CS23532 -

COMPUTER NETWORKS

Name : D. Y. Monish

Roll No : 230701195

Dept : B.E. CSE

INDEX

Ex.No.1
14/7/25

BASIC NETWORKING COMMANDS

★ **Aim:**

To study various network commands used in Linux and windows.

★ **Windows Networking commands:**

(1) **arp -a**

ARP is short form of Address Resolution Protocol, it will show the IP address of your computer along with the IP address and MAC address of your router.

output:

```
Interface: 172.16.10.114 --- 0xb
  Internet Address      Physical Address      Type
  172.16.10.181         00-27-0e-13-eb-34     dynamic
  172.16.10.84          d8-bb-c1-c5-cb-13     dynamic
  224.0.0.22            01-00-5e-00-00-16     static

Interface: 172.24.176.1 ---0x1b
  Internet Address      Physical Address      Type
  224.0.0.251           01-00-5e-00-00-fb     static
```

(2) **hostname**

This is the simplest of all TCP/IP commands. It simply displays the name of your computer

output:

```
DESKTOP-KMBFN8L
```

(3) **ipconfig /all**

This command displays detailed configuration information about your TCP/IP connection including Router, Gateway, DNS, DHCP and Type of Ethernet adapter in your system.

output:

```
Windows IP Configuration

    Host Name . . . . . . . . . . . : DESKTOP-KMBFN8L
    Primary DNS Suffix . . . . . . . :
    Node Type . . . . . . . . . . . . : Hybrid
    IP Routing Enabled . . . . . . . : No
    WINS Proxy Enabled . . . . . . . : No
...
```

(4) **nbtstat -a**

This command helps solve problems with NetBIOS name resolution. (Nbt stands for NetBIOS over TCP/IP)

output:

```
nbtstat -a
```

Displays protocol statistics and current TCP/IP connections using NBT (NetBIOS over TCP/IP)

```
nbtstat -r

NetBIOS Names Resolution and Registration Statistics
----------------------------------------------------

Resolved By Broadcast     = 0
Resolved By Name Server   = 0

Registered By Broadcast   = 12
Registered By Name Server = 0
```

(5) **net stat**

Network statistics - netstat displays a variety of statistics about a computers active TCP/IP connections. It is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc.

Output:

```
netstat -r
===========================================
Interface List
 11...d8 bb c1 c5 cd 56 ..... Intel(R) Ethernet Connection
                              (11) I219-V
 1........................... Software Loopback
                              Interface 1

 27...00 15 6d 3c 31 25 .... Hyper-V Virtual
                              Ethernet Adapter

===========================================
```

(6) **nslookup**

Name Server Lookup - is a tool used to perform DNS lookups in Linux. It is used to display DNS details, such as IP address of a particular computer, the MX records for a domain or the NS servers of a domain. nslookup can operate in two modes : interactive and non-interactive

Output: nslookup www.google.com

```
DNS request timed out.
    timeout was 2 seconds.
Server : Unknown
Address : fe80::cc0s:faff:fe4b:5264
*** Request to Unknown timed-out.
```

(7) **pathping**

Pathping is unique to Windows and is basically a combination of the ping and Tracert commands. Pathping traces the route to the destination address then launches a 25 second test of each router along the way, gathering statistic on the rate of data loss along each hop.

Example: pathping -n -q 5 -w 100 google.com

output:

Tracing route to google.com [2404:6800:4007:833::200e] over a maximum of 30 hops:

```
0  2401:4900:9271:251:9c2b:adae:519b:4f89
1  2401:4900:9271:251::25
2  2401:4900:1:a869::3
3  *        2401:4900:1:a882::4
4  2401:4900:1:a882::1
5  *        *        *
```

computing statistics for 5 seconds....

(8) **ping** (Packet INternet Grouper)

ping command is the best way to test connectivity between two nodes. Ping use ICMP (Internet Control Message Protocol) to communicate to other devices.

Example: ping -n 2 youtube.com

output:

Pinging youtube.com [2404:6800:4007:810::200e] with 32 bytes of data.

Reply from 2404:6800:4007:810::200e: time=126ms
Reply from 2404:6800:4007:810::200e: time=27ms

Ping statistics for 2404:6800:4007:810::200e:

Packets: sent = 2, Received = 2, Lost = 0 (0% Loss)

Approximate round trip times in milli-seconds:

Minimum = 27ms, Maximum = 126ms, Average = 76ms

(9) **route**

Route command is used to show / manipulate the IP routing table. It is primarily used to setup static routes to specific host or networks via an interface.

Eg. route PRINT *154

```
===========================================================
Interface List
  34...fa 54 f6 b5 00 24 ..... Microsoft Wifi
                               Direct virtual Adapter
  1........................... Software Loopback
                               Interface 1
===========================================================
```

★ Linux Networking Commands:

(1) **ip**

The ip command is one of the basic commands entry administrator will need in daily work from setting up new system and assigning IPs to trouble shooting system.

ip <options> <objects> <commands>

To show the IP address assigned to our interface on your server

Output:

IP address:

Ip address show

100.91.125.130

Ip address del 192.168.1.254/24 dev enp80s3

Ip address add 192.168.1.254/24 dev enp80s3

(2) **ifconfig**

The ifconfig command is a staple in many sepradmins tool belt for configuration and troubleshooting.

(3) **mtr**

Combines ping and traceroute to diagnose network paths and latency.

output: mtr google.com

Host : server	loss %	snt	last	Avg	Best
192.168.0.24	0.0%	10	1.2	1.1	0.9
192.168.0.24	0.0%	10	1.1	1.0	0.9

(4) **tcp dump**

Captures and analyses network packets on the interface.

output:

Capture on interface echo ~ tcp dump - i eth0
Capture on 10 packets : ~ tcp dump - c eth0-c10
tcp dump -i eth0-c10 host 8.8.8.8
Only resource traffic from 8.8.8.8 :
~ tcpdump -i eth0 src host 8.8.8.8

(5) **ping**

Sends ICMP echo requests to test network connectivity.

output: ping google.com
~ ping google.com (216.58.206/174-56/184)
64 bytes from so pe2527 - in-f 14.28100
net (216.58.206.174 icmp.req=1 ttl=5.6
time - 10.7 ms

★ CONFIGURING AN ETHERNET CONNECTION BY USING nmcli:

- **nmcli**

  A command line tool for managing network connections using network manager.

- Procedure:

(i) List Network Profiles:

```
# nmcli connection show
```

(ii) Add a new ethernet connection

```
# nmcli connection add con-name <name>
  ifname <device> type ethernet
```

(iii) Rename connection

```
# nmcli connection modify "wired connection 1"
```

(iv) Show current settings

```
# nmcli connection show "wired connection 1"
```

(V) configure IPV4 settings

```
# nmcli connection modify "wired connection 1"
ipv4 method auto
```

(vi) ping google.com

```
ping google.com (142.250.72.14) 56 (184) bytes
of data 64 bytes from eax175 63-in-f14.1e100
net
```

★ Result:

Here the basic windows and linux commands were executed and the output was verified successfully.

Ex. No. 2

TYPES OF NETWORK CABLES

★ **Aim:**

To study about different types of network cables.

★ **Procedure:**

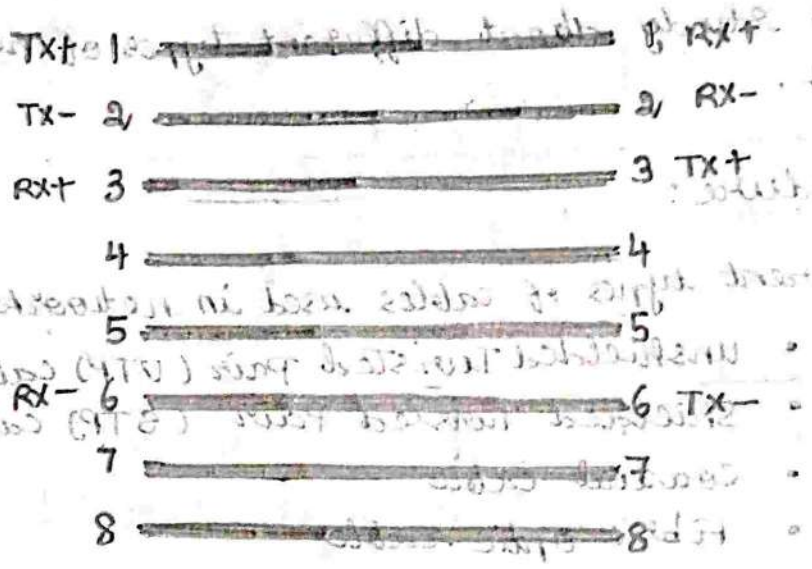
Different types of cables used in networking are:

- Unshielded Twisted Pair (UTP) cable
- Shielded Twisted Pair (STP) cable
- Coaxial cable
- Fibre optic cable

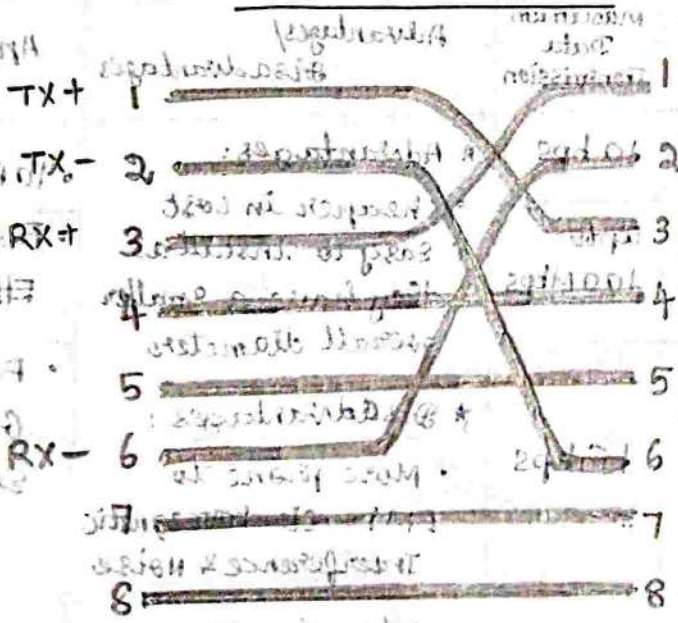
Table:

Cable type	Category	Maximum Data Transmission	Advantages/ Disadvantages	Application/use
UTP	Category 3	10 bps	★ Advantages: • cheaper in cost • easy to install as they have a smaller overall diameter ★ Disadvantages: • More prone to EMI – Electromagnetic Interference & Noise	• 10 Base – T Ethernet
	Category 5	Upto 100 Mbps		• Faster, Gigabit Ethernet
	Category 5e	1 Gbps		• Fast Ethernet Gigabit Ethernet
STP	Category 6, 6a	10 Gbps	★ Advantages: • Shielded, Faster than UTP • Less susceptible to noise and interference ★ Disadvantages: • Expensive • Greater installation effort	Gigabit Ethernet, 10G Ethernet (55m) widely used in data centers
SSTP	Category 7	10 Gbps		Gigabit Ethernet, 10G Ethernet (100m)

A. Straight through Cable



B. Cross-over Cable



Cable type	Category	Maximum Data transmission	Advantages / Disadvantages	Application / Use
Coaxial cable	RG-6 RG-59 RG-11	10-100 Mbps	* Advantages : • High bandwidth • Immune to interference • Low loss bandwidth • Versatile * Disadvantages : • Limited distance • Cost • Size is bulky	• Speed of signal is 500 m • Television network • High speed internet connections
Fibre optics Cable	Single mode Multi mode	100 Gbps	* Advantages : • High speed • High bandwidth • High security • Long distance * Disadvantages • Expensive • Requires skilled installers	Maximum distance of fibre optics cable is around 100 m.

★ STEPS TO MAKE OUR OWN ETHERNET CROSSOVER CABLE / STRAIGHT CABLE :

(i) To start construction of the device, begin by threading shields onto the cable.

(ii) Next, strip approximately 1.5 cm of cable shielding from both ends. The crimping tool has a round area to complete this task.

(iii) After, you'll need to untangle the wires; there should be four 'twisted pairs'. Refering back to the sheet, arrange them from top to bottom. one end should be in arrangement A and the other in B.

(iv) Once the order is correct, bunch them together in a line, and if there are any that stick out further than others, strip them back to create an even level. The difficult aspect is placing these into RJ45 plug without messing up the order. To do, hold the plug with the clip side facing away from you and have the gold pins facing towards you.

(V) Next, push the cable right in the notch at the end of the plug needs to be just over the cable shielding, and if isn't that means that you stripped off too much shielding. simply snip the cables back a little more.

(vi) After the wires are securely sitting inside the plug, insert it into the crimping and push down. It should be shaped correctly, but pushing too hard can crack the fragile plastic plug.

(vii) Lastly, repeat for the other end using diagram B (to make a crossover cables) or using diagram A (to make a straight through cable)

★ Result :

Thus the study about different types of cable network and creating a ethernet cable is done successfully.

Ex.No.3

UNDERSTANDING CISCO PACKET TRACER ENVIRONMENT TO DESIGN SIMPLE NETWORK

★ **Aim:**

To study the Packet tracer tool installation and User Interface overview.

★ **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.

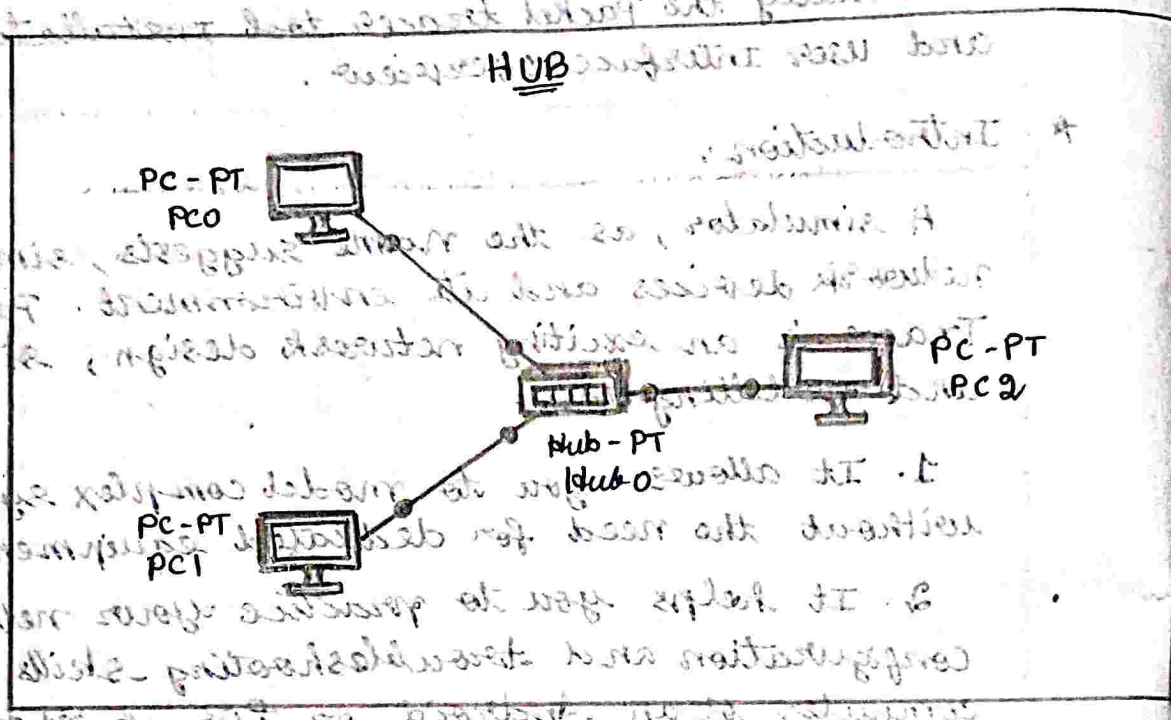
★ **User Interface Overview:**

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.

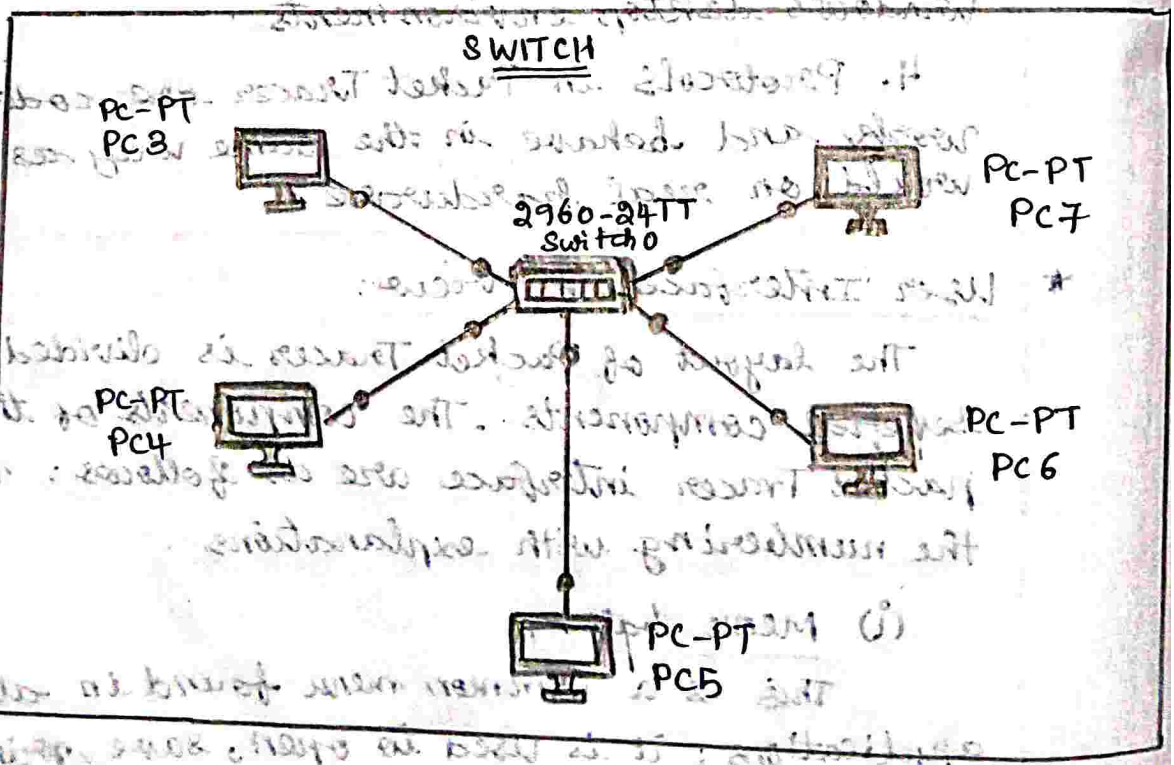
(i) **Menu bar:-**

This is a common menu found in all software applications; it is used to open, save, print, change performance and so on.

* Connection diagram :



Active
Not active.



ii) 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.

iii) Logical/Physical Workspace Tabs :-

These tabs allow you to toggle between the Logical and Physical work areas.

iv) Workspace :-

This is the area where topologies are created and simulations are displayed.

v) Common tools bar :-

This is the area where topologies are created and simulation done. This has select, move layout, place note, delete, inspect, resize shape and add simple/complex PDU.

vi) 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.

vii) 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-specified selection box - When a device category is selected, this selection box displays the different device models within that category.

★ IP configuration :

PC 0 :

IP Configuration	
O DHCP ⊙ Static	
IP Address :	181.084.195.1
Subnet Mask :	255.0.0.0
Default Gateway :	
DNS Server :	

PC1 :

IP Configuration	
O DHCP ⊙ Static	
IP Address :	181.084.195.2
Subnet Mask :	255.0.0.0
Default Gateway :	
DNS Server :	

★ Observed table :

	Source	Destination	Type	Time	Periodic	Num
Successful	PC2	PC0	ICMP	0.000	N	0
Successful	PC2	PC1	ICMP	0.000	N	1
Successful	PC1	PC0	ICMP	0.000	N	2
Successful	PC3	PC5	ICMP	0.000	N	3
Successful	PC3	PC4	ICMP	0.000	N	4

(viii) 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.

★ Procedure:

1) Add components
   → drag 3 PCs → 1 Hub
   → drag 5 PCs → 1 switch
2) Click connect devices
   → use copper straight through cables
   → connect each PCs to Hub and the switch.
3) Assign IPS (for PCs connected to Hub & switch)
   → Go to desktop IP configuration
   → Assign unique IP addresses and subnet masks
4) Send PDU
   → Use PDU (message tool) to send from 1PC to another on the Hub
   → Observe the PDU flow in Real time mode.
5) Repeat PDU test on switch connected PCs.

★ Observation:

(a) A hub forwards packages to all connected devices, while a switch forwards packets only to the intended of recepient based on MAC address.

(b) The network topologies used in most colleges is star topology where all devices are connected to a central switch or routers.

★ Result:

Hence simulation of hub and switch connection and its implementation is done and verified output successfully.

EX.NO.4

SETUP AND CONFIGURE A LAN USING A SWITCH AND ETHERNET CABLES

★ Aim:

To setup and configure a LAN (Local Area Network) using a switch and Ethernet cables in your lab.

★ LAN:

- LAN refers to network that connects devices within a limited area.
- It enables users to share resources, data, printers and internet access.
- LAN connects devices to promote collaboration and transfer information between users, such as computers, printers, servers and switches.
- A LAN switch serves as a primary connecting device within the local network
- Each connected device on a LAN switch can communicate directly with each other, allowing for fast and secure data transfer.

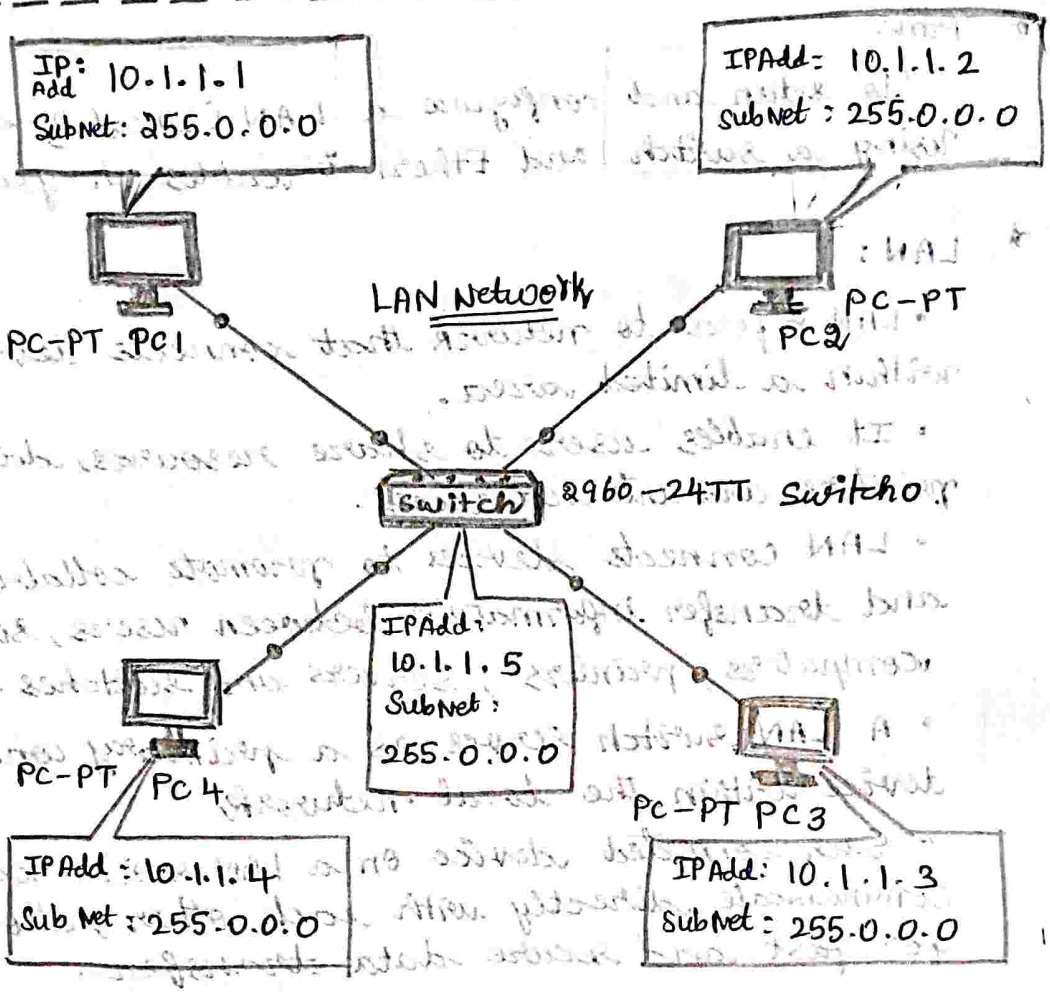
★ How to set up a LAN:

(i) Plan and Design an appropriate network topology taking into account network requirements and equipment location.

(ii) you can take 4 computers, switch which is sufficient for network of the sizes and 4 Ethernet cables.

(iii) connect your computers to network switch via an Ethernet cable, which is as simple as plugging one end of the Ethernet cable into your computer and the other end into your network switch.

✦ Diagram for LAN connection :



- **challenges faced :**
  - → Incorrect cable type
  - → Using wrong IP Address / subNetting
  - → Missing or wrong gateway
  - → Interface or Port Down
  - → Packet Loss
- **Outcomes :**
  - → Successful LAN communication
  - → Proper IP configuration
  - → Crossover cable simulation

(iv) Assign IP address to your Pc.

- Log on to the client computer as Administrator or as Owner
- Click Network and Internet connections.
- Right click local area connection / Ethernet → Go to properties → select internet protocol → click on Properties → Select 'use the following ip address' option and assign ip address

similarly assign IP address to all the PCs connected to switch.

PC1 - IP address : 10.1.1.1 , subnet mask : 255.0.0.0
PC2 - IP address : 10.1.1.2 , subnet mask : 255.0.0.0
PC3 - IP address : 10.1.1.3 , subnet mask : 255.0.0.0
PC4 - IP address : 10.1.1.4 , subnet mask : 255.0.0.0

(v) Configure a network switch

- Connect your computer to switch. connect your computer to the switch using Ethernet cable for web interface.
- Login and open it and enter IP address of switch in address bar. This should bring up the login page for the login page for switch's interface
- configure basic settings for the switch. Assign Ip address as 10.1.1.5 with subnet mask 255.0.0.0.

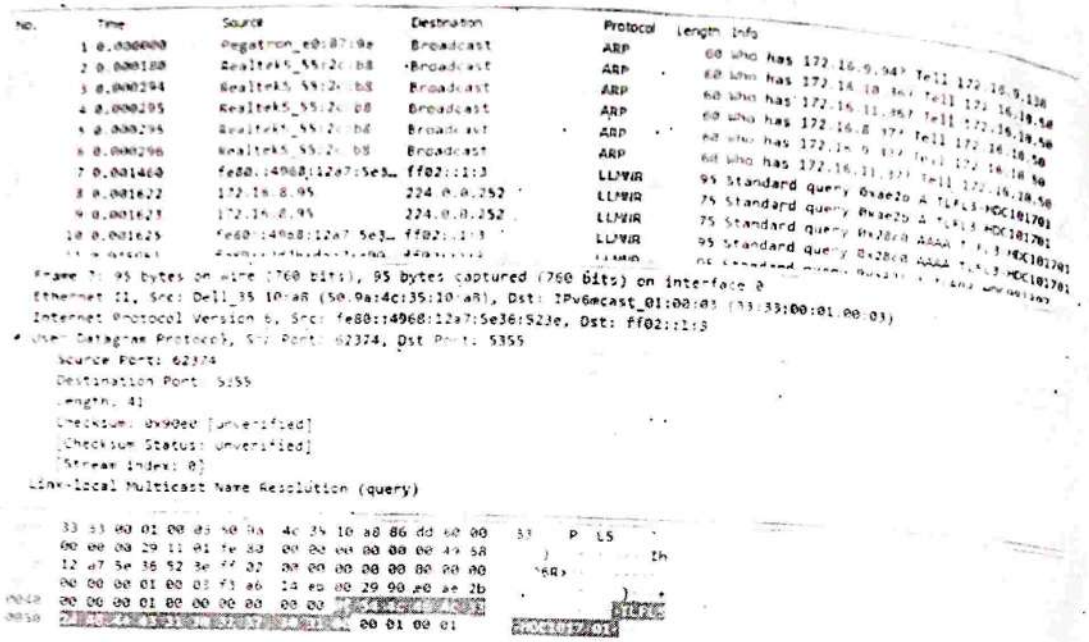
(vi) check connectivity between the switch and other machine by using ping command in the end of the device.

(vii) Select a folder → Go to properties → click Showing Tab → Share it with everyone on the same LAN.

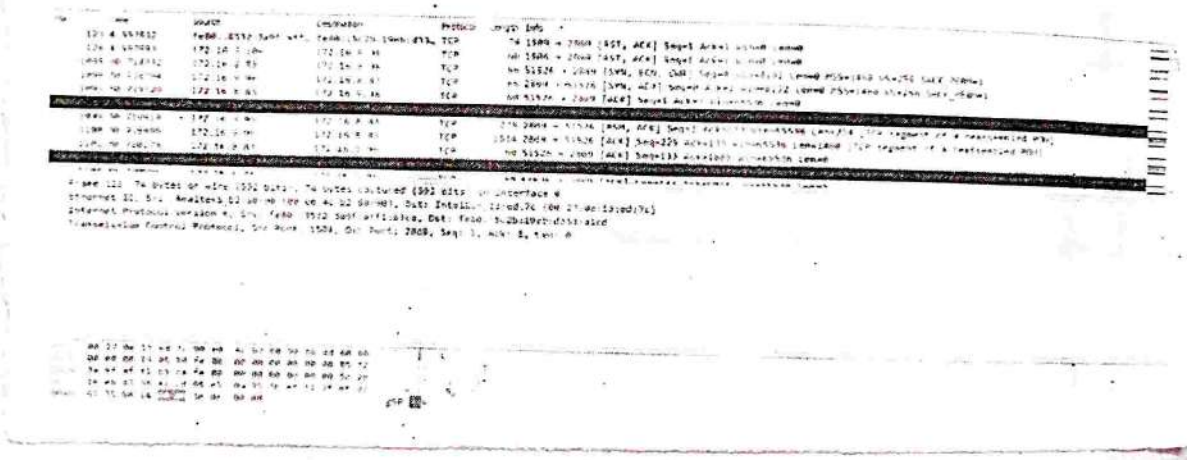
(viii) Try to access the shared folder from other computers of the network.

★ Result :

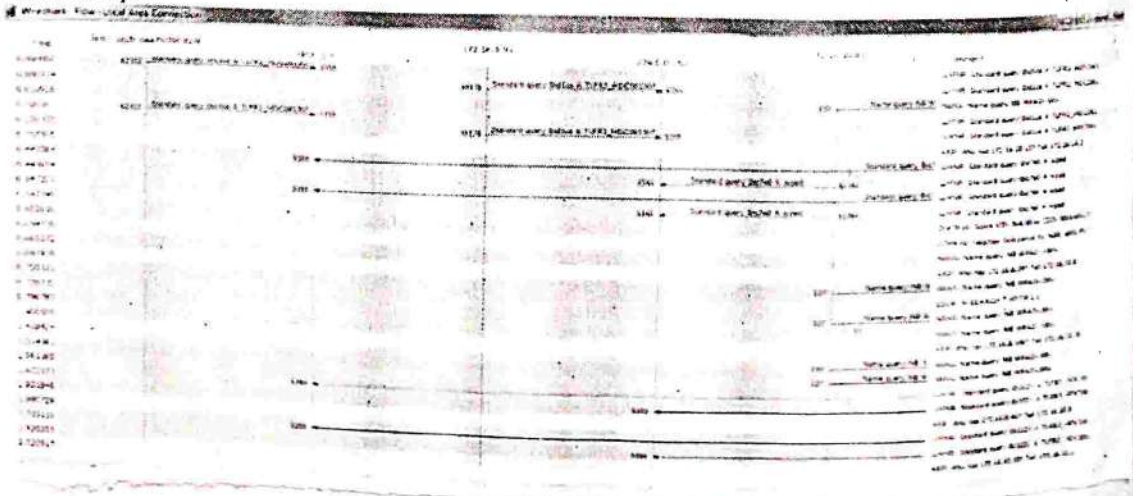
Hence we have configured the LAN connection using switch and ethernet cables successfully.


No. Time Source Destination Protocol Length Info
1 0.000000 Pegatron_e0:87:0e Broadcast ARP 60 Who has 172.16.9.94? Tell 172.16.9.128
2 0.000188 Realtek5_55:2c:b8 Broadcast ARP 60 Who has 172.16.10.56? Tell 172.16.10.50
3 0.000294 Realtek5_55:2c:b8 Broadcast ARP 60 Who has 172.16.11.85? Tell 172.16.10.50
4 0.000295 Realtek5_55:2c:b8 Broadcast ARP 60 Who has 172.16.8.97? Tell 172.16.10.50
5 0.000296 Realtek5_55:2c:b8 Broadcast ARP 60 Who has 172.16.11.97? Tell 172.16.10.50
6 0.000296 Realtek5_55:2c:b8 Broadcast ARP 60 Who has 172.16.11.97? Tell 172.16.10.50
7 0.001460 fe80::4968:12a7:5e3… ff02::1:3 LLMNR 95 Standard query 0xae2b A TLFL3-HDC101701
8 0.001622 172.16.8.95 224.0.0.252 LLMNR 75 Standard query 0xae2b A TLFL3-HDC101701
9 0.001623 172.16.8.95 224.0.0.252 LLMNR 75 Standard query 0x28c8 AAAA TLFL3-HDC101701
10 0.001625 fe80::4968:12a7:5e3… ff02::1:3 LLMNR 95 Standard query 0x28c8 AAAA TLFL3-HDC101701
Frame 7: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface 0
Ethernet II, Src: Dell_35:10:a8 (50:9a:4c:35:10:a8), Dst: IPv6mcast_01:00:03 (33:33:00:01:00:03)
Internet Protocol Version 6, Src: fe80::4968:12a7:5e36:523e, Dst: ff02::1:3
User Datagram Protocol, Src Port: 62374, Dst Port: 5355
Source Port: 62374
Destination Port: 5355
Length: 41
Checksum: 0x90ed [unverified]
[Checksum Status: Unverified]
[Stream index: 0]
Link-local Multicast Name Resolution (query)
33 33 00 01 00 03 50 9a 4c 35 10 a8 86 dd 60 00
00 00 00 29 11 01 fe 80 00 00 00 00 00 00 49 68
12 a7 5e 36 52 3e ff 02 00 00 00 00 00 00 00 00
00 00 00 01 00 03 f3 a6 14 eb 00 29 90 ed ae 2b
00 00 00 01 00 00 00 00 00 00
00 01 00 01

1



Flow Graph



Ex No 6
11/8/25

IMPLEMENTING ENCAPSULATION OF INFO USING WIRESHARK

★ **Aim:**

To experiment on packet capture tool using Wireshark.

★ **Packet Sniffer:**

- Sniffes message being sent / received from / by your computer.
- Store and display the contents of the various protocol fields in the messages.
- Passive program
  - never send packets itself
  - no packets addressed to it
  - receives a copy of all packets

★ **Packet Sniffer Structure Diagnostic Tools:**

- TCPDUMP

  Eg. tcpdump -enx host 10.129.41.2 -w exe3.out

- wireshark

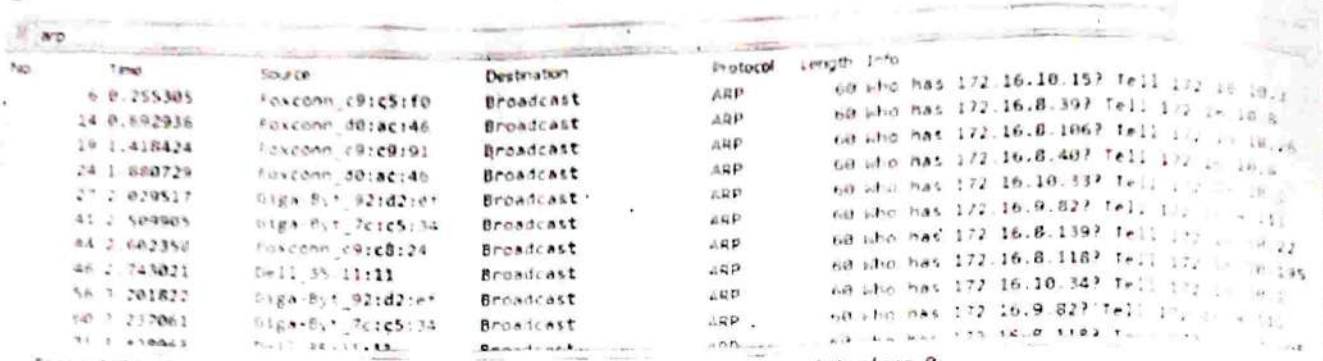
  Eg. wireshark -r exe3.out

★ **Description:**

WIRESHARK:

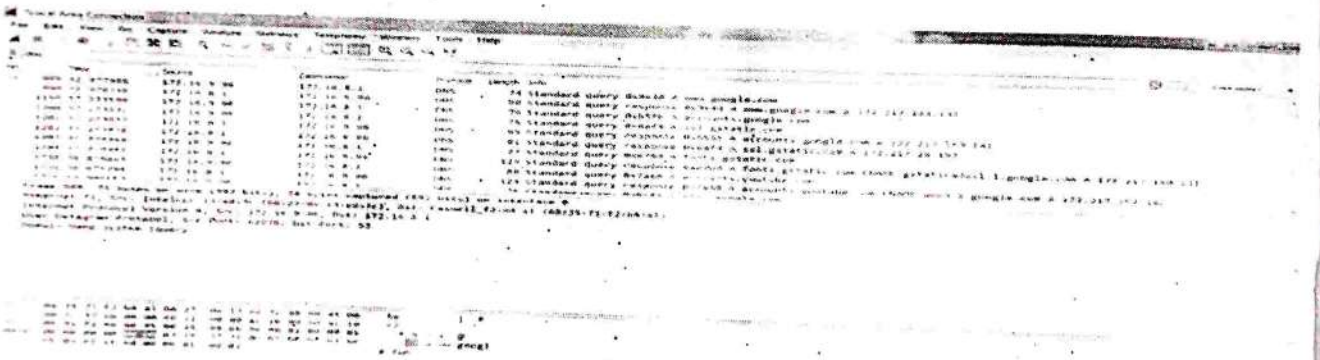
Wireshark, a network analysis tool primarly known as Ethereal, captures packet in real time and display them in human-readable format. Wireshark include filters, color coding and other feature that let you dig deep into network traffic.

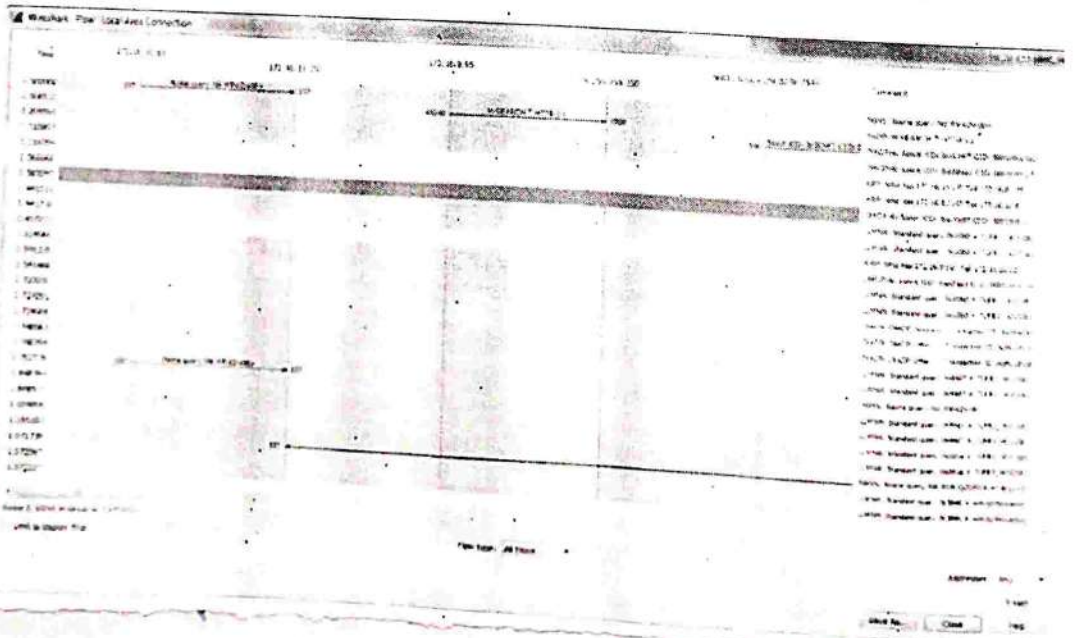
2


arp
No. Time Source Destination Protocol Length Info
6 0.255305 Foxconn_c9:c5:f0 Broadcast ARP 60 Who has 172.16.10.15? Tell
14 0.692936 Foxconn_d0:ac:46 Broadcast ARP 60 Who has 172.16.8.39? Tell
19 1.418424 Foxconn_c9:c9:91 Broadcast ARP 60 Who has 172.16.8.106? Tell
24 1.880729 Foxconn_d0:ac:46 Broadcast ARP 60 Who has 172.16.8.40? Tell
27 2.029517 Giga-Byt_92:d2:e* Broadcast ARP 60 Who has 172.16.10.33? Tell
41 2.509905 Giga-Byt_7c:c5:34 Broadcast ARP 60 Who has 172.16.9.82? Tel
44 2.602350 Foxconn_c9:c8:24 Broadcast ARP 60 Who has 172.16.8.139? Tell
46 2.743021 Dell_35:11:11 Broadcast ARP 60 Who has 172.16.8.118? Tell
56 3.201822 Giga-Byt_92:d2:e* Broadcast ARP 60 Who has 172.16.10.34? Tell
60 3.237061 Giga-Byt_7c:c5:34 Broadcast ARP 60 Who has 172.16.9.82? Tell
Frame 119: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
Ethernet II, Src: IntelCor_13:ed:7c (00:27:0e:13:ed:7c), Dst: Realtek5_b2:60:90 (00:e0:4c:b2:60:90)
Address Resolution Protocol (reply)


00 e0 4c b2 60 90 00 27 0e 13 ed 7c 08 06 00 01
00 00 06 04 00 02 00 27 0e 13 ed 7c
00 e0 4c b2 60 90 ac 10 09 6a

3





★ What we can do with wireshark:

- capture network traffic
- Decode packet protocols using dissectors.
- Defines filters - capture and display.
- Watch smart statistics.
- Analyze problems.
- Interactively browse the traffic.

★ Wireshark used for:

- Network administrators: troubleshoot network problems
- Network security engineers: examine security problems.
- Developers: debug protocol implementations
- People: learn network protocol internals.

★ Capturing packets:

Launch wireshark → select a network interface under Capture to start

Promiscuous Mode: default - captures all network traffic.

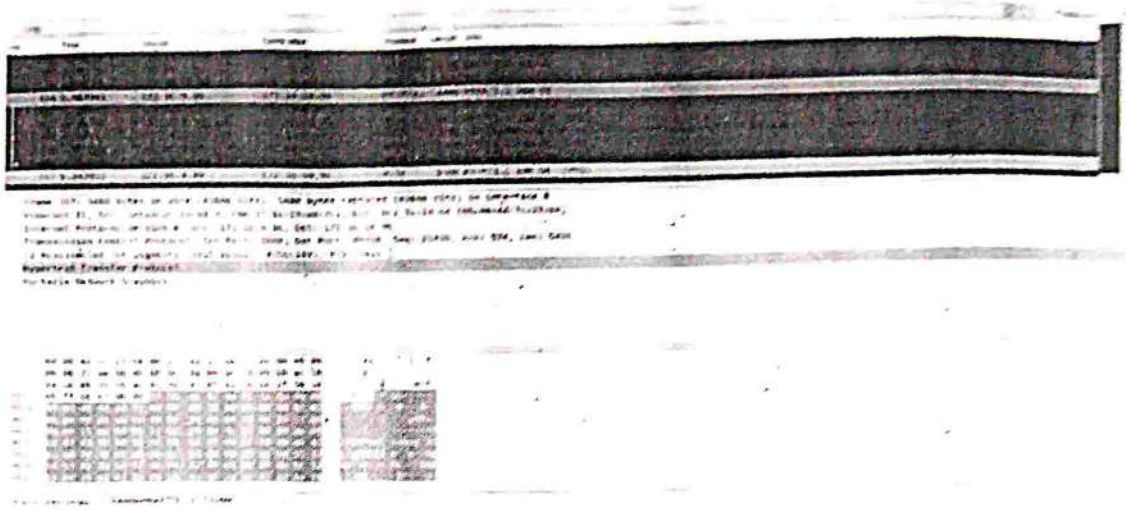
★ wireshark Interface:

- Packet List Pane: Displays all captured packets (each line = 1 packet)
- Packet Details Pane: Show detailed protocol breakdown
- Packet Bytes Pane: Display raw data (hexdump format)

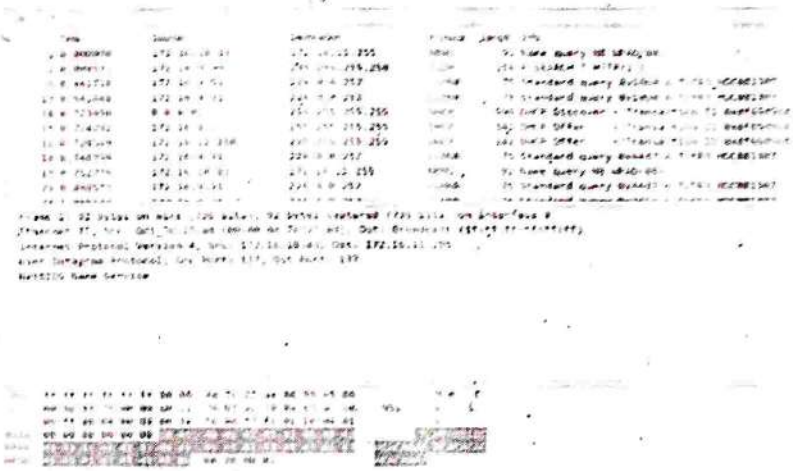
★ Color coding:

Light Purple - TCP, Light Blue - UDP
Black: Packet errors / out of order

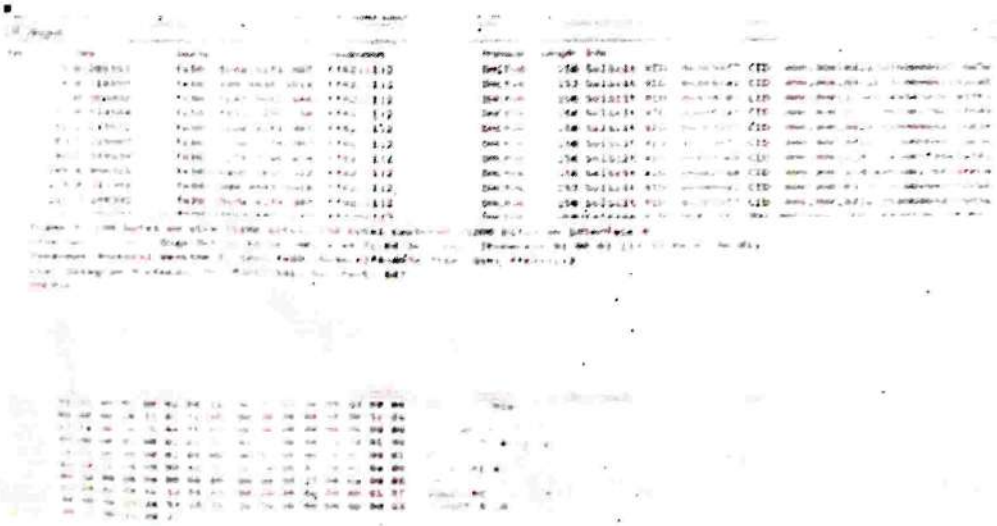
4



5



6



* **Sample captures:**

Load sample captures files from Wireshark Wiki via File > Open. Save your own captures with File > Save.

* **Filtering Packets:**

- Apply filters in the top filter bar.
  Eg. dns → shows only DNS packets.
- Follow Stream: Follow > TCP Stream to view full conversation.

* **Inspecting rackets:**

- Select a packet to explore detailed fields and protocol information.

* **Advanced Features:**

- Flow Graph: Visualizes communication flow between hosts for better understanding.

* **Result:**

Thus the given wireshark tool is used to implement encapsulation is done successfully

Ex. No. 6.

HAMMING CODE

★ Aim:

To write a program to implement error detection and correction using Hamming code concept. Make a test run to input data stream and verify error detection 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:

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

★ Create a receiver program with below features:

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

* Program :

```
def encodeData(data):
    data = list(map(int, data))
    k = len(data)
    r = 0
    while (2**r) < (k+r+1): r += 1
    encoded = [0] * (k+r)
    j = 1
    for i in range(1, k+r+1):
        if i & (i-1) == 0: continue
        encoded[-i] = data[-j]
        j += 1

    for i in range(r):
        pos = 2 ** i
        parity = 0
        for j in range(1, k+r+1):
            if j & pos: parity ^= encoded[-j]
        encoded[-pos] = parity
    return ''.join(map(str, encoded))

def makeError(encoded, pos):
    if pos == 0: return encoded
    l = list(encoded)
    l[-pos] = '1' if l[-pos] == '0' else '0'
    return ''.join(l)
```

```
def correctData(data):
    code = list(map(int, data))
    n = len(data)
    r = 0
    while (2**r) <= n: r += 1
    err_pos = 0

    for i in range(r):
        pos = 2**i
        parity = 0

        for j in range(1, n+1):
            if j & pos: parity ^= code[-j]

        if parity != 0: err_pos += pos

    if err_pos != 0 and err_pos <= n: code[-err_pos] ^= 1
    return ''.join(map(str, code)), err_pos

if __name__ == '__main__':
    # Sender side
    data = input("Enter the data to be sent: ")
    print(f"Actual Data: {data}")
    encoded = encodeData(data)
    print(f"Send Data: {encoded}")

    # Make Error
    err = int(input(f"Choose the bit position to be
            changed (1 to {len(encoded)}): "))

    received = makeError(encoded, err)
```

Output:

Enter the data to be sent : 1111

Actual Data : 1111

Sent Data : 1111111

Choose the bit position to be changed (1 to 7) : 2

Received Data : 1111101

Corrected Data : 1111111

Error at position 2.

```
# Receiver side
print(f"Received Data : {received}")
corrected, pos = correctData(received)
print(f"corrected Data : {corrected}")
print(f"Error at position {pos}" if pos
      else "No Error")
```

* Result:

Thus the program to implement Hamming code is executed successfully.

Ex. No. 7

SLIDING WINDOW PROTOCOL

★ **Aim:**

To simulate the flow of frames from one node to another using sliding window protocol.

★ **Procedure:**

(i) Create a sender program with following features:

- Input window size from the user.
- Input a text message from the user.
- Consider 1 character per frame.
- Create a frame with following fields [Frame no., DATA].
- Send the frames. (Print the output on screen and save it in a file called sender_Buffer.)
- Wait for the acknowledgement from the Receiver.
- Reader a file called Receiver_Buffer.
- check ACK field for Acknowledgement no.,
- If the ACKnowledgement number is as expected, send new set of frames accordingly, else if NACK is received, resend the frames accordingly

ii) Create a receiver file with following features:

- Reader a file called Sender_Buffer.
- check the frame no.
- If the frame no. are as expected, write the appropriate ACK no. in the Receiver_Buffer file, else write NACK no. in the Receiver_Buffer file.

★ code :

sender.py

```
import time, os

SENDER_FILE = "Sender_Buffer.txt"
RECEIVER_FILE = "Receiver_Buffer.txt"

def write_sender(frames):
    with open(SENDER_FILE, "w") as f:
        for fno, ch in frames:
            f.write(f"{fno}:{ch}\n")

def read_receiver():
    if not os.path.exists(RECEIVER_FILE):
        return None
    with open(RECEIVER_FILE) as f:
        return f.read().strip()

def main():
    win = int(input("Enter window size: "))
    msg = input("Enter message: ")
    frames = [(i, ch) for i, ch in enumerate(msg)]
    base = 0

    while base < len(frames):
        window = frames[base : base+win]
        print("Sending:", window)
        write_sender(window)

        time.sleep(1)
        ack = read_receiver()
        print("Got:", ack)

        if ack and ack.startswith("ACK"):
            base = int(ack.split()[1])
        elif ack and ack.startswith("NACK"):
            nack_no = int(ack.split()[1])
            base = nack_no
```

```
        else:
            print("NO ACK, resending ...")
    print("All frames sent successfully!")

if __name__ = "__main__"
    main().
```

★ receiver.py

```
import time, os
SENDER_FILE = "sender_buffer.txt"
RECEIVER_FILE = "receiver_buffer.txt"
def read_sender():
    if not os.path.exists(SENDER_FILE):
        return []
    with open(SENDER_FILE) as f:
        lines = f.read().strip().splitlines()
        return [tuple(line.split(":")) for
                line in lines if line]
def main():
    expected = 0
    while True:
        frames = read_sender()
        if not frames:
            time.sleep(1)
            continue
        print("Received:", frames)
        fno, data = frames[0]
        fno = int(fno)
        if fno = expected:
            print("Accepted:", data)
            expected += len(frames)
            write_receiver(f"ACK {expected}")
            time.sleep(1)
```

* Output : sender console

Enter window size : 3
Enter message : HELLO
Sending : [(0, 'H'), (1, 'E'), (2, 'L')]
Got : ACK 3
Sending : [(3, 'L'), (4, 'O')]
Got : ACK 5
All frames sent successfully!

Receiver console

Received : [(0, 'H'), (1, 'E'), (2, 'L')]
Accepted : H
Accepted : E
Accepted : L

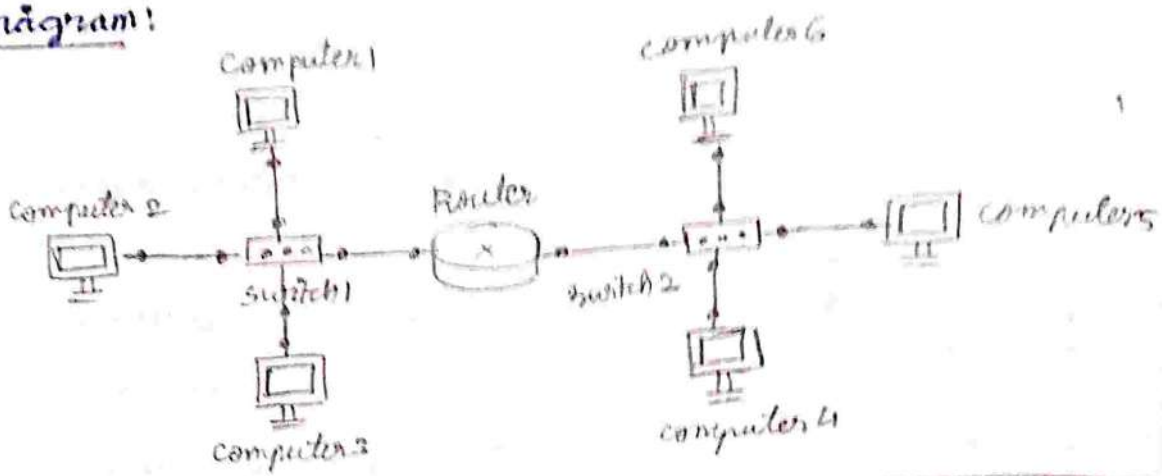
Received : [(3, 'L'), (4, 'O')]
Accepted : L
Accepted : O

```
if __name__ = "__main__":
    main()
```

Result:

Thus the given program to implement sliding window is executed successfully.

* Diagram:



1. Send a packet from computer 1 to computer 1 (broadcast)
Packet type: "ARP Request".
Data: computer 6.

i) How many devices can see the ARP request? 4
ii) Did computer 6 receive ARP request? NO

2. Send a packet from computer 4 to computer 4 (broadcast)
Packet type: "ARP request".
Data: computer 6

i) How many devices can see the ARP request? 4
ii) Did computer 6 receive ARP request? Yes

3. What is the first IP address Nmap would scan if you provide 10.10.12.13/29 as your target? 10.10.12.8

4. How many IP addresses will NMap scan if you provide the following range 10.10.0-255.101-125?

6400 ⇒ 256 × 25

5. Send a packet from computer 1 to computer 3 of packet type 'ping request'.

i) What is the packet type computer 1 sent before the ping?
ARP request

ii) What is the type of packet that computer 1 received before being able to send the ping?
ARP response

Ex. No. 8
1/9/25

NMAP – TRY HACK ME

★ **Aim:**

This experiment outlines the processes that Nmap takes before port-scanning to find which system are online.

★ **objective:**

This stage is crucial since attempting to port scan offline systems will merely waste time and create unneeded network noise.

The following is the information that will be covered in an attempt to discover live hosts:

1) **ARP scan:**

This scan uses ARP requests to discover live hosts.

2) **ICMP scan:**

This scan uses ICMP requests to identify live hosts.

3) **TCP/UDP ping scan:**

This scan sends packets to TCP ports and UDP ports to determine live hosts.

There will be two scanners introduced:

- arp-scan
- masscan

iii) How many computers responded to the ping request? 1

6. Send a packet from computer 2 to computer 5 - ping request
   i) What is the name of the 1st device that responded to the first ARP request? Router
   ii) What is the name of the 1st device that responded to the second ARP request? computer 5
   iii) Send another Ping Request. Did it require new ARP request? NO

7. How many devices are you able to discover using ARP request when broadcast from computer 1? 3

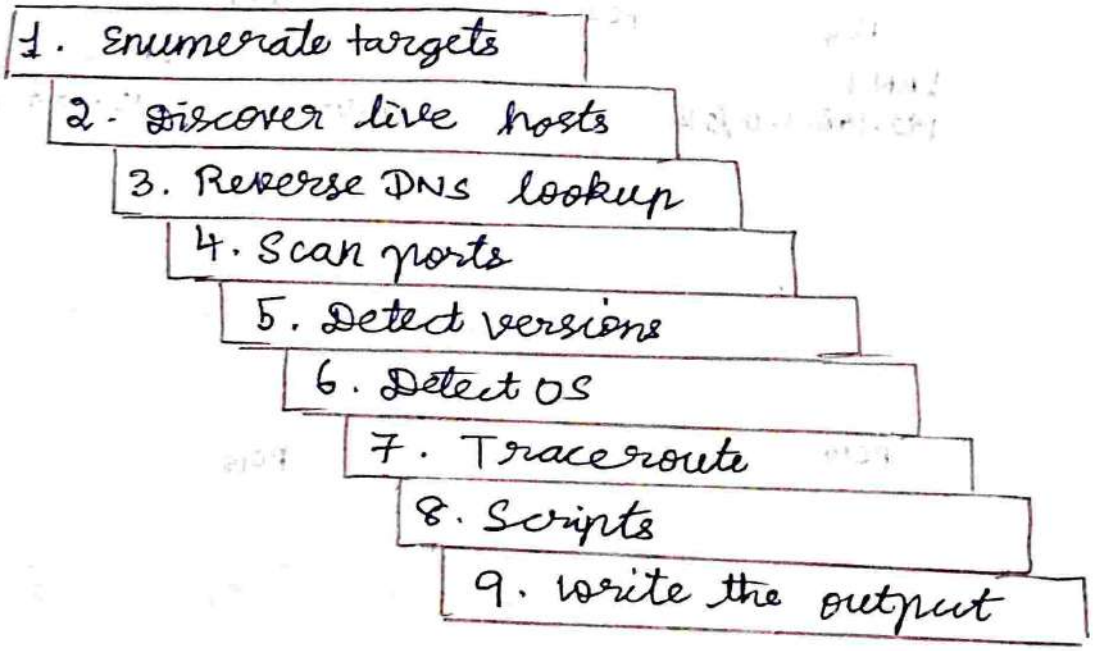
8. What is the option required to tell Nmap to use ICMP timestamp to discover live hosts? -PP
9. What is the option required to tell Nmap to use ICMP address mask to discover live hosts? -PM
10. What is the option required to tell Nmap to use ICMP echo to discover live hosts? -PE

11. Which TCP ping scan doesnot require a privileged account? TCP SYN Ping
12. Which TCP ping scan requires a priviledged account? TCP ACK Ping
13. What option do you need to add to Nmap to run a TCP SYN ping scan on the telnet port? -PS23

14. We want Nmap to issue Reverse DNS lookup for all possibles hosts on a subnet, hoping to get some insights from the names. What option should we add? -R

Nmap is a well known tool for mapping networks, locating live hosts and detecting running services. Its scripting engine can be used to extend its capabilities such as fingerprinting services and exploiting flaws.

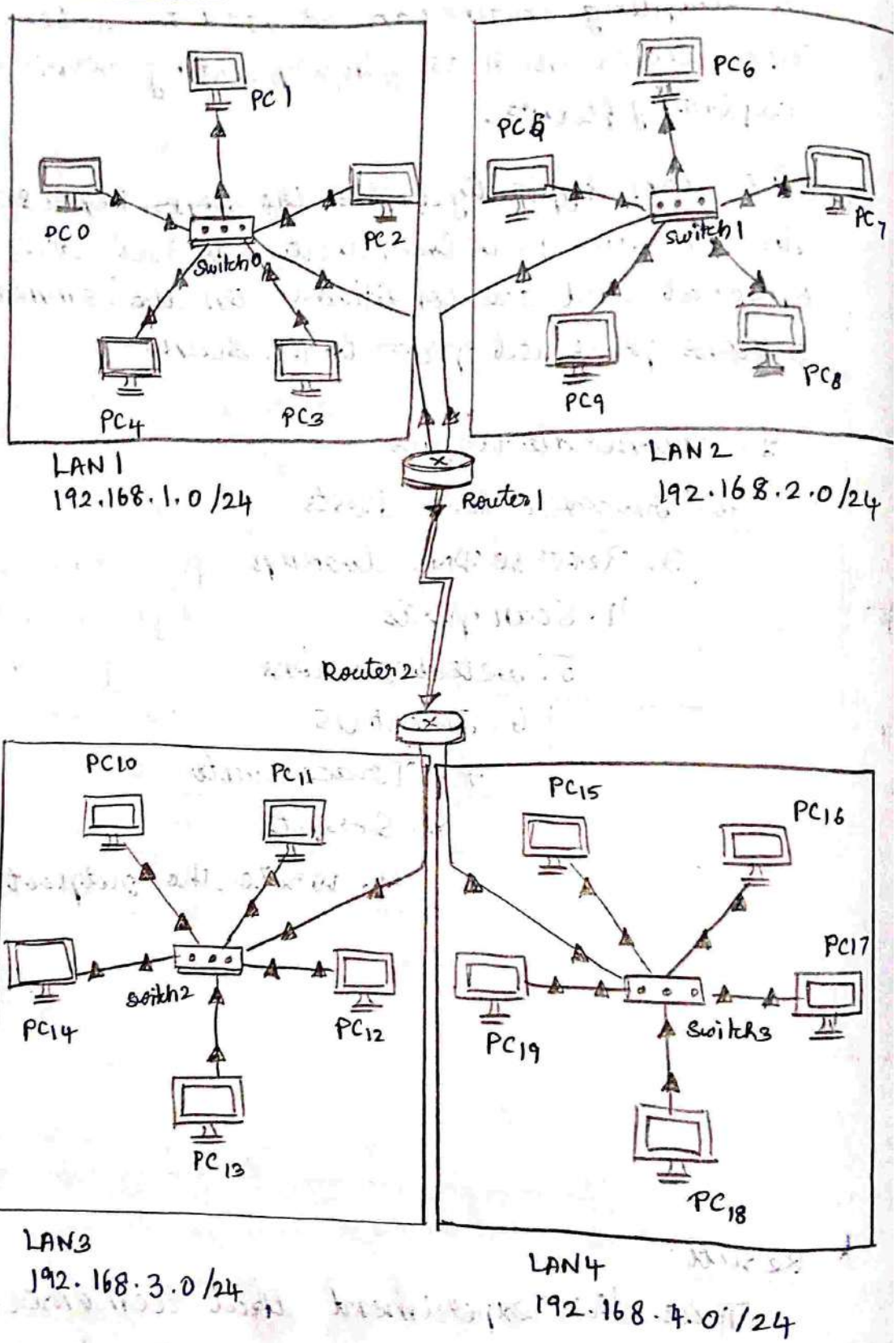
The scan typically follow the steps represented in the image below, but several are optional and are conditional on the 'command-line' options provided prior to the scan.



★ Result:

Thus the experiment that demonstrates the process of Nmap for port scanning is executed successfully.

* circuit diagram:



EX.NO.9
5/9/25

Implementation of subnetting in CISCO PACKET TRACER

★ **Aim:**

To implement subnetting in CISCO packet tracer simulator.

★ **Overview:**

classless IP subnetting (CIDR) allows dividing a network into smaller subnets using custom subnet masks, improving IP address utilization. In this simulation, subnetting is applied to create efficient, multinetwork connectivity using routers, switches and PCs.

★ **Procedure:**

1) **Create Network Topology:**
- open CISCO packet Tracer → New → Network → Generic
- Add devices: Routers (R_1, R_2), Switches (S_1, S_2), PCs
- Connect them using straight through cables.

2) **Subnetting:**
- Given Network: 192.168.1.0/24
- Required atleast 5 hosts per subnet so use /27 mask (255.255.255.224) → 8 subnets, 30 hosts each.

3) **IP Addressing:**
- **Routers R1:**
  - R_1 – Gig0/0 → 192.168.1.1/27
  - R_1 – Gig0/1 → 192.168.2.1/27
- **Router R_2:**
  - R_2 – Fast0/0 – 192.168.3.1/27
  - R_2 – Fast0/1 – 192.168.4.1/27
- **Switch S_1:**
  connected PCs:
  192.168.1.11–15 (/27)
  192.168.2.11–15 (/27)
- **Switch S_2:**
  connected PCs:
  192.168.3.11–15 (/27)
  192.168.4.11–15 (/27)

★ Observation:

Fire	Last Status	Source	Destination	Type	color	Time	Periodic	Num
*	Successful	PC1	PC5	ICMP		0.0	N	1
*	Successful	PC6	PC10	ICMP		0.0	N	2
*	Successful	PC11	PC0	ICMP		0.0	N	3

We can see the output in the Realtime/Simulation view.

4) Configuration commands :

Configure the terminal for the routers, switches and PCs clearly denoting its IP and subnet mask with default gateway.

5) Testing :

- Use ping command between PCs and routers successful ping = Proper connectivity and subnetting implementation.

* Result :

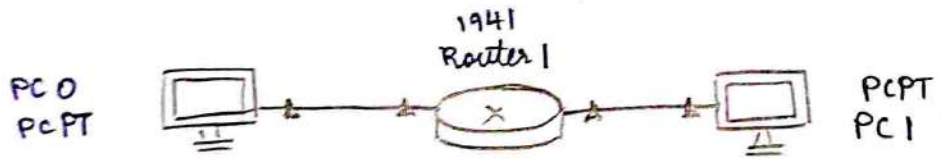
Thus the given implementation of subnetting mask is executed successfully.

(a) ⊕ Router Configuration Table:

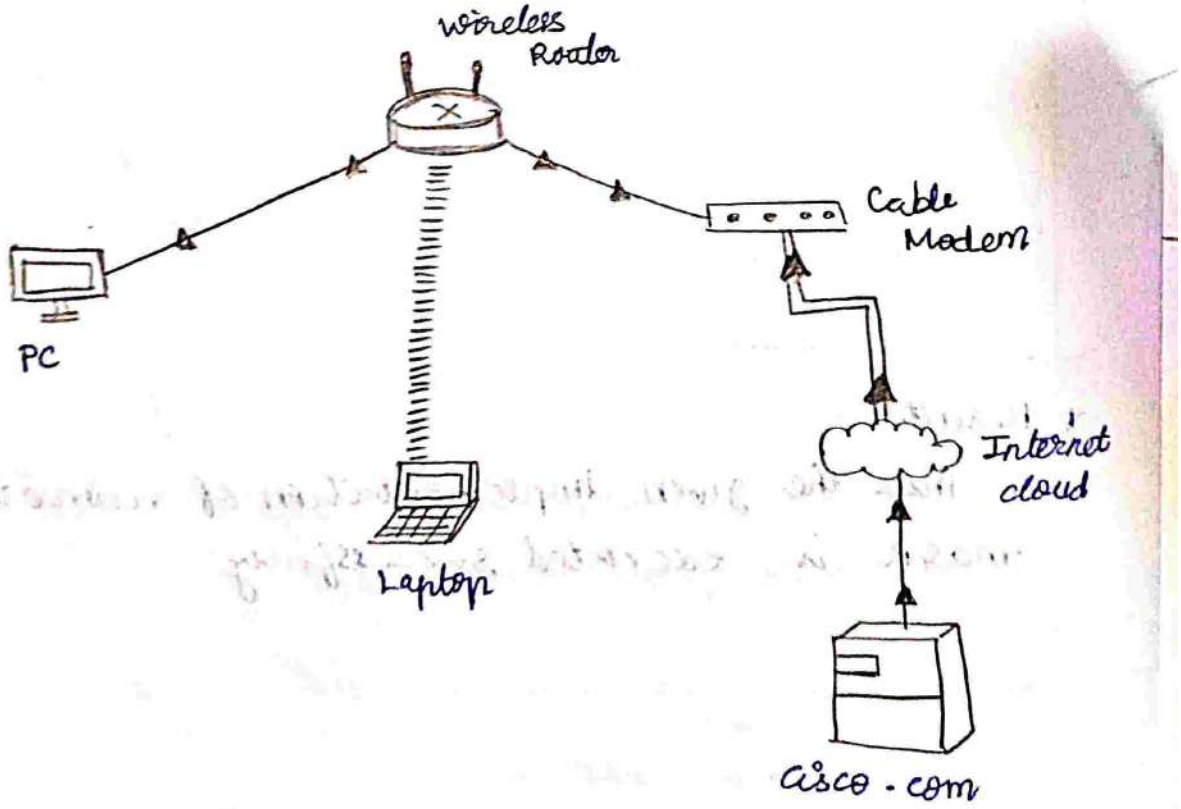
Device Name	IP address Fast Ethernet 0/0	Subnet mask	IP address Fast Ethernet 0/1	Subnet Mask
Router1	192.168.10.1	255.255.255.0	192.168.20.1	255.255.255.0

⊕ PC configuration Table:

Device Name	IP Address	Subnet Mask	Gateway.
PC0	192.168.10.2	255.255.255.0	192.168.10.1
PC1	192.168.20.2	255.255.255.0	192.168.20.1



b) Diagram:



Ex.No.10
12/9/25

INTERNETWORKING WITH ROUTERS

* **Aim:**

To perform internetworking with routers in CISCO packet tracer simulator.

(a) **Design and Configure Internetwork using Routers:**

* (Configure router1):
  - Select the router > Open CLI
  - Press ENTER to start configuring Router1
  - Type enable to activate priviledged mode, Router 1 CLI.

* (Configuring PCs):
  - Assign IP address to every PC.
  - Select PC, go to desktop > IP configuration, assign IP, gateway & subnet mask.
  - PC0 = 192.168.10.1 — default gateway.

* (connecting PCs with Router):
  - Connect FastEthernet0 port of PC0 with FastEthernet0/0 port of Router1 using copper-straight-through cable.
  - Router configuration is done.

(b) **Design and configure internetwork using wireless Router DHCP server and internet cloud.**

* objectives:
  - Build simple network in logical topology workspace.
  - configure Network devices
  - Test connectivity between them.
  - Save files and close packet tracer.

★ IP configuration: · Fast Ethernet0 .

⊙ DHCP ○ Static

IP Address : 192.168. 0. 1

Subnet Mask : 255.255.255. 0

Default Gateway : 192.168.0.1

DNS Server : 208.67.220.220.

★ configuration :

Cable:

Cable

Coaxial 7 ✓ ⟷ Ethernet 6 ✓

Port Port

From Port	To Port	
Coaxial 7	Ethernet 6	

Add Remove

(i) Part 1:

- Launch Packet Tracer and Build Topology

(ii) Part 2:

- configure wireless Router, Laptop then the PC.
- configure Internet cloud
- configure Cisco.com server.

(iii) Part 3:

- Refresh IPV4 settings on PC.

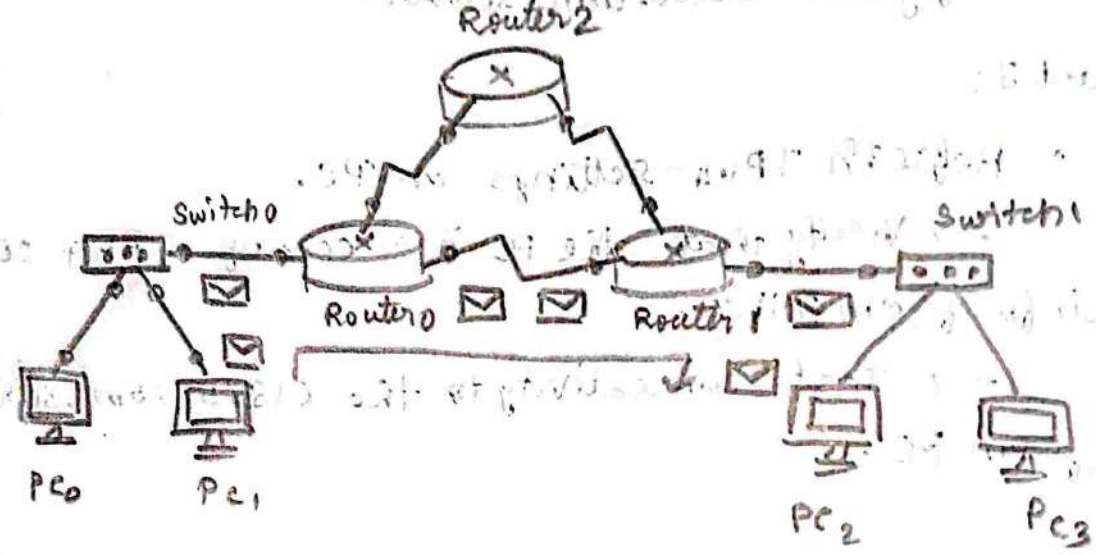
→ Verify that the PC is receiving IPV4 config. info from DHCP.

→ Test connectivity to the CISCO.com server from PC.

* Result:

Thus the above experiment to implement internetworkin is executed successfully

* Diagram :



Ex.No-11

SIMULATE STATIC ROUTING CONFIGURATION USING CISCO PACKET TRACER AND RIP

* **Aim:**

To simulate the RIP using CISCO packet tracer.

* **Assign IP addresses to PCs:**

Double click PCs and click Desktop and click IP configure. Assign IP address referring the table.

* **Assign IP address to interface of routers:**

Double click Router0 and click CLI and press enter key to access the command prompt of router0.

router > enable
router# config terminal
Enter config commands, one per line

enter with CNTL/Z
return (config) #

interface fast ethernet 0/0 command is used to enter in interface mode

ip address 10.0.0.1 255.0.0.0 command will assign IP address to interface

no shutdown command will bring the interface up exit command is used to return in global config mode. Every serial cable has two ends DTE and DCE.

router > enable
router# configure terminal
enter configuration commands, one per line
end with CNTL/Z

router (config) # interface serial 0/0/0
router (config-if) # ip address 192.168.1.250 255.255.255.252

router (config-if) # no shutdown

router (config-if) # exit
router (config) # interface serial 0/0/1
router (config-if) # ip address 192.168.1.240
255.255.255.252

router (config-if) # clock rate 6400
router (config-if) # bandwidth 64
router (config-if) # no shutdown
router (config-if) # exit.

By default RIP will use the route that has low-hops counts between source and destination. In our network, route 1 has low hops so it will be selected. we can use tracert command to verify it.

* Result:

Hence the experiment on simulating RIP config using CISCO packet tracer has been executed successfully

EX.NO.12 (a)

IMPLEMENTATION OF ECHO CLIENT SERVER USING TCP/UDP SOCKETS

★ **Aim:**

To implement an echo client server by using TCP/UDP sockets.

★ **Server Side Algorithm:**

```
import socket
server_socket = socket.socket(socket.AF_INET,
                              socket.SOCK_STREAM)
server_socket.bind(('localhost', 12345))
server_socket.listen(1)
print("server is waiting for connection")
conn, addr = server_socket.accept()
print(f"connected to {addr}")
while True:
    data = conn.recv(1024).decode()
    if not data or data.lower() == 'bye':
        print("connection closed")
        break
    print(f"Received from client: {data}")
    conn.send(data.encode());
conn.close()
```

★ **client side algorithm:**

```
import socket
client_socket= socket.socket(socket.AF_INET,
                             socket.SOCK_STREAM)
client_socket.connect(('localhost', 12345))
while True:
    message = input("Enter message: ")
    client_socket.send(message.encode())
    if message.lower() == 'bye':
        break
```

output screen:

* client side:

Enter message: Hello world

Echo from server: Hello world

Enter message: How are you?

Echo from server: Hello are you?

Enter message: Bye.

* Server side:

server is waiting for connection

Connected to ('127.0.0.1', 58944)

Received from client: Hello world

Received from client: How are you?

connection closed.

```
data = client_socket.recv(1024).decode()
print(f"Echo from server: {data}")
client_socket.close()
```

Result:

Hence the experimentation implementing echo on a client server using the TCP/UDP socket is done successfully.

* output screen:

- Server side

sever waiting for connection
Connected to ('127.0.0.1', 59010)
client: Hi server!
You : Hello client!
Client : How are you?
You : I'm fine, thanks!
Client : Bye!
client disconnected.

- Client side :

You : Hi server!
Server : Hello client
You : How are you?
Server : I'm fine, thakes!
You : Bye!

Ex.No.12 (b)

IMPLEMENTATION OF CHAT CLIENT AND SERVER USING TCP/UDP SOCKETS

★ **Aim:**

To implement chat client server using TCP / UDP sockets

★ **Server side algorithm:**

```
import socket
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('localhost', 12346))
server.listen(1)
print("server is waiting for connection...")
conn, addr = server.accept()
print(f"connected to {addr}")
while True:
    msg = conn.recv(1024).decode()
    if msg.lower() == 'bye':
        print("client Disconnected")
        break
    print(f" client: {msg}")
    reply = input("you : ")
    conn.send(reply.encode())
    if reply.lower() == 'bye': break
conn.close()
```

★ **client side algorithm:**

```
import socket
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('localhost', 12346))
while True:
    message = input("you: ")
    client.send(message.encode())
    if message.lower() == 'bye': break
    reply = client.recv(1024).decode()
    print(f"server: {reply}")
    if reply.lower() == "bye": break
client.close().
```

★ **Result:**

Hence we implemented client server chat successfully.

Ex No. 13

★ output screen:

- server side:

command: python server.py

output:

UDP server running on 127.0.0.1:12345
received from (127.0.0.1 : 55437) - ping

- client side:

Sample I/p: Enter host: 127.0.0.1
Sample O/p: Reply from 127.0.0.1 : ping

Ex No. 13

IMPLEMENTATION OF PING PROGRAM

★ **Aim:**

To implement a ping program by self.

★ **Client side Algorithm:**

```
import socket
import time
def ping_server(host = '127.0.0.1', port = 12345):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s:
        try:
            s.sendto(b'Hello', (host, port))
        except s.timeout:
            print("Request timed out")

ping server()
```

★ **Server side algorithm:**

```
import socket
def start_server(host ='127.0.0.1', port = 12345):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s:
        s.bind((host, port))
        print(f"UDP server running on {host}:{port}")
    while True:
        data, addr = s.recofrom(1024)
        print(f"Received message from {addr}:{data.decode()}")

start_server()
```

★ **Result:**

Hence the experiment on implementing the ping program by self has been executed successfully.

Ex.No.14

Sample I/P:

- Open a WebBrowser and visit

  `https://www.google.com`

- Run the command in another terminal

  `8.8.8.8 ping`

Sample O/P:

Protocol : TCP
Source IP : 192.168.1.5
Destination IP : 142.250.183.110

Ex.No.14

IMPLEMENTATION OF PACKET SNIFFING USING RAW SOCKETS

★ **Aim:**

To implement packet sniffing using RAW sockets.

★ **Algorithm:**

```
from scapy.all import sniff
from scapy.layers.inet import IP, TCP, UDP, ICMP
def packet_callback(packet):
    if IP in packet:
        ip_layer = packet[IP]
        protocol = ip_layer.proto
        src_ip = ip_layer.src
        dst_ip = ip_layer.dst
        if protocol == 1:
            protocol_name = "ICMP"
        elif protocol == 6:
            protocol_name = "TCP"
        elif protocol == 17:
            protocol_name = "UDP"
        else:
            protocol_name = "Unknown Protocol"
        print(f"Protocol: {protocol_name}")
        print(f"Source IP: {src_ip}")
        print(f"DestinationIP: {dst_ip}")
        print("-" * 50)
sniff(iface="Wi-Fi", prn=packet_callback,
      filter="ip", store=0)
```

★ **Result:**

Hence the experiment on implementing packet sniffing using sockets has been executed successfully.