DYPIU

D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

D Y Patil International University Akurdi, Pune

Second Year B.Tech. CSE

Computer Networks (CSE 2104)

Lab Manual

School of Computer Science Engineering and Applications

DYPIU

D Y Patil International University Akurdi, Pune

Vision of the University:
To Create a vibrant learning environment – fostering innovation and creativity, experiential learning, which is inspired by research, and focuses on regionally, nationally and globally relevant areas.

Mission
- **To provide a diverse, vibrant and inspirational learning environment.**
- **To establish the university as a leading experiential learning and research-oriented center.**
- **To Become a responsive university serving the needs of industry and society.**
- **To embed internationalization, employability and value thinking.**

Computer Networks

Course Objectives:
- To understand the fundamentals of networking topologies. OSI, TCP/IP Model
- To illustrate the working and function of Physical and data link layer
- To analyze different routing algorithms
- To understand UDP and TCP protocol.
- To demonstrate socket programming

Course Outcomes:
- On completion of the course, learner will be able to
- CO1: Summarize the fundamental concepts of computer network, topologies,
- CO2: Illustrate the working functions of physical and data link layer.
- CO3: To illustrate different routing algorithms.
- CO4: Implement client server application.
- CO5: Illustrate any one automation system using cisco packet tracer.

Rules and Regulations for Laboratory:
- Students should be regular and punctual to all the Lab practical
- Lab assignments and practical's should be submitted within a given time.
- Mobile phones are strictly prohibited in the Lab.
- Please shut down the Electronic Devices before leaving the Lab.
- Please keep the chair in proper position before leaving the Lab
- Maintain proper discipline in Lab

D Y Patil International University Akurdi, Pune

Computer Networks (CSE 2104)

Index

CERTIFICATE

This is to certify that **Mr**. **Neeraj Chormale**, PRN No.**20220802071** of B. tech CSE 2nd Year Class has completed practical in the course of Computer Networks Second Year, within DYPIU Akurdi, Pune during the academic year 2023 - 2024.

Date: 30/4/24 **TA** **Faculty**

Experiment No. 1

Objective: Exploring Different Network Topologies with Cisco Packet Tracer.

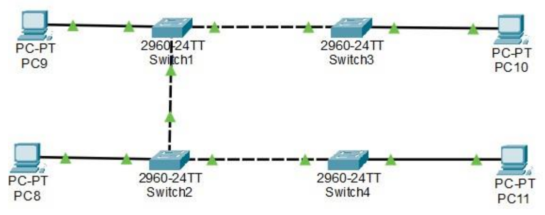
Software Required: Cisco Packet Tracer

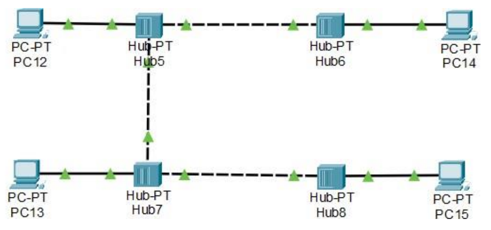
Prerequisites: Basic understanding of networking concepts such as nodes, links, and network protocols. Familiarity with Cisco Packet Tracer interface.

Theory:

1) Ring Topology -

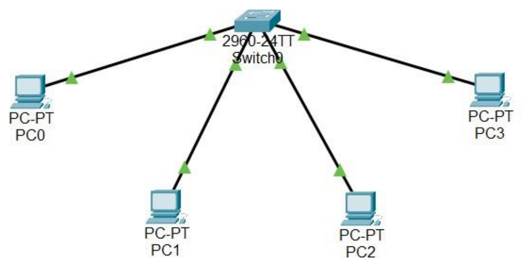
- Objective: Build a network with a ring topology.
- Topology: Connect multiple PCs or switches in a circular configuration, ensuring that each node has exactly two neighbors.
- Configuration: Configure IP addresses for devices. Test connectivity by sending data packets around the ring.

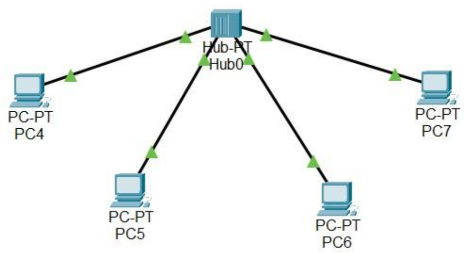




2) Star Topology -

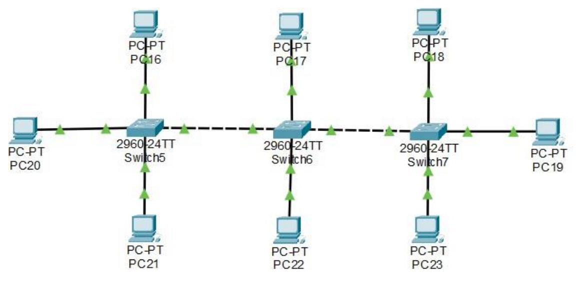
- Objective: Build a network with a Star topology.
- Topology: Connect multiple PCs or switches to a central switch or router.
- Configuration: Assign IP addresses to devices. Test connectivity between devices and the central switch/router.

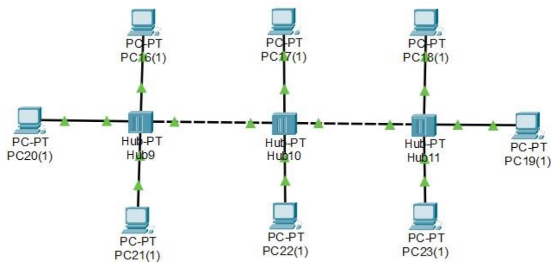




3) Bus Topology Objective -

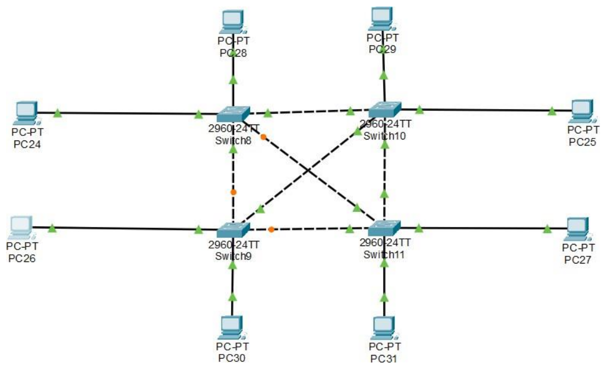
- Objective: Construct a network with a bus topology.
- Topology: Connect multiple PCs or switches in a linear arrangement, all sharing a common communication channel.
- Configuration: Assign IP addresses to devices. Test data transmission along the bus.





4) Mesh Topology

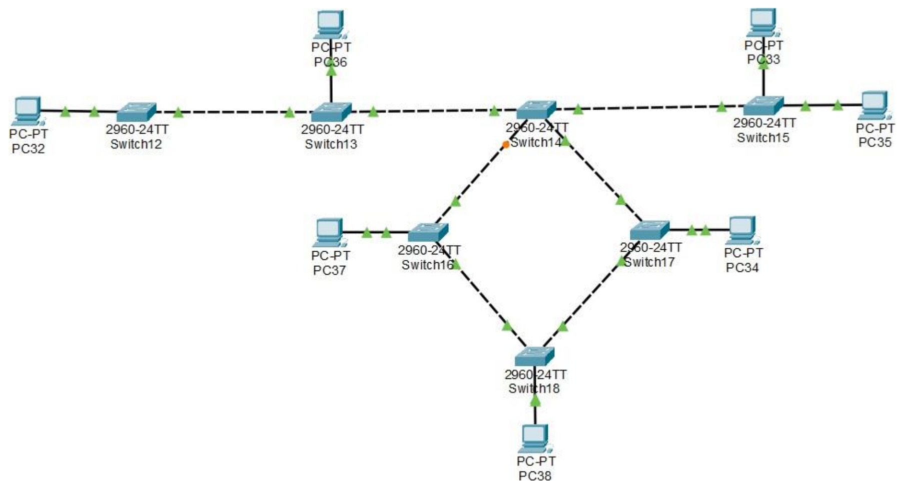
- Objective: Implement a network with a mesh topology.
- Topology: Create a fully connected mesh network where every node is connected to every other node.
- Configuration: Assign IP addresses to devices. Test connectivity between any



two nodes within the mesh.

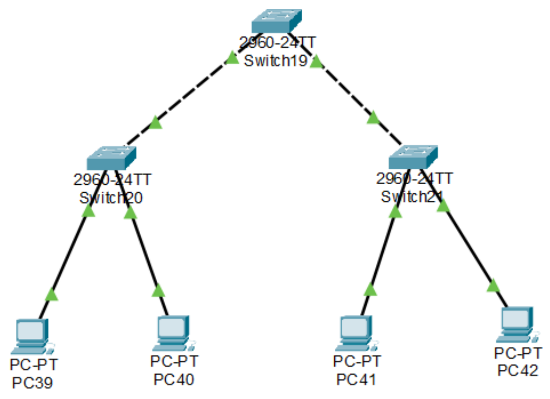
5) Hybrid Topology -

- Objective: Design a network with a hybrid topology.
- Topology: Combine elements of different topologies such as star, ring, and mesh to form a hybrid network.
- Configuration: Configure IP addresses for devices. Test connectivity and analyze the advantages of the hybrid approach.



6) Tree Topology

- Objective: Construct a network with tree topology.
- Topology: Build a hierarchical network structure resembling a tree, with a root node and multiple branches.
- Configuration: Assign IP addresses to devices. Test data flows between nodes at different levels of the tree.



Steps:

1. Take the required amount of switch and end devices to represent respective topology.
2. Connect them using the respective type of connector required.
3. Assign IP to every end device.
4. Use CPT protocol data unit to check the connection is properly established or not.

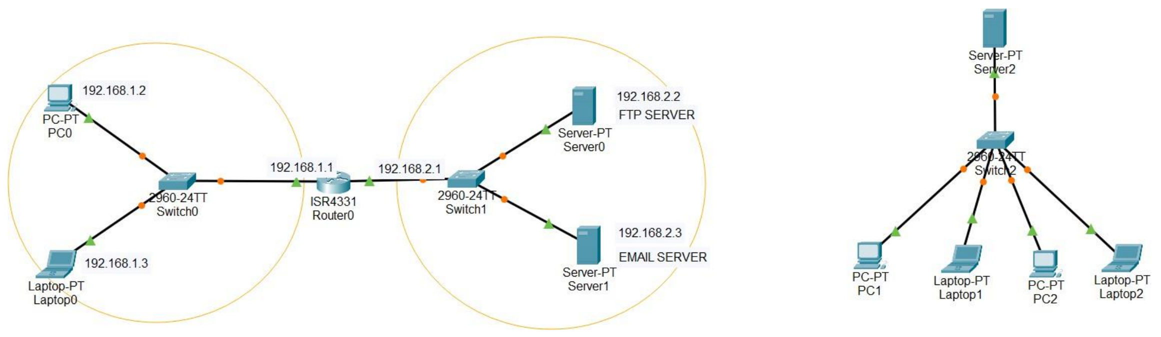
Conclusion:

Experiment No. 2

Objective: The objective of this lab is to guide through the process of configuring Email, Web, and DHCP servers using Cisco Packet Tracer.

Prerequisites: Basic understanding of networking concepts such as IP addressing and server administration. Familiarity with Cisco Packet Tracer interface.

Software Required: Cisco Packet Tracer



Theory:

1: DHCP Server Configuration

Objective: Set up a Dynamic Host Configuration Protocol (DHCP) server to automatically assign IP addresses to network devices.

Topology: Create a network topology with PCs and a router. Introduce a DHCP server connected to the router.

Configuration: Configure DHCP server settings such as IP address range, subnet mask, default gateway, and DNS server. Test DHCP functionality by connecting new devices to the network and verifying IP address assignment.

2: Email Server Configuration

Objective: Configure an Email server to send and receive emails within the network. Topology: Expand the existing network topology with a dedicated Email server. Connect Email clients (PCs) to the network.

Configuration: Set up Email accounts, domains, and mailboxes. Test Email functionality by sending and receiving emails between clients.

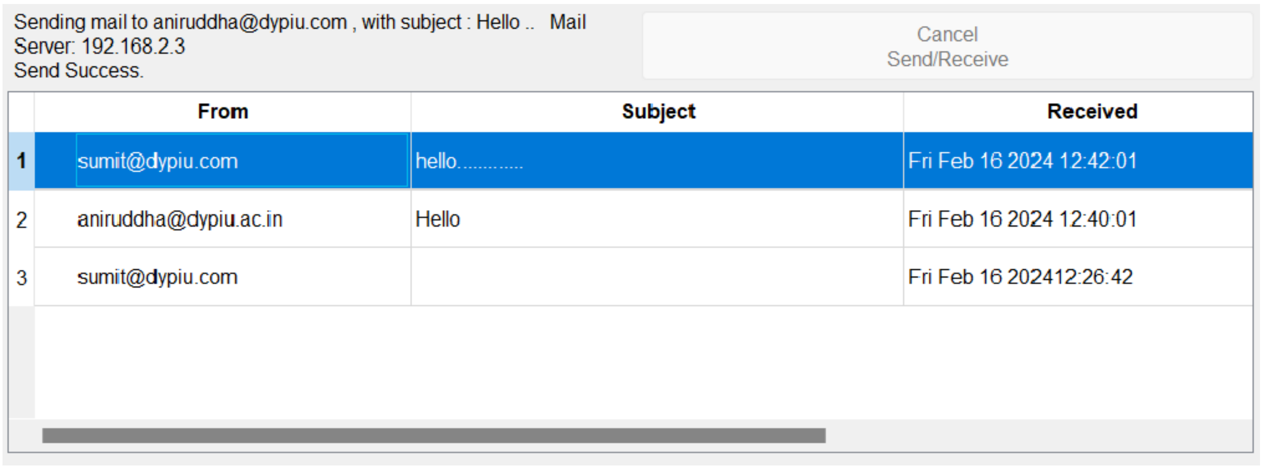
3: Web Server Configuration

Objective: Set up a Web server to host and serve web pages over the network.
Topology: Further expand the network topology with a Web server. Connect Web clients (PCs) to the network.

Configuration: Install and configure a Web server software (e.g., Apache, Microsoft IIS). Create and upload web pages to the Web server's root directory. Test Web server functionality by accessing hosted web pages from client PCs.

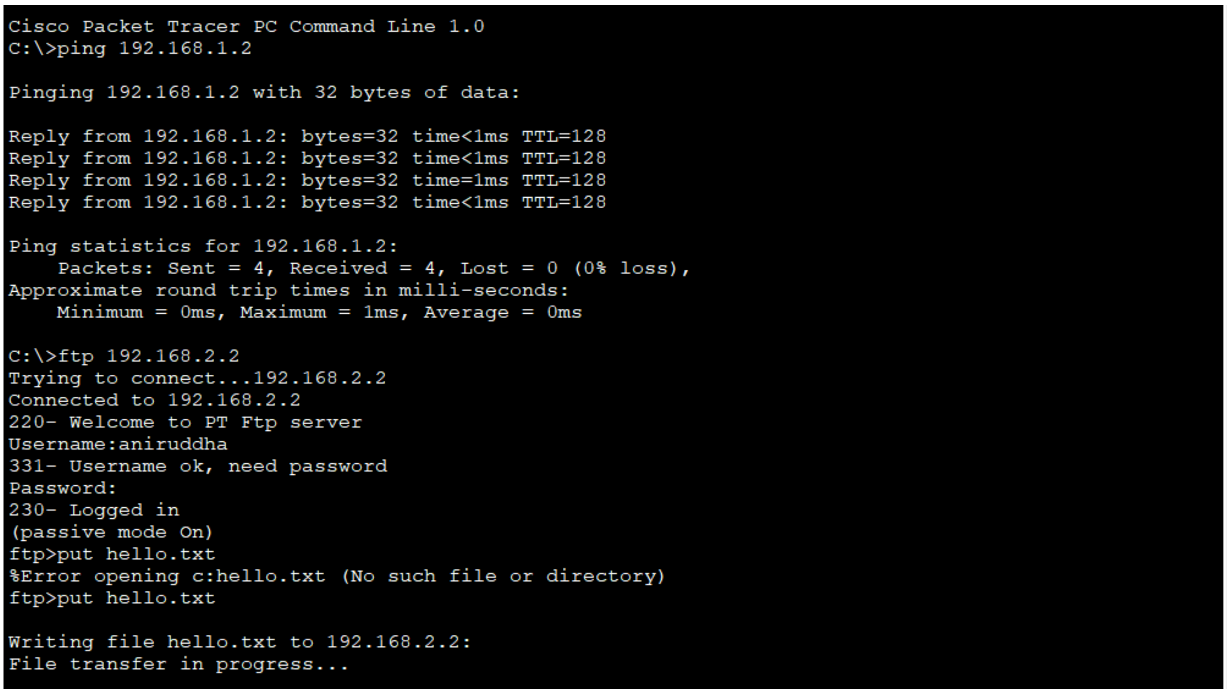
Output:

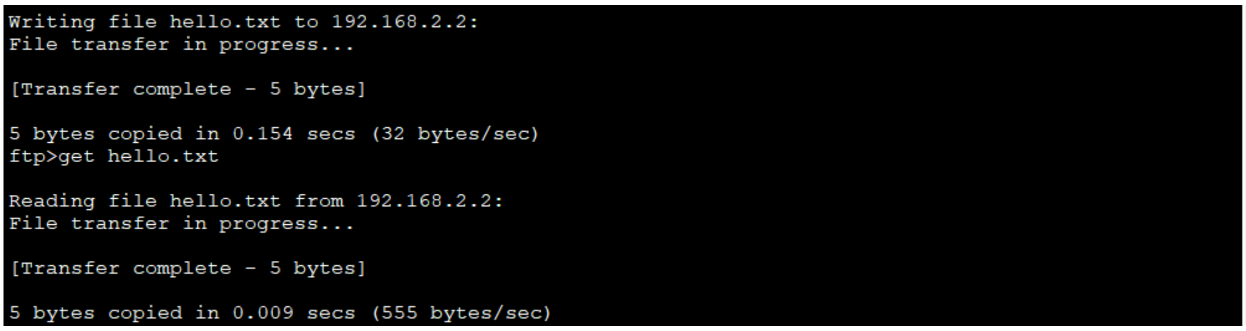
1. Email


Sending mail to aniruddha@dypiu.com , with subject : Hello .. Mail Server: 192.168.2.3
Send Success.

	From	Subject	Received
1	sumit@dypiu.com	hello............	Fri Feb 16 2024 12:42:01
2	aniruddha@dypiu.ac.in	Hello	Fri Feb 16 2024 12:40:01
3	sumit@dypiu.com		Fri Feb 16 202412:26:42

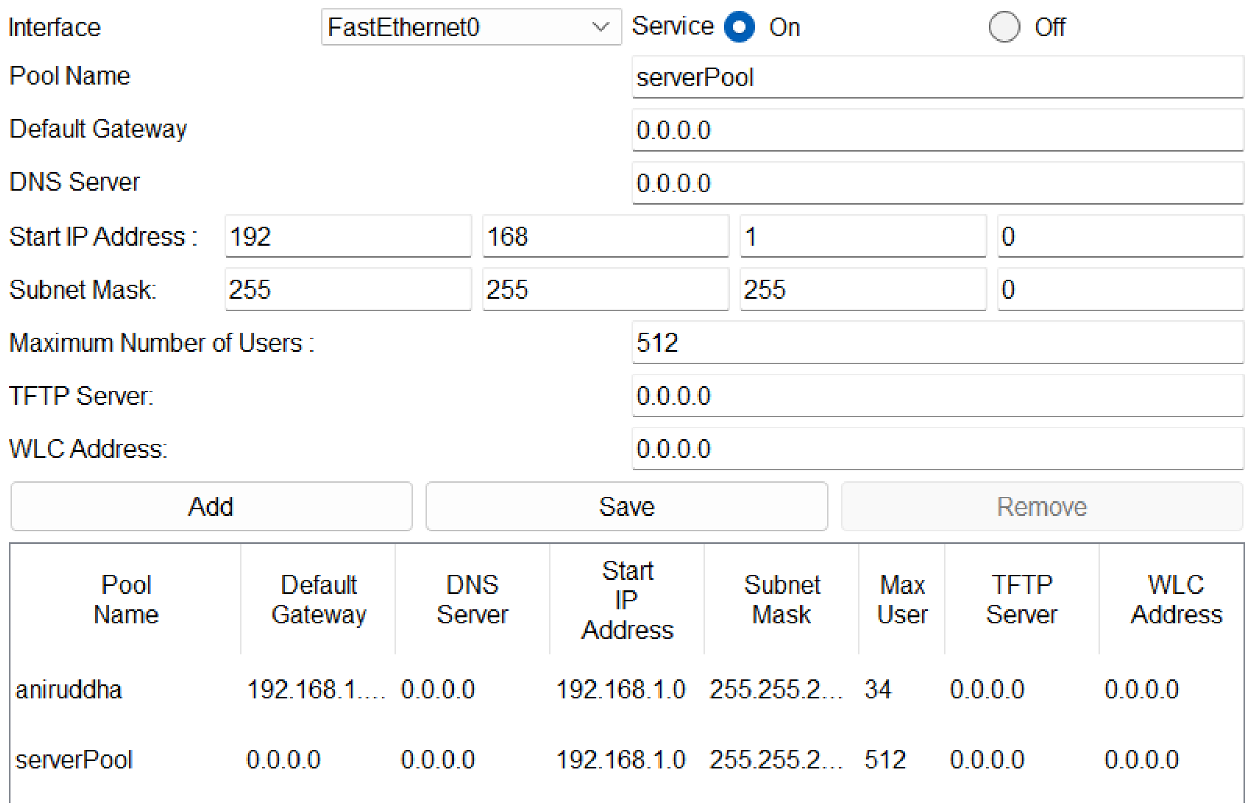
2. FTP


```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time=1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>ftp 192.168.2.2
Trying to connect...192.168.2.2
Connected to 192.168.2.2
220- Welcome to PT Ftp server
Username:aniruddha
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>put hello.txt
%Error opening c:hello.txt (No such file or directory)
ftp>put hello.txt

Writing file hello.txt to 192.168.2.2:
File transfer in progress...
```



```
Writing file hello.txt to 192.168.2.2:
File transfer in progress...

[Transfer complete - 5 bytes]

5 bytes copied in 0.154 secs (32 bytes/sec)
ftp>get hello.txt

Reading file hello.txt from 192.168.2.2:
File transfer in progress...

[Transfer complete - 5 bytes]

5 bytes copied in 0.009 secs (555 bytes/sec)
```

3. DHCP



Interface: FastEthernet0 — Service [✓ On] Off

Pool Name	serverPool
Default Gateway	0.0.0.0
DNS Server	0.0.0.0
Start IP Address :	192 . 168 . 1 . 0
Subnet Mask:	255 . 255 . 255 . 0
Maximum Number of Users :	512
TFTP Server:	0.0.0.0
WLC Address:	0.0.0.0

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max User	TFTP Server	WLC Address
aniruddha	192.168.1....	0.0.0.0	192.168.1.0	255.255.2...	34	0.0.0.0	0.0.0.0
serverPool	0.0.0.0	0.0.0.0	192.168.1.0	255.255.2...	512	0.0.0.0	0.0.0.0

Conclusion:

Experiment No.3

Objective: The objective of this lab is to introduce you to the physical aspects of networking using Cisco Packet Tracer, including devices, cables, and connections.

Prerequisites: Basic understanding of networking concepts such as devices, cables, and network topologies. Familiarity with Cisco Packet Tracer interface.

Software Required: Cisco Packet Tracer

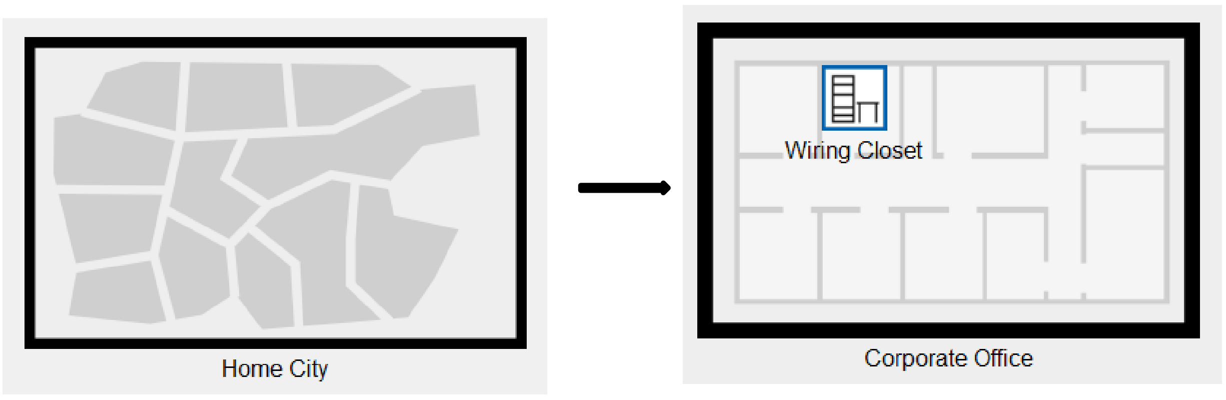
Theory:
1: Basic Device Configuration
Objective: Familiarize students with different networking devices and their physical characteristics.
Topology: Create a simple network topology with devices such as routers, switches, PCs, and servers.
Configuration: Drag and drop various devices onto the workspace. Examine the physical attributes of each device, including ports, interfaces, LEDs, and chassis.

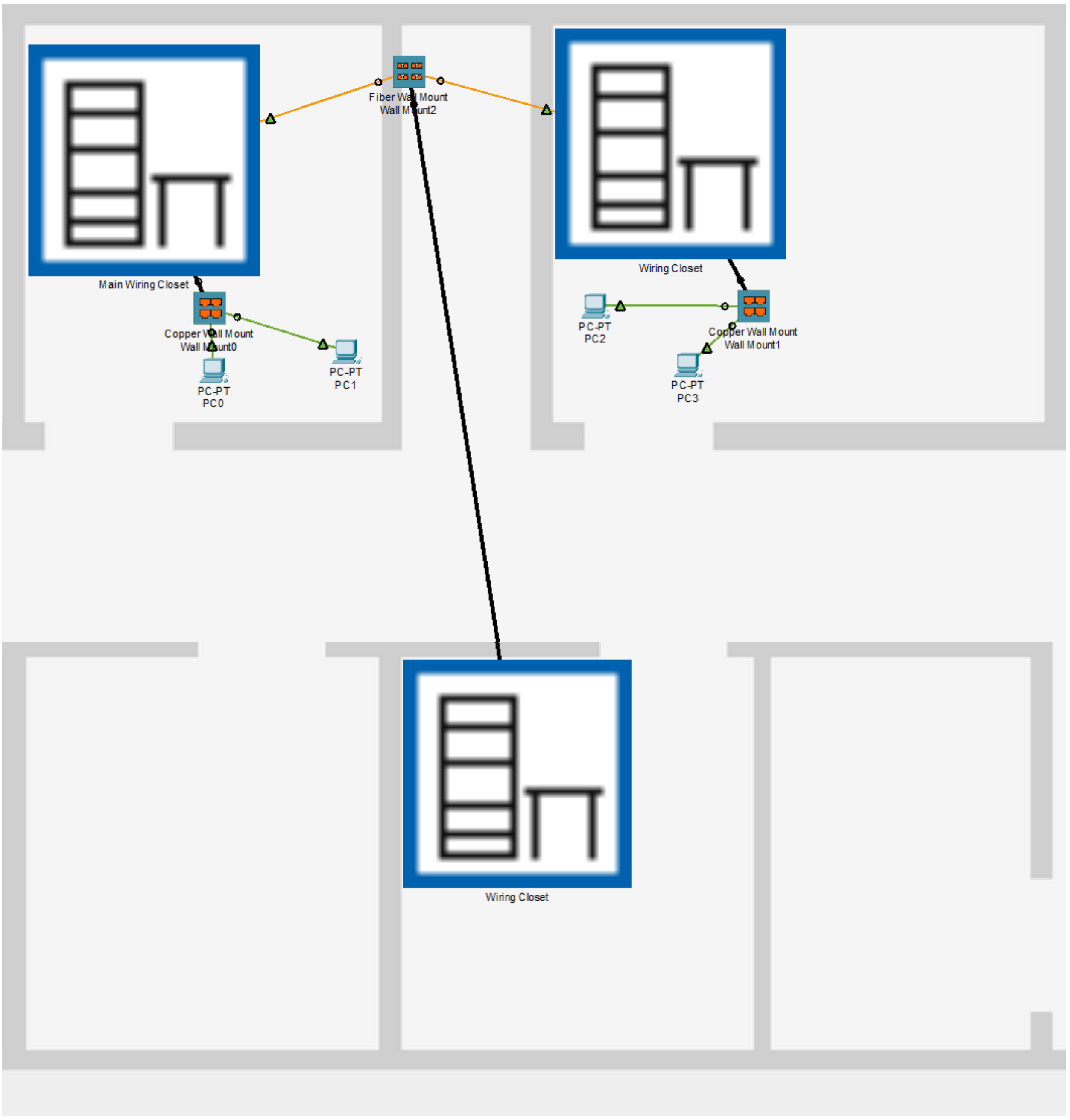
2: Cable Types and Connections
Objective: Learn about different types of cables and how to make physical connections between devices.
Topology: Expand the existing network topology with additional devices. Add cables (Ethernet, serial, console) to establish connections between devices.
Configuration: Select appropriate cable types for different connections (e.g., straight-through, crossover). Practice making physical connections by connecting devices with cables.

3: Rack View Configuration
Objective: Understand how devices are mounted in racks and organized within network cabinets.
Topology: Create a rack or network cabinet using Packet Tracer's rack view feature. Mount devices such as switches, routers, and servers in the rack.
Configuration: Arrange devices in the rack according to standard practices (e.g., rack units, cable management). Label devices and cables for easy identification.

4: Wall Mount and Patch Panel Configuration
Objective: Learn about wall-mounted equipment and the use of patch panels for efficient cable management.
Topology: Expand the existing network topology with wall-mounted devices and patch panels. Include additional switches, servers, and PCs as needed.
Configuration:
1) Wall Mount Installation: Identify locations for wall-mounted equipment such as switches and patch panels in the topology. Mount switches and patch panels securely on the walls using Packet Tracer's wall mount feature. Ensure proper alignment and spacing for easy access and cable management.
2) Patch Panel Connection: Introduce patch panels into the network topology and connect them to switches using Ethernet cables. Use appropriate patch panel ports for incoming and outgoing connections. Organize and label cables neatly within the patch panel.
3) Cable Management: Route Ethernet cables from devices to the patch panels efficiently, maintaining proper cable lengths. Utilize cable management accessories such as cable ties or Velcro straps to bundle and organize cables. Ensure cables are neatly arranged and do not obstruct access to devices or patch panels.
4) Testing and Verification: Verify connectivity between devices by testing network connections through the patch panel. Conduct cable testing to ensure proper termination and connectivity. Troubleshoot any connectivity issues and make necessary adjustments to cable connections.

Output:





Steps:

1. Accessing the Physical View:

- Open Cisco Packet Tracer.
- By default, you'll be in the Logical View, which shows a simplified layout of your network devices.
- Click the Physical View button located behind the Logical View button in the top left corner of the workspace.

2. Navigating the Physical View:

- The default view might show the Intercity container, representing a broader geographical area.
- Use the navigation panel on the right to zoom in and out, change your location (e.g., Home City, Building), and explore different containers within your network.

3. Placing Devices in Containers:

- Drag and drop network devices (switches, routers, PCs) from the Device Inventory onto the desired container within the Physical View.
- You can move existing devices in the Logical View by right-clicking them and selecting "Go to Physical View."
- Use your mouse to position the devices within the container to reflect their physical placement.

4. Switching Back to Logical View:

- Click the Logical View button at any time to return to the traditional network device layout. The physical placement of devices in the Physical View is reflected in their positions within the Logical View.

Conclusion:

Experiment No.4

Objective: The objective of this lab manual is to introduce you to dynamic and static routing protocols using Cisco Packet Tracer.

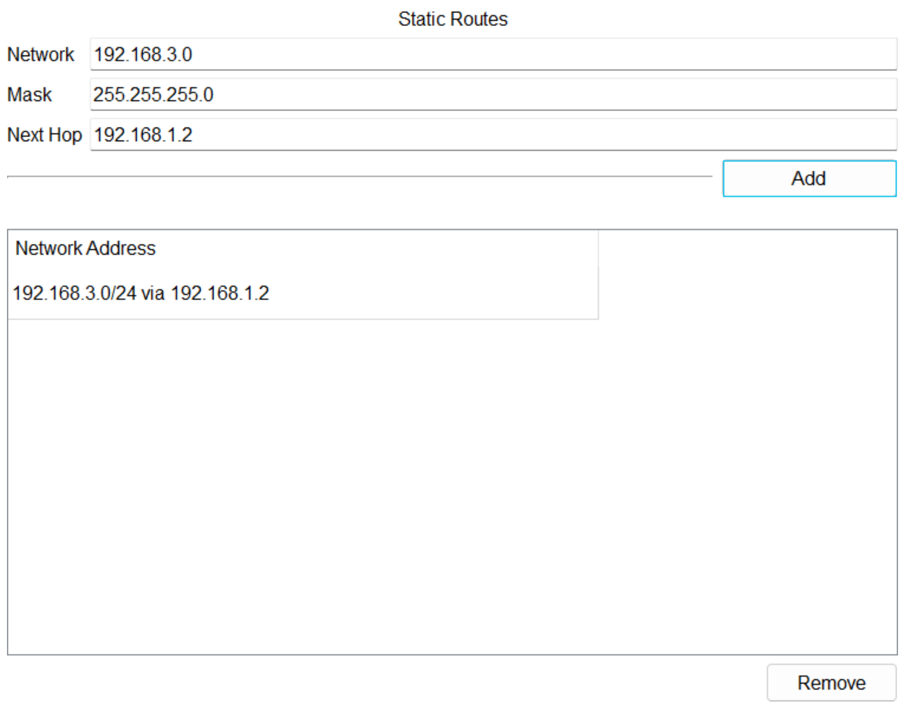
Prerequisites: Basic understanding of networking concepts such as IP addressing, subnetting, and routing. Familiarity with Cisco Packet Tracer interface.

Software Required: Cisco Packet Tracer

Theory:

1: Static Routing

- Configuration Objective: Configure static routes to establish network communication between multiple networks.
- Topology: Create a network topology with multiple routers and subnets. Ensure each network is connected to a router.
- Configuration: Configure static routes on routers. Test connectivity between devices in different networks by sending packets or using ping commands.


Static Routes

Network 192.168.3.0

Mask 255.255.255.0

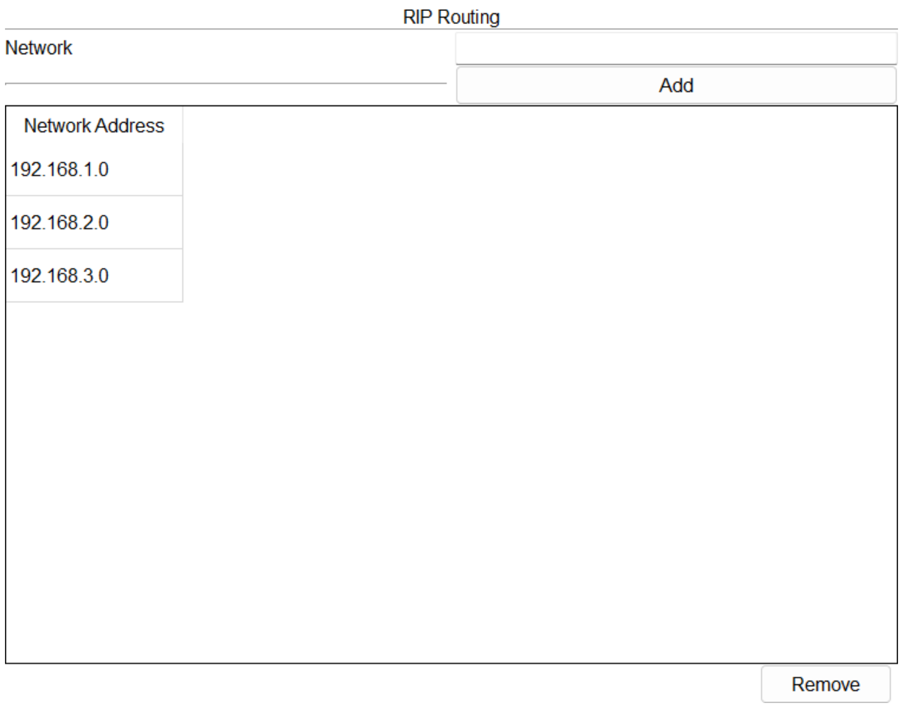
Next Hop 192.168.1.2

Network Address

192.168.3.0/24 via 192.168.1.2

2: RIP (Routing Information Protocol)

- Configuration Objective: Configure RIP routing protocol to enable dynamic routing between routers.
- Topology: Expand the existing network topology with additional routers. Remove static routes if configured in static routing.
- Configuration: Enable RIP routing protocol on routers. Configure RIP and network. Test dynamic routing by introducing new networks and verifying automatic route propagation.

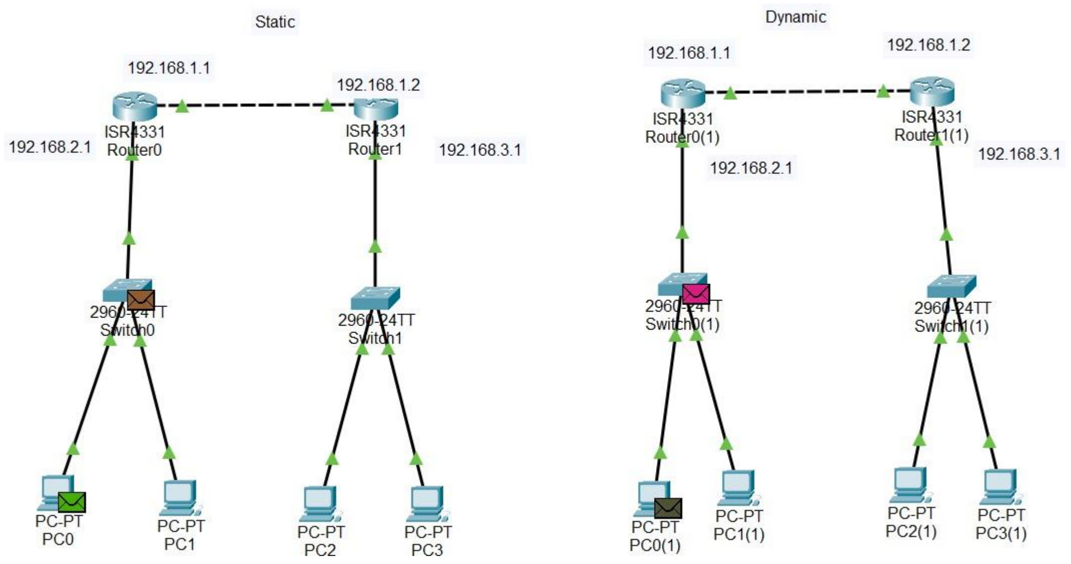


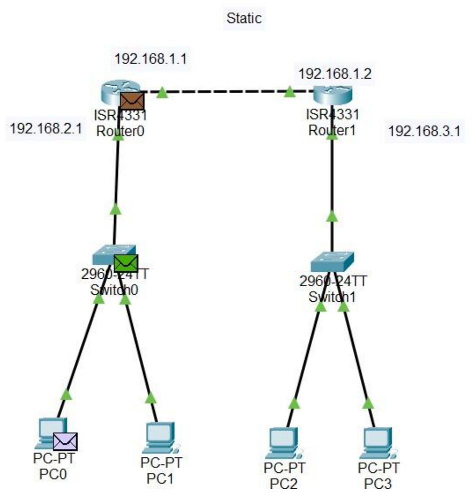
RIP Routing

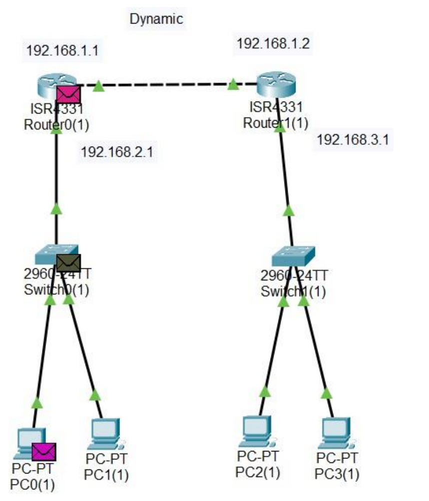
Network

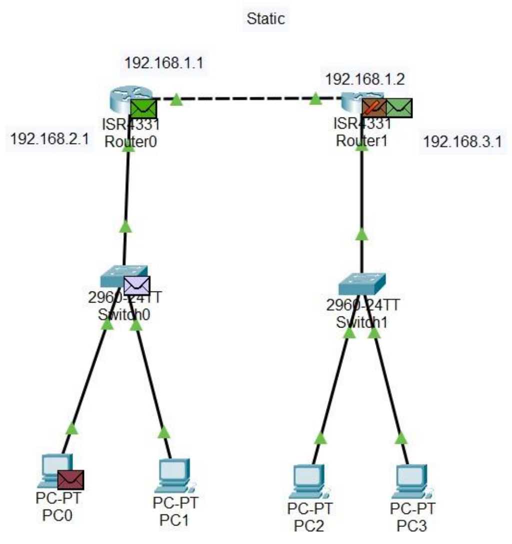
Network Address
192.168.1.0
192.168.2.0
192.168.3.0

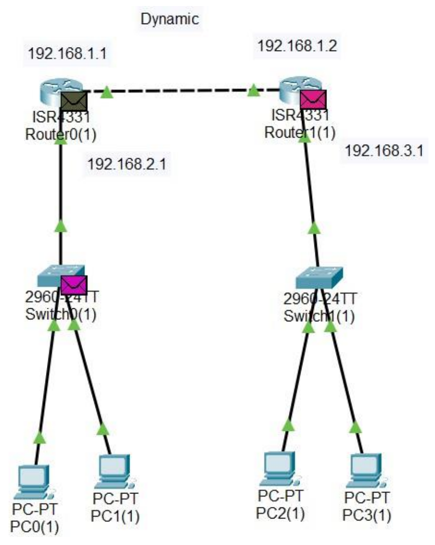
Output:

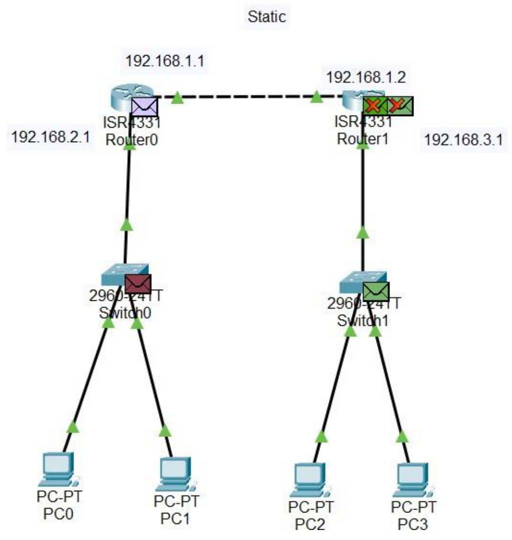


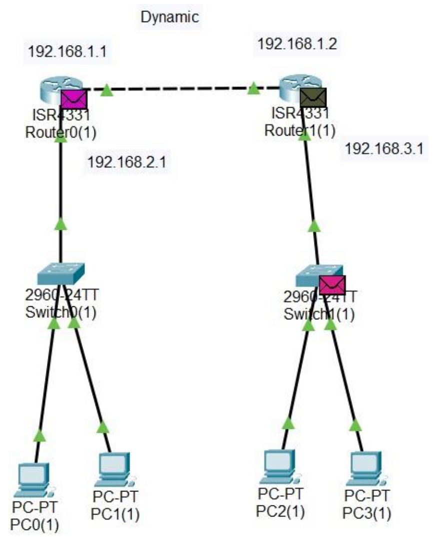

Static
192.168.1.1
192.168.1.2
192.168.2.1
ISR4331
Router0
ISR4331
Router1
192.168.3.1
2960-24TT
Switch0
2960-24TT
Switch1
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3

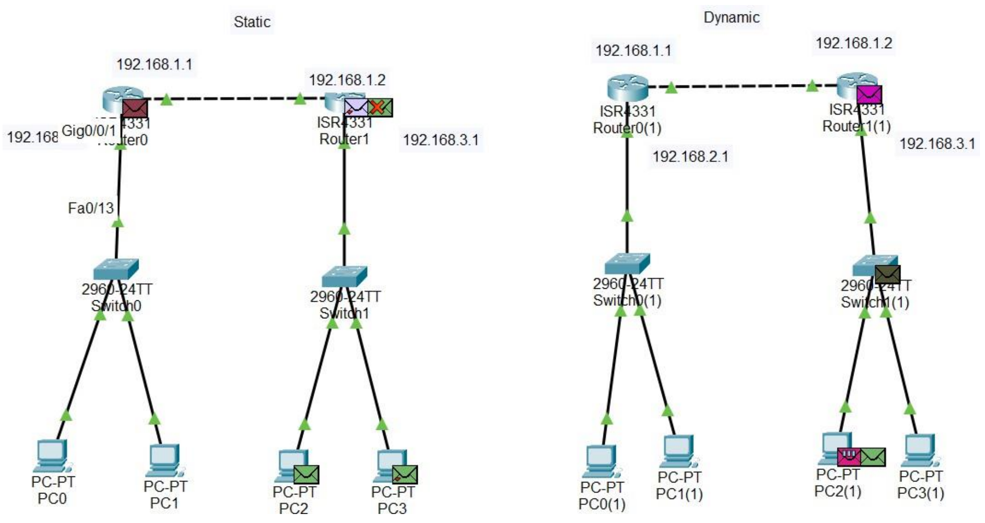

Dynamic
192.168.1.1
192.168.1.2
ISR4331
Router0(1)
ISR4331
Router1(1)
192.168.2.1
192.168.3.1
2960-24TT
Switch0(1)
2960-24TT
Switch1(1)
PC-PT
PC0(1)
PC-PT
PC1(1)
PC-PT
PC2(1)
PC-PT
PC3(1)


Static
192.168.1.1
192.168.1.2
192.168.2.1
ISR4331
Router0
ISR4331
Router1
192.168.3.1
2960-24TT
Switch0
2960-24TT
Switch1
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3


Dynamic
192.168.1.1
192.168.1.2
ISR4331
Router0(1)
ISR4331
Router1(1)
192.168.2.1
192.168.3.1
2960-24TT
Switch0(1)
2960-24TT
Switch1(1)
PC-PT
PC0(1)
PC-PT
PC1(1)
PC-PT
PC2(1)
PC-PT
PC3(1)


Static
192.168.1.1
192.168.1.2
192.168.2.1
ISR4331
Router0
ISR4331
Router1
192.168.3.1
2960-24TT
Switch0
2960-24TT
Switch1
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3


Dynamic
192.168.1.1
192.168.1.2
ISR4331
Router0(1)
ISR4331
Router1(1)
192.168.2.1
192.168.3.1
2960-24TT
Switch0(1)
2960-24TT
Switch1(1)
PC-PT
PC0(1)
PC-PT
PC1(1)
PC-PT
PC2(1)
PC-PT
PC3(1)


Static
192.168.1.1
192.168.1.2
Gig0/0/1
ISR4331
Router0
ISR4331
Router1
192.168.3.1
Fa0/13
2960-24TT
Switch0
2960-24TT
Switch1
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3
Dynamic
192.168.1.1
192.168.1.2
ISR4331
Router0(1)
ISR4331
Router1(1)
192.168.2.1
192.168.3.1
2960-24TT
Switch0(1)
2960-24TT
Switch1(1)
PC-PT
PC0(1)
PC-PT
PC1(1)
PC-PT
PC2(1)
PC-PT
PC3(1)

Conclusion:

Experiment No.5

Objective: Socket Programming and Wireshark

Software Required: Wireshark, python 3.9+, VS Code

Theory:

Wireshark

- Wireshark is a packet sniffer and analysis tool. It captures network traffic from ethernet, Bluetooth, wireless (IEEE.802.11), token ring, and frame relay connections, among others, and stores that data for offline analysis.
- Note: A "packet" is a single message from any network protocol (e.g., TCP, DNS, etc.).
- LAN traffic is in broadcast mode, meaning a single computer with Wireshark can see traffic between two other computers. To see traffic to an external site, you need to capture the packets on the local computer.
- Wireshark allows you to filter the log before the capture starts or during analysis, so you can narrow down and zero in on what you're looking for in the network trace. For example, you can set a filter to see TCP traffic between two IP addresses, or you can set it only to show you the packets sent from one computer. The filters in Wireshark are one of the primary reasons it has become the standard tool for packet analysis.

Download: http://www.wireshark.org/download.html

Socket Programming

- Sockets and the socket API are used to send messages across a network. They provide a form of inter-process communication (IPC). The network can be a logical, local network to the computer, or one that's physically connected to an external network, with its own connections to other networks.
- A network socket is an endpoint of an inter-process communication flow across a computer network. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents. Today, most communication between computers is based on the internet protocol; therefore most network sockets are internet sockets. To create a connection between machines, Python programs import the socket module, create a socket object, and call the object's methods to establish connections and send and receive data. Sockets are the endpoints of a bidirectional communications channel.

There are two basic types of communication

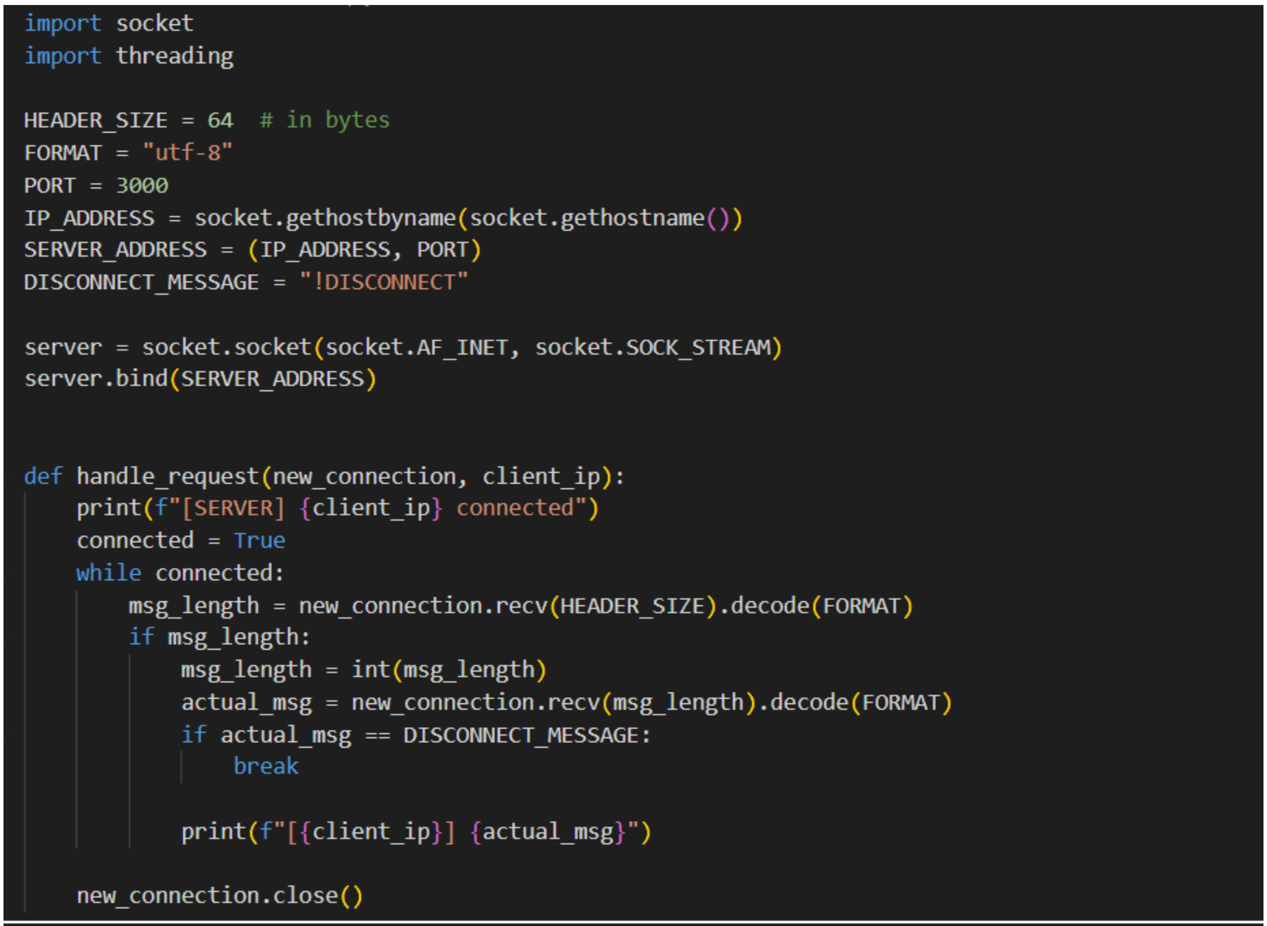
• Streams (TCP): Computers establish a connection with each other and read/write data in a continuous stream of bytes---like a file. This is the most common.

• Datagrams (UDP): Computers send discrete packets (or messages) to each other. Each packet contains a collection of bytes, but each packet is separate and self-contained.

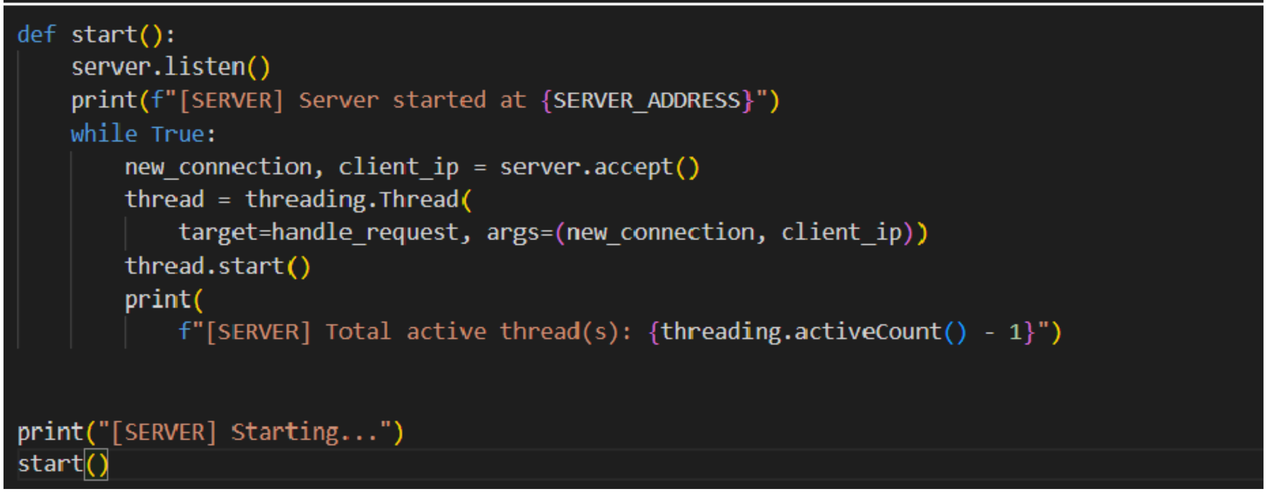
Client - Server Concept

• Each endpoint is a running program
• Servers wait for incoming connections and provide a service (e.g., web, mail, etc.)
• Clients make connections to servers

Server Code:

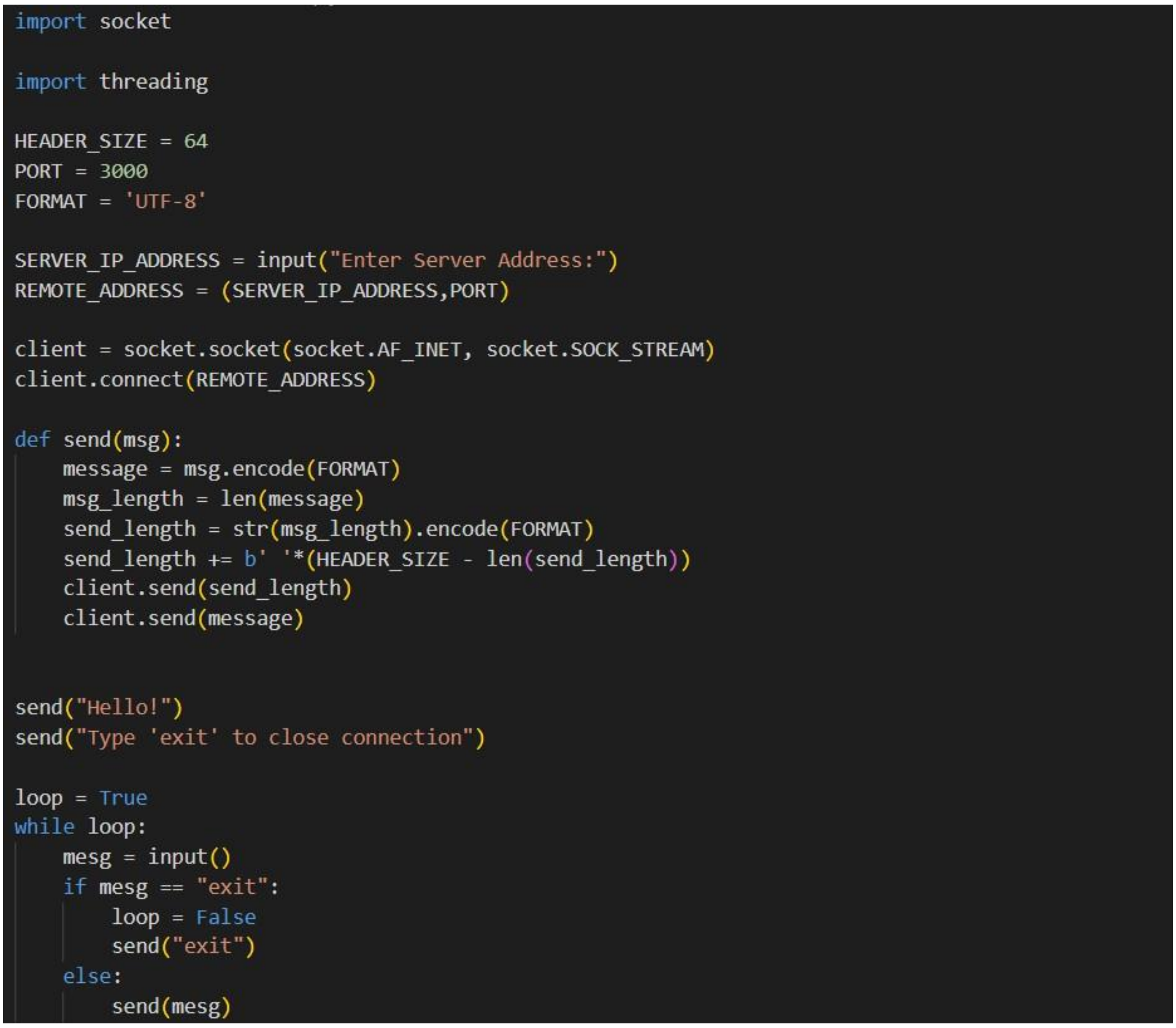


```python
import socket
import threading

HEADER_SIZE = 64  # in bytes
FORMAT = "utf-8"
PORT = 3000
IP_ADDRESS = socket.gethostbyname(socket.gethostname())
SERVER_ADDRESS = (IP_ADDRESS, PORT)
DISCONNECT_MESSAGE = "!DISCONNECT"

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(SERVER_ADDRESS)


def handle_request(new_connection, client_ip):
    print(f"[SERVER] {client_ip} connected")
    connected = True
    while connected:
        msg_length = new_connection.recv(HEADER_SIZE).decode(FORMAT)
        if msg_length:
            msg_length = int(msg_length)
            actual_msg = new_connection.recv(msg_length).decode(FORMAT)
            if actual_msg == DISCONNECT_MESSAGE:
                break

            print(f"[{client_ip}] {actual_msg}")

    new_connection.close()
```



```python
def start():
    server.listen()
    print(f"[SERVER] Server started at {SERVER_ADDRESS}")
    while True:
        new_connection, client_ip = server.accept()
        thread = threading.Thread(
            target=handle_request, args=(new_connection, client_ip))
        thread.start()
        print(
            f"[SERVER] Total active thread(s): {threading.activeCount() - 1}")


print("[SERVER] Starting...")
start()
```

Client Code:



```
import socket

import threading

HEADER_SIZE = 64
PORT = 3000
FORMAT = 'UTF-8'

SERVER_IP_ADDRESS = input("Enter Server Address:")
REMOTE_ADDRESS = (SERVER_IP_ADDRESS,PORT)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(REMOTE_ADDRESS)

def send(msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' '*(HEADER_SIZE - len(send_length))
    client.send(send_length)
    client.send(message)


send("Hello!")
send("Type 'exit' to close connection")

loop = True
while loop:
    mesg = input()
    if mesg == "exit":
        loop = False
        send("exit")
    else:
        send(mesg)
```

Network/Packets Analysis

- Wireshark shows you three different panes for inspecting packet data. The Packet List, the top pane, lists all the packets in the capture. When you click on a packet, the other two panes change to show you the details about the selected packet. You can also tell if the packet is part of a conversation. Here are details about each column in the top pane:
- No.: This is the number order of the packet captured. The bracket indicates that this packet is part of a conversation.
- Time: This column shows how long after you started the capture this particular packet was captured. You can change this value in the Settings menu to display a different option.

➢ Source: This is the address of the system that sent the packet.
➢ Destination: This is the address of the packet destination.
➢ Protocol: This is the type of packet. For example: TCP, DNS, DHCPv6, or ARP.

- Length: This column shows you the packet’s length, measured in bytes.
- Info: This column shows you more information about the packet contents, which will vary depending on the type of packet.

Flags Associated with the packets.

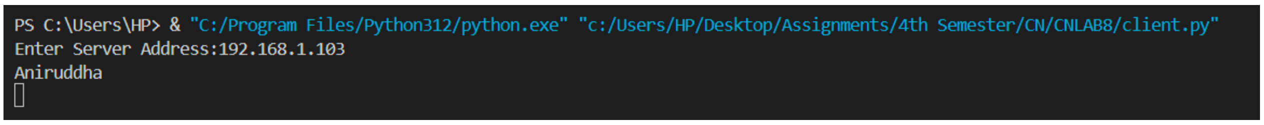
1. (SYN): Synchronize
2. This flag is set to 1 in the initial packet sent from the client to the server when establishing a TCP connection. It's used to initiate a connection and synchronize sequence numbers between the two endpoints.
3. (FIN): Finish
4. When set to 1, this flag indicates that the sender has finished sending data and wants to close the connection. It's part of the connection termination process.
5. (ACK): Acknowledgment
6. This flag signifies that the ACK number in the TCP header is acknowledging the data received from the other endpoint. It's used in the establishment, maintenance, and termination phases of a TCP connection.
7. (PSH): Push
8. When set, the PSH flag tells the receiving endpoint to push the data to the receiving application immediately without waiting for the buffer to fill up. This is used to ensure that data is processed quickly.
9. (URG): Urgent
10. The URG flag indicates that the segment contains urgent data, and it should be processed immediately. The location of the urgent data within the segment is defined by the urgent pointer field of the TCP header.
11. (RST): Reset
12. This flag is used to forcibly abort a connection in response to an error or to reject an invalid segment. It can also be used to refuse a connection request.
13. Three-way Handshake
14. Step 1 (SYN): In the first step, the client wants to establish a connection with a server, so it sends a segment with SYN (Synchronize Sequence Number) which informs the server that the client is likely to start communication and with what sequence number it starts segments with
15. Step 2 (SYN + ACK): Server responds to the client request with SYN-ACK signal bits set. Acknowledgement (ACK) signifies the response of the segment it received, and SYN signifies with what sequence number it is likely to start the segments with
16. Step 3 (ACK): In the final part client acknowledges the response of the server and they both establish a reliable connection with which they will start the actual data transfer.

Steps:

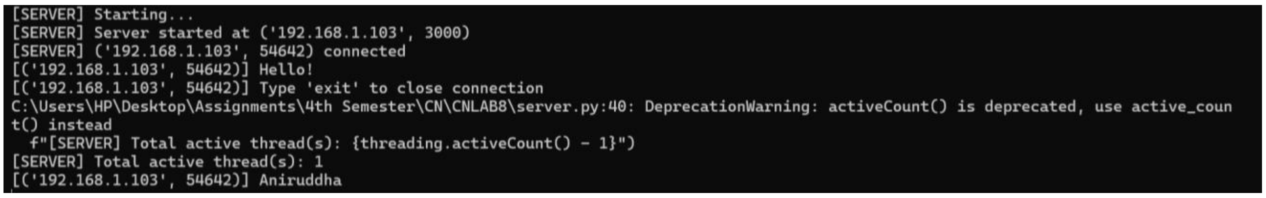
- Run the server.py on a machine.
- Now run the client.py on another machine and enter the IP address of the server machine and make sure wireshark is running and capturing packets in the background.
- After establishing the connection send 3-4 to server
- Now go to wireshark and stop capturing packets, then apply the display filter "ip.dst=='server ip'" to find the packets which where send to server from client
- After finding all the packets which were sent to server, analyze the packets, find the flags associated with packets and the original message sent.

Output:

Client Side



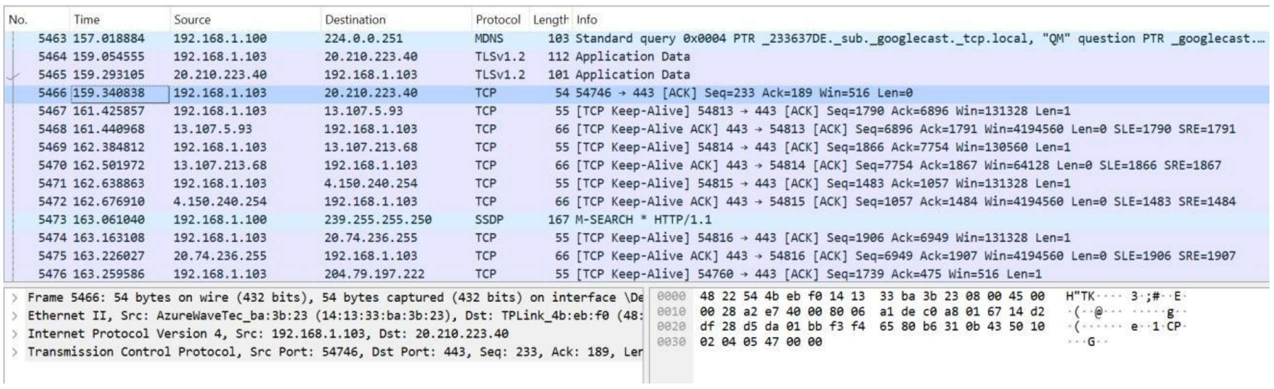
```
PS C:\Users\HP> & "C:/Program Files/Python312/python.exe" "c:/Users/HP/Desktop/Assignments/4th Semester/CN/CNLAB8/client.py"
Enter Server Address:192.168.1.103
Aniruddha
```



```
[SERVER] Starting...
[SERVER] Server started at ('192.168.1.103', 3000)
[SERVER] ('192.168.1.103', 54642) connected
[('192.168.1.103', 54642)] Hello!
[('192.168.1.103', 54642)] Type 'exit' to close connection
C:\Users\HP\Desktop\Assignments\4th Semester\CN\CNLAB8\server.py:40: DeprecationWarning: activeCount() is deprecated, use active_coun
t() instead
  f"[SERVER] Total active thread(s): {threading.activeCount() - 1}")
[SERVER] Total active thread(s): 1
[('192.168.1.103', 54642)] Aniruddha
```

Server Side

Wireshark



No.	Time	Source	Destination	Protocol	Length	Info
5463	157.018884	192.168.1.100	224.0.0.251	MDNS	103	Standard query 0x0004 PTR _233637DE._sub._googlecast._tcp.local, "QM" question PTR _googlecast…
5464	159.054555	192.168.1.103	20.210.223.40	TLSv1.2	112	Application Data
5465	159.293105	20.210.223.40	192.168.1.103	TLSv1.2	101	Application Data
5466	159.340838	192.168.1.103	20.210.223.40	TCP	54	54746 → 443 [ACK] Seq=233 Ack=189 Win=516 Len=0
5467	161.425857	192.168.1.103	13.107.5.93	TCP	55	[TCP Keep-Alive] 54813 → 443 [ACK] Seq=1790 Ack=6896 Win=131328 Len=1
5468	161.440968	13.107.5.93	192.168.1.103	TCP	66	[TCP Keep-Alive ACK] 443 → 54813 [ACK] Seq=6896 Ack=1791 Win=4194560 Len=0 SLE=1790 SRE=1791
5469	162.384812	192.168.1.103	13.107.213.68	TCP	55	[TCP Keep-Alive] 54814 → 443 [ACK] Seq=1866 Ack=7754 Win=130560 Len=1
5470	162.501972	13.107.213.68	192.168.1.103	TCP	66	[TCP Keep-Alive ACK] 443 → 54814 [ACK] Seq=7754 Ack=1867 Win=64128 Len=0 SLE=1866 SRE=1867
5471	162.638863	192.168.1.103	4.150.240.254	TCP	55	[TCP Keep-Alive] 54815 → 443 [ACK] Seq=1483 Ack=1057 Win=131328 Len=1
5472	162.676910	4.150.240.254	192.168.1.103	TCP	66	[TCP Keep-Alive ACK] 443 → 54815 [ACK] Seq=1057 Ack=1484 Win=4194560 Len=0 SLE=1483 SRE=1484
5473	163.061040	192.168.1.100	239.255.255.250	SSDP	167	M-SEARCH * HTTP/1.1
5474	163.163108	192.168.1.103	20.74.236.255	TCP	55	[TCP Keep-Alive] 54816 → 443 [ACK] Seq=1906 Ack=6949 Win=131328 Len=1
5475	163.226027	20.74.236.255	192.168.1.103	TCP	66	[TCP Keep-Alive ACK] 443 → 54816 [ACK] Seq=6949 Ack=1907 Win=4194560 Len=0 SLE=1906 SRE=1907
5476	163.259586	192.168.1.103	204.79.197.222	TCP	55	[TCP Keep-Alive] 54760 → 443 [ACK] Seq=1739 Ack=475 Win=516 Len=1

```
> Frame 5466: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \De
> Ethernet II, Src: AzureWaveTec_ba:3b:23 (14:13:33:ba:3b:23), Dst: TPLink_4b:eb:f0 (48:
> Internet Protocol Version 4, Src: 192.168.1.103, Dst: 20.210.223.40
> Transmission Control Protocol, Src Port: 54746, Dst Port: 443, Seq: 233, Ack: 189, Ler
```

```
0000  48 22 54 4b eb f0 14 13  33 ba 3b 23 08 00 45 00   H"TK····3·;#··E·
0010  00 28 a2 e7 40 00 80 06  a1 de c0 a8 01 67 14 d2   ·(··@·······g··
0020  df 28 d5 da 01 bb f3 f4  65 80 b6 31 0b 43 50 10   ·(······e··1·CP·
0030  02 04 05 47 00 00                                  ···G··
```

Conclusion: