

# RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution

Affiliated to Anna University, Chennai,  
Rajalakshmi Nagar, Thandalam – 602 105



## RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution  
Affiliated to ANNA UNIVERSITY, Chennai

### DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

CS19541 - COMPUTER NETWORKS

Laboratory Record Note Book

Name : TARUN SAI N S

Register No. : 2116221501159

Year / Branch / Section : III AI&ML - C

Semester : V

Academic Year: 2024-2025

**RAJALAKSHMI ENGINEERING COLLEGE**

An Autonomous Institution

Affiliated to Anna University, Chennai,  
Rajalakshmi Nagar, Thandalam – 602 105

**BONAFIDE CERTIFICATE**

Name: TARUN SAI N S

Academic Year: 2024-25 Semester: V Branch: AI&ML

Reg. No: 2116221501159

Certified that this is the bonafide record of work done by the above student in  
the CS19541 – COMPUTER NETWORKS Laboratory during the academic year  
2024- 2025

Signature of Faculty in-charge

Submitted for the Practical Examination held on.....

Internal Examiner

External Examiner

**EX: 1****BASIC NETWORK COMMANDS****AIM:**

To study the various Network commands used in Linux and Windows

**BASIC NETWORK COMMANDS:**

- arp -a:
  - Address Resolution Protocol (ARP) shows IP address of Computer along with ip and Mac address of router

```
C:\Users\vijay\221501172>arp -a
```

Internet Address	Physical Address	Type
192.168.31.1	4c-82-a9-d6-49-79	dynamic
224.0.0.22	01-00-5e-00-00-16	static
224.0.0.251	01-00-5e-00-00-fb	static
224.0.0.252	01-00-5e-00-00-fc	static
239.255.255.250	01-00-5e-7f-ff-fa	static
255.255.255.255	ff-ff-ff-ff-ff-ff	static

- hostname:
  - Displays the name of the computer

```
C:\Users\vijay\221501172>hostname
Vijay
```

- ping
  - Packet Internet Groping testing connectivity

```
C:\Users\vijay\221501172>ping www.google.com

Pinging www.google.com [2404:6800:4007:819::2004] with 32 bytes of data:
Reply from 2404:6800:4007:819::2004: time=32ms
Reply from 2404:6800:4007:819::2004: time=33ms
Reply from 2404:6800:4007:819::2004: time=31ms
Reply from 2404:6800:4007:819::2004: time=33ms

Ping statistics for 2404:6800:4007:819::2004:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 31ms, Maximum = 33ms, Average = 32ms
```

- netstat
  - Network statistics monitors all the connections

```
C:\Users\vijay\221501172>netstat

Active Connections

  Proto  Local Address        Foreign Address      State
  TCP    127.0.0.1:49670      Vijay:49671        ESTABLISHED
  TCP    127.0.0.1:49671      Vijay:49670        ESTABLISHED
  TCP    127.0.0.1:49672      Vijay:49673        ESTABLISHED
  TCP    127.0.0.1:49673      Vijay:49672        ESTABLISHED
  TCP    127.0.0.1:52310      Vijay:52313        ESTABLISHED
  TCP    127.0.0.1:52313      Vijay:52310        ESTABLISHED
  TCP    192.168.31.48:49414   20.198.119.143:https ESTABLISHED
  TCP    192.168.31.48:54868   4.195.14.14:https   ESTABLISHED
  TCP    192.168.31.48:55487   20.212.88.117:https ESTABLISHED
  TCP    [2409:40f4:1128:928:5ffc:4a5b:956d:387e]:55461  [2603:1040:a06:6::1]:https ESTABLISHED
  TCP    [2409:40f4:1128:928:5ffc:4a5b:956d:387e]:55469  [64:ff9b::22e1:a5be]:https ESTABLISHED
  TCP    [2409:40f4:1128:928:5ffc:4a5b:956d:387e]:55486  dns:https           ESTABLISHED
  TCP    [2409:40f4:1128:928:5ffc:4a5b:956d:387e]:55495  [64:ff9b::142a:4919]:https TIME_WAIT
```

- ipconfig
  - Displays the detailed configuration of about all the connections in a system

```
C:\Users\vijay\221501172>ipconfig
Windows IP Configuration

Ethernet adapter Ethernet:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Ethernet adapter Ethernet 2:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Unknown adapter Local Area Connection:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 1:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 2:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:
  Connection-specific DNS Suffix . : lan
  IPv6 Address . . . . . : 2409:40f4:1128:928:5ffc:4a5b:956d:387e
  Link-local IPv6 Address . . . . . : fe80::963:1a4d:7607:2937%7
  IPv4 Address . . . . . : 192.168.31.48
```

- nslookup
  - Name server lookup to perform DNS lookups

```
C:\Users\vijay\221501172>nslookup
Default Server: jiofiber.local.html
Address: 192.168.31.1

> www.google.com
Server: jiofiber.local.html
Address: 192.168.31.1

Non-authoritative answer:
Name: www.google.com
Addresses: 2404:6800:4007:819::2004
           142.250.196.36
```

- route
  - Shows / Manipulates the IP routing table

```
C:\Users\vijay\221501172>route print
=====
Interface List
 16...5c 60 ba 3b 43 15 .....Realtek PCIe GbE Family Controller
 14...00 ff c4 74 de 3c .....ExpressVPN TAP Adapter
 13.....ExpressVPN TUN Driver
 15...32 03 c8 42 72 2f .....Microsoft Wi-Fi Direct Virtual Adapter
 20...b2 03 c8 42 72 2f .....Microsoft Wi-Fi Direct Virtual Adapter #2
 7...30 03 c8 42 72 2f .....Realtek RTL8822CE 802.11ac PCIe Adapter
 1.....Software Loopback Interface 1
=====
```

## Result:

Thus the basic network commands have been executed successfully.

<b>EX: 2</b>	<b>DIFFERENT TYPES OF NETWORK CABLES</b>
--------------	--

### **AIM:**

To study the different types of network cable and their specializations

### **TYPES OF NETWORK CABLES:**

#### **→ Coaxial Cable**

- ◆ It contains a conductor, insulator, braiding, and sheath which covers each other from the center respectively.
- ◆ There are two types based on the number of cores present. They are

Single Core Coaxial Cable

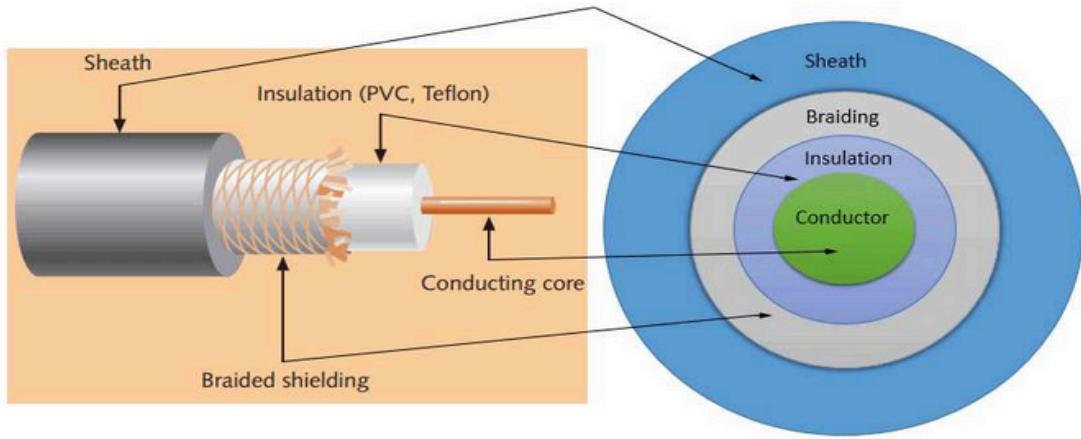
Multi Core Coaxial Cable

- ◆ Some of the few Coaxial Cable used in Computer Networks are,  
**RG-6** - Used in cable networks to provide cable Internet service and cable TV over long distances.

**RG-8** - Used in the earliest computer networks. This cable was used as the backbone cable in the bus topology. In Ethernet standards, this cable is documented as the 10base5 Thicknet cable.

**RG-58** - This cable is thinner, easier to handle and install than the RG-8 cable. This cable was used to connect a system with the backbone cable. In Ethernet standards, this cable is documented as the 10base2 Thinnet cable.

**RG-59** - Used in cable networks to provide short-distance service.



**Single core coaxial cable**



**Multi-core coaxial cable**

#### → Twisted pair cable

- ◆ It has been primarily developed for computer networks commonly known as Ethernet Cable. It is commonly used for most of the Local Area Network (LAN)
- ◆ This cable consists of color-coded pairs of insulated copper wires. Every two wires are twisted around each other to form a pair. Usually, there are four pairs. Each pair has one solid color and one striped color wire. Solid colors are blue, brown, green,

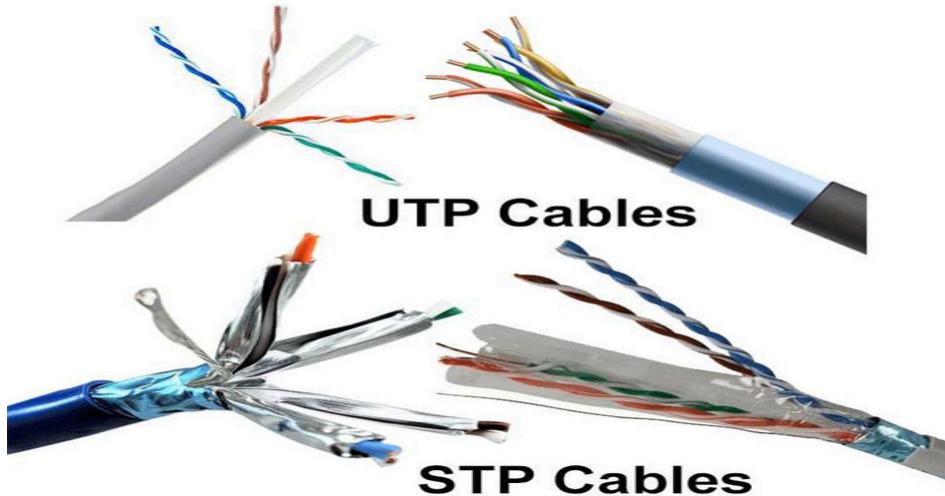
and orange. In striped color, the solid color is mixed with the white color.

- ◆ Based on how pairs striped in the sheath , there are two types they are,

UTP (Unshielded Twisted Pair)

STP (Shielded Twisted Pair)

- ◆ Some of the Twisted Pair Cables are cat1, cat2, cat3, cat4, cat5, cat5e, cat6, cat6a, cat7 among them cat5e, cat6, cat6a are commonly used.

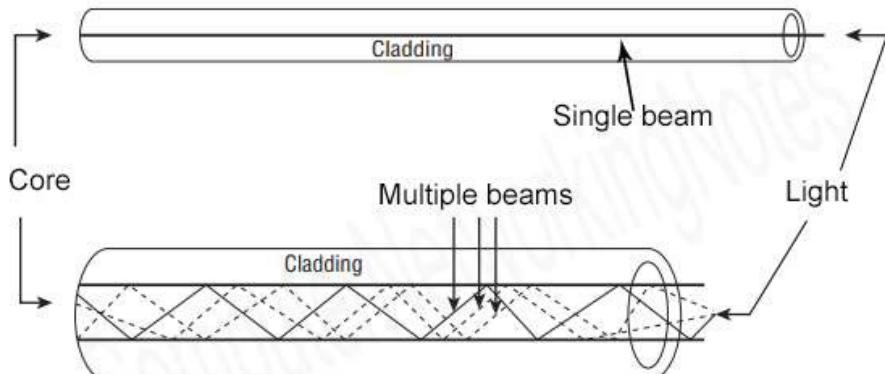


## → Fiber cable

- ◆ It consists of core, cladding, buffer and jacket.
  - ◆ Core carries the data signals in the form of light.
  - ◆ Cladding reflects light back to the core.
  - ◆ Buffer protects the light from leaking.
  - ◆ The jacket protects the cable from physical damage.

- ◆ Fiber optic cable is completely immune to EMI (Electromagnetic Interference) and RFI (Radio Frequency Interference)
- ◆ Based on how many beams of light transmit there are two types they are,
  - ◆ **SMF (Single mode Fiber) Optical Cable**
  - ◆ **MMF (Multi mode Fiber) Optical Cable**

### **SMF (Single mode fiber) optical cable**



### **MMF (multi-mode fiber) optical cable**

#### **◆ SMF (Single-mode fiber) optical cable**

This cable carries only a single beam of light. This is more reliable and supports much higher bandwidth and longer distances than the MMF cable. This cable uses a laser as the light source and transmits 1300 or 1550 nano-meter wavelengths of light.

#### **◆ MMF (multi-mode fiber) optical cable**

This cable carries multiple beams of light. Because of multiple beams, this cable carries much more data than the SMF cable.

This cable is used for shorter distances. This cable uses an LED as the light source and transmits 850 or 1300 nano-meter wavelengths of light.



### **Result:**

Thus the different types of cables has been studied.

### a. Implementation of Echo Message

**Aim:**

To implement a TCP echo server-client communication where the client sends a message, and the server echoes it back.

**Procedure:**

- **Server:**
  - Create a socket and bind it to a port.
  - Listen for incoming connections.
  - Accept a connection, receive a message, and send it back.
  - Stop the server when "end" is received.
- **Client:**
  - Connect to the server.
  - Send a message and receive the echoed response.
  - Close the connection.

**Program Code:****Server:**

```
from socket import *
s = socket(AF_INET, SOCK_STREAM)
s.bind(("" , 8000))
s.listen(5)
while True:
    c, a = s.accept()
    print("Received connection from", a)
    data = c.recv(100).decode()
    print(data)
    c.send(data.encode('utf-8'))
    if data.lower() == 'end':
        break
c.close()
```

## **Client:**

```
from socket import *

s = socket(AF_INET, SOCK_STREAM)
s.connect(("127.0.0.1", 8000))
op = input("Enter a message: ")
s.send(op.encode('utf-8'))
data = s.recv(100).decode()
print(data)
s.close()
```

## **Output:**

### **Server Output:**

```
Received connection from ('127.0.0.1', 56321)
```

```
Hello Server
```

```
Received connection from ('127.0.0.1', 56322)
```

```
Python Sockets are fun!
```

```
Received connection from ('127.0.0.1', 56323)
```

```
end
```

### **Client Outputs:**

#### **Client 1:**

```
Enter a message: Hello Server
```

```
Hello Server
```

**Client 2:**

Enter a message: Python Sockets are fun!

Python Sockets are fun!

**Client 3 (terminates server):**

Enter a message: end

end

**Result:**

Successfully implemented TCP echo communication.

## **b.Implementation of Chat Communication**

### **Aim:**

To implement a TCP chat application for server-client communication.

### **Procedure/Algorithm:**

- **Server:**

- Create a socket, bind it, and listen for incoming connections.
- Accept a connection, and communicate by sending and receiving messages in a loop.
- Stop communication if "end" is received.

- **Client:**

- Connect to the server.
- Continuously send and receive messages.

### **Program Code:**

#### **Server:**

```
from socket import *
s = socket(AF_INET, SOCK_STREAM)
s.bind((" ", 8000))
s.listen(5)
cont = 'yes'
while cont == 'yes':
    c, a = s.accept()
    print("Received connection from", a)
    while True:
        data = c.recv(100).decode()
        print(data)
        res = input("Enter Reply: ")
        c.send(res.encode('utf-8'))
        if data.lower() == 'end' or res.lower() == 'end':
            break
    cont = input("Receive Connection (yes or no): ")
c.close()
```

## **Client:**

```
from socket import *

s = socket(AF_INET, SOCK_STREAM)
s.connect(("127.0.0.1", 8000))

while True:
    message = input("Enter a message: ")
    s.send(message.encode('utf-8'))
    if message.lower() == 'end':
        break
    reply = s.recv(100).decode()
    print(reply)

s.close()
```

## **Output:**

### **Server Output**

```
Received connection from ('127.0.0.1', 56321)
```

```
Hello Server
```

```
Enter Reply: Hi Client!
```

```
Python Sockets are fun!
```

```
Enter Reply: Yes, they are! What are you learning?
```

```
end
```

```
Enter Reply: Goodbye!
```

```
Receive Connection (yes or no): yes
```

```
Received connection from ('127.0.0.1', 56322)
```

```
Hello Again!
```

```
Enter Reply: Welcome back! Ready for more socket programming?
```

```
end
```

```
Enter Reply: Bye!
```

```
Receive Connection (yes or no): no
```

**Client Output (Client 1)**

Enter a message: Hello Server

Hi Client!

Enter a message: Python Sockets are fun!

Yes, they are! What are you learning?

Enter a message: end

Goodbye!

**Client Output (Client 2)**

Enter a message: Hello Again!

Welcome back! Ready for more socket programming?

Enter a message: end

Bye!

**Result:**

Successfully implemented TCP chat communication.

**a. Implementation of Echo Message****Aim:**

To implement a UDP echo server-client communication where the client sends a message, and the server echoes it back.

**Procedure/Algorithm:****Server:**

Create a socket and bind it to a port.

Receive a message, and send it back to the client.

Stop the server when "end" is received.

**Client:**

Send a message to the server and receive the echoed response.

Close the connection when "end" is sent.

**Program Code:****Server:**

```
import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('', 12500))
while True:
    message, address = server_socket.recvfrom(1024)
    message = message.decode()
    print(message)
    server_socket.sendto(message.encode('utf-8'), address)
    if message == 'end':
        break
server_socket.close()
```

## **Client:**

```
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
addr = ("127.0.0.1", 12500)
message = input("Enter Message: ")
client_socket.sendto(message.encode('utf-8'), addr)
data, server = client_socket.recvfrom(1024)
print(data.decode())
if message.lower() == 'end':
    print("Connection Closed....")
```

## **Output:**

### **Server Output**

Hello Server

Python Sockets over UDP!

end

### **Client Outputs**

#### **Client 1:**

Enter Message: Hello Server

Hello Server

#### **Client 2:**

Enter Message: Python Sockets over UDP!

Python Sockets over UDP!

**Client 3 (terminates communication):**

Enter Message: end

end

Connection Closed.....

**Result:**

Successfully implemented UDP echo communication.

## **b. Implementation of Chat communication**

### **Aim:**

To implement a UDP chat application for server-client communication.

### **Procedure/Algorithm:**

#### **Server:**

Create a socket and bind it to a port.

Continuously send and receive messages.

Stop the communication if "end" is received.

#### **Client:**

Send messages to the server and receive replies in a loop.

### **Program Code:**

#### **Server:**

```
import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('', 12500))
cont = 'no'
while cont == 'no':
    while True:
        message, address = server_socket.recvfrom(1024)
        print("Received Message from ", address)
        message = message.decode()
        print(message)
        res = input("Enter Response: ")
        server_socket.sendto(res.encode('utf-8'), address)
        if message == 'end' or res == 'end':
            break
    cont = input("Break Connection: ")
server_socket.close()
```

## **Client:**

```
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
addr = ("127.0.0.1", 12500)
while True:
    message = input("Enter Message: ")
    client_socket.sendto(message.encode('utf-8'), addr)
    if message.lower() == 'end':
        print("Connection Closed....")
        break
    data, server = client_socket.recvfrom(1024)
    print(data.decode())
```

## **Output:**

### **Server Output**

Received Message from ('127.0.0.1', 56321)

Hello Server

Enter Response: Hi Client!

Received Message from ('127.0.0.1', 56321)

How are you?

Enter Response: I'm good. How about you?

Received Message from ('127.0.0.1', 56321)

end

Enter Response: Goodbye!

Break Connection: yes

### **Client Output**

Enter Message: Hello Server

Hi Client!

Enter Message: How are you?

I'm good. How about you?

Enter Message: end

Goodbye!

Connection Closed....

### **Result:**

Successfully implemented UDP chat communication.

# **PACKET TRACKING USING WIRESHARK**

## **AIM:**

To study and understand the working and usage of Packet Capturing Tool: Wireshark and perform different tasks with the help of it.

## **WIRESHARK:**

Wireshark is a popular open-source network protocol analyzer. It allows users to capture and examine data packets traveling across a network. This can be incredibly useful for troubleshooting network issues, analyzing network performance, and ensuring security.

With Wireshark, you can:

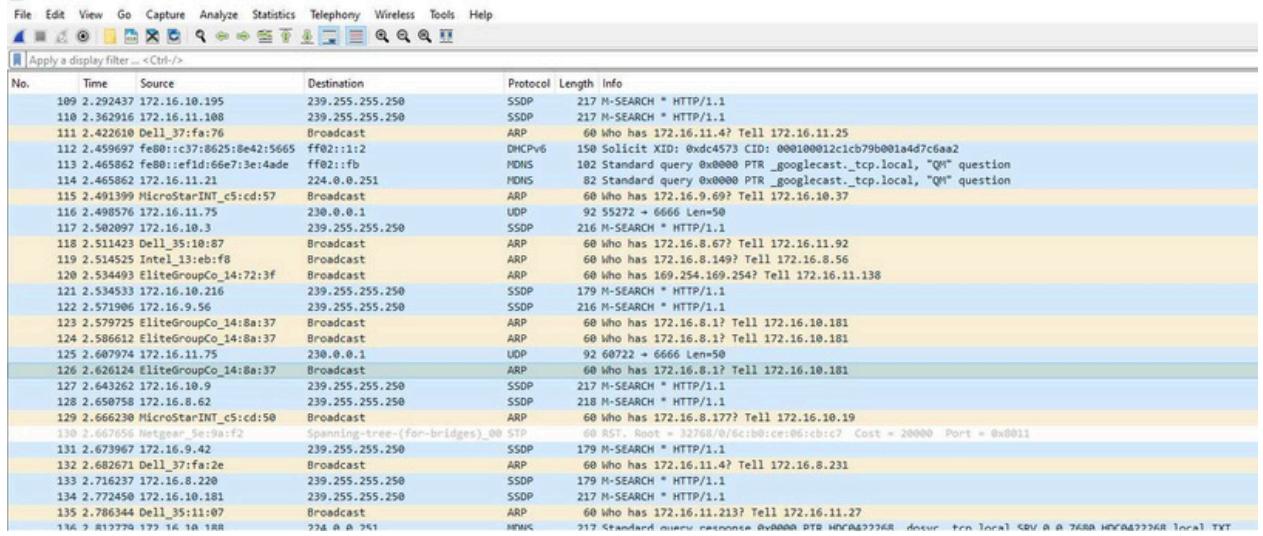
- **Capture Live Network Traffic:** Monitor data packets in real-time from various network interfaces.
- **Analyze Protocols:** Decode and inspect packets from numerous network protocols and services.
- **Filter and Search:** Use advanced filtering options to focus on specific types of traffic or data.
- **Visualize Data:** Generate statistics and visualizations to understand network behavior better.

## **I. CAPTURE LIVE NETWORK TRAFFIC:**

After downloading the Wireshark application, By launching it and selecting the network that you want to track the packets, you can stop the tracking by pressing the stop icon in the top left corner or by pressing the ctrl + E button. In the homepage you can view the following,

## PACKET LISTS:

It shows the list of packets that are captured from the interval of start and end of the tracking. It shows the serial number, time, its source, destination , protocol, length and its information.



The screenshot shows a packet capture interface with the following columns: No., Time, Source, Destination, Protocol, Length, and Info. The table lists numerous network packets, primarily ARP and SSDP messages, indicating a local network environment. The 'Info' column provides detailed protocol-specific information for each packet.

No.	Time	Source	Destination	Protocol	Length	Info
109	2.292437	172.16.10.195	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
110	2.362916	172.16.11.108	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
111	2.422618	Dell_37:fa:76	Broadcast	ARP	60	Who has 172.16.11.4? Tell 172.16.11.25
112	2.459697	fe80::c37:8625:8e42:5665	ff02::1:2	DHCPv6	158	Solicit XID: 0xd4573 CID: 000100012c1cb79b001a4d7c6aa2
113	2.465862	fe80::ef1d:66e7:3e:4ade	ff02::fb	NDNS	182	Standard query 0x0000 PTR _googlecast._tcp.local, "QNAME" question
114	2.465862	172.16.11.21	224.0.0.251	NDNS	82	Standard query 0x0000 PTR _googlecast._tcp.local, "QNAME" question
115	2.491399	MicroStar-INT_c5:cd:57	Broadcast	ARP	60	Who has 172.16.9.69? Tell 172.16.10.37
116	2.499576	172.16.11.75	230.0.0.1	UDP	92	55272 - 6666 Len=58
117	2.502097	172.16.10.3	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
118	2.511423	Dell_35:10:87	Broadcast	ARP	60	Who has 172.16.8.67? Tell 172.16.11.92
119	2.514525	Intel_13:eb:f8	Broadcast	ARP	60	Who has 172.16.8.149? Tell 172.16.8.56
120	2.534493	EliteGroupCo_14:72:3f	Broadcast	ARP	60	Who has 169.254.169.254? Tell 172.16.11.138
121	2.534533	172.16.10.216	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
122	2.571906	172.16.9.56	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
123	2.579725	EliteGroupCo_14:8a:37	Broadcast	ARP	60	Who has 172.16.8.1? Tell 172.16.10.181
124	2.586612	EliteGroupCo_14:8a:37	Broadcast	ARP	60	Who has 172.16.8.1? Tell 172.16.10.181
125	2.607974	172.16.11.75	230.0.0.1	UDP	92	60722 - 6666 Len=58
126	2.626124	EliteGroupCo_14:8a:37	Broadcast	ARP	60	Who has 172.16.8.1? Tell 172.16.10.181
127	2.643262	172.16.10.9	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
128	2.650758	172.16.8.62	239.255.255.250	SSDP	218	M-SEARCH * HTTP/1.1
129	2.666230	MicroStar-INT_c5:cd:50	Broadcast	ARP	60	Who has 172.16.8.17? Tell 172.16.10.19
130	2.667656	Negear_Sc9a:f2	Spanning-tree-(for-bridges)_00	STP	60	RST, Root = 32768/0/6c:b0:c0:06:ebc7 Cost = 20000 Port = 0x0011
131	2.673967	172.16.9.42	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
132	2.682671	Dell_37:fa:2e	Broadcast	ARP	60	Who has 172.16.11.4? Tell 172.16.8.231
133	2.716237	172.16.8.220	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
134	2.772450	172.16.10.181	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
135	2.786344	Dell_35:11:07	Broadcast	ARP	60	Who has 172.16.11.213? Tell 172.16.11.27
136	2.817779	172.16.10.188	238.0.0.91	HTTP	217	Standard query response Av80000 PTR NPFAd777ER docur.tcn.local.SRV A A 7680 1000A777ER local.TXT

## PACKET DETAILS:

It shows the information of the packet it contains arrival time, section number, frame number, length, protocol etc.,

```

< 
  ▾ Frame 126: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{C6BA0698-F8CD-4D60-B6F6-F23F1C1D910F}
    Section number: 1
    > Interface id: 0 (\Device\NPF_{C6BA0698-F8CD-4D60-B6F6-F23F1C1D910F})
      Encapsulation type: Ethernet (1)
      Arrival Time: Aug 30, 2024 08:40:48.810620000 India Standard Time
      UTC Arrival Time: Aug 30, 2024 03:10:48.810620000 UTC
      Epoch Arrival Time: 1724987448.810620000
      [Time shift for this packet: 0.000000000 seconds]
      [Time delta from previous captured frame: 0.018150000 seconds]
      [Time delta from previous displayed frame: 0.018150000 seconds]
      [Time since reference or first frame: 2.626124000 seconds]
      Frame Number: 126
      Frame Length: 60 bytes (480 bits)
      Capture Length: 60 bytes (480 bits)
      [Frame is marked: False]
      [Frame is ignored: False]
      [Protocols in frame: eth:ethertype:arp]
      [Coloring Rule Name: ARP]
      [Coloring Rule String: arp]
    > Ethernet II, Src: EliteGroupCo_14:8a:37 (88:ae:dd:14:8a:37), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
    > Address Resolution Protocol (request)
  <

```

## PACKET BYTES:

It shows the byte representation of the packet in hexadecimal format which can also be changed to binary, decimal or octal.

0000	01 00 5e 7f ff fa 50 9a 4c 37 fb 08 08 00 45 00	..^..P L7...E
0010	00 a5 a0 30 00 00 04 11 70 e3 ac 10 09 2a ef ff	..0....p....*
0020	ff fa d0 9d 07 6c 00 91 8f 6f 4d 2d 53 45 41 52	....I..oM-SEAR
0030	43 48 20 2a 20 48 54 54 50 2f 31 2e 31 0d 0a 48	CH * HTT P/1.1..H
0040	6f 73 74 3a 20 32 33 39 2e 32 35 35 2e 32 35 35	ost: 239 .255.255
0050	2e 32 35 30 3a 31 39 30 30 0d 0a 53 54 3a 20 75	.250:190 0..ST: u
0060	72 6e 3a 73 63 68 65 6d 61 73 2d 75 70 6e 70 2d	rn:schem as-upnp-
0070	6f 72 67 3a 64 65 76 69 63 65 3a 49 6e 74 65 72	org:devi ce:Inter
0080	6e 65 74 47 61 74 65 77 61 79 44 65 76 69 63 65	netGatew ayDevice
0090	3a 31 0d 0a 4d 61 6e 3a 20 22 73 73 64 70 3a 64	:1..Man: "ssdp:d
00a0	69 73 63 6f 76 65 72 22 0d 0a 4d 58 3a 20 33 0d	iscover" ..MX: 3-
00b0	0a 0d 0a	***

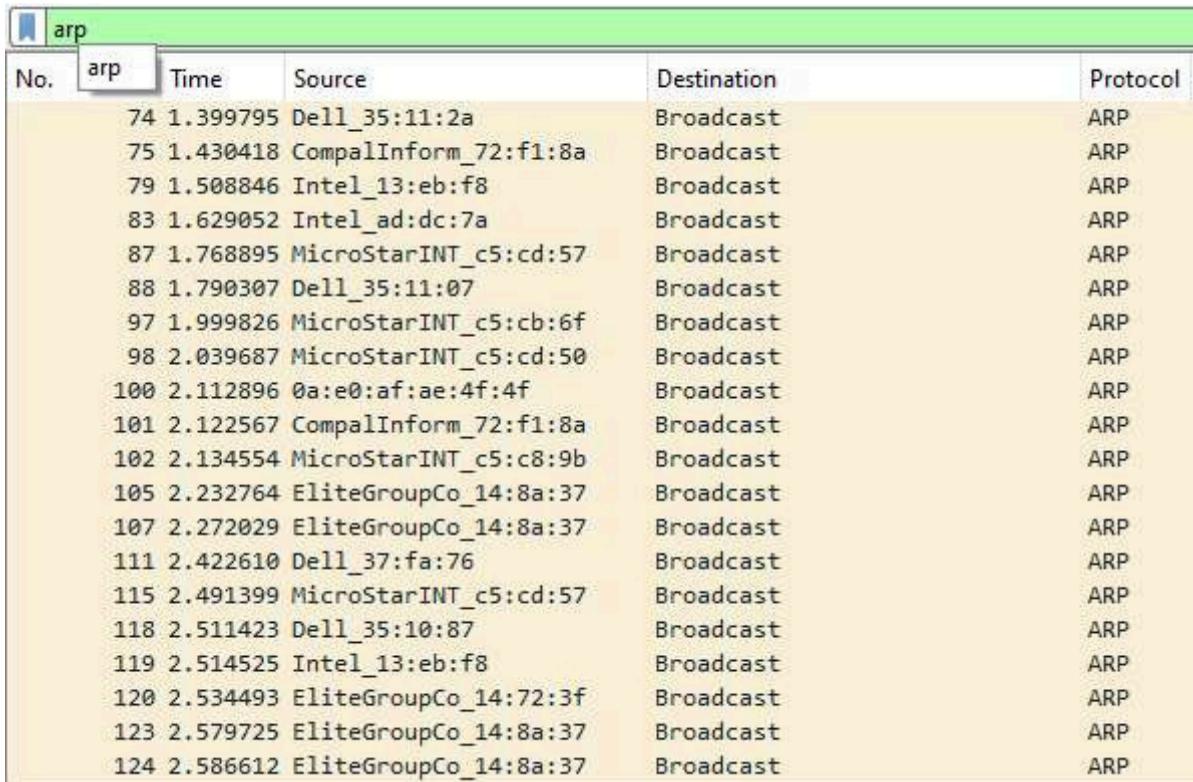
## II. ANALYZE PROTOCOLS

We can inspect the packets by double clicking on the packet from the packet list pane and there is a colorization mechanism is present which show different colors to different protocols which can be customizable over the networks through right clicking the packet and chose colourization to view the current colourization we can view it by customize colourization option.

Name	Filter
<input checked="" type="checkbox"/> New coloring rule	eth.addr eq d8:bb:c1:c5:cb:6f and eth.addr eq 01:00:5e:7f:ffff:fa
<input checked="" type="checkbox"/> Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update && !tcp.analysis.keep_alive && !tcp.analysis.keep_alive_ack
<input checked="" type="checkbox"/> HSRP State Change	hsrp.state != 8 && hsrp.state != 16
<input checked="" type="checkbox"/> Spanning Tree Topology Change	stp.type == 0x80
<input checked="" type="checkbox"/> OSPF State Change	ospf.msg != 1
<input checked="" type="checkbox"/> ICMP errors	icmp.type.in { 3..5, 11 }    icmpv6.type.in { 1..4 }
<input checked="" type="checkbox"/> ARP	arp
<input checked="" type="checkbox"/> ICMP	icmp    icmpv6
<input checked="" type="checkbox"/> TCP RST	tcp.flags.reset eq 1
<input checked="" type="checkbox"/> SCTP ABORT	sctp.chunk_type eq ABORT
<input checked="" type="checkbox"/> IPv4 TTL low or unexpected	(ip.dst != 224.0.0.0/4 && ip.ttl < 5 && !(pm    ospf    eigrp    bgp    tcp.port==179))    (ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.ttl < 5)
<input checked="" type="checkbox"/> IPv6 hop limit low or unexpected	(ipv6.dst != ff00::/8 && ipv6.hlim < 5 && !(ospf   bgp    tcp.port==179))    (ipv6.dst == ff00::/8 && ipv6.hlim not in {1, 64, 255})
<input checked="" type="checkbox"/> Checksum Errors	eth.fcs.status=="Bad"    ip.checksum.status=="Bad"    tcp.checksum.status=="Bad"    udp.checksum.status=="Bad"    sctp.checksum.status=="Bad"
<input checked="" type="checkbox"/> SMB	smb    nbss    nbns    netbios
<input checked="" type="checkbox"/> HTTP	http    tcp.port == 80    http2
<input checked="" type="checkbox"/> DCERPC	dcerpc
<input checked="" type="checkbox"/> Routing	hsrp    eigrp    ospf    bgp    cdp    vrrp    carp    gvrp    igmp    ismp
<input checked="" type="checkbox"/> TCP SYN/FIN	tcp.flags & 0x02    tcp.flags.fin == 1
<input checked="" type="checkbox"/> TCP	tcp
<input checked="" type="checkbox"/> UDP	udp

### III. FILTER AND SEARCH

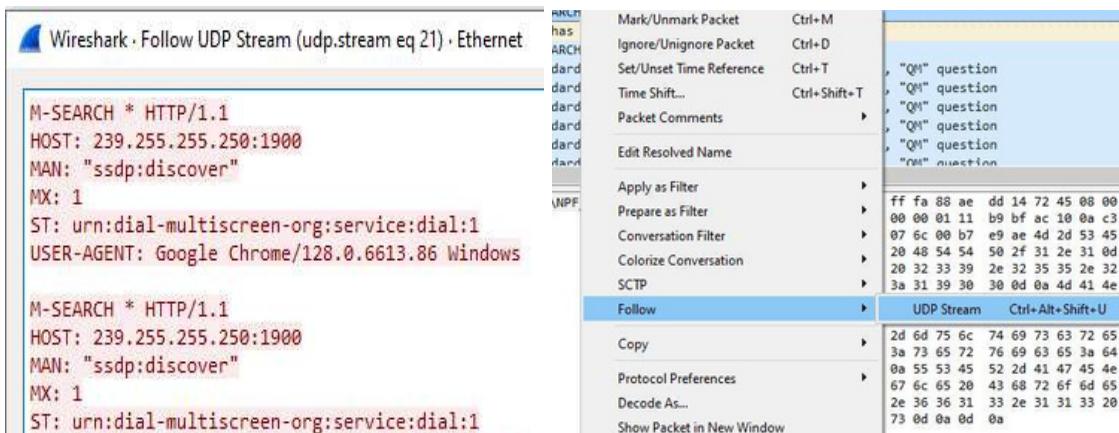
If it needs to search for a specific protocol or any packet we can search from the search column present above the packet list pane and we can use already defined filters also that are shown when typing through the search box by using it.



The screenshot shows the Wireshark interface with a search bar at the top containing the text "arp". Below the search bar is a table with columns: No., Time, Source, Destination, and Protocol. All 24 rows in the table are highlighted in light orange, indicating they match the search criteria. The "Protocol" column for all entries shows "ARP".

No.	Time	Source	Destination	Protocol
74	1.399795	Dell_35:11:2a	Broadcast	ARP
75	1.430418	CompaInform_72:f1:8a	Broadcast	ARP
79	1.508846	Intel_13:eb:f8	Broadcast	ARP
83	1.629052	Intel_ad:dc:7a	Broadcast	ARP
87	1.768895	MicroStarINT_c5:cd:57	Broadcast	ARP
88	1.790307	Dell_35:11:07	Broadcast	ARP
97	1.999826	MicroStarINT_c5:cb:6f	Broadcast	ARP
98	2.039687	MicroStarINT_c5:cd:50	Broadcast	ARP
100	2.112896	0a:e0:af:ae:4f:4f	Broadcast	ARP
101	2.122567	CompaInform_72:f1:8a	Broadcast	ARP
102	2.134554	MicroStarINT_c5:c8:9b	Broadcast	ARP
105	2.232764	EliteGroupCo_14:8a:37	Broadcast	ARP
107	2.272029	EliteGroupCo_14:8a:37	Broadcast	ARP
111	2.422610	Dell_37:fa:76	Broadcast	ARP
115	2.491399	MicroStarINT_c5:cd:57	Broadcast	ARP
118	2.511423	Dell_35:10:87	Broadcast	ARP
119	2.514525	Intel_13:eb:f8	Broadcast	ARP
120	2.534493	EliteGroupCo_14:72:3f	Broadcast	ARP
123	2.579725	EliteGroupCo_14:8a:37	Broadcast	ARP
124	2.586612	EliteGroupCo_14:8a:37	Broadcast	ARP

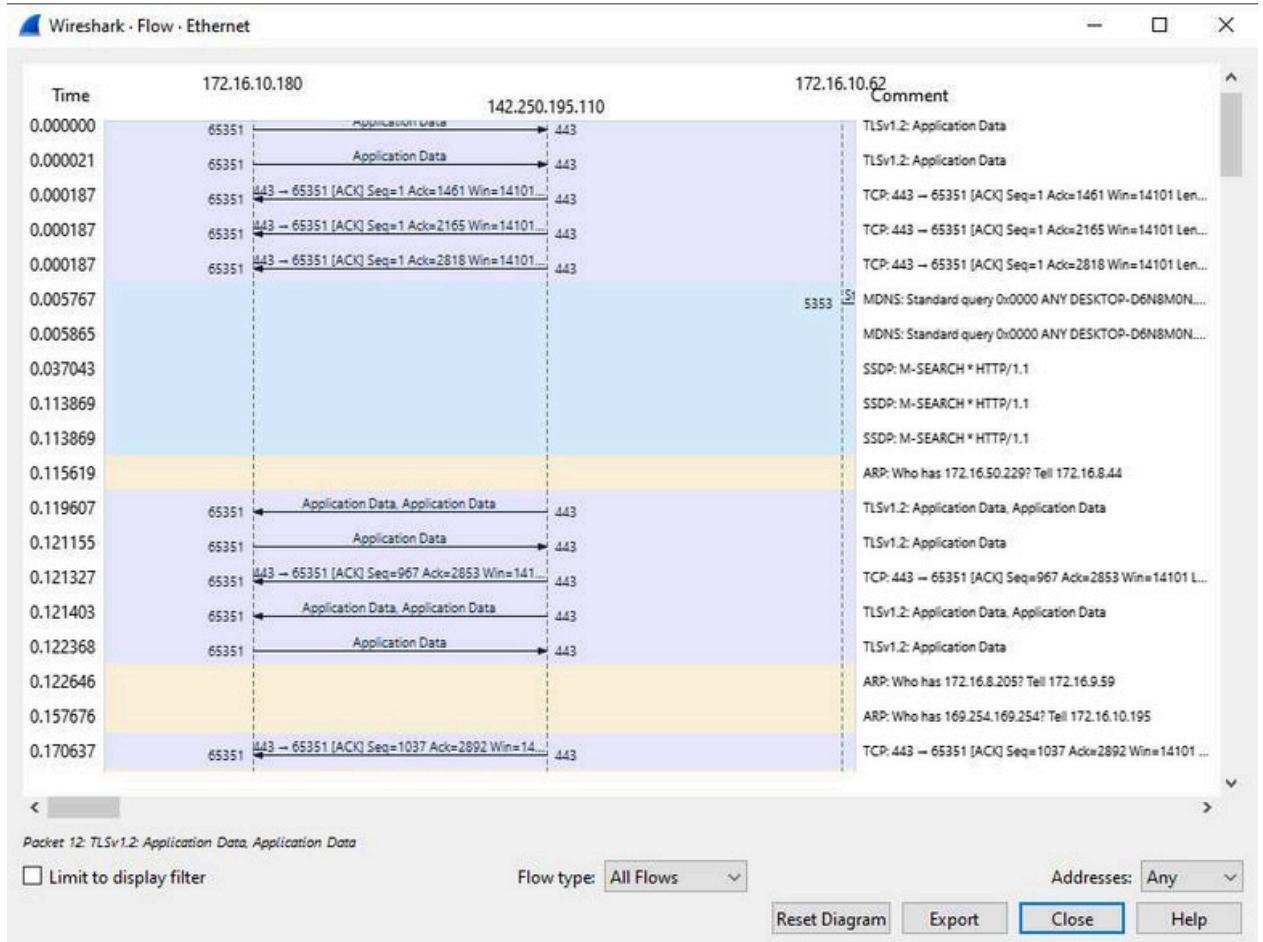
We can also find a packet and track the flow of the packet by right clicking the packet from the list and clicking on follow -> TCP/UDP stream to view the conversation.



The screenshot shows the Wireshark interface with a selected UDP packet. A context menu is open, and the "Follow" submenu is expanded. The submenu includes options like "Mark/Unmark Packet", "Ignore/Unignore Packet", "Set/Unset Time Reference", "Time Shift...", "Packet Comments", "Edit Resolved Name", "Apply as Filter", "Prepare as Filter", "Conversation Filter", "Colorize Conversation", "SCTP", and "Follow". The "Follow" option is highlighted. To the right of the menu, the "UDP Stream" is displayed, showing the raw hex and ASCII data of the selected packet.

## IV. VISUALIZE THE DATA

We can visualize the data through flow graphs and other methods by clicking on the statistics from the top bar.



## CAPTURING AND ANALYZING PACKETS USING WIRESHARK

### 1. Create a Filter to display only TCP/UDP packets, inspect the packets and provide the flow graph

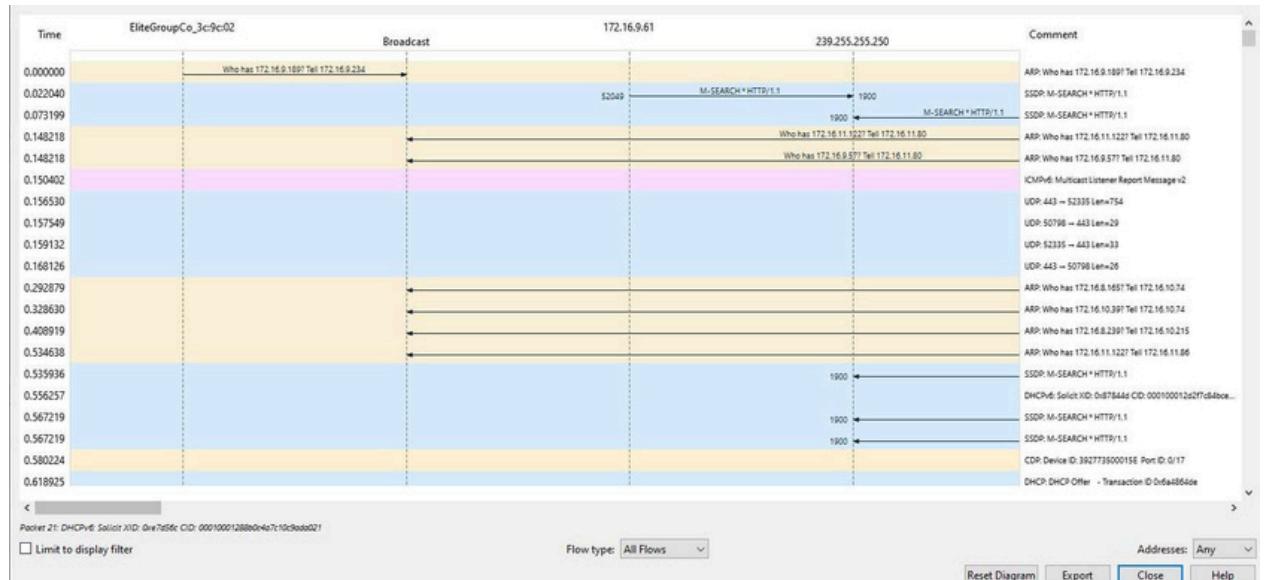
#### PROCEDURE:

- Select Start capture in a network
- Stop the capture and save the packets
- Search for TCP in the search bar to view TCP packets
- Search for UDP to view UDP packets
- To view flow graph click statistics and select flow graph

#### PACKETS:

No.	Source	Destination	Protocol	Length	Info
1	udp.stream eq 21	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
2	udp.stream eq 19	2.16.9.61	SSDP	217	M-SEARCH * HTTP/1.1
3	udp.stream eq 13	2.16.10.60	SSDP	217	M-SEARCH * HTTP/1.1
4	udp.stream eq 57	2.250.195.42	UDP	796	443 → 52335 Len=754
5	udp.stream eq 97	2.16.10.180	UDP	71	50798 → 443 Len=29
6	udp.stream eq 27	172.16.10.180	UDP	75	52335 → 443 Len=33
7	udp	142.250.195.42	UDP	68	443 → 50798 Len=26
8	udpcp	2.16.10.180	SSDP	217	M-SEARCH * HTTP/1.1
9	udpcap	12.250.195.42	DHCPv6	150	Solicit XID: 0x87844d CID: 000100012d2f7c84b
10	udplite	2.16.9.42	SSDP	212	M-SEARCH * HTTP/1.1
11	18.0.567219	172.16.9.42	SSDP	217	M-SEARCH * HTTP/1.1
12	20.0.618925	172.16.8.1	DHCP	342	DHCP Offer - Transaction ID 0x6a4864de
13	21.0.628448	fe80::10a1:3b91:30c9:d636	DHCPv6	147	Solicit XID: 0xe7d56c CID: 00010001288b0c4a7
14	22.0.630572	fe80::8025:cf40:cea:ce0e	DHCPv6	157	Solicit XID: 0x41c88e CID: 000100012a991df10

#### FLOW GRAPH:



## 2. Create a Filter to display only ARP packets and inspect the packets.

### PROCEDURE:

- Select Start capture in a network
- Stop the capture
- Search for arp in the search bar
- Save the packets

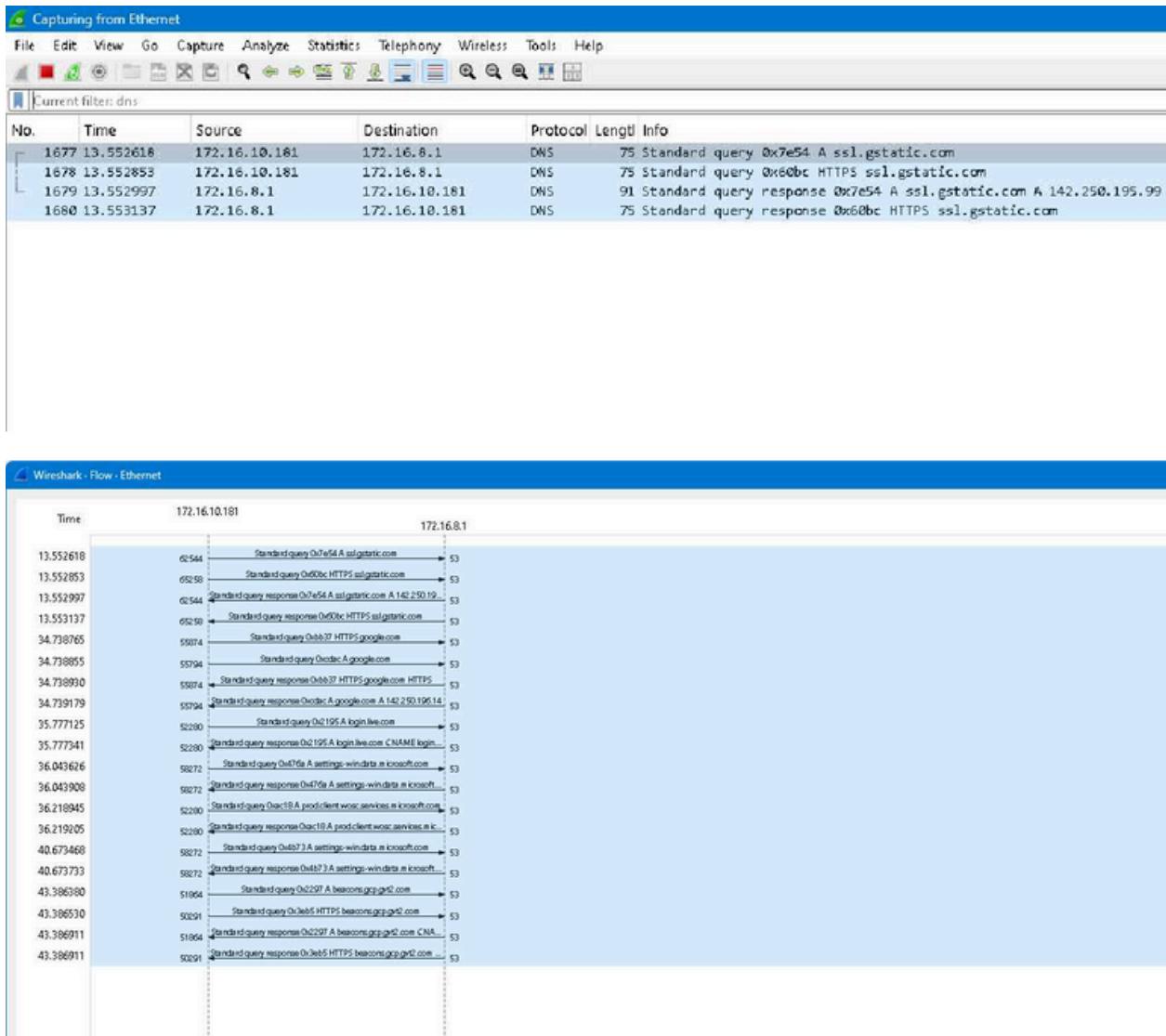
### PACKETS:

No.	arp	Time	Source	Destination	Protocol	Length	Info
40	1.052040	MicroStarINT_c5:cd:ef	Broadcast		ARP	60	Who has 172.16.11.122? Tell 172.16.10.3
42	1.144863	EliteGroupCo_14:72:3f	Broadcast		ARP	60	Who has 172.16.9.57? Tell 172.16.11.80
54	1.206147	MicroStarINT_c5:c8:9b	Broadcast		ARP	60	Who has 172.16.8.165? Tell 172.16.10.31
74	1.286187	MicroStarINT_c5:cc:1e	Broadcast		ARP	60	Who has 172.16.10.39? Tell 172.16.10.74
75	1.326882	Pegatron_e0:85:fe	Broadcast		ARP	60	Who has 172.16.11.8? Tell 172.16.9.153
78	1.440383	MicroStarINT_c5:cb:cf	Broadcast		ARP	60	Who has 172.16.10.115? Tell 172.16.10.52
79	1.532163	EliteGroupCo_15:e7:f4	Broadcast		ARP	60	Who has 172.16.8.187? Tell 172.16.10.194
83	1.639393	AzureWaveTec_2b:10:53	Broadcast		ARP	60	Who has 169.254.51.124? (ARP Probe)
86	1.676708	EliteGroupCo_3c:9c:02	Broadcast		ARP	60	Who has 172.16.9.189? Tell 172.16.9.234
87	1.691502	MicroStarINT_c5:cd:ef	Broadcast		ARP	60	Who has 172.16.11.122? Tell 172.16.10.3
91	1.796887	MicroStarINT_c5:cd:57	Broadcast		ARP	60	Who has 172.16.8.172? Tell 172.16.10.37
96	1.894453	MicroStarINT_c5:c8:9b	Broadcast		ARP	60	Who has 172.16.8.165? Tell 172.16.10.31
97	1.895007	EliteGroupCo_14:72:45	Broadcast		ARP	60	Who has 169.254.169.254? Tell 172.16.10.195
99	1.958019	HikvisionDig_aa:a0:4f	Broadcast		ARP	60	Who has 172.16.9.250? Tell 172.16.11.254

## 3. Create a Filter to display only DNS packets and provide the flow graph.

### PROCEDURE:

- Select Start capture in a network
- Stop the capture
- Search for DNS in the search bar
- Select the flowgraph from the statistics
- Save the packets



#### 4. Create a Filter to display only HTTP packets and inspect the packets

##### PROCEDURE:

- Select Start capture in a network
- Stop the capture
- Search for HTTP in the search bar
- Save the packets

## 5. Create a Filter to display only IP/ICMP packets and inspect the packets.

### PROCEDURE:

- Select Start capture in a network
- Stop the capture
- Search for IP in the search bar
- Double click on the packet to inspect the packet.
- Save the packets

No.	Time	Source	Destination	Protocol	Length	Info
2	0.024092	172.16.11.81	172.16.11.255	NBNS	110	Registration NB MCQUEEN<00>
3	0.024092	172.16.11.81	172.16.11.255	NBNS	110	Registration NB WORKGROUP<00>
8	0.025545	172.16.11.81	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.251 for any sources
9	0.025545	172.16.11.81	224.0.0.22	IGMPv3	60	Membership Report / Join group 239.255.255.250 for any sources
10	0.025548	172.16.11.81	224.0.0.252	LLNMR	67	Standard query 0x09bc ANY McQueen
12	0.028413	20.87.44.201	172.16.10.181	TLSv1.2	157	Application Data
13	0.028413	20.87.44.201	172.16.10.181	TLSv1.2	116	Application Data
14	0.028667	172.16.10.181	20.87.44.201	TCP	54	55406 + 443 [ACK] Seq=1 Ack=166 Win=1023 Len=0
15	0.030215	172.16.10.181	20.87.44.201	TLSv1.2	85	Application Data
16	0.030352	20.87.44.201	172.16.10.181	TCP	60	443 + 55406 [ACK] Seq=166 Ack=32 Win=254 Len=0
21	0.038350	172.16.9.48	172.16.11.255	UDP	82	56309 + 1947 Len=40
22	0.038350	172.16.9.48	255.255.255.255	UDP	82	56310 + 1947 Len=40
23	0.045170	172.16.9.229	224.0.0.251	MDNS	81	Standard query 0x0000 ANY LAPTOP-6IFUU8BO.local, "QN" question
25	0.045170	172.16.9.229	224.0.0.251	MDNS	119	Standard query response 0x0000 AAAA fe80::8c8c:22da:cb6:4808 A 172.16.9.229
31	0.045220	172.16.9.229	224.0.0.252	LLNMR	75	Standard query 0xc816 ANY LAPTOP-6IFUU8BO
34	0.045225	172.16.9.229	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources
35	0.045225	172.16.9.229	224.0.0.22	IGMPv3	60	Membership Report / Leave group 224.0.0.252
36	0.045225	172.16.9.229	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources
37	0.045225	172.16.9.229	224.0.0.22	IGMPv3	60	Membership Report / Leave group 224.0.0.252
38	0.045225	172.16.9.229	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources

## 6. Create a Filter to display only DHCP packets and inspect the packets.

### PROCEDURE:

- Select Start capture in a network
- Stop the capture
- Search for DHCP in the search bar
- Double click on the packet to inspect the packet.
- Save the packets

\*Ethernet

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dhcp

No.	Time	Source	Destination	Protocol	Length	Info
73	0.157428	0.0.0.0	255.255.255.255	DHCP	344	DHCP Discover - Transaction ID 0x38e9e697
99	0.219819	0.0.0.0	255.255.255.255	DHCP	354	DHCP Request - Transaction ID 0x38e9e697
660	2.590471	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x65af1c1
775	3.591020	172.16.8.1	255.255.255.255	DHCP	342	DHCP Offer - Transaction ID 0x65af1c1
778	3.603182	0.0.0.0	255.255.255.255	DHCP	370	DHCP Request - Transaction ID 0x65af1c1
779	3.603182	172.16.8.1	255.255.255.255	DHCP	342	DHCP ACK - Transaction ID 0x65af1c1
916	4.945590	0.0.0.0	255.255.255.255	DHCP	354	DHCP Request - Transaction ID 0xb78660c1
917	4.945597	172.16.8.1	255.255.255.255	DHCP	342	DHCP ACK - Transaction ID 0xb78660c1
1643	11.518313	0.0.0.0	255.255.255.255	DHCP	342	DHCP Inform - Transaction ID 0x3f85c413
1737	12.612307	0.0.0.0	255.255.255.255	DHCP	364	DHCP Request - Transaction ID 0x77953508
2131	12.925909	0.0.0.0	255.255.255.255	DHCP	364	DHCP Request - Transaction ID 0x77953508
2196	13.018650	0.0.0.0	255.255.255.255	DHCP	364	DHCP Request - Transaction ID 0x77953508
2319	13.510848	0.0.0.0	255.255.255.255	DHCP	342	DHCP Inform - Transaction ID 0x3f85c413
2453	13.938490	0.0.0.0	255.255.255.255	DHCP	345	DHCP Discover - Transaction ID 0x1bca9941
2679	14.939832	172.16.8.1	255.255.255.255	DHCP	342	DHCP Offer - Transaction ID 0x1bca9941
2752	15.392426	0.0.0.0	255.255.255.255	DHCP	342	DHCP Inform - Transaction ID 0x3f85c413
3122	17.233582	0.0.0.0	255.255.255.255	DHCP	366	DHCP Request - Transaction ID 0x6d72766d
3123	17.233584	172.16.8.1	255.255.255.255	DHCP	342	DHCP ACK - Transaction ID 0x6d72766d
3141	17.389540	172.16.8.1	255.255.255.255	DHCP	342	DHCP Offer - Transaction ID 0x1bca9941
3142	17.389539	0.0.0.0	255.255.255.255	DHCP	345	DHCP Discover - Transaction ID 0x1bca9941

Wireshark - Packet 2679 - Ethernet

Frame 2679: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface \Device\NPF\_{76F0E25E-9435-4A81-9A01-46D4A07D73F9}, id 0

Section number: 1

> Interface id: 0 (\Device\NPF\_{76F0E25E-9435-4A81-9A01-46D4A07D73F9})

Encapsulation type: Ethernet (1)

Arrival Time: Oct 4, 2024 08:29:05.738648000 India Standard Time

UTC Arrival Time: Oct 4, 2024 02:59:05.738648000 UTC

Epoch Arrival Time: 1728010745.738648000

[Time shift for this packet: 0.000000000 seconds]

[Time delta from previous captured frame: 0.005162000 seconds]

[Time delta from previous displayed frame: 1.001342000 seconds]

[Time since reference or first frame: 14.939832000 seconds]

Frame Number: 2679

Frame Length: 342 bytes (2736 bits)

Capture Length: 342 bytes (2736 bits)

[Frame is marked: False]

[Frame is ignored: False]

[Protocols in frame: eth:ether:type:ip:udp:dhcp]

[Coloring Rule Name: UDP]

[Coloring Rule String: udp]

0000	ff ff ff ff ff ff 7c 5a 1c cf be 45 08 00 45 10	..... Z ..E..-E-
0001	01 48 00 00 00 80 11 85 84 ac 10 08 01 ff ff	H.....
0002	ff ff 00 43 00 44 01 34 60 f1 02 01 06 00 1b ca	..C-D-4 .....
0030	99 41 00 00 00 00 00 00 00 ac 10 09 3d 00 00	A.....
0040	00 00 00 00 00 00 d4 6a 6a 82 6d 35 00 00 00 00	.....j j-mS..
0050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....j j-mS..
0060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00a0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00b0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00c0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00d0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00e0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00f0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0110	00 00 00 00 00 00 63 82 53 63 35 01 02 36 04 ac	.....c ..Sc5 ..6..
0120	10 08 01 33 04 00 00 0e 10 01 04 ff fc 00 03	..3.....
0130	04 ac 10 08 01 06 04 ac 10 08 01 ff 00 00 00 00	.....
0140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

## Result:

Thus the uses of the Wireshark Tool has been studied and understood.

# **CS19541-COMPUTER NETWORKS-LAB MANUAL**

## **Practical -3**

### **AIM: To study the Packet tracer tool Installation and User Interface Overview**

- c) **To understand environment of CISCO PACKET TRACER to design simple network.**

#### **INTRODUCTION:**

A simulator, as the name suggests, simulates network devices and its environment. Packet Tracer is an exciting network design, simulation and modelling tool.

1. It allows you to model complex systems without the need for dedicated equipment.
2. It helps you to practice your network configuration and troubleshooting skills via computer or an Android or iOS based mobile device.
3. It is available for both the Linux and Windows desktop environments.
4. Protocols in Packet Tracer are coded to work and behave in the same way as they would on real hardware.

#### **INSTALLING PACKET TRACER:**

To download Packet Tracer, go to <https://www.netacad.com> and log in with your Cisco Networking Academy credentials; then, click on the Packet Tracer graphic and download the package appropriate for your operating system. (Can be used to download in your laptop).

##### Windows

Installation in Windows is pretty simple and straightforward; the setup comes in a single file named Packettracer\_Setup6.0.1.exe. Open this file to begin the setup wizard, accept the license agreement, choose a location, and start the installation.

##### Linux

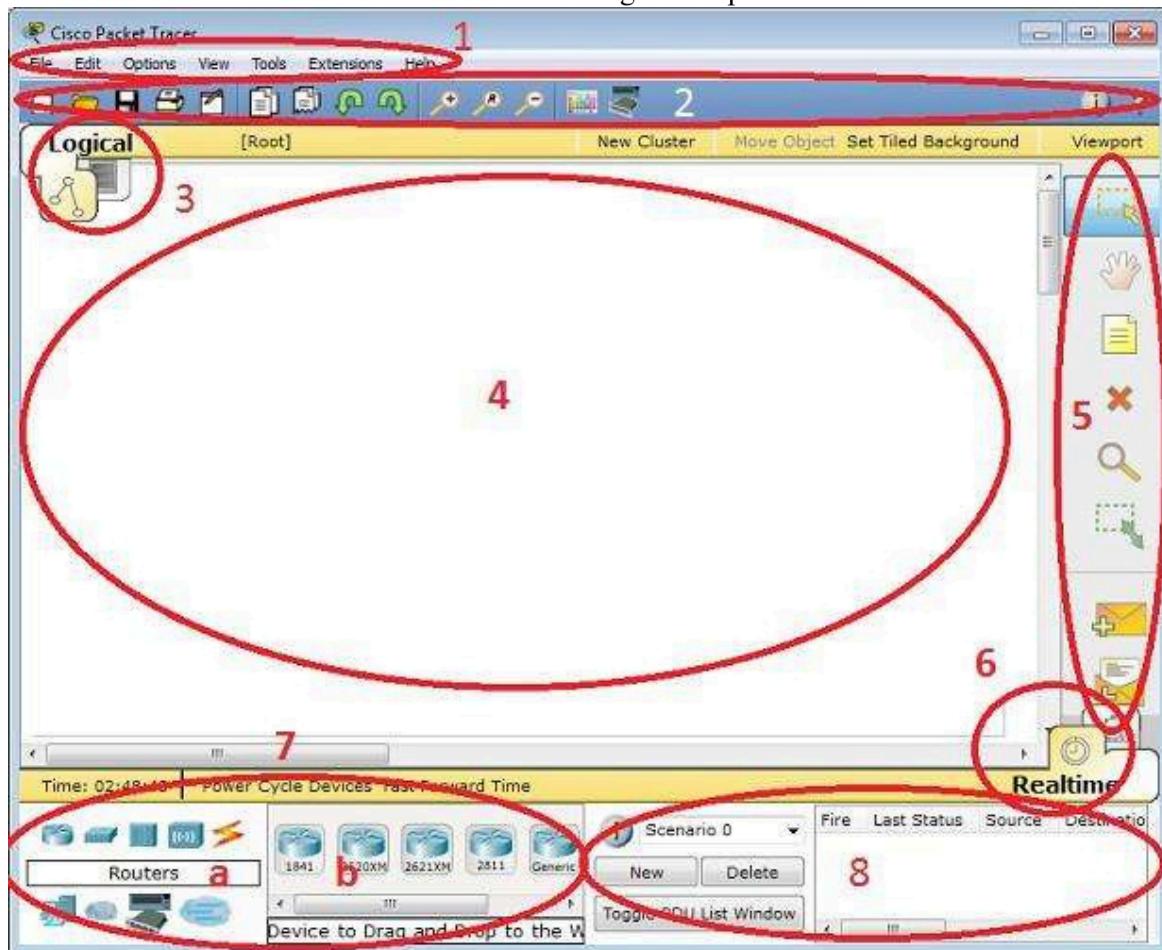
Linux users with an Ubuntu/Debian distribution should download the file for Ubuntu, and those using Fedora/Redhat/CentOS must download the file for Fedora. Grant executable permission to this file by using chmod, and execute it to begin the installation.

```
chmod +x PacketTracer601_i386_installer-rpm.bin  
./PacketTracer601_i386_installer-rpm.bin
```

# CS19541-COMPUTER NETWORKS-LAB MANUAL

## USER INTERFACE OVERVIEW:

The layout of Packet Tracer is divided into several components. The components of the Packet Tracer interface are as follows: match the numbering with explanations.



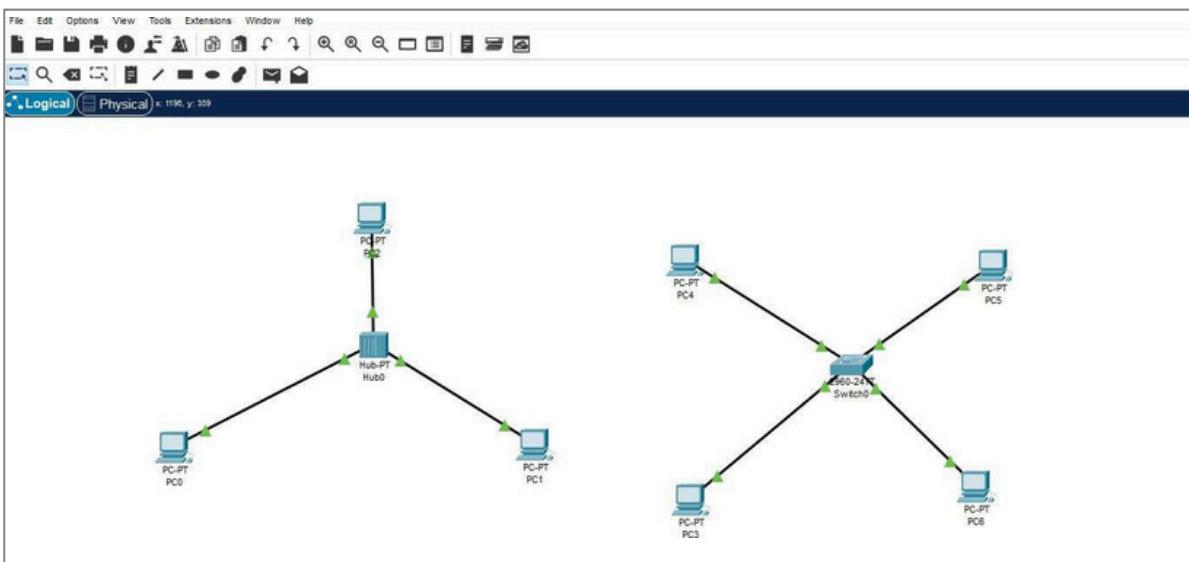
1. Menu bar – This is a common menu found in all software applications; it is used to open, save, print, change preferences, and so on.
2. Main toolbar – This bar provides shortcut icons to menu options that are commonly accessed, such as open, save, zoom, undo, and redo, and on the right-hand side is an icon for entering network information for the current network.
3. Logical/Physical workspace tabs – These tabs allow you to toggle between the Logical and Physical work areas.
4. Workspace – This is the area where topologies are created and simulations are displayed.
5. Common tools bar – This toolbar provides controls for manipulating topologies, such as select, move layout, place note, delete, inspect, resize shape, and add simple/complex PDU.
6. Real-time/Simulation tabs – These tabs are used to toggle between the real and simulation modes. Buttons are also provided to control the time, and to capture the packets.
7. Network component box – This component contains all of the network and end devices available with Packet Tracer, and is further divided into two areas: Area 7a: Device-type selection box – This area contains device categories Area 7b: Device-specific selection box – When a device category is selected, this selection box displays the different device models within that category

## **CS19541-COMPUTER NETWORKS-LAB MANUAL**

8. User-created packet box – Users can create highly-customized packets to test their topology from this area, and the results are displayed as a list.

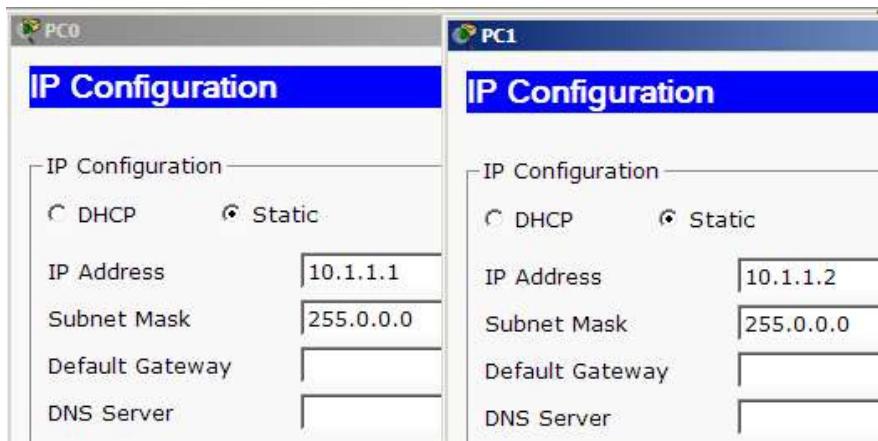
### **d) Analyse the behaviour of network devices using CISCO PACKET TRACER simulator.**

1. From the network component box, click and drag-and-drop the below components:
  - a. 4 Generic PCs and One HUB
  - b. 4 Generic PCs and One switch
2. Click on Connections:
  - a. Click on Copper Straight-Through cable,
  - b. Select one of the PC and connect it to HUB using the cable. The link LED should glow in green, indicating that the link is up. Similarly connect remaining 3 PCs to the HUB.
  - c. Similarly connect 4 PCs to the switch using copper straight-through cable.



## CS19541-COMPUTER NETWORKS-LAB MANUAL

3. Click on the PCs connected to hub, go to the Desktop tab, click on IP Configuration, and enter an IP address and subnet mask. Here, the default gateway and DNS server information is not needed as there are only two end devices in the network.



Click on the PDU (message icon) from the common tool bar,

- a. Drag and drop it on one of PC (source machine) and then drop it on another PC (destination machine) connected to the HUB.
4. Observe the flow of PDU from source PC to destination PC by selecting the Realtime mode of simulation.
5. Repeat step #3 to step #5 for the PCs connected to the switch.
6. Observe how HUB and switch are forwarding the PDU and write your observation and conclusion about the behaviors of Switch and HUB.

---

### **Student observation:**

- From your observation write down the behavior of Switch and HUB in terms of forwarding the packets received by them.**
- Find out the network topology implemented in your college and draw and label that topology in your observation book.**

# CS19541-COMPUTER NETWORKS-LAB MANUAL

## Practical-6

**AIM:** Write a program to implement error detection and correction using HAMMING code concept. Make a test run to input data stream and verify error correction feature.

### **Error Correction at Data Link Layer:**

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is transmitted from the sender to the receiver. It is a technique developed by R.W. Hamming for error correction.

### **Create sender program with below features.**

1. Input to sender file should be a text of any length. Program should convert the text to binary.
2. Apply hamming code concept on the binary data and add redundant bits to it.
3. Save this output in a file called channel.

### **Create a receiver program with below features**

1. Receiver program should read the input from Channel file.
2. Apply hamming code on the binary data to check for errors.
3. If there is an error, display the position of the error.
4. Else remove the redundant bits and convert the binary data to ascii and display the output.

### **Student observation:-**

Write the code here:

```
import numpy as np

# Function to convert text to binary
def text_to_binary(text):
    return ''.join(format(ord(char), '08b') for char in text)

# Function to convert binary to text
def binary_to_text(binary_data):
    chars = [chr(int(binary_data[i:i+8], 2)) for i in range(0, len(binary_data), 8)]
    return ''.join(chars)

# Function to calculate redundant bits needed for Hamming Code
def calculate_redundant_bits(m):
    r = 0
    while (2 ** r) < (m + r + 1):
        r += 1
    return r

# Function to apply Hamming Code to add redundant bits to binary data
def apply_hamming_code(data):
    m = len(data)
    r = calculate_redundant_bits(m)
```

# CS19541-COMPUTER NETWORKS-LAB MANUAL

## Practical-6

```
hamming_code = ['0'] * (m + r)
j = 0
k = 1
for i in range(1, m + r + 1):
    if i == 2 ** j:
        j += 1
    else:
        hamming_code[i - 1] = data[-k]
        k += 1

for i in range(r):
    x = 2 ** i
    one_count = 0
    for j in range(1, m + r + 1):
        if j & x and hamming_code[j - 1] == '1':
            one_count += 1
    hamming_code[x - 1] = '1' if one_count % 2 != 0 else '0'

return ".join(hamming_code[::-1])
```

### **# Function to detect and correct errors in Hamming Code**

```
def detect_and_correct_hamming_code(data):
    n = len(data)
    r = calculate_redundant_bits(n)
    error_position = 0
    for i in range(r):
        x = 2 ** i
        one_count = 0
        for j in range(1, n + 1):
            if j & x and data[-j] == '1':
                one_count += 1
        if one_count % 2 != 0:
            error_position += x

    if error_position > 0:
        print(f'Error detected at position: {error_position}')
        data = list(data)
        data[-error_position] = '1' if data[-error_position] == '0' else '0'
        data = ".join(data)
    else:
        print("No error detected.")
```

### **# Removing redundant bits**

```
corrected_data = []
j = 0
for i in range(1, n + 1):
    if i != 2 ** j:
        corrected_data.append(data[-i])
    else:
```

## CS19541-COMPUTER NETWORKS-LAB MANUAL

### Practical-6

```
j += 1

return ".join(corrected_data[::-1])

# Sender Program
def sender(text):
    binary_data = text_to_binary(text)
    encoded_data = apply_hamming_code(binary_data)
    with open("channel.txt", "w") as f:
        f.write(encoded_data)
    print("Data has been sent to channel.txt.")

# Receiver Program
def receiver():
    with open("channel.txt", "r") as f:
        encoded_data = f.read()
    print(f'Received encoded data: {encoded_data}')
    corrected_data = detect_and_correct_hamming_code(encoded_data)
    decoded_text = binary_to_text(corrected_data)
    print(f'Decoded text after error correction: {decoded_text}')

# Example Usage
# Enter text to send
text_to_send = "Hello" # Modify the text to test
sender(text_to_send)
receiver()
```

Input:-

**Text to send:** "Hello"

Output:

```
Received encoded data: 0100100001100110101101100011010100011001110100
No error detected.
Decoded text after error correction: Hello
```

# CS19541-COMPUTER NETWORKS-LAB MANUAL

## Practical-7

**AIM:** Write a program to implement flow control at data link layer using SLIDING WINDOW PROTOCOL. Simulate the flow of frames from one node to another.

Program should achieve at least below given requirements. You can make it a bidirectional program wherein receiver is sending its data frames with acknowledgement (Piggybacking).

**Create a sender program with following features:-**

1. Input Window size from the user.
2. Input a Text message from the user.
3. Consider 1 character per frame.
4. Create a frame with following fields [Frame no., DATA].
5. Send the frames. [Print the output on screen and save it in a file called Sender\_Buffer.]
6. Wait for the acknowledgement from the Receiver. [Induce delay in the program]
7. Reader a file called Receiver\_Buffer.
8. Check ACK field for the Acknowledgement number.
9. If the Acknowledgement number is as expected, send new set of frames accordingly, [overwrite the Sender\_Buffer file with new frames] Else if NACK is received, resend the frames accordingly. [Overwrite the Sender\_Buffer with old frame].

**Create a receiver file with following features**

1. Reader a file called Sender\_Buffer.
2. Check the Frame no.
3. If the Fame no. are as expected, write the appropriate ACK no. in the Receiver\_Buffer file. Else write NACK no. in the Receiver\_Buffer file.

**NOTE: Induce error and verify the behaviour of the program. Manually Change the Frame no and Ack no in the files].**

### Student observation:

Write the code here:

```
import random

# Constants for ACK and NACK
ACK = "ACK"
NACK = "NACK"

# Sliding window protocol implementation
def sliding_window_protocol(window_size, message):
    # Split the message into frames (1 character per frame)
    frames = [(i, message[i]) for i in range(len(message))]
    current_window_start = 0
    current_window_end = min(window_size, len(frames))
    expected_ack = current_window_start

    while current_window_start < len(frames):
        print("\n--- SENDING WINDOW ---")
        print(frames[current_window_start:current_window_end])
        ack = input("Enter ACK or NACK: ")
        if ack == ACK:
            expected_ack += 1
        else:
            current_window_start = expected_ack
```

## **CS19541-COMPUTER NETWORKS-LAB MANUAL**

### **Practical-7**

```
# Send frames in the current window
for i in range(current_window_start, current_window_end):
    frame_no, data = frames[i]
    print(f"Sent frame {frame_no} with data '{data}'.")

# Simulate Receiver side: Acknowledge frames
print("\n--- RECEIVER WINDOW ---")
for i in range(current_window_start, current_window_end):
    frame_no, data = frames[i]

# Randomly decide to send ACK or NACK (to simulate error)
ack_type = ACK if random.random() > 0.2 else NACK # 20% chance of NACK
if ack_type == ACK:
    print(f"Frame {frame_no} received correctly. Sending {ACK}.")
else:
    print(f"Error in frame {frame_no}. Sending {NACK}.")
    expected_ack = frame_no # Reset to the frame needing retransmission
    break
else:
    # If no errors in the window, move to the next set of frames
    expected_ack = current_window_end

# Update window based on the acknowledgments received
if expected_ack == current_window_end:
    print("All frames in the window acknowledged correctly. Moving window forward.")
    current_window_start += window_size
    current_window_end = min(current_window_start + window_size, len(frames))
else:
    print(f"Resending frames from frame {expected_ack} due to error.")
    current_window_start = expected_ack
    current_window_end = min(current_window_start + window_size, len(frames))

print("\nAll frames sent and acknowledged successfully.")

# User input for window size and message
window_size = int(input("Enter window size: "))
message = input("Enter the message to send: ")

# Run the sliding window protocol
sliding_window_protocol(window_size, message)
```

# CS19541-COMPUTER NETWORKS-LAB MANUAL

## Practical-7

Input:

Enter window size: 3

Enter the message to send: HELLO

Output:

--- SENDING WINDOW ---

Sent frame 0 with data 'H'.

Sent frame 1 with data 'E'.

Sent frame 2 with data 'L'.

--- RECEIVER WINDOW ---

Error in frame 0.

Sending NACK.

Resending frames from frame 0 due to error.

--- SENDING WINDOW ---

Sent frame 0 with data 'H'.

Sent frame 1 with data 'E'.

Sent frame 2 with data 'L'.

--- RECEIVER WINDOW ---

Frame 0 received correctly.

Sending ACK.

Frame 1 received correctly.

Sending ACK.

Frame 2 received correctly.

Sending ACK.

All frames in the window acknowledged correctly.

Moving window forward.

--- SENDING WINDOW ---

Sent frame 3 with data 'L'.

Sent frame 4 with data 'O'.

--- RECEIVER WINDOW ---

Error in frame 3.

Sending NACK.

Resending frames from frame 3 due to error.

--- SENDING WINDOW ---

Sent frame 3 with data 'L'.

Sent frame 4 with data 'O'.

--- RECEIVER WINDOW ---

Frame 3 received correctly.

Sending ACK. Error in frame 4.

Sending NACK.

Resending frames from frame 4 due to error.

## **CS19541-COMPUTER NETWORKS-LAB MANUAL**

### **Practical-7**

--- SENDING WINDOW ---

Sent frame 4 with data 'O'.

--- RECEIVER WINDOW ---

Frame 4 received correctly.

Sending ACK.

All frames in the window acknowledged correctly.

Moving window forward.

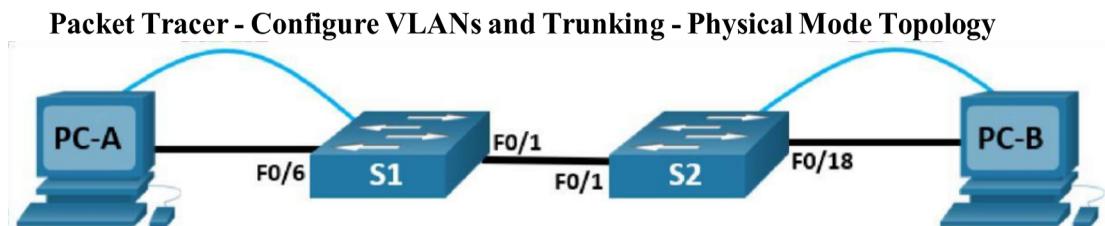
All frames sent and acknowledged successfully.

---

# CS19541-COMPUTER NETWORKS-LAB MANUAL

## Practical-8

**AIM:- a) Simulate Virtual LAN configuration using CISCO Packet Tracer Simulation.**



Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
S1	VLAN 1	192.168.1.11	255.255.255.0	N/A
S2	VLAN 1	192.168.1.12	255.255.255.0	N/A
PC-A	NIC	192.168.10.3	255.255.255.0	192.168.10.1
PC-B	NIC	192.168.10.4	255.255.255.0	192.168.10.1

*Blank Line - no additional information*

### Objectives

**Part 1: Build the Network and Configure Basic Device Settings**

**Part 2: Create VLANs and Assign Switch Ports**

**Part 3: Maintain VLAN Port Assignments and the VLAN Database Part 4: Configure an 802.1Q Trunk between the Switches**

### Background / Scenario

Modern switches use virtual local-area networks (VLANs) to improve network performance by separating large Layer 2 broadcast domains into smaller ones. VLANs can also be used as a security measure by controlling which hosts can communicate. In general, VLANs make it easier to design a network to support the goals of an organization.

VLAN trunks are used to span VLANs across multiple devices. Trunks allow the traffic from multiple VLANs to travel over a single link, while keeping the VLAN identification and segmentation intact.

In this Packet Tracer Physical Mode (PTPM) activity, you will create VLANs on both switches in the topology, assign VLANs to switch access ports, and verify that VLANs are working as expected. You will then create a VLAN trunk between the two switches to allow hosts in the same VLAN to communicate through the trunk, regardless of which switch to which the host is attached.

### Instructions

**Part 1: Build the Network and Configure Basic Device Settings**

#### Step 1: Build the network as shown in the topology.

Attach the devices as shown in the topology diagram, and cable as necessary. a. Click and drag both switch **S1** and **S2** to the **Rack**.

- b. Click and drag both **PC-A** and **PC-B** to the **Table** and use the power button to turn them on.

## **CS19541-COMPUTER NETWORKS-LAB MANUAL**

- c. Provide network connectivity by connecting **Copper Straight-through** cables, as shown in the topology.
- d. Connect **Console Cable** from device **PC-A** to **S1** and from device **PC-B** to **S2**.

### **Step 2: Configure basic settings for each switch.**

- a. From the **Desktop Tab** on each PC, use the **Terminal** to console into each switch and enable privileged EXEC mode.  
*Open configuration window*
- b. Enter configuration mode.
- c. Assign a device name to each switch.
- d. Assign **class** as the privileged EXEC encrypted password.
- e. Assign **cisco** as the console password and enable login.
- f. Assign **cisco** as the vty password and enable login.
- g. Encrypt the plaintext passwords.
- h. Create a banner that warns anyone accessing the device that unauthorized access is prohibited.
- i. Configure the IP address listed in the Addressing Table for VLAN 1 on the switch.  
**Note:** The VLAN 1 address is not grade because you will remove it later in the activity. However, you will need VLAN 1 to test connectivity later in this Part.
- j. Shut down all interfaces that will not be used.
- k. Set the clock on each switch.  
**Note:** The clock setting cannot be graded in Packet Tracer.
- l. Save the running configuration to the startup configuration file.  
*Close configuration window*

### **Step 3: Configure PC hosts.**

From the **Desktop** tab on each **PC**, click **IP Configuration** and enter the addressing information as displayed in the Addressing Table.

### **Step 4: Test connectivity.**

Test network connectivity by attempting to ping between each of the cabled devices.

Questions:

- Can PC-A ping PC-B?
- Can PC-A ping S1?
- Can PC-B ping S2?
- Can S1 ping S2?

*Close configuration window*

# CS19541-COMPUTER NETWORKS-LAB MANUAL

## Part 2: Create VLANs and Assign Switch Ports

In Part 2, you will create Management, Operations, Parking Lot, and Native VLANs on both switches. You will then assign the VLANs to the appropriate interface. The **show vlan** command is used to verify your configuration settings.

### Step 1: Create VLANs on the switches.

From the **Desktop Tab** on each **PC**, use Terminal to continue configuring both network switches.

*Open configuration window*

- Create the VLANs on **S1**.

```
S1(config)# vlan 10
S1(config-vlan)# name Operations
S1(config-vlan)# vlan 20
S1(config-vlan)# name Parking_Lot
S1(config-vlan)# vlan 99
S1(config-vlan)# name Management
S1(config-vlan)# vlan 1000
S1(config-vlan)# name Native
S1(config-vlan)# end
```

- Create the same VLANs on **S2**.
- Issue the **show vlan brief** command to view the list of VLANs on **S1**.

```
S1# show vlan brief
```

VLAN Name	Status	Ports
1 default	active	Fa0/1, Fa0/2, Fa0/3, Fa0/4 Fa0/5, Fa0/6, Fa0/7, Fa0/8 Fa0/9, Fa0/10, Fa0/11, Fa0/12 Fa0/13, Fa0/14, Fa0/15, Fa0/16 Fa0/17, Fa0/18, Fa0/19, Fa0/20 Fa0/21, Fa0/22, Fa0/23, Fa0/24 Gi0/1, Gi0/2
	active	
	active	
	active	
1002 fddi-default	active	
1003 token-ring-default	active	
1004 fddinet-default	active	
1005 trnet-default	active	

Questions:

What is the default VLAN?

What ports are assigned to the default VLAN?

# **CS19541-COMPUTER NETWORKS-LAB MANUAL**

## **Step 2: Assign VLANs to the correct switch interfaces.**

- a. Assign VLANs to the interfaces on **S1**.
  - 1) Assign PC-A to the Operation VLAN.  
S1(config)# **interface f0/6**  
S1(config-if)# **switchport mode access**  
S1(config-if)# **switchport access vlan 10**
  - 2) From VLAN 1, remove the management IP address and configure it on VLAN 99.  
S1(config)# **interface vlan 1**  
S1(config-if)# **no ip address**  
S1(config-if)# **interface vlan 99**  
S1(config-if)# **ip address 192.168.1.11 255.255.255.0**  
S1(config-if)# **end**
- b. Issue the **show vlan brief** command and verify that the VLANs are assigned to the correct interfaces. c. Issue the **show ip interface brief** command.  
Question:  
What is the status of VLAN 99? Explain.
- d. Assign **PC-B** to the Operations VLAN on **S2**.
- e. From VLAN 1, remove the management IP address and configure it on VLAN 99 according to the Addressing Table.
- f. Use the **show vlan brief** command to verify that the VLANs are assigned to the correct interfaces.  
Questions:  
Is S1 able to ping S2? Explain.  
Is PC-A able to ping PC-B? Explain.

## **Part 3: Maintain VLAN Port Assignments and the VLAN Database**

In Part 3, you will change port VLAN assignments and remove VLANs from the VLAN database.

### **Step 1: Assign a VLAN to multiple interfaces.**

From the **Desktop Tab** on each PC, use **Terminal** to continue configuring both network switches.

*Open configuration window*

- a. On S1, assign interfaces F0/11 – 24 to VLAN99.  
S1(config)# **interface range f0/11-24**  
S1(config-if-range)# **switchport mode access**  
S1(config-if-range)# **switchport access vlan 99**  
S1(config-if-range)# **end**
- b. Issue the **show vlan brief** command to verify VLAN assignments.
- c. Reassign F0/11 and F0/21 to VLAN 10.
- d. Verify that VLAN assignments are correct.

### **Step 2: Remove a VLAN assignment from an interface.**

- a. Use the **no switchport access vlan** command to remove the VLAN 99 assignment to F0/24.  
S1(config)# **interface f0/24**  
S1(config-if)# **no switchport access vlan**  
S1(config-if)# **end**
- b. Verify that the VLAN change was made.  
Question:

## **CS19541-COMPUTER NETWORKS-LAB MANUAL**

Which VLAN is F0/24 now associated with?

### **Step 3: Remove a VLAN ID from the VLAN database.**

- a. Add VLAN 30 to interface F0/24 without issuing the global VLAN command.

S1(config)# **interface f0/24**

S1(config-if)# **switchport access vlan 30**

% Access VLAN does not exist. Creating vlan 30

**Note:** Current switch technology no longer requires that the **vlan** command be issued to add a VLAN to the database. By assigning an unknown VLAN to a port, the VLAN will be created and added to the VLAN database.

- b. Verify that the new VLAN is displayed in the VLAN table.

Question:

What is the default name of VLAN 30?

- c. Use the **no vlan 30** command to remove VLAN 30 from the VLAN database.

S1(config)# **no vlan 30**

S1(config)# **end**

- d. Issue the **show vlan brief** command. F0/24 was assigned to VLAN 30.

Question:

After deleting VLAN 30 from the VLAN database, why is F0/24 no longer displayed in the output of the **show vlan brief** command? What VLAN is port F0/24 now assigned to? What happens to the traffic destined to the host that is attached to F0/24?

- e. On interface F0/24, issue the **no switchport access vlan** command.

- f. Issue the **show vlan brief** command to determine the VLAN assignment for F0/24.

Questions:

To which VLAN is F0/24 assigned?

**Note:** Before removing a VLAN from the database, it is recommended that you reassign all the ports assigned to that VLAN.

Why should you reassign a port to another VLAN before removing the VLAN from the VLAN database?

*Close configuration window.*

### **Part 4: Configure an 802.1Q Trunk Between the Switches**

In Part 4, you will configure interface F0/1 to use the Dynamic Trunking Protocol (DTP) to allow it to negotiate the trunk mode. After this has been accomplished and verified, you will disable DTP on interface F0/1 and manually configure it as a trunk.

#### **Step 1: Use DTP to initiate trunking on F0/1.**

The default DTP mode of a 2960 switch port is dynamic auto. This allows the interface to convert the link to a trunk if the neighboring interface is set to trunk or dynamic desirable mode.

*Open configuration window*

- a. On **S1**, set F0/1 to negotiate trunk mode.

S1(config)# **interface f0/1**

S1(config-if)# **switchport mode dynamic desirable**

# CS19541-COMPUTER NETWORKS-LAB MANUAL

Sep 19 02:51:47.257: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up

Sep 19 02:51:47.291: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan99, changed state to up

You should also receive link status messages on S2.

S2#

Sep 19 02:42:19.424: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up

Sep 19 02:42:21.454: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan99, changed state to up

Sep 19 02:42:22.419: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up

- b. On **S1** and **S2**, issue the **show vlan brief** command. Interface F0/1 is no longer assigned to VLAN 1. Trunked interfaces are not listed in the VLAN table.
- c. Issue the **show interfaces trunk** command to view trunked interfaces. Notice that the mode on **S1** is set to desirable, and the mode on **S2** is set to auto.

S1# **show interfaces trunk**

S2# **show interfaces trunk**

**Note:** By default, all VLANs are allowed on a trunk. The **switchport trunk** command allows you to control what VLANs have access to the trunk. For this activity, keep the default settings. This allows all VLANs to traverse F0/1.

*Close configuration window*

- d. Verify that VLAN traffic is traveling over trunk interface F0/1.

Questions:

Can S1 ping S2?

Can PC-A ping PC-B?

Can PC-A ping S1?

Can PC-B ping S2?

## Step 2: Manually configure trunk interface F0/1.

The **switchport mode trunk** command is used to manually configure a port as a trunk. This command should be issued on both ends of the link.

- a. On interface F0/1, change the switchport mode to force trunking. Make sure to do this on both switches.

*Open configuration window*

S1(config)# **interface f0/1**

S1(config-if)# **switchport mode trunk**

- b. Issue the **show interfaces trunk** command to view the trunk mode. Notice that the mode changed from **desirable** to **on**.

S1# **show interfaces trunk**

- c. Modify the trunk configuration on both switches by changing the native VLAN from VLAN 1 to VLAN 1000.

S1(config)# **interface f0/1**

S1(config-if)# **switchport trunk native vlan 1000**

- d. Issue the **show interfaces trunk** command to view the trunk. Notice the Native VLAN information is updated.

S2# **show interfaces trunk**

Questions:

Why might you want to manually configure an interface to trunk mode instead of using DTP?

Why might you want to change the native VLAN on a trunk?

*Close configuration window*

## **CS19541-COMPUTER NETWORKS-LAB MANUAL**

### **Reflection Questions**

1. What is needed to allow hosts on VLAN 10 to communicate to hosts on VLAN 99?
  2. What are some primary benefits that an organization can receive through effective use of VLANs?
- 
- 

### **b) Design and configure a VLAN for the below given scenario.**

There are 10 faculty in Robotics department, sitting in 3 different blocks. Design and configure a Virtual LAN for Robotics department (using switch and Ethernet cables) so that all the faculty are logically in the same LAN.

### **Student observation:-**

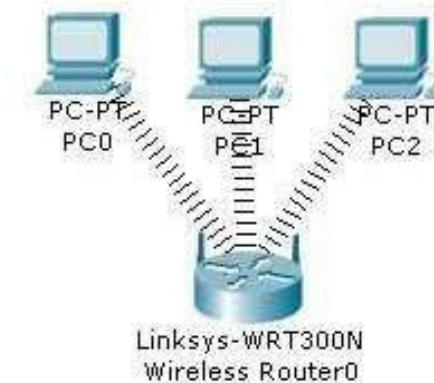
- a) Draw and Label the VLAN for Qb).
- b) Show the ip configuration for each device.
- c) Write the commands used for VLAN configuration in switch.

# CS19541-COMPUTER NETWORKS-LAB MANUAL

## Practical-8

### AIM:-b) Configuration of Wireless LAN using CISCO Packet Tracer.

Design a topology with three PCs connected from Linksys Wireless routers.



Perform following configuration:-

- Configure Static IP on PC and Wireless Router
- Set SSID to MotherNetwork
- Set IP address of router to 192.168.0.1, PC0 to 192.168.0.2, PC1 to 192.168.0.3 and PC2 to 192.168.0.4.
- Secure your network by configuring WAP key on Router
- Connect PC by using WAP key

To complete these tasks follow these step by step instructions:-

Step1:- Click on wireless router,

- Select Administration tab from top Menu, set username and password to admin and click on Save Setting.



## CS19541-COMPUTER NETWORKS-LAB MANUAL

- Next click on wireless tab and set default SSID to MotherNetwork.
- Now Select wireless security and change Security Mode to WEP



- Set Key1 to 0123456789

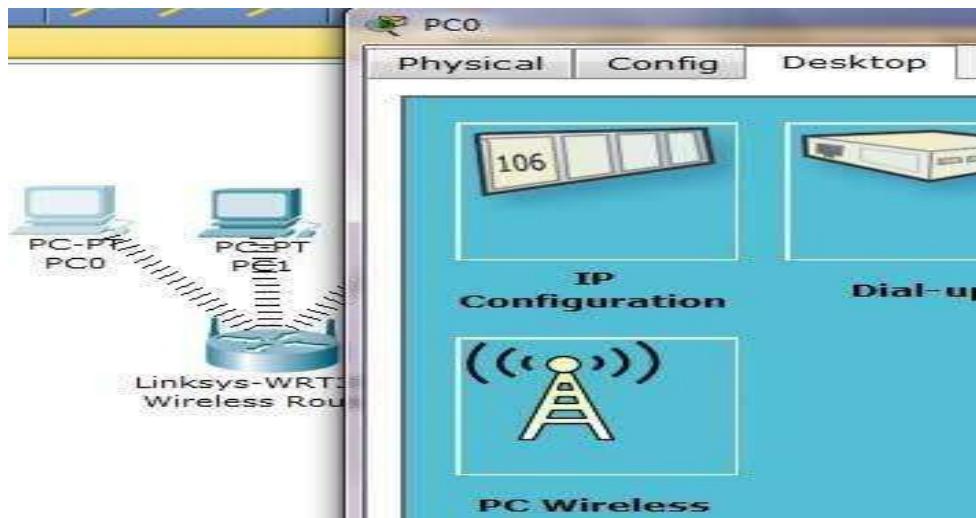


- Again go in the end of page and Click on Save Setting
- Now we have completed all given task on Wireless router. Now configure the static IP on all three PC's
- Double click on pc select Desktop tab click on IP configuration select Static IP and set IP as given below

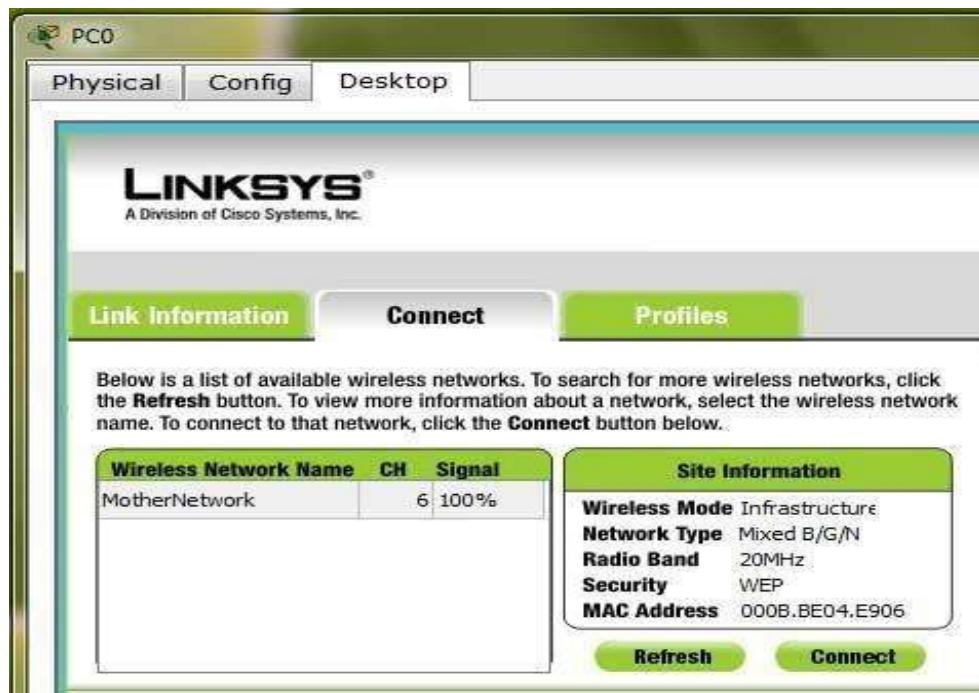
PC	IP	Subnet Mask	Default Gateway
PC0	192.168.0.2	255.255.255.0	192.168.0.1
PC1	192.168.0.3	255.255.255.0	192.168.0.1
PC2	192.168.0.4	255.255.255.0	192.168.0.1

## CS19541-COMPUTER NETWORKS-LAB MANUAL

- Now it's time to connect PC's from Wireless router. To do so click PC select Desktop click on PC Wireless



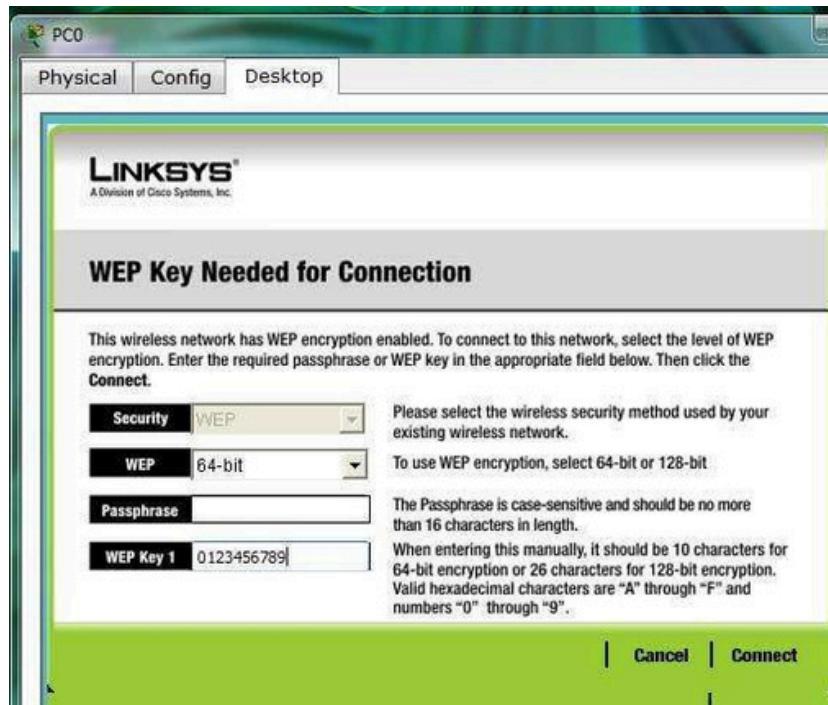
- Click on connect tab and click on Refresh button



As you can see in image that Wireless device is accessing MotherNetwork on CH 6 and signal strength is 100%. In left side you can see that WEP security is configured in network. Click on connect button to connect MotherNetwork

- It will ask for WAP key insert 0123456789 and click connect

## CS19541-COMPUTER NETWORKS-LAB MANUAL



It will connect you with wireless router.

As you can see in image below that system is connected. And PCI card is active.



- Repeat same process on PC1 and PC2.

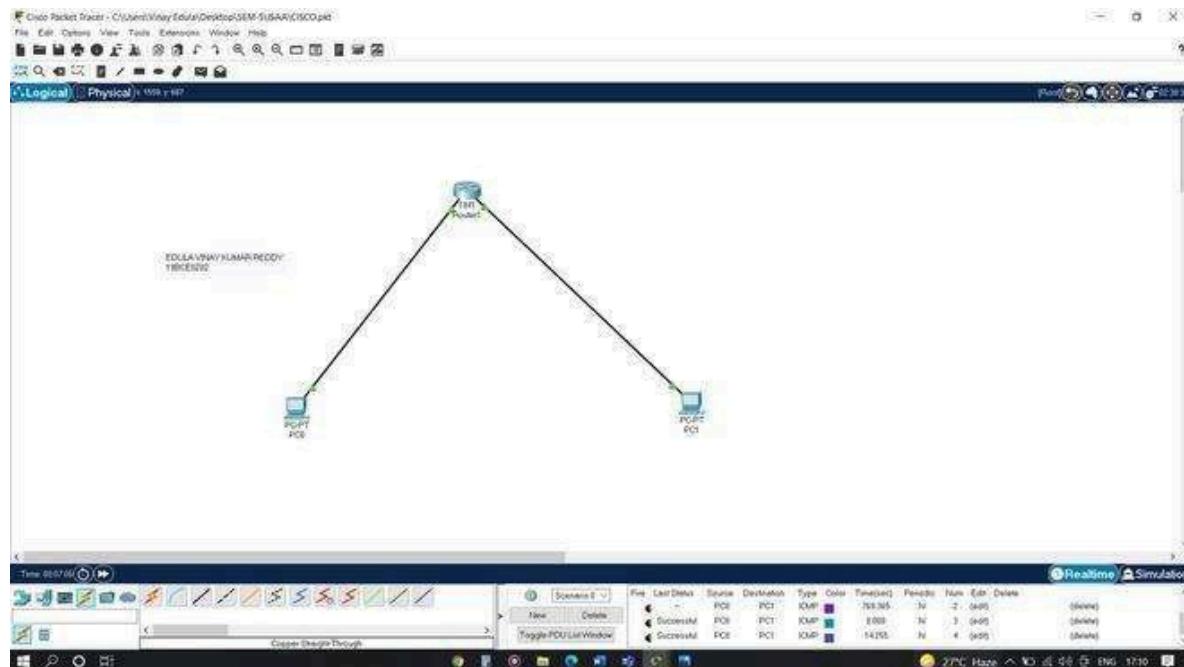
# CS19541-COMPUTER NETWORKS-LAB MANUAL

## Practical-10

**AIM:-a) Internetworking with routers in CISCO PACKET TRACER simulator.**

**d) Design and configure a simple internetwork using a router.**

In this network, a router and 2 PCs are used. Computers are connected with routers using a copper straight-through cable. After forming the network, to check network connectivity a simple PDU is transferred from PC0 to PC1.



**Procedure:**

**Step-1(Configuring Router1):**

1. Select the router and Open CLI.
2. Press ENTER to start configuring Router1.
3. Type enable to activate the privileged mode.

**Router1 Command Line Interface:**

```
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#ip address 192.168.10.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up
Router(config-if)#interface FastEthernet0/1
Router(config-if)#ip address 192.168.20.1 255.255.255.0
Router(config-if)#no shutdown
```

**Step-2(Configuring PCs):**

## **CS19541-COMPUTER NETWORKS-LAB MANUAL**

1. Assign IP Addresses to every PC in the network.
2. Select the PC, Go to the desktop and select IP Configuration and assign an IP address, Default gateway, Subnet Mask
3. Assign the default gateway of PC0 as 192.168.10.1.
4. Assign the default gateway of PC1 as 192.168.20.1.

### **Step-3(Connecting PCs with Router):**

1. Connect FastEthernet0 port of PC0 with FastEthernet0/0 port of Router1 using a copper straight-through cable.
2. Connect FastEthernet0 port of PC1 with FastEthernet0/1 port of Router1 using a copper straight-through cable.

#### **Router Configuration Table:**

<b>Device Name</b>	<b>IP address FastEthernet0 /0</b>	<b>Subnet Mask</b>	<b>IP Address FastEthernet0/1</b>	<b>Subnet Mask</b>
Router1	192.168.10.1	255.255.255.0	192.168.20.1	255.255.255.0

#### **PC Configuration Table:**

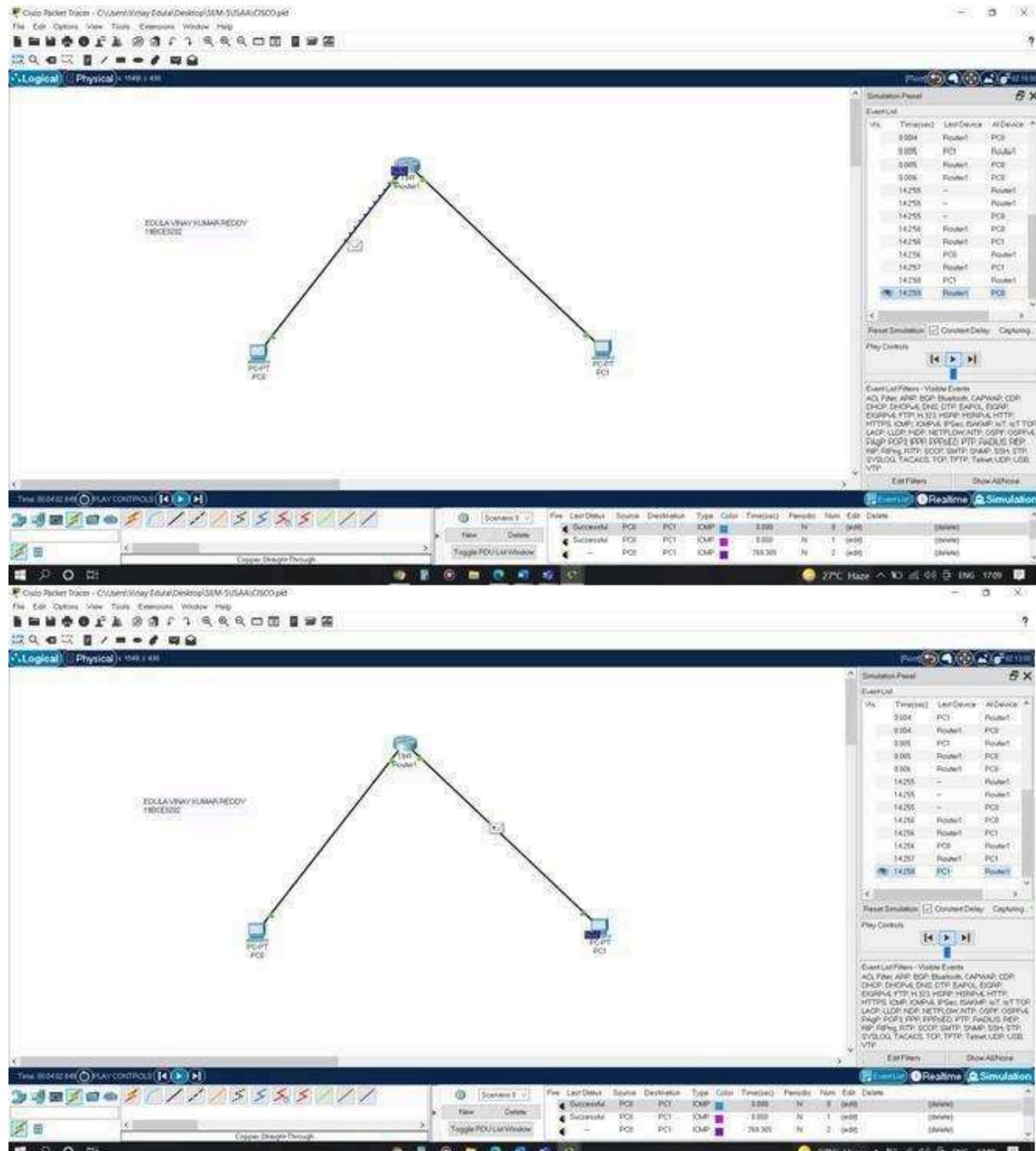
<b>Device Name</b>	<b>IP address</b>	<b>Subnet Mask</b>	<b>Gateway</b>
PC 0	192.168.10.2	255.255.255.0	192.168.10.1
PC 1	192.168.20.2	255.255.255.0	192.168.20.1

# CS19541-COMPUTER NETWORKS-LAB MANUAL

## **Designed Network topology:**

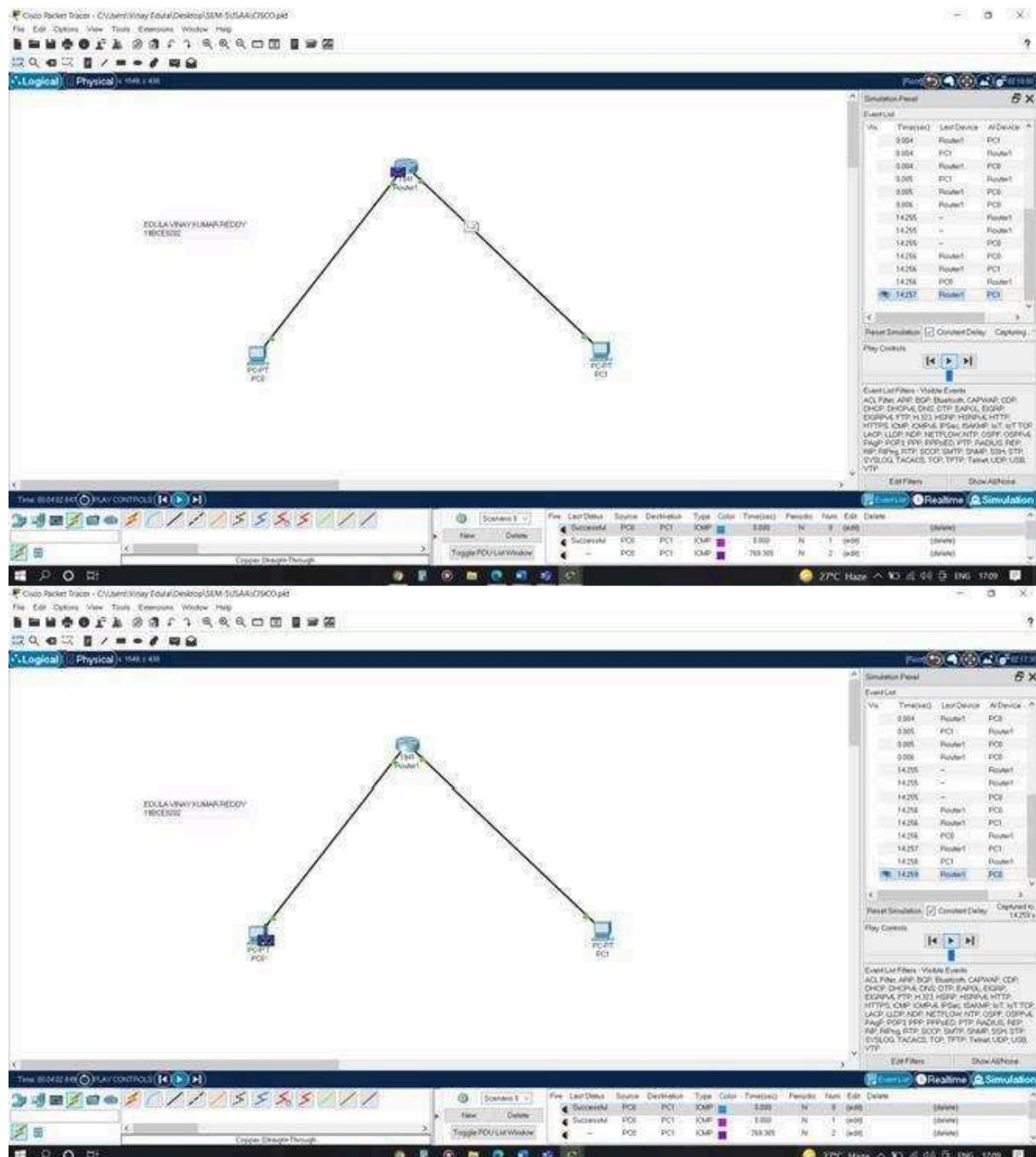
Simulation of Designed Network Topology:

### **Sending a PDU From PC0 to PC1:**



# CS19541-COMPUTER NETWORKS-LAB MANUAL

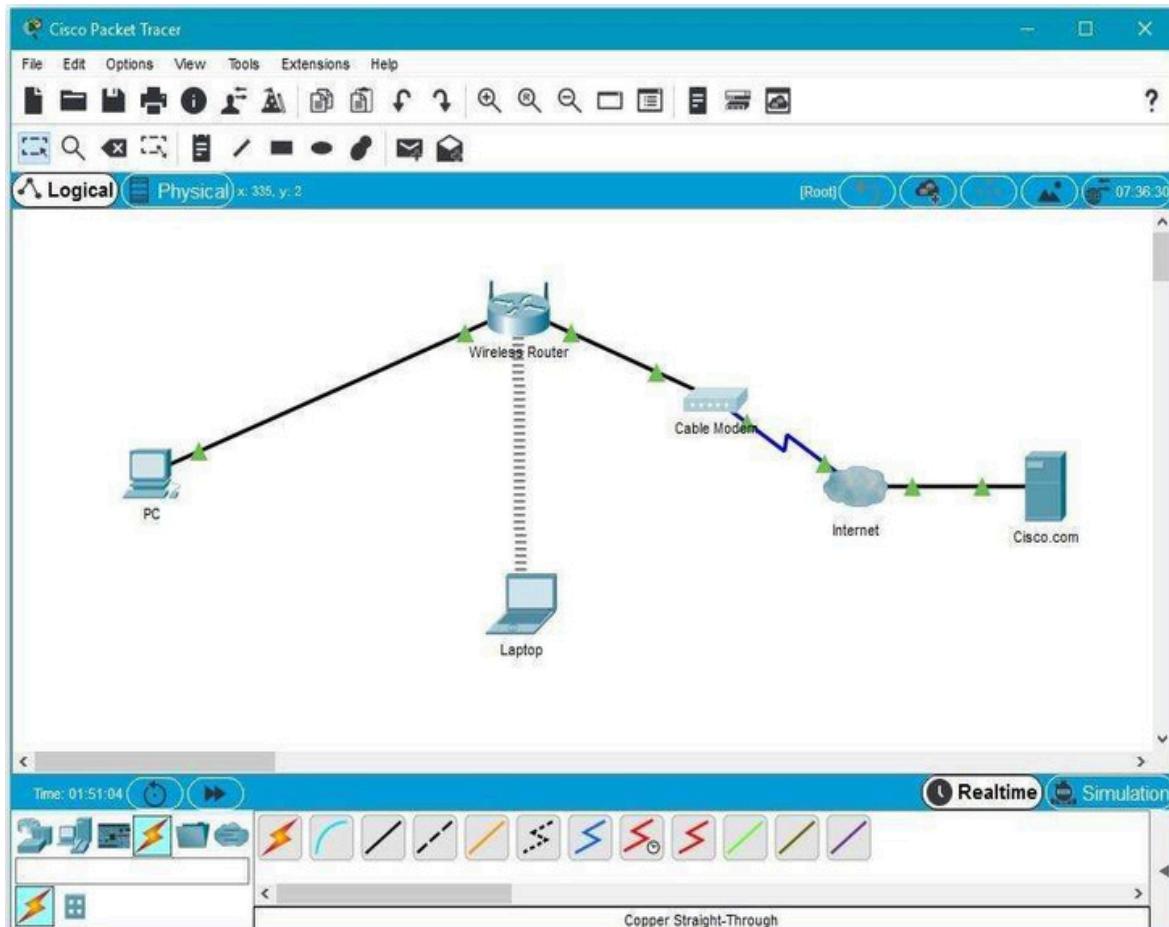
## Acknowledgment From PC1 to PC0:



# CS19541-COMPUTER NETWORKS-LAB MANUAL

## Practical 10

**AIM:- b)** Design and configure an internetwork using wireless router, DHCP server and internet cloud.



**Addressing Table**

Device	Interface	IP Address	Subnet Mask	Default Gateway
PC	Ethernet0	DHCP		192.168.0.1
Wireless Router	LAN	192.168.0.1	255.255.255.0	
Wireless Router	Internet	DHCP		
Cisco.com Server	Ethernet0	208.67.220.220	255.255.255.0	
Laptop	Wireless0	DHCP		

### **Objectives**

**Part 1: Build a Simple Network in the Logical Topology Workspace**

# CS19541-COMPUTER NETWORKS-LAB MANUAL

- Part 2: Configure the Network Devices**
- Part 3: Test Connectivity between Network Devices**
- Part 4: Save the File and Close Packet Tracer**

## **Part 1: Build a Simple Network in the Logical Topology Workspace**

### **Step 1: Launch Packet Tracer.**

#### **Step 2: Build the topology**

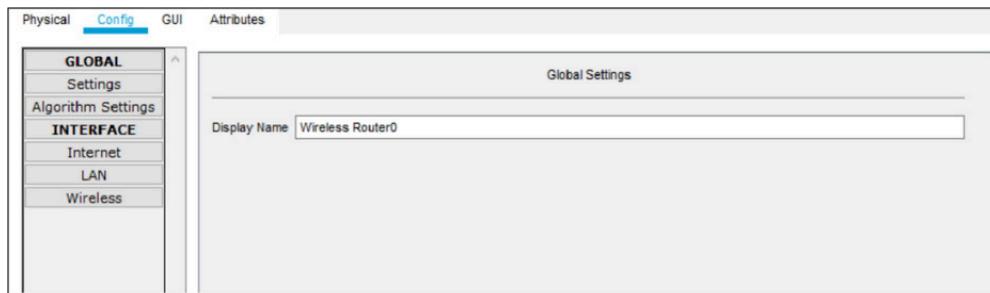
- a. Add network devices to the workspace.

Using the device selection box, add the network devices to the workspace as shown in the topology diagram.

To place a device onto the workspace, first choose a device type from the **Device-Type Selection** box. Then, click on the desired device model from the **Device-Specific Selection** box. Finally, click on a location in the workspace to put your device in that location. If you want to cancel your selection, click the **Cancel** icon for that device. Alternatively, you can click and drag a device from the **Device-Specific Selection** box onto the workspace.

- b. Change display names of the network devices.

To change the display names of the network devices click on the device icon on the Packet Tracer **Logical** workspace, then click on the **Config** tab in the device configuration window. Type the new name of the device into the **Display Name** box as show in the figure below.



- c. Add the physical cabling between devices on the workspace

Using the device selection box, add the physical cabling between devices on the workspace as shown in the topology diagram.

The PC will need a copper straight-through cable to connect to the wireless router. Select the copper straight-through cable in the device selection box and attach it to the FastEthernet0 interface of the PC and the Ethernet 1 interface of the wireless router.

# CS19541-COMPUTER NETWORKS-LAB MANUAL

The wireless router will need a copper straight-through cable to connect to the cable modem. Select the copper straight-through cable in the device-selection box and attach it to the Internet interface of the wireless router and the Port 1 interface of the cable modem.

The cable modem will need a coaxial cable to connect to the Internet cloud. Select the coaxial cable in the device-selection box and attach it to the Port 0 interface of the cable modem and the coaxial interface of the Internet cloud.

The Internet cloud will need copper straight-through cable to connect to the Cisco.com server. Select the copper straight-through cable in the device-selection box and attach it to the Ethernet interface of the Internet cloud and the FastEthernet0 interface of the Cisco.com server.

## **Part 2: Configure the Network Devices**

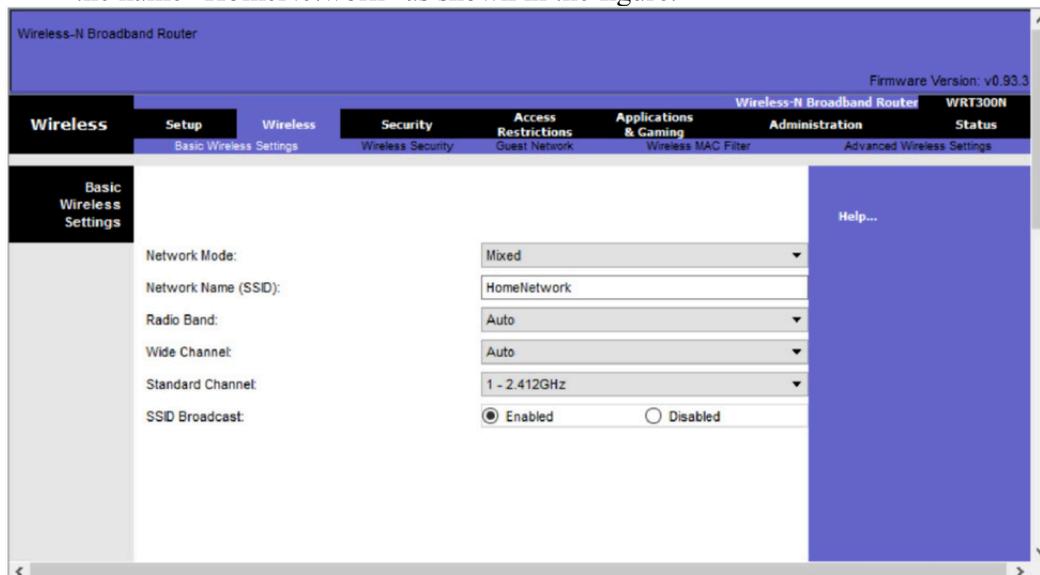
### **Step 1: Configure the wireless router**

- Create the wireless network on the wireless router

Click on the **Wireless Router** icon on the Packet Tracer **Logical** workspace to open the device configuration window.

In the wireless router configuration window, click on the **GUI** tab to view configuration options for the wireless router.

Next, click on the **Wireless** tab in the GUI to view the wireless settings. The only setting that needs to be changed from the defaults is the **Network Name (SSID)**. Here, type the name “HomeNetwork” as shown in the figure.



# CS19541-COMPUTER NETWORKS-LAB MANUAL

Configure the Internet connection on the wireless router  
Click on the **Setup** tab in the wireless router GUI.

In the **DHCP Server** settings verify that the **Enabled** button is selected and configure the static IP address of the DNS server as 208.67.220.220 as shown in the figure.

- b. Click on the **Save Settings** tab.

The screenshot shows the configuration interface for a Wireless-N Broadband Router (WRT300N). The top navigation bar includes tabs for Setup, Wireless, Security, Access Restrictions, Applications & Gaming, Administration, and Status. The Firmware Version is v0.93.3. The main menu on the left has sections for Internet Setup and Network Setup. In the Internet Setup section, the 'Internet Connection type' is set to 'Automatic Configuration - DHCP'. In the Network Setup section, under Router IP, the IP Address is 192.168.0.1 and the Subnet Mask is 255.255.255.0. Under DHCP Server Settings, the 'DHCP Server' is enabled (radio button selected), and the Start IP Address is 192.168.0.100 with a maximum of 50 users. The IP Address Range is 192.168.0.100 - 149. The Client Lease Time is set to 0 minutes (0 means one day). Static DNS 1 is configured with values 208, 67, 220, 220. Other fields for Static DNS 2, 3, and WINS are empty.

## Step 2: Configure the laptop

- a. Configure the Laptop to access the wireless network

Click on the Laptop icon on the Packet Tracer **Logical** workspace and in the laptop configuration windows select the **Physical** tab.

In the **Physical** tab you will need to remove the Ethernet copper module and replace it with the Wireless WPC300N module.

To do this, you first power the Laptop off by clicking the power button on the side of the laptop. Then remove the currently installed Ethernet copper module by clicking on the module on the side of the laptop and dragging it to the **MODULES** pane on the left of the laptop window. Then install the Wireless WPC300N module by clicking on it in the

**MODULES** pane and dragging it to the empty module port on the side of the laptop. Power the laptop back on by clicking on the Laptop power button again.

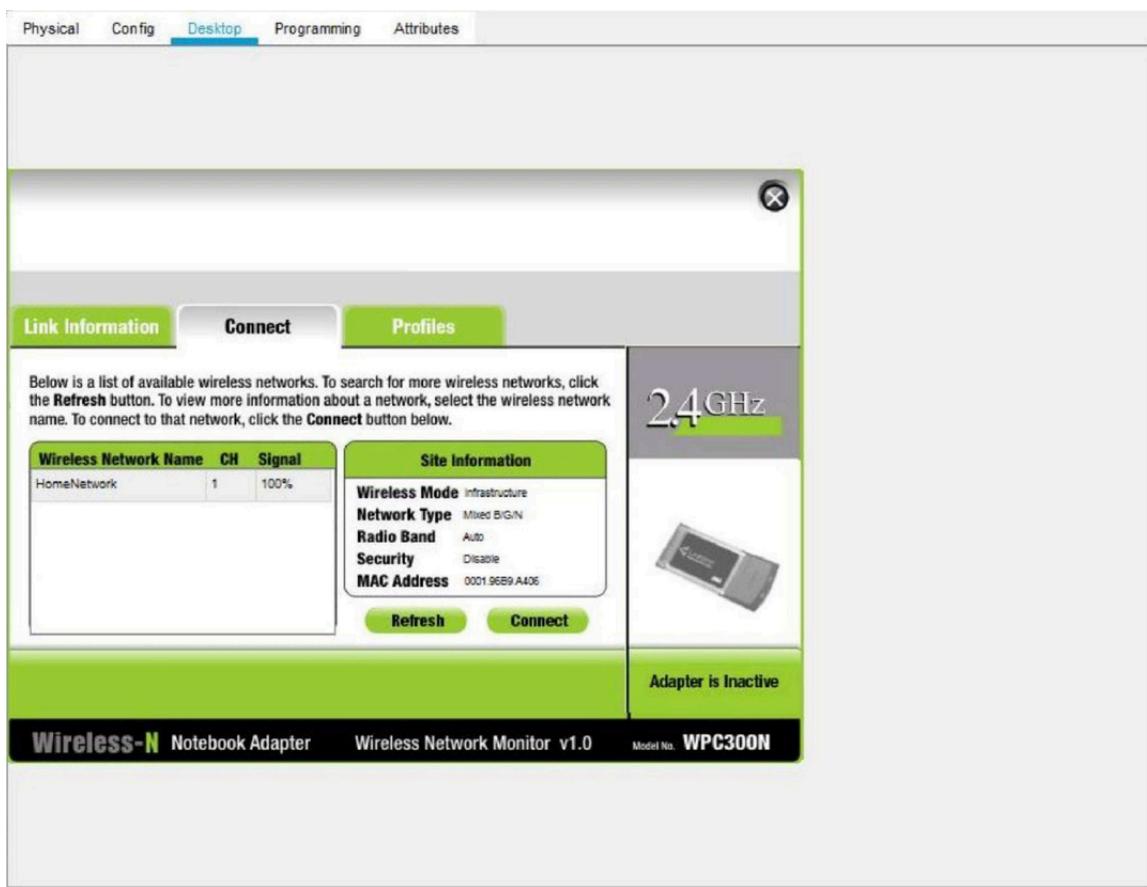
## CS19541-COMPUTER NETWORKS-LAB MANUAL

With the wireless module installed, the next task is to connect the laptop to the wireless network.

Click on the **Desktop** tab at the top of the Laptop configuration window and select the **PC Wireless** icon.

Once the Wireless-N Notebook Adapter settings are visible, select the **Connect** tab. The wireless network “HomeNetwork” should be visible in the list of wireless networks as shown in the figure.

Select the network, and click on the **Connect** tab found below the **Site Information** pane.



### Step 3: Configure the PC

a. Configure the PC for the wired network

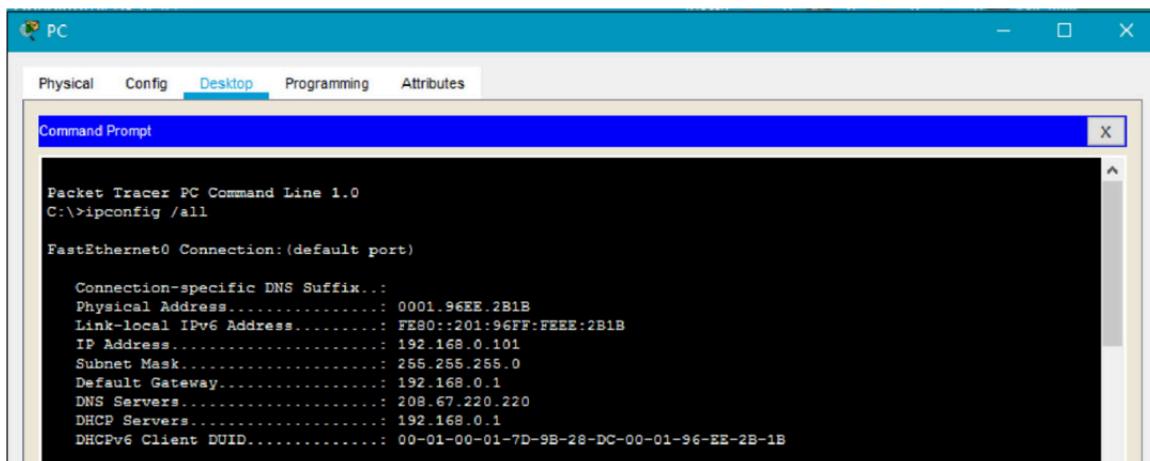
Click on the **PC** icon on the Packet Tracer **Logical** workspace and select the **Desktop** tab and then the **IP Configuration** icon.

In the IP Configuration window, select the **DCHP** radio button as shown in the figure so that the PC will use DCHP to receive an IPv4 address from the wireless router. Close the IP Configuration window.

# CS19541-COMPUTER NETWORKS-LAB MANUAL



Click on the Command Prompt icon. Verify that the PC has received an IPv4 address by issuing the **ipconfig /all** command from the command prompt as shown in the figure. The PC should receive an IPv4 address in the 192.168.0.x range.



## **Step 4: Configure the Internet cloud**

- Install network modules if necessary

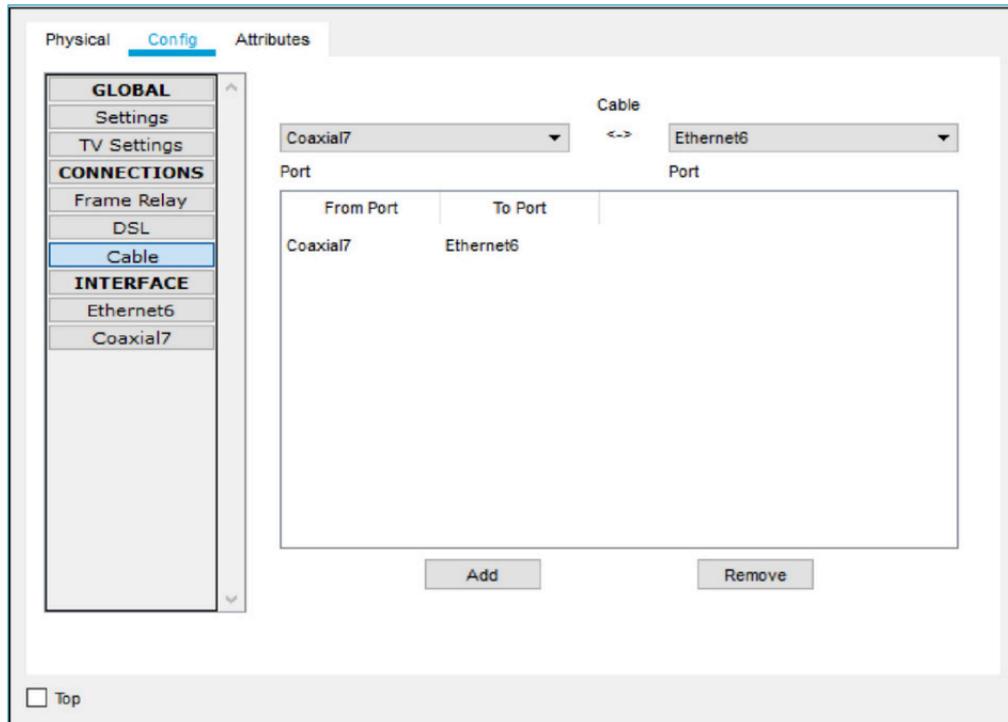
Click on the **Internet Cloud** icon on the Packet Tracer **Logical** workspace and then click on the **Physical** tab. The cloud device will need two modules if they are not already installed. The PT-CLOUD-NM-1CX which is for the cable modem service connection and the PT-CLOUD-NM-1CFE which is for a copper Ethernet cable connection. If these modules are missing, power off the physical cloud devices by clicking on the power button and drag each module to an empty module port on the device and then power the device back on.

- Identify the From and To Ports

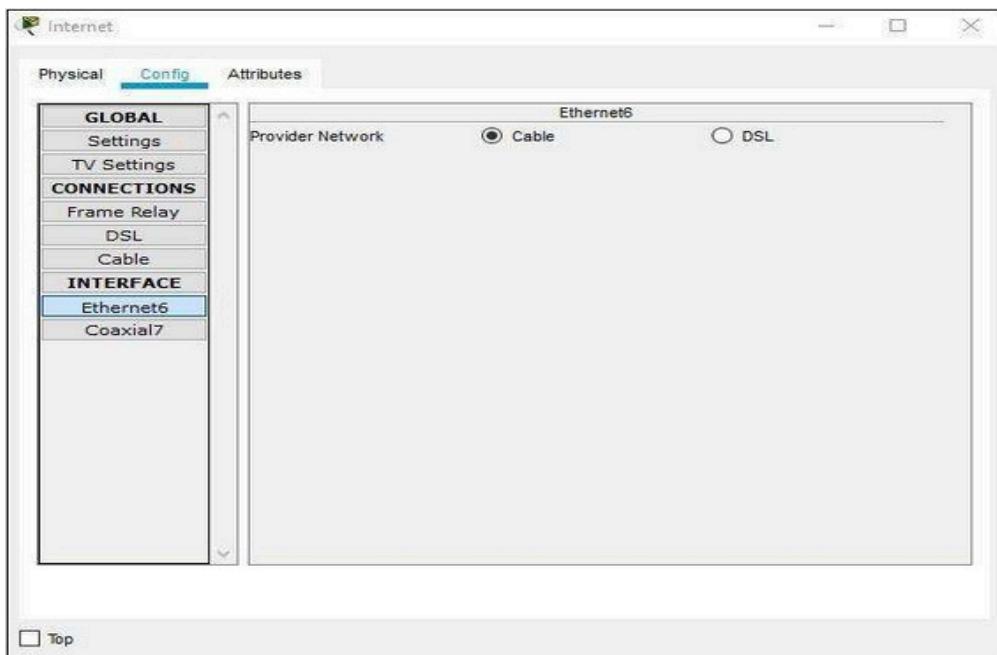
Click on the **Config** tab in the Cloud device window. In the left pane click on **Cable** under **CONNECTIONS**. In the first drop down box choose Coaxial and in the second drop down box choose

Ethernet then click the **Add** button to add these as the **From Port** and **To Port** as shown in the figure.

## CS19541-COMPUTER NETWORKS-LAB MANUAL



- c. Identify the type of provider  
While still in the **Config** tab click Ethernet under **INTERFACE** in the left pane. In the Ethernet configuration window select **Cable** as the Provider Network as shown in the figure.



# CS19541-COMPUTER NETWORKS-LAB MANUAL

## Step 5: Configure the Cisco.com server

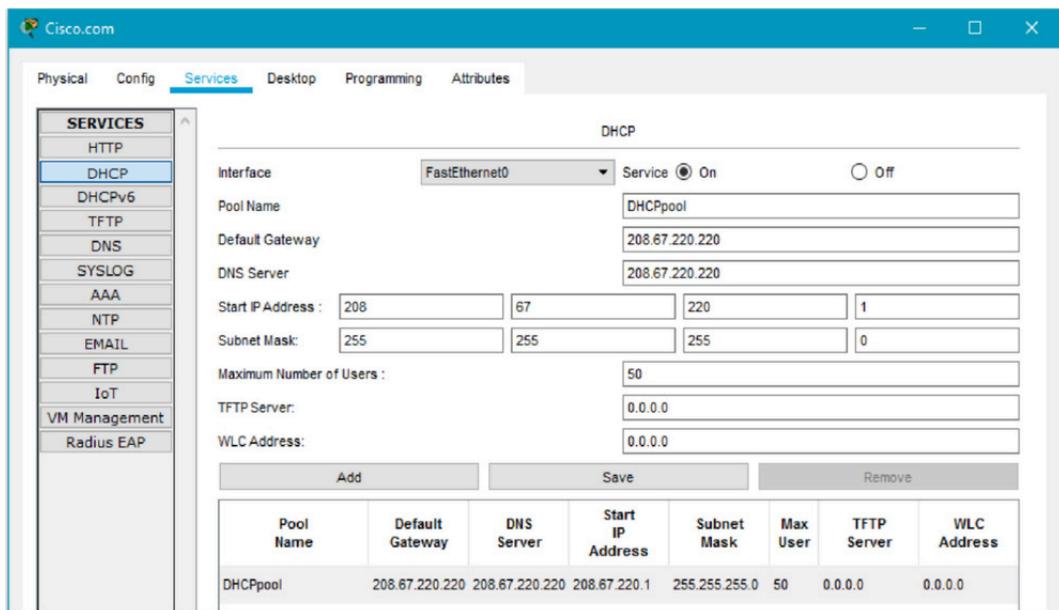
- Configure the Cisco.com server as a DHCP server

Click on the Cisco.com server icon on the Packet Tracer **Logical** workspace and select the **Services** tab. Select **DHCP** from the **SERVICES** list in the left pane.

In the DHCP configuration window, configure a DHCP as shown in the figure with the following settings.

- Click **On** to turn the DHCP service on
- Pool name: DHCPpool
- Default Gateway: 208.67.220.220
- DNS Server: 208.67.220.220
- Starting IP Address: 208.67.220.1
- Subnet Mask 255.255.255.0
- Maximum number of Users: 50

Click **Add** to add the pool



- Configure the Cisco.com server as a DNS server to provide domain name to IPv4 address resolution.

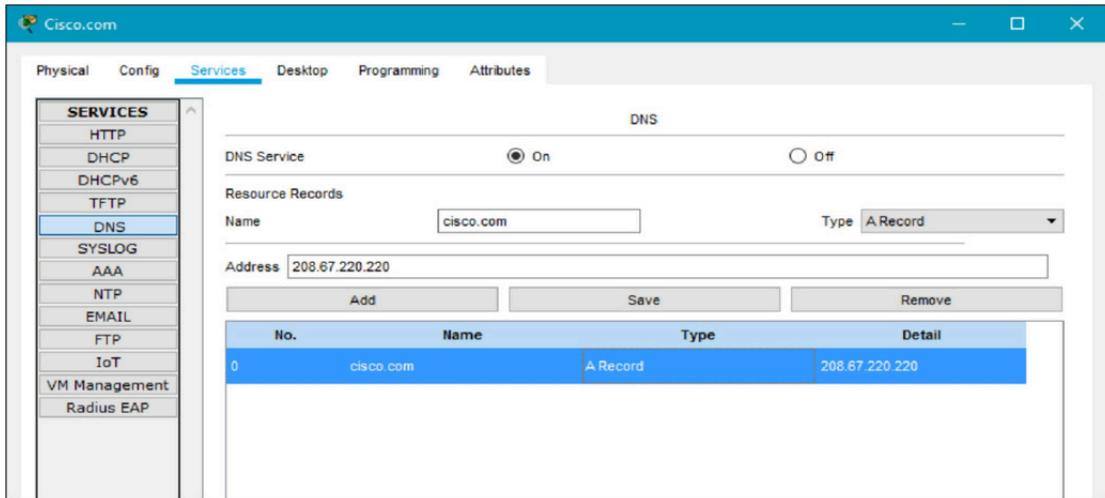
While still in the **Services** tab, select **DNS** from the **SERVICES** listed in the left pane.

Configure the DNS service using the following settings as shown in the figure.

- Click **On** to turn the DNS service on
- Name: Cisco.com
- Type: A Record
- Address: 208.67.220.220

Click **Add** to add the DNS service settings

# CS19541-COMPUTER NETWORKS-LAB MANUAL



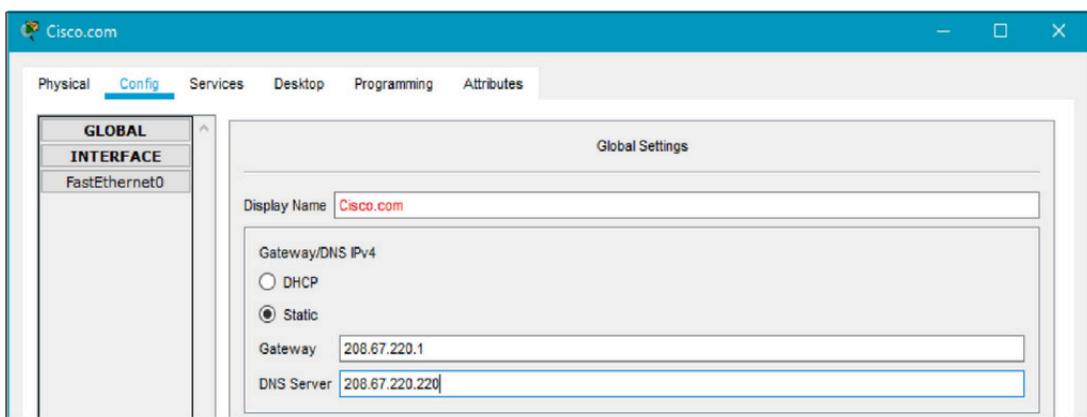
c. Configure the Cisco.com server Global settings.

Select the **Config** tab.

Click on **Settings** in left pane.

Configure the Global settings of the server as follows:

- Select **Static**
- Gateway: 208.67.220.1
- DNS Server: 208.67.220.220



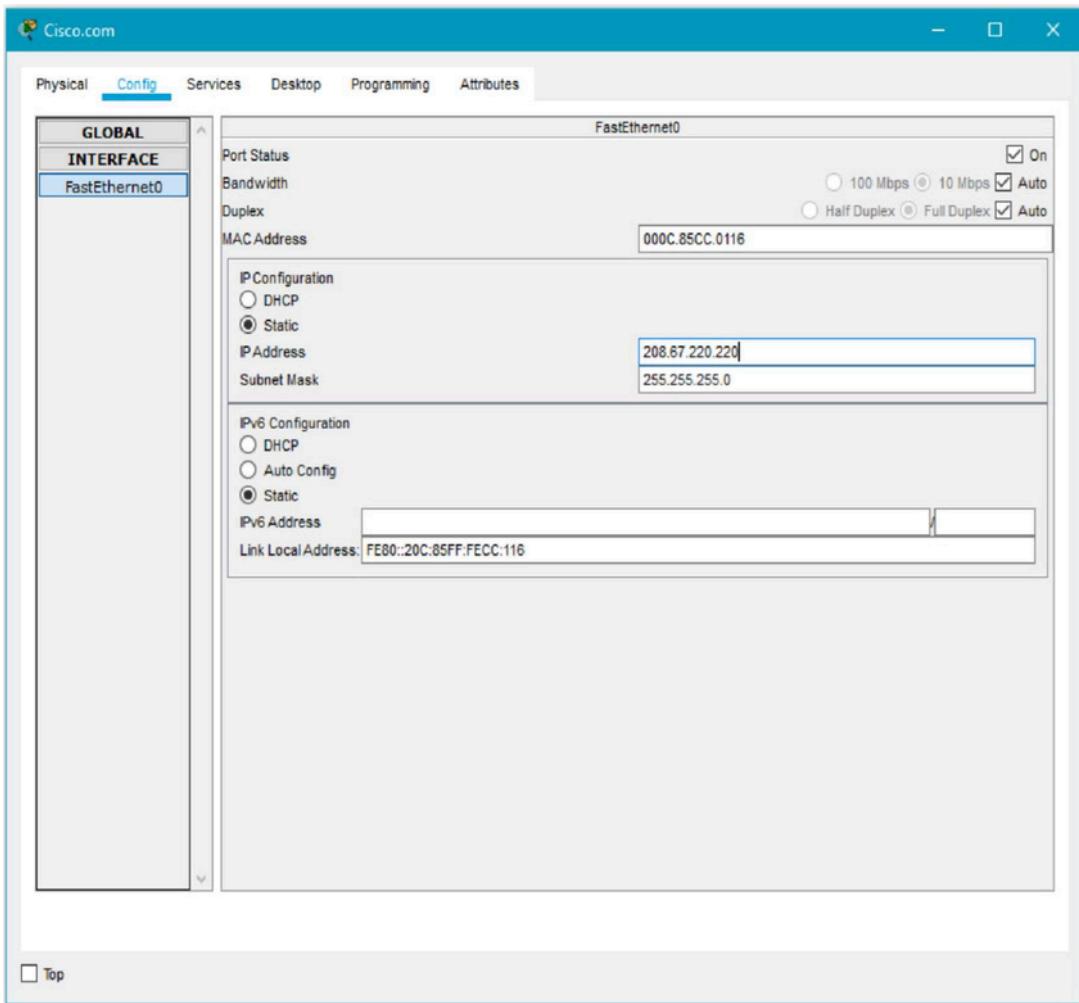
d. Configure the Cisco.com server FastEthernet0 Interface settings.

Click on **Fast Ethernet** in left pane of the **Config** tab

Configure the Fast Ethernet Interface settings of the server as follows:

- Select **Static** under IP Configuration
- IP Address: 208.67.220.220
- Subnet Mask: 255.255.255.0

# CS19541-COMPUTER NETWORKS-LAB MANUAL



## Part 3: Verify Connectivity

### **Step 1: Refresh the IPv4 settings on the PC**

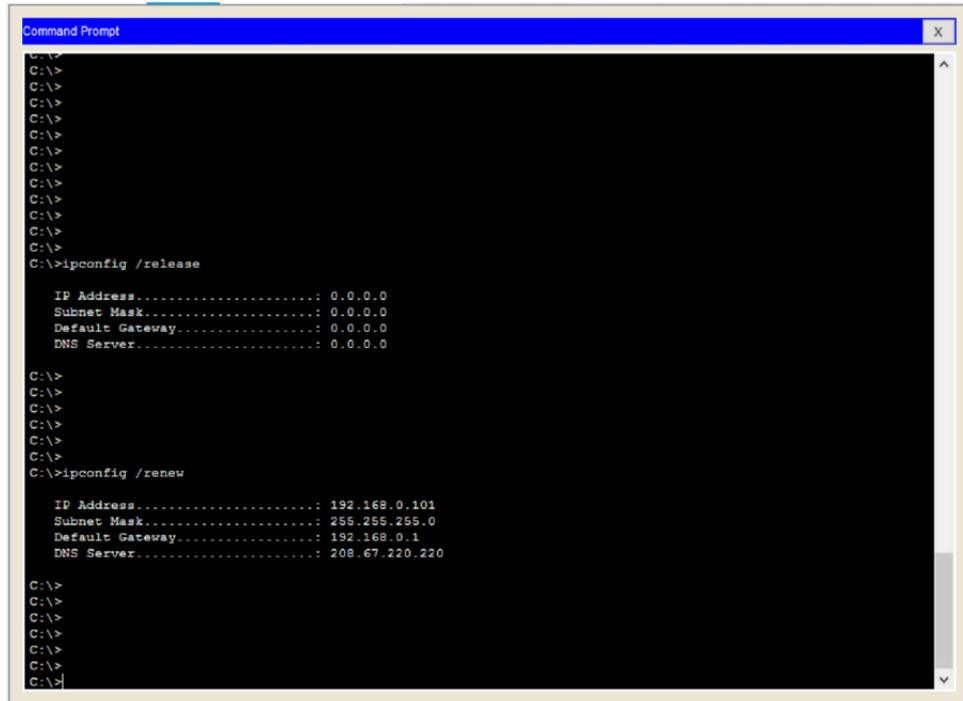
- Verify that the PC is receiving IPv4 configuration information from DHCP.

Click on the **PC** on the Packet Tracer **Logical** workspace and then the select the **Desktop** tab of the PC configuration window.

Click on the **Command Prompt** icon

In the command prompt refresh the IP settings by issuing the commands **ipconfig /release** and then **ipconfig /renew**. The output should show that the PC has an IP address in the 192.168.0.x range, a subnet mask, a default gateway, and DNS server address as shown in the figure.

# CS19541-COMPUTER NETWORKS-LAB MANUAL



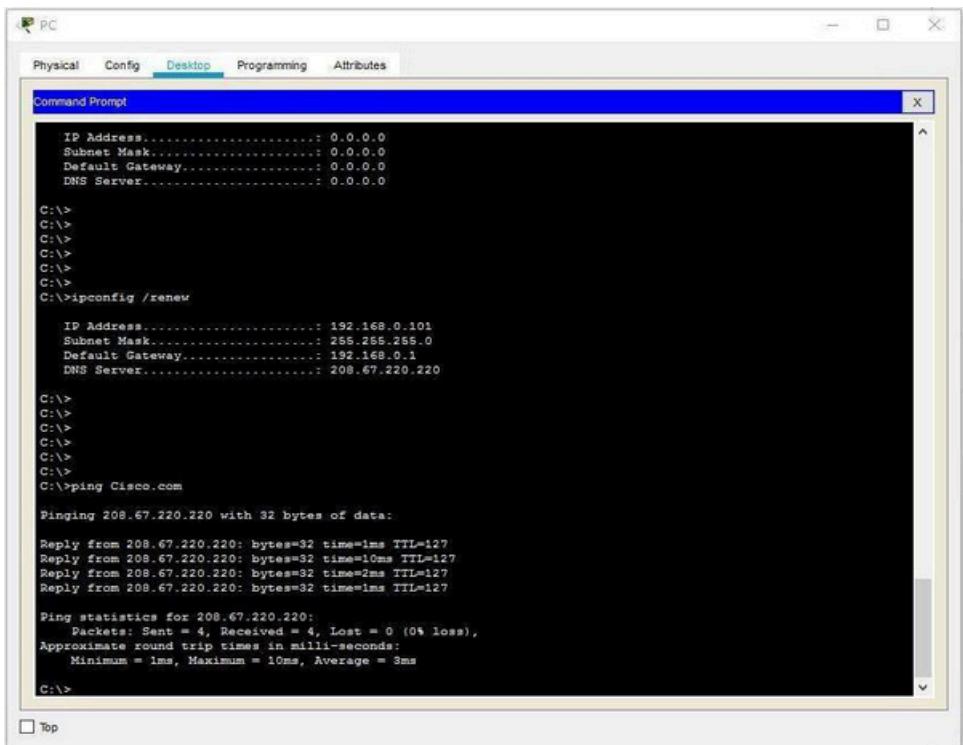
```
Command Prompt
C:\>
C:\>ipconfig /release
IP Address.....: 0.0.0.0
Subnet Mask....: 0.0.0.0
Default Gateway.: 0.0.0.0
DNS Server.....: 0.0.0.0

C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>ipconfig /renew
IP Address.....: 192.168.0.101
Subnet Mask....: 255.255.255.0
Default Gateway.: 192.168.0.1
DNS Server.....: 208.67.220.220

C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
```

- b) Test connectivity to the Cisco.com server from the PC

From the command prompt, issue the command **ping Cisco.com**. It may take a few seconds for the ping to return. Four replies should be received as shown in the figure.



```
PC
Physical Config Desktop Programming Attributes
Command Prompt
IP Address.....: 0.0.0.0
Subnet Mask....: 0.0.0.0
Default Gateway.: 0.0.0.0
DNS Server.....: 0.0.0.0

C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>ipconfig /renew
IP Address.....: 192.168.0.101
Subnet Mask....: 255.255.255.0
Default Gateway.: 192.168.0.1
DNS Server.....: 208.67.220.220

C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>ping Cisco.com
Pinging 208.67.220.220 with 32 bytes of data:
Reply from 208.67.220.220: bytes=32 time=1ms TTL=127
Reply from 208.67.220.220: bytes=32 time=10ms TTL=127
Reply from 208.67.220.220: bytes=32 time=2ms TTL=127
Reply from 208.67.220.220: bytes=32 time=1ms TTL=127

Ping statistics for 208.67.220.220:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 10ms, Average = 3ms
C:\>
```