CSA0713:-COMPUTER NETWORKS FOR CLOUD STORAGE

PRACTICAL OBSERVATION

SHAIK SAMEER PASHA
(192425383)

LIST OF EXPERIMENTS

1. Configuration of Network Devices using Packet Tracer tools (Hub, Switch, Ethernet, Broadcast).
2. Design and Configuration of Star Topologies using Packet Tracer.
3. Design and Configuration of BUS Topologies using Packet Tracer.
4. Design and Configuration of RING Topologies using Packet Tracer.
5. Design and Configuration of Mesh Topologies using Packet Tracer.
6. Design and Configuration of Tree Topologies using Packet Tracer.
7. Design and Configuration of Hybrid Topologies using Packet Tracer.
8. Data Link Layer Traffic Simulation using Packet Tracer Analysis of ARP.
9. Data Link Layer Traffic Simulation using Packet Tracer Analysis of LLDP.
10. Data Link Layer Traffic Simulation using Packet Tracer Analysis of CSMA/CD & CSMA/CA.
11. Designing two different networks with Static Routing techniques using Packet Tracer.
12. Designing two different networks with Dynamic Routing techniques (RIP & OSPF) using Packet Tracer.
13. Design the Functionalities and Exploration of TCP using Packet Tracer.
14. Design the Functionalities and Exploration of UDP using Packet Tracer.
15. Design the network model for Subnetting – Class C Addressing using Packet Tracer.
16. Simulating X, Y, Z Company Network Design and simulating using Packet Tracer.
17. Configuration of DHCP (dynamic host configuration protocol) in packet Tracer.

18. Configuration of firewall in packet tracer.

19. Make a Computer Lab to transfer a message from one node to another to design and simulate using Cisco Packet Tracer.

20. Simulate a Multimedia Network in Cisco Packet Tracer.

21. IoT based smart home applications.

22. Implementation of IoT based smart gardening.

23. Implementation of IoT devices in networking.

24. IOT Based Smart building using WPA Security & Radius Server.

25. Transport layer protocol header analysis using Wire shark- TCP

26. Transport layer protocol header analysis using Wire shark- UDP

27. Network layer protocol header analysis using Wire shark – SMTP

28. Network layer protocol header analysis using Wireshark - ICMP

29. Network layer protocol header analysis using Wire shark – ARP

30. Network layer protocol header analysis using Wire shark – HTTP

31. Identify and monitor the IP, network address, Trace the router information, how to take remote system and check the node connection in network

32. Demonstration of PING operation using ICMP in Wireshark

33. Implementation of Bit stuffing mechanism using C.

34. Implementation of server – client using TCP socket programming.

35. Implementation of server – client using UDP socket programming.

Date:

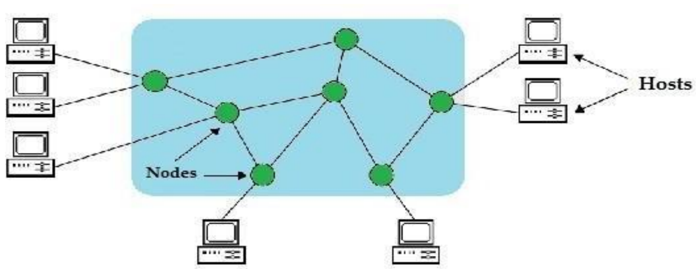
EXPERIMENT-1 CONFIGURATION OF NETWORK COMPONENTS

Aim: To Study the following Network Devices in Detail

- PC
- Server
- Repeater
- Hub
- Switch
- Bridge
- Router
- Gateway
- Transmission medium

Apparatus (Software): CISCO Packet tracer.

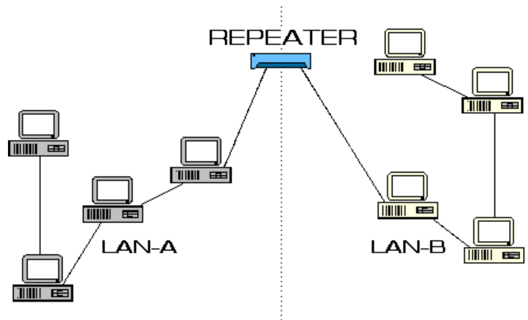
1. Node: In a communications **network**, a **network node** is a connection point that can receive, create, store or send data along distributed **network** routes.



Computer Network

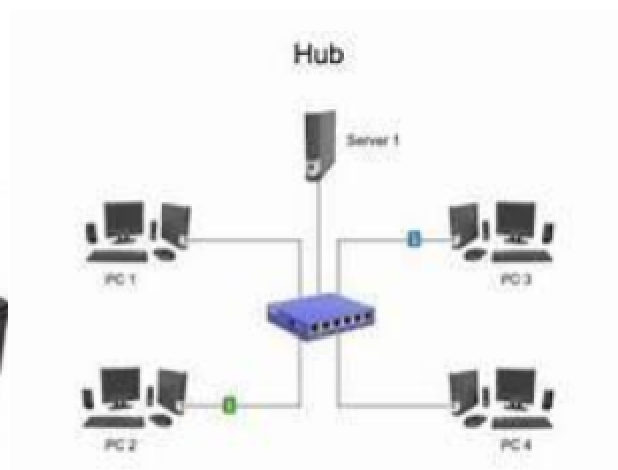
2. **Repeater:** Functioning at Physical Layer.

A **repeater** is an electronic device that receives a signal and retransmits it at a higher level and/or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances.

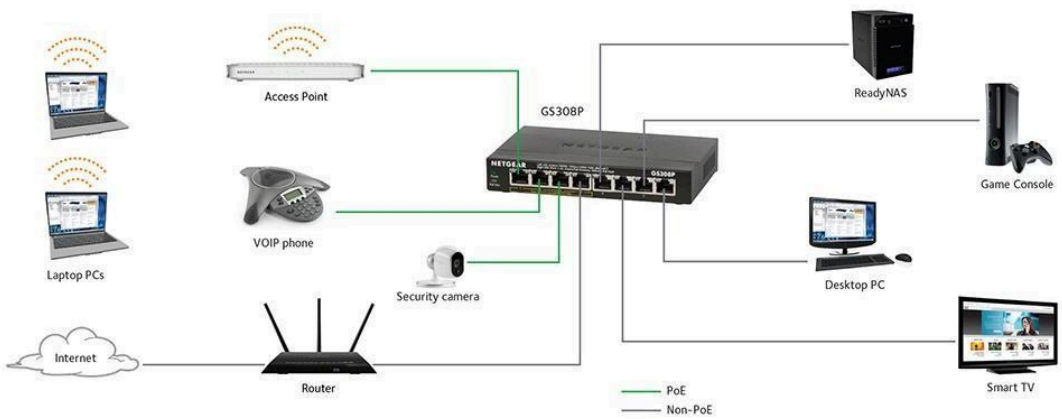


3. Hub: Ethernet hub, **active hub**, **network hub**, **repeater hub**

Hub or concentrator is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and making them act as a single network segment. Hubs work at the physical layer (layer 1) of the OSI model. The device is a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision.

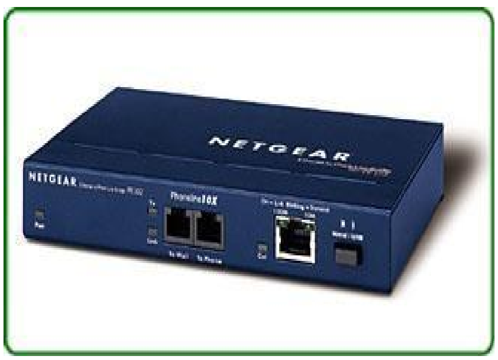



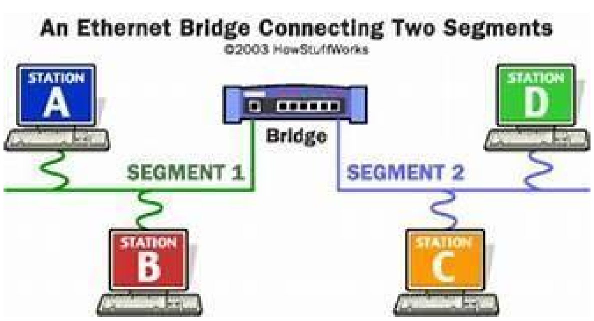
4. **Switch:** A **network switch** or **switching hub** is a computer networking device that connects network segments. The term commonly refers to a network bridge that processes and routes data at the data link layer (layer 2) of the OSI model. Switches that additionally process data at the network layer (layer 3 and above) are often referred to as Layer 3 switches or multilayer switches.


Access Point
GS308P
ReadyNAS
Game Console
VOIP phone
Laptop PCs
Security camera
Desktop PC
Internet
Router
Smart TV
PoE
Non-PoE

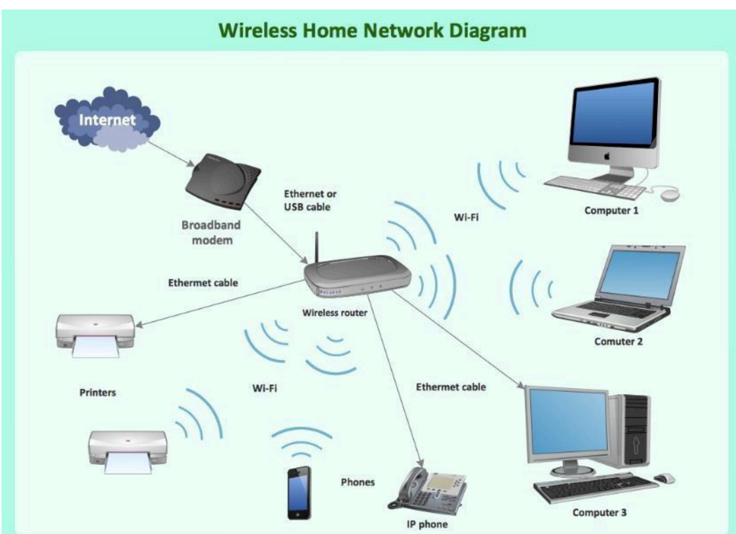

Switch

5. **Bridge:** A **network bridge** connects multiple network segments at the data link layer (Layer 2) of the OSI model. In Ethernet networks, the term bridge formally means a device that behaves according to the IEEE 802.1D standard. A bridge and switch are very much alike; a switch being a bridge with numerous ports. Switch or Layer 2 switch is often used interchangeably with bridge. Bridges can analyze incoming data packets to determine if the bridge is able to send the given packet to another segment of the network.





6. **Router:** A **router** is an electronic device that interconnects two or more computer networks, and selectively interchanges packets of data between them. Each data packet contains address information that a router can use to determine if the source and destination are on the same network, or if the data packet must be transferred from one network to another. The multiple routers are used in a large collection of interconnected networks, the routers exchange information about target system addresses, so that each router can build up a table showing the preferred paths between any two systems on the interconnected networks.



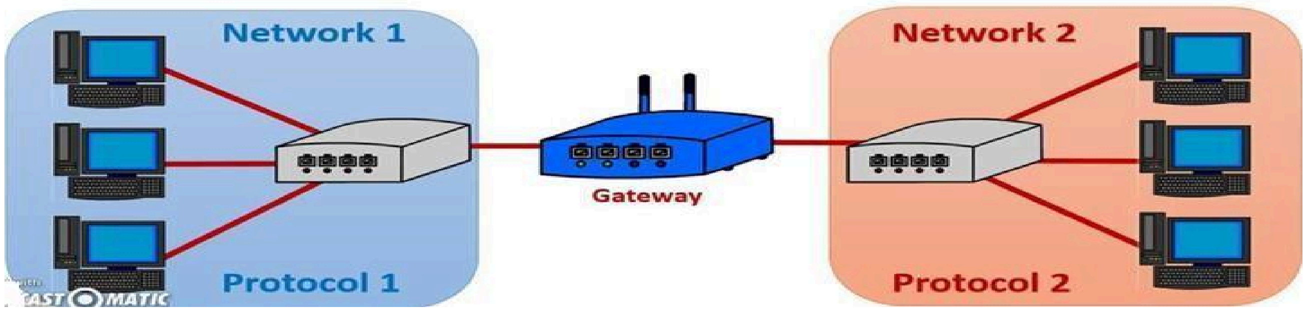
7. **Gate Way:** In a communication network, a network node equipped for interfacing with another network that uses different protocols. A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks.

- A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversions.

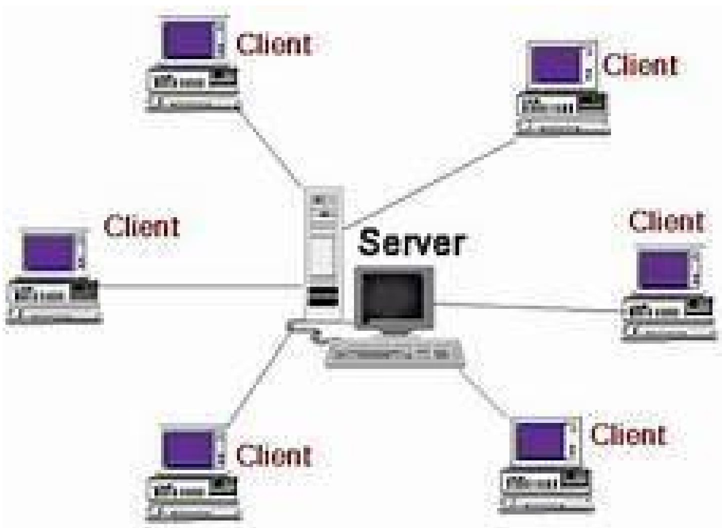
Hardware Components used in Communication Systems

Gateway

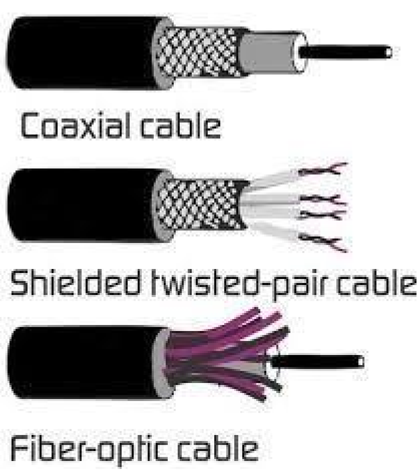
A gateway is required to connect a network with other types of networks that are running different protocols.

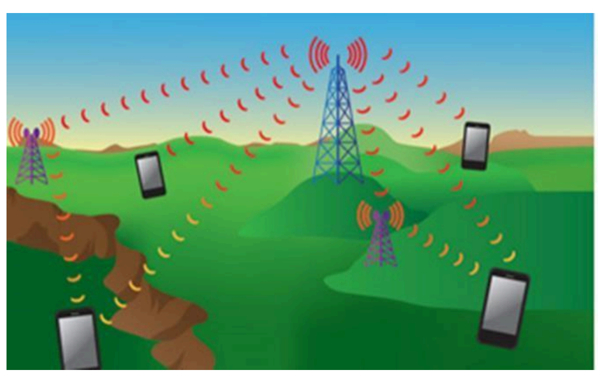


8. Server: A server is a type of computer or device on a network that manages network resources. Servers are often dedicated, meaning that they perform no other tasks besides their server tasks. On multiprocessing operating systems, however, a single computer can execute several programs at once. A server in this case could refer to the program that is managing resources rather than the entire computer.



9. **Transmission media**: The medium through which the signals travel from one device to another. These are classified as guided and unguided. Guided media are those that provide a conduit from one device to another. Eg. Twisted pair, coaxial cable etc. Unguided media transport signals without using physical cables. Eg. Air.





Result: Thus the network components are studied in detail.

Date:

EXPERIMENT-2

IMPLEMENTATION OF STAR TOPOLOGY USING PACKET TRACER

Aim: To Implement a star topology using packet tracer and hence to transmit data between the devices connected using star topology.

Software/Apparatus required: Packet Tracer/End devices, bridge, connectors.

Steps for building topology:

Step 1: Start Packet Tracer

Step 2: Choosing Devices and Connections Step

3: Building the Topology – Adding Hosts

Single click on the **End Devices**.

Single click on the **Generic** host. Move

the cursor into topology area.

Single click in the topology area and it copies the device.

Step 4: Building the Topology – Connecting the Hosts to Switches

Select a switch, by clicking once on **Switches** and once on a **2950-24** switch. Add

the switch by moving the plus sign "+"

Step 5: Connect PCs to switch by first choosing Connections

Click once on the **Copper Straight-through** cable Click

once on **PC2**

Choose **Fast Ethernet** Drag

the cursor to **Switch0** Click

once on **Switch0**

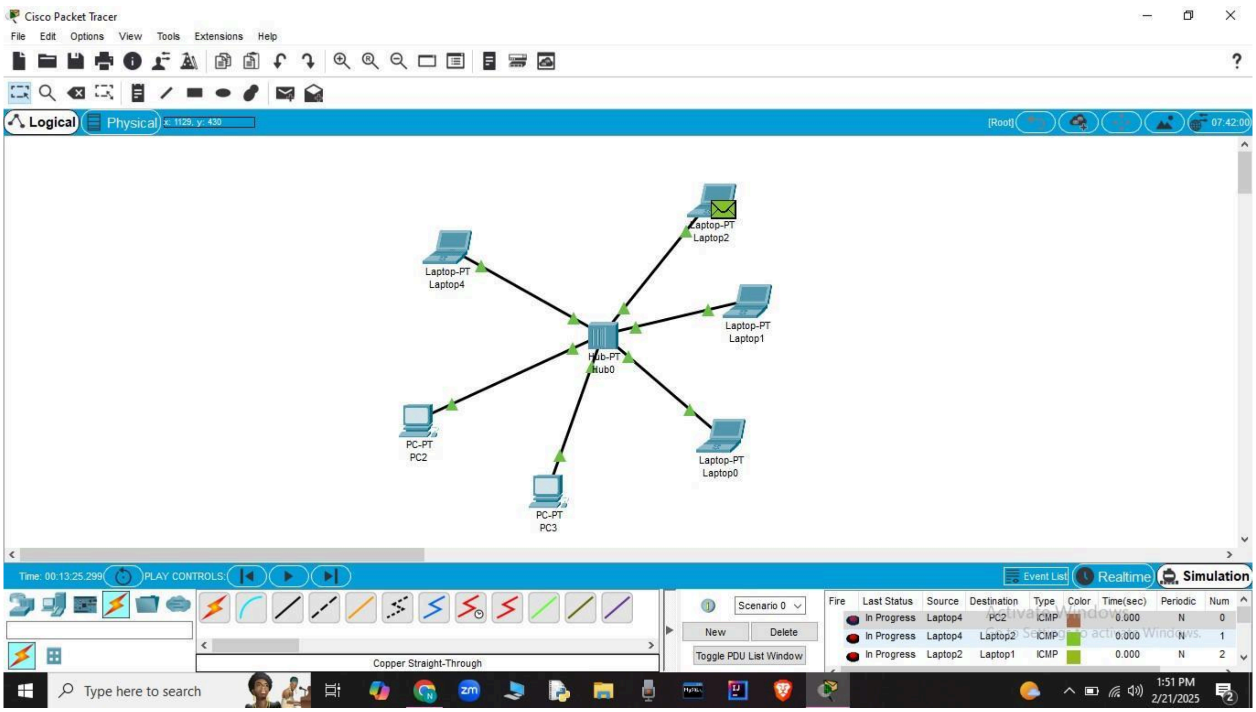
Notice the green link lights on **PC** Ethernet NIC and amber light **Switch port**. The switch port is temporarily not forwarding frames, while it goes through the stages for the Spanning Tree Protocol (STP) process. After about 30 seconds the amber light will change to green indicating that the port has entered the forwarding stage. Frames can now forwarded out the switch port.

Step 6: Configuring IP Addresses and Subnet Masks on the Hosts

To start communication between the hosts IP Addresses and Subnet Masks had to be Configured on the devices. Click once on PC0. Choose the Config tab and click on FastEthernet0. Type the IP address in its field. Click on the subnet mask it will be generated automatically.

Step 7: To confirm Data transfer between the devices

Click on the node. Select desktop option and then command prompt. Once the window pops up, ping the IP address of the device to which node0 is connected. Ping statistics will be displayed.



Result: Thus the Star topology is implemented with Packet Tracer simulation Tool.

Date:

EXPERIMENT-3

IMPLEMENTATION OF BUS TOPOLOGY USING PACKET TRACER

Aim: To Implement a Bus topology using packet tracer and hence to transmit data between the devices connected using Bus topology.

Software / Apparatus required: Packet Tracer / End devices, Hubs, connectors.

Steps for building topology:

Step 1: Start Packet Tracer

Step 2: Choosing Devices and Connections Step

3: Building the Topology – Adding Hosts

Single click on the **End Devices**.

Single click on the **Generic** host. Move

the cursor into topology area.

Single click in the topology area and it copies the device.

Step 4: Building the Topology – Connecting the Hosts to Switches

Select a switch, by clicking once on **Switches** and once on a **2950-24** switch. Add the switch by moving the plus sign "+"

Step 5: Connect PCs to switch by first choosing connections

Click once on the **Copper Straight-through** cable Click

once on **PC2**

Choose **Fast Ethernet** Drag

the cursor to **Switch0** Click

once on **Switch0**

Notice the green link lights on **PC** Ethernet NIC and amber light **Switch port**. The switch port is temporarily not forwarding frames, while it goes through the stages for the Spanning Tree Protocol (STP) process. After about 30 seconds the amber light will change to green indicating that the port has entered the forwarding stage. Frames can now forward out the switch port.

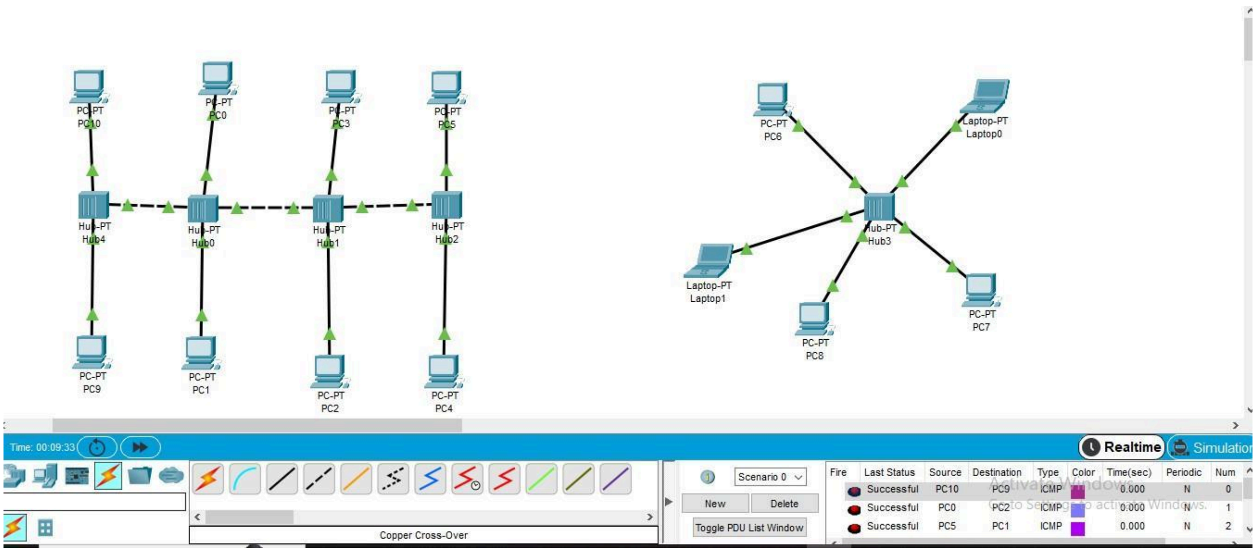
Step 6: Configuring IP Addresses and Subnet Masks on the Hosts

To start communication between the hosts IP Addresses and Subnet Masks had to be configured on the devices. Click once on PC0. Choose the Config tab and click on FastEthernet0. Type the IP address in its field. Click on the subnet mask it will be generated

automatically.

Step 7: To confirm Data transfer between the devices

Click on the node. Select desktop option and then command prompt. Once the window pops up, ping the IP address of the device to which node0 is connected. Ping statistics will be displayed.



Result: Thus the Bus topology is implemented with Packet Tracer simulation Tool.

Date:

EXPERIMENT-4:IMPLEMENTATION OF RING TOPOLOGY USING PACKET TRACER

Aim: To Implement a Ring topology using packet tracer and hence to transmit data between the devices connected using Ring topology.

Software / Apparatus required: Packet Tracer / End devices, Hubs, Connectors.

Steps for building topology:

Step 1: Start Packet Tracer

Step 2: Choosing Devices and Connections Step 3: Building the Topology – Adding Hosts

Single click on the **End Devices**.

Single click on the **Generic** host. Move the cursor into topology area.

Single click in the topology area and it copies the device.

Step 4: Building the Topology – Connecting the Hosts to Switches

Select a switch, by clicking once on **Switches** and once on a **2950-24** switch. Add the switch by moving the plus sign "+"

Step 5: Connect PCs to switch by first choosing connections

Click once on the **Copper Straight-through** cable Click once on **PC2**

Choose **Fast Ethernet** Drag the cursor to **Switch0** Click once on **Switch0**

Notice the green link lights on **PC** Ethernet NIC and amber light **Switch port**. The switch port is temporarily not forwarding frames, while it goes through the stages for the Spanning Tree Protocol (STP) process. After about 30 seconds the amber light will change to green indicating that the port has entered the forwarding stage. Frames can now forward out the switch port.

Step 6: Configuring IP Addresses and Subnet Masks on the Hosts

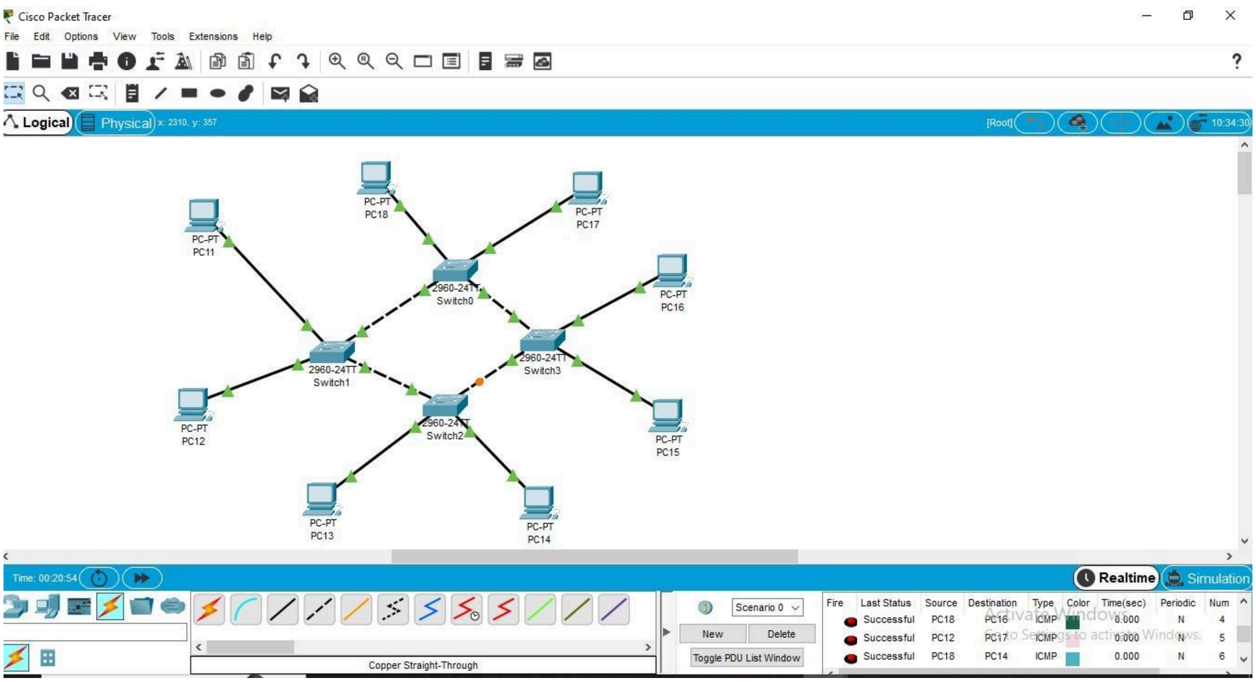
To start communication between the hosts IP Addresses and Subnet Masks had to be configured on the devices. Click once on PC0. Choose the Config tab and click on

FastEthernet0. Type the IP address in its field. Click on the subnet mask it will be generated

automatically.

Step 7: To confirm Data transfer between the devices

Click on the node. Select desktop option and then command prompt. Once the window pops up, ping the IP address of the device to which node0 is connected. Ping statistics will be displayed.



Result: Thus the Ring topology is implemented with Packet Tracer simulation Tool.

Date:

EXPERIMENT-5

IMPLEMENTATION OF MESH TOPOLOGY USING PACKET TRACER

Aim: To Implement a Mesh topology using packet tracer and hence to transmit data between the devices connected using Mesh topology.

Software / Apparatus required: Packet Tracer / End devices, Hubs, Connectors.

Steps for building topology:

Step 1: Start Packet Tracer

Step 2: Choosing Devices and Connections Step

3: Building the Topology – Adding Hosts

Single click on the **End Devices**.

Single click on the **Generic** host. Move

the cursor into topology area.

Single click in the topology area and it copies the device.

Step 4: Building the Topology – Connecting the Hosts to Switches

Select a switch, by clicking once on **Switches** and once on a **2950-24** switch. Add the switch by moving the plus sign "+"

Step 5: Connect PCs to switch by first choosing connections

Click once on the **Copper Straight-through** cable Click

once on **PC2**

Choose **Fast Ethernet** Drag

the cursor to **Switch0** Click

once on **Switch0**

Notice the green link lights on **PC** Ethernet NIC and amber light **Switch port**. The switch port is temporarily not forwarding frames, while it goes through the stages for the Spanning Tree Protocol (STP) process. After about 30 seconds the amber light will change to green indicating that the port has entered the forwarding stage. Frames can now forward out the switch port.

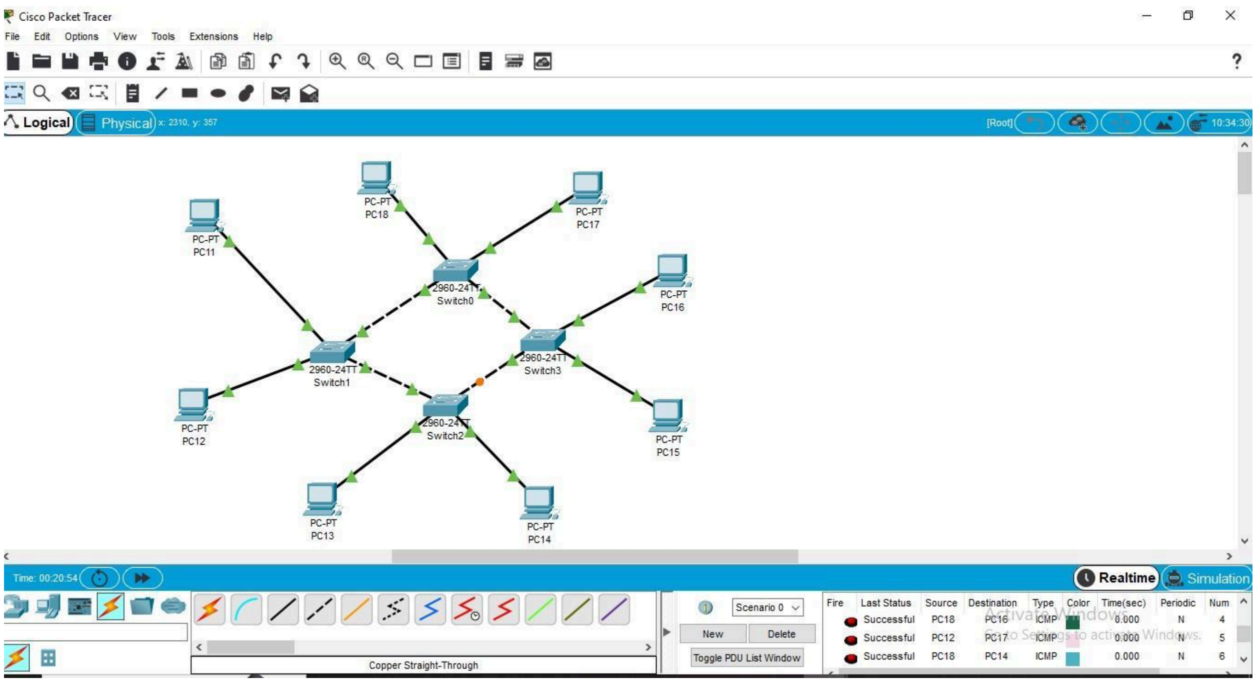
Step 6: Configuring IP Addresses and Subnet Masks on the Hosts

To start communication between the hosts IP Addresses and Subnet Masks had to be configured on the devices. Click once on PC0. Choose the Config tab and click on FastEthernet0. Type the IP address in its field. Click on the subnet mask it will be generated

automatically.

Step 7: To confirm Data transfer between the devices

Click on the node. Select desktop option and then command prompt. Once the window pops up, ping the IP address of the device to which node0 is connected. Ping statistic will be displayed.



Result: Thus the Mesh topology is implemented with Packet Tracer simulation Tool.

Date:

EXPERIMENT-6

IMPLEMENTATION OF TREE TOPOLOGY USING PACKET TRACER

Aim: To Implement a tree topology using packet tracer and hence to transmit data between the devices connected using tree topology.

Software / Apparatus required: Packet Tracer / End devices, Hubs, connectors.

Procedure:

Steps for building topology:

Step 1: Start Packet Tracer

Step 2: Choosing Devices and Connections Step

3: Building the Topology – Adding Hosts

Single click on the **End Devices**.

Single click on the **Generic** host. Move

the cursor into topology area.

Single click in the topology area and it copies the device.

Step 4: Building the Star Topology – Connecting the Hosts to Hubs Select a

Hub, by clicking once on **Hub** and once on a **generic Hub** Add the

Hub by moving the plus sign "+"

Step 5: Connect PCs to Hub by first choosing Connections

Click once on the **Automatic cable selector**

Click once on **PC2** Choose

Fast Ethernet Drag the

cursor to **Hub0** Click once

on **Hub0**

Proceeding in this way create three star topologies

Step 6: Building the Tree Topology – Connecting the Hubs to Active Hub

Connect the hubs of star topologies to active hub to create tree topology.

Step 7: Configuring IP Addresses and Subnet Masks on the Hosts

To start communication between the hosts IP Addresses and Subnet Masks had to be configured on the devices. Click once on PC0. Choose the Config tab and click on Fast Ethernet0. Type the IP address in its field. Click on the subnet mask. It will be generated

automatically.

Step 8: Verifying Connectivity in Real time Mode

Be sure you are in **Real time** mode.

Select the **Add Simple PDU** tool used to ping devices. Click once on PC0, then once on PC3.

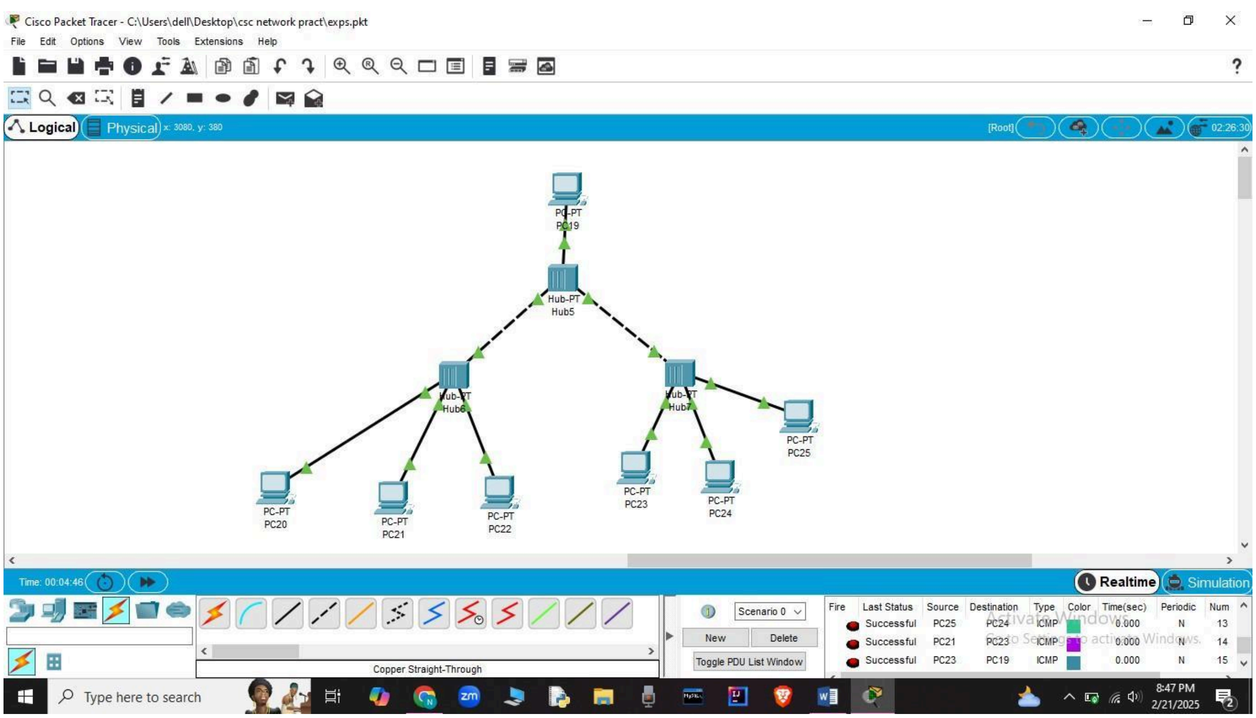
The PDU **Last Status** should show as **Successful**.

Step 9: **Verifying Connectivity in Simulation Mode**

Be sure you are in **Simulation** mode.

Deselect all filters (All/None) and select only **ICMP**. Select the **Add Simple PDU** tool used to ping devices Click once on PC0, then once on PC3.

Continue clicking **Capture/Forward** button until the ICMP ping is completed. You should see the ICMP messages move between the hosts, hub and switch. The PDU **last status** should show as **Successful**.



Result: Thus the Tree topology is implemented with Packet Tracer simulation Tool.

Date:

EXPERIMENT-7

IMPLEMENTATION OF HYBRID TOPOLOGY (BUS AND RING TOPOLOGY) USING PACKET TRACER

Aim: To Implement a hybrid topology using packet tracer and hence to transmit data between the devices connected using tree topology.

Software / Apparatus required: Packet Tracer / End devices, Hubs, connectors.

Steps for building topology:

Step 1: Start Packet Tracer

Step 2: Choosing Devices and Connections Step

3: Building the Topology – Adding Hosts

Single click on the **End Devices**.

Single click on the **Generic** host. Move

the cursor into topology area.

Single click in the topology area and it copies the device.

Step 4: Building the Bus Topology – Connecting the Hosts to Hubs Select a

Hub, by clicking once on **Hub** and once on a **generic Hub** Add the

Hub by moving the plus sign "+"

Step 5: Building the Ring Topology – Connecting the Hosts to Hubs Select a

Hub, by clicking once on **Hub** and once on a **generic Hub** Add the

Hub by moving the plus sign "+"

Step 5: Connect PCs to Hub by first choosing Connections

Click once on the **Automatic cable selector**

Click once on **PC2** Choose

Fast Ethernet Drag the

cursor to **Hub0** Click once

on **Hub0**

Proceeding in this way create three Bus topologies

Step 6: Building the Tree Topology – Connecting the Hubs to Active Hub

Connect the hubs of star topologies to active hub to create tree topology.

Step 7: Configuring IP Addresses and Subnet Masks on the Hosts

To start communication between the hosts IP Addresses and Subnet Masks had to be configured on the devices. Click once on PC0. Choose the Config tab and click on FastEthernet0. Type the IP address in its field. Click on the subnet mask. It will be Generated automatically.

Step 8: Verifying Connectivity in Realtime Mode

Be sure you are in **Realtime** mode.

Select the **Add Simple PDU** tool used to ping devices. Click once on PC0, then once on PC3.

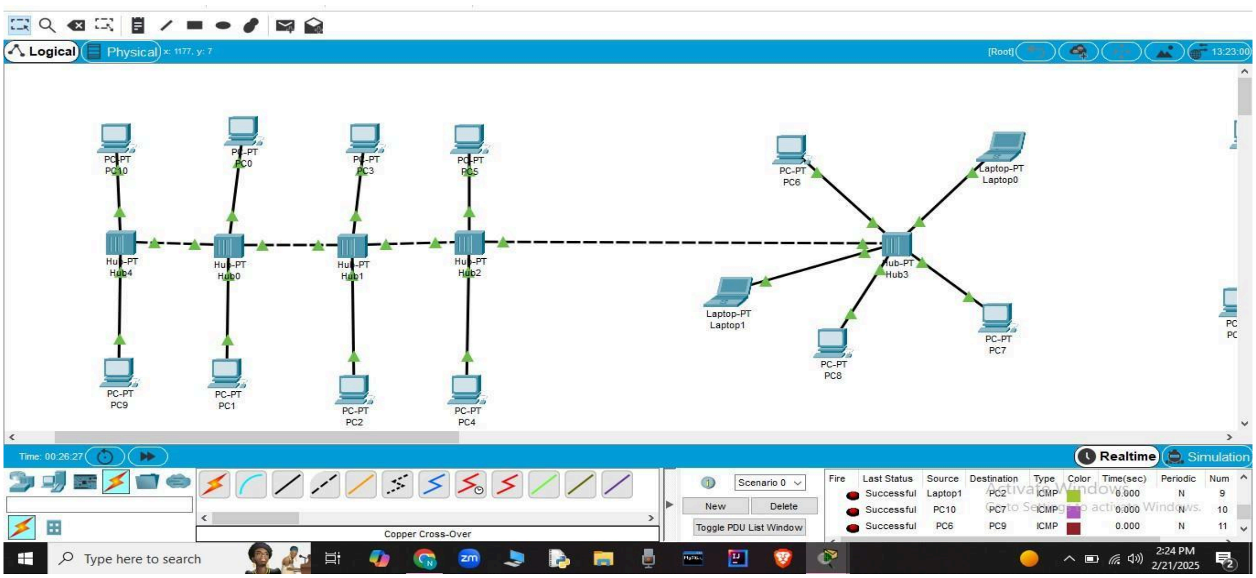
The PDU **Last Status** should show as **Successful**.

Step 9: **Verifying Connectivity in Simulation Mode**

Be sure you are in **Simulation** mode.

Deselect all filters (All/None) and select only **ICMP**. Select the **Add Simple PDU** tool used to ping devices Click once on PC0, then once on PC3.

Continue clicking **Capture/Forward** button until the ICMP ping is completed. The ICMP messages move between the hosts, hub and switch. The PDU **Last Status** should show as **Successful**.



Result: Thus the Hybrid topology is implemented with Packet Tracer simulation Tool.

Date:

EXPERIMENT-8

DATA LINK LAYER TRAFFIC SIMULATION USING PACKET TRACER ANALYSIS OF ARP

Aim: To implement Data Link Layer Traffic Simulation using Packet Tracer Analysis of ARP.

Software / Apparatus required: Packet Tracer / End devices, Switches, connectors.

Requirements:

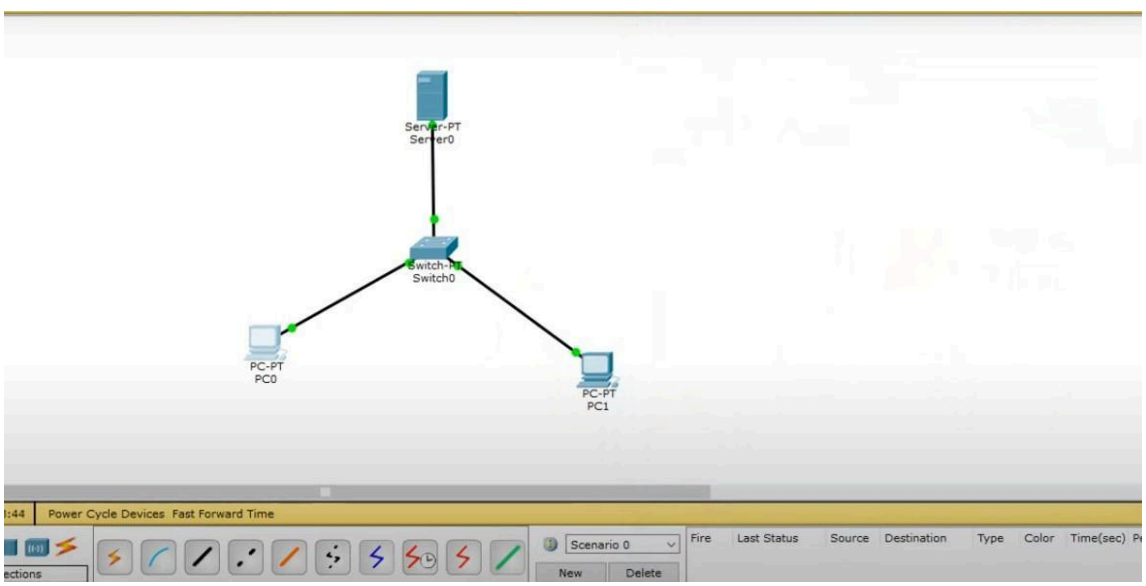
1. End device - They are the devices through which we can pass message from one device to another and they are interconnected.

2 Switch/Hub - Interface Between two devices.

3. Cable - Used to connect two devices

Procedure:

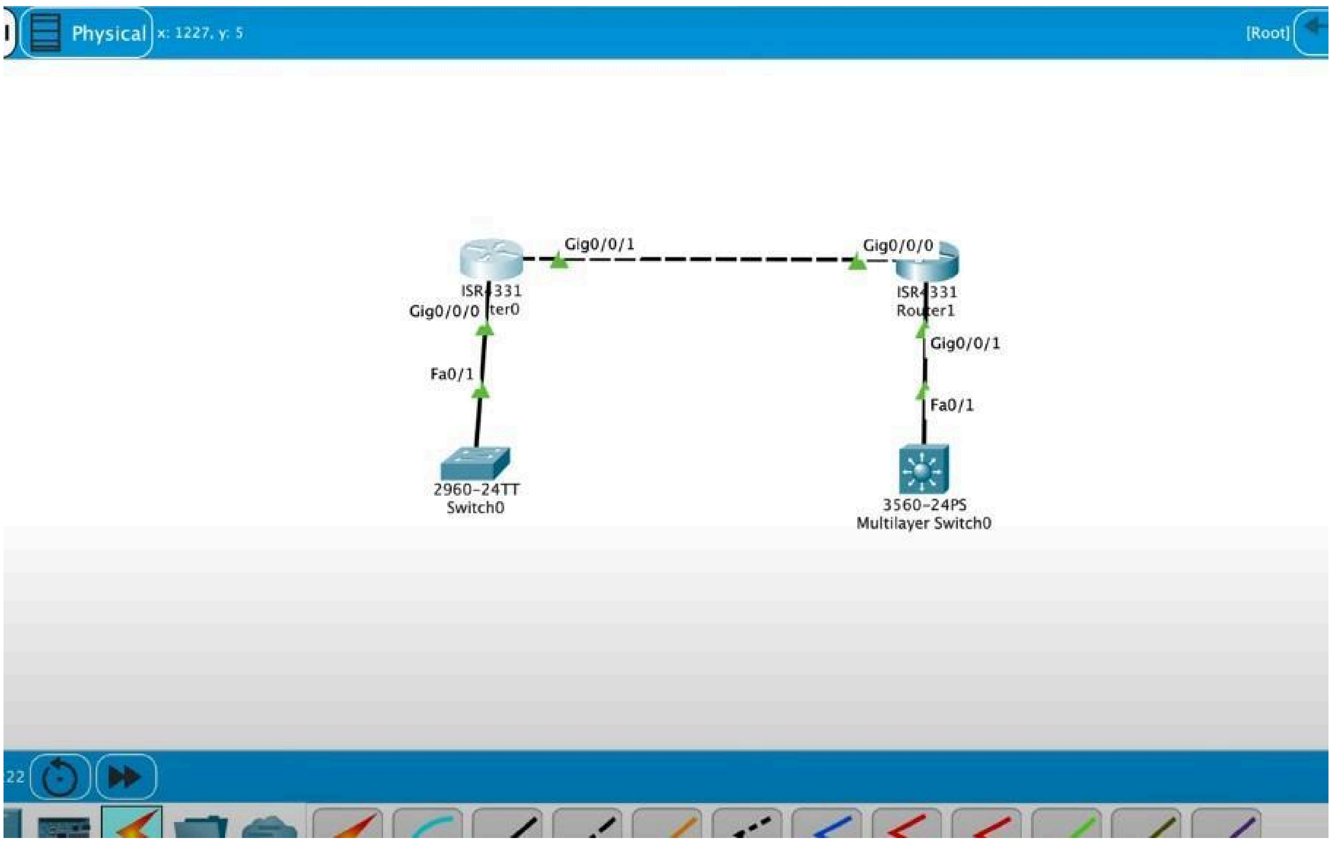
1. Open packet tracer.
2. Click on the list the available capture interface.
3. Choose the PCS, server and Hub.
4. Later give connection from hub to the remaining pcs.
5. Give IP address to the pcs with configuration.
6. Simulate the source and destination.



Result: Thus the Data Link Layer Traffic Simulation using Packet Tracer Analysis of ARP is implemented.

EXPERIMENT-9

DATA LINK LAYER TRAFFIC SIMULATION USING PACKET TRACER ANALYSIS OF LLDP



Date:

EXPERIMENT-10

DATA LINK LAYER TRAFFIC SIMULATION USING PACKET TRACER

ANALYSIS OF CSMA/CD & CSMA/CA

Aim: To implement Data Link Layer Traffic Simulation using Packet Tracer Analysis of CSMA/CD & CSMA/CA.

Software / Apparatus required: Packet Tracer / End devices, Switches, connectors.

Requirements:

1. End device - They are the devices through which we can pass message from one device to another and they are interconnected.
2 Switch/Hub - Interface Between two devices.
3. Cable - Used to connect two devices

Procedure:

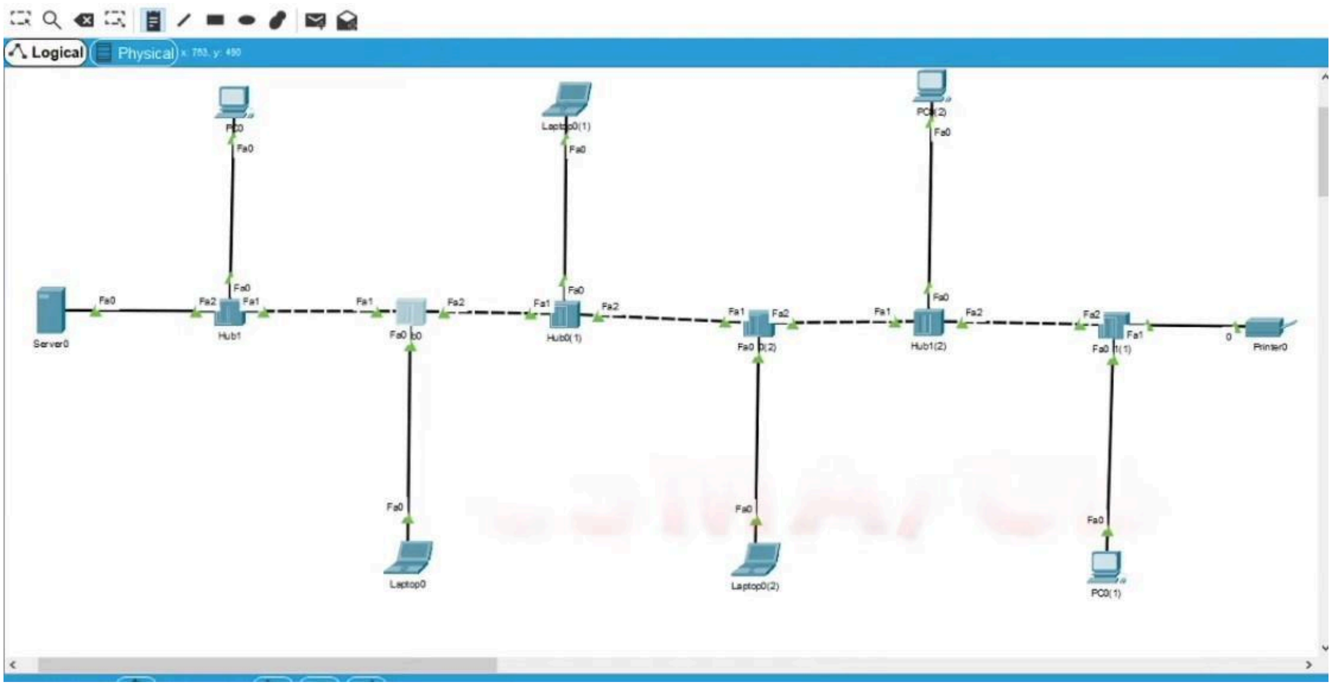
STEP 1: Click on end devices, select generic Pc's drag and drop it on the window.
Click on SWITCH drag and drop it on the window.

STEP 2: Select the straight through cable and connect all end device to switch. Assign the IP address for all end devices. (Double click the end device Select →
desktop → IP configuration static)

STEP 3: Now set the IP address to Host A (192.168.1.1) in static mode. Similarly set IP address for Host B (192.168.1.2) and Host C (192.168.1.3)

STEP 4: To view the IP address, give ip config command in command prompt. Using ping command, we can establish communication between two host devices.

STEP 5: Now display the packet transmission in simulation mode.



Result: Thus Data Link Layer Traffic Simulation using Packet Tracer Analysis of CSMA/CD & CSMA/CA is implemented successfully.

regular star topology+in cmd prompt while clicking on pc, write’arp -a’

Date:

EXPERIMENT-11

CONFIGURATION OF A SIMPLE STATIC ROUTING IN PACKET TRACER USING A SIMPLE TOPOLOGY WITH TWO ROUTERS

Aim: To Configure a router using packet tracer software and hence to transmit data between the devices in real time mode and simulation mode.

Software/Apparatus required: Packet Tracer/End devices, Hubs, connectors.

Procedure:

Steps for building topology:

Step 1: Start Packet Tracer

Step 2: Choosing Devices and Connections

Step 3: Single click on the **End Devices**.

Single click on the **Generic** Host.

Place PC0, PC1 on topology area.

Connect PCs to Switch 1.

Similarly Place PC2, PC3 on topology area for receiver side

Connect these PCs with switch 1 and 2 respectively through connecting wires.

Select Router and place the router between two switches.

Connect these switches into router through connecting wires.

Step 3: Configuring IP Addresses, Gate Way and Subnet Masks on the Hosts

To start communication between the hosts IP Addresses, subnet Masks and Gate way had to be configured on the devices. Click once on PCs. Choose the Config tab and click on FastEthernet0. Type the IP address in its field. Based on router create gate way click on the subnet mask. It will be generated automatically.

Step 4: Verifying Connectivity in Real time Mode

Be sure you are in **Real time** mode.

Select the **Add Simple PDU** tool used to ping devices.

Click once on PC0, then once on PC3.

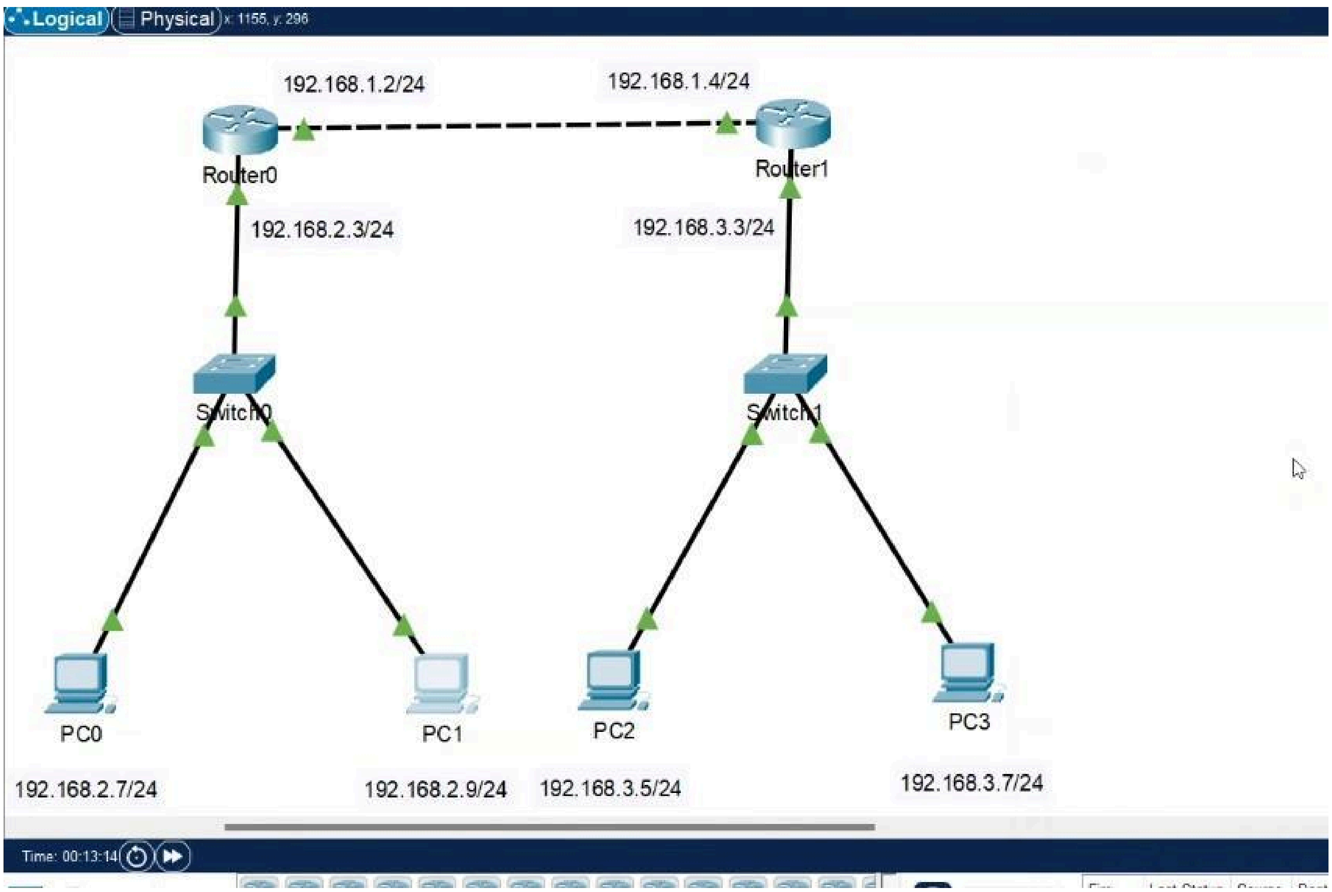
The PDU **Last Status** should show as **Successful**.

Step 5: **Verifying Connectivity in Simulation Mode**

Be sure you are in **Simulation** mode.

Deselect all filters (All/None) and select only **ICMP**. Select the **Add Simple PDU** tool used to ping devices Click once on PC0, then once on PC3.

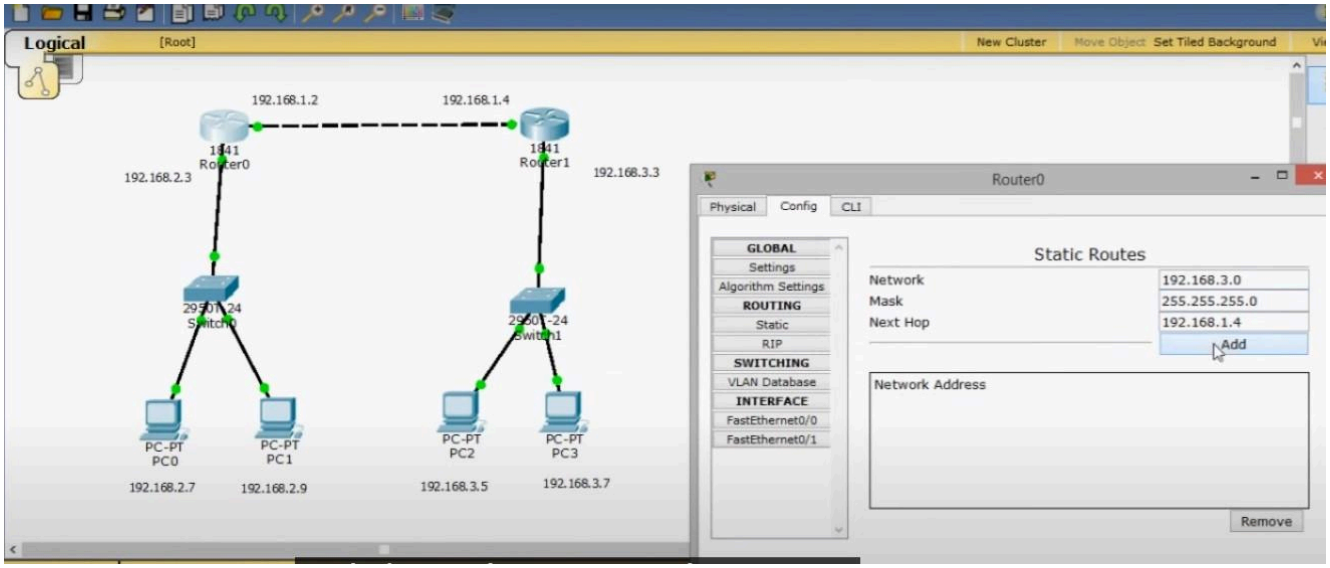
Continue clicking **Capture/Forward** button until the ICMP ping is completed. The ICMP messages move between the hosts, hub and switch. The PDU **Last Status** should show as **Successful**.

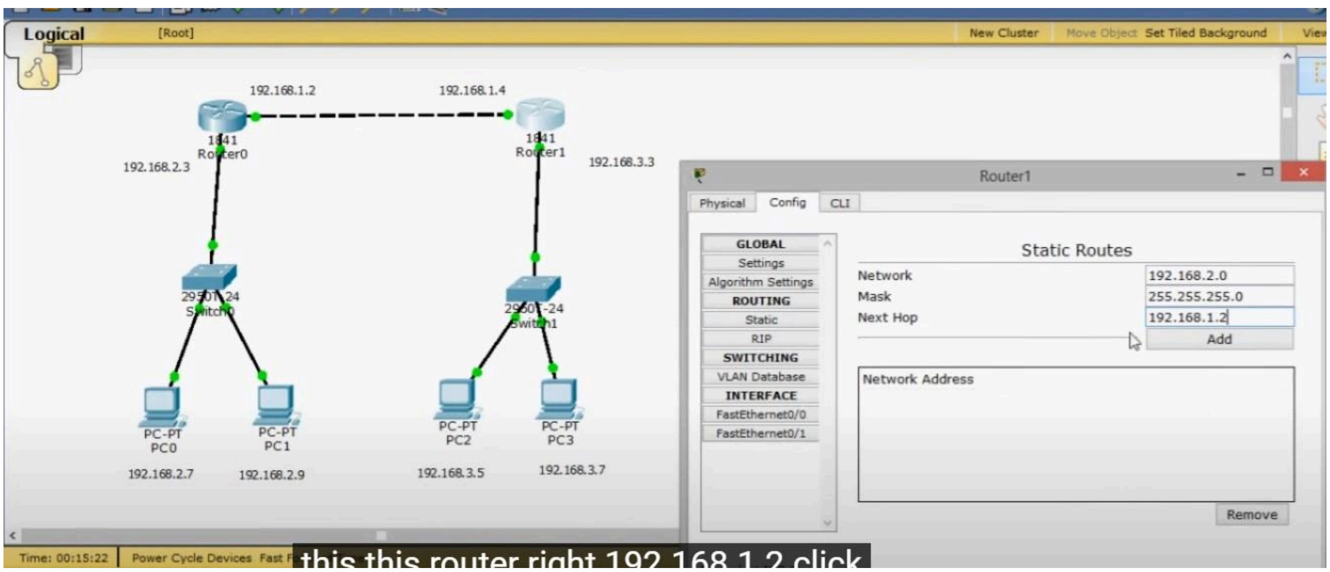


Result: Thus Configuration of a simple static routing in packet tracer using a simple topology with two routers was done successfully.

config both routers with ip addresses of both ports (fastethernet0/0 , fastethernet0/1) in each router. ip add is in pic

while config pcs, give default gateway as 192.168.2.3 (for pcs in router 0) & give default gateway as 192.168.3.3 (for pcs in router 1)



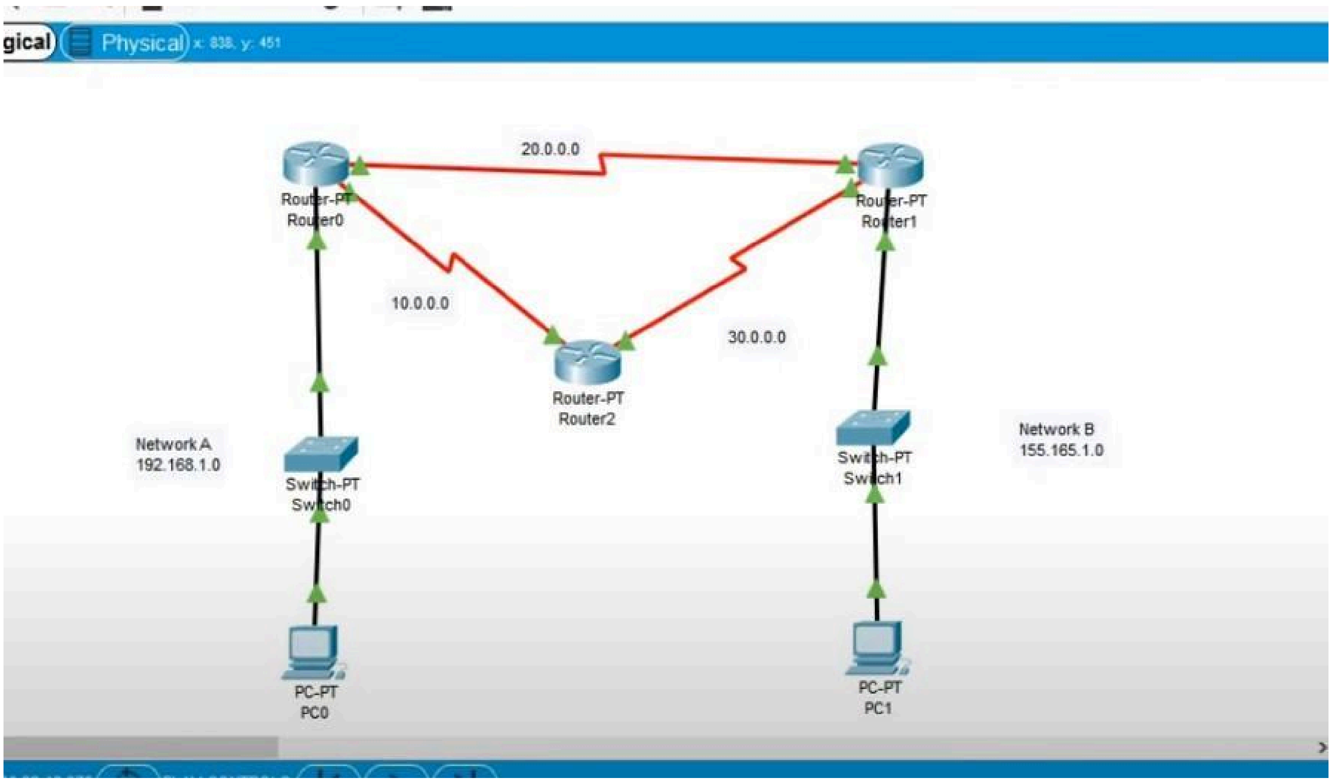


after this, packets can be transferred for eg:pc0 to pc3

Date:

EXPERIMENT-12

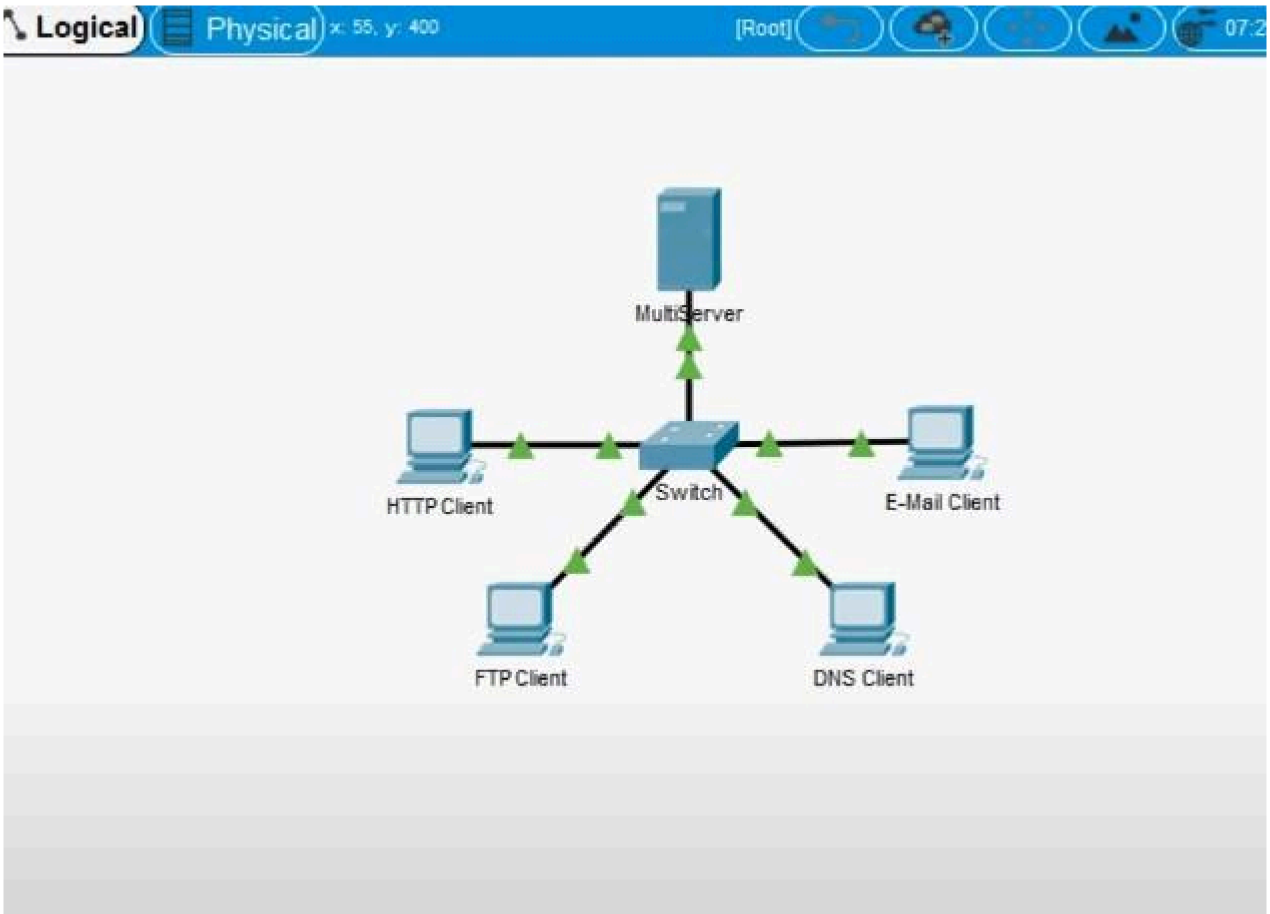
DESIGNING TWO DIFFERENT NETWORKS WITH DYNAMIC ROUTING TECHNIQUES (RIP & OSPF) USING PACKET TRACER



Date:

EXPERIMENT-13

DESIGN THE FUNCTIONALITIES AND EXPLORATION OF TCP USING PACKET TRACER



while configuring ip for each pc, give ip, generate subnet mask and give dns server –192.168.11.5, ie., ip of server to all pcs

ip of pc0–192.168.11.1

ip of pc1–192.168.11.2

ip of pc2–192.168.11.3

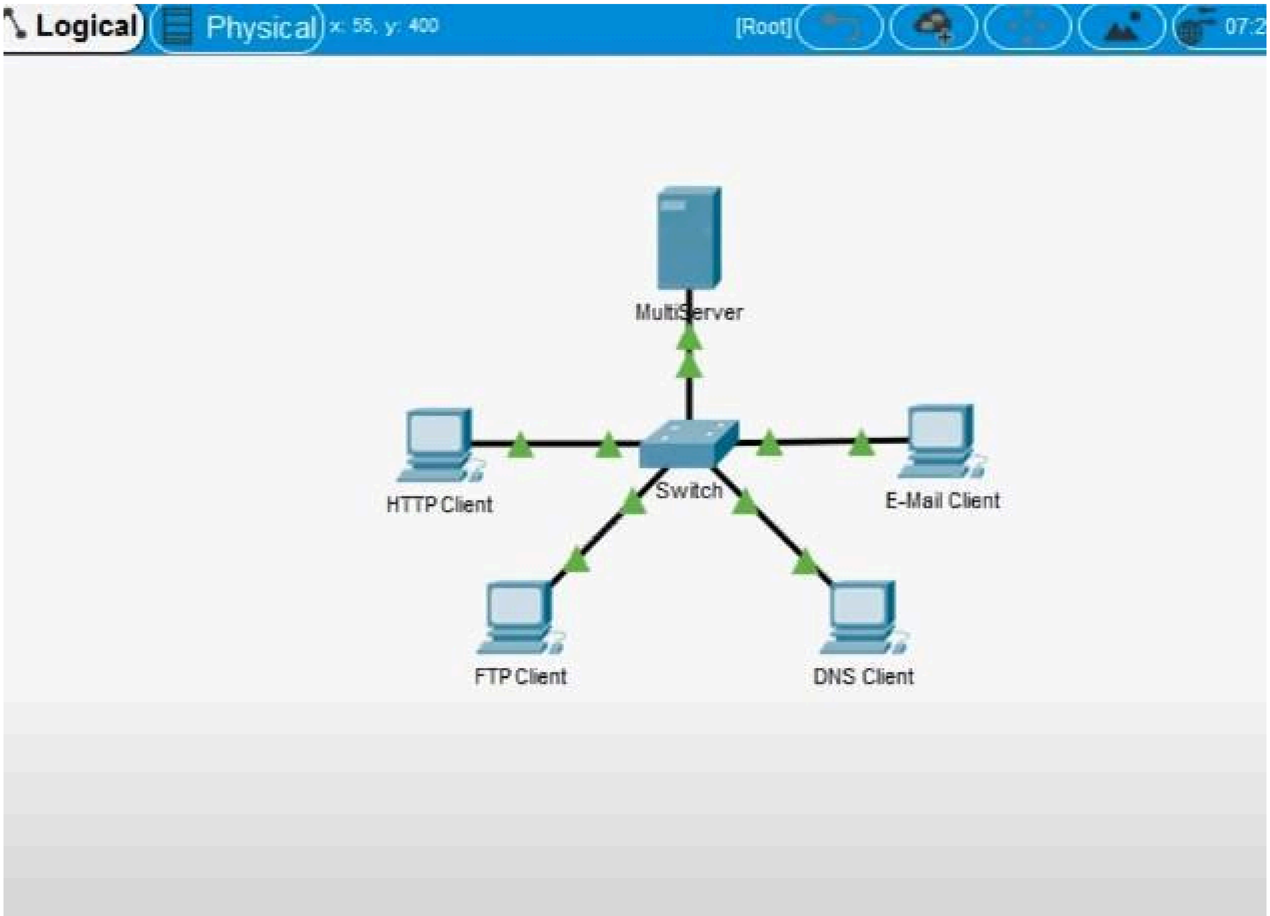
ip of pc3–192.168.11.4

ip of server–192.168.11.5

Date:

EXPERIMENT-14

DESIGN THE FUNCTIONALITIES AND EXPLORATION OF UDP USING PACKET TRACER



Date:

EXPERIMENT-15

DESIGN THE NETWORK MODEL FOR SUBNETTING – CLASS C ADDRESSING USING PACKET TRACER

AIM: To design the network model for subnetting-class C addressing using packet tracer.

Software/Apparatus required: Packet Tracer/End devices, Hubs, connectors.

Algorithm:

1. Determine the network requirements: Identify the number of subnets and hosts required for each subnet.
2. Choose a subnet mask: Select a subnet mask that can accommodate the required number of subnets and hosts.
3. Calculate the subnet mask and prefix length: Use the formula 2^p - 2 >= n, where p is the number of host bits and n is the required number of hosts per subnet, to calculate the number of host bits required. Add these host bits to the Class C network address to create the subnet address. The remaining bits in the subnet mask will be the prefix length.
4. Configure the router: Configure the router interface with the subnet address and subnet mask.
5. Configure the hosts: Configure each host with an IP address and subnet mask that matches the subnet address and subnet mask used on the router interface.
6. Test the network: Verify that the hosts can communicate with each other and with devices on other subnets.
7. Monitor network traffic: Use Packet Tracer's built-in network monitoring tools to monitor network traffic and identify any potential issue.

Procedure:

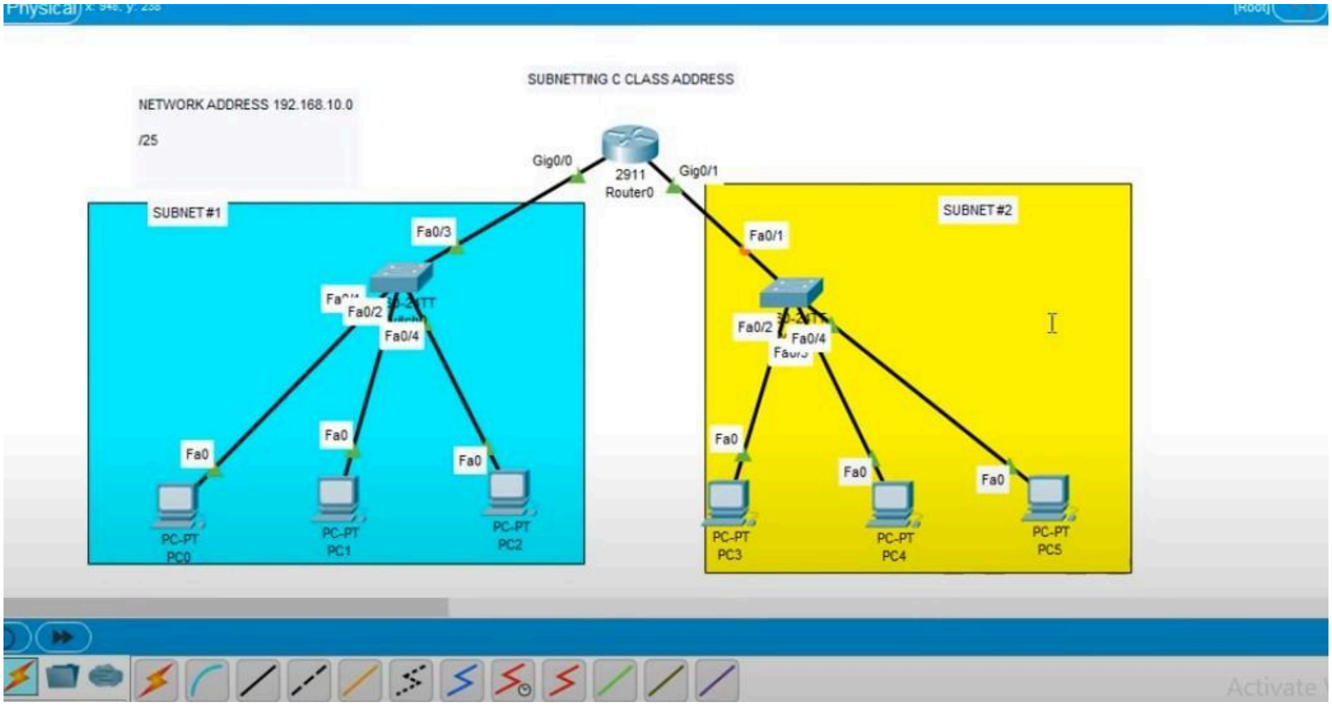
STEP 1: Click on end devices, select generic Pc’s drag and drop it on the window. Click on SWITCH drag and drop it on the window.

STEP 2: Select the straight through cable and connect all end device to switch. Assign the IP address for all end devices. (Double click the end device Select → desktop → IP configuration static

STEP 3: Now set the IP address to Host A (192.168.1.1) in static mode. Similarly set IP address for Host B (192.168.1.2) and Host C (192.168.1.3)

STEP 4: To view the IP address, give ipconfig command in command prompt. Using ping command, we can establish communication between two host devices.

STEP 6: Now display the packet transmission in simulation mode.



Result:

There for designing for network model subnetting has been successfully implemented using packet tracer.

192.168.10.0 N.A. GIVEN

/25 GIVEN

HON MANY SUBNETS -- X.X.X.10000000 2^1 - 2

HOW MANY HOSTS/SUBNET -- 2^7 - 128

VALID SUBNETS -- 256 - 128 - 128

0 & 128

BROADCAST ADDRESS FOR EACH SUBNET -- 127 & 255

VALID HOSTS -- 1 126 (SUBNET 0)

129 254 (SUBNET 128)

SUBNET 0

IP ADDRESS–192.168.10.1

SUBNET MASK—255.255.255.128

DEFAULT GATEWAY–192.168.10.4

SUBNET 128

IP ADDRESS–192.168.10.129

SUBNET MASK—255.255.255.128

DEFAULT GATEWAY–192.168.10.132

ROUTER CONFIG:

1st click 'on' in gig0/0

ip address– default gateway–192.168.10.4

subnet mask–255.255.255.128

1st click 'on' in gig0/1

ip address– default gateway–192.168.10.132

subnet mask–255.255.255.128

Date:

EXPERIMENT: 16

SIMULATING X, Y, Z COMPANY NETWORK DESIGN AND SIMULATE USING PACKET TRACER

Aim: To simulate X,Y,Z company network design and stimulate using packet tracer.

Software/Apparatus required: Packet Tracer/End devices, Hubs, connectors.

Algorithm:

1. Identify the network requirements: Determine the number of users, devices, and servers that will be connected to the network.
2. Create a network diagram: Use a network diagramming tool to create a visual representation of the network design, including the devices, servers, switches, routers, and connections.
3. Configure the routers: Configure the routers with IP addresses, subnet masks, and routing protocols as needed.
4. Configure the switches: Configure the switches with VLANs, and assign ports to each VLAN.
5. Configure the servers: Configure the servers with IP addresses, subnet masks, and any necessary applications or services.
6. Configure the workstations: Configure the workstations with IP addresses, subnet masks, and any necessary applications or services.
7. Configure security: Configure security measures such as firewalls, access control lists, and intrusion detection systems as needed.
8. Test the network: Test the network connectivity by pinging devices and verifying that data can be transmitted between them.
9. Monitor network traffic: Use Packet Tracer's built-in network monitoring tools to monitor network traffic and identify any potential issues.
10. Make adjustments as needed: Make adjustments to the network configuration as needed to improve performance, security, or functionality.

Procedure:

1. Start Packet Tracer: Launch Packet Tracer on your computer.
2. Create a new project: Click on "File" and select "New", then select "Network" from the options.
3. Add devices: Click on the "Devices" tab in the bottom-left corner of the window, and drag and drop devices onto the workspace. Add devices such as routers, switches, servers, and workstations.

4. Connect devices: Use the "Cable" tool to connect the devices together. Configure the connections as needed.

5. Configure devices: Double-click on each device to open its configuration menu, and configure its settings such as IP address, subnet mask, and routing protocols. Configure security measures such as firewalls, access control lists, and intrusion detection systems as needed.

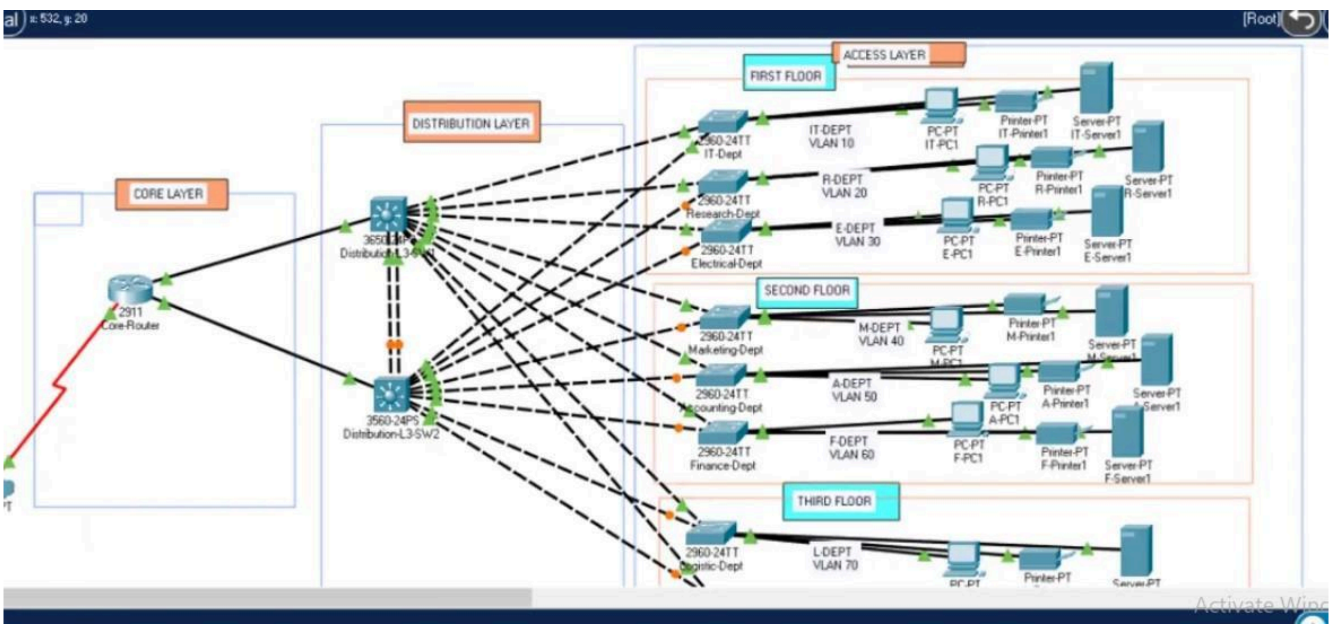
6. Add applications: Click on the "Applications" tab in the bottom-left corner of the window, and drag and drop applications onto the workstations and servers. Configure the applications as needed.

7. Test the network: Use Packet Tracer's built-in testing tools to verify that the network is working correctly. Test the network connectivity by pinging devices and verifying that data can be transmitted between them.

8. Monitor network traffic: Use Packet Tracer's built-in network monitoring tools to monitor network traffic and identify any potential issues.

9. Make adjustments as needed: Make adjustments to the network configuration as needed to improve performance, security, or functionality.

10. Save the project: Click on "File" and select "Save" to save the project.



Result: Therefore stimulating of companies network designing has been successfully done using packet tracer.

Date:

EXPERIMENT: 17

CONFIGURATION OF DHCP (DYNAMIC HOST CONFIGURATION PROTOCOL) IN PACKET TRACER

Aim: To configure DHCP (dynamic host configuration protocol) in packet tracer.

Software/Apparatus required: Packet Tracer/End devices, Hubs, connectors.

Algorithm:

1. Start:

- Set up the network topology in Packet Tracer with a DHCP server and DHCP clients connected to a switch.

2. Configure the DHCP server:

- Assign an IP address to the server interface.
- Enable the DHCP service on the server.
- Define the IP address pool range that the server can assign to clients.
- Specify additional DHCP options like default gateway, DNS server, and subnet mask.

3. Configure the switch.

- Enable the switch interfaces that connect to the DHCP clients.

4. Configure the DHCP clients.

- Configure the clients to obtain their IP addresses automatically using DHCP.
- Verify that the clients are set to use DHCP as the preferred method for IP assignment.

5. Client request and server response:

- When a DHCP client boots up or its lease expires, it sends a DHCP discover message as a broadcast on the local network.
- The DHCP server receives the discover message and responds with a DHCP offer message containing an available IP address from the configured IP address pool.
- The server includes other network configuration parameters in the offer message.

6. Client selection and request:

- The client receives multiple offer messages from different DHCP servers if available.
- The client selects one offer and sends a DHCP request message to the chosen server, requesting the offered IP address and confirming other network parameters.

7. Server acknowledgement:

- The DHCP server receives the request message and sends a DHCP acknowledge (ACK) message to the client, confirming the IP address assignment and providing additional network configuration details.

8. Client configuration:

• The client receives the ACK message and configures its network interface with the assigned IP address, subnet mask, default gateway, DNS server, and any other parameters provided by the DHCP server.

9. Lease renewal and expiration:

• The client periodically contacts the DHCP server to renew its lease before it expires.

• If the client doesn't renew the lease or is unable to contact the DHCP server, the IP address lease eventually expires, and the IP address returns to the pool for future assignment.

10. End:

Procedure:

1. Launch Cisco Packet Tracer and create a new network topology or open an existing one.

2. Add the necessary network devices to your topology, including a DHCP server, switch, and DHCP clients. Connect them using appropriate cables.

3. Configure the DHCP server:

• Select the DHCP server device and open its configuration panel.

• Assign an IP address to the server interface connected to the switch.

• Enable the DHCP service on the server by checking the "DHCP" option.

• Define the IP address pool range that the server can assign to clients. Specify the starting and ending IP addresses.

• Optionally, set other DHCP options like default gateway, DNS server, and subnet mask.

• Save the configuration.

4. Configure the switch:

• Select the switch device and open its configuration panel.

• Enable the interfaces that connect to the DHCP clients. This allows the clients to communicate with the DHCP server.

• Save the configuration.

5. Configure the DHCP clients:

• Select each DHCP client device and open its configuration panel.

• Set the IP address assignment method to "DHCP" or "Obtain an IP address automatically."

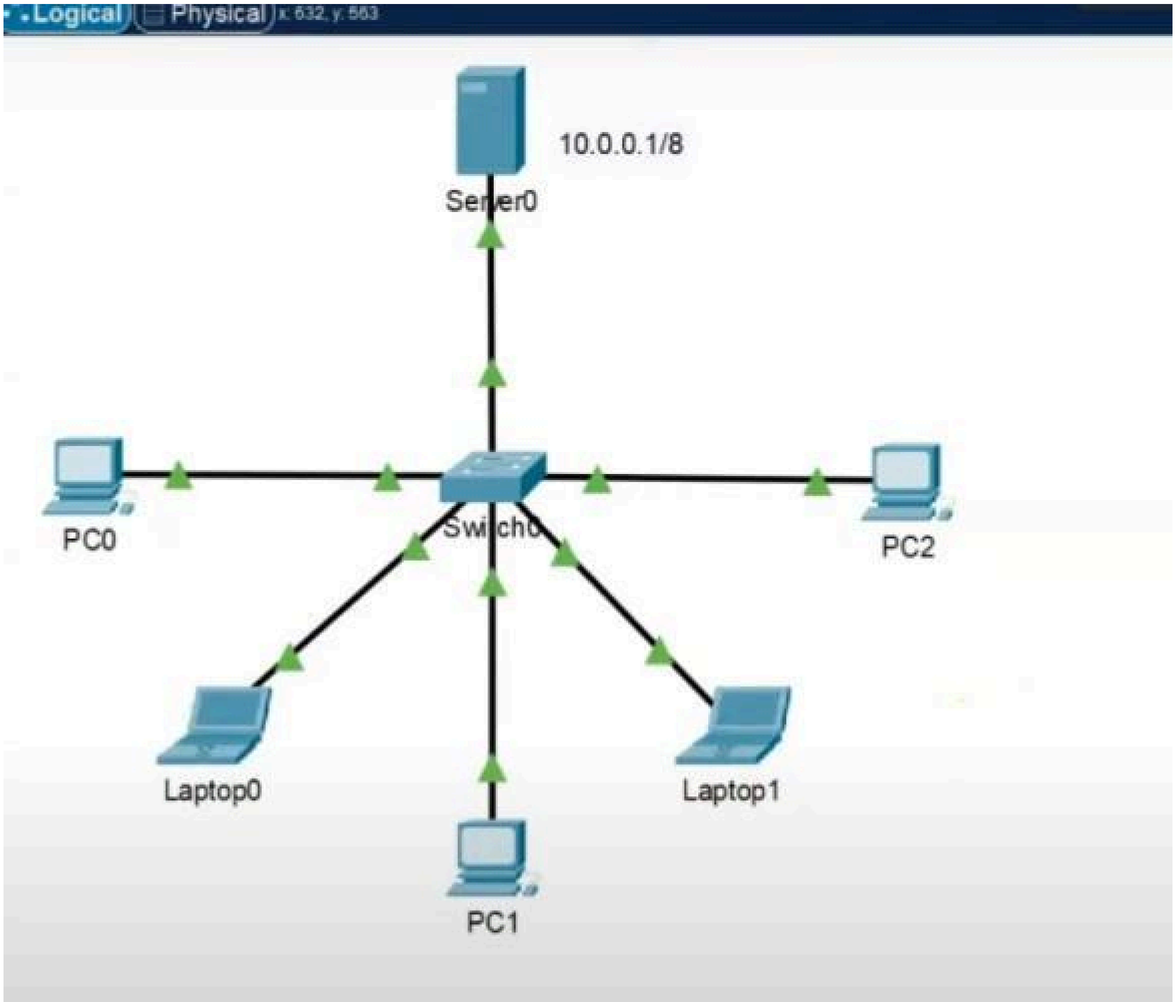
• Save the configuration for each client.

6. Start the simulation:

- Click the "Start/Stop Simulation" button to start the simulation.

7. Verify DHCP operation:

- Wait for the DHCP clients to boot up or refresh their IP configurations.
- Check if the DHCP clients receive IP addresses from the DHCP server.
- Verify that the clients have the correct IP address, subnet mask, default gateway, and DNS server settings.



Result: Therefore the configuration for DHCP has been successfully executed using packet tracer.

Date:

E

XPERIMENT-18

CONFIGURATION

OF FIREWALL IN

PACKET TRACER

Aim: To configure firewall in packet tracer.

Software/Apparatus required: Packet Tracer/End devices, Hubs, connectors.

Procedure:

Step 1: Set up the network topology

To begin, we will create a simple network topology consisting of three computers, a router, and a firewall. Open Packet Tracer and drag three PCs, a router, and a firewall onto the workspace. Connect the three PCs to the router using Ethernet cables, and connect the firewall to the router using another Ethernet cable.

Step 2: Configure IP addresses

Next, we will configure IP addresses for the computers. Double-click on each PC to open the configuration window and navigate to the Desktop tab. Click on the IP Configuration icon and enter the IP address and subnet mask for each computer. For example, PC1 can have an IP address of 192.168.1.1 with a subnet mask of 255.255.255.0, PC2 can have an IP address of 192.168.1.2 with the same subnet mask, and PC3 can have an IP address of 192.168.1.3 with the same subnet mask

Step 3: Configure the router

Now, we will configure the router. Double-click on the router to open the configuration window and navigate to the CLI tab. Enter the following commands:

Commands :

enable

configure terminal interface

FastEthernet0/0

ip address 192.168.1.254 255.255.255.0

no shutdown exit

Step 4: Configure the firewall

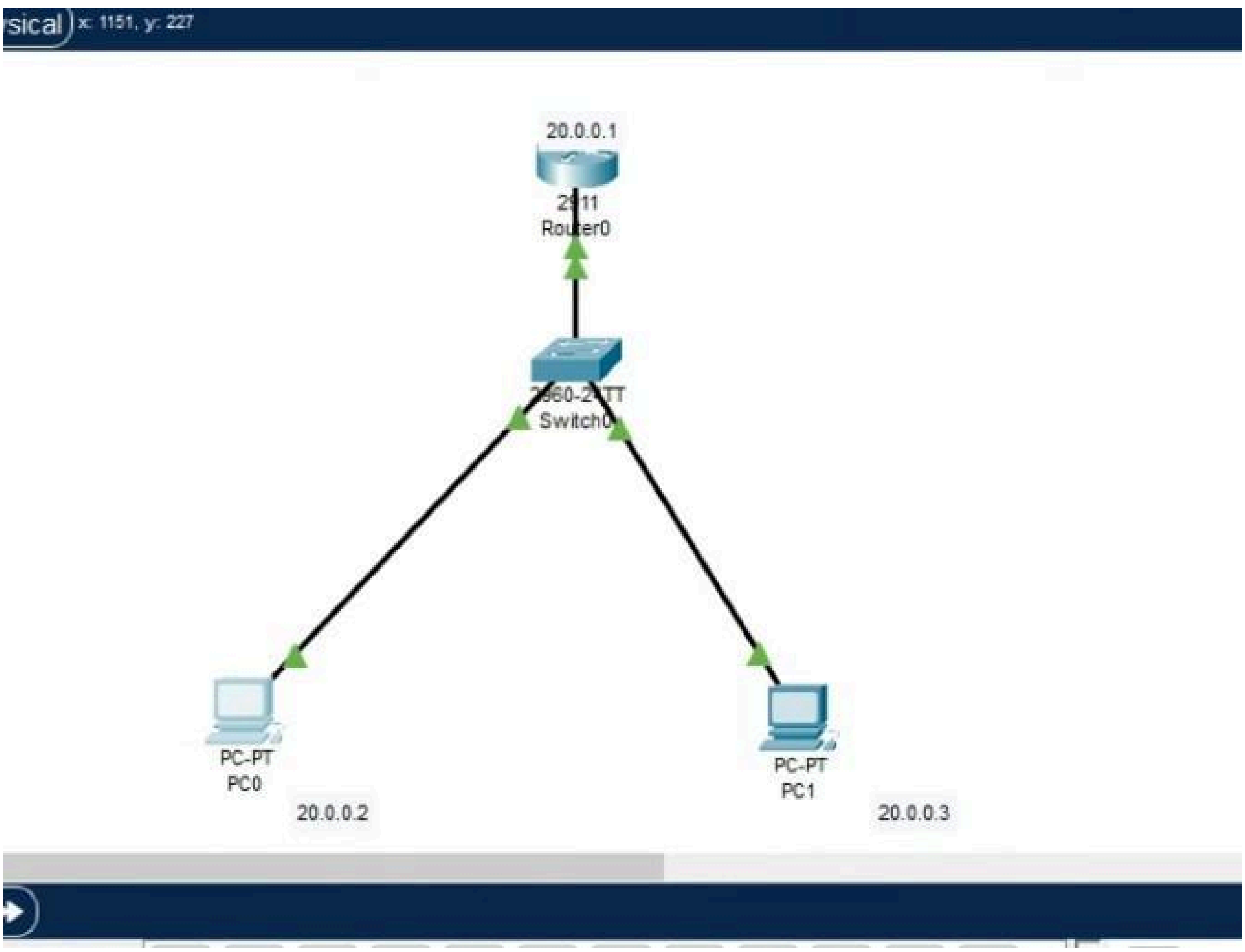
Now, we will configure the firewall. Double-click on the firewall to open the configuration window

Step 5: Test the connection

Now that the firewall is configured, we can test the connection between the computers. Open a command prompt on PC1 and ping PC2 and PC3 by typing ping 192.168.1.2 and ping 192.168.1.3 in the command prompt. If the pings are successful, it means that the computers are communicating with each other.

Step 6: Test the firewall

To test the firewall, try to connect to PC1 from the internet using a protocol or port that is not allowed by the access rule. For example, you can try to connect to PC1 using Telnet on port 23.



Result: Hence the configuration of firewall in packet tracer is successful.

Date:

EXPERIMENT-19

MAKE A COMPUTER LAB TO TRANSFER A MESSAGE FROM ONE NODE TO ANOTHER TO DESIGN AND SIMULATE USING CISCO PACKET TRACER

Aim: To make a Computer Lab to transfer a message from one node to another to design and simulate using Cisco Packet Tracer.

Software/Apparatus required: Packet Tracer/End devices, Hubs, connectors.

Procedure:

Step 1: Create the network topology

First, we need to create the network topology for the computer lab. In Packet Tracer, drag two computers, a switch, and two routers onto the workspace. Connect the computers to the switch using Ethernet cables, and connect the switch to the two routers using Ethernet cables. The network should look like this:

CODE:

```
PC1        PC2
 |          |
Switch -----Router1        Router2
```

Step 2: Configure IP addresses

Next, we will configure IP addresses for the computers. Double-click on each PC to open the configuration window and navigate to the Desktop tab. Click on the IP Configuration icon and enter the IP address and subnet mask for each computer. For example, PC1 can have an IP address of 192.168.1.1 with a subnet mask of 255.255.255.0, and PC2 can have an IP address of 192.168.1.2 with the same subnet mask.

Step 3: Configure the routers

Now, we will configure the routers. Double-click on Router1 to open the configuration window and navigate to the CLI tab. Enter the following commands:

COMMANDS:

enable

configure terminal interface

FastEthernet0/0

ip address 192.168.1.254 255.255.255.0

no shutdown interface

Serial0/0/0

ip address 10.0.0.1 255.255.255.252

no shutdown exit

Now, double-click on Router2 to open the configuration window and navigate to the CLI tab.
Enter the following commands enable

configure terminal

interface Serial0/0/0

ip address 10.0.0.2 255.255.255.252

no shutdown

interface FastEthernet0/0

ip address 192.168.2.254 255.255.255.0

no shutdown exit

Step 4: Configure routing

We need to configure routing between the routers so that they can communicate with each other. Enter the following commands on Router1:

enable

configure terminal

ip route 192.168.2.0 255.255.255.0 10.0.0.2

exit

These commands will configure a static route on Router1 to reach the 192.168.2.0/24 network, which is connected to Router2's Fast Ethernet interface.

enable

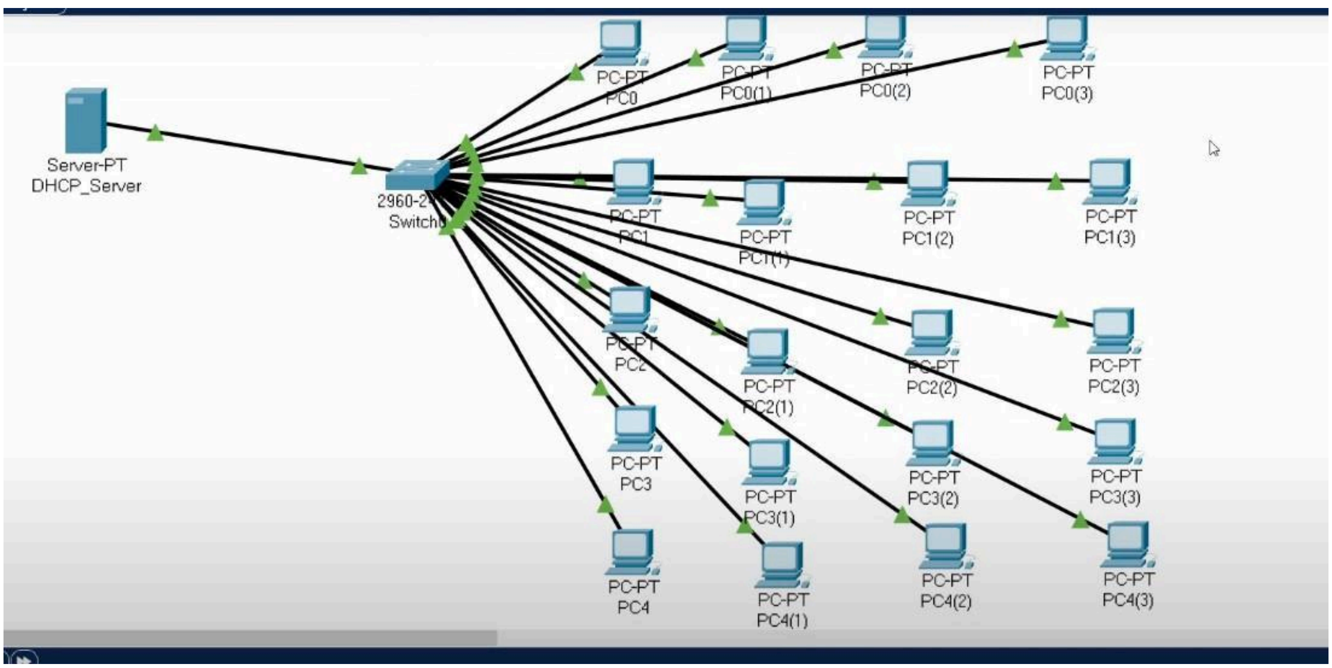
configure terminal

ip route 192.168.1.0 255.255.255.0 10.0.0.1

exit

Step 5: Send a message

To send a message from PC1 to PC2, open the command prompt on PC1 and type: ping 192.168.1.2



Result: Hence the message is transferred from one node to another to design and simulate using Cisco Packet Tracer successfully .

Date:

EXPERIMENT-20

SIMULATE A MULTIMEDIA NETWORK IN CISCO PACKET TRACER

Aim: To simulate a Multimedia Network in Cisco Packet Tracer.

Software/Apparatus required: Packet Tracer/End devices, Hubs, connectors.

Algorithm:

Procedure:

Step 1: Launch Cisco Packet Tracer and create a new project.

Step 2: Select the appropriate network devices for your multimedia network. You will need computers, switches, routers, and multimedia devices such as IP phones and IP cameras. You can find these devices in the "End Devices," "Switches," "Routers," "Phones," and "IP Cameras" sections of the device list.

Step 3: Design the network topology. Determine the layout of your network and the connections between devices. For example, you can connect the computers, IP phones, and IP cameras to a switch, and then connect the switch to a router for internet connectivity.

Step 4: Drag and drop the devices onto the workspace area. Connect the devices using appropriate cables or wireless connections. For example, use Ethernet cables to connect computers and IP phones to the switch.

Step 5: Configure IP addresses on the devices. Assign IP addresses, subnet masks, and default gateways to the computers, IP phones, and IP cameras. Configure the router's interface with an IP address provided by your ISP or use a DHCP server if available.

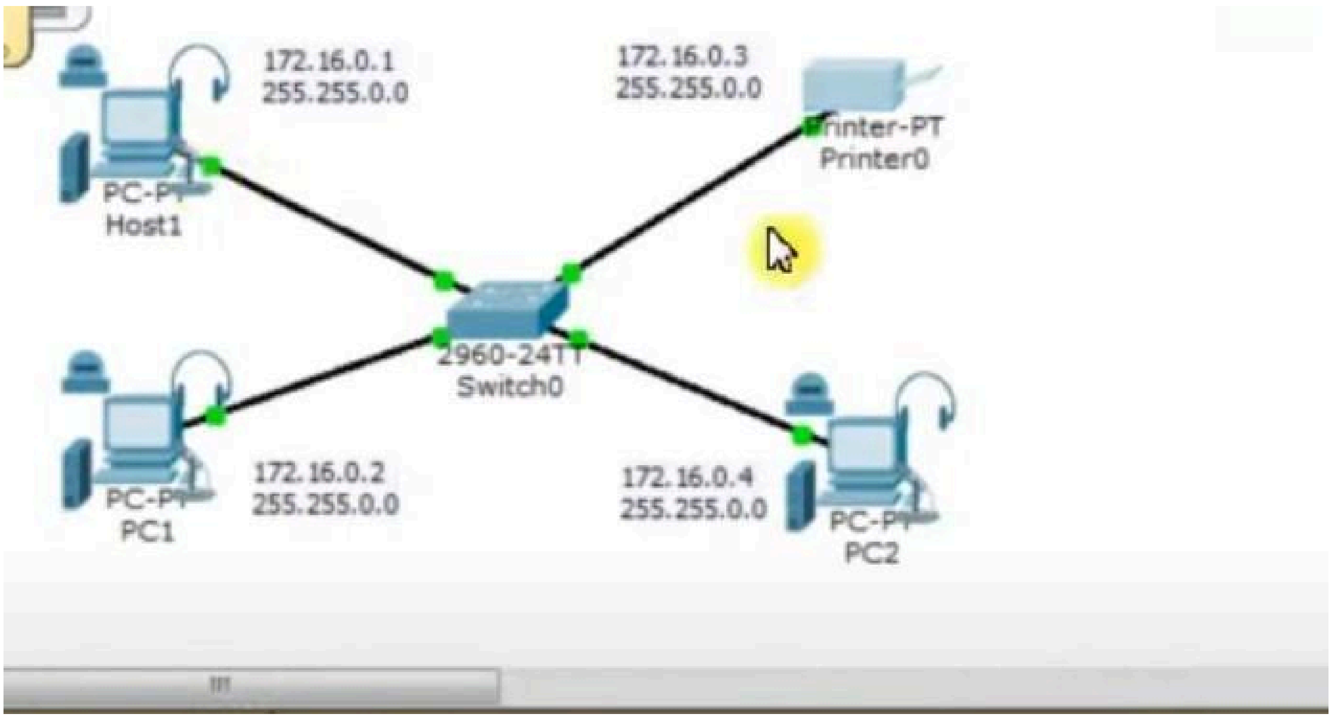
Step 6: Set up multimedia services. Configure the necessary services for multimedia communication, such as VoIP (Voice over IP) for IP phones and streaming protocols for IP cameras. This may involve configuring protocols like SIP (Session Initiation Protocol) for IP phones or RTSP (Real-Time Streaming Protocol) for IP cameras.

Step 7: Test connectivity and multimedia services. Verify that devices can communicate with each other and multimedia services are functioning correctly. For example, try making a call between IP phones or access the video feed from IP cameras.

Step 8: Monitor and troubleshoot. Use the network monitoring tools in Cisco Packet Tracer to observe network traffic and performance. Troubleshoot any issues that arise, such as connectivity problems or audio/video quality degradation.

Step 9: Document the lab experiment. Record observations, configurations, and any issues encountered during the simulation. This documentation will help to analyze the results and make improvements if necessary.

Remember to save your project regularly to preserve your progress. Cisco Packet Tracer provides a simulated environment to experiment with multimedia networks, allowing you to understand the challenges and requirements of such networks in a virtual setting.



Result: Thus a Multimedia Network in Cisco Packet Tracer is simulated successfully.

Date:

EXPERIMENT-21

IOT BASED SMART HOME APPLICATIONS

Aim: To implement IoT based smart home applications in Cisco Packet Tracer.

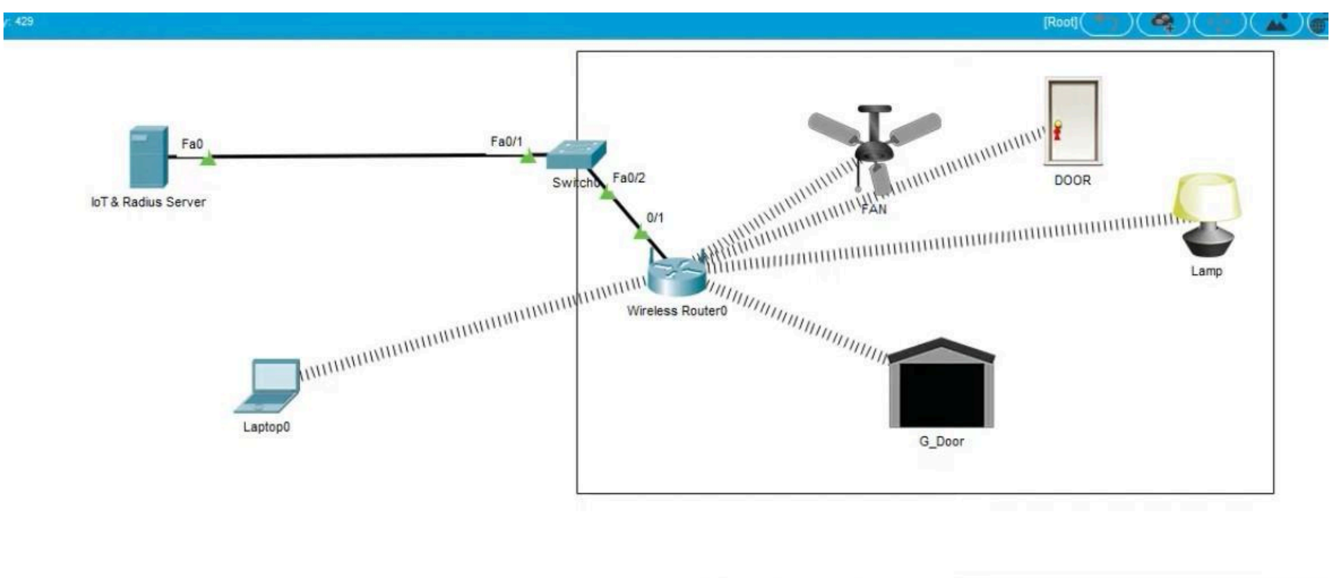
Software/Apparatus required: Packet Tracer/End devices, Hubs, connectors.

Procedure:

Steps:

1. Create a network topology in Cisco Packet Tracer that includes IoT devices such as sensors, actuators, and gateways.
2. Configure the IoT devices with appropriate IP addresses, subnet masks, and gateway addresses.
3. Set up a communication protocol between the IoT devices using MQTT, CoAP, or any other protocol of your choice.
4. Write a code to collect data from the sensors and send it to the gateway.
5. Use the gateway to process the data and send commands to the actuators.
6. Finally, use a web interface or mobile application to monitor and control the IoT devices.

By following these steps an IoT-based smart application in Cisco Packet Tracer , can be created. This can be used for various applications such as home automation, smart cities, and industrial automation.



Result: Thus IoT based smart home applications in Cisco Packet Tracer is implemented successfully.

Date: **EXP**

ERIMENT: 22

IMPLEMENTATION OF IOT BASED SMART GARDENING

Aim: To implement IOT based smart gardening using Cisco packet tracer.

Software/Apparatus required: Packet Tracer/End devices, Hubs, Connectors.

Procedure:

Step 1: Create a new project in Cisco Packet Tracer and drag a generic IoT device from the IoT devices section onto the workspace.

Step 2: Right-click on the IoT device and select Config/Attributes.

Step 3: In the Configuration tab, select the device's IoT server from the drop-down list. You can choose Cisco IoT Cloud or another cloud service of your choice.

Step 4: In the Attributes tab, add the following attributes:

- Temperature
- Humidity
- Soil Moisture
- Light Intensity

Step 5: Create a soil moisture sensor and a light sensor from the Sensors section of the devices panel. Drag and drop these sensors onto the workspace.

Step 6: Connect the sensors to the IoT device using the wiring tool.

Step 7: Configure the sensors by right-clicking on them and selecting Config/Attributes. Set the sensor type, unit of measurement, and other necessary parameters.

Step 8: Create a water pump and a light bulb from the Actuators section of the devices panel. Drag and drop these actuators onto the workspace.

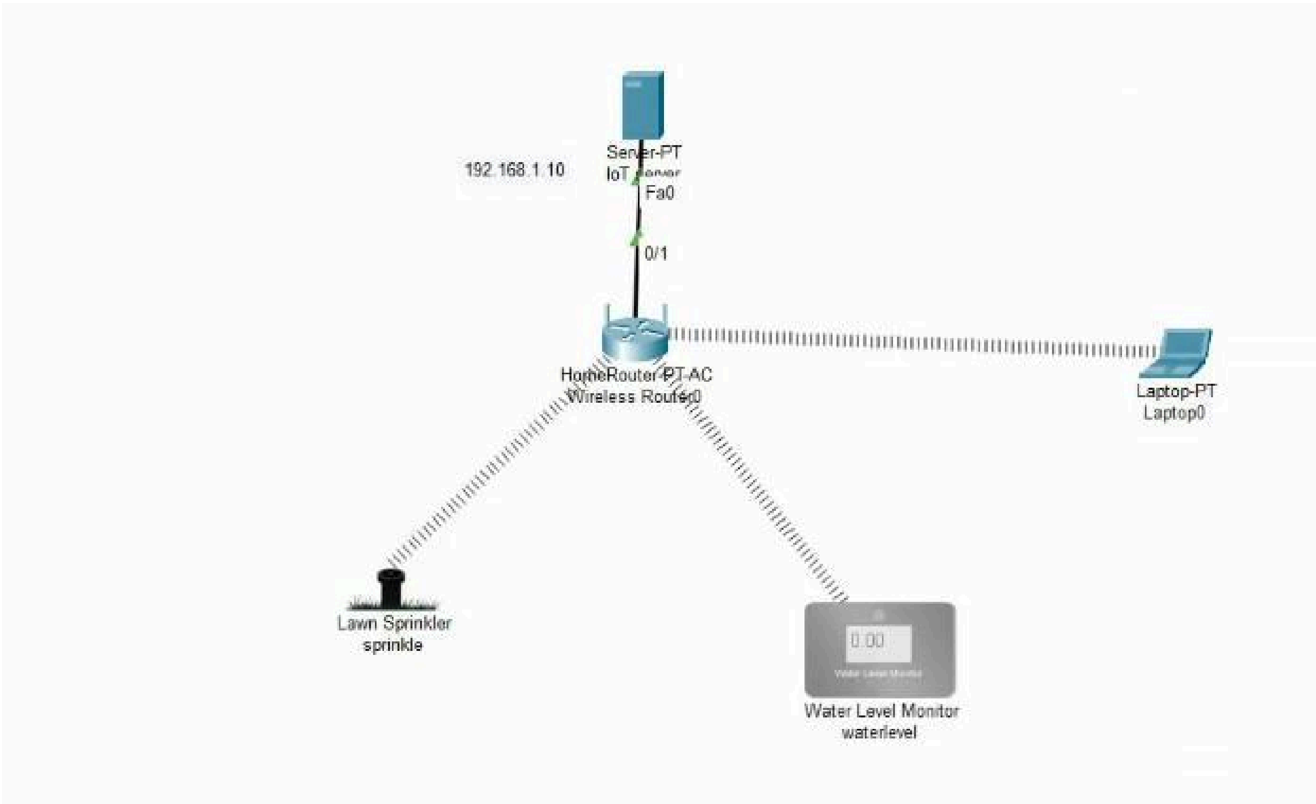
Step 9: Connect the actuators to the IoT device using the wiring tool.

Step 10: Configure the actuators by right-clicking on them and selecting Config/Attributes. Set the actuator type, command, and other necessary parameters.

Step 11: Save the configuration and run the simulation to test your IoT Smart Garden.

Step 12: Monitor the temperature, humidity, soil moisture, and light intensity readings on the IoT device dashboard.

Step 13: Use the dashboard to control the water pump and light bulb based on the sensor readings.



Result: Implementation of smart gardening is carried out using IOT successfully.

Date: **EX**

PERIMENT: 23

IMPLEMENTATION OF

IOT DEVICES IN

NETWORKING

Aim: To implement an IOT devices in networking using Cisco Packet Tracer.

Software/Apparatus required: Packet Tracer/End devices, Hubs, connectors. **Procedure:**

Steps:

1. Open Cisco Packet Tracer and create a new project.

Drag and drop a router from the "Devices" panel onto the workspace area.

2. Connect the router to the Internet by dragging and dropping a "Cloud" device from the "Devices" panel onto the workspace area, and then connecting the router to the cloud using a straight-through cable.

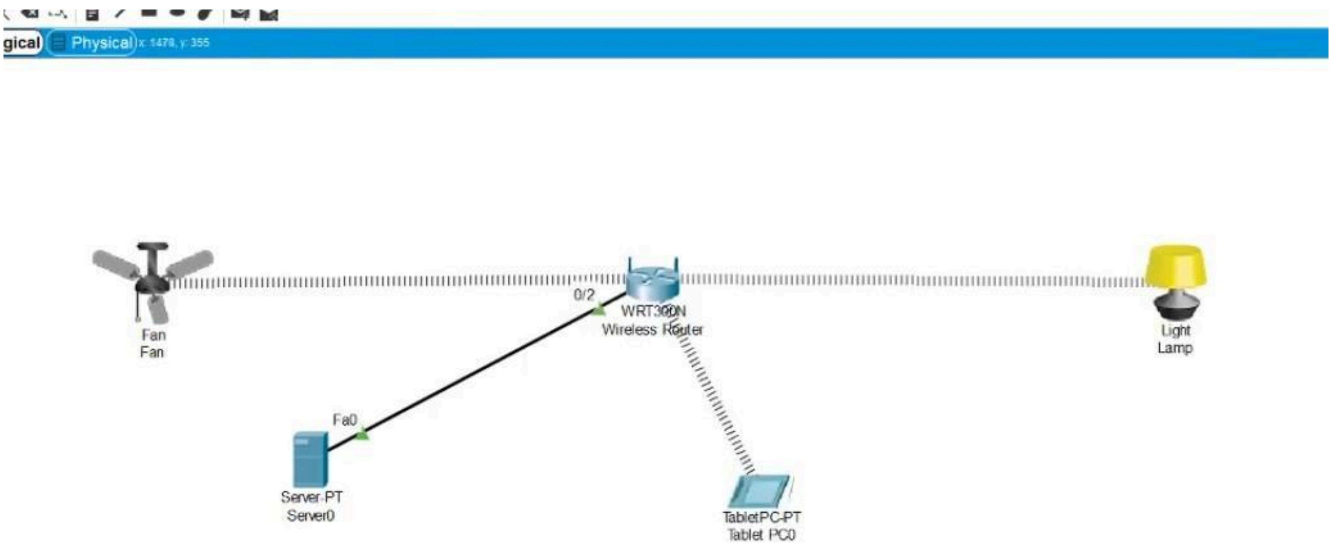
3. Add an IoT device to the network by dragging and dropping a device from the "Devices" panel onto the workspace area. There are various IoT devices available in the "Devices" panel, such as a Raspberry Pi or an Arduino.

4. Connect the IoT device to the router using an Ethernet cable. To do this, click on the IoT device and then click on the "Config" tab. Under the "Interfaces" section, select the Ethernet interface and then click on the "+" button to add a new interface. Connect the new interface to the router.

5. Configure the IoT device by clicking on it and then clicking on the "CLI" tab. This will bring up the command line interface for the IoT device, where you can configure its settings.

6. Test the connectivity of the IoT device by pinging it from the router or from another device on the network.

7. These are just general steps and the specifics of the implementation will depend on the specific IoT device and network configuration you want to create. Additionally, you may need to configure the router and the cloud device to enable Internet connectivity for the IoT device.

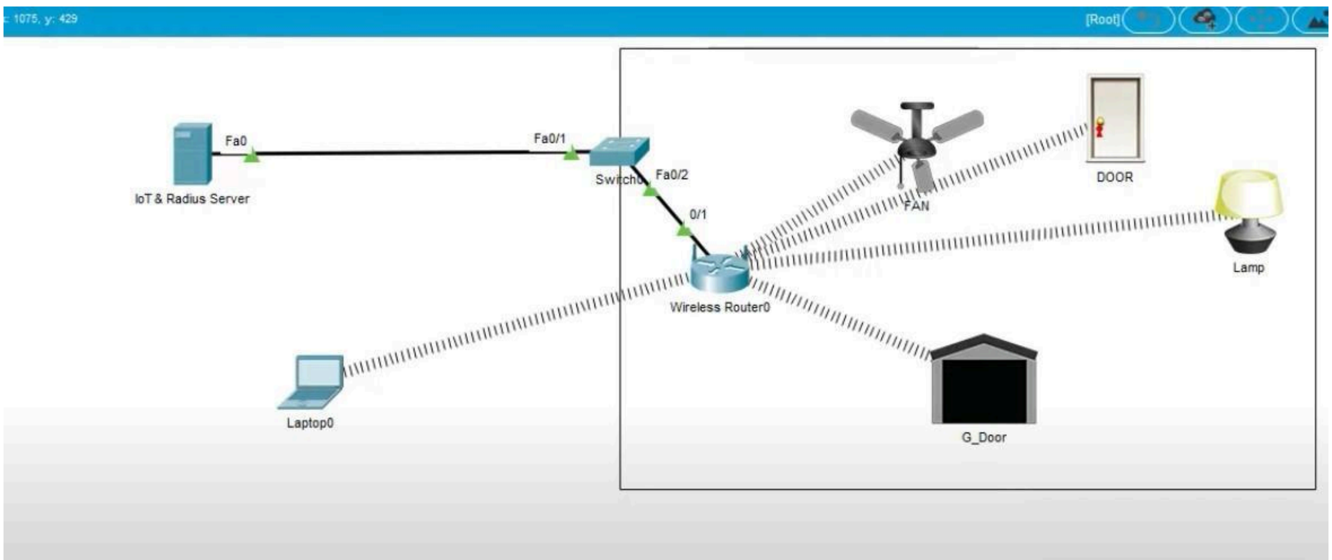


Result: Thus an IOT device in networking is implemented using Cisco Packet Tracer successfully.

Date:

EXPERIMENT: 24

IOT BASED SMART BUILDING USING WPA SECURITY AND RADIUS SERVER



Date:

EXPERIMENT: 25

TRANSPORT LAYER PROTOCOL HEADER ANALYSIS USING WIRE SHARK-TCP

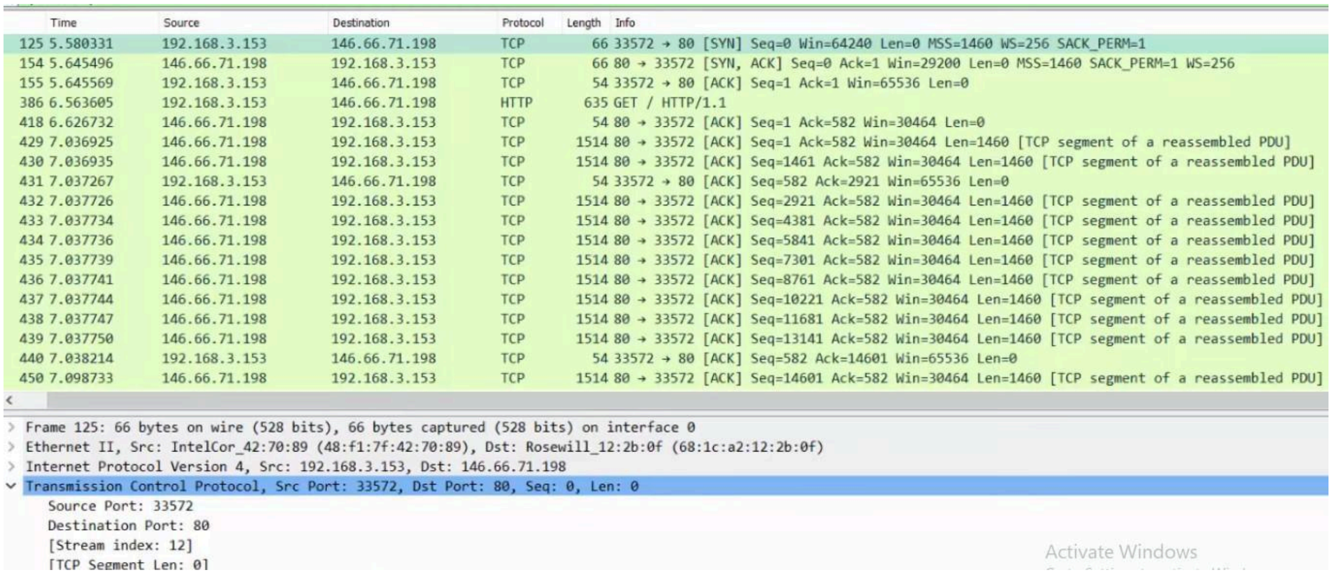
Aim: To analyze capturing of Transport layer protocol header analysis using Wire shark- TCP.

SOFTWARE USED:

Wire shark network analyzer

Procedure:

1. Open wire shark.
2. Click on list the available capture interface.
3. Choose the LAN interface.
4. Click on start button.
5. Active packets will be displayed.
6. Capture the packets & select any IP address from the source.
7. Click on the expression and select IPV4 IP addr source address in the field name.
8. Select the double equals (==) from the selection and enter the selected IP source address.
9. Click on the apply button.
10. All the packets will be filtered using the source address.



	Time	Source	Destination	Protocol	Length	Info
125	5.580331	192.168.3.153	146.66.71.198	TCP	66	33572 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
154	5.645496	146.66.71.198	192.168.3.153	TCP	66	80 → 33572 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=256
155	5.645569	192.168.3.153	146.66.71.198	TCP	54	33572 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
386	6.563605	192.168.3.153	146.66.71.198	HTTP	635	GET / HTTP/1.1
418	6.626732	146.66.71.198	192.168.3.153	TCP	54	80 → 33572 [ACK] Seq=1 Ack=582 Win=30464 Len=0
429	7.036925	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=1 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]
430	7.036935	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=1461 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]
431	7.037267	192.168.3.153	146.66.71.198	TCP	54	33572 → 80 [ACK] Seq=582 Ack=2921 Win=65536 Len=0
432	7.037726	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=2921 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]
433	7.037734	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=4381 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]
434	7.037736	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=5841 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]
435	7.037739	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=7301 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]
436	7.037741	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=8761 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]
437	7.037744	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=10221 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]
438	7.037747	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=11681 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]
439	7.037750	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=13141 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]
440	7.038214	192.168.3.153	146.66.71.198	TCP	54	33572 → 80 [ACK] Seq=582 Ack=14601 Win=65536 Len=0
450	7.098733	146.66.71.198	192.168.3.153	TCP	1514	80 → 33572 [ACK] Seq=14601 Ack=582 Win=30464 Len=1460 [TCP segment of a reassembled PDU]

```
> Frame 125: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: IntelCor_42:70:89 (48:f1:7f:42:70:89), Dst: Rosewill_12:2b:0f (68:1c:a2:12:2b:0f)
> Internet Protocol Version 4, Src: 192.168.3.153, Dst: 146.66.71.198
v Transmission Control Protocol, Src Port: 33572, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 33572
    Destination Port: 80
    [Stream index: 12]
    [TCP Segment Len: 0]
```


Result: Hence, the capturing of packets using wire shark network analyzer was analyzed for TCP.

Date:

EXPERIMENT: 26

TRANSPORT LAYER PROTOCOL HEADER ANALYSIS USING WIRESHARK-UDP

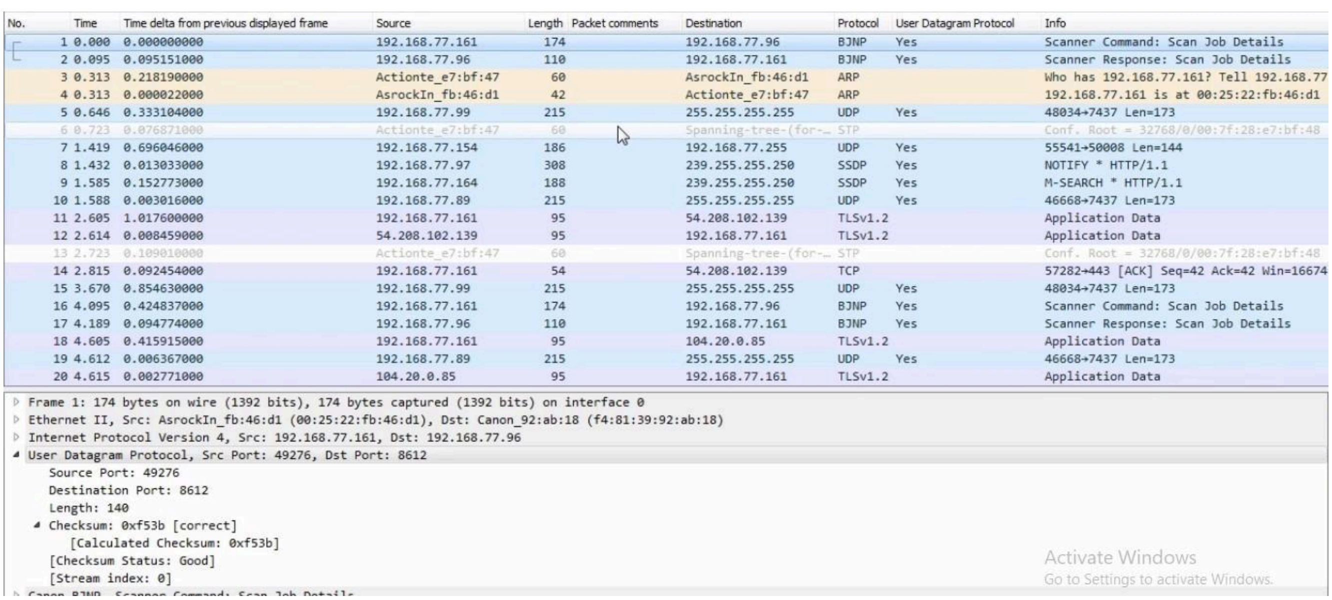
Aim: To analyze capturing of Transport layer protocol header analysis using Wire shark- UDP.

SOFTWARE USED:

Wire shark network analyzer

Procedure:

1. Open wire shark.
2. Click on list the available capture interface.
3. Choose the LAN interface.
4. Click on the start button.
5. Active packets will be displayed.
6. Capture the packets & select any IP address from the source.
7. Click on the expression and select IPV4 IP addr source address in the field name.
8. Select the double equals (==) from the selection and enter the selected IP source address.
9. Click on the apply button.
10. All the packets will be filtered using the source address.



No.	Time	Time delta from previous displayed frame	Source	Length	Packet comments	Destination	Protocol	User Datagram Protocol	Info
1	0.000	0.000000000	192.168.77.161	174		192.168.77.96	BJNP	Yes	Scanner Command: Scan Job Details
2	0.095	0.095151000	192.168.77.96	110		192.168.77.161	BJNP	Yes	Scanner Response: Scan Job Details
3	0.313	0.218190000	Actionte_e7:bf:47	60		AsrockIn_fb:46:d1	ARP		Who has 192.168.77.161? Tell 192.168.77
4	0.313	0.000022000	AsrockIn_fb:46:d1	42		Actionte_e7:bf:47	ARP		192.168.77.161 is at 00:25:22:fb:46:d1
5	0.646	0.333104000	192.168.77.99	215		255.255.255.255	UDP	Yes	48034→7437 Len=173
6	0.723	0.076871000	Actionte_e7:bf:47	60		Spanning-tree-(for-…	STP		Conf. Root = 32768/0/00:7f:28:e7:bf:48
7	1.419	0.696046000	192.168.77.154	186		192.168.77.255	UDP	Yes	55541→50008 Len=144
8	1.432	0.013033000	192.168.77.97	308		239.255.255.250	SSDP	Yes	NOTIFY * HTTP/1.1
9	1.585	0.152773000	192.168.77.164	188		239.255.255.250	SSDP	Yes	M-SEARCH * HTTP/1.1
10	1.588	0.003016000	192.168.77.89	215		255.255.255.255	UDP	Yes	46668→7437 Len=173
11	2.605	1.017600000	192.168.77.161	95		54.208.102.139	TLSv1.2		Application Data
12	2.614	0.008459000	54.208.102.139	95		192.168.77.161	TLSv1.2		Application Data
13	2.723	0.109010000	Actionte_e7:bf:47	60		Spanning-tree-(for-…	STP		Conf. Root = 32768/0/00:7f:28:e7:bf:48
14	2.815	0.092454000	192.168.77.161	54		54.208.102.139	TCP		57282→443 [ACK] Seq=42 Ack=42 Win=16674
15	3.670	0.854630000	192.168.77.99	215		255.255.255.255	UDP	Yes	48034→7437 Len=173
16	4.095	0.424837000	192.168.77.161	174		192.168.77.96	BJNP	Yes	Scanner Command: Scan Job Details
17	4.189	0.094774000	192.168.77.96	110		192.168.77.161	BJNP	Yes	Scanner Response: Scan Job Details
18	4.605	0.415915000	192.168.77.161	95		104.20.0.85	TLSv1.2		Application Data
19	4.612	0.006367000	192.168.77.89	215		255.255.255.255	UDP	Yes	46668→7437 Len=173
20	4.615	0.002771000	104.20.0.85	95		192.168.77.161	TLSv1.2		Application Data

Frame 1: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface 0
Ethernet II, Src: AsrockIn_fb:46:d1 (00:25:22:fb:46:d1), Dst: Canon_92:ab:18 (f4:81:39:92:ab:18)
Internet Protocol Version 4, Src: 192.168.77.161, Dst: 192.168.77.96
User Datagram Protocol, Src Port: 49276, Dst Port: 8612
Source Port: 49276
Destination Port: 8612
Length: 140
Checksum: 0xf53b [correct]
[Calculated Checksum: 0xf53b]
[Checksum Status: Good]
[Stream index: 0]

Result: Hence, the capturing of packets using wire shark network analyzer was analyzed for UDP.

Date:

EXPERIMENT-27

NETWORK LAYER PROTOCOL HEADER ANALYSIS USING WIRE SHARK – SMTP

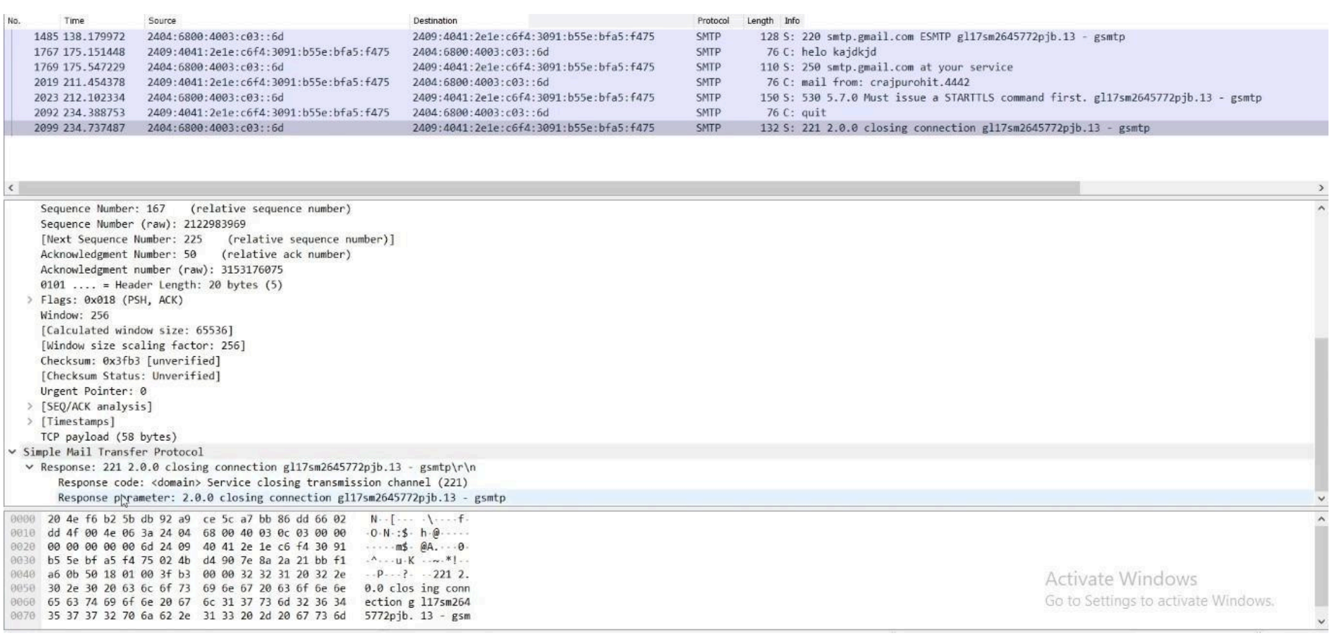
Aim: To analyze capturing of Transport layer protocol header analysis using Wire shark- SMTP.

SOFTWARE USED:

Wire shark network analyzer

Procedure:

1. Open wire shark.
2. Click on list the available capture interface.
3. Choose the LAN interface.
4. Click on start button.
5. Active packets will be displayed.
6. Capture the packets & select any IP address from the source.
7. Click on the expression and select IPV4 IP addr source address in the field name.
8. Select the double equals (==) from the selection and enter the selected IP source address.
9. Click on apply button.
10. All the packets will be filtered using source address.



Result: Hence, the capturing of packets using wire shark network analyzer was analyzed for SMTP.

Date:

EXPERIMENT-28

NETWORK LAYER PROTOCOL HEADER ANALYSIS USING WIRE SHARK – ICMP

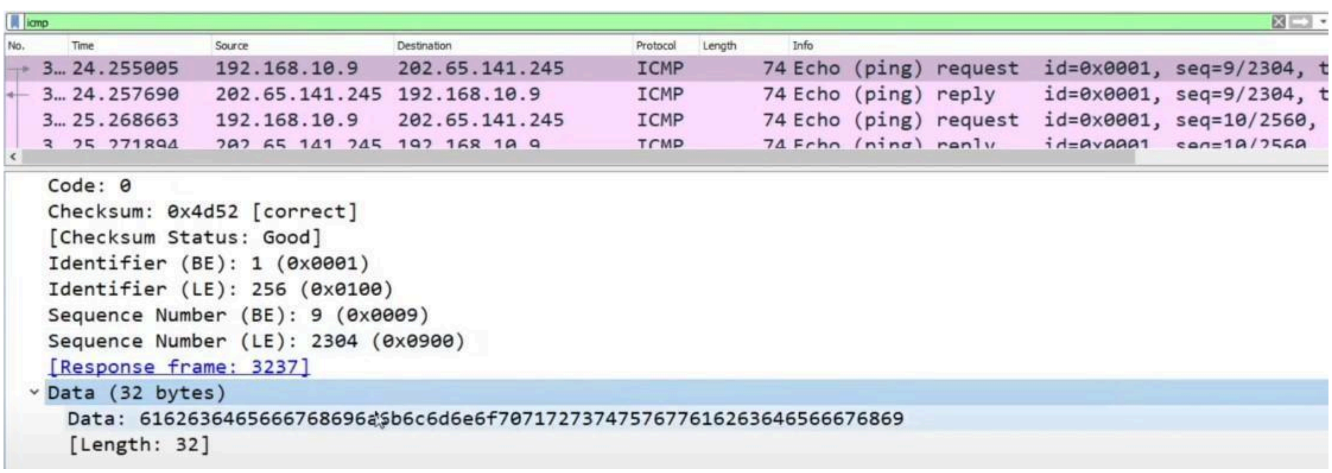
Aim: To analyze capturing of Transport layer protocol header analysis using Wire shark- ICMP.

SOFTWARE USED:

Wire shark network analyzer

Procedure:

1. Open wire shark.
2. Click on list the available capture interface.
3. Choose the LAN interface.
4. Click on start button.
5. Active packets will be displayed.
6. Capture the packets & select any IP address from the source.
7. Click on the expression and select IPV4 IP addr source address in the field name.
8. Select the double equals (==) from the selection and enter the selected IP source address.
9. Click on the apply button.
10. All the packets will be filtered using the source address.



Result: Hence, the capturing of packets using wire shark network analyzer was analyzed for ICMP.

Date:

EXPERIMENT-29

NETWORK LAYER PROTOCOL HEADER ANALYSIS USING WIRESHARK – ARP

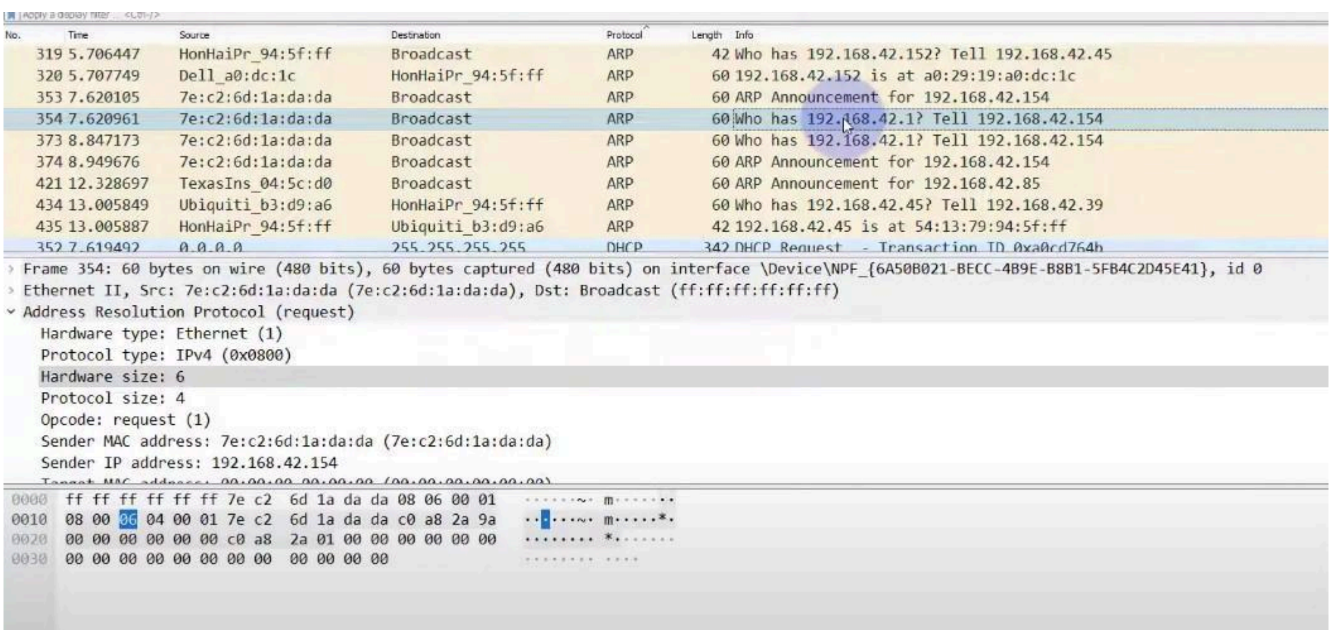
AIM: To analyze capturing of Transport layer protocol header analysis using Wire shark- ARP.

SOFTWARE USED:

Wire shark network analyzer

PROCEDURE:

1. Open wire shark.
2. Click on list the available capture interface.
3. Choose the LAN interface.
4. Click on the start button.
5. Active packets will be displayed.
6. Capture the packets & select any IP address from the source.
7. Click on the expression and select IPV4 IP addr source address in the field name.
8. Select the double equals (==) from the selection and enter the selected IP source address.
9. Click on the apply button.
10. All the packets will be filtered using the source address.



Result: Hence, the capturing of packets using wire shark network analyzer was analyzed for ARP.

Date:

EXPERIMENT-30

NETWORK LAYER PROTOCOL HEADER ANALYSIS USING WIRESHARK – HTTP

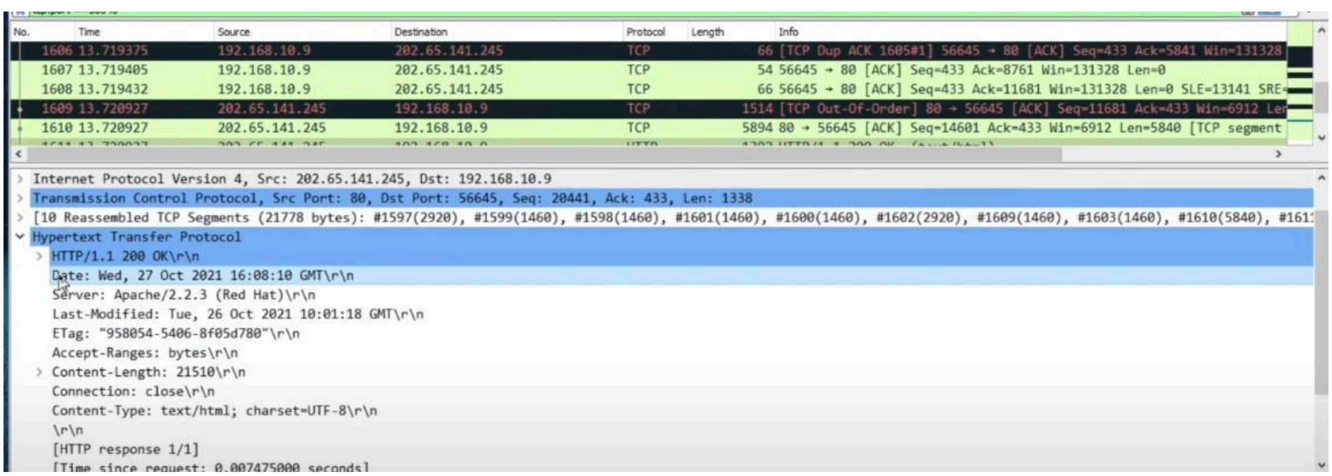
AIM: To analyze capturing of Transport layer protocol header analysis using Wire shark- HTTP.

SOFTWARE USED:

Wire shark network analyzer

PROCEDURE:

1. Open wire shark.
2. Click on list the available capture interface.
3. Choose the LAN interface.
4. Click on the start button.
5. Active packets will be displayed.
6. Capture the packets & select any IP address from the source.
7. Click on the expression and select IPV4 IP addr source address in the field name.
8. Select the double equals (==) from the selection and enter the selected IP source address.
9. Click on the apply button.
10. All the packets will be filtered using the source address.



Result: Hence, the capturing of packets using wire shark network analyzer was analyzed for HTTP.

Date:

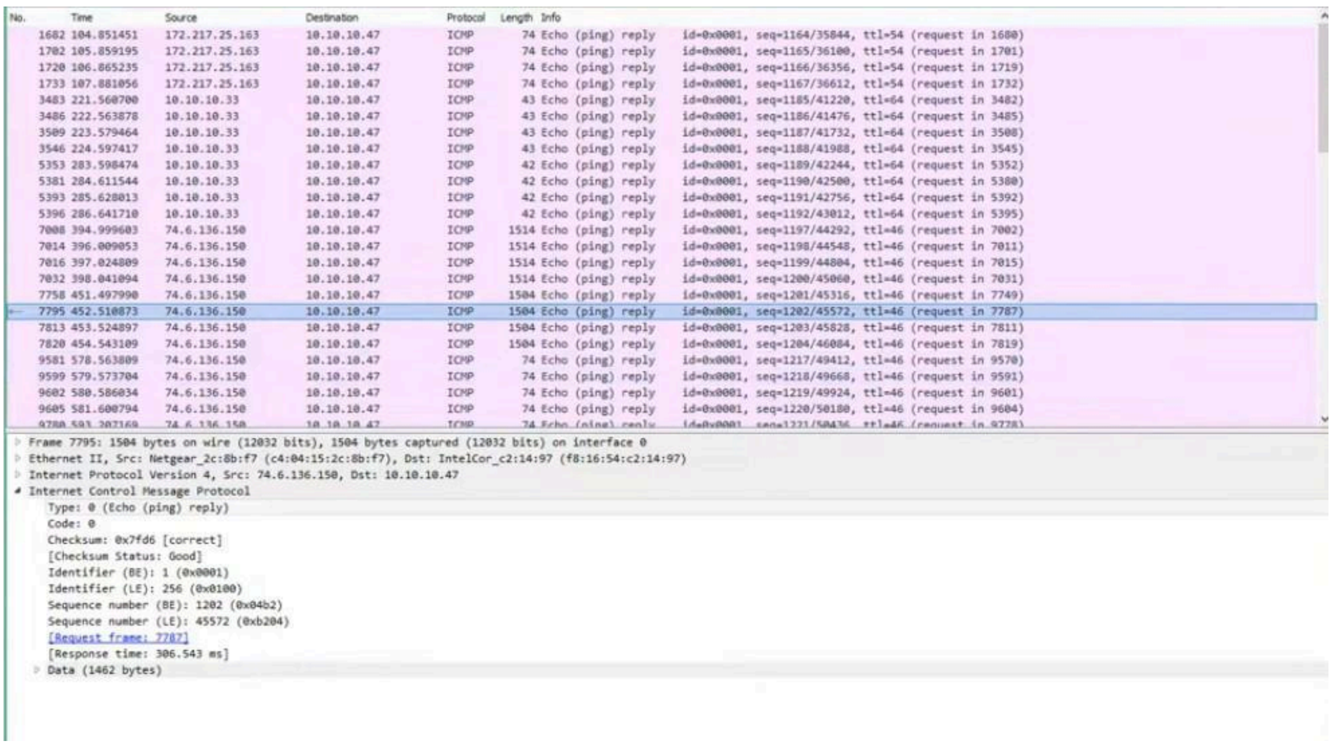
EXPERIMENT-31

Identify and monitor the IP, network address, Trace the router information, how to take remote system and check the node connection in network

Date:

EXPERIMENT-32

DEMONSTRATION OF PING OPERATION USING ICMP IN WIRESHARK



No.	Time	Source	Destination	Protocol	Length	Info
1682	104.851451	172.217.25.163	10.10.10.47	ICMP	74	Echo (ping) reply id=0x0001, seq=1164/35844, ttl=54 (request in 1680)
1702	105.859195	172.217.25.163	10.10.10.47	ICMP	74	Echo (ping) reply id=0x0001, seq=1165/36100, ttl=54 (request in 1701)
1720	106.865235	172.217.25.163	10.10.10.47	ICMP	74	Echo (ping) reply id=0x0001, seq=1166/36356, ttl=54 (request in 1719)
1733	107.881056	172.217.25.163	10.10.10.47	ICMP	74	Echo (ping) reply id=0x0001, seq=1167/36612, ttl=54 (request in 1732)
3483	221.560700	10.10.10.33	10.10.10.47	ICMP	43	Echo (ping) reply id=0x0001, seq=1185/41220, ttl=64 (request in 3482)
3486	222.563878	10.10.10.33	10.10.10.47	ICMP	43	Echo (ping) reply id=0x0001, seq=1186/41476, ttl=64 (request in 3485)
3509	223.579464	10.10.10.33	10.10.10.47	ICMP	43	Echo (ping) reply id=0x0001, seq=1187/41732, ttl=64 (request in 3508)
3546	224.597417	10.10.10.33	10.10.10.47	ICMP	43	Echo (ping) reply id=0x0001, seq=1188/41988, ttl=64 (request in 3545)
5353	283.598474	10.10.10.33	10.10.10.47	ICMP	42	Echo (ping) reply id=0x0001, seq=1189/42244, ttl=64 (request in 5352)
5381	284.611544	10.10.10.33	10.10.10.47	ICMP	42	Echo (ping) reply id=0x0001, seq=1190/42500, ttl=64 (request in 5380)
5393	285.628013	10.10.10.33	10.10.10.47	ICMP	42	Echo (ping) reply id=0x0001, seq=1191/42756, ttl=64 (request in 5392)
5396	286.641710	10.10.10.33	10.10.10.47	ICMP	42	Echo (ping) reply id=0x0001, seq=1192/43012, ttl=64 (request in 5395)
7008	394.999603	74.6.136.150	10.10.10.47	ICMP	1514	Echo (ping) reply id=0x0001, seq=1197/44292, ttl=46 (request in 7002)
7014	396.009053	74.6.136.150	10.10.10.47	ICMP	1514	Echo (ping) reply id=0x0001, seq=1198/44548, ttl=46 (request in 7011)
7016	397.024809	74.6.136.150	10.10.10.47	ICMP	1514	Echo (ping) reply id=0x0001, seq=1199/44804, ttl=46 (request in 7015)
7032	398.041094	74.6.136.150	10.10.10.47	ICMP	1514	Echo (ping) reply id=0x0001, seq=1200/45060, ttl=46 (request in 7031)
7758	451.497990	74.6.136.150	10.10.10.47	ICMP	1504	Echo (ping) reply id=0x0001, seq=1201/45316, ttl=46 (request in 7749)
7795	452.510873	74.6.136.150	10.10.10.47	ICMP	1504	Echo (ping) reply id=0x0001, seq=1202/45572, ttl=46 (request in 7787)
7813	453.524897	74.6.136.150	10.10.10.47	ICMP	1504	Echo (ping) reply id=0x0001, seq=1203/45828, ttl=46 (request in 7811)
7820	454.543109	74.6.136.150	10.10.10.47	ICMP	1504	Echo (ping) reply id=0x0001, seq=1204/46084, ttl=46 (request in 7819)
9581	578.563809	74.6.136.150	10.10.10.47	ICMP	74	Echo (ping) reply id=0x0001, seq=1217/49412, ttl=46 (request in 9570)
9599	579.573704	74.6.136.150	10.10.10.47	ICMP	74	Echo (ping) reply id=0x0001, seq=1218/49668, ttl=46 (request in 9591)
9602	580.586034	74.6.136.150	10.10.10.47	ICMP	74	Echo (ping) reply id=0x0001, seq=1219/49924, ttl=46 (request in 9601)
9605	581.600794	74.6.136.150	10.10.10.47	ICMP	74	Echo (ping) reply id=0x0001, seq=1220/50180, ttl=46 (request in 9604)

Frame 7795: 1504 bytes on wire (12032 bits), 1504 bytes captured (12032 bits) on interface 0
Ethernet II, Src: Netgear_2c:8b:f7 (c4:04:15:2c:8b:f7), Dst: IntelCor_c2:14:97 (f8:16:54:c2:14:97)
Internet Protocol Version 4, Src: 74.6.136.150, Dst: 10.10.10.47
Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0x7fd6 [correct]
[Checksum Status: Good]
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence number (BE): 1202 (0x04b2)
Sequence number (LE): 45572 (0xb204)
[Request frame: 7787]
[Response time: 306.543 ms]
Data (1462 bytes)

Date:

EXPERIMENT: 33

IMPLEMENTATION OF BIT STUFFING MECHANISM USING C

Aim: To implement Bit stuffing mechanism using C program.

Bit suffering: It is a technique used in communication systems to prevent data loss or corruption during transmission. It involves inserting one or more extra bits into a data packet to differentiate it from the control characters. Bit suffering is implemented using bitwise operators in the C programming language. In this code, the `bit Stuffing` function takes an input byte array, its length, an output byte array, and a pointer to the output length variable. It performs bit stuffing on the input data and stores the stuffed data in the output array. The main logic of the bit stuffing is implemented using bitwise operations. The input data is processed byte by byte, and each bit is checked for consecutive 1's. If five consecutive 1's are found, a 0 bit is stuffed into the output frame. The flag sequence (01111110) is added at the beginning and end of the output frame. In the `main` function, an example input frame is provided, and the bit stuffing is performed by calling the `bit Stuffing` function. The input and output frames are then printed for verification. Note that in this example, the input frame is hard-coded, and the output frame is printed in hexadecimal format for better readability. You can modify the input frame and test the code with different inputs.

code:

```
#include <stdio.h>

void bitStuffing(int input[], int n) {
   int output[50], j = 0, count = 0;
```

```
    for (int i = 0; i < n; i++) {
        output[j++] = input[i];
        count = (input[i] == 1) ? count + 1 : 0;

        if (count == 5) {
            output[j++] = 0;
            count = 0;
        }
    }

    printf("Stuffed Bit Stream: ");
    for (int i = 0; i < j; i++) {
        printf("%d", output[i]);
    }
    printf("\n");
}

int main() {
    int input[20], n;

    printf("Enter number of bits: ");
    scanf("%d", &n);

    printf("Enter bit stream: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", input[i]);
    }

    bitStuffing(input, n);
    return 0;
}
```

<u>OUTPUT:</u>

Enter frame size (Example: 8):12

Enter the frame in the form of 0 and 1 :0 1 0 1 1 1 1 1 1 0 0 1

After Bit Stuffing :01011111**0**1001

Result : Therefore bit suffering mechanism has been successfully implemented using c program.

Date:

EXPERIMENT-34

IMPLEMENTATION OF SERVER – CLIENT USING TCP SOCKET PROGRAMMING

server code:

```
#include <stdio.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h> // read(), write(), close()

#define MAX 80

#define PORT 8080

#define SA struct sockaddr
```

```
// Function designed for chat between client and server.
void func(int connfd)
{
        char buff[MAX];
        int n;
        // infinite loop for chat
        for (;;) {
    bzero(buff, MAX);

        // read the message from client and copy it in buffer
    read(connfd, buff, sizeof(buff));
        // print buffer which contains the client contents
    printf("From client: %s\t To client : ", buff);
    bzero(buff, MAX);
        n = 0;
        // copy server message in the buffer
         while ((buff[n++] = getchar()) != '\n')
        ;

        // and send that buffer to client
    write(connfd, buff, sizeof(buff));

        // if msg contains "Exit" then server exit and chat ends.
```

```
        if (strncmp("exit", buff, 4) == 0) {
        printf("Server Exit...\n");
        break;
        }
        }
}

// Driver function
int main()
{
        int sockfd, connfd, len;
        struct sockaddr_in servaddr, cli;

        // socket create and verification
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd == -1) {
      printf("socket creation failed...\n");
        exit(0);
        }
        else
      printf("Socket successfully created..\n");
   bzero(servaddr, sizeof(servaddr));

        // assign IP, PORT
```

```
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);

    // Binding newly created socket to given IP and verification
    if ((bind(sockfd, (SA*)servaddr, sizeof(servaddr))) != 0) {
  printf("socket bind failed...\n");
    exit(0);
    }
    else
  printf("Socket successfully binded..\n");

    // Now server is ready to listen and verification
    if ((listen(sockfd, 5)) != 0) {
  printf("Listen failed...\n");
    exit(0);
    }
    else
  printf("Server listening..\n");
    len = sizeof(cli);

    // Accept the data packet from client and verification
    connfd = accept(sockfd, (SA*)&cli, &len);
    if (connfd < 0) {
```

```
            printf("server accept failed...\n");
            exit(0);
            }
            else
        printf("server accept the client...\n");

            // Function for chatting between client and server
    func(connfd);

            // After chatting close the socket
            close(sockfd);
}
```

client code:

```
#include <arpa/inet.h> // inet_addr()
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h> // bzero()
#include <sys/socket.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
```

```
#define SA struct sockaddr
void func(int sockfd)
{
	char buff[MAX];
	int n;
	for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
	n = 0;
	while ((buff[n++] = getchar()) != '\n')
	;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
	if ((strncmp(buff, "exit", 4)) == 0) {
	printf("Client Exit...\n");
	break;
	}
	}
}

int main()
{
```

```
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
    printf("socket creation failed...\n");
    exit(0);
    }
    else
  printf("Socket successfully created..\n");
bzero(servaddr, sizeof(servaddr));

    // assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
servaddr.sin_port = htons(PORT);

    // connect the client socket to server socket
    if (connect(sockfd, (SA*)servaddr, sizeof(servaddr))
    != 0) {
  printf("connection with the server failed...\n");
    exit(0);
    }
```

```
        else
    printf("connected to the server..\n");

        // function for chat
    func(sockfd);

        // close the socket
    close(sockfd);
}
```

Output –

Server side:

```
Socket successfully created..
Socket successfully binded..
Server listening..

server accepts the client...
 From client: hi
         To client : hello
 From client: exit
         To client : exit
 Server Exit...
```

Client side:

```
Socket successfully created..
 connected to the server..
 Enter the string : hi
 From Server : hello
 Enter the string : exit
 From Server : exit
 Client Exit...
```

Date:

EXPERIMENT-35

IMPLEMENTATION OF SERVER – CLIENT USING UDP SOCKET PROGRAMMING

server side:

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>


#define PORT 8080

#define MAXLINE 1024


// Driver code

int main() {

	int sockfd;

	char buffer[MAXLINE];

	char *hello = "Hello from server";
```

```
struct sockaddr_in servaddr, cliaddr;

// Creating socket file descriptor
if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
}

memset(servaddr, 0, sizeof(servaddr));
memset(ciaddr, 0, sizeof(cliaddr));

// Filling server information
servaddr.sin_family = AF_INET; // IPv4
servaddr.sin_addr.s_addr = INADDR_ANY;
servaddr.sin_port = htons(PORT);

// Bind the socket with the server address
if ( bind(sockfd, (const struct sockaddr *)servaddr,
                sizeof(servaddr)) < 0 )
{
        perror("bind failed");
        exit(EXIT_FAILURE);
}
```

```
        int len, n;

        len = sizeof(cliaddr); //len is value/result

        n = recvfrom(sockfd, (char *)buffer, MAXLINE,
                                MSG_WAITALL, ( struct sockaddr *) ciaddr,
                                &len);
        buffer[n] = '\0';
        printf("Client : %s\n", buffer);
        sendto(sockfd, (const char *)hello, strlen(hello),
                MSG_CONFIRM, (const struct sockaddr *) ciaddr,
                        len);
        printf("Hello message sent.\n");

        return 0;
}
```

client side:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
```

```
#include <netinet/in.h>

#define PORT 8080
#define MAXLINE 1024

// Driver code
int main() {
	int sockfd;
	char buffer[MAXLINE];
	char *hello = "Hello from client";
	struct sockaddr_in	servaddr;

	// Creating socket file descriptor
	if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
		perror("socket creation failed");
		exit(EXIT_FAILURE);
	}

	memset(servaddr, 0, sizeof(servaddr));

	// Filling server information
	servaddr.sin_family = AF_INET;
	servaddr.sin_port = htons(PORT);
	servaddr.sin_addr.s_addr = INADDR_ANY;
```

```
	int n, len;

	sendto(sockfd, (const char *)hello, strlen(hello),
		MSG_CONFIRM, (const struct sockaddr *) servaddr,
			sizeof(servaddr));
	printf("Hello message sent.\n");

	n = recvfrom(sockfd, (char *)buffer, MAXLINE,
				MSG_WAITALL, (struct sockaddr *) servaddr,
				&len);
	buffer[n] = '\0';
	printf("Server : %s\n", buffer);

	close(sockfd);
	return 0;
}
```