Experiment 8

Aim: The aim of this activity is to gather and analyze PDU (Protocol Data Unit) information as a packet travels within a local network and between remote networks. By studying the packet's journey, you will learn about MAC addresses, IP addresses, and how network devices such as switches, routers, and access points handle data.(Activity 9.1.3)

Objectives:

1. Gather PDU information for local network communication.
2. Gather PDU information for remote network communication.
3. Analyze the gathered data to answer reflection questions.

Instructions

Part 1: Gather PDU Information for Local Network Communication

1. Open Cisco Packet Tracer and switch to Simulation Mode.
2. Click on device 172.16.31.5 and open the Command Prompt.
3. Enter the following command:
   ping 172.16.31.2
4. A PDU will appear next to 172.16.31.5. Click on the PDU and record the following information from the OSI Model and Outbound PDU Layer tabs:
   - Destination MAC Address: 000C:85CC:1DA7
   - Source MAC Address: 00D0:D311
   - Source IP Address: 172.16.31.5
   - Destination IP Address: 172.16.31.2
   - At Device: 172.16.31.5
5. Click Capture/Forward to move the PDU to the next device. Gather the same information from the next device. Repeat this process until the PDU reaches its destination.
6. Record the gathered data in a spreadsheet using a table format similar to the one shown below:

At Device	Dest. MAC	Src. MAC	Src. IPv4	Dest. IPv4
172.16.31.5	000C:85CC:1DA7	00D0:D311	172.16.31.5	172.16.31.2
Switch1	000C:85CC:1DA7	00D0:D311	N/A	N/A
Hub	N/A	N/A	N/A	N/A
172.16.31.2	00D0:D311	000C:85CC:1DA7	172.16.31.2	172.16.31.5

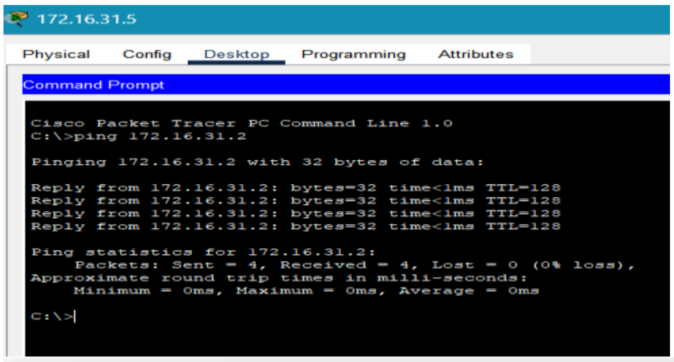
7. Repeat the process for these tests:
   - Ping from 172.16.31.3 to 172.16.31.2.
   - Ping from 172.16.31.5 to 172.16.31.4.

Part 2: Gather PDU Information for Remote Network Communication

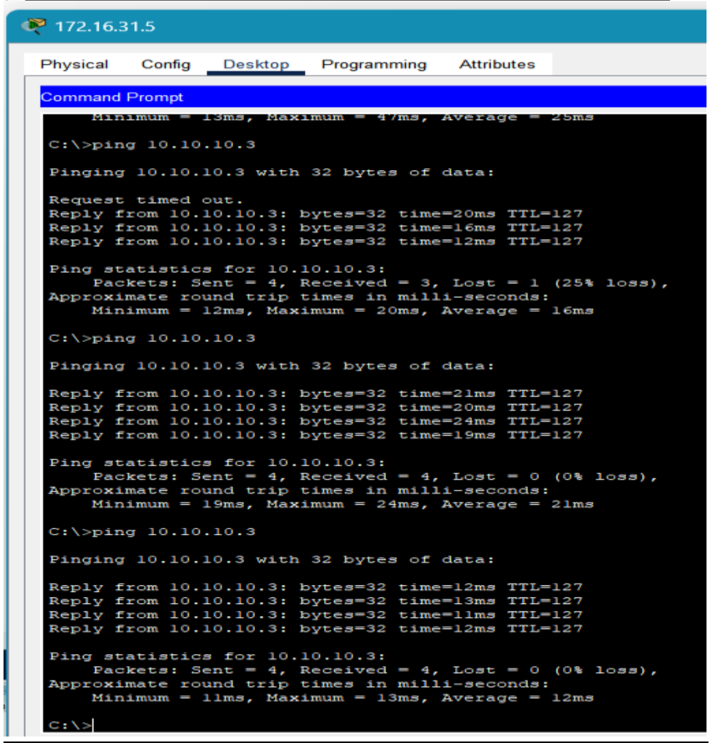
1. Switch to Simulation Mode and click on device 172.16.31.5. Open the Command Prompt.
2. Enter the following command: ping 10.10.10.2
3. A PDU will appear next to 172.16.31.5. Click on the PDU and record the following information:
   - Destination MAC Address: 00D0:BA8E:741A
   - Source MAC Address: 00D0:D311
   - Source IP Address: 172.16.31.5
   - Destination IP Address: 10.10.10.2
   - At Device: 172.16.31.5
4. Click Capture/Forward to move the PDU to the next device. Repeat this process until the PDU reaches the destination and record the gathered information in a table.

At Device	Dest. MAC	Src. MAC	Src. IPv4	Dest. IPv4
172.16.31.5	00D0:BA8E:741A	00D0:D311	172.16.31.5	10.10.10.2
Switch1	00D0:BA8E:741A	00D0:D311	N/A	N/A
Router	0060:2F84:4AB6	00D0:588C:2401	172.16.31.5	10.10.10.2
10.10.10.2	00D0:588C:2401	0060:2F84:4AB6	10.10.10.2	172.16.31.5

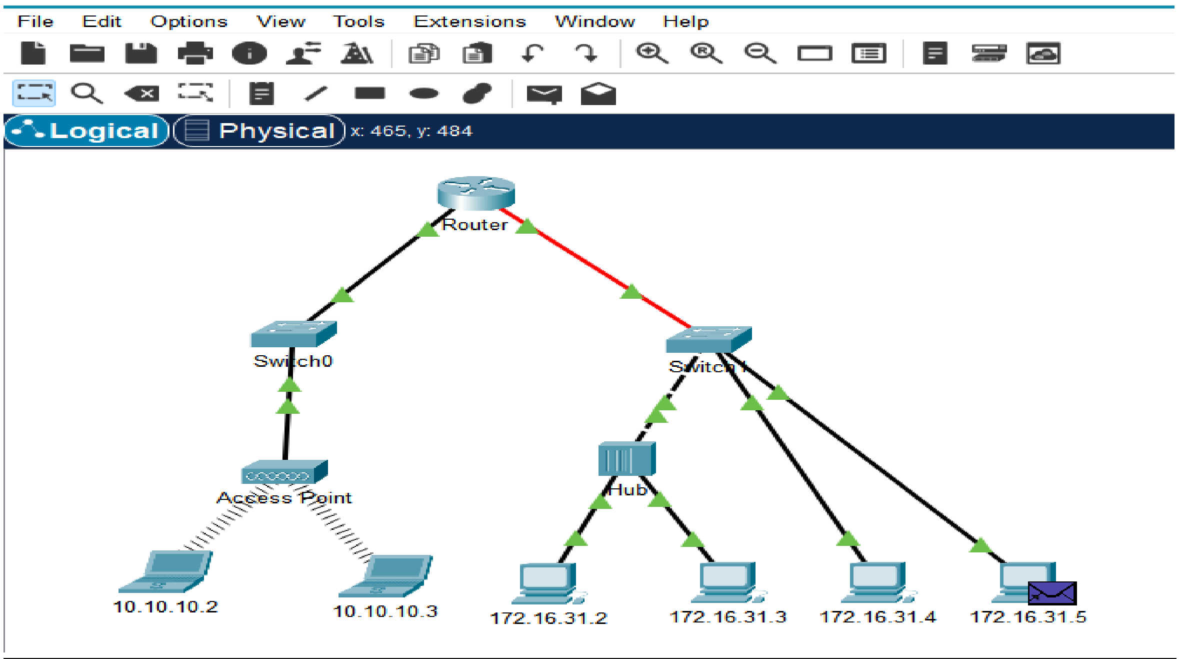
Result:



```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 172.16.31.2

Pinging 172.16.31.2 with 32 bytes of data:

Reply from 172.16.31.2: bytes=32 time<1ms TTL=128
Reply from 172.16.31.2: bytes=32 time<1ms TTL=128
Reply from 172.16.31.2: bytes=32 time<1ms TTL=128
Reply from 172.16.31.2: bytes=32 time<1ms TTL=128

Ping statistics for 172.16.31.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```



```
    Minimum = 13ms, Maximum = 47ms, Average = 25ms

C:\>ping 10.10.10.3

Pinging 10.10.10.3 with 32 bytes of data:

Request timed out.
Reply from 10.10.10.3: bytes=32 time=20ms TTL=127
Reply from 10.10.10.3: bytes=32 time=16ms TTL=127
Reply from 10.10.10.3: bytes=32 time=12ms TTL=127

Ping statistics for 10.10.10.3:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 12ms, Maximum = 20ms, Average = 16ms

C:\>ping 10.10.10.3

Pinging 10.10.10.3 with 32 bytes of data:

Reply from 10.10.10.3: bytes=32 time=21ms TTL=127
Reply from 10.10.10.3: bytes=32 time=20ms TTL=127
Reply from 10.10.10.3: bytes=32 time=24ms TTL=127
Reply from 10.10.10.3: bytes=32 time=19ms TTL=127

Ping statistics for 10.10.10.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 19ms, Maximum = 24ms, Average = 21ms

C:\>ping 10.10.10.3

Pinging 10.10.10.3 with 32 bytes of data:

Reply from 10.10.10.3: bytes=32 time=12ms TTL=127
Reply from 10.10.10.3: bytes=32 time=13ms TTL=127
Reply from 10.10.10.3: bytes=32 time=11ms TTL=127
Reply from 10.10.10.3: bytes=32 time=12ms TTL=127

Ping statistics for 10.10.10.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 11ms, Maximum = 13ms, Average = 12ms

C:\>
```



Aryan Nair A2305222105 5-CSE-2X

Experiment 9

Aim: To examine ARP Table and process in network simulation Packet Tracer(Activity 9.2.9)

Instructions

Part 1: Examine an ARP Request

Step 1: Generate ARP requests by pinging 172.16.31.3 from 172.16.31.2.

- Open a command prompt:
  - a. Click 172.16.31.2 and open the Command Prompt.
  - b. Enter the arp -d command to clear the ARP table.
- Close the command prompt:
  - c. Enter Simulation mode and enter the command ping 172.16.31.3. Two PDUs will be generated. The ping command cannot complete the ICMP packet without knowing the MAC address of the destination. So the computer sends an ARP broadcast frame to find the MAC address of the destination.
  - d. Click Capture/Forward once. The ARP PDU moves to Switch1 while the ICMP PDU disappears, waiting for the ARP reply. Open the PDU and record the destination MAC address.

Answer: The MAC address will be 0060.7036.2849 since the ping is targeting 172.16.31.3.

e. Click Capture/Forward to move the PDU to the next device.

- Answer: Switch1 made 3 copies of the PDU.
- Answer: The IP address of the device that accepted the PDU is 172.16.31.3.

f. Open the PDU and examine Layer 2.

- Answer: The source MAC address changes to that of 172.16.31.2 and the destination MAC address changes to that of 172.16.31.3 after ARP resolution.

g. Click Capture/Forward until the PDU returns to 172.16.31.2.

- Answer: The switch made 1 copy of the PDU during the ARP reply.

Step 2: Examine the ARP table.

a. Note that the ICMP packet reappears. Open the PDU and examine the MAC addresses.

- Answer: Yes, the MAC addresses of the source and destination align with their respective IP addresses.

b. Switch back to Realtime and the ping completes.

c. Click 172.16.31.2 and enter the arp –a command.

Answer: The IP address corresponding to the MAC address will be 172.16.31.3.

Answer: An end device issues an ARP request when it needs to communicate with another device whose MAC address it doesn't know but knows the IP address.

Part 2: Examine a Switch MAC Address Table

Step 1: Generate additional traffic to populate the switch MAC address table.

- Open a command prompt:
  - a. From 172.16.31.2, enter the command ping 172.16.31.4.
  - b. Click 10.10.10.2 and open the Command Prompt.
  - c. Enter the command ping 10.10.10.3.

Answer: The number of replies sent and received depends on the network setup. Typically, 4 packets are sent and received.

Close the command prompt.

Step 2: Examine the MAC address table on the switches.

a. Click Switch1 and then the CLI tab. Enter the command show mac-address-table.

Answer: Yes, the entries correspond to those in the table above.

b. Click Switch0, then the CLI tab. Enter the command show mac-address-table.

Answer: Yes, the entries correspond to those in the table above.

Answer: Two MAC addresses are associated with one port because both devices connected to the port may have communicated through it at different times.

Part 3: Examine the ARP Process in Remote Communications

Step 1: Generate traffic to produce ARP traffic.

- Open a command prompt:
  - a. Click 172.16.31.2 and open the Command Prompt.
  - b. Enter the command ping 10.10.10.1.
  - c. Type arp –a.

Answer: The new IP address in the ARP table will be 10.10.10.1.

d. Enter arp -d to clear the ARP table and switch to Simulation mode.

e. Repeat the ping to 10.10.10.1.

Answer: Typically, 2 PDUs will appear: one for the ARP request and one for the ARP reply.

Close the command prompt.

f. Click Capture/Forward. Click the PDU that is now at Switch1.

Answer: The target destination IP address of the ARP request is 10.10.10.1.

g. The destination IP address is not 10.10.10.1.

Answer: This is because the ARP request must first resolve the MAC address for the next hop router, not the final destination.

Step 2: Examine the ARP table on Router1.

a. Switch to Realtime mode. Click Router1 and then the CLI tab.

b. Enter privileged EXEC mode and then the command show mac-address-table.

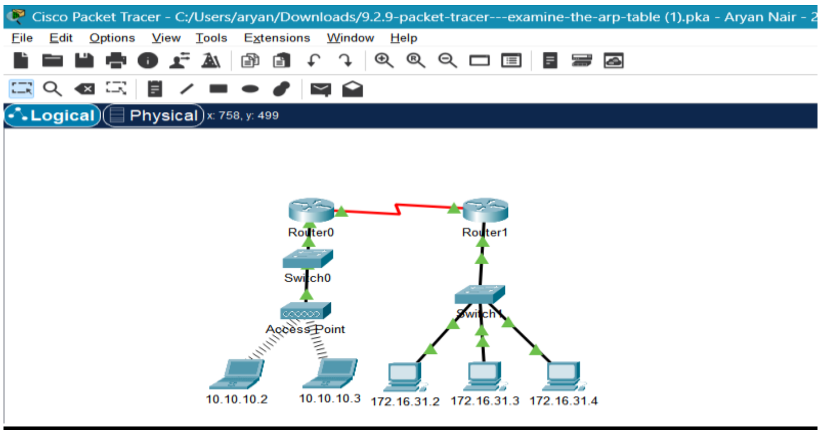
Answer: The MAC address table might have 1 or 2 entries, depending on recent communication.

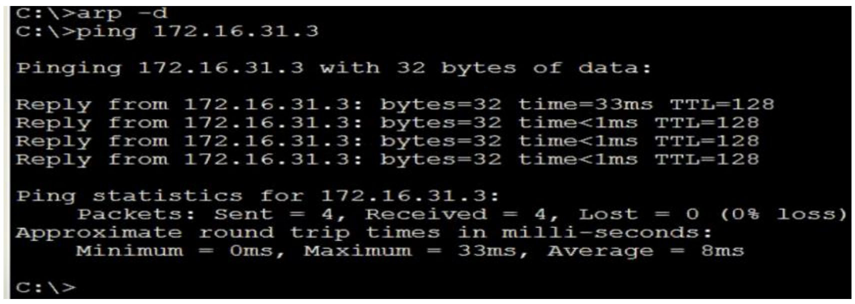
c. Enter the command show arp.

Answer: Yes, there should be an entry for 172.16.31.2.

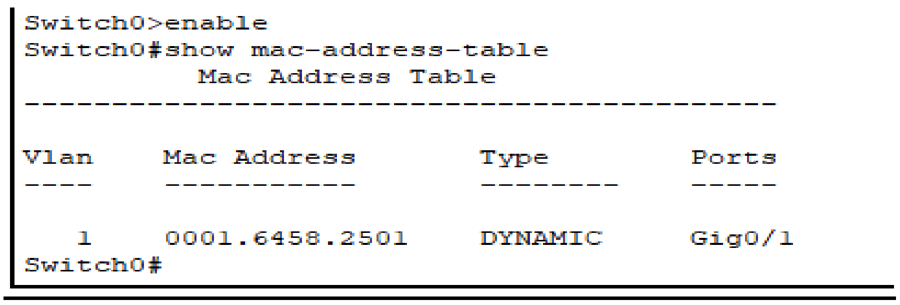
Answer: The first ping may time out because the router must first respond to the ARP request before it can forward the ping packet to the correct destination.

Result:

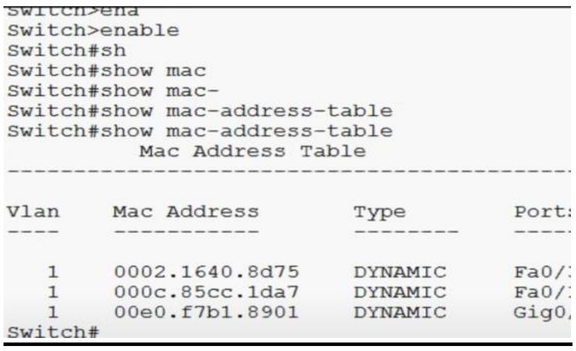

Cisco Packet Tracer - C:/Users/aryan/Downloads/9.2.9-packet-tracer---examine-the-arp-table (1).pka - Aryan Nair
File Edit Options View Tools Extensions Window Help
Logical
Physical
x: 758, y: 499
Router0
Router1
Switch0
Switch1
Access Point
10.10.10.2
10.10.10.3
172.16.31.2
172.16.31.3
172.16.31.4



```
C:\>arp -d
C:\>ping 172.16.31.3

Pinging 172.16.31.3 with 32 bytes of data:

Reply from 172.16.31.3: bytes=32 time=33ms TTL=128
Reply from 172.16.31.3: bytes=32 time<1ms TTL=128
Reply from 172.16.31.3: bytes=32 time<1ms TTL=128
Reply from 172.16.31.3: bytes=32 time<1ms TTL=128

Ping statistics for 172.16.31.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 33ms, Average = 8ms

C:\>
```



```
Switch0>enable
Switch0#show mac-address-table
          Mac Address Table
-------------------------------------------

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

   1    0001.6458.2501    DYNAMIC     Gig0/1
Switch0#
```



```
Switch>ena
Switch>enable
Switch#sh
Switch#show mac
Switch#show mac-
Switch#show mac-address-table
Switch#show mac-address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Port
----    -----------       --------    ----

   1    0002.1640.8d75    DYNAMIC     Fa0/
   1    000c.85cc.1da7    DYNAMIC     Fa0/
   1    00e0.f7b1.8901    DYNAMIC     Gig0
Switch#
```

Experiment 10

Aim: To understand the IPv6 Neighbor Discovery process and analyze the exchange of PDUs (Protocol Data Units) using Packet Tracer simulation. The activity covers both local and remote network communication to demonstrate how Neighbor Discovery Protocol (NDP) works in different scenarios.(Activity 9.3.4)

Instructions:

Part 1: IPv6 Neighbor Discovery - Local Network

Step 1: Check the router for any discovered neighbors.

1. Click on the RTA Router.
2. Select the CLI tab and issue the following command in privileged exec mode:
   show ipv6 neighbors
   If there are any entries, remove them using:
   clear ipv6 neighbors
3. Click on PCA1, select the Desktop tab, and click the Command Prompt icon.

Step 2: Switch to Simulation Mode to capture events.

1. Click the Simulation button in the lower right corner of the Packet Tracer Topology window.
2. Click the Show All/None button in the lower-left part of the Simulation Panel. Ensure the Event List Filters – Visible Events displays *None*.
3. From the command prompt on PCA1, issue the following command to ping PCA2:
   ping -n 1 2001:db8:acad:1::b
   This will start the process of pinging PCA2.
4. Click the Play Capture Forward button (arrow pointing right with a vertical bar). The status bar above the Play Controls should read "Captured to 150" (the exact number may vary).
5. Click the Edit Filters button, select the IPv6 tab at the top, and check the boxes for ICMPv6 and NDP. Click the red X in the upper right of the Edit ACL Filters window. The captured events should now be listed with approximately 12 entries.

Why are ND PDUs present?

ND PDUs are present because the MAC address of the destination is unknown and the Neighbor Discovery Protocol is used to resolve it.

What is the Message Type listed for ICMPv6 under the OSI Model tab?

The message type is ***Neighbor Solicitation***.

6. Click the next event in the Simulation Panel (should be NDP at PCA1).

What changed in the Layer 3 addressing?

The source IP address remains the same, but the destination IP address changes from PCA1 to the multicast address.

What Layer 2 addresses are shown?

A special multicast MAC address for the destination and PCA1's MAC address as the source are shown.

7. Click the first NDP event at SwitchA.

Is there any difference between the In Layers and Out Layers for Layer 2?

No, there is no difference between the In Layers and Out Layers for Layer 2.

8. Click the first NDP event at PCA2 and view the Outbound PDU Details.

What are the following addresses displayed?

- Ethernet II DEST ADDR: Special multicast address
- Ethernet II SRC ADDR: PCA1's MAC address
- IPv6 SRC IP: 2001:db8:acad:1::a
- IPv6 DST IP: 2001:db8:acad:1::b

9. Select the first NDP event at RTA.

Why are there no Out Layers?

There are no Out Layers because the router is not forwarding the packet; it's simply observing it.

10. Click through the Next Layer >> button and read steps 4 through 7 for further explanation.

11. Click the next ICMPv6 event at PCA1.

    Does PCA1 now have all the necessary information to communicate with PCA2?
    Yes, PCA1 has the required information to communicate with PCA2.

12. Click the last ICMPv6 event at PCA1.

    What is the ICMPv6 Echo Message Type?
    The message type is *Echo Request*.

13. Reset the simulation using the Reset Simulation button in the Simulation Panel. Repeat the ping to PCA2 using the same command. Click the Capture Forward button 5 times to complete the ping process.

    Why weren't there any NDP events this time?
    No NDP events occurred because PCA1 already learned the MAC address of PCA2 during the previous ping.

<u>Part 2: IPv6 Neighbor Discovery - Remote Network</u>

<u>Step 1: Capture events for remote communication.</u>

1. Display and clear any entries in the IPv6 neighbor table, as done in Part 1.
2. Switch to simulation mode and ensure Event List Filters – Visible Events displays *None*.
3. From the PCA1 command prompt, issue the command:
   ping -n 1 2001:db8:acad:2::a
   This will initiate a ping to PCB1.
4. Click the Play Capture Forward button. The status bar should read "Captured to 150" (the number may vary).
5. Click Edit Filters, select the IPv6 tab, and check the boxes for ICMPv6 and NDP. The previous events should now be listed, with more entries than in Part 1.
6. Click the first ICMPv6 event.

   What address is being used for the Src IP in the inbound PDU?
   The source IP address is **2001:db8:acad:1::a.**

7. Click the second ICMPv6 event at PCA1. PCA1 should now have enough information to create an ICMPv6 echo request.

   What MAC address is being used for the destination MAC?
   The MAC address used is the router's MAC address.

8. Click the ICMPv6 event at RTA.

   What is missing in the outbound Layer 2 information?
   The destination Layer 2 address is missing.

9. The next few NDP events associate the remaining IPv6 addresses to MAC addresses.

10. Skip to the last ICMPv6 event and verify that all addresses are known, allowing PCB1 to send an echo reply to PCA1.

11. Reset the simulation and repeat the ping from PCA1 to PCB1. Click the Capture Forward button nine times to complete the process.

    Were there any NDP events?
    No, there were no NDP events this time because the router had already learned the necessary addresses.

    What does the destination MAC address correspond to?
    The destination MAC address corresponds to the MAC address of the router interface.

    Why is PCB1 using the router interface MAC address to make its ICMP PDUs?
    PCB1 uses the router's MAC address because the packet must be forwarded through the router to reach PCA1 on a different network.

<u>Step 2: Examine router outputs.</u>

1. Return to Realtime mode and issue the command:
   show ipv6 neighbors

   How many addresses are listed?
   Three addresses are listed.

   What devices are these addresses associated with?
   The addresses are associated with PCA1, PCA2, and PCB1.

Are there any entries for PCA2? Why or why not?

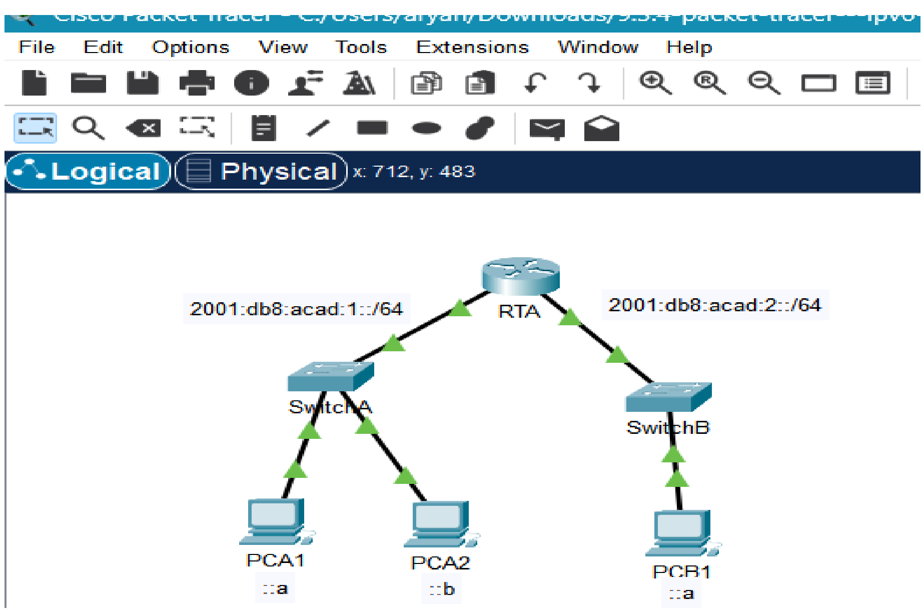
Yes, because the router communicates with PCA2 as part of the IPv6 Neighbor Discovery process.

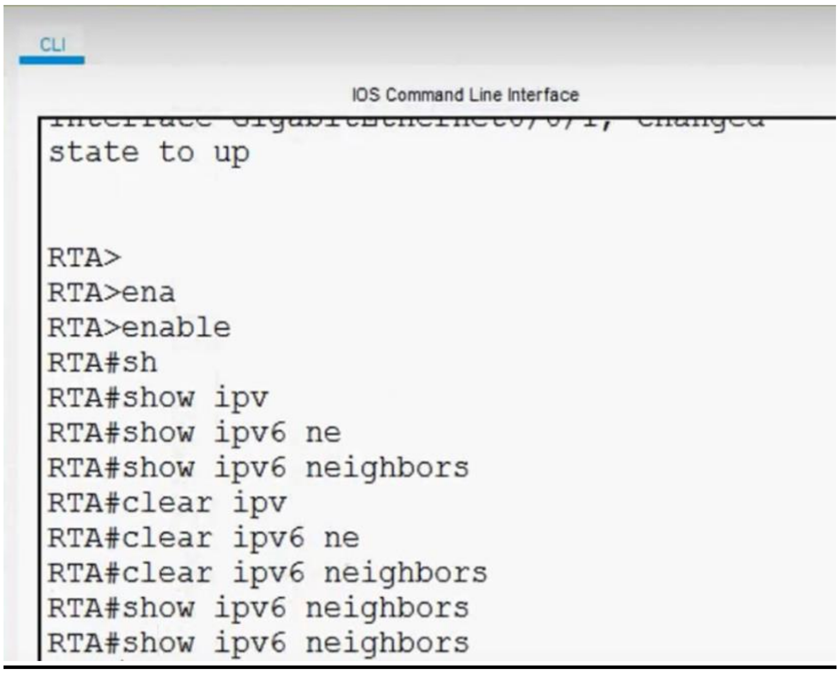
2. Ping PCA2 from the router, then reissue the same command.

Are there entries for PCA2?

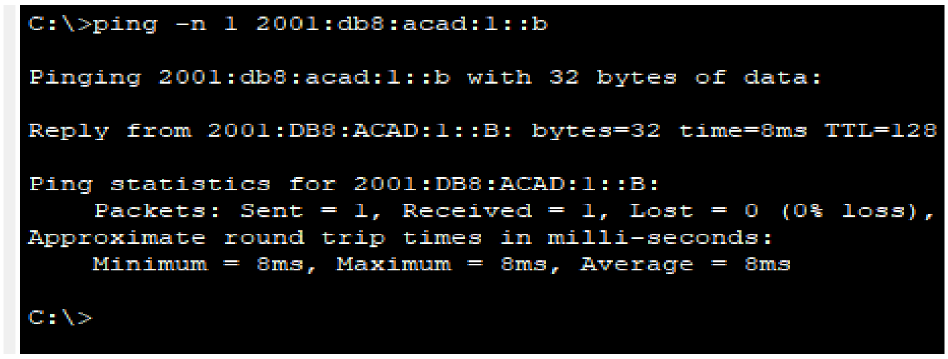
Yes, there are entries for PCA2 after the ping.

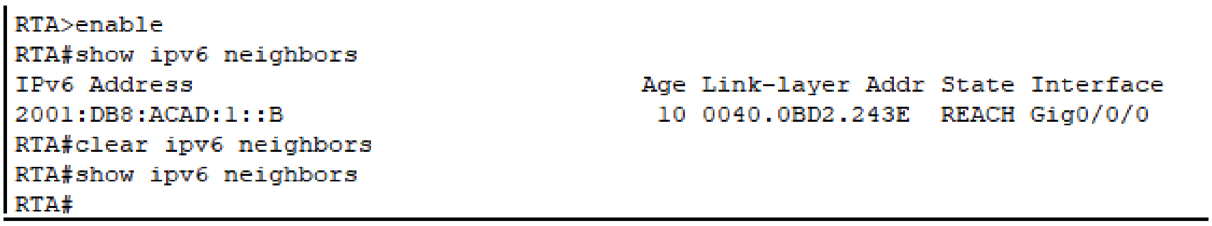
<u>Result:</u>





```
state to up


RTA>
RTA>ena
RTA>enable
RTA#sh
RTA#show ipv
RTA#show ipv6 ne
RTA#show ipv6 neighbors
RTA#clear ipv
RTA#clear ipv6 ne
RTA#clear ipv6 neighbors
RTA#show ipv6 neighbors
RTA#show ipv6 neighbors
```


```
C:\>ping -n 1 2001:db8:acad:1::b

Pinging 2001:db8:acad:1::b with 32 bytes of data:

Reply from 2001:DB8:ACAD:1::B: bytes=32 time=8ms TTL=128

Ping statistics for 2001:DB8:ACAD:1::B:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 8ms, Maximum = 8ms, Average = 8ms

C:\>
```


```
RTA>enable
RTA#show ipv6 neighbors
IPv6 Address                              Age Link-layer Addr State Interface
2001:DB8:ACAD:1::B                         10 0040.0BD2.243E  REACH Gig0/0/0
RTA#clear ipv6 neighbors
RTA#show ipv6 neighbors
RTA#
```

Experiment 11

Objectives

- Part 1: Verify the Default Router Configuration
- Part 2: Configure and Verify the Initial Router Configuration
- Part 3: Save the Running Configuration File

Background

In this activity, you will perform basic router configuration tasks. You will secure access to the CLI and console port using encrypted and plain-text passwords. You will also configure messages for users logging into the router. These banners warn unauthorized users that access is prohibited. Finally, you will verify and save your running configuration.

Instructions

Part 1: Verify the Default Router Configuration

Step 1: Establish a console connection to R1.

- a. Choose a Console cable from the available connections.
- b. Click PCA and select RS 232.
- c. Click R1 and select Console.
- d. Click PCA > Desktop tab > Terminal.
- e. Click OK and press ENTER. You are now able to configure R1.

Step 2: Enter privileged mode and examine the current configuration.

- You can access all the router commands from privileged EXEC mode. However, because many of the privileged commands configure operating parameters, privileged access should be password-protected to prevent unauthorized use.
    - a. Enter privileged EXEC mode by entering the *enable* command.

```
Router> enable
Router#
```

        - Notice that the prompt changed in the configuration to reflect privileged EXEC mode.
    - b. Enter the *show running-config* command.

      Router# show running-config

        - Answers:
            - The router's hostname is Router.
            - The number of Fast Ethernet interfaces will vary depending on the model.
            - The number of Gigabit Ethernet and Serial interfaces also depends on the model.
            - The VTY lines range can be seen in the output as line vty 0 4 or something similar.
    - c. Display the current contents of NVRAM.

      Router# show startup-config

If the router responds with *startup-config is not present*, it means no configuration has been saved to NVRAM yet.

Part 2: Configure and Verify the Initial Router Configuration

Step 1: Configure the initial settings on R1.

- Open a configuration window:
    - a. Configure R1 as the hostname using the command *hostname R1*.
    - b. Configure Message of the Day (MOTD) with the text *Unauthorized access is strictly prohibited*.
    - c. Encrypt all plain-text passwords using the following:
        - Privileged EXEC (unencrypted): *enable password cisco*
        - Privileged EXEC (encrypted): *enable secret itsasecret*
        - Console: *line console 0*, then *password letmein* and *login*.

Step 2: Verify the initial settings on R1.

- Open a configuration window:
    - a. Verify the settings by viewing the configuration using the *show running-config* command.
    - b. Exit the current console session until you see the message:

      R1 con0 is now available
    - c. Press ENTER; you should see:

      Unauthorized access is strictly prohibited.
      User Access Verification
      Password:
    - d. Enter the passwords necessary to return to privileged EXEC mode.
        - Answers:
            - The enable secret password is preferred because it's stored in an encrypted form, making it more secure.
            - Any additional passwords will also appear encrypted once they're configured with the *service password-encryption* command.

Part 3: Save the Running Configuration File

Step 1: Save the configuration file to NVRAM.

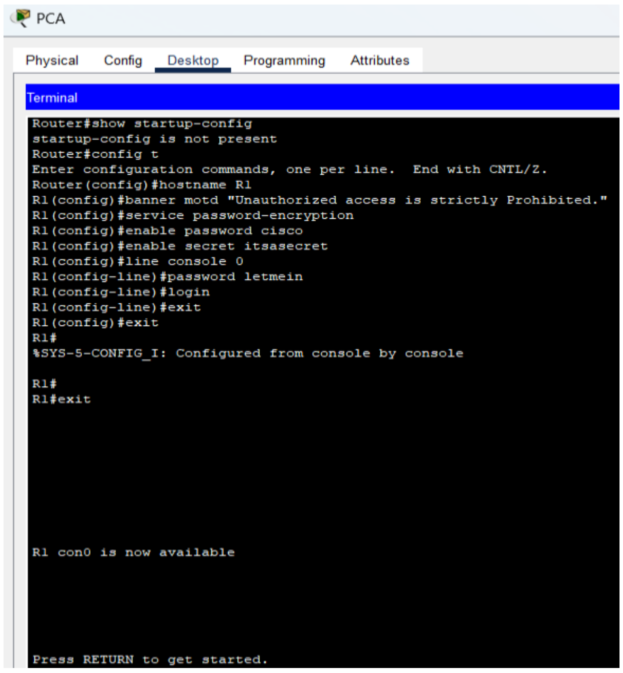
- a. Use the command *copy running-config startup-config* to save the configuration.
- The shortest version of this command is *wr* (write memory).

Step 2: Optional: Save the startup configuration file to flash.

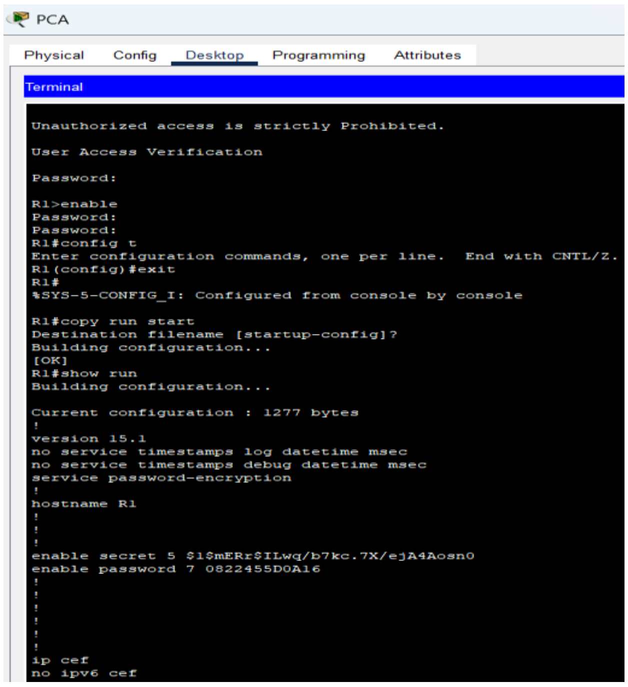
- a. Examine the contents of flash using the command *show flash*.
    - The files listed will include the IOS image, which usually ends with a .bin extension.
- b. Save the startup configuration to flash using *copy startup-config flash*.

When prompted for the destination filename, just press ENTER to accept the default.

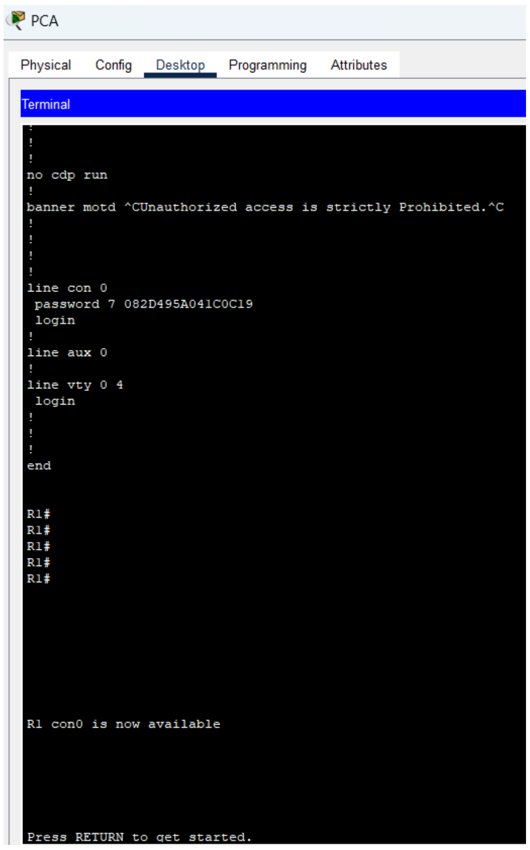
- c. Verify the startup configuration file in flash using the *show flash* command.



```
Router#show startup-config
startup-config is not present
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname R1
R1(config)#banner motd "Unauthorized access is strictly Prohibited."
R1(config)#service password-encryption
R1(config)#enable password cisco
R1(config)#enable secret itsasecret
R1(config)#line console 0
R1(config-line)#password letmein
R1(config-line)#login
R1(config-line)#exit
R1(config)#exit
R1#
%SYS-5-CONFIG_I: Configured from console by console

R1#
R1#exit









R1 con0 is now available






Press RETURN to get started.
```

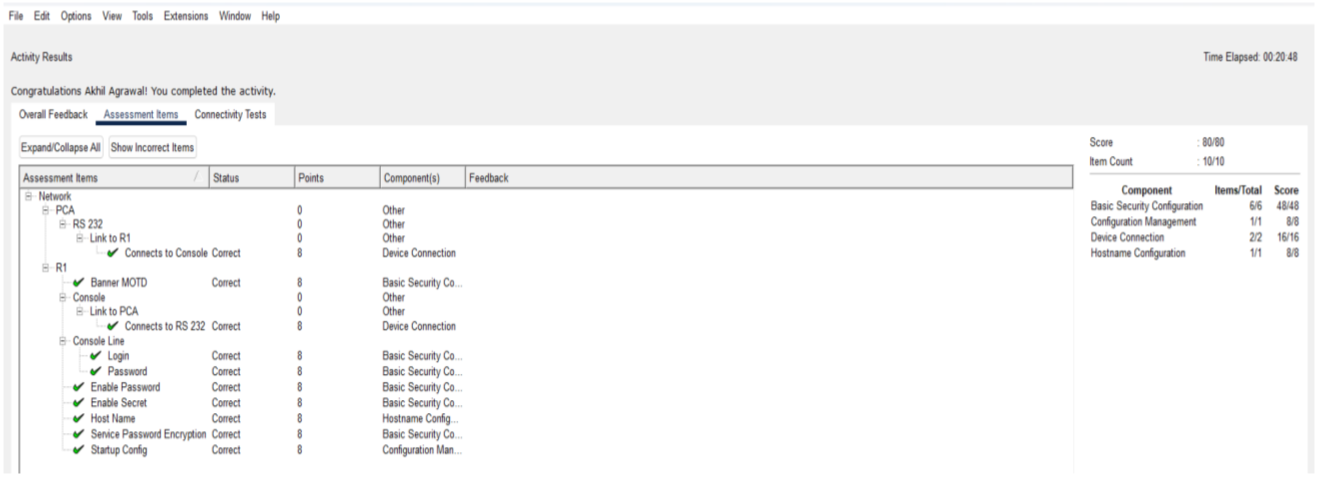


```

Unauthorized access is strictly Prohibited.

User Access Verification

Password:

R1>enable
Password:
Password:
R1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#exit
R1#
%SYS-5-CONFIG_I: Configured from console by console

R1#copy run start
Destination filename [startup-config]?
Building configuration...
[OK]
R1#show run
Building configuration...

Current configuration : 1277 bytes
!
version 15.1
no service timestamps log datetime msec
no service timestamps debug datetime msec
service password-encryption
!
hostname R1
!
!
!
enable secret 5 $1$mERr$ILwq/b7kc.7X/ejA4Aosn0
enable password 7 0822455D0A16
!
!
!
!
!
!
ip cef
no ipv6 cef
```



```
!
!
!
no cdp run
!
banner motd ^CUnauthorized access is strictly Prohibited.^C
!
!
!
!
line con 0
 password 7 082D495A041C0C19
 login
!
line aux 0
!
line vty 0 4
 login
!
!
!
end


R1#
R1#
R1#
R1#
R1#









R1 con0 is now available





Press RETURN to get started.
```

Result:



Activity Results

Time Elapsed: 00:20:48

Congratulations Akhil Agrawal! You completed the activity.

Assessment Items	Status	Points	Component(s)	Feedback
Network				
PCA		0	Other	
RS 232		0	Other	
Link to R1		0	Other	
✓ Connects to Console	Correct	8	Device Connection	
R1				
✓ Banner MOTD	Correct	8	Basic Security Co...	
Console		0	Other	
Link to PCA		0	Other	
✓ Connects to RS 232	Correct	8	Device Connection	
Console Line				
✓ Login	Correct	8	Basic Security Co...	
✓ Password	Correct	8	Basic Security Co...	
✓ Enable Password	Correct	8	Basic Security Co...	
✓ Enable Secret	Correct	8	Basic Security Co...	
✓ Host Name	Correct	8	Hostname Config...	
✓ Service Password Encryption	Correct	8	Basic Security Co...	
✓ Startup Config	Correct	8	Configuration Man...	

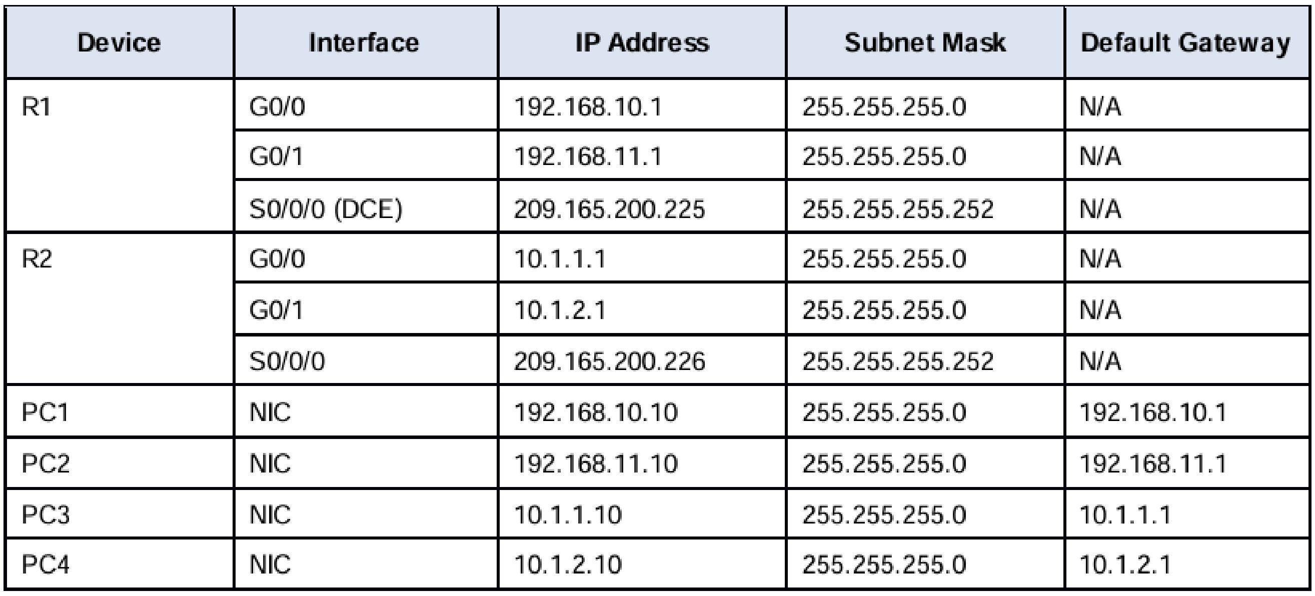
Score : 80/80
Item Count : 10/10

Component	Items/Total	Score
Basic Security Configuration	6/6	48/48
Configuration Management	1/1	8/8
Device Connection	2/2	16/16
Hostname Configuration	1/1	8/8

Experiment 12

AIM: Packet Tracer - Connect a Router to a LAN - Instructions

Addressing Table



Device	Interface	IP Address	Subnet Mask	Default Gateway
R1	G0/0	192.168.10.1	255.255.255.0	N/A
	G0/1	192.168.11.1	255.255.255.0	N/A
	S0/0/0 (DCE)	209.165.200.225	255.255.255.252	N/A
R2	G0/0	10.1.1.1	255.255.255.0	N/A
	G0/1	10.1.2.1	255.255.255.0	N/A
	S0/0/0	209.165.200.226	255.255.255.252	N/A
PC1	NIC	192.168.10.10	255.255.255.0	192.168.10.1
PC2	NIC	192.168.11.10	255.255.255.0	192.168.11.1
PC3	NIC	10.1.1.10	255.255.255.0	10.1.1.1
PC4	NIC	10.1.2.10	255.255.255.0	10.1.2.1

Objectives

Part 1: Display Router Information

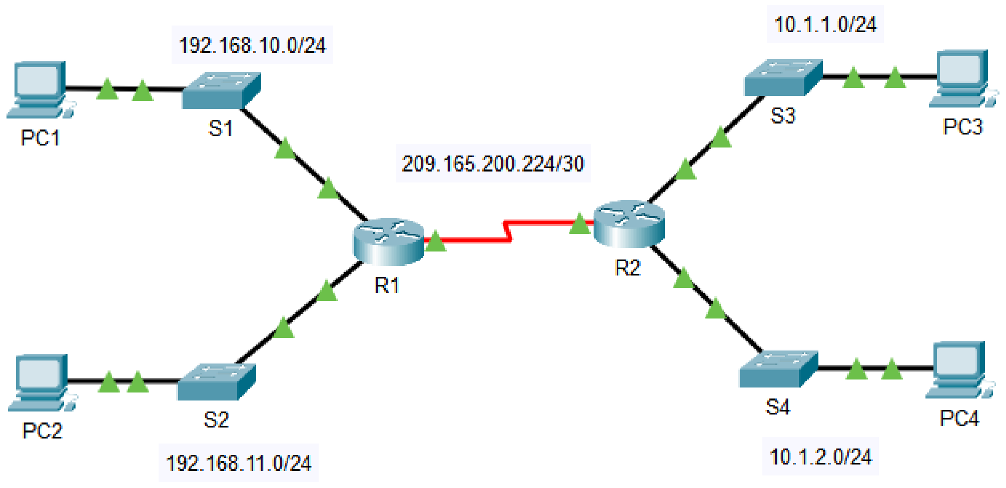
1. Display interface information on R1 using CLI commands.
   - Use the command: `show ip interface brief` to display interface statistics.
   - Use specific commands to display information for the Serial and GigabitEthernet interfaces.

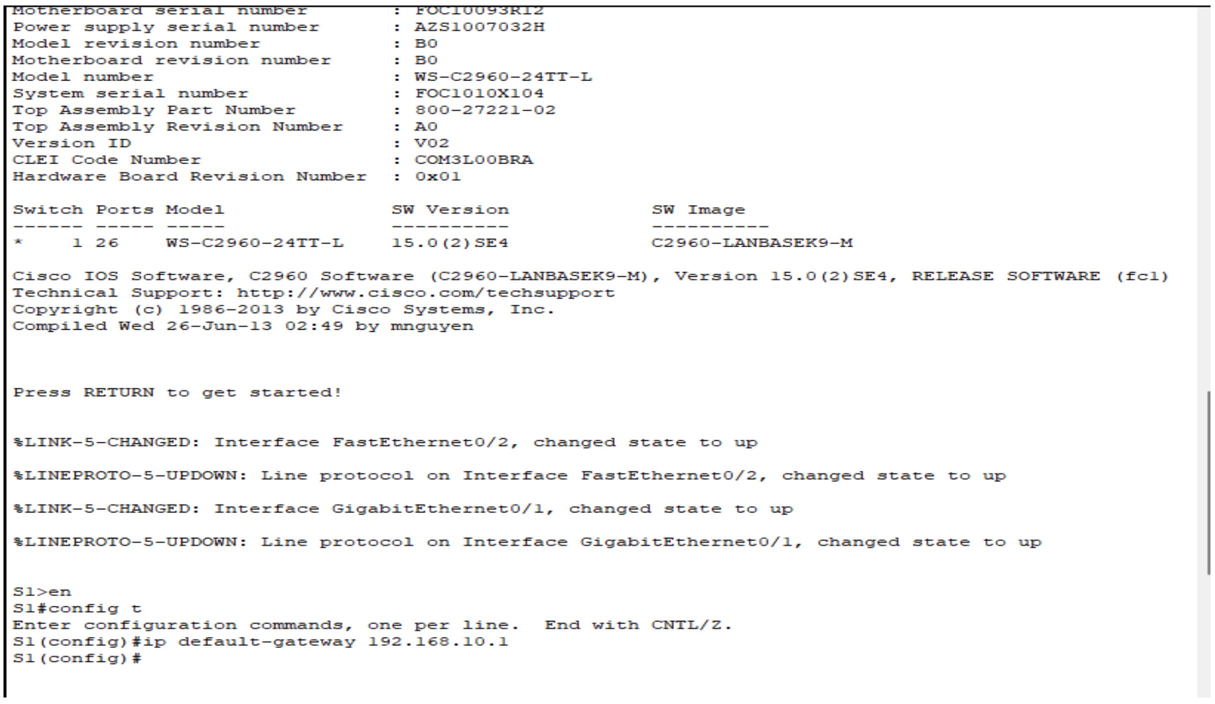
Part 2: Configure Router Interfaces

1. Configure the GigabitEthernet 0/0 interface on R1:
   - Use the commands:
     `interface gigabitethernet 0/0`
     `ip address 192.168.10.1 255.255.255.0`
     `no shutdown`
2. Configure the remaining interfaces on R1 and R2 using the addressing table provided.

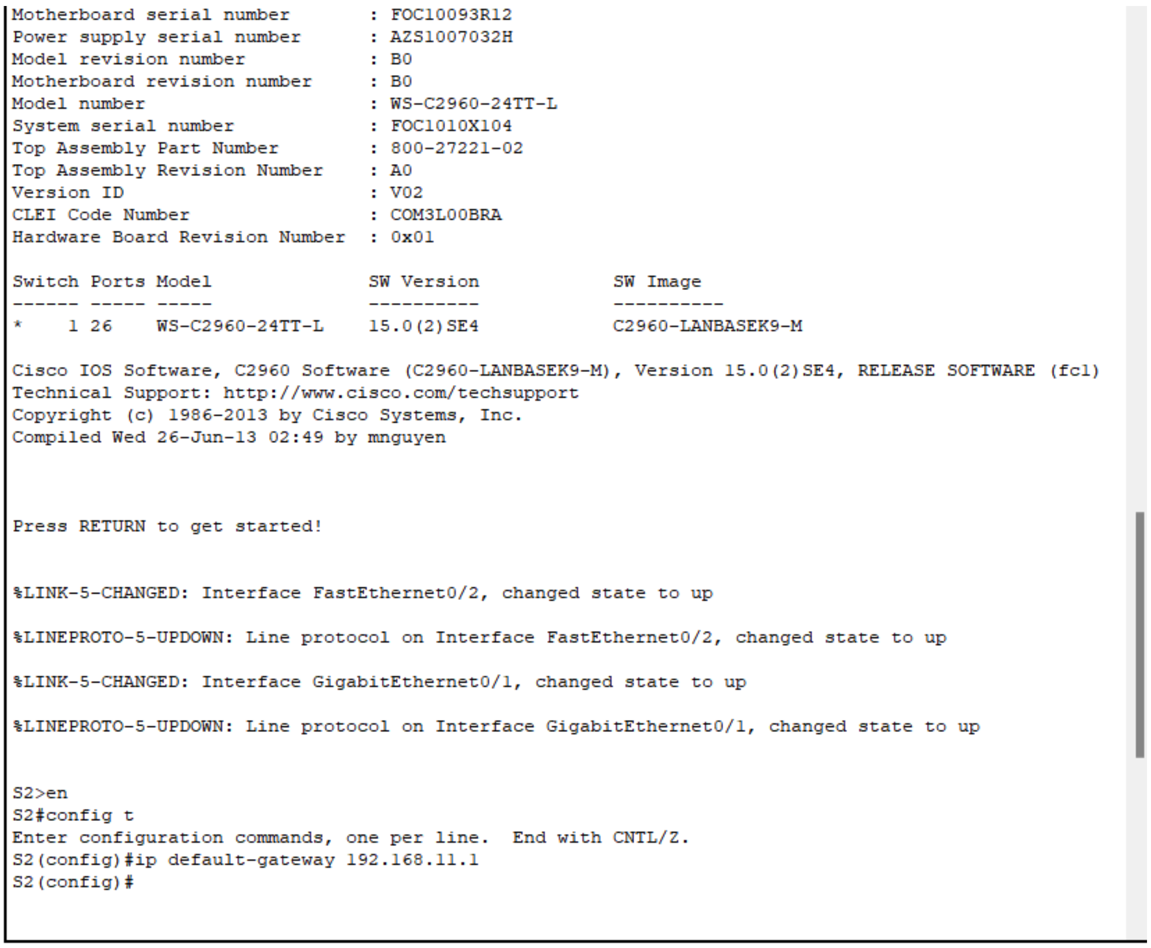
Part 3: Verify the Configuration

1. Use verification commands like `show ip interface brief` and `ping` to check the configurations and test connectivity.
2. Ensure end-to-end connectivity by pinging between devices across the network.

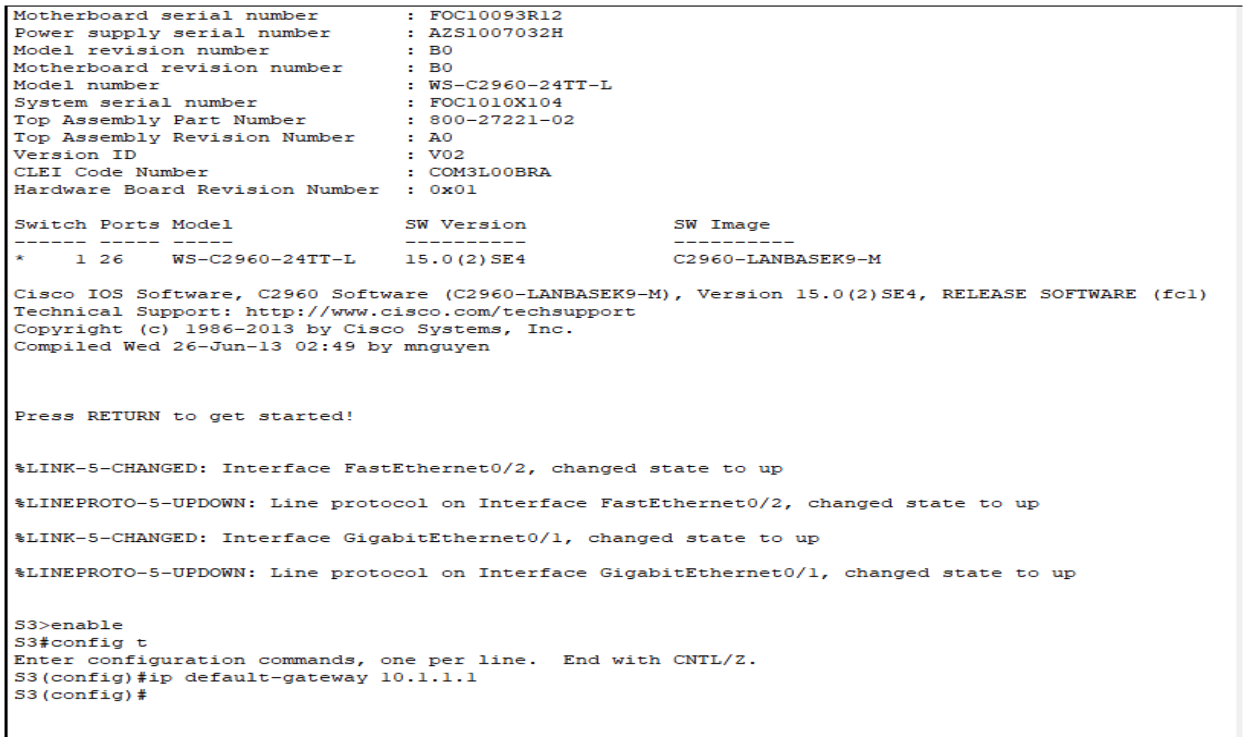




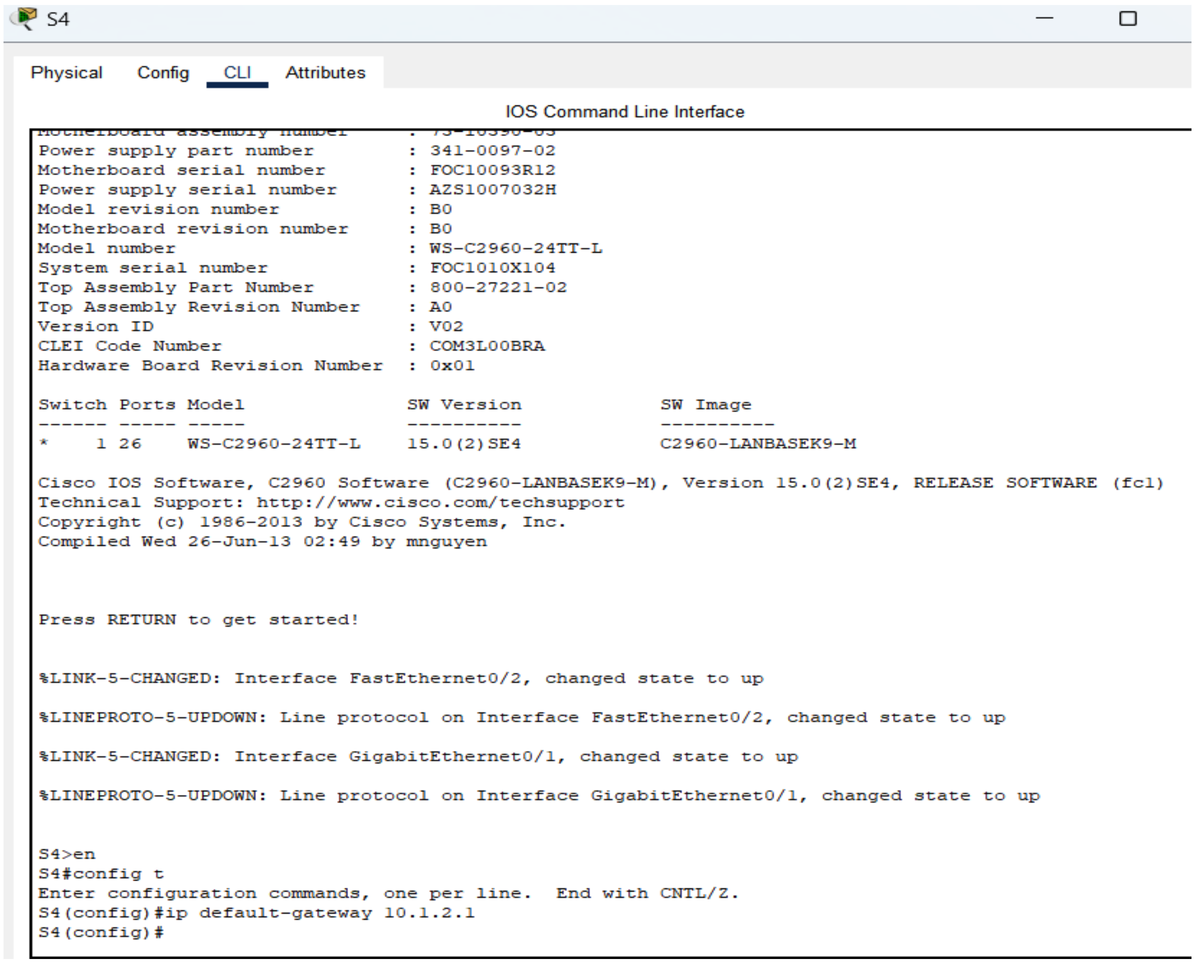
```
Motherboard serial number       : FOC10093R12
Power supply serial number      : AZS1007032H
Model revision number           : B0
Motherboard revision number     : B0
Model number                    : WS-C2960-24TT-L
System serial number            : FOC1010X104
Top Assembly Part Number        : 800-27221-02
Top Assembly Revision Number    : A0
Version ID                      : V02
CLEI Code Number                : COM3L00BRA
Hardware Board Revision Number  : 0x01

Switch Ports Model              SW Version            SW Image
------ ----- -----              ----------            ----------
*    1 26    WS-C2960-24TT-L    15.0(2)SE4            C2960-LANBASEK9-M

Cisco IOS Software, C2960 Software (C2960-LANBASEK9-M), Version 15.0(2)SE4, RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2013 by Cisco Systems, Inc.
Compiled Wed 26-Jun-13 02:49 by mnguyen


Press RETURN to get started!

%LINK-5-CHANGED: Interface FastEthernet0/2, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/2, changed state to up

%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to up


S1>en
S1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
S1(config)#ip default-gateway 192.168.10.1
S1(config)#
```



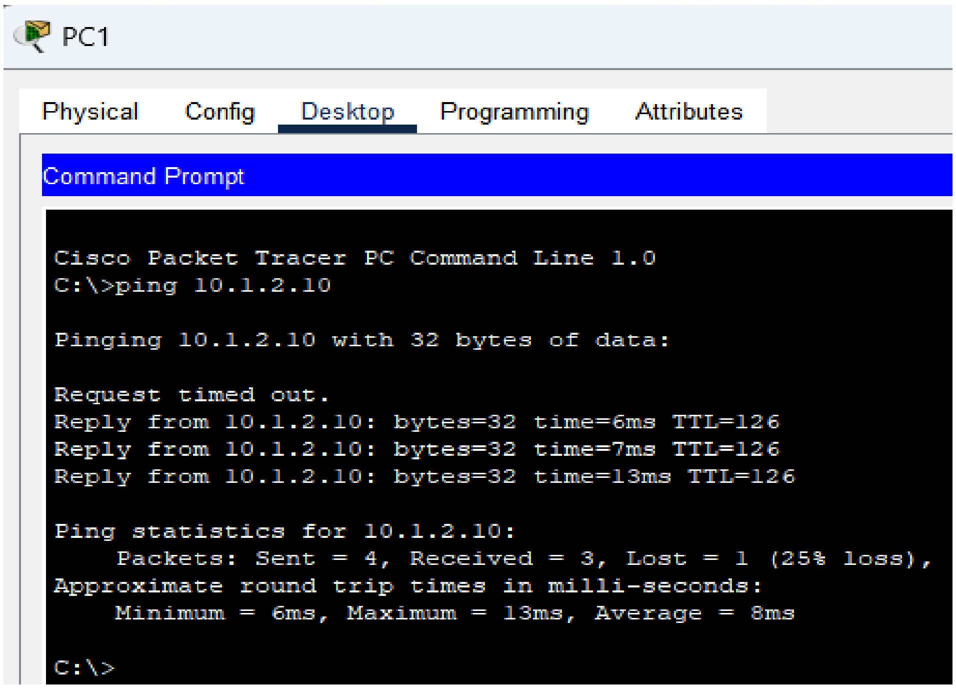
```
Motherboard serial number       : FOC10093R12
Power supply serial number      : AZS1007032H
Model revision number           : B0
Motherboard revision number     : B0
Model number                    : WS-C2960-24TT-L
System serial number            : FOC1010X104
Top Assembly Part Number        : 800-27221-02
Top Assembly Revision Number    : A0
Version ID                      : V02
CLEI Code Number                : COM3L00BRA
Hardware Board Revision Number  : 0x01

Switch Ports Model              SW Version            SW Image
------ ----- -----              ----------            ----------
*    1 26    WS-C2960-24TT-L    15.0(2)SE4            C2960-LANBASEK9-M

Cisco IOS Software, C2960 Software (C2960-LANBASEK9-M), Version 15.0(2)SE4, RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2013 by Cisco Systems, Inc.
Compiled Wed 26-Jun-13 02:49 by mnguyen



Press RETURN to get started!


%LINK-5-CHANGED: Interface FastEthernet0/2, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/2, changed state to up

%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to up


S2>en
S2#config t
Enter configuration commands, one per line.  End with CNTL/Z.
S2(config)#ip default-gateway 192.168.11.1
S2(config)#
```



```
Motherboard serial number       : FOC10093R12
Power supply serial number      : AZS1007032H
Model revision number           : B0
Motherboard revision number     : B0
Model number                    : WS-C2960-24TT-L
System serial number            : FOC1010X104
Top Assembly Part Number        : 800-27221-02
Top Assembly Revision Number    : A0
Version ID                      : V02
CLEI Code Number                : COM3L00BRA
Hardware Board Revision Number  : 0x01

Switch Ports Model              SW Version            SW Image
------ ----- -----              ----------            ----------
*    1 26    WS-C2960-24TT-L    15.0(2)SE4            C2960-LANBASEK9-M

Cisco IOS Software, C2960 Software (C2960-LANBASEK9-M), Version 15.0(2)SE4, RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2013 by Cisco Systems, Inc.
Compiled Wed 26-Jun-13 02:49 by mnguyen


Press RETURN to get started!


%LINK-5-CHANGED: Interface FastEthernet0/2, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/2, changed state to up

%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to up


S3>enable
S3#config t
Enter configuration commands, one per line.  End with CNTL/Z.
S3(config)#ip default-gateway 10.1.1.1
S3(config)#
```

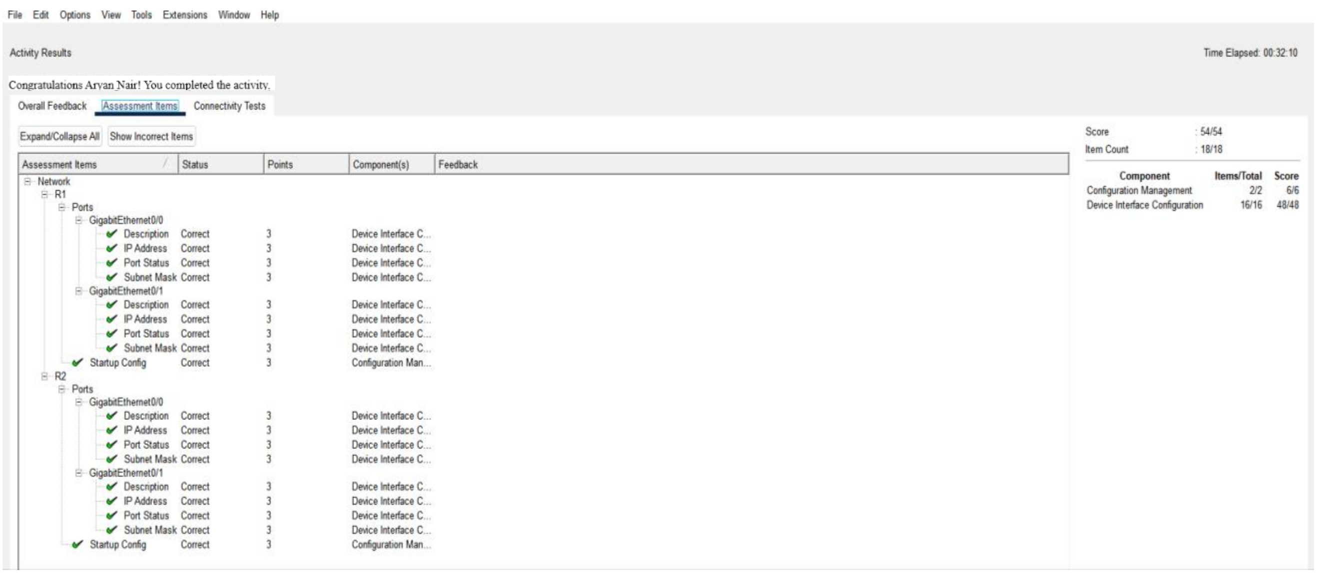


```
Power supply part number          : 341-0097-02
Motherboard serial number         : FOC10093R12
Power supply serial number        : AZS1007032H
Model revision number             : B0
Motherboard revision number       : B0
Model number                      : WS-C2960-24TT-L
System serial number              : FOC1010X104
Top Assembly Part Number          : 800-27221-02
Top Assembly Revision Number      : A0
Version ID                        : V02
CLEI Code Number                  : COM3L00BRA
Hardware Board Revision Number    : 0x01

Switch Ports Model                SW Version            SW Image
------ ----- -----                ----------            ----------
*    1 26    WS-C2960-24TT-L      15.0(2)SE4            C2960-LANBASEK9-M

Cisco IOS Software, C2960 Software (C2960-LANBASEK9-M), Version 15.0(2)SE4, RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2013 by Cisco Systems, Inc.
Compiled Wed 26-Jun-13 02:49 by mnguyen



Press RETURN to get started!


%LINK-5-CHANGED: Interface FastEthernet0/2, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/2, changed state to up

%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to up


S4>en
S4#config t
Enter configuration commands, one per line.  End with CNTL/Z.
S4(config)#ip default-gateway 10.1.2.1
S4(config)#
```



```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.1.2.10

Pinging 10.1.2.10 with 32 bytes of data:

Request timed out.
Reply from 10.1.2.10: bytes=32 time=6ms TTL=126
Reply from 10.1.2.10: bytes=32 time=7ms TTL=126
Reply from 10.1.2.10: bytes=32 time=13ms TTL=126

Ping statistics for 10.1.2.10:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 13ms, Average = 8ms

C:\>
```

Result:


File Edit Options View Tools Extensions Window Help

Activity Results — Time Elapsed: 00:32:10

Congratulations Aryan Nair! You completed the activity.

Overall Feedback | Assessment Items | Connectivity Tests

Expand/Collapse All | Show Incorrect Items

Assessment Items	Status	Points	Component(s)	Feedback
Network				
R1				
Ports				
GigabitEthernet0/0				
✓ Description	Correct	3	Device Interface C...	
✓ IP Address	Correct	3	Device Interface C...	
✓ Port Status	Correct	3	Device Interface C...	
✓ Subnet Mask	Correct	3	Device Interface C...	
GigabitEthernet0/1				
✓ Description	Correct	3	Device Interface C...	
✓ IP Address	Correct	3	Device Interface C...	
✓ Port Status	Correct	3	Device Interface C...	
✓ Subnet Mask	Correct	3	Device Interface C...	
✓ Startup Config	Correct	3	Configuration Man...	
R2				
Ports				
GigabitEthernet0/0				
✓ Description	Correct	3	Device Interface C...	
✓ IP Address	Correct	3	Device Interface C...	
✓ Port Status	Correct	3	Device Interface C...	
✓ Subnet Mask	Correct	3	Device Interface C...	
GigabitEthernet0/1				
✓ Description	Correct	3	Device Interface C...	
✓ IP Address	Correct	3	Device Interface C...	
✓ Port Status	Correct	3	Device Interface C...	
✓ Subnet Mask	Correct	3	Device Interface C...	
✓ Startup Config	Correct	3	Configuration Man...	

Score : 54/54
Item Count : 18/18

Component	Items/Total	Score
Configuration Management	2/2	6/6
Device Interface Configuration	16/16	48/48

Experiment 13

AIM: Instructions for Subnetting an IPv4 Network Using Packet Tracer

Part 1: Subnet the Assigned Network

1. Create a Subnetting Scheme

Subnet the 192.168.0.0/24 network to meet the following requirements.

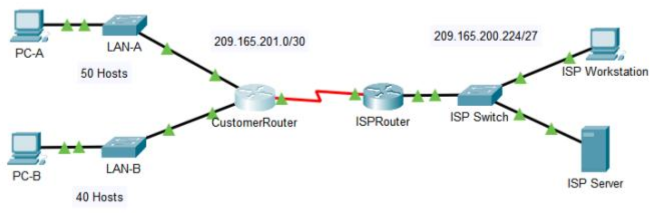
- LAN-A needs a minimum of 50 host IP addresses.
- LAN-B needs a minimum of 40 host IP addresses.
- Create at least two additional unused subnets for future expansion.
- Use the same subnet mask for all devices (no variable-length subnet masks).

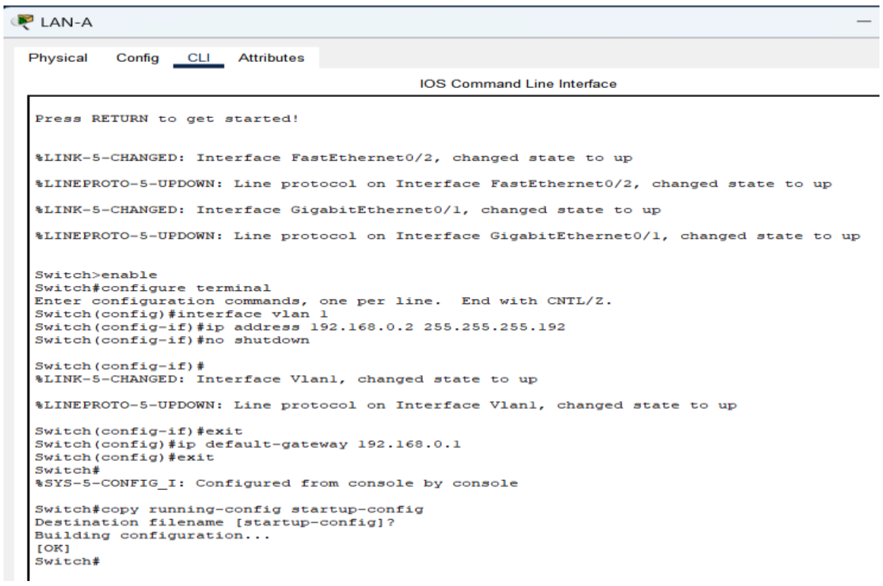
2. Answer the Subnetting Questions:

- Calculate the required number of host addresses for the largest subnet.
- Determine the minimum number of subnets.
- Convert the /24 subnet mask to binary.
- Identify what the ones and zeros represent in the subnet mask.
- Choose appropriate subnet masks and calculate the number of subnets and hosts for each.

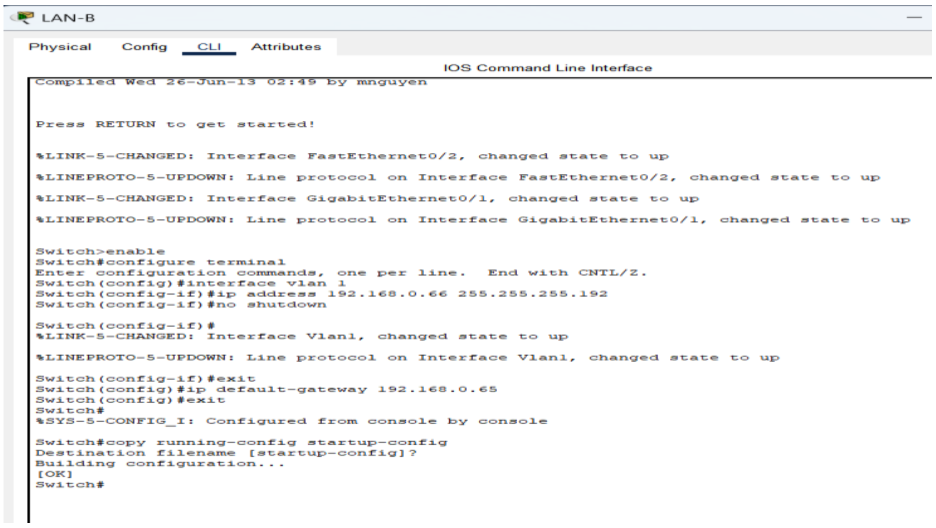
3. Fill in the Subnet Table:

- Determine the subnet mask, then list and fill out the subnets.
- Use the first subnet for LAN-A and the second subnet for LAN-B.





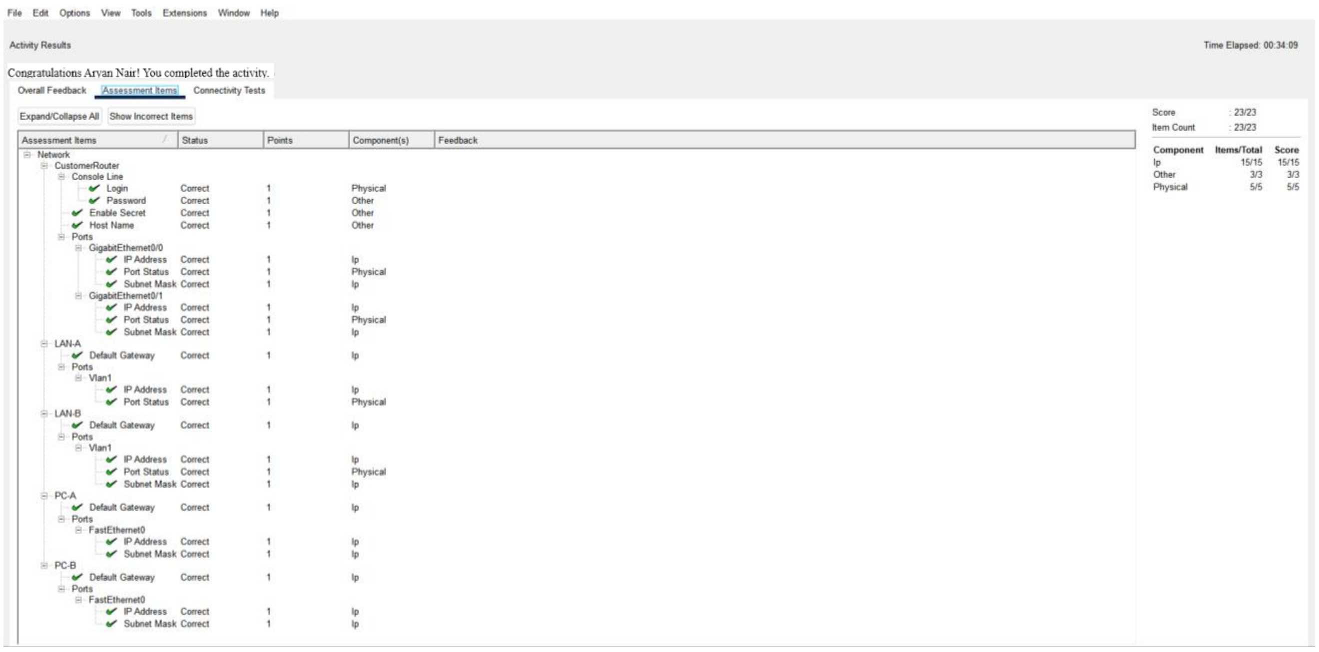
```
Press RETURN to get started!

%LINK-5-CHANGED: Interface FastEthernet0/2, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/2, changed state to up

%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#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Switch(config)#interface vlan 1
Switch(config-if)#ip address 192.168.0.2 255.255.255.192
Switch(config-if)#no shutdown

Switch(config-if)#
%LINK-5-CHANGED: Interface Vlan1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan1, changed state to up

Switch(config-if)#exit
Switch(config)#ip default-gateway 192.168.0.1
Switch(config)#exit
Switch#
%SYS-5-CONFIG_I: Configured from console by console

Switch#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
Switch#
```



```
Compiled Wed 26-Jun-13 02:49 by mnguyen


Press RETURN to get started!


%LINK-5-CHANGED: Interface FastEthernet0/2, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/2, changed state to up

%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#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Switch(config)#interface vlan 1
Switch(config-if)#ip address 192.168.0.66 255.255.255.192
Switch(config-if)#no shutdown

Switch(config-if)#
%LINK-5-CHANGED: Interface Vlan1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan1, changed state to up

Switch(config-if)#exit
Switch(config)#ip default-gateway 192.168.0.65
Switch(config)#exit
Switch#
%SYS-5-CONFIG_I: Configured from console by console

Switch#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
Switch#
```

```
CustomerRouter(config)#interface gigabitEthernet
% Incomplete command.
CustomerRouter(config)#interface gigabitEthernet 0/0
CustomerRouter(config-if)#ip address 192.168.0.1 255.255.255.192
CustomerRouter(config-if)#no shutdown
CustomerRouter(config-if)#int g0/1
CustomerRouter(config-if)#ip address 192.168.0.65 255.255.255.192
CustomerRouter(config-if)#exit
CustomerRouter(config)#enable secret Class123
CustomerRouter(config)#console 0
                          ^
% Invalid input detected at '^' marker.

CustomerRouter(config)#line console 0
CustomerRouter(config-line)#password Class123
CustomerRouter(config-line)#login
CustomerRouter(config-line)#exit
CustomerRouter(config)#hostname CustomerRouter
CustomerRouter(config)#exit
CustomerRouter#
%SYS-5-CONFIG_I: Configured from console by console

CustomerRouter#enable
CustomerRouter#configutre termianl
                          ^
% Invalid input detected at '^' marker.

CustomerRouter#configure termianl
                              ^
% Invalid input detected at '^' marker.

CustomerRouter#enable secret Class123
                      ^
% Invalid input detected at '^' marker.

CustomerRouter#en
CustomerRouter#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
CustomerRouter(config)#enable secret Class123
CustomerRouter(config)#line console 0
CustomerRouter(config-line)#password Cisco123
CustomerRouter(config-line)#
```

Result:



Activity Results

Time Elapsed: 00:34:09

Congratulations Aryan Nair! You completed the activity.

Assessment Items	Status	Points	Component(s)	Feedback
Network				
CustomerRouter				
Console Line				
✓ Login	Correct	1	Physical	
✓ Password	Correct	1	Other	
✓ Enable Secret	Correct	1	Other	
✓ Host Name	Correct	1	Other	
Ports				
GigabitEthernet0/0				
✓ IP Address	Correct	1	Ip	
✓ Port Status	Correct	1	Physical	
✓ Subnet Mask	Correct	1	Ip	
GigabitEthernet0/1				
✓ IP Address	Correct	1	Ip	
✓ Port Status	Correct	1	Physical	
✓ Subnet Mask	Correct	1	Ip	
LAN-A				
✓ Default Gateway	Correct	1	Ip	
Ports				
Vlan1				
✓ IP Address	Correct	1	Ip	
✓ Port Status	Correct	1	Physical	
LAN-B				
✓ Default Gateway	Correct	1	Ip	
Ports				
Vlan1				
✓ IP Address	Correct	1	Ip	
✓ Port Status	Correct	1	Physical	
✓ Subnet Mask	Correct	1	Ip	
PC-A				
✓ Default Gateway	Correct	1	Ip	
Ports				
FastEthernet0				
✓ IP Address	Correct	1	Ip	
✓ Subnet Mask	Correct	1	Ip	
PC-B				
✓ Default Gateway	Correct	1	Ip	
Ports				
FastEthernet0				
✓ IP Address	Correct	1	Ip	
✓ Subnet Mask	Correct	1	Ip	

Score : 23/23
Item Count : 23/23

Component	Items/Total	Score
Ip	15/15	15/15
Other	3/3	3/3
Physical	5/5	5/5

Experiment 14

(Open Ended Question)

Aim: The aim of this project is to develop a secure chat application using Python's TCP socket programming and implementing message encryption using symmetric cryptography to explore network communication between client and server.

Theory:

- **TCP Protocol:** Transmission Control Protocol (TCP) is a reliable, connection-oriented protocol that ensures data delivery in the correct sequence between client and server. TCP guarantees that no data is lost during transmission.
- **Encryption:** To ensure secure communication between the client and server, we implement message encryption using a symmetric encryption algorithm from Python's cryptography library.

Server code:

```
import socket

from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes

from cryptography.hazmat.backends import default_backend

import os

SECRET_KEY = b'\x00' * 32

# Encryption / Decryption functions

def encrypt_message(key, message):

    iv = os.urandom(16)  # Initialization vector

    cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())

    encryptor = cipher.encryptor()

    ciphertext = encryptor.update(message.encode()) + encryptor.finalize()

    return iv + ciphertext  # Concatenate IV with the ciphertext


def decrypt_message(key, ciphertext):

    iv = ciphertext[:16]  # Extract the IV

    actual_ciphertext = ciphertext[16:]

    cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())

    decryptor = cipher.decryptor()

    decrypted_message = decryptor.update(actual_ciphertext) + decryptor.finalize()
    return decrypted_message.decode()
def handle_client(client_socket):
    """Handles communication with the connected client."""
    try:
        while True:
            encrypted_message = client_socket.recv(1024)
            if not encrypted_message:
                break
            decoded_message = decrypt_message(SECRET_KEY, encrypted_message)
            if decoded_message.lower() in ['quit', 'q']:
                print("Client has exited the chat.")
                break
            print(f"Client: {decoded_message}")
            response = input("Server: ")
            encrypted_response = encrypt_message(SECRET_KEY, response)
            client_socket.send(encrypted_response)
    except Exception as e:
        print(f"Error during communication: {e}")
    finally:
        client_socket.close()
        print("Connection with the client has been closed.")
def start_server(host='localhost', port=4123):
    """Starts the server and waits for a client to connect."""
    try:
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_socket.bind((host, port))
        server_socket.listen(1)
        print(f"Server is listening on {host}:{port}...")

        client_socket, client_address = server_socket.accept()
        print(f"Connection established with {client_address}!")
        handle_client(client_socket)
    except Exception as e:
        print(f"Server error: {e}")
    finally:
        server_socket.close()
if __name__ == "__main__":
    start_server()
```

Client code:

```
import socket
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
import os
# Key for encryption/decryption (must match the server's key)
SECRET_KEY = b'\x00' * 32

# Encryption / Decryption functions
def encrypt_message(key, message):
    iv = os.urandom(16)  # Initialization vector
    cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())
    encryptor = cipher.encryptor()
    ciphertext = encryptor.update(message.encode()) + encryptor.finalize()
    return iv + ciphertext  # Concatenate IV with the ciphertext

def decrypt_message(key, ciphertext):
    iv = ciphertext[:16]  # Extract the IV
    actual_ciphertext = ciphertext[16:]
    cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())
    decryptor = cipher.decryptor()
    decrypted_message = decryptor.update(actual_ciphertext) + decryptor.finalize()
    return decrypted_message.decode()
```

```
def start_client(host='localhost', port=4123):
    """Connects to the server and starts communication."""
    try:
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client_socket.connect((host, port))
        print(f"Connected to the server at {host}:{port}")
        while True:
            message = input("Client: ")
            encrypted_message = encrypt_message(SECRET_KEY, message)
            client_socket.send(encrypted_message)
            if message.lower() in ['quit', 'q']:
                print("Closing connection.")
                break
            encrypted_response = client_socket.recv(1024)
            if not encrypted_response:
                break
            decoded_response = decrypt_message(SECRET_KEY, encrypted_response)
            print(f"Server: {decoded_response}")

    except Exception as e:
        print(f"Client error: {e}")
    finally:
        client_socket.close()
if __name__ == "__main__":
    start_client()
```

Explanation of the Code (Server)

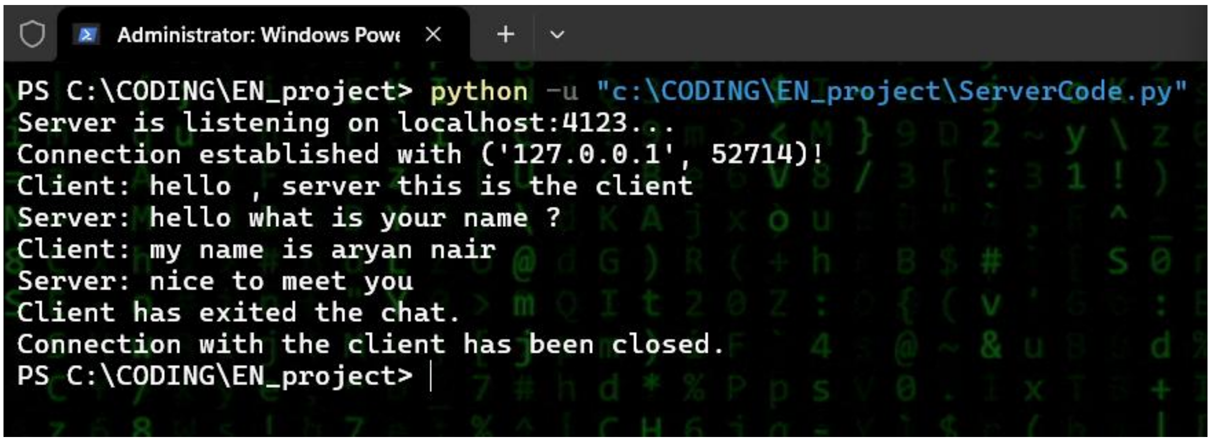
1. Imports:
    - The code imports necessary modules for socket programming and cryptography.
    - socket is used for network communication, while cryptography provides functions for encryption and decryption.
2. Secret Key:
    - A secret key, SECRET_KEY, is defined as 32 bytes of zeros. This key is used for AES encryption.
3. Encryption Function (encrypt_message):
    - Takes a key and a message as parameters.
    - Generates a random 16-byte Initialization Vector (IV) using os.urandom().
    - Sets up an AES cipher in CFB mode using the provided key and IV.
    - Encrypts the message and concatenates the IV with the resulting ciphertext.
    - Returns the combined result.
4. Decryption Function (decrypt_message):
    - Takes a key and ciphertext as parameters.
    - Extracts the IV from the first 16 bytes of the ciphertext.
    - Sets up the same AES cipher as used in encryption.
    - Decrypts the ciphertext and returns the original message as a string.
5. Client Handler Function (handle_client):
    - Manages the communication with a connected client.
    - Continuously listens for encrypted messages from the client.
    - Decrypts incoming messages and checks if the client wants to exit (by sending 'quit' or 'q').
    - If the client sends a message, the server prompts for a response, encrypts it, and sends it back to the client.
6. Server Initialization Function (start_server):
    - Sets up a TCP server socket, binds it to a host and port (default: localhost:4123), and listens for incoming connections.
    - Accepts a client connection and calls handle_client() to manage the communication.
7. Main Block:
    - The server starts listening for clients when the script is run directly.

Explanation of the Code (Client)

1. Imports:
    - Similar to the server, the client imports the necessary modules for socket programming and cryptography.
2. Secret Key:
    - The same SECRET_KEY is defined here to ensure encryption and decryption compatibility with the server.
3. Encryption Function (encrypt_message):
    - Works the same way as in the server code. It generates an IV, creates an AES cipher, encrypts the message, and returns the IV concatenated with the ciphertext.
4. Decryption Function (decrypt_message):
    - Similar to the server's function, it extracts the IV and decrypts the message.
5. Client Initialization Function (start_client):
    - Creates a socket and connects to the server using the specified host and port.
    - Once connected, it enters a loop where it allows the user to input messages.
    - For each message, it encrypts it and sends it to the server. If the message is 'quit' or 'q', it closes the connection.
    - The client also waits for a response from the server, decrypts it, and displays it.
6. Main Block:
    - The client connects to the server and starts communication when the script is run directly.

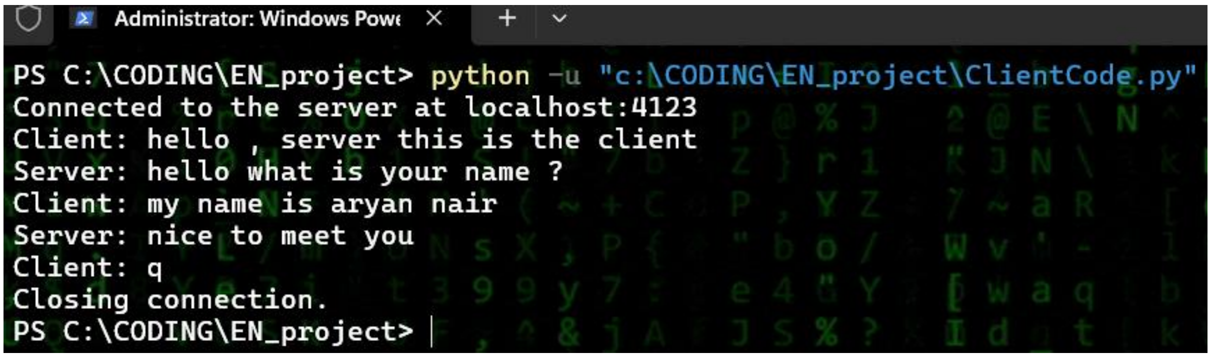
Outputs:

Server Output:



```
PS C:\CODING\EN_project> python -u "c:\CODING\EN_project\ServerCode.py"
Server is listening on localhost:4123...
Connection established with ('127.0.0.1', 52714)!
Client: hello , server this is the client
Server: hello what is your name ?
Client: my name is aryan nair
Server: nice to meet you
Client has exited the chat.
Connection with the client has been closed.
PS C:\CODING\EN_project>
```

Client Output:



```
PS C:\CODING\EN_project> python -u "c:\CODING\EN_project\ClientCode.py"
Connected to the server at localhost:4123
Client: hello , server this is the client
Server: hello what is your name ?
Client: my name is aryan nair
Server: nice to meet you
Client: q
Closing connection.
PS C:\CODING\EN_project>
```

Conclusion:

In conclusion, the provided server and client code implements a secure communication system using AES encryption in CFB mode. By encrypting messages with a shared secret key, the code ensures confidentiality and protects against eavesdropping during transmission. The use of an Initialization Vector (IV) for each message enhances security, preventing identical messages from producing the same ciphertext. This setup is robust for simple text-based communication, demonstrating fundamental principles of socket programming and cryptography. Overall, it serves as a foundational example for developers seeking to understand secure client-server interactions.