20/7/24

Practical - 1

Aim:-

Study of Network commands used in Linux and windows

Basic Networking Commands:-

(i) arp -a

Interface: 192.168.72.1 ... 0x3

Internet Address	Physical Address	Type
192.168.72.5 24	00-50-56-e4-31-a3	dynamic
192.168.72 .255	ff-ff-ff-ff-ff-ff	static
224.0.0.2	01-00-5e-00-00-02	static
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-ff-ff-fa	static
255.255.255.255	ff-ff-ff-ff-ff-ff	static

(ii) hostnamo : It displays name of computer

DESKTOP - MATHITT

(3) ipconfig /all

Windows IP configuration

Hostname . . . . . : DESKTOP-MATHITT

Primary Dns Suffix . . . . :

Node Type . . . . . : Mixed

IP Routing Enabled . . . . : NO

WINS Proxy Enabled . . . . : NO

Ethernet adapter Ethernet:

connection-specific Dns Suffix . :

Description . . . . . : Realtex PCIe GbE Family Controller

Physical Address . . . . . : 50-9A-4C-35-0F-45

DHCP Enabled . . . . . : NO

Auto Configuration Enabled . . . . . : Yes

Link-Local IPV6 Address . . . . : fe80::bc8c:9910:dbd : a9f9%18(Pref)

(4) nbtstat -a

Displays protocol statistics and current TCP/IP connections using NBT

NBTSTAT [[-a Remote Name][-A IP address][-c][-n] [-r] [-R] [-RR] [-s] [-S][interval]]

(5) netstat

Activate Connections

Proto	Local Address	Foreign Address	State
TCP	192.168.8.112:50186	[illegible]	ESTABLISHED
TCP	192.168.8.112:50187	[illegible]	ESTABLISHED
TCP	192.168.8.112:50188	[illegible]	LAST_ACK

(6) nslookup

Default Server : Unknown

Address : 192.168.8.1

> www.google.com

Server : Unknown

Address : 192.168.8.1

Non-authoritative answer:

Name : www.google.com

Addresses : 2404:6800:4007:810::2004

142.250.183.228

(7) pathping

Usage : pathping [-g host-list] [-h maximum_hops]

[-i address] [-n] [-p period] [-q num_queries]

[-w timeout] [-4] [-6] target_name

(7) Ping 172.16.8.112

~~Usage~~ Pinging 172.16.8.112 with 32 bytes of data:

Reply from 172.16.8.112: bytes=32 times<1ms TTL=128

Reply from 172.16.8.112: bytes=32 times<1ms TTL=128

Reply from 172.16.8.112: bytes=32 times<1ms TTL=128

Reply from 172.16.8.112: bytes=32 times<1ms TTL=128

Ping statistics for 172.16.8.112:

Packets: Sent=4, Received=4, Lost=0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 0ms, Maximum = 0ms, Average 0ms.

(8) route

Manipulates network routing Tables

Route [-f] [-p] [-4] [-6] command [destination]

[mask netmask] [gateway] [metric metric]

[if interface]

Some Basic Commands for linux:-

(1) ip

ip address show

1: lo : <LoopBack, UP, Lower_UP> mtu 65536 qdisc
noqueue state unknown group default qlen 1000
link /loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet 6::1/128 scope host
valid_lft forever preferred_lft forever.

2) ifconfig

enp2s0 : flags = 4163 <UP, BROADCAST,
RUNNING MULTICAST> mtu 1500
inet 172.16.8.94 netmask 255.255.255.0
broadcast 172.16.11.255
Lo : flags = 73 <UP, LOOPBACK, RUNNING
>mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet 6 ::1 prefixlen 128 scopeid 0x10
<host>

wlp3s0 : flags = 4099 <UP, Broadcast,
Multicast> mtu 1500
ether 2a:cd:c4:4f:48:cb

3) Mtr google.com

localhost. local domain (0.0.0.0)

Ping Pabit Pattern : 0

Pattern Range : 0(0x00) - 255 (0xff) < 0
random

Host

1) 172.168.1

2) static-41.229.249-tataidco.in

3) 142.250.171.162

4) 142.251.127.215

5) 142.250.288.1

6) Maa 05512-in-f14.1e100.net

5) Ping:

Ping google.com (216.58.200.142) 56 (84) bytes of data

64 bytes from Maa 05510-in f14.1e.100.net (216.58.200.142):

icmp-seq = 1 ttl = 120 time : 3.31 ms

64 bytes from Maa 05510 - in f14.1e100.net (216.58.200.142).

ICMP-seq = 2 ttl = 120 time: 3.51 ms

1) Which command is used to find the reachability of a host machine from your device

Ping <hostname>

2) Which command will be give the details of hops taken by a packet to reach its destination

traceroute <hostname>

3) Which command display the IP configuration of your machine

IPconfig, ifconfig

4) Which commands displays the TCP port status in your machine

netstat

5] Write the modify the IP configuration in linux machine

→ login as a root user and start terminal
→ Input "ifconfig" command prompt
→ input "ifconfig" followed by the network interface and your new IP address
→ Press "Enter".

Result:-

Thus the Network Basic commands of windows & linux are studied.

24/7/24

Practical 2

24/7/24

Aim:

Study of different types of Network Cables

Cable type	Category	Maximum Data Transmission	Advantages / Disadvantages	Appl/ Use
UTP	Category 3	10 bps	Advantages · cheaper in cost · Easy to install as they have small Disadvantages · More prone to (EMI) Electromagnetic interference and noise	10-Base-TE the
	Category 5	Up to 100 mbps		Fast Ethernet Gigabit Ethernet
	Category 5e	1 Gbps		
STP	Category 6, 6a	10 Gbps	Advantages · Shielded · Faster that UTP · Less susceptible to noise Disadvantages - Expensive · Greater installation effort	Gigabit Ethernet 10G Ethernet Widely used in data centers
SSTP	Category 7	10 Gbps		Gigabit Ethernet 10G Ethernet

Types of Network	Advantages / Disadvantages	Applications / Use
	Advantages · cheaper in cost · Easy to install as they have small Disadvantages · More prone to (EMI) Electromagnetic interference and noise	10-Base-TE then Fast Ethernet Gigabit Ethernet
	Advantages · Shielded · Faster than UTP · Less sensitive to noise Disadvantages · Expensive · Greater installation effort	Gigabit Ethernet 10G Ethernet Widely used in data centers Gigabit Ethernet 10G Ethernet
Coaxial cable	RG-6 RG-59 RG-11	10-100mbps	· High bandwidth · Less low bandwidth · Versatile Disadvantages · Limited distance · Cost · Size is bulky	Speed of signal is soon Television network High speed Ethernet connection
fibre optics cable	single mode Multi mode	100Gbps	Advantages High Speed High security long distance Disadvantages · Expensive · Requires skilled installers	Maximum distance of fibre around 100 meters

b) Make your own Ethernet Cross-over Cable/ straight Cable

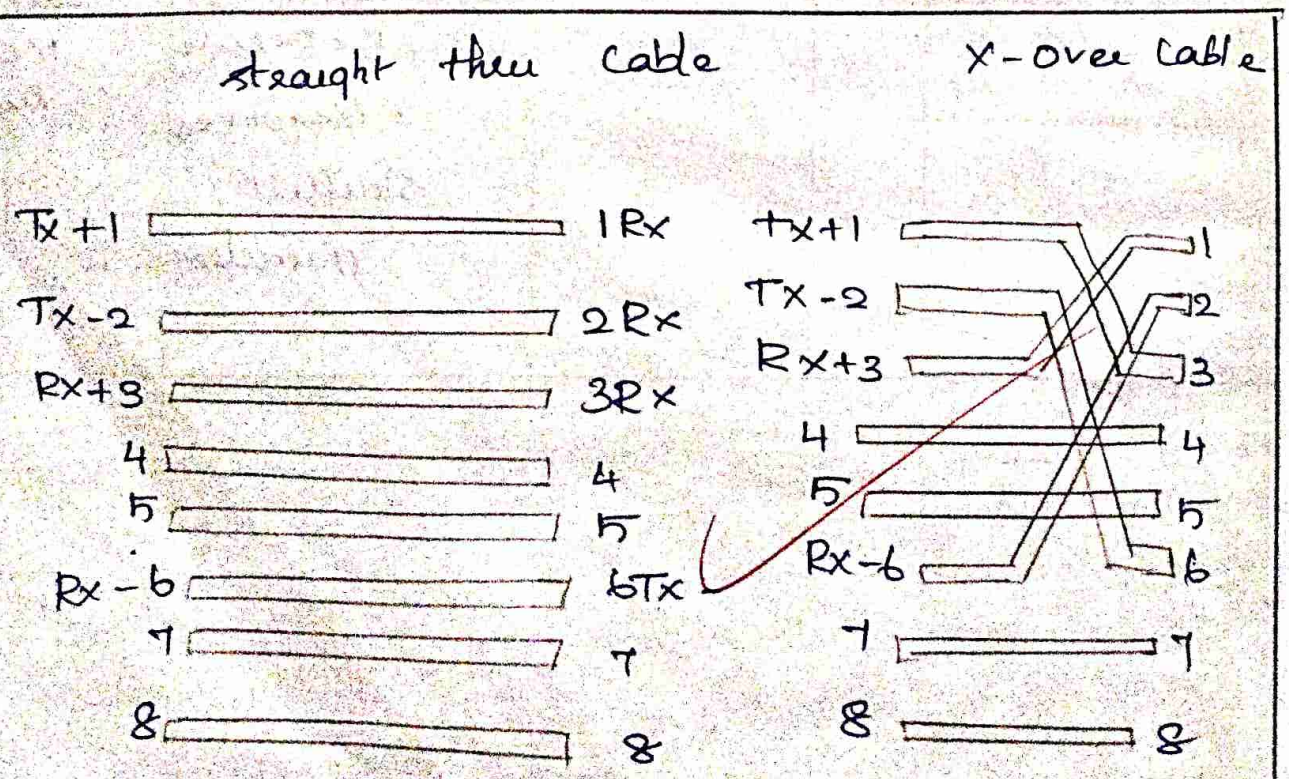
Tools and Parts needed:

Ethernet cabling, CAT 5e is certified for gigabit support, but CAT5 cabling works as well just over shorter distances

A crimping tool. This is an all-in-one networking tool shaped to push down the pins in the plug and strip and cut the shielding off the cables

Two RJ45 plugs

optional two plug shields



Observation:

1) Cross cable - connected for same devices
   Straight cable - connected for different devices

2) Cross cable

3) Straight cable

n) Untangling Involves arranging cables
Challenges: correct wire arrangement
Stripping without damage
Output: successful cables enable network.

Result

Thus the different types of network cable is studied.

31/7/24

31/7/24

Practical - 3

CISCO PACKET TRACER

Aim :-

To understand and study the Packet Tracer tool installation and user interface.

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

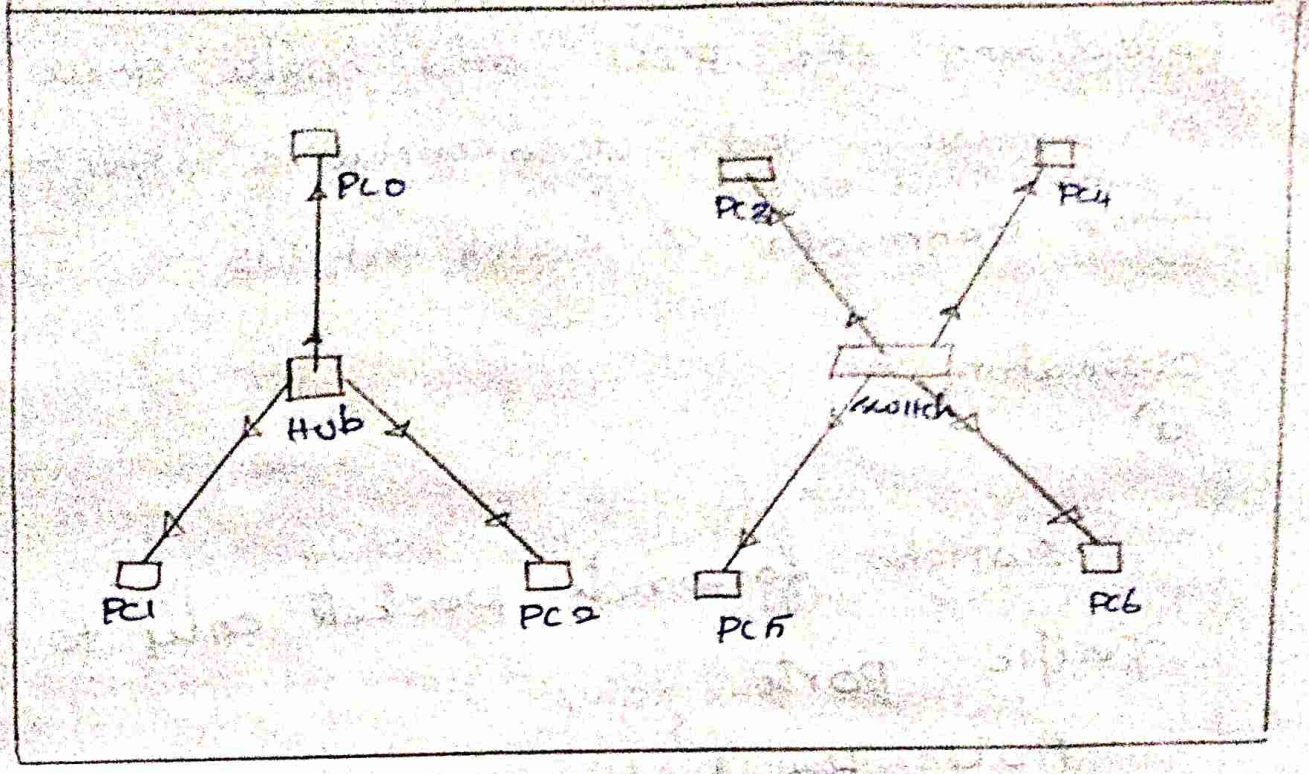
b. 4 Generic PCs and One switch

2. Click on Connections:

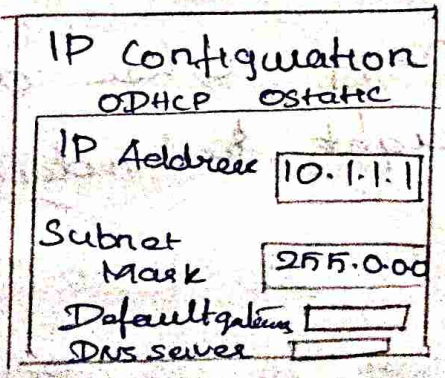
a. Click on copper straight-Through cable

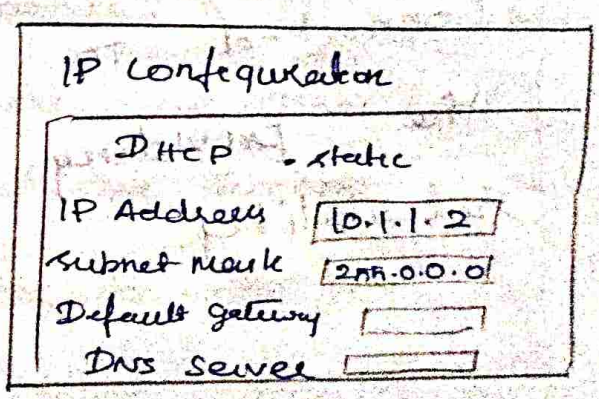
b. Select one of the PC and connect it to HUB using the cable. The Link LB should glow

c. Similarly connect 4PC to the switch using copper straight-Through cable



3. Click on the PCs connected to hub, Click on IP Configure and enter IP, the default gateway and DNS server information is not needed as there only two end devices.





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 switch

6. Observe how the HUB and switch can

forwarding the PDU and write your observation and draw conclusion about the behaviour of switch and HUB

Observation

a) :

Switch: forwards packets only to the specific ports

Hub: Broadcasts packets to all the connected devices

b) :

Mesh topology

In a mesh topology each devices is connected to every other devices in the providing high redundancy and reliability.

Result :-

The equipment was successfully executed and o/p is verified.

15/8/24

Practical-4

AIM: Setup and configure a LAN using a Switch and Ethernet cables in your lab.

How to setup a LAN

Step1: Plan and Design an appropriate network topology taking into account network requirement and equipment location

Step2: You can take 4 computers, a Switch with 8, 16 or 24 ports which is sufficient for networks of these sizes and 4 Ethernet cables

Step3: Connect your computers to network switch via Ethernet cable which is as simple as plugging one end of the Ethernet cable into your computer and the other end into your computer network switch

Step4: Assign IP address to your PC's

1. Log on to the client computer as Administrator or as Owner

2. click Network and Internet Connections

3. Right click Local Area Connection/Ethernet

Step5: Configure a network switch

1) Connect your computer to switch.

2) Log on to web interface

3. Configure basic settings once you're logged in. you will be able to configure basic setting

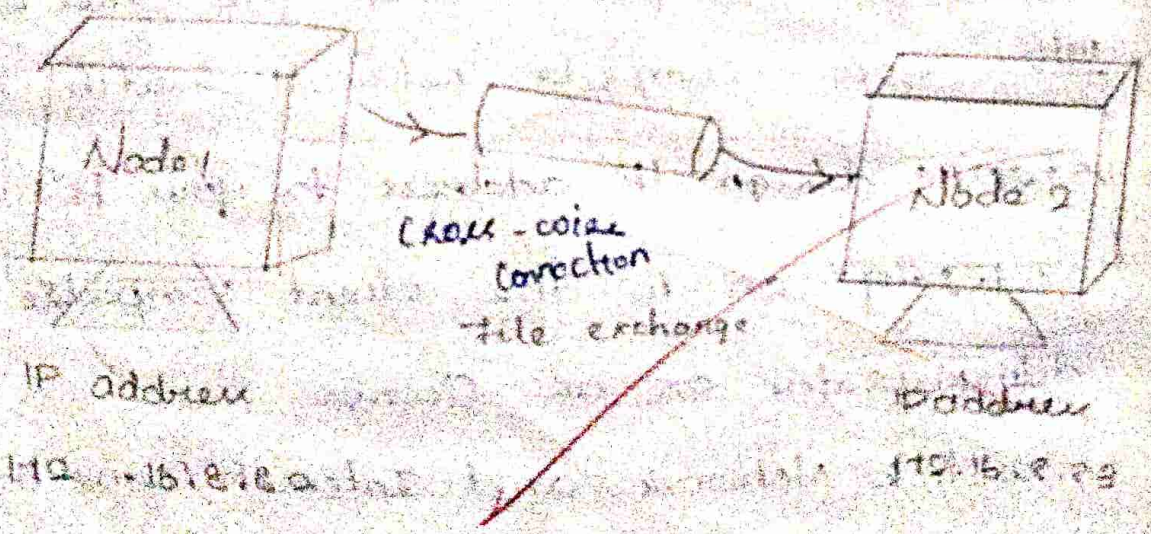
4. Assign IP address as 10.1.1.5 subnet Mask 255.0.0.0

Step 6: Check connectivity between switch and other machine by using ping command on the command prompt of device.

Step 7: select a folder → go to properties → Click sharing tab → share it with everyone to the same LAN

Step 8: Try to access the shared folder from others computers of the network.

Student observation:-



Result:-

IP configuration

Node : 1

IPV4 Address : 172.16.8.82

Node - 2 :

IPV4 Address : 172.16.8.83

Outcome :-

The file that sent that through [click Run [//172.16.8.82] → /users //REC //Desktop] is sent successfully.

Result :-

Thus the set up LAN is using switch and Ethernet cables is successfully connected and sent files.

10/8/7

9/8/24

Practical - 5

AIM:

Experiments on Packet capture tool: Wireshark

Packet Sniffer:

→ Sniffs messages being sent/received from/by Protocol fields in the messages.

→ Store and displays the contents of the Variance protocols fields in the Messages

→ Passive program

→ never sends Packets itself

→ no Packets addressed to it

→ receives a copy of all packets (Sent/received)

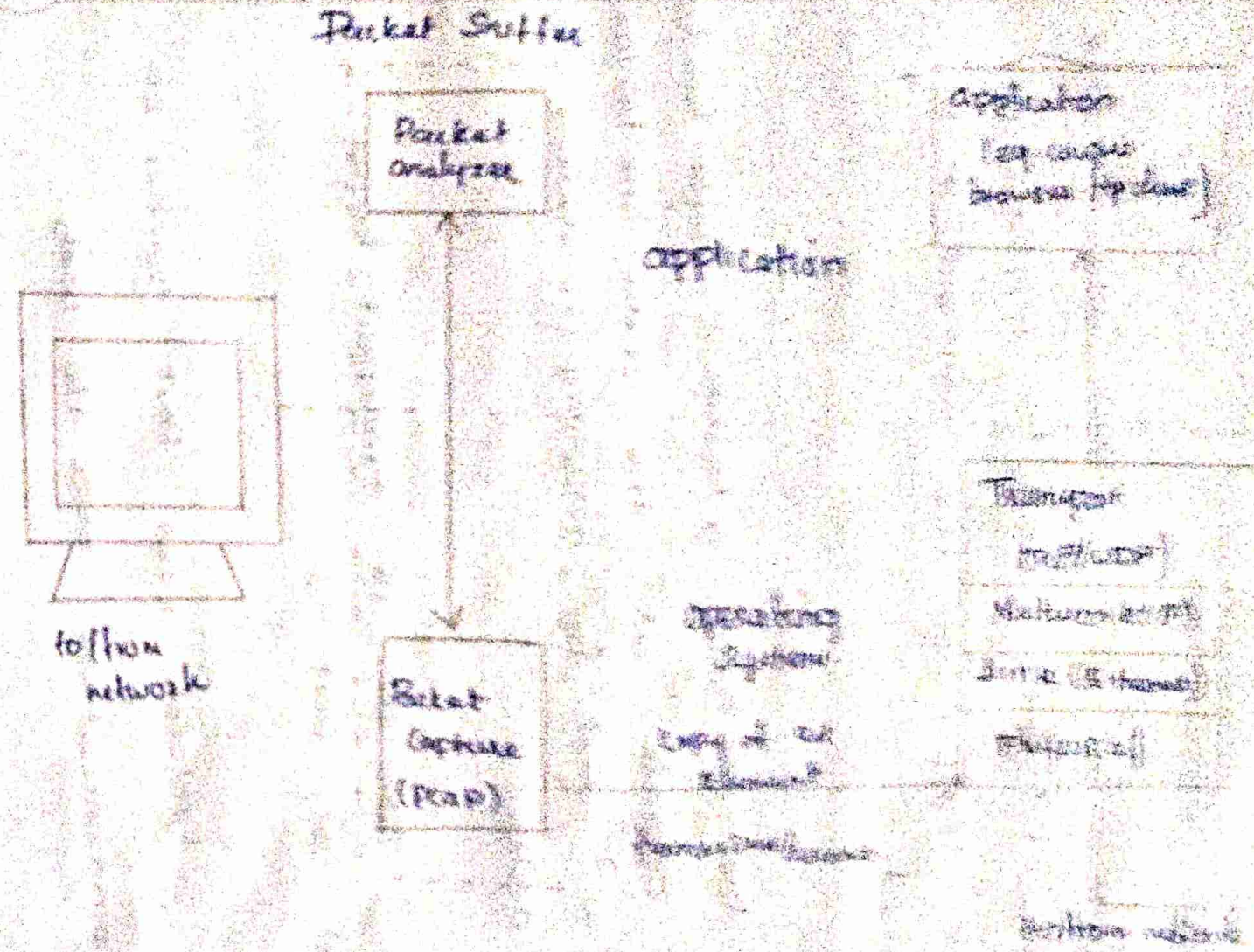
Packet Sniffer Structure Diagnostic Tools:

* TCPdump

→ tcpdump_exn host 10.129.41.2-u

* Wireshark

→ Wireshark -r-exe3.out


Packet Sniffer
Packet analyzer
application
Packet Capture (PCAP)
operating system
Transport
Network
Link (Ethernet)
Physical
Bottom network

Student observation

1) It refers to setting that allows the network interface card to capture all network traffic on the segment it is connected.

2) It doesnot have a transport layer header. they operate at the data link layer and are used for mapping IP address to MAC address

3) Uses UDP as its transport layer protocal on port 53

4) HTTP protocol uses port number 80 for communication over the web

5) It is typically '255.255.255.255' is used to send a packet to all device in network segment.

RESULT :-

Thus, wireshark tool has been experiment on packet captured and studied

21/8/24

Exp. no. 6 Hamming Code
Date: 21/8/24

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.

Student observation:-

Sender.Py (filename)

```
import os
def text_to_binary(text):
    return ''.join(format(ord(char), '08b')
                   for char in text)

def calculate_redundant_bits(m):
    r = 0
    while (2**r) < (m+r+1):
```

```
    r += 1
    return r

def position_redundant_bits(data,r):
    j = 0
    k = 1
    m = len(data)
    res = ''

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

def calculate_parity_bits(arr, r):
    n = len(arr)
    for i in range(r):
        val = 0
        for j in range(1, n+1):
            if j + (2**i) = (2**i):
                val = val ^ int(arr[-j])
        arr = arr[:n - (2**i)] + str(val) +
```

```
return arr

def save_to_channel(hamming_code):
    with open('channel', 'w') as file:
        file.write(hamming_code)

if __name__ == "__main__":
    text = input("Enter the text:")
    binary_data = text_to_binary(text)
    hamming_code = apply_hamming_code(
                                binary_data)
    Save_to_channel(hamming_code)

#receiver.Py
def read_from_channel():
    with open('channel', 'r') as file:
        return file.read()

def calculate_redundant_bit
        r=0
        while (2**r) < (m+r+1):
            r+=1
        return r

def detect_error(arr_nr):
        val=0
    for j in range(1,n+1):
        if j & (2**i) == (2**i):
            val = val ^ int(arr[-j])
        res = res + val * (10**i)
        return int(str(res),2)
```

```
def insert_error(arr, pos):
    if pos > len(arr):
        arr = arr[:len(arr)-pos] + str(1 - int
                (arr[len(arr)-pos:]))
    return arr
def remove_redundant_bits(arr, n):
    n = len(arr)
    j = 0
    res = ''
    for i in range(1, n+1):
        if i == 2**j:
            res += arr[i]
        else:
            j += 1
    return res[::-1]
def binary_to_text(binary_data):
    text = ''
    for i in range(0, len(binary_data), 8):
        byte = binary_data[i:i+8]
        text += chr(int(byte, 2))
    return text.
```

Result:

Thus the Hamming code to implement error detection is studied

11/9/24

14/9/24

Practical - 7
SLIDING WINDOW

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 atleast below give requirements. You can make it bidirectional program wherein receiver is sending its data frames with acknowledgement.

Student Observation:-

Sender.py

```
import time
import os
def input_window_size():
    return int(input("Enter window size:"))
def input_text_message():
    return input("Enter text message:")
def create_frames(text_message):
    frames=[(i,char) for i,char in enumerate
                                (text_message)]
    frames.append((len(text_message),'END'))
    return frames
def write_to_file(filename,data):
    with open(filename,'w') as file:
        for frame in data:
```

```
    file.write(f"{frame[0]},{frame[1]}\n")
def read_from_file(filename):
    if not os.path.exists(filename):
        return []
    with open(filename, 'r') as file:
        return [line.strip().split(',') for line in file.readlines()]

def send_frames(frames, window_size):
    i = 0
    while i < len(frames):
        window = frames[i:i + window_size]
        Print(f"Sending frames: {window}")
        write_to_file("Sender_Buffer.txt", window)
        time.sleep(3)
        receiver_buffer = read_from_file("Receiver_Buffer.txt")
        if not receiver_buffer:
            Print("No acknowledgement received yet.")
            continue
        ack_frame = receiver_buffer[0]
        ack_number, ack_type = int(ack_frame[0]),
                               ack_frame[1]
        if ack_type == "Ack":
            Print(f"Ack received for frame {ack_number}
        Sending next set of frames.")
            i += window_size
        elif ack_type == 'NAck':
            Print(f"NAck received for frame {ack_number}
        Re Sending frames from frame {ack_number}.")
            i = ack_number
```

```
def main_sender():
    window_size = input_window_size()
    text_message = input_text_message()
    frames = create_frames(text_message)
    send_frames(frames, window_size)
if __name__ == "__main__":
```

receiver.py

```
import random
import time
import os
def write_to_file(filename, data):
    with open(filename, 'w') as file:
        file.write(data)
def read_from_file(filename):
    if not os.path.exists(filename):
        return []
    with open(filename, 'r') as file:
        return[line.strip().split(',') for line in file.re
                                                    lines
def process_frames(frames):
    acks = []
    frame_number = int(frame[0])
    data = frame[1]
    if frame_number in frame_seen:
        continue
    Print(f"Received frame {frame_number} : {data}")
    if random.choice([True,False]):
```

```
        acks.append(f"{frame_number}, ACK\n")
        frame_seen.add(frame_number)
    else:
        Print(f"Sending NACK for frame {frame_number}")
        acks.append(f"{frame_number}, NACK\n")
        break
    return ''.join(acks)
def main_receiver():
    while True:
        time.sleep(3)
        frames = read_from_file('Sender_Buffer.txt')
        If not frames:
            Print("No frames to process, waiting...")
            continue
        acks = process_frames(frames)
        write_to_file('Receiver_Buffer.txt', acks)
        If any(frame[1] == 'END' for frame in frames):
            Print("END of transmission received.")
            break
If __name__ == '__main__':
    main_receiver()
```

Output

Python Sender.Py

Enter Window Size: 3

Enter text message: hello

Sending frames: [(0, 'h'), (1, 'e') (2, 'l')]

Ack received for frame 0, Sending next set of frames.

Sending frames: [(3, 'l'), (4, 'o'), (5, 'END')]

Ack received for frame 3, Sending next set of frames.

Python receiver.Py

Received frame 0: h

Sending Ack for frame 0

Received frame 1: e

Sending Ack for frame 1

Received frame 2: l

Sending Ack for frame 2

Result:

Thus the program for sliding window Protocol is successfully executed.

18/9/24

Exp.no-8a a) Visitual LAN

Date - 18/9/24

AIM:-

Simulate Virtual configuration using Cisco Packet Tracer simulation.

Procedure:-

1) create the network using 1 switch & 4 PCs as shown in figure.

2) check the network by sending packets (optional)

3) Assign IP address to the PCs

PC0 - 10.0.0.1
PC1 - 10.0.0.2
PC2 - 10.0.0.3
PC3 - 10.0.0.4

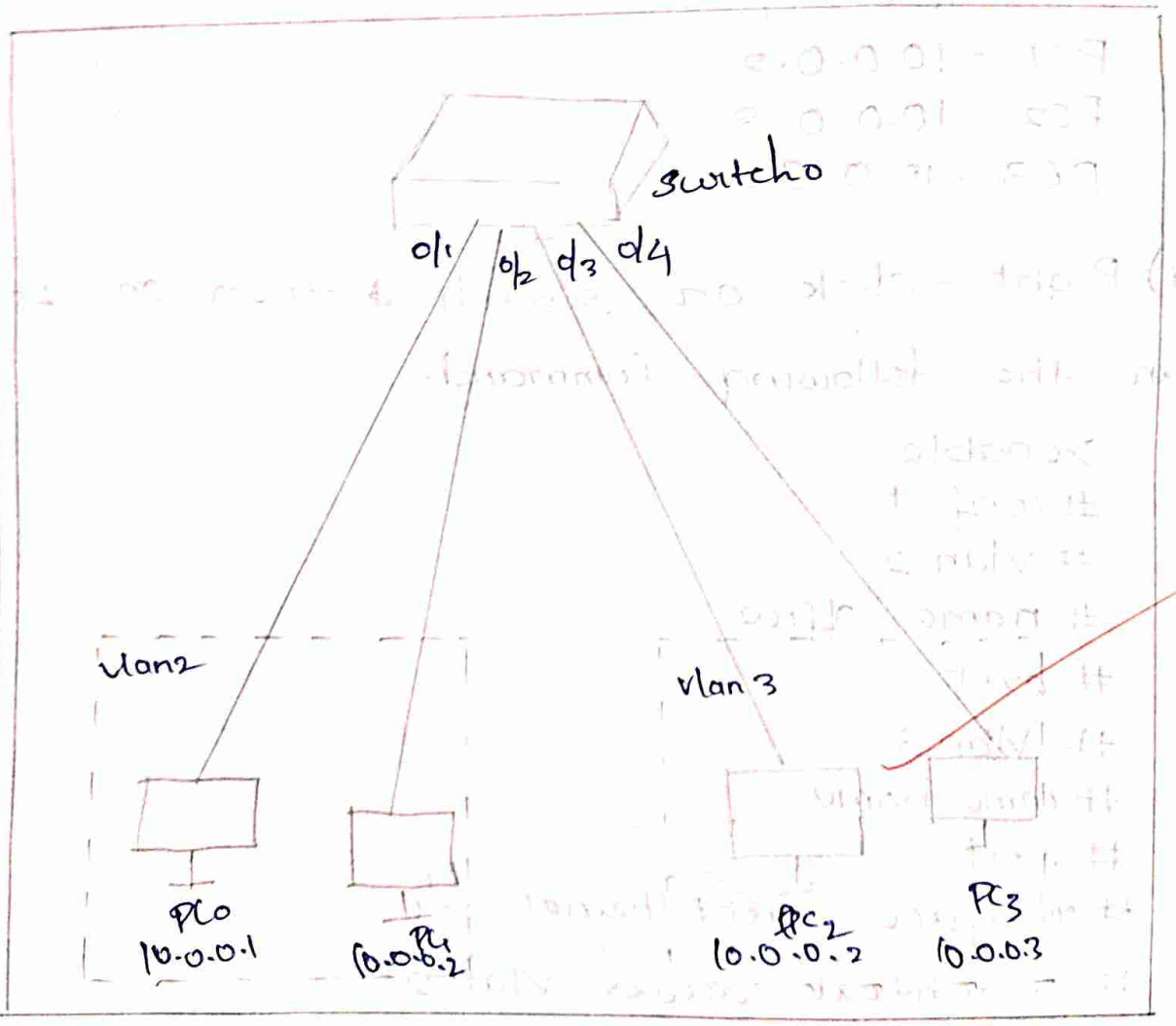
4) Right - click on switch & then on CLI run the following command.

```
>enable
# conf t
# vlan 2
# name office
# exit
# Vlan 3
# name home
# exit
# interface fastEthernet 0/1
# switchPort access Vlan 2
# exit.
# interface fastEthernet 0/1
# switchPort access Vlan 2
# exit
```

#interface fastEthernet 0/2
#switchPort access vlan 2
#exit
#interface fastethernet 0/3
#switchPort access vlan 3
#exit
#interface fastethernet 0/4
#switchPort access vlan 4
#exit

5) Now try pinging packet from PCs to one one's another.

Diagramatic Representation.



Observation

When packet transferred from PC0 to PC1 the packet are successfully transferred similarly for PC2 within same VLAN packets are transferred successfully

Whereas for PC0 can't transfer packets out of the VLAN

Output:

Fire	Last Status	Source	Destination	Type	Color	Time	Periodic	Num
	Successful	PC0	PC1	ICMP		0.000	N	0
	failed	PC1	PC2	ICMP		0.000	N	1

Result:-

Thus the Virtual LAN has been stimulated using ciscopacket tracer.

9/4/24

Exp. No. 8b
7/10/24

Wireless LAN

Aim:-

To configure wireless LAN using cisco packet tracer.

Procedure:-

1. Create network 3PCs & a wireless Router

2. Rightclick on the wireless router
Note the IP address it would be 192.168.0.1

→ Disable the DHCP Server
→ save settings

3. Click on the wireless on the same Page

→ Change the network name to "MyHome Network

→ Save setting

→ Click on wireless security under wireless & change security mode to WEP from disabled

→ Add key1: 0123456789

→ Save settings

→ then close the window.

4. Now assign IP to the PC

PC0 - 192.168.0.5

->gateway 192.168.0.1

PC1 - 192.168.0.6

->gateway 192.168.0.1

PC2 - 192.168.0.7

->gateway - 192.168.0.1

5. Now click on PC0

-> click Physical option

-> Turn off that

-> Drag the Part in the diagram

-> to the left menu

-> Drag the 'WMP300N' to the Past from the left

-> Turn on that

-> Now click on desktop -> PC wireless

-> click connect -> site information -> connect

-> Type '0123456789' in the WEP key

-> Connect

6. Repeat steps for the PC1 + PC2

#interface fastEthernet 0/2

#SwitchPort access vlan 2

#exit

interface fastEthernet 0/3

#SwitchPort accessvlan 3

#exit

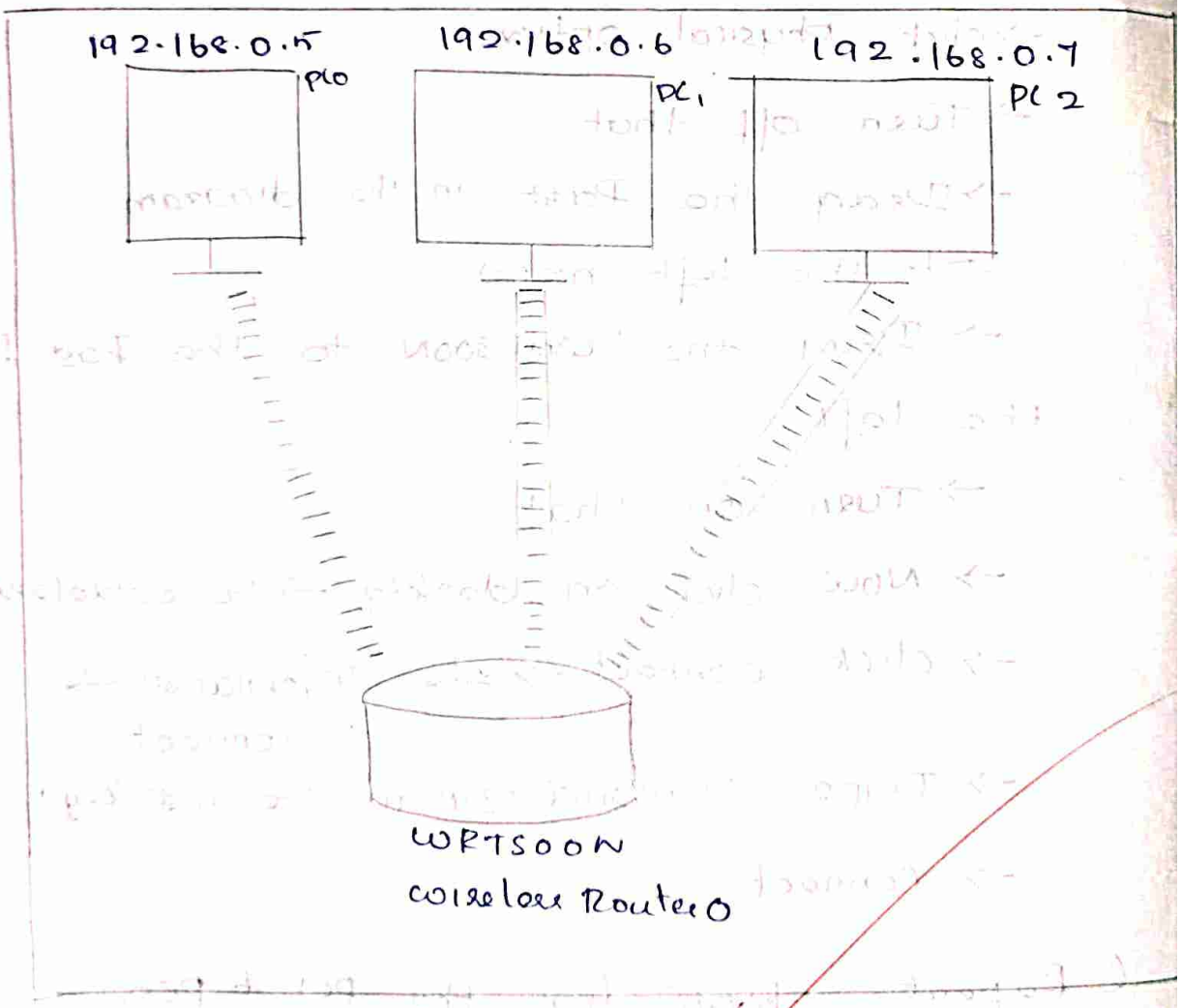
#Interface fastEthernet 0/4

switchPort access Vlan 3

exit

5) Now try pinging Packet from PC. to one another.

Diagramatic Representation



Output:-

Thus the packets could be transmitted via wireless router.

time	last status	Source	destination	type	color	[illegible]	[illegible]
0	successful	PC0	PC2	ICMP	[illegible]	"	0
0	successful	PC1	PC3	ICMP	[illegible]	"	1

Simulation Panel

Event list

VIS	Time (sec)	Last Device
	0.000	--
	0.001	PC0
	0.003	--
	0.004	wireless router 0
	0.004	wireless router 0
	0.004	wireless router 0
	0.005	--
	0.006	PC2
	0.008	--
	0.009	wireless router 0
	0.009	wireless router 0
Visible	0.012	--

Student observation.

a) What is SSID of wireless Router?

Ans:- Service set identifier is the name assigned to a wireless network, allowing devices to identify and connect to the correct wifi. the SSID is broadcast by the router and helps distinguish one network from other in same area.

b) What is Security key in wireless router

Ans:-

A Security key is a password or code to protect a wireless network ensuring only authorised users can connect. Common types are WEP, WPA etc..

Result:-

Thus wireless Lan has been configured successfully using cisco packet tracer

9/14

• PC 3

IP : 192.168.1.14

Gateway : 192.168.1.1

• PC4

IP: 192.168.1.15

Gateway : 192.168.1.1

→ Switch S_2

* NO : IP

→ LAN - 2

• PC 5

IP address : 192.168.2.11

Gateway : 192.168.2.1

• PC_6

IP address : 192.168.2.12

Gateway : 192.168.2.1

• PC7

IP address : 192.168.2.13

Gateway : 192.168.2.1

• PC_8

IP address : 192.168.2.14

Gateway : 192.168.2.1

• PC_9

IP address : 192.168.2.15

Gateway : 192.168.2.1

Router 2

gig 0/0 - 201.1.2

0/1 - 172.1.1.1

0/2 - 200.1.1.2

Ex. 9
10/10/24

Classfull Subnetting

Aim:-

implementation of subnetting in cisco packet tracer ~~test~~ simulation

Procedure:

1) Create the network using switches router and PCs

2) The IP address will be follows

→ Router R1

* Gigabit Ethernet 0/0 : 192.168.1.1
* Gigabit Ethernet 0/1 : 192.168.2.1

Also enable the 'on' option for both gigaethernet on the Page.

→ Switch S1

* no IP

→ LAN - 1

- PC 0

  IP Address : 192.168.1.11

  Gateway : 192.168.1.1

- PC 1

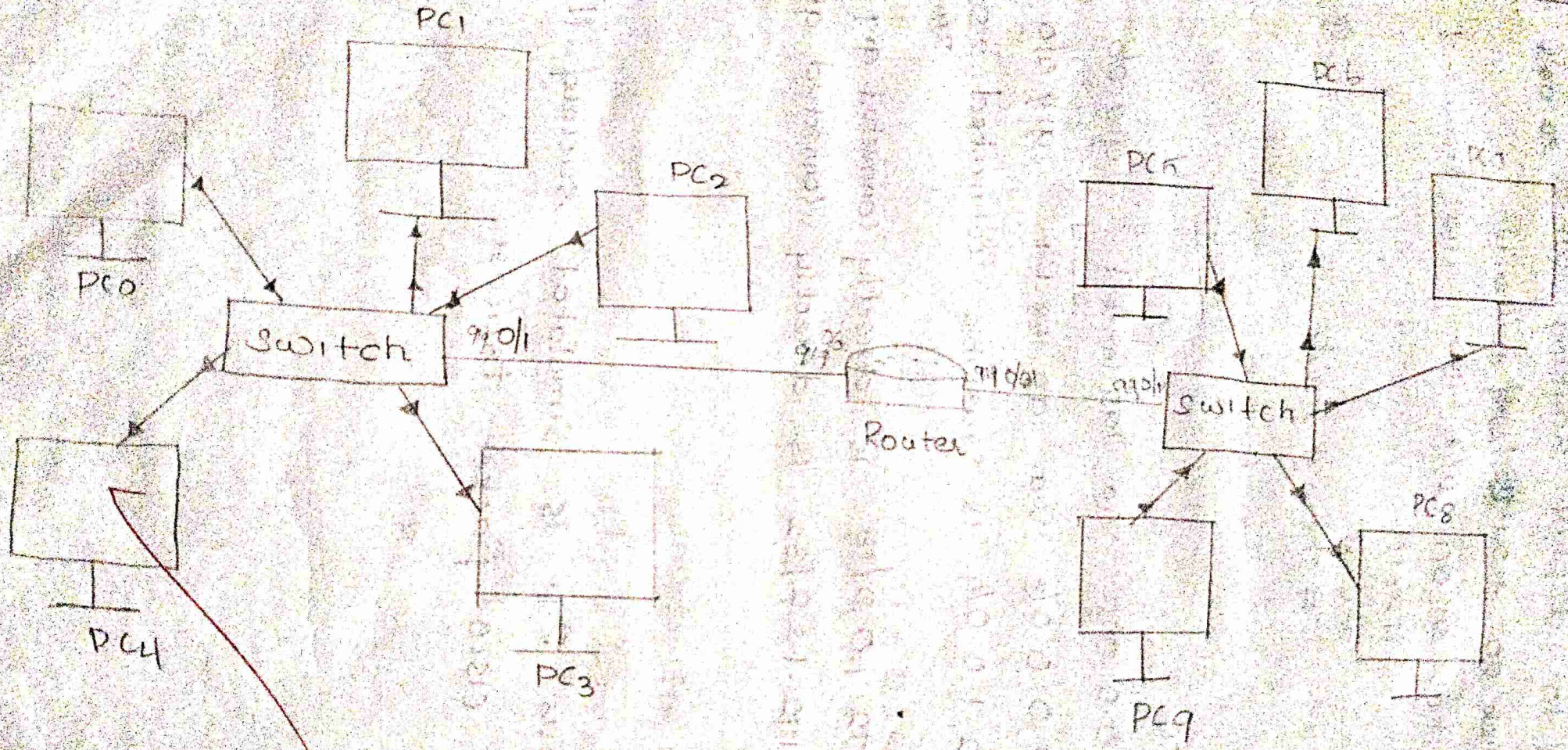
  IP : 192.168.1.2

  Gateway : 192.168.1.1

- PC 2

  IP : 192.168.1.13

  Gateway : 192.168.1.1


PC1
PC2
PC0
Switch
Router
PC4
PC3
PC6
PC7
Switch
PC8
PC9

Output

Now let assume the sender is PC1 and receiver is PC9

while simulating I observing we get the simulation panel

Simulation Panel

Time	Last Device
0.000	--
0.003	PC1
0.005	Switch1
0.008	Router1
0.010	Rswitch2
0.013	PC9
0.015	Switch2
0.018	Router1
0.020	Switch1
0.023	--

Result Simulation ☐ Constant

Play controls

Fire	Last Status	Source	Destination	Type	Color	Time	Periodic	Num
●	Successful	PC1	PC9	ICMP		0.000	N	0

Student observation:-

a) write down your understanding of subnetting

Ans:-

Subnetting is the process of dividing a large IP network and manageable smaller called subnets. Each subnet act as independent network & allow divides connected to it to communicate within the subnet & control traffic btw subnet.

b) Advantages of implementing subnetting within a network

Ans:-

↳ efficient IP management - based on requirement

↳ Reduce network congestion - limit broadcast traffic to individual subnet

Result:-

The implementation of subnetting in Cisco has been done successfully.

Exp: 10

Date 16/10/24

Router

Aim:-

To design and configure an internet work using wireless router DHCP server & internet cloud.

Procedure:-

1. Arrange the PC + laptop in page
2. Assign IP address

Server - 10.0.0.1

↳ Service → DHCP → default gateway 10.0.0

→ Server - ON.

PC0

- desktop - DHCP ⊙

Laptop 0

- desktop - DHCP ⊙

PC1

- desktop - dHCP ⊙

Laptop 1

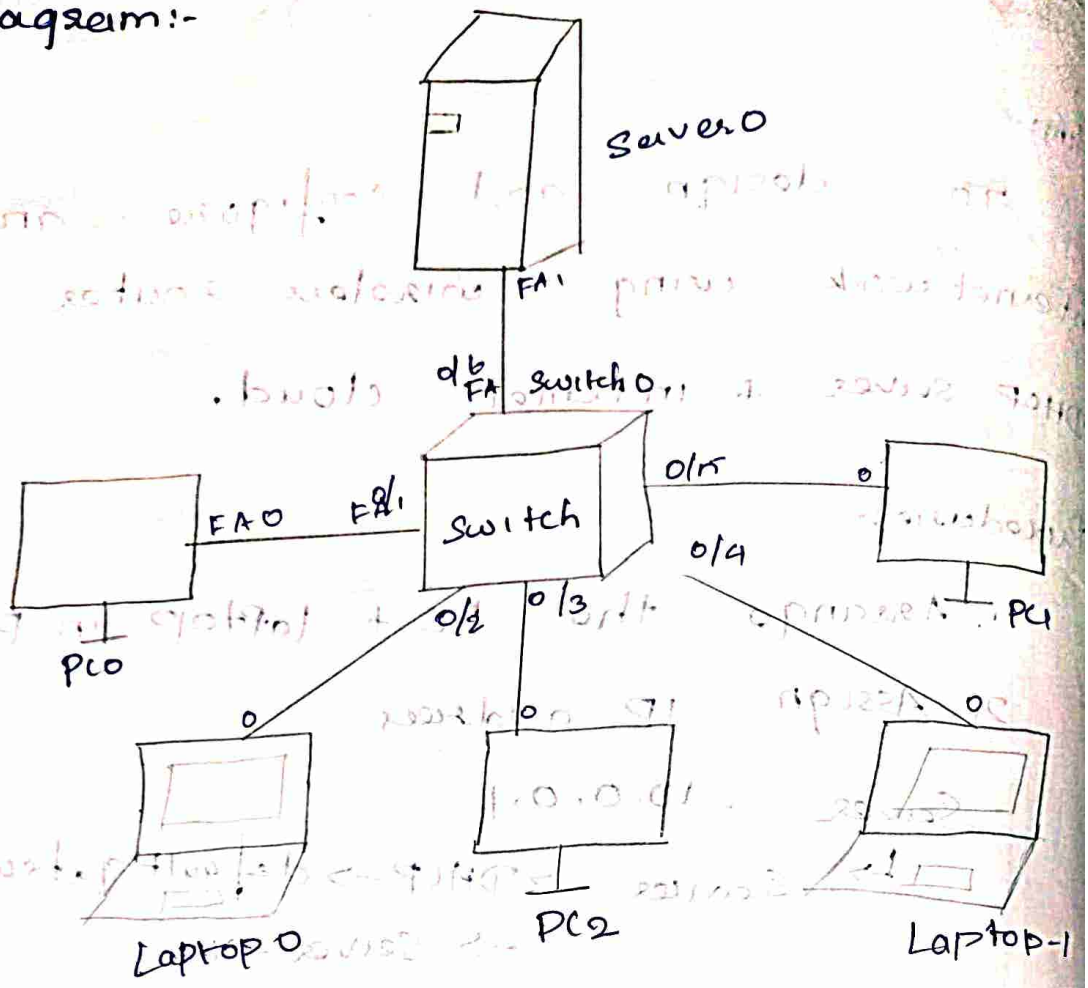
- desktop - dHCP ⊙

PC2

- desktop - dHCP ⊙

Now Ping from PCs to server

Diagram:-



Output

Ping laptop to PC2

Event list		
V I B	Time (sec)	Last Device
	0.000	--
	0.001	Laptop 0
	0.002	switch 0
	0.003	PC2
	0.004	~~Sec~~
visible		

Result:-

Thus the configuration of DHCP has been executed.

Practical 11

22/10/24

Aim:-

(a) Simulate Static Routing configuration using Cisco Packet Tracer

(b) Simulate RIP using Cisco Packet tracer

(a) 1. Adding static Router: each router knows only networks directly connected to it add static route to reach a network not directly connected.

Eg: Router, network 10.0.0.0/8, 20.0.0.0/8 & 40.0.0.0/8 are directly connected but 30.0.0.0/8 & 50.0.0.0/8

2. Creating main & Backup Router

Administrative Distance device preference of router the lower the AD, the higher preference.

3. Router Configuration

Configure static route on each router for networks not directly connected

4. Verifying Router:-

Verify route by using command show IP route static,

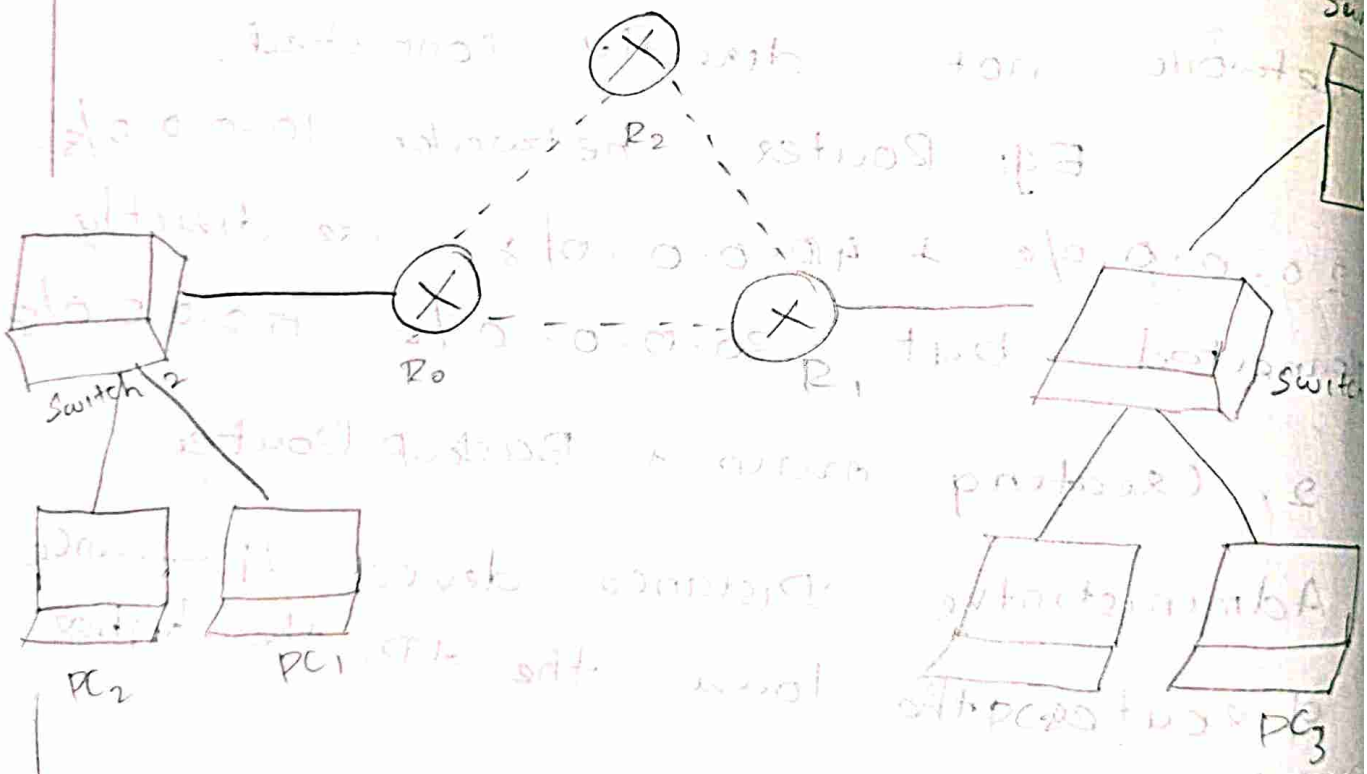
5) Testing Route Fail over.

→ Test connectivity using ping from a device on a connected network

↳ disconnect or "break" the link of the main route.

6. deleting a static route.

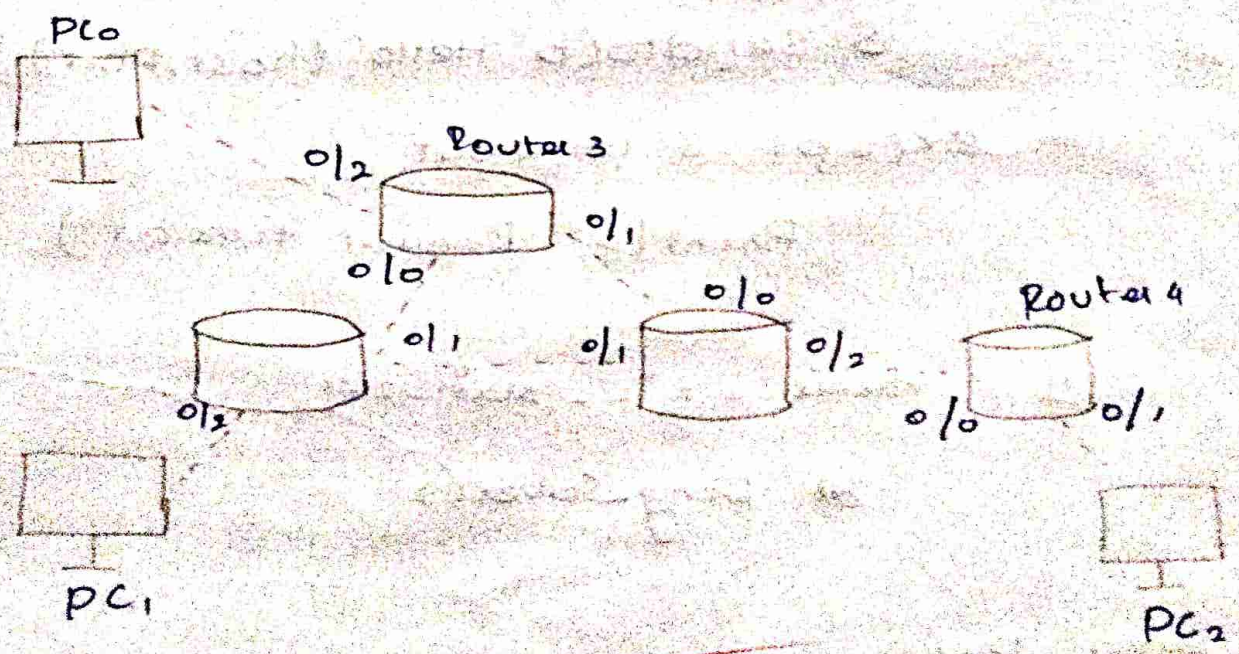
show ip route static



b)

1. Initial IP configuration for devices
2. Assign IP address to device for PCs and routers
3. enable, configure interface on routers
4. Configure RIP on routers
5. Verify and test redundancy
   - use ping command on PC1
   - use tracer to see RIP redirecting traffic through an alternative route

Diagramatic representation



Result:-

Thus RIP is simulated using cisco Packet Tracer successfully.

Practical - 12

22/10/24

Aim:

Implementing echo client server using Socket

Client.py

```
import socket
import time
def ping_server(host="127.0.0.1" port="1234")
    with socket.socket(socket.AF_INET,
                       socket.SOCK_DGRAM) as s
    try:
        s.sendto(b"Hello", (host,Port))
    except s.timeout:
        Print(f" Request time out")

if __name__ = "__main__"
    Ping_Server()
```

Server.py

```
import socket
def start_server(host="127.0.0.1", Port=
    with socket.socket = (socket.AF_IN
                          Socket.SOCK_DGRAM
    S.bind(
```

```
Print (f" UDP running on port {host}:{port}")
while True:
    data, addr = S.recv.from (1024)
    Print (f" Received message")
if __name__ = "__Main__"
    Start. Server()
```

O/P :

```
Python .receiver.py
    UDP Server running on 127.0.0.1 12345
    Received message from (127.0.0.1, 59290
Python client. py
    Received reply from Server : Hello. client
```

b) Implement chat client Server using TCP/UDP

```
Chat server. py
import socket
def receiver():
    Port = 12345
    host = '127.0.0.1'
with Socket. socket ('socket: AF_INET,
                    sock - PROGRAM) as S:
    S. bind ((host, port))
while (True):
    d. add - S. recvfrom ('1024')
```

```
Print ("client")
a=input(" Enter reply")
s.sendto (a.encoded.add)
if (a=="end")
    break
    exit
recvfl()
```

receiver2.py

```
import socket
import time
def receiver (a):
    host ='127.0.0.1'

while (true) :
    a=input (" Enter message")
    if (a=="end"):
        receiver(0)
        break
    else :

        receive(a)
```

o/p:

```
Python \chat-serv.py
client {'hi'}
Client {'How are you'}
Enter reply ##fere
```

Python

./recv.py

Enter message hi

{' hallo '}

Enter message how are you

{" I am fine "}

Enter message

Output

Server.py

>Python Server.py

UDP server running on 127.0.0.1:12345

Client.Py

>Python client.Py

Message sent

Server terminal (127.0.0.1' 56003): Hello.

Result:

Thus program of echo client server using UDP socket has been implemented executed successfully

Exp.13
4/4/24

Aim: To implement your own ping Program

Algorithm:

Create UDP socket it in a specific address

wait for a message

Print message + client's address

Send back pong to client

code:

Server.py:

```
import socket

def start_server (host='127.0.0.1' port=123

    with Socket.socket = (Socket.AF_INET,
                Socket.SOCK_DGRAM) as s:

        S.binder((host, port)).

Printf("UDP running on ~~port~~ ~~{addr}~~
            {host}:{port}")

while True
```

```
data, addr = s.recvfrom(1024)
print(f"Received message from
        {addr} : {data.decode()}")
s.sendto(b'Pong', addr)

if __name__ == "__main__"
start_server()
```

Client.Py

```
import socket
import time
def Ping_server(host = '127.0.0.1',
                Port = '12345')
    with socket.socket(socket.AF_INET,
        socket.SOCK_DGRAM) as s

        s.settimeout = 5

        try
            s.settime(2)
            start = time.time()
            s.sendto(b'Ping'),(host,Port))
            data, add = s.recvfrom(1024)
            end = time.time()
```

```
Print(f"Received {data.decode()} from
        {addr} in {end-start:.4f}
except Socket.~~sock~~timeout:
    Print(f"request timeout")

if __name__ = "__main__"
    Ping-Server()
```

Output Terminal:

```
Python Server.Py
    UDP Server running on 127.0.0.1:12
~~Ping~~client.Py
    Received Pong from (127.0.0.1,
Server.Py
    Received message from 127.0.0.1
    50001 : Ping
```

Result:

Thus the Program has been executed Successfully.

Exp 01
Sniffer

Aim:

To write code using PAU Scapy to implement packet sniffing

Algorithm:

Start sniffing, capture packets on specified network

check packet type for each packet

check if IP in packet

identify protocol between the Protocol

Display
report

Code:

```
Packet sniff.py
from Scapy.all import sniff
from Scapy.layers.inet import IP, TCP, UDP
def packet_callback(packet):
    if IP in packet:[IP]
```

```
Protocol = IP_layer.Proto
Src.IP = IP_layer.Src
dst.IP = IP_layer.dst

Protocol name = " "
If protocol == '1' :
    Protocol name = "IP"

elif Protocol == "6" :
    Protocol name = "TCP"

elif Protocol == "17" :
    Protocol name = "UPD"

else
    Protocol name = " ICMP"

Printf" Protocol : {Protocol~~name~~}")
Printf" ~~Protocol~~ Source : { Src.IP}")
Printf " dst : {ds.IP}")

def main()
    Sniff(iface = wlf0  prn = ~~protocol~~ Packet_callbac
          filter=IP , store=0 )
If __name__ = "__main__"
    Main()
```

Output

Protocol : TCP

Source IP : 151.130.199.105

Destination IP : 193.168.94.193

Result:

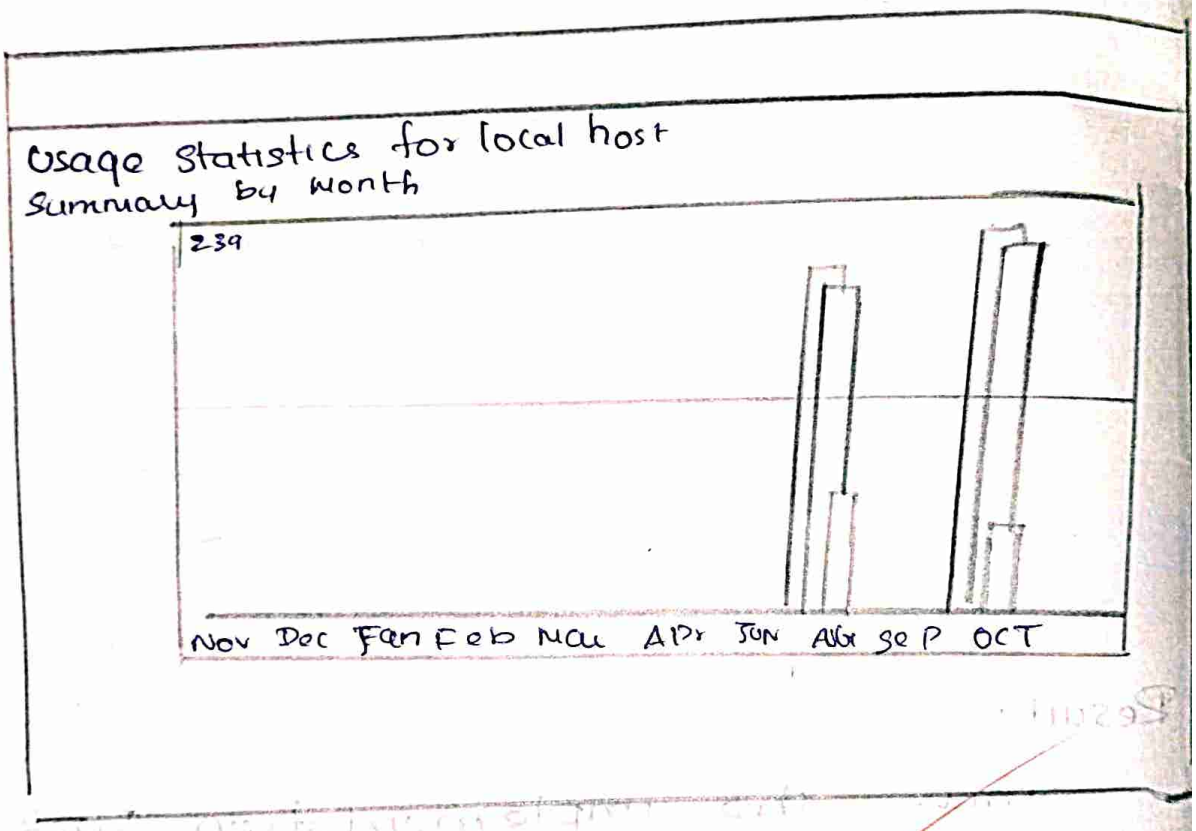
Thus the implementation Packet Sniff is done using Raw Sockets.

19/12

Exp. 15
10/11/24 webalizer tool

Aim

To analyze the different types of log using webalizer tool



Result:-

Thus the different types of weblogs has been analysed using webaliza tool.

19/11