

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:

✓ TERMINAL

```

ARP poisoning detection is capturing packets
Possible ARP Poisoning Attack Detected: 192.168.0.197 is at e0:37:bf:f3:82
:bb but should be at e0-37-bf-f3-82-bb
ARP poisoning detection is capturing packets
Possible ARP Poisoning Attack Detected: 192.168.0.1 is at 50:2b:73:88:50:c
0 but should be at 50-2b-73-88-50-c0
ARP poisoning detection is capturing packets
Possible ARP Poisoning Attack Detected: 192.168.0.1 is at 50:2b:73:88:50:c
0 but should be at 50-2b-73-88-50-c0
ARP poisoning detection is capturing packets
Possible ARP Poisoning Attack Detected: 192.168.0.1 is at 50:2b:73:88:50:c
0 but should be at 50-2b-73-88-50-c0
ARP poisoning detection is capturing packets
Possible ARP Poisoning Attack Detected: 192.168.0.1 is at 50:2b:73:88:50:c
0 but should be at 50-2b-73-88-50-c0
ARP poisoning detection is capturing packets

```

```

PS C:\Users\HP\Desktop\Net Assignment VI & VII
& VIII> python .\attack.py
Enter Target IP:192.168.0.193
Enter Gateway IP:192.168.0.1
Target MAC b4:8c:9d:e0:2d:e6
Gateway MAC: 50:2b:73:88:50:c0
Sending spoofed ARP responses
ARP spoofing stopped
ARP Table restored to normal for 192.168.0.1
ARP Table restored to normal for 192.168.0.193
PS C:\Users\HP\Desktop\Net Assignment VI & VII
& VIII>

```

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: *

```

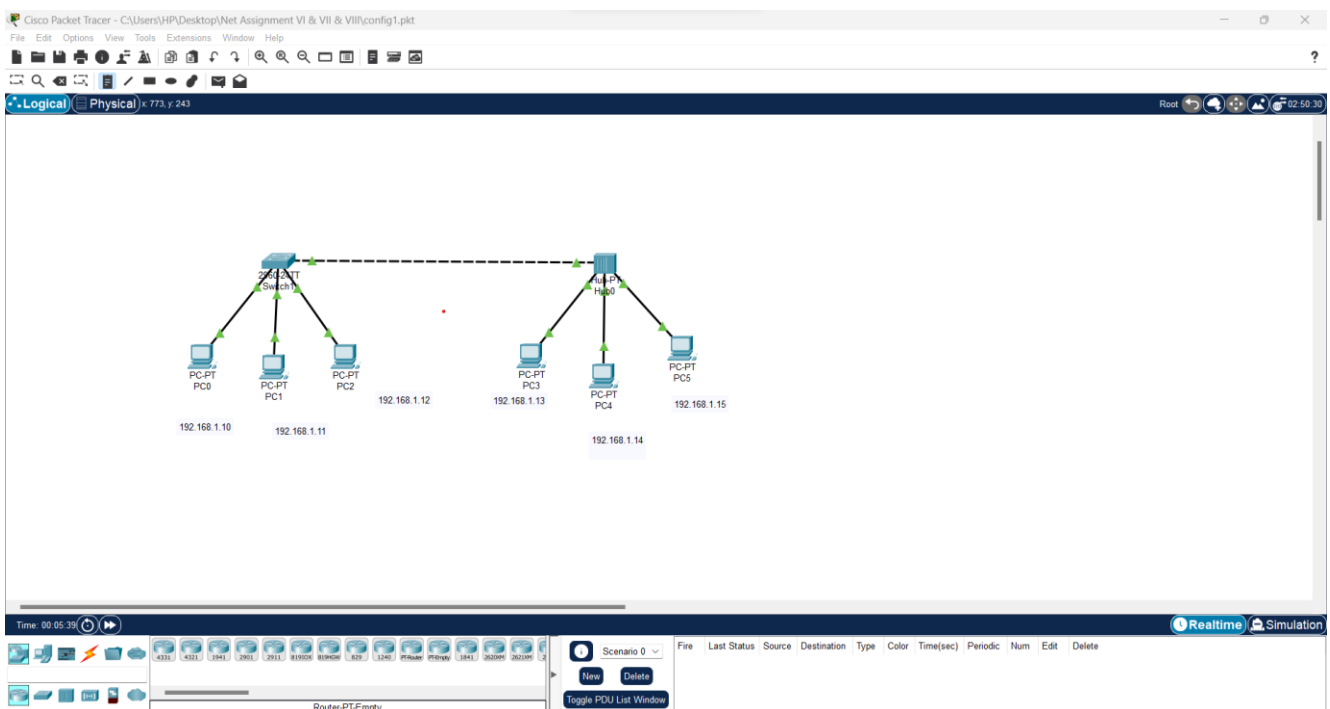
Assignment VIII

1. Problem Statement

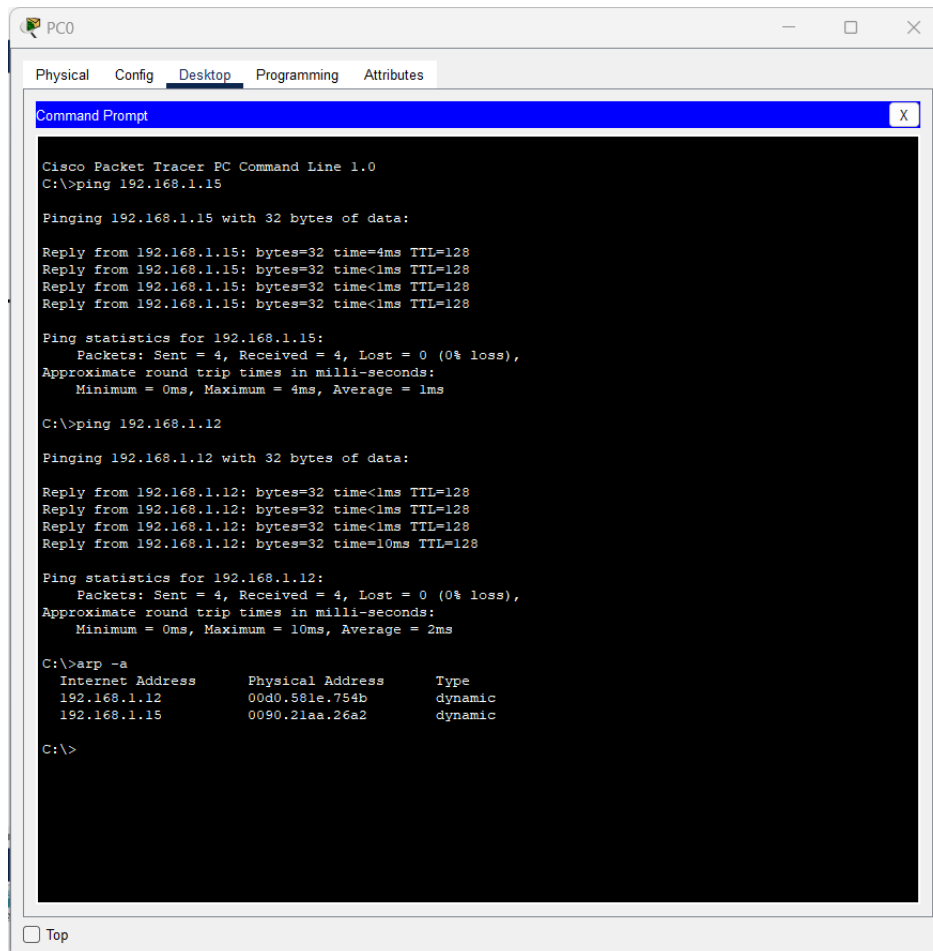
Create basic LAN topologies

- Connect two hosts back-to-back with a cross over cable. Assign IP addresses, and see whether they are able to ping each other.
- Create a LAN (named LAN-A) with 3 hosts using a hub.
- 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.
- 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.



The screenshot shows the PC0 Command Prompt window in Cisco Packet Tracer. The window has tabs for Physical, Config, Desktop, Programming, and Attributes. The Desktop tab is active, showing a black command prompt with white text. The user has entered several commands: 'ping 192.168.1.15', 'arp -a', and 'ping 192.168.1.12'. The output shows successful ping results and the ARP table.

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.15

Pinging 192.168.1.15 with 32 bytes of data:

Reply from 192.168.1.15: bytes=32 time=4ms TTL=128
Reply from 192.168.1.15: bytes=32 time<1ms TTL=128
Reply from 192.168.1.15: bytes=32 time<1ms TTL=128
Reply from 192.168.1.15: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.15:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 4ms, Average = 1ms

C:\>ping 192.168.1.12

Pinging 192.168.1.12 with 32 bytes of data:

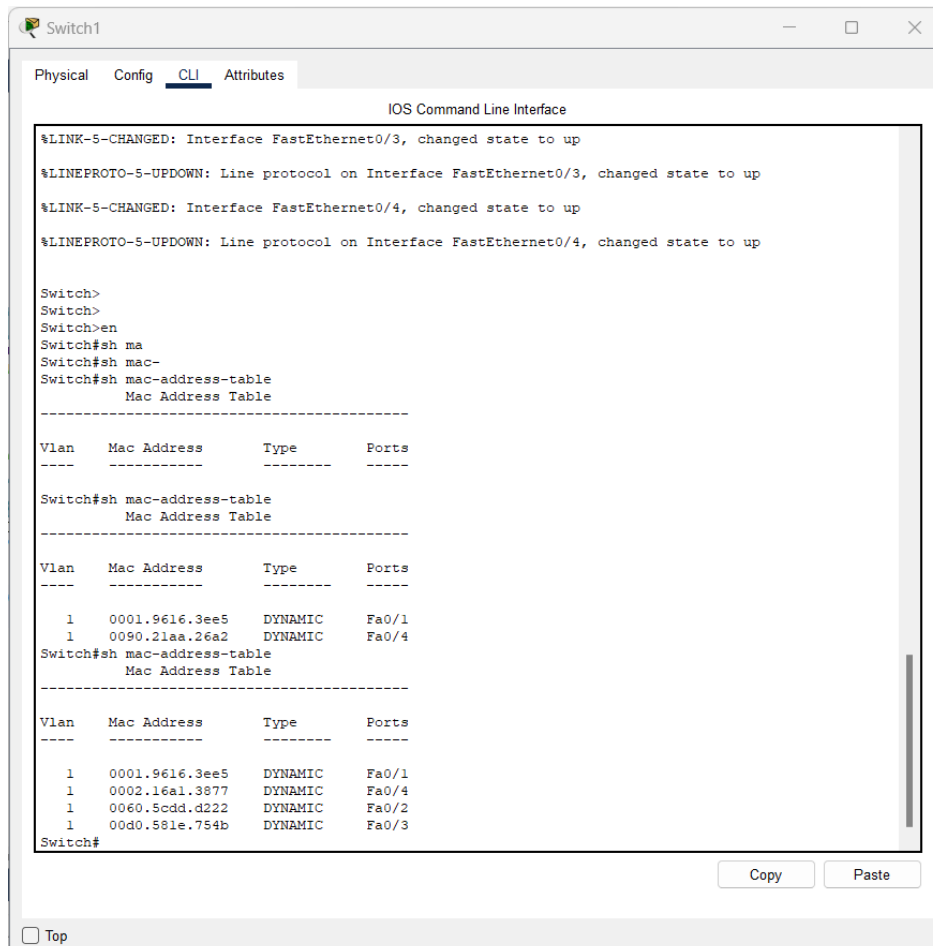
Reply from 192.168.1.12: bytes=32 time<1ms TTL=128
Reply from 192.168.1.12: bytes=32 time<1ms TTL=128
Reply from 192.168.1.12: bytes=32 time<1ms TTL=128
Reply from 192.168.1.12: bytes=32 time=10ms TTL=128

Ping statistics for 192.168.1.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 10ms, Average = 2ms

C:\>arp -a

Internet Address      Physical Address      Type
192.168.1.12          00d0.581e.754b        dynamic
192.168.1.15          0090.21aa.26a2        dynamic

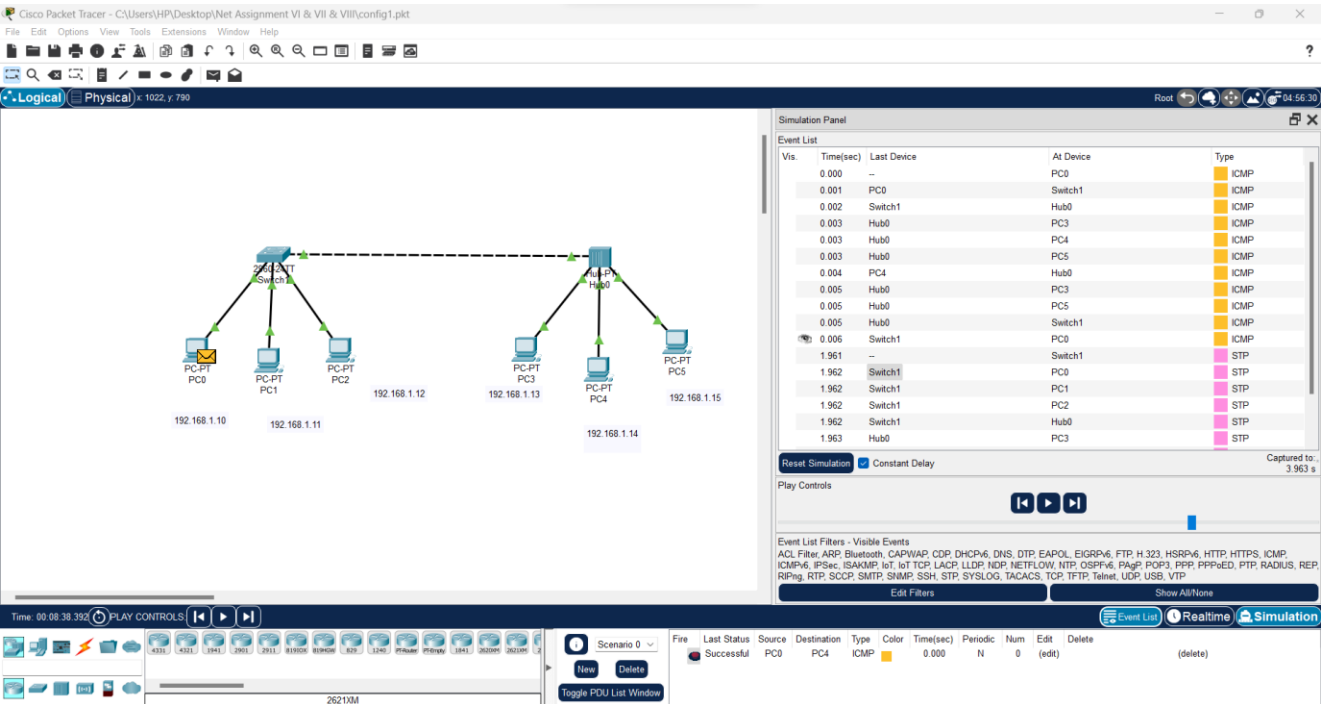
C:\>
```



The screenshot shows the Switch1 CLI window in Cisco Packet Tracer. The window has tabs for Physical, Config, CLI, and Attributes. The CLI tab is active, showing a white command prompt with black text. The user has entered several commands: 'show interface FastEthernet0/3', 'show interface FastEthernet0/4', 'show mac-address-table', and 'show mac-address-table'. The output shows the status of the interfaces and the MAC address table.

```
Switch1
Switch1>
Switch1>
Switch1>en
Switch1#sh ma
Switch1#sh mac-
Switch1#sh mac-address-table
      Mac Address Table
-----
Vlan    Mac Address      Type      Ports
----    -
Switch1#sh mac-address-table
      Mac Address Table
-----
Vlan    Mac Address      Type      Ports
----    -
1       0001.9616.3ee5    DYNAMIC   Fa0/1
1       0090.21aa.26a2    DYNAMIC   Fa0/4
Switch1#sh mac-address-table
      Mac Address Table
-----
Vlan    Mac Address      Type      Ports
----    -
1       0001.9616.3ee5    DYNAMIC   Fa0/1
1       0002.16a1.3877    DYNAMIC   Fa0/4
1       0060.5cdd.d222    DYNAMIC   Fa0/2
1       00d0.581e.754b    DYNAMIC   Fa0/3
Switch1#
```

Screen shots of successful run

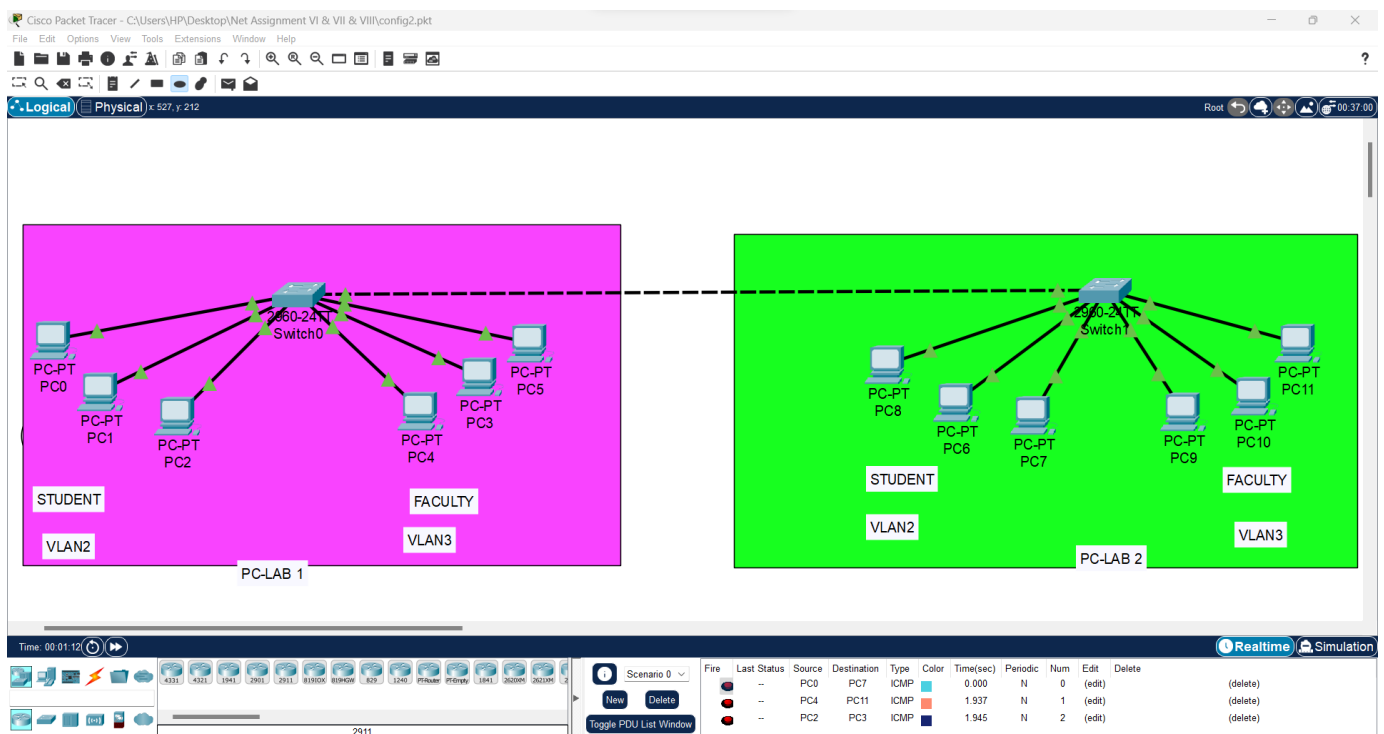


2.Problem Statement

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.

The screenshot shows the configuration window for Switch0. The left sidebar has tabs for Physical, Config, CLI, and Attributes. Under the Config tab, there are sections for GLOBAL (Settings, Algorithm Settings), SWITCHING (VLAN Database), and INTERFACE (FastEthernet0/1 through FastEthernet0/17). FastEthernet0/7 is selected. The main area shows settings for FastEthernet0/7: Port Status is On, Bandwidth is 100 Mbps, Duplex is Full Duplex, Trunk is selected, VLAN is 1-1005, and Tx Ring Limit is 10. Below the settings is a section for Equivalent IOS Commands.

```
Switch(config-if)#exit
Switch(config)#interface FastEthernet0/7
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#interface FastEthernet0/1
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#interface FastEthernet0/2
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#interface FastEthernet0/7
Switch(config-if)#
```

Top

The screenshot shows the configuration window for Switch1. The left sidebar has tabs for Physical, Config, CLI, and Attributes. Under the Config tab, there are sections for GLOBAL (Settings, Algorithm Settings), SWITCHING (VLAN Database), and INTERFACE (FastEthernet0/1 through FastEthernet0/17). FastEthernet0/7 is selected. The main area shows settings for FastEthernet0/7: Port Status is On, Bandwidth is 100 Mbps, Duplex is Full Duplex, Trunk is selected, VLAN is 1-1005, and Tx Ring Limit is 10. Below the settings is a section for Equivalent IOS Commands.

```
$LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/7, changed state to up

Switch>enable
Switch#
Switch#configure terminal
Enter configuration commands, one per line. End with CNTRL/Z.
Switch(config)#interface FastEthernet0/6
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#interface FastEthernet0/7
Switch(config-if)#
```

Top

Screen shots of successful run

Cisco Packet Tracer - C:\Users\HP\Desktop\Net Assignment VI & VII & VIII\config2.pkt

File Edit Options View Tools Extensions Window Help

Logical Physical 656 188

PC-LAB 1

PC-LAB 2

STUDENT

VLAN2

FACULTY

VLAN3

PC-LAB 1

PC-LAB 2

STUDENT

VLAN2

FACULTY

VLAN3

Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device
	3.870	--	Switch0
	3.870	--	Switch0
	3.870	--	Switch0
	3.870	--	Switch0
	3.870	--	Switch0
	3.870	--	Switch1
	3.870	Switch1	PC7
	3.870	Switch0	PC1
	3.871	Switch0	PC0
	3.871	Switch0	PC2
	3.871	Switch0	PC4
	3.871	Switch0	PC3
	3.871	Switch0	PC5
	3.871	Switch0	Switch1

Reset Simulation Constant Delay Captured to... 3.934 s

Play Controls

Event List Filters - Visible Events

ACL Filter, ARP, Bluetooth, CAPWAP, CDP, DHCPv6, DNS, DTP, EAPOL, EIGRPv6, FTP, H.323, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPsec, ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPFv6, PaGP, POP3, PPP, PPPoE, PTP, RADIUS, REP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Edit Filters Show All/None

Scenario 0

Fire Last Status Source Destination Type Color Time(sec) Periodic Num Edit Delete

Successful	PC0	PC7	ICMP		0.000	N	0	(edit)	(delete)
Successful	PC4	PC11	ICMP		1.937	N	1	(edit)	(delete)
Failed	PC2	PC3	ICMP		1.945	N	2	(edit)	(delete)

3. Problem Statement

Create two LANs and connect them via a router

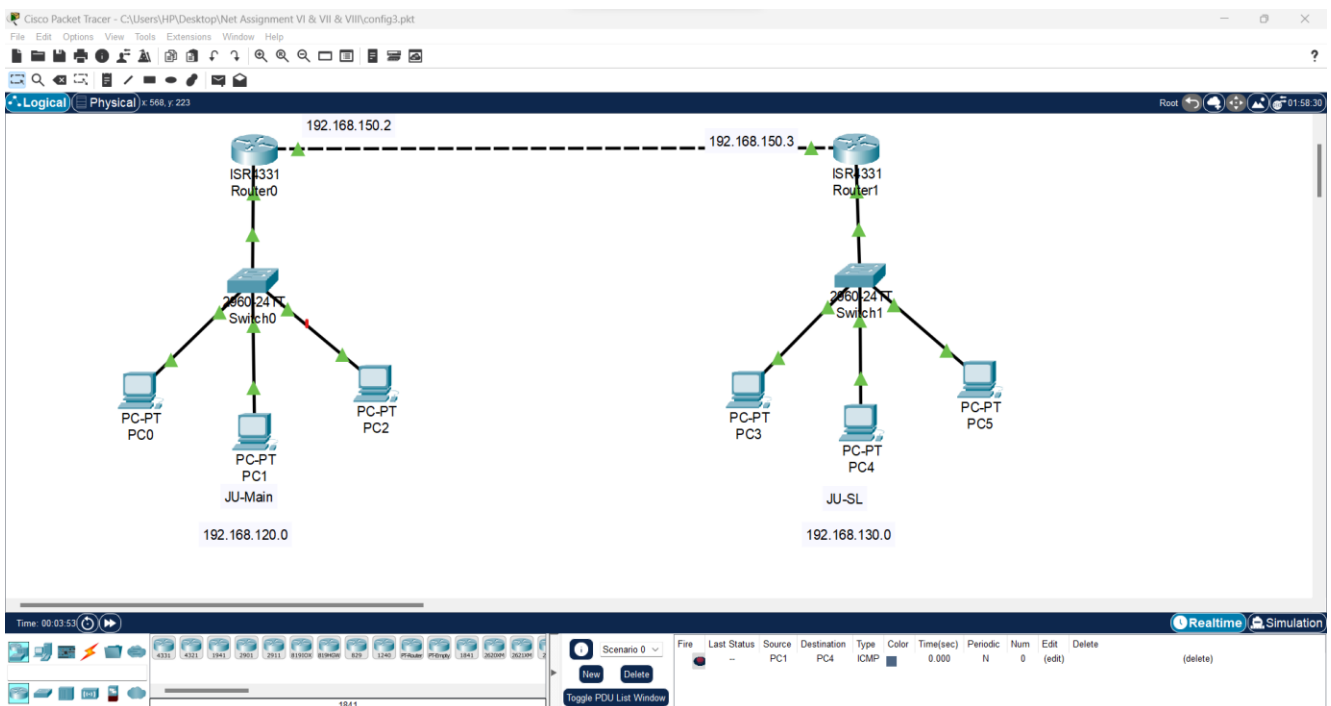
(a) Create a LAN (named JU-Main) with three hosts connected via a layer-2 switch. Connect the switch to a router. Assign IP addresses to all the hosts and the router interface connected to this LAN from network address 192.168.120.0/24. Configure default gateway of each host as the IP address of the 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.

Router0

Physical Config CLI Attributes

GLOBAL

- Settings
- Algorithm Settings

ROUTING

- Static
- RIP

SWITCHING

- VLAN Database

INTERFACE

- GigabitEthernet0/0/0
- GigabitEthernet0/0/1
- GigabitEthernet0/0/2

GigabitEthernet0/0/0

Port Status ☒ On

Bandwidth ☐ 1000 Mbps ☒ 100 Mbps ☐ 10 Mbps ☒ Auto

Duplex ☐ Half Duplex ☒ Full Duplex ☒ Auto

MAC Address 00D0.D3D6.DB01

IP Configuration

IPv4 Address 192.168.120.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0/2
Router(config-if)#
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0/0
Router(config-if)#
```

☐ Top

Router1

Physical Config CLI Attributes

GLOBAL

- Settings
- Algorithm Settings

ROUTING

- Static
- RIP

SWITCHING

- VLAN Database

INTERFACE

- GigabitEthernet0/0/0
- GigabitEthernet0/0/1
- GigabitEthernet0/0/2

GigabitEthernet0/0/0

Port Status ☒ On

Bandwidth ☐ 1000 Mbps ☒ 100 Mbps ☐ 10 Mbps ☒ Auto

Duplex ☐ Half Duplex ☒ Full Duplex ☒ Auto

MAC Address 0007.EC5C.D601

IP Configuration

IPv4 Address 192.168.130.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface GigabitEthernet0/0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0/2
Router(config-if)#
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0/0
Router(config-if)#
```

☐ Top

Screen shots of successful run

Cisco Packet Tracer - C:\Users\HP\Desktop\Net Assignment VI & VII & VIII\config3.pkt

File Edit Options View Tools Extensions Window Help

Logical Physical 684, y: 57

192.168.150.2

192.168.150.3

192.168.120.0

192.168.130.0

ISR-331 Router0

ISR-331 Router1

Switch0

Switch1

PC-PT PC0

PC-PT PC1

PC-PT PC2

PC-PT PC3

PC-PT PC4

PC-PT PC5

Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device	Type
	0.000	--	PC1	ICMP
	0.001	PC1	Switch0	ICMP
	0.002	Switch0	Router0	ICMP
	0.003	Router0	Router1	ICMP
	0.004	Router1	Switch1	ICMP
	0.005	Switch1	PC4	ICMP
	0.006	PC4	Switch1	ICMP
	0.007	Switch1	Router1	ICMP
	0.008	Router1	Router0	ICMP
	0.009	Router0	Switch0	ICMP
	0.010	Switch0	PC1	ICMP
	0.784	--	Switch1	STP

Reset Simulation

☒ Constant Delay

Captured to: 0.784 s

Play Controls

Event List Filters - Visible Events

ACL Filter, ARP, Bluetooth, CAPWAP, CDP, DHCPv6, DTP, EAPOL, EIGRPv6, FTP, H.323, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPsec, ISAKMP, IoT, IoT.TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPFv6, PAgP, POF3, PPP, PPPoE, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Edit Filters

Show All/None

Time: 00:06:41.340

PLAY CONTROLS

Scenario 0

New Delete

Toggle PDU List Window

Fire

Last Status

Source

Destination

Type

Color

Time(sec)

Periodic

Num

Edit

Delete

Successful

PC1

PC4

ICMP

0.000

N

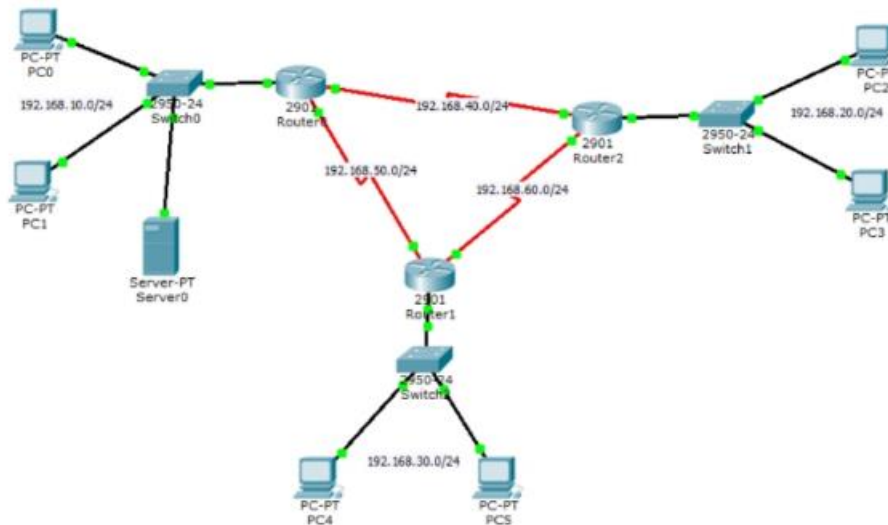
0

(edit)

(delete)

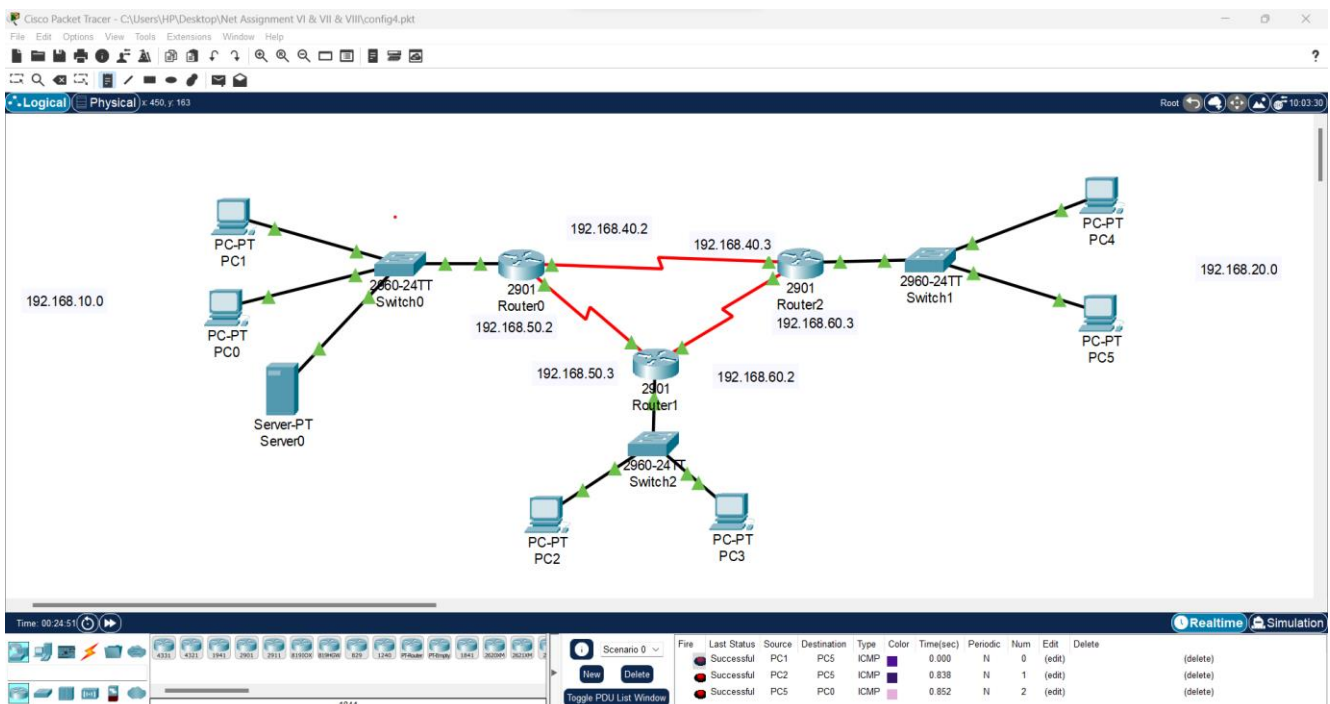
4. Problem Statement

Configure dynamic routing using RIP



- Create a network topology as shown above.
- Configure all the routers to use dynamic routing protocol RIP.
- 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

GigabitEthernet0/0

Port Status ☒ On

Bandwidth ☐ 1000 Mbps ☒ 100 Mbps ☐ 10 Mbps ☒ Auto

Duplex ☐ Half Duplex ☒ Full Duplex ☒ Auto

MAC Address 0000.0CDC.6301

IP Configuration

IPv4 Address 192.168.10.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/1
Router(config-if)#
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 GigabitEthernet0/0
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

Serial0/0/0

Port Status ☒ On

Duplex ☒ Full Duplex

Clock Rate 2000000

IP Configuration

IPv4 Address 192.168.50.1

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 GigabitEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/0/0
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**

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

