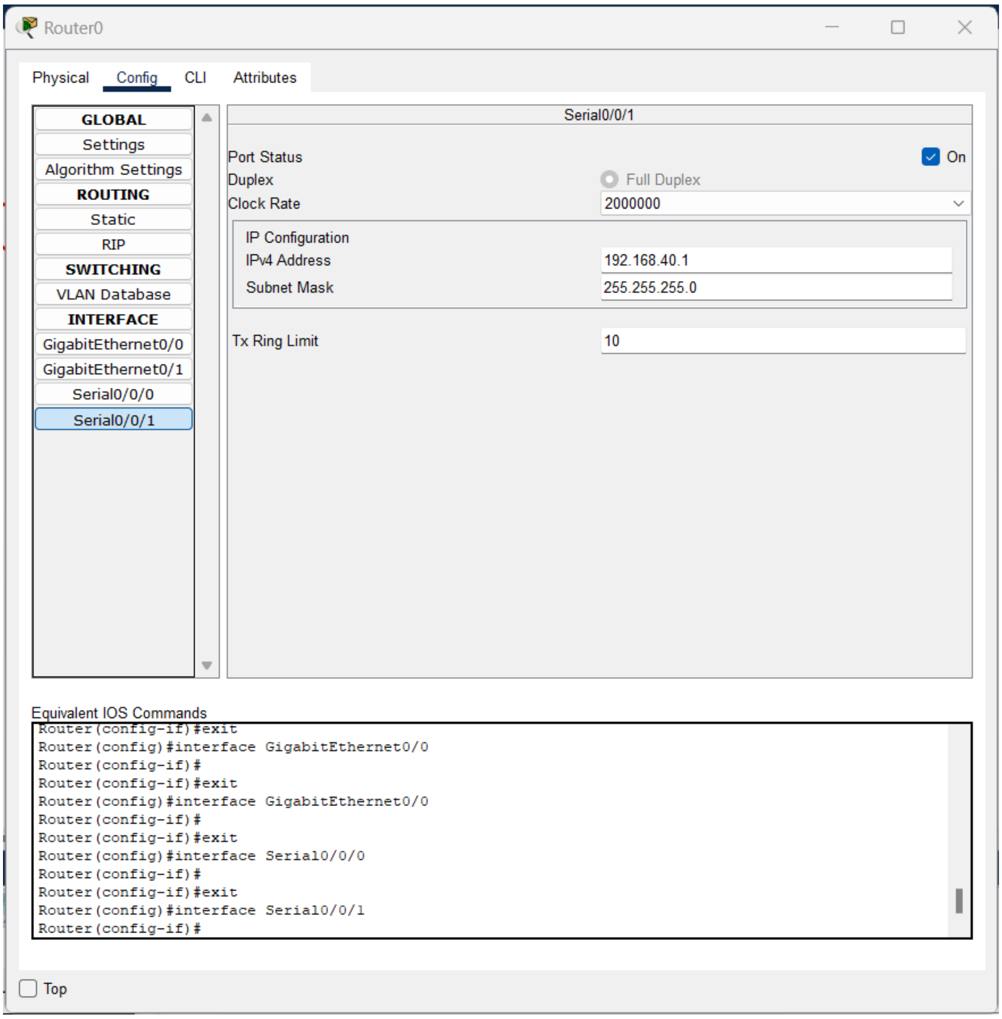
Design of the network:

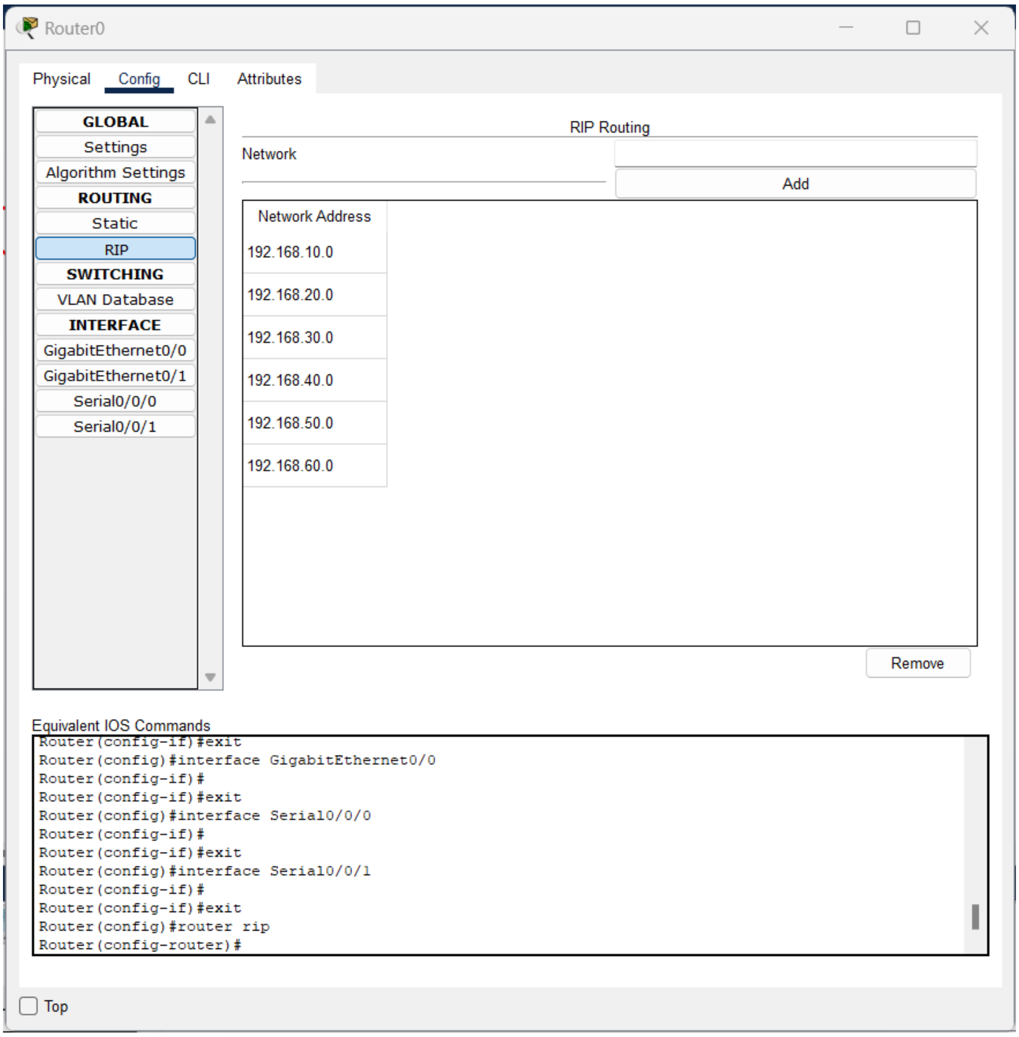


Configuration of individual hosts/switches/routers etc.






Router0
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
SWITCHING
VLAN Database
INTERFACE
GigabitEthernet0/0
GigabitEthernet0/1
Serial0/0/0
Serial0/0/1
Serial0/0/1
Port Status
On
Duplex
Full Duplex
Clock Rate
2000000
IP Configuration
IPv4 Address
192.168.40.1
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/0/1
Router(config-if)#
Top


Router0
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
SWITCHING
VLAN Database
INTERFACE
GigabitEthernet0/0
GigabitEthernet0/1
Serial0/0/0
Serial0/0/1
RIP Routing
Network
Add
Network Address
192.168.10.0
192.168.20.0
192.168.30.0
192.168.40.0
192.168.50.0
192.168.60.0
Remove
Equivalent IOS Commands
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#router rip
Router(config-router)#
Top

Screen shots of successful run