INDEX

Name : Mathan kumar. M Subject : CN-LAB

Std : Div. Roll No : 230701181

School / College :

S.No.	Date	Title	Marks	Teacher's Sign / Remarks
1.		Basic Networking commands		
2.		Types of Network cables		
3.		CISCO PACKET TRACER ENVIRONMENT		
		To design simple network.		
4.		LAN using switch & Ethernet cables		
5.		Encapsulation of INFO using wireshark		
6.		Hamming code		
7.		sliding window protocol		
8.		NMAP - TRY HACK ME		
9.		Subnetting CISCO PACKET TRACER		
10.		Internetworking with routers		
11.		static routing config using		
		CISCO packet tracer & RIP		
12.		a) ECHO client/server using TCP/UDP sock		
		b) CHAT client/server using TCP/UDP sock		
13.		Ping program		
14		PACKET sniffing using RAW Sockets		

Ex NO : 1

14/07/2025

Basic Networking Commands

Aim :

Study of various Network commands used in Window

Window networking commands :

i) 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 : 172.16.10.115 --- 0x9

Internet Address	physical address	Type
172.16.8.1	7C-5a-1C-cf-be-45	dynamic
172.16.8.195	7C-57-58-38-e6-9e	dynamic
172.16.8.208	40-C2-ba-7C-a2-12	dynamic
172.16.11.255	ff-ff-ff-ff-ff-ff	static
224.0.0.22	01-00-5e-00-00-16	static.

ii) hostname

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

output :

LUFFY

iii) 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

Hostname . . . . . : LUFFY
primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled . . . . . : No
WINS Proxy Enabled . . . . . : No
. . .

iv) nbtstat -a :

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

output :

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

nbtstat -r

NetBIOS Names Resolution and Registration statist

Resolved By Broadcast = 0
Resolved By Name server = 0
Registered By Broadcast = 3
Registered By Name server = 0

v) netstat :

(network statistics) netstat displays a variety of statistics about a computer 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

9 ... d8 bb c1 c5 cd as ... Intel(R) Ethernet connection (11) I219-V

1 ..... Software Loopback Interface 1

vi) 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.

Servers : Unknown.

Address : FE80 : CC08 : 1aff : fe4b : 5264.

*** Request to unknown timed-out.

vii) 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 seconds test of each router along the way, gathering statistic on the rate of data loss along each hop.

output:

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

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

0 2401 : 4900 : 9271 : 251 : 9c2b : adae : 579b : 4889

1 2401 : 4900 : 9271 : 251 : : 25

2 2401 : 4900 : 1 : a809 : : 3

3 * 2401 : 4900 : 1 : a882 : : 4

4 2401 : 4900 : 1 : a882 : : 1

5 * * *

computing statistics for 5 seconds..

viii) 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.

output :

ping -n 2 youtube.com

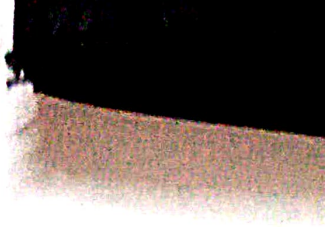
pinging youtube.com [2404 : 6800 : 4007 : 810 : : 200e] with 32 bytes of data.

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

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

ix) Router.

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 interfaces.



output :

route PRINT -4

Interface List

34 ...9a 51 f6 65 00 24 ... Microsoft

Wifi Direct virtual Adapter

1 ..... Software loopback

Interface 1

Linux Networking commands.

1) ip : serving up new system and assigning ip to troubleshooting existing systems.

ip <options> <object> <command>

a) ip address show:

output : 1. lo : <LOOPBACK> UP, LOWER-UP>

2. enps 3s6 : < NO CARRIER, BROADCAST, MULTICAST, UP>

3. WIP2s0 : <BROADCAST, MULTICAST, UP, LOWER.UP>

b) Add an IP address:

ip address add 192.168.1254/24 dev enps3s6.

RTNETDWK answer : File exists.

c) To delete an IP :

ip address del 192.1681.254/24 dev enps3s6

d) Alter the status of the interface. [online]

ip link set enp33s6 up.

e) Alter the status of the interface [offline]

ip link set enps3s6 down.

f) Alter the status by enabling promiscous mode

ip link set enps3s6 promisc on.

g) Add a default route.

ip route add default via 192.168.1254 dev enps3s6.

h) Add a route via gateway

ip route add 192.168.1.0124 via 192.168.1.254.

i) Add a route that can be reached on device

ip route add 192.168.1.0/24 dev enps3s6.

j) Delete the route.

ip route ~~delete~~ 192.168.1.0/24 via 192.168.1.2

k) Display the route.

ip route get 10.10.1.4.

2) Ifconfig:

It is a staple in many sysdminx tool belt for configuring and trouble shooting networks.

output:

enps8ifs: flags: 4099 <UP, BROADCAST, MOALKAST,>
mtr 1500

ether 80:88:10:86:8d:da txqueuelen 1000 (Ethernet).

RX Packets 0 bytes 0 (0.0.B)

RX errors 0 dropped 0 overun 0 frame 0

Tx packets 0 bytes 0 (0.0.B)

Tx errors 0 dropped 0 overun 0 carrier 0
collision 0

device interrupt 19 memory 0x70600000 -> 06200

Lo: flag =73 <up, LOOPBACK, RUNNING> MTU 65536.

inet 127.00.1 netmask 255.000

inet ::1 prefixlen 128. scopeid 0x10<host>

wdp2s0: flags = 4163 <UP. BROADCAST, RUNNING,
MULTICAST > MTU 1500

inet: 172.16.75.86 Network 255.255.248.0

broadcast 172.16.79.255.

3) mtr: (Matt's traceroute) is a program with a command line interface that serve as a network diagnostic and trouble shooting tool.

Syntax: mtr <options> hostname IP.

a) The basic mtr command shows [redacted] statistics including each hop (hostname) with time and Loss %

mtr google.com.

Output:

My traceroute [v0.95]

fedom (172.16.75.86) → google.com (142.251.221.206)

2025-07-14 09:04 32+0530

Keys: Help Display mode Restart statistics order of field quit..

	Host	Loss%	Packet Snt	Last	Avg	pings Best	Wrst	Stdev
1.	gateway	0.0%	219	2.2	5.6	1.8	8.25	10.7
2.	142-250-171-161	0.0%	261	7.2	12.0	4.6	253.2	21.9
3.	142.251.227.215	17.0%	261	1755	185.6	154.9	344.2	32.8
4.	Pnvaa-ba-in-84 1e1 do.net	12.9%	388	287.6	186.5	169.2	50.2	35.7

b) Show numeric IP address

mtr.g google.com.

c) Show the numeric IP address and hostname to

mtr:b google.com

d) Set the number of pings

mtr.c to google.com.

4) tcpdump:

It is designed for (got using and display packets.

For install: dnf install -y tcpdump.

Result:

Hence the various Network command used in linux and windows are studied successfully.

Ex.NO :2

18/7/25

Study of Network cables.

Aim:

Study of different types of Network cables.

a) understand different types of network cables.

Different type of cables used in networking are:

1. unshielded Twisted pair (UTP) cable.
2. shield Twisted pair (STP) cable.
3. Coaxial cable.
4. Fibre optic cable.

Cable type	category	Maximum Data Transmission	Advantage & disadvantage	Application use
UTP	category 3	10 bps	Advantage → cheaper in cost → Easy to install as they have smaller overall diameter Disadvantages → More phone to (EMI) Electromagnetic interference and noise	10 base - Ethernet Fast Ethernet Gigasit Ethernet
	Category 5	upto to 100 Mbps		
	category 5e	1Gps		Fast Ethernet Gigiadit Ethernet
STP	category, 6a	10Gps	Advantages → shielded → Faster than → less susceptible to noise & Inferenc Disadvantage → Expensive → greater installation effort	Grigasit Ethernet, 10G Ethernet (55 m) widely used in data centres
	category 7	10Gps		Gigasit Ethernet, 10G Ethernet (100 m)

Conical cable	RG-6 RG-59 RG-11	100 to 100 Mbps	→ High bandwidth → Immune to interference → Less bandwidth → versatile Disadvantage → limited distance → cost → size is bulky	Speed of signal is 500m Television network High speed internet connection.
Fibre optics cable	single mode Multi mode	100Gbps	Advantages → High speed → High bandwidth → High security → Long distance disadvantages → Expensive → Requires skilled installer	Maximum distance of fibre optics cable is around 100 metres.

student observation!

1) Difference between cross cable and straight case.

A straight cable connects different device while cross cable connects the similar device by swapping transmit and receive wires.

2) which type of cable is used by connect two pc?

cross cable is used to connect two pc's

3) which cable is used to connect a router to pc?

straight.

4) Find the category of twisted pair cables used to connect the pc to network socket?

category 5e and category 6.

5) write down understanding challenges faced & output receiver?

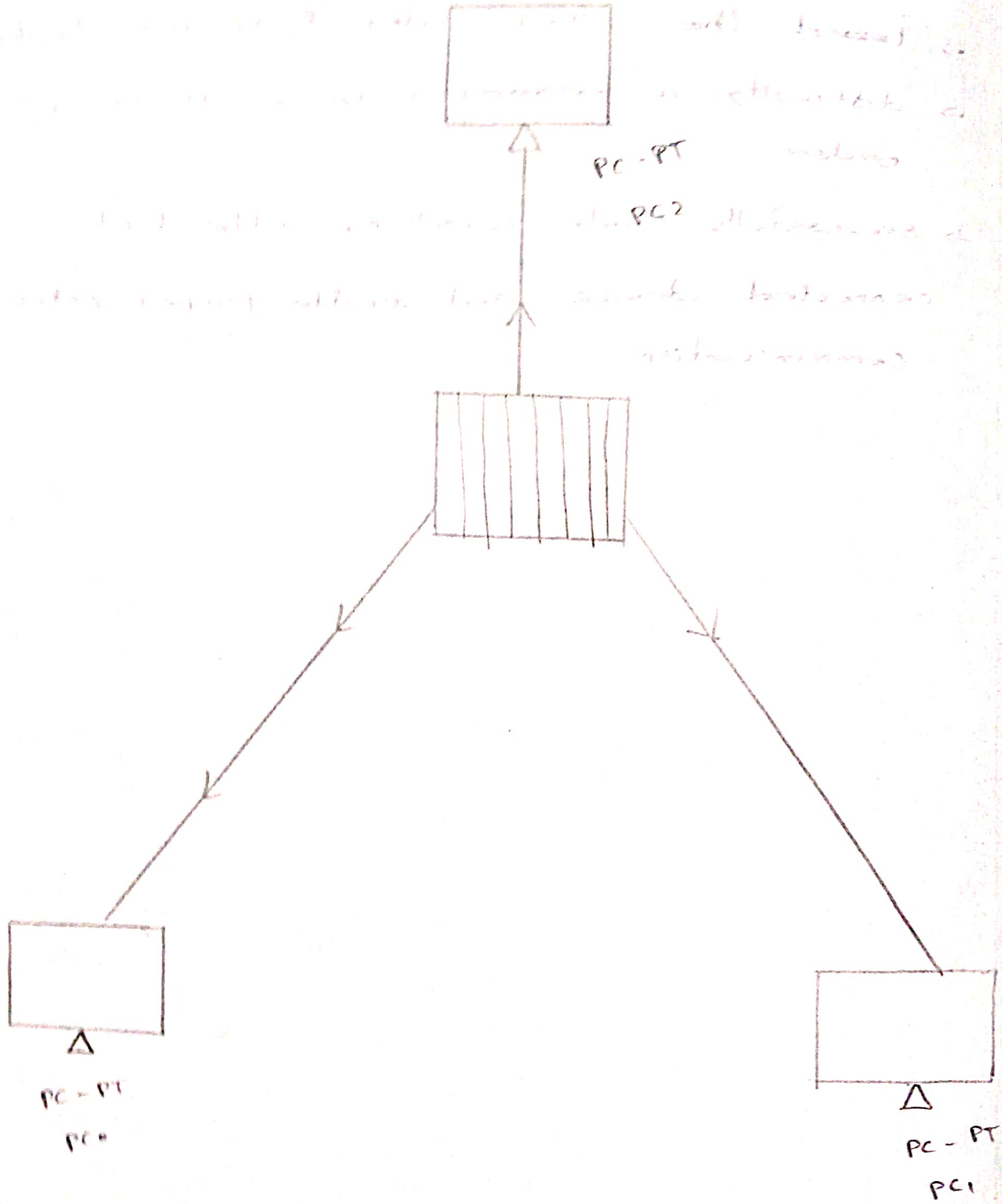
→ Learnt the colour codes & wiring standard.

→ difficulty in arranging wires in correct order.

→ Successfully Made working cable that connected device and enable proper network communication.

Result:

Hence the different type of network cable have been successfully studies.


PC - PT
PC2
PC - PT
PC0
PC - PT
PC1

Ex.NO: 03

21/07/25

Experiment on CISCO PACKET TRACER (Simulation Tool):

Aim: To study the packet tracer tool Installation and user Interface overview.

a) To understand environment of CISCO PACKET TRACER to design simple network.

* CISCO packet tracer (simulation Tool) was successfully installed on desktop.

b) Analyse the behaviour of network device using CISCO PACKET TRACER SIMULATOR.

1. From the network components box, click & drag & drop the below components
   a) 4 Generic PC's & one HUB
   b) 4 Generic PC's & one Switch.

2. click on connection:
   a) click on copper straight-Through cable,
   b) Select one of the PC & connect it to HUB using the cable. The link LED should glow in green, indicating that link is up.

Similarly connected remaining 3PCS to HUB.

c) Similarly connect 4PCS to the Switch using copper straight through cable.

PC0

IP configuration

IP configuration

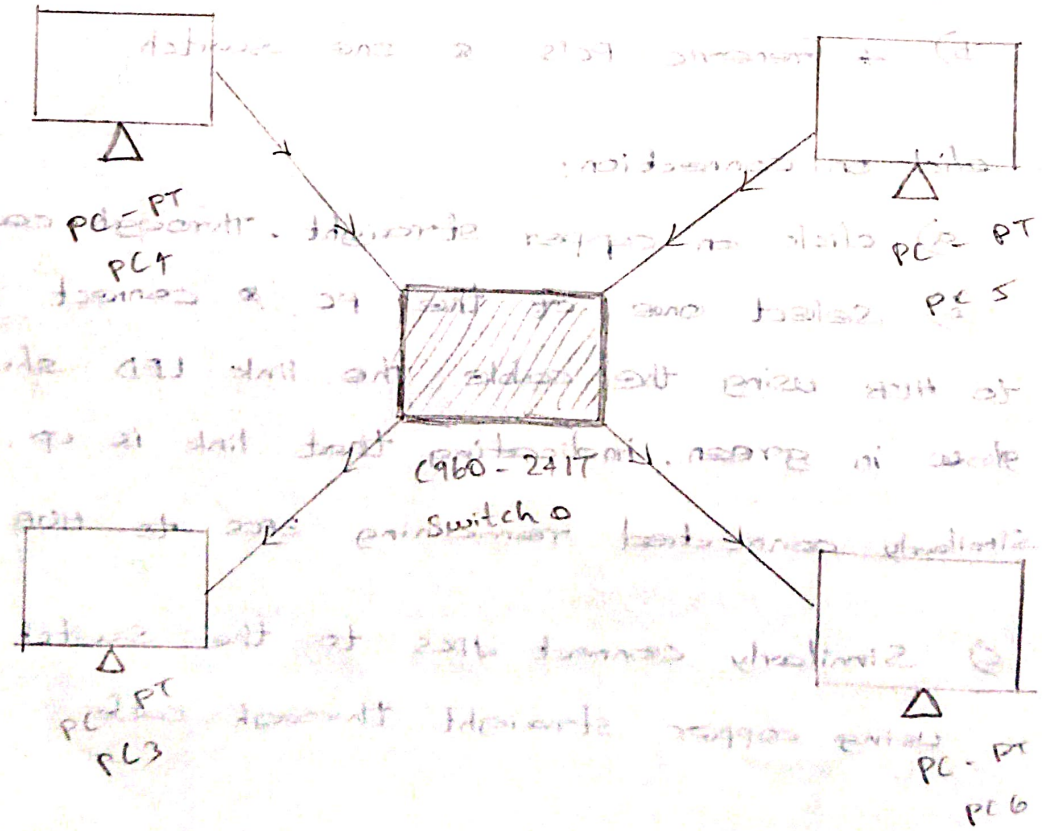
○ DHCP ⊙ static

IP address : 10.1.1.1

Subnet : 255.0.0.0

Default : [illegible]

Server : -



3) click on PC's connected to hub, go to Desktop tab, click on IP configuration & Enter on IP address & Subnet mask. Here default gateway & DNS server information is not needed there are only two end device is network.

4) observe the flow of PDU from source PC to destination PC by selecting Real time mode of simulation.

5) Repeat step #3 to step #5 for PC's connected to switch.

6) observe how HUB and switch are forwarding PDU & write your observation & conclusion about the behaviour of switch & HUB,

student observation:

a) From your observation write down behaviour of switch & HUB in terms of forwarding the packet received by them..

Switch: Forwards packet only to device with the Matching MAC address using MAC table.

HUB: Broadcast packets to all connect device regardless of the destination.

b) Findout the network topology implemented in your college & draw label the topology.

Star topology where all device connect to a central switch or hub..

Result:

Hence, study of packet tracer tool was successfully installed & drag & drop successfully.

Ex. NO: 4

Setup and configure a LAN (Local area network) using a Switch and Ethernet cable in your Lab.

Aim:

Setup and configure a LAN (Local area Network) using a switch and Ethernet cables in your Lab.

LAN:

-> LAN refers to network that connects device within a Limited area.

-> It enables users to share resources, data, printer and internet access.

-> LAN connects device to promote collaboration & transfer information between users such as computers, printers, servers, and Switches.

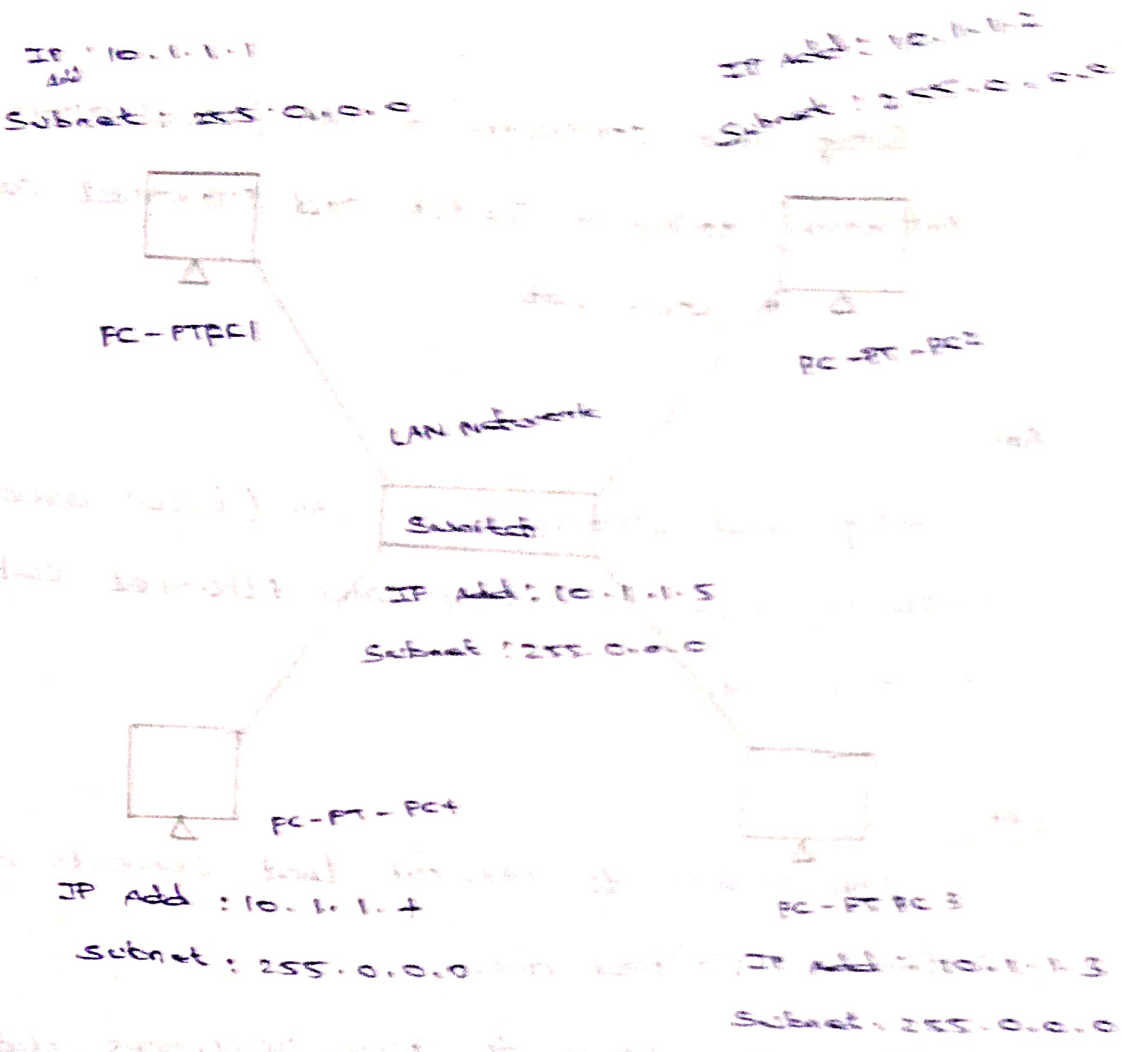
-> A LAN switch servers as a primary connecting device within the local network.

How to setup a LAN:

i) Plan and Design an appropriate network topology taking into account network requiremen & equipment location.

ii) You can take + computer switch is sufficient for network of the size & 4 ethernet calls.

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.

iii) connect your computers to network switch via. an Ethernet table, which is as simple plugging one call end of ethernet cable into network switch.

iv) Assign IP address to your PC.

→ Log on to client compter as owner.

→ click network & Internet protocol → click.

PC1 - IP Address : 10.1.1.1, Subnet Mark : 255.0.0.0
PC2 - IP Address : 10.1.1.2, Subnet Mark : 255.0.0.0
PC3 - IP Address : 10.1.1.3, Subnet Mark : 255.0.0.0
PC4 - IP Address : 10.1.1.4, Subnet Mark : 255.0.0.0

v) configure network switch convert your computer to switch using ethernet cable for web Interface.

→ Login and open it and Enter IP address of switch in address bar. This should bring up Login page for Login page switch Interface.

→ Assign IP address as 10.1.1.5 with subnet mask 255.0.0.0.

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

vii) select as folder - Go to Properties → click showing tab → share it with everyone on same LAN.

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

Result:

Hence, we have configured the LAN connection using switch & ethernet cables sucessfully.

EX.NO: 5 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 & display the control of the various protocols field in messages.

→ passive program :- never send protocol itself.
- no packets addressed to it
- receives a copy of all packet

packet sniffer structure Diagnostic Tools:

- TCPDUMP

Eg; tcpdump -enx host 10.09.41.2 -w exe-3.out

- Wireshark

Eg ; wireshark -r exe3.out.

Description:

Wireshark:

A network analysis tool primarily known as Ethereal, captures packet in real time and display them in human-readable format Wireshark include filters, color coding and other feature into network traffic.

What we can do with wireshark:

→ Capture network traffic.

→ Decode packet protocols using dissectors.

→ Watch smart statistics.
→ Analyse problems.

Wireshark uses:

→ Network administrators: troubleshoot network 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 (# line = 1 packet)
→ packet detail pane: show detail protect down.
→ packet Bytes pane: Display raw data.

Color coding:

Light purple: TCP, Light Blue - UDP
Black: packet errors / out of order.

Sample captures:

Load sample captures files from Wireshark via File → open. Save your own captures with File → save.

Filtering packets:

- Apply filters in top filters bar.

  Eg: dns → shows only DNS packets.

- Follow stream: Follows → TCP stream to view full conversation.

Inspecting packet:

- select a packet to explore detailed field 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: 06

Experiment on packet capture tool.

Hamming code

Aim:

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

Error correction at Data Link Layer:

→ Hamming code is a set of error correction code that can used to detect & correct errors that occur when data is transmit from sender to receiver. It technique developed by Hamming for EC.

Create sender program with below features:

→ Input if sender file should be a text of any length. program should convert text into binary.

→ Apply Hamming code concept on binary data & add redundant bit to it.

→ save this output in file called channel.

Create a receiver program with below features:

→ Receiver program should read input from channel file.

→ Apply hamming code on binary data to error.

→ If there is error, display position of error.

→ else remove the redundant bits and convert binary data into & 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):
    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
    erro- pos = 0
    for i in range(r):
        pos = 2**i
        parity.=0
    for j in range(1, n+1):
        if j & pos: parity n = code[-j]
    if parity != 0: err-post= pos
    if err-pos != 0 and err-pos <= n, code[err-pos]=
    return ''.join(map(str, code)), err-pos,
```

```
if __name__ == '__main__':
```

* Sender side

```
data = input("Enter the data to send:")
print(f"Original data: {data}")
encoded = encode_data(data)
print(f"Sent data: {encoded}")
```

* Make error

```
err = int(input(f"Choose bit position to be
      changed (1 to {len(encoded)}):"))
received = make_error(encoded, err)
```

* Receiver side

```
print(f"Received data: {received}")
corrected, pos = correct_data(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:07

Flow control at Data Link Layer

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.

Features:

- → Inputs window size & Message.
- → Sends window size frames at time.
- → Writes frames to senders buffer.
- → Receiver reads frames, sends Acks (or) Nack to Receiver buffer.
- → Sender reads Ack/ NACK and continues (or) resends frames.
- → You can manually edit file to simulate errors.

Code:

```
import time
import random
class Sender:
    def __init__(self, total_frames, window_size)
        self.total_Frames = total_Frames
        self.window_size = window_size
        self.base = 0
        Self.next_seq = 0
    def send_Frames(self):
        Print(f'\n [Sender] total frames to send:
                                {self.total_frames}")
    while self.base < self.total_frames:
```

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')]

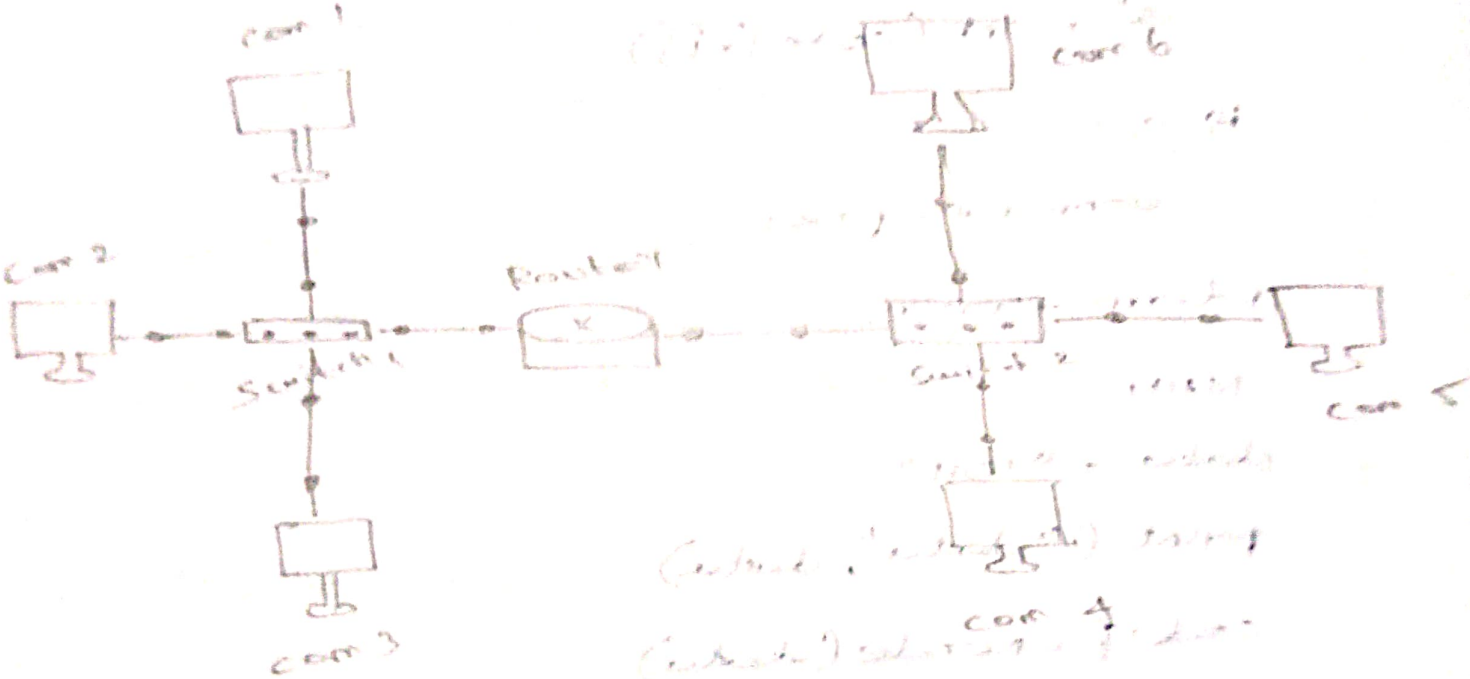
Accepted: L

Accepted : O

```
    error = 0
    for i in range(r):
        pos = 2 ** i
        s = 0
    for j in range (1, n+1)
        if j pos:
            s^= int (code [-j])
        if s:
            error += pos:
    return error.
-- MAIN --
    data = "1011"
    print (:" data", data)
    code = encode (data)
    print(" encoded", code)
    code = code [:3] + ('i' if code [3] = '0' else '0')
                                + code [+1]
    print ("received :", code)
    if err:
      print ("error at position :", err)
    code = code [: un (code) - err] + ('i' if code
                                [(code) - err]
      == '0' else '0'] + code [len (code) err +1]
    print (" corrected: " code)
    else
     print ("No error")
```

Result:

Hence the error detection & correction using Hamming code is done.


com 1
com 2
25
com 3
Router
com 4
25
com 5
com 6

EX.NO:8

NMAP - TRY HACKME

Aim:

This experimental 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 system will merely waste time and create unneeded network issue.

1) ARP Scan:

This scan uses ARP requests to discover live hosts.

2) ICMP Scan:

This scan uses ICMP requests to discover live hosts.

3) TCP / UDP ping Scan:

this scan sends packets to TCP ports & UDP ports to determine live hosts.

- arp-scan
- masscan.

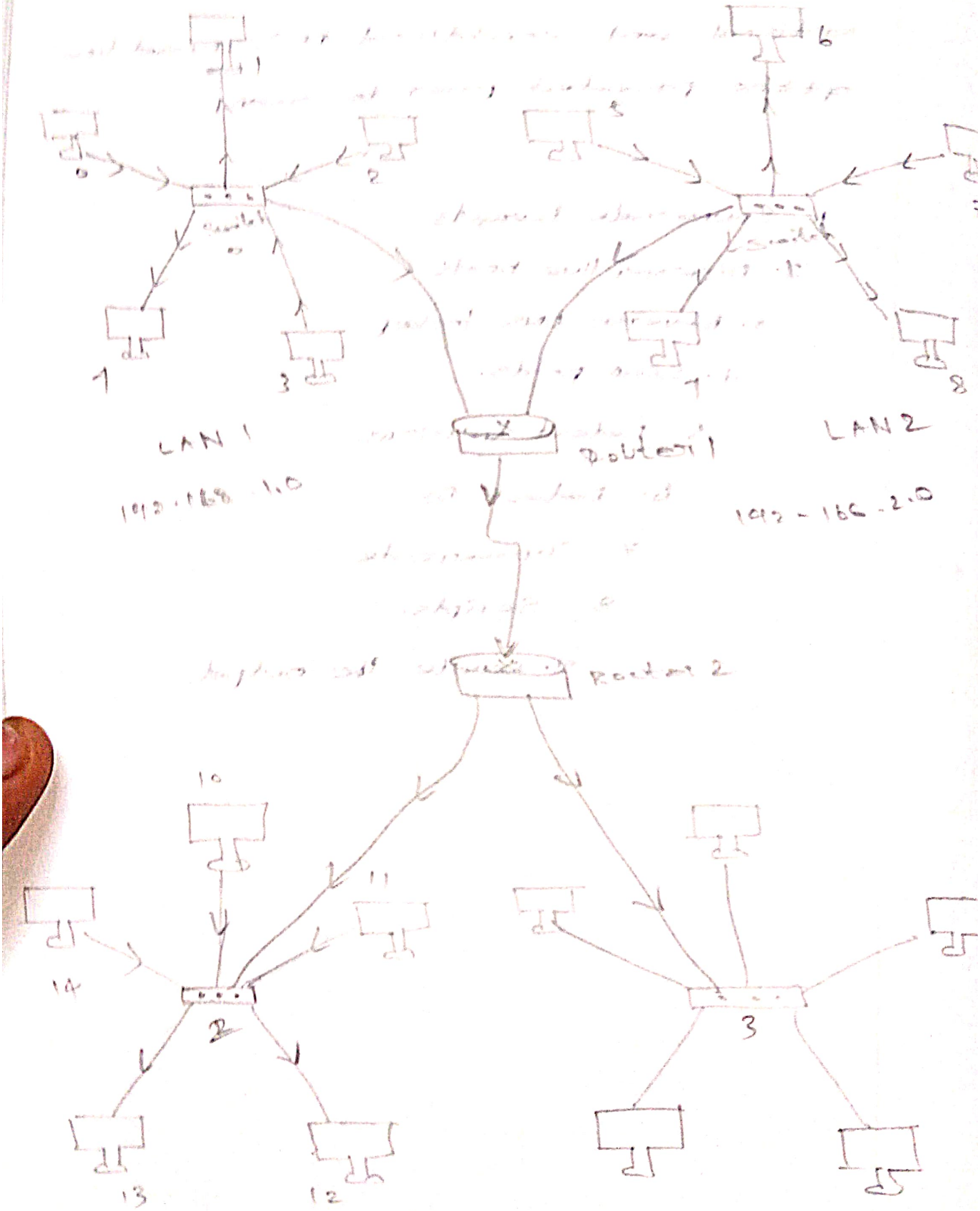
Nmap is a well known tool for mapping networks, locating live host & detecting running services, Its scripting engine can be used to extend its capabilities such as fingerprinting services & exploiting flaws.

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

1. Enumerate targets
2. Discover live hosts
3. Reverse DNS lookup
4. Scan ports.
5. Detect versions.
6. Detect OS.
7. Traceroute.
8. Scripts.
9. Write the output.

Result:

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


1
3
LAN 1
192.168.1.0
Router 1
LAN 2
192.168.2.0
Router 2
10
11
14
2
13
12
3
LAN 3
192.168.3.0
LAN 4
192.168.4.0

Ex.NO: 9

Implementation of subnetting in CISCO PACKET TRACER.

Aim:

To implement subnetting in cisco packet tracer simulator.

Procedure:

1) Create Network Topology:

- open CISCO packet tracker → New → Network → Generic.
- Add devices: Routers (R_1, R_2), switches (S_1, S_2)
- Connect them using straight 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 subnet 30 hosts.

3) IP Addressing

- Router R_1

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:

connect 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).

4) Configuration commands:

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

observation:

Fmc	Last status	Source	Dest	Type	color	Time	Period	Num
*	Success	PC1	PC5	ICMP		0.0	N	1
*	Success	PC6	PC10	ICMP		0.0	N	2
*	Success	PC11	PC0	ICMP		0.0	N	3

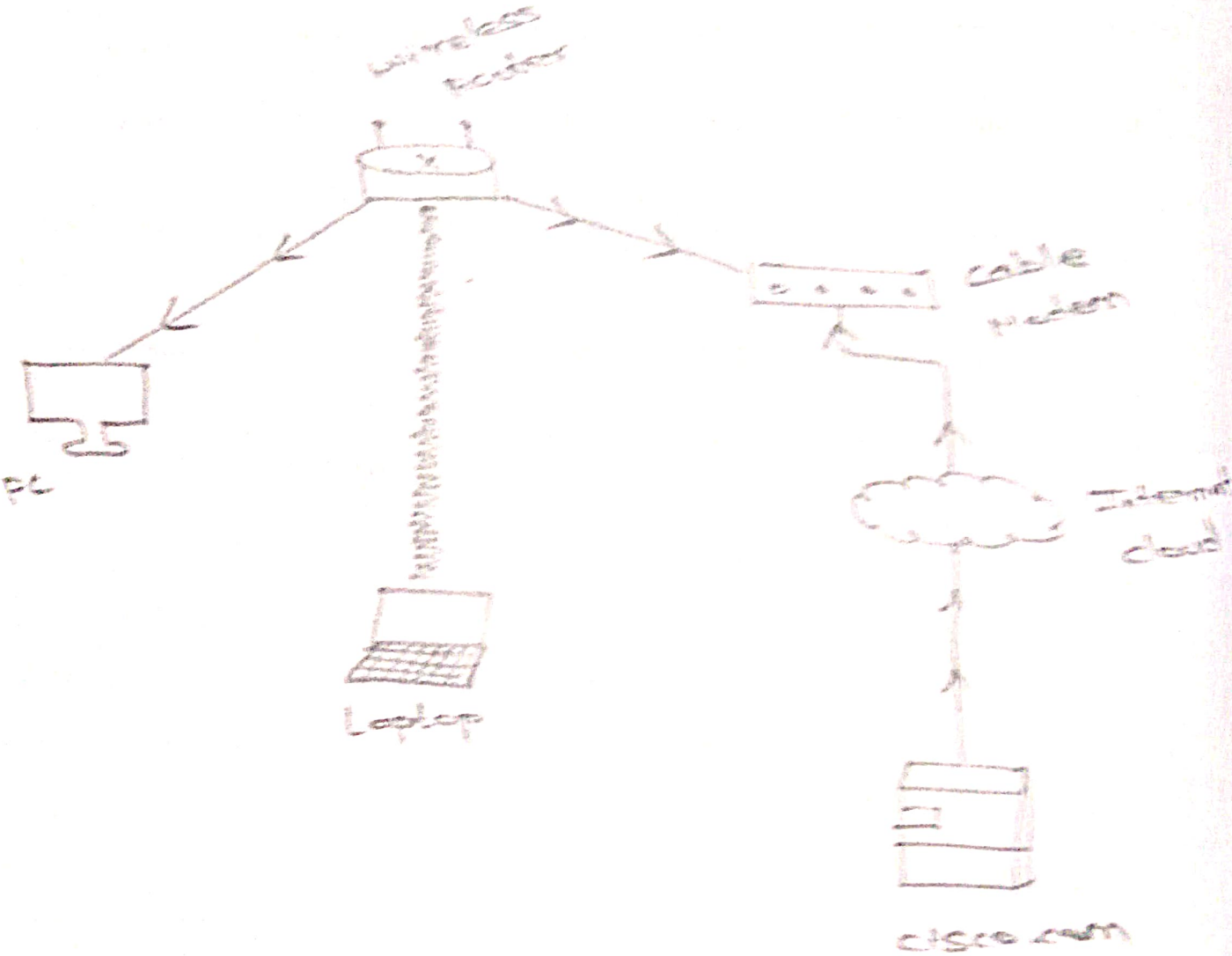
5) Testing:

- Use ping command between pcs and routers successful ping - proper connectivity & subnetting implementation.

Result:

Thus the given implementation of subnetting mask is executed successfully.

Diagram:



Ex. NO:10

INTERNETWORKING WITH ROUTERS.

Aim:

To perform internetworking with routers in CISCO packet tracer simulator.

a) Design & Configure Internetwork using Routers:

* (Configure router 1)

→ select router > open CLI

→ Press ENTER to start configuring Router 1.

→ Type enable to activate priviledged mode.
Router # CLI.

* (Configure PCs)

→ Assign IP address to energy PC.

→ Select PC go to desktop > IP configuration.
assign IP, gateway & subnet mask.

→ PC=0 = 192.168.10.1 - default gateway.

* (connecting PCs with router)

→ connect FastEthernet0 port of PC0 with FastEthernet 0/0 port of Router 1 using copper. - straight through cable.

→ Router configuration is done.

b) Design & configure internetwork using wireless Router DHCP server & internet cloud

Objectives:

- Build simple network in logical topology. Workspace.
- Configure network devices
- Test connectivity between them.
- Save files & close packet tracer.

IP configuration : Fast Ethernet 0

⊙ 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) port 1:

- Launch packet tracer & Building topology.

ii) port 2:

- configure wireless Router, Laptop then PC.
- Configure Internet cloud.
- configure cisco.com server.

iii) port 3:

- Refresh IPV4 setting on PC

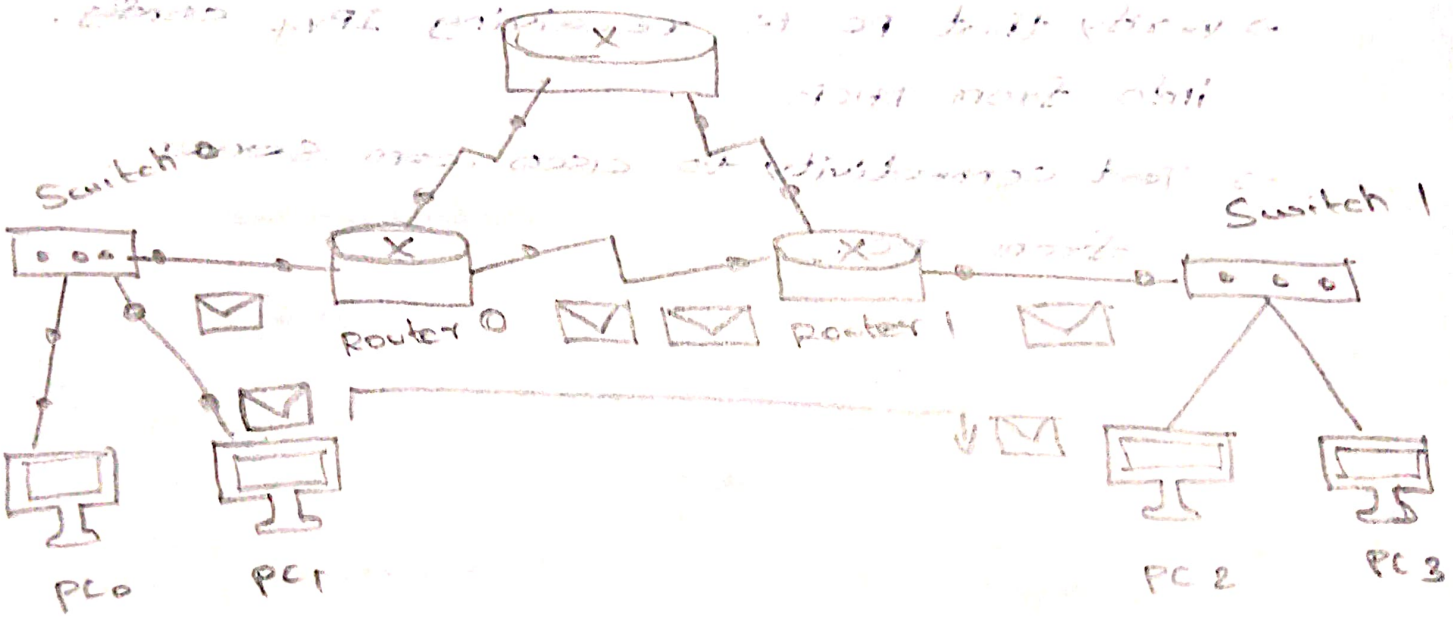
→ verify that PC is receiving IPV4 config. info from DHCP.

→ Test connectivity to cisco.com server from PC.

Result:

Thus the above experiment to implement internetwork is executed successfully.

Diagram :



EX.NO:11

SIMULATE STATIC ROUTING CONFIGURATION USING CISCO PACKET TRACER AND RIP

Aim:

To Simulate RIP using cisco packet tracer.

Assign IP address to interface of routers:

Double click Router0 & click CLI & press enter key to access the command prompt & return 0.

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 10.0.0.01 255.0.0.0 command will assign IP address to interface

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
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

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 less hops counts between source & destination. In our network, route 1 has less hops so it will be selected we can use traceroute command to verify it

Result:

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

output screen;

client:

Enter message : Hello world

Echo from server : Hello world

Enter message : How are you?

Echo from server : Hello are you?

Enter message : Bye.

Server:

Server is waiting for connection

Connected to ('127.0.0.1', 9999)

Received from client : Hello world

Received from client : How are you?

connection closed.

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
```

```
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 tcp / udp socket is done successfully.

output screen:

server side:

Server waiting for connection
connected to ('127.0.0.1', 9010)
client : Hi server !
you: Hello client !
client : How are you ?
you: Im fine, thanks !
client. Bye !
client disconnected.

client side:

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

(b) Aim:

To implement chat client server using TCP/UDP sockets.

Server Side Algo:

```
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
    reply = input("you: ")
    if reply.lower() == 'bye': break
    conn.close()
```

client side Algo:

```
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 case implemented client server chat successfully.

output screen:

server side:

command : python server.py

output:

UDP server running on 127.0.0.1 : 12345

received from (127.0.0.1 : 5437) . ping

client server:

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 Algo:

```
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 Algo:

```
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))
    Printf" UDP server run on {host} : {port}")
    while true:
        data, addr = s.recvfrom(1024)
        print(f"Received msg from {addr}:
                                {data.decode}")
    start_server()
```

Result:

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

Sample I/p

- Open a webbrowser & visit https://www.google.com
- Run the command in another terminated 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 experimental on implementing packet sniffing using sockets has been executed successfully.