

PRACTICAL - 1.

Date: 16.07.2024 (A)

AIM :- Study of various Network commands used in Linux and Windows.

BASIC NETWORKING COMMANDS:WINDOWS COMMAND:

1) arp -a

Interface: 192.168.100.1 ---0xd

Internet Address	Physical Address	Type
192.168.100.254	00-50-56-fc-56-7b	dynamic
172.168.8.1	7c-5a-1c-cf-3b-e-45	dynamic
224.0.0.2	01-00-5e-00-00-02	static
239.255.255.250	01-00-50-7f-ff-fa	static

2) Hostname

DESKTOP-HCVOANO

3) ipconfig /all

Windows IP configuration

Host Name : DESKTOP-HCVOANO

Primary Dns Suffix :

Node Type : Mixed

TFTP Routing Enabled : No

WINS Proxy Enabled : No

Ethernet adapter Ethernet 2

connection-specific DNS suffix :

Description : Realtek PCIe GBE Family Controller

DNS Servers : 172.16.8.1

NetBIOS over Tcpip : Enabled

Default gateway [d] [F1] [Delete] [Alt-W]

4) nbtstat -a

~~nbtstat [-a RemoteName] [-A IP address] [-c] [-n]~~
NBTSTAT [-a RemoteName] [-A IP address] [-c] [-n]
 [-r] [-R] [-RR] [-S] [-S] [interval]

RemoteName Remote host machine name
 IP address Dotted decimal representation of IP add.
 interval Redisplays selected statistics, pausing
 interval sec b/w each display.

5) netstat

Active Connections			
Proto	Local Add	Foreign Add	state
TCP	172.16.8.85:7680	172.16.8.179:55342	ESTABLISHED
TCP	172.16.8.85:7680	172.16.71.52:38881	TIME-WAIT
TCP	172.16.8.85:62716	123:http	TIME-WAIT
TCP	172.16.8.85:62734	172.16.11.105:ms-do	SYN-SENT

6) nslookup

nslookup www.google.com

Server : Unknown

Address : 172.16.8.1

Non-authoritative answers:

Name : www.google.com

Addresses: 2404:6800:4007:81e: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~~

8) Ping

Ping www.rajalakshmi.org

Pinging www.rajalakshmi.org [14.99.10.232] with 32 bytes of data:

Reply from 14.99.10.232: bytes=32 time<1ms TTL=127

Reply from 14.99.10.232: bytes=32 time=1ms TTL=127

Ping statistics for 14.99.10.232:

packets: sent=4, received=4, lost=0 (0% loss),
max=0ms, min=0ms, Avg=0ms

9) Route

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

[MASK netmask] [gateway] [METRIC metric] [IF inter-

Command one of these:

PRINT Prints a route

ADD Adds a route

DELETE Deletes a route

CHANGE Modifies an existing route

LINUX COMMANDS:

1) arp -a

gateway (172.16.8.1) at 7c:5a:1c:cf:be:45 [ether] on
enp2s0

2) Hostname

local host, localdomain (172.16.8.1) has been set

172.16.8.1 is listed in /etc/hosts (172.16.8.1) has been set

3) ifconfig

enp2s0 : flags=4163 <UP,BROADCAST,RUNNING,MULTICAST>
inet brdcast 192.168.1.255 mtu 1500
inet6 fe80::41c:2ff%enp2s0 brd fe80::ff:fe1c:2ff%enp2s0 mtu 1280

lo : flags=73 <UP,LOOPBACK,RUNNING> mtu 65536

wlp3s0 : flags=4099 <UP,BROADCAST,MULTICAST> mtu 1500

4) nmblookup -A <ip address>

nmblookup -A 14.99.10.232

Looking up status of 14.99.10.232

WORKGROUP <00> - <GROUP> B <ACTIVE>

DESKTDP - B@498VC <00> - B <ACTIVE>

MAC Address = 50-9A-4C-34-D3-C3

5) nslookup www.google.com

Server: 172.16.8.1#53

Address: 172.16.8.1#53

Non-authorized answer:

Name: www.google.com

Address: 142.250.183.228

6) Ping

i) Ping localhost

PING localhost (localhost (:)) 56 data bytes

64 bytes from localhost (:) : icmp-seq=1 ttl=64 time=0.07ms

ii) Ping 4.2.2.2

PING 4.2.2.2 (4.2.2.2) 56(84) bytes of data.

64 bytes from 4.2.2.2: icmp-seq=1 ttl=53 time=25.2 ms

iii) Ping www.facebook.com
 PING start-mini.clor.facebook.com (157.240.192.35) 56(84)
 bytes of data.
 by bytes from edge-star-mini-shv-02-maa2.facebook.com
 (157.240.192.35); ICMP_seq=1 ttl=59
 time=2.79ms

7) Route

Kernel IP Routing Table							
destination	Gateway	Genmask	Flags	Metric	Ref	Use	
default	gateway	0.0.0.0	UG	100	0	0	
172.16.8.0	0.0.0.0	255.255.252.0	U	100	0	0	

Some important Linux networking commands.

1. ip

ip <options> <object> <command>

a) # ip address show

1: lo: <LOOPBACK, UP, LOWER_UP> mtu 65536

inet 127.0.0.1/8 scope host

valid_lft forever

inet6 ::1/128

2: enp2s0: <BROADCAST, MULTICAST, UP, LOWER_UP> mtu 1500

link/ether 50:9a:4c:34:d8:85 brd

3: wlp3s0: <BROADCAST, MULTICAST, UP, LOWER_UP> mtu 1500

link/ether d4:6a:6a:82:ca:fb brd

b) # ip address add 192.168.1.254/24 dev enp8s03

c) # ip address del 192.168.1.254/24 dev enp8s03

d) # ip link set eth0 up

e) # ip link set eth0 down

f) # ip link set eth0 promisc on

- g) #ip route add default via 192.168.1.254
- h) #ip route add 192.168.1.0/24 via 192.168.1.254
- i) #ip route delete 192.168.1.0/24 via 192.168.1.254
- j) #ip route add 192.168.1.0/24 dev eth0
- k) #ip route get 10.10.1.4

10.10.1.4 via 172.16.8.1 dev ens2s0 src 172.16.8.84

origin cache

2. ifconfig

ens2s0 flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
mtu 1500

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536

wlp3s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

3. mtr

mtr <options> host/re/1P

a) # mtr google.com

Host	Packets	Pings					
		Loss%	Snt	Avg	Best	Worst	StDev
172.16.8.1	160	0.2	0.2	0.2	0.2	0.3	0.0
142.250.171.161	181	0.9	3.6	2.6	43.8	4.2	

b) # mtr -g google.com

c) # mtr -b google.com

d) # mtr -c 3 google.com

e) # mtr -l google.com

d) `tcpdump`

`tcpdump`: 287 packets captured
(kernel) 1033 packets received by filter
740 packets dropped by kernel.

a) `# dnf install -y tcpdump`

Last metadata expiration check: 2:50:40 ago on
Tue 23 Jul 2024 08:23:12 AM IST.
Package `tcpdump-14:4.9.0-2.fc26.1686` is already
installed, skipping.

Dependencies resolved.
Nothing to do.
Complete!

b) `# tcpdump -D`

1. `enp2s0` [Up, Running; Loopback]
2. any (Pseudo-device that captures on all interfaces)
3. `lo` [Up, Running, Loopback]
4. `wlp3s0` [Up]

c) `# tcpdump -i eth0` [`# tcpdump -i enp2s0`]

`tcpdump: eth0: NO device`

[`tcpdump: verbose output suppressed, use -v for full
listening on enp2s0, link-type EN10MB (Ethernet), capture
11:31:24.517943 IP 172.16.9.164. 51018 > 239.255.255.250.
11:31:24.518748 IP localhost.localdomain.localdomain.4515
18 packets captured, 328 packets received, 328 dropped,
18 packets lost, 0% loss.`]

d) `# tcpdump -i enp2s0 -c 4`

~~Listening on enp2s0, link-type EN10MB (Ethernet)~~

~~11:36:56.347974 IP 172.16.9.46.mdns > 224.0.0.251.mdns~~

~~4 packets captured~~

~~252 packets received by filter~~

~~243 packets dropped by kernel.~~

e) #tcpdump -i enp2so -c 4 host 8.8.8.8

tcpdump: verbose output suppressed, use -v
listening on enp2so, link-type EN10MB (Ethernet)
0 packets captured

0 packets received by filter
0 packets dropped by kernel.

f) #tcpdump -i enp2so src host 8.8.8.8 -c 20

tcpdump: verbose output suppressed, use -v (or -vv
for more)
0 packets captured
0 packets received by filter
0 packets dropped by kernel

nmcli connection show

Name	Type	Device
wired connection	ethernet	enp2s0

nmcli connection add con-name enp0s2 type
ethernet

connection. enp0s2 (640679c-6702-4761-af63-
0480901aeb74d) script-shelved as present

nmcli connection modify "wired connection"

ipv4.method auto. ipv6.method ignore

nmcli connection modify "wired connection"

ipv6.method auto

ip address show enp0s2

2: enp0s2: <BROADCAST, MULTICAST, PROMISE, UP, LOWER_UP>
mtu 1500 qdisc mq-codel state UP qlen 1000
link/ether 08:00:27:1f:01:65 brd ff:ff:ff:ff:ff:ff

inet brd 192.168.1.0 mask 255.255.255.0 state UP
inet6 fe80::1f01:65ff:fe00:27%enp0s2 brd fe80::ff:fe00:27/64 state
inet6 :: brd :: state UNKNOWN

ip route show default

default via 192.168.137.1 dev enp0s2 proto dhcp src
192.168.137.92 metric 1000

cat /etc/resolv.conf

nameserver 127.0.0.53

options ndots:0 trust-ad

search mshome.net

~~Ques 16/24~~ (912) first believe habibis - 8

Student Observation:

1. which command is used to find the reachability of a host machine from your device?
Ans: * Ping cmd.
2. which command will give the details of hops taken by a packet to reach its destination?
Ans: * mtr (Mrt's of traceroute)
3. Which command displays the IP config of your machine?
Ans: * IP & Options > object > command
4. Which command displays the TCP port status in your machine?
Ans: * netstat
5. Write the modify ip config in a Linux machine.

* address add 192.168.1.254/24 dev enp0s3
* ip address del 192.168.1.279/99 dev enp0s4

Ques 17/24 (912) first believe habibis - 8

RESULT:

Thus networking commands of both Linux & windows are studied and executed successfully

PRACTICE - 2

~~Topics and concepts are 1. LAN, 2. LAN via Switches
3. LAN via Routers~~
AIM:- Study of different types of Network cables.

Different type of cables of network:

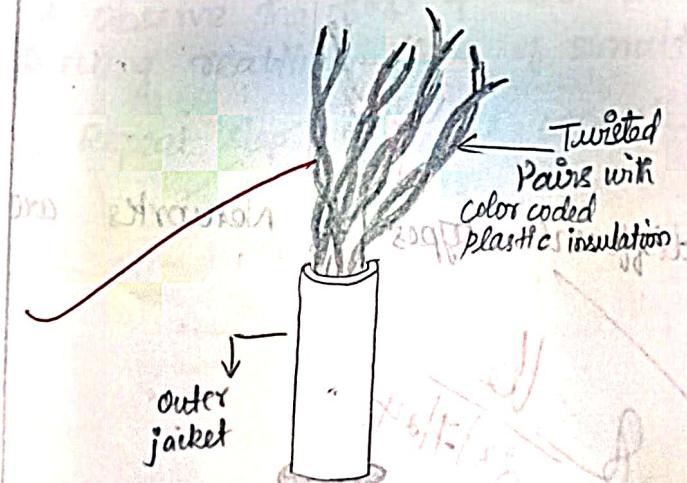
1. Unshielded Twisted Pair (UTP) cable
2. Shielded Twisted Pair (STP) cable
3. Coaxial cable
4. Fibre Optic cable.

Cable Type	Category	Maximum Data Transmission	Advantages / Disadvantages	Application / Use
UTP	Category 3	10 bps	<u>Advantages</u>	10Base-T Ethernet
	Category 5	Up to 100 Mbps	<ul style="list-style-type: none"> cheapest in cost Easy to install as they have a smaller overall diameter. 	Fast Ethernet, Gigabit Ethernet
	Category 5e	1 Gbps	<u>Disadvantages</u>	Fast Ethernet, Gigabit Ethernet
STP	Category 6, 6a	10 Gbps	<u>Advantages</u> <ul style="list-style-type: none"> shielded Faster than UTP less susceptible to noise and interference 	Gigabit Ethernet, 10G Ethernet (50m)
SSTP	Category 7	10Gbps	<u>Disadvantages</u> <ul style="list-style-type: none"> Expensive Greater installation effort 	widely used in data centres. Gigabit Ethernet, 10G Ethernet (100 m)

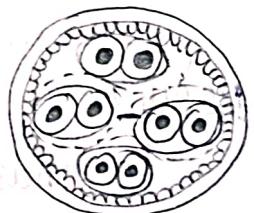
Category	Data Transmission	Advantages / Disadvantages	Application / Use
Coaxial cable	RG-6 RG-59 RG-11	10-100 Mbps	<u>Advantages</u> <ul style="list-style-type: none"> High bandwidth Immune to interference Low loss bandwidth versatile <u>Disadvantages</u> <ul style="list-style-type: none"> Limited distance cost Size is bulky
Fibre optics cable	Single mode Multi-mode	100 Gbps	<u>Advantages</u> <ul style="list-style-type: none"> High speeds High bandwidth High security long distance <u>Disadvantages</u> <ul style="list-style-type: none"> Expensive Requires skilled installers <p>• Maximum distance of fibre optics cable is around 100 metre</p>

Images:

a) UTP



STP



SSTP

PRACTICAL - 3

AIM: To Study the packet tracer tool Installation and User Interface Overview

1. From the network component box, click and drag-and drop the below components:

- 4 Generic PCs and One Hub
- 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 cable. The link LED should glow in green, indicating that the link is up.

c. Similarly connect 4 PCs to the switch using copper straight-through cable.

3. Click and PC's connected to hub, go to desktop tab, click IP Configuration and enter IP address.

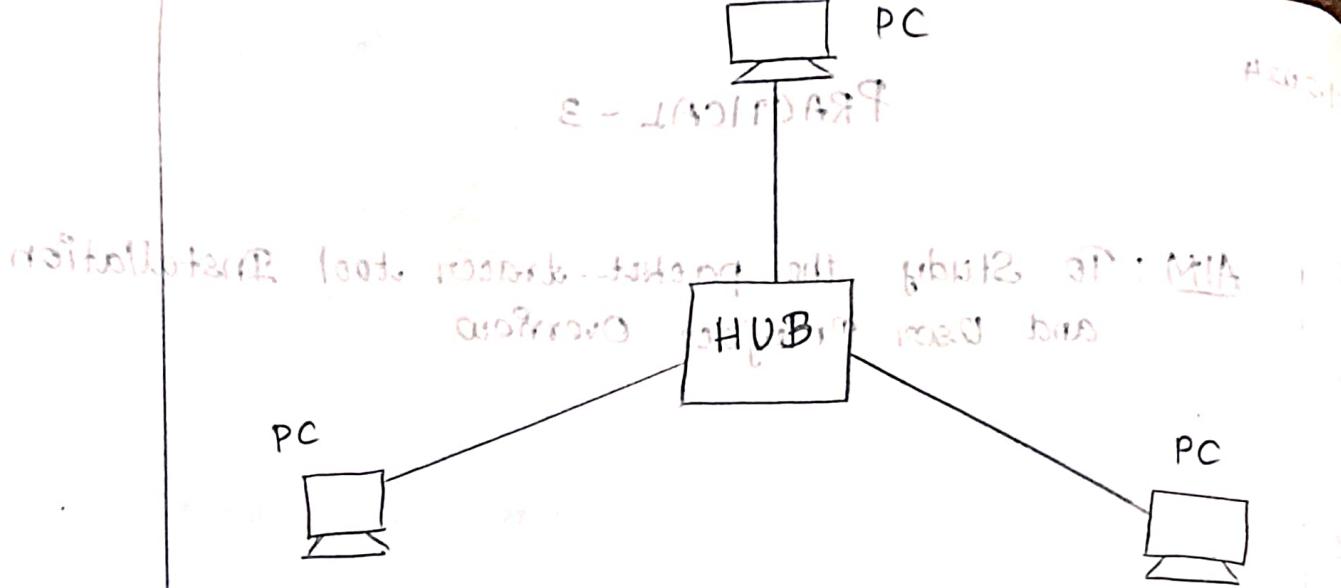
click message icon from common tool bar,

for a) Drag and Drop it once on PC and drop it on another PC connected to HUB.

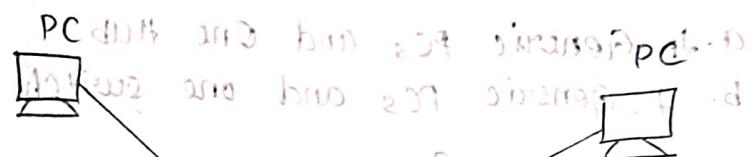
4. Observe the flow of PDU from sender to receiver by selecting realtime mode of simulation.

5. Repeat step 3 to 5 PCs connected to Switch.

and select it in realtime e.g. 1000 frames ATM protocol with header size 40 bytes. At 1000 bytes payload. equal frame transmission time. with all setting properly. repeat step and send message of random - 1000 bytes protocol.



- bus-topology:
 * shared transmission downstream cells remain in the bus
 * each computer selects its own



* collisions on the bus are common

• after going to the bus from the computer, the bus goes to the other computer and so on. If one computer sends data, then all other computers receive it. This is called broadcast. Broadcast is a way of sending data to all computers in a network.

switches do not share the bus. They have their own dedicated paths.

Observation:
 Switches forward unicast frames only.

a) Behaviour of switch and Hub in terms of forwarding the packets received by them

Switch:-
 A network switch is a device that connects multiple devices within a LAN and routes data between them using MAC addresses. It operates at the data layer, directing packets to the correct destination and supporting full-duplex for efficient data transfer.

HUB:-

A Hub is a basic network device that connects multiple devices in a LAN. It operates at physical layer of OSI model and broadcasts incoming data to all connected devices, regardless of the destination.

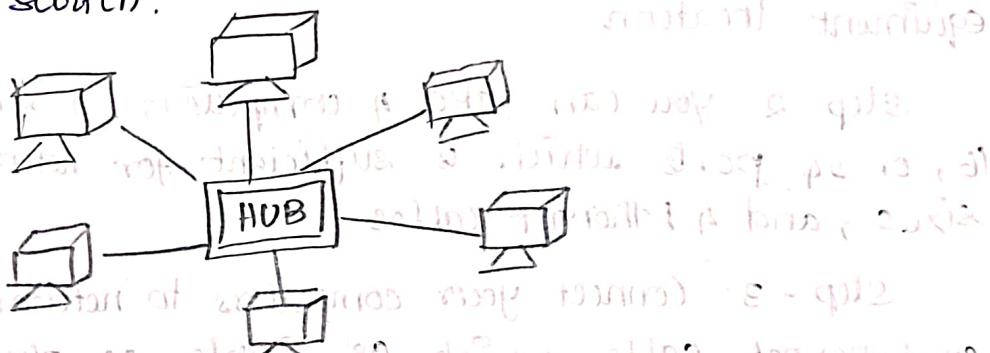
- b) Network topology implementation in your college and draw and label that topology is your observation book.

* Topology implemented in our college is star and hybrid topology.

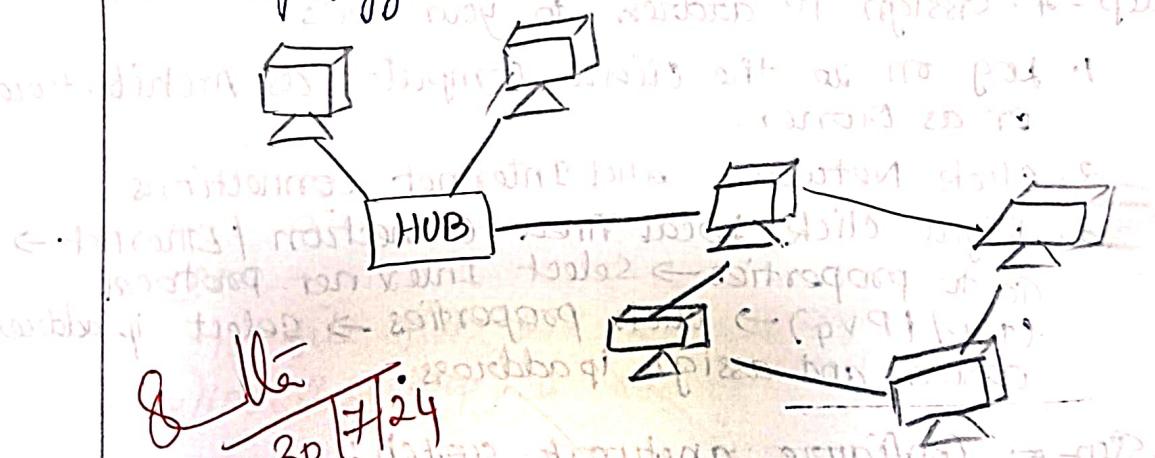
characteristics of Star

→ all cable run to a central connection point

→ as network grows (or) changes, computer are simple added (or) removed from central connection pt. hub (or) switch.



→ Hybrid topology is combination of more than one topology



RESULT:-

thus successfully studied the packet tracer tool and analysed the behaviour of network.

6.8.24

PRACTICAL - 4

AIM : Setup and configure a LAN (Local area network) using a Switch and Ethernet cables in your lab.

What is a LAN?

A Local Area Network (LAN) refers to a network that connects devices within a limited area, such as office building, school, or home. It enables users to share resources, including data, printers, and internet access.

How to set up a LAN

Step - 1 : Plan and Design an appropriate network topology taking into account network requirements and equipment location.

Step - 2 : 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.

Step - 3 : Connect your computers to network switch via an Ethernet cable, which as simple as plugging one end of the Ethernet cable into your computer and other end into your network switch.

Step - 4 : Assign IP address to your PCs

1. Log on to the client computer as Admin or as Owner.
2. Click Network and Internet Connections.
3. Right click Local Area Connection / Ethernet → Go to properties → Select Internet Protocol (TCP / IPv4) → click properties → select ip address option and assign ip address.

Step - 5 : Configure a network switch :

1. Connect your computer to the switch.
2. Log in to the web interface.
3. Configure basic settings.
4. Assign IP Address as . with Subnet mask

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

Step - 7 : Select a folder, → go to properties → click sharing shortcut → share it with everyone on the same LAN.

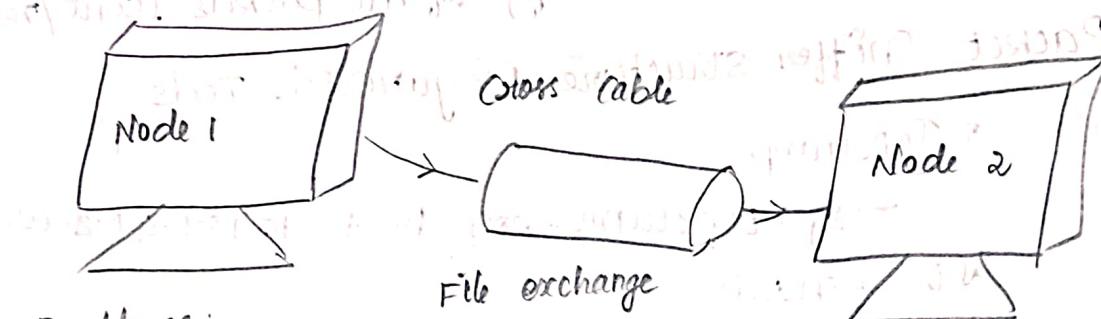
Step - 8 : Try to access the shared folder from others computers of the network.

Observation:

Draw a diagram of LAN in the configuration observation book that you have implemented in your lab. Write the IP configuration of each & every device. Write the outcome and challenge faced while configuring the LAN.

Diagram of the LAN:

(Chassis) and mounting the PCB in chassis



IP address:

172.16.8.82

IP configuration:

Node - 1 :

IPV4 Address: 172.16.8.82

Node - 2 :

IPV4 address : 172.16.8.83

IP address:

172.16.8.83

Outcome:

The file that sent that through [click Run: C:\1172.16.8.82\userB\REC\Desktop] is sent successfully.

Q: Yes
6/8/24

RESULT :

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

9-8-24

PRACTICAL - 5

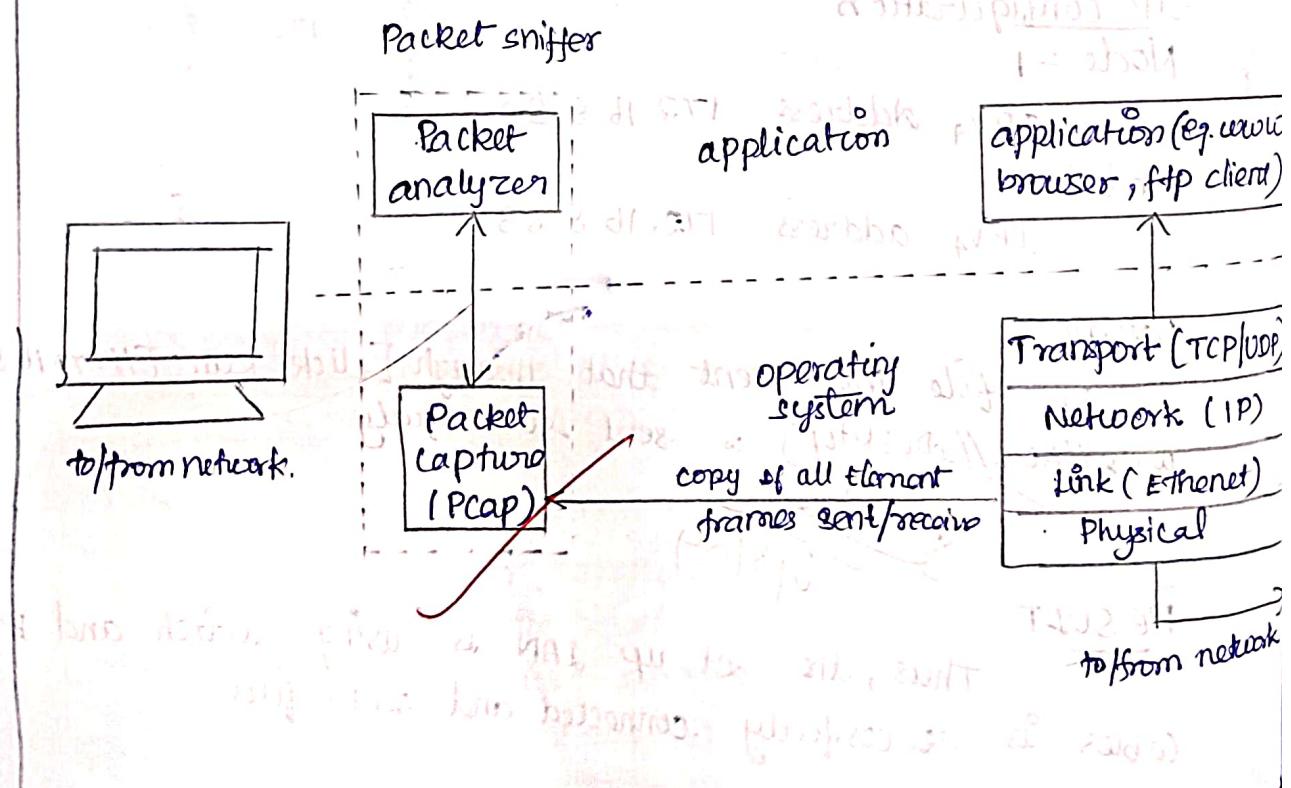
AIM: Experiments on Packet capture tool: Wireshark

Packet Sniffer :

- * Sniffs messages being sent/received from/by your computer.
 - * Store and display the contents of the various protocol 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
 - ~~if~~: tcpdump -en interface
 - * Wireshark
 - wireshark -r file.pcap



Whreshark

* Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and displays that in human-relatable format.

what we can do with Wireshark?

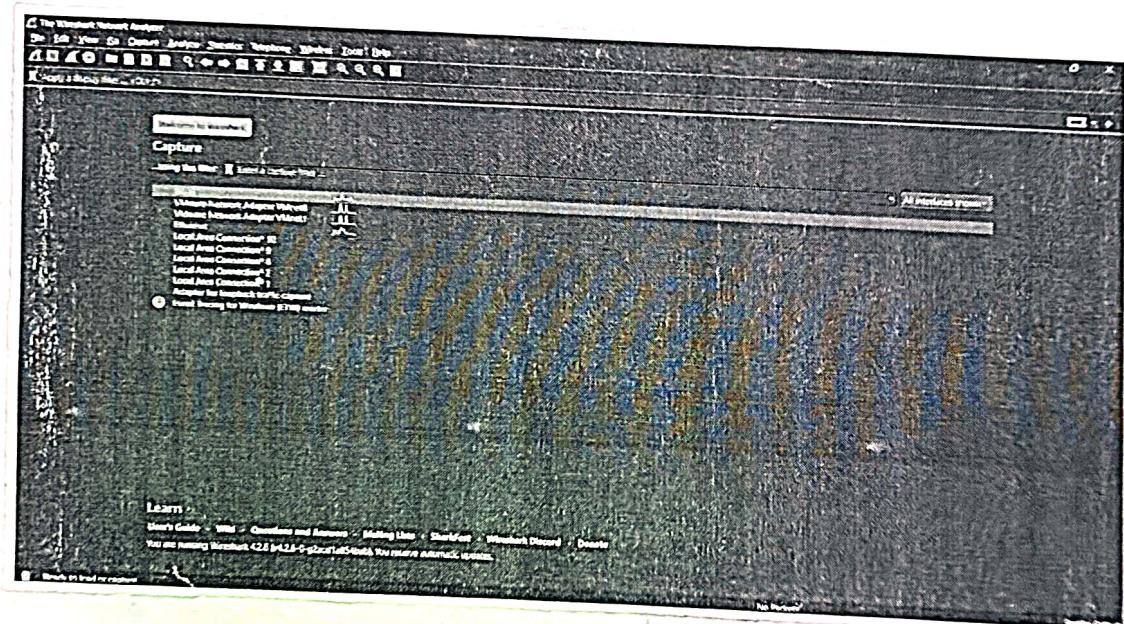
- * Capture network traffic
 - * Decode packet protocols using dissectors.

Wireshark used for:

- * Network administrators: troubleshoot network problems
 - * Network security engineers: examine security problems

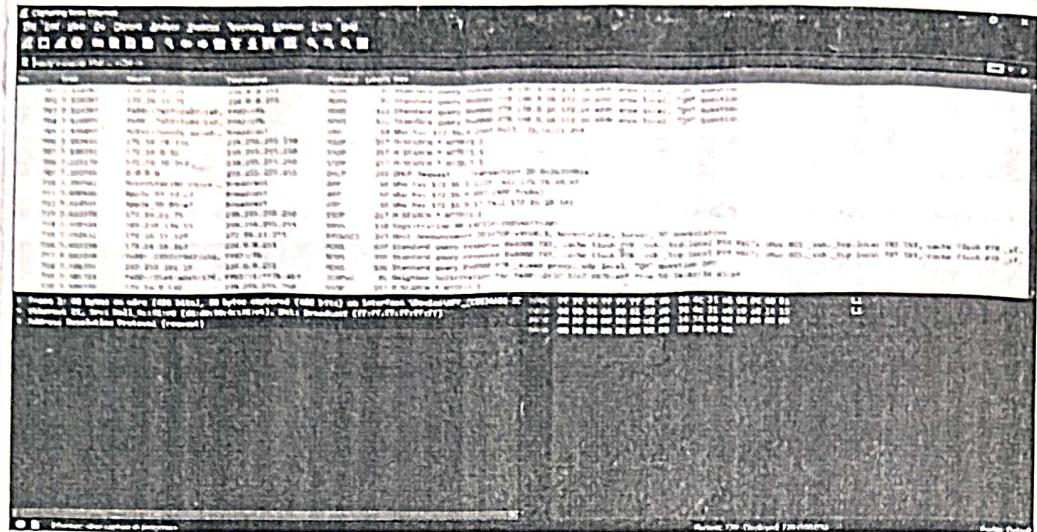
Capturing Packets.

* After downloading and installing, launch it and double-click the name of a network interface under Capture to start capturing packets on that interface.



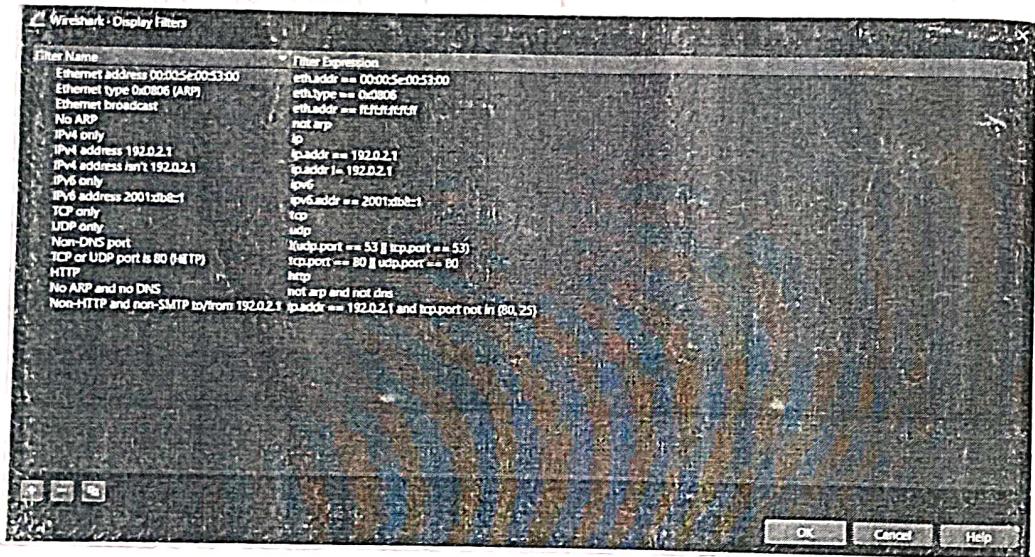
Color Coding

* Wireshark uses color to help you identify the type of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors.



Filtering Packets

* the basic way to apply a filter is by typing it into the filter box at the top of the window and clicking apply.



Inspecting Packets

* click a packet to select it and you can dig down to view its details.

dig down to see the details of the selected packet.



Packet and port in network work.

Student Observation:

1. What is promiscuous mode?
 → It refers to setting that allows the network interface card to capture all network traffic on the segment it is connected.
 2. Does ARP packets has transport layer header? Explain.
 → It does not have a Transport layer header. They operate at the Data Link layer (Layer 2) and are used for mapping IP address to MAC address.
 3. Which transport layer protocol is used DNS?
 → Uses UDP as its transport layer protocol on port 53.
 4. What is the port number used by HTTP protocol?
 → HTTP protocol uses port number 80 for communication over the web.
 5. What is a broadcast IP address?
 → It typically 255.255.255.255 is used to send a packet to all device in network segment.
- ~~Q. What is the result of experiment?~~
- ~~RESULT:-~~
- Thus, Wireshark tool has been experiments on packet captured and studied.

Practical - 6

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

Error correction at Data Link Layer:

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

Create Sender program with below functions.

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

Create a receiver program with below features.

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

Code

```

def to_binary(char):
    return format(ord(char), '08b')

def calculate_redundant_bits(m):
    r = m - m // 2
    r = redundant_bits(m)
    r = r + 1
    return r

r = 0
while (1 << r) < (m+r+1):
    r += 1
return r

def insert_redundant_bits(data, m, r):
    encoded = ['0'] * (m+r)
    j = 0
    for i in range(1, m+r+1):
        if (i & (i-1)) == 0:
            continue
        encoded[i-1] = data[j]
        j += 1
    for i in range(r):
        pos = (1 << i)
        xor_sum = 0
        for j in range(pos-1, m+r, pos*2):
            for k in range(pos*j, min(j+pos, m+r)):
                xor_sum ^= int(encoded[k])
        encoded[pos-1] = str(xor_sum)
    return ''.join(encoded)

def apply_hamming_code(text):
    binary_data = ''.join([to_binary(char) for char
                           in text])
    print("Binary representation: " + binary_data)
    m = len(binary_data)
    r = calculate_redundant_bits(m)
    encoded_data = insert_redundant_bits(binary_data, m, r)
    print("Encoded data with redundant bits: " + encoded_data)

```

Printf "Redundant bit positions : { $i \ll i$ for i in range(r)}

return encoded_data, r

def check_and_correct(encoded_data, r):

m = len(encoded_data) - r

error_position = 0

for i in range(r):

pos = ($i \ll i$)

xor_sum = 0

for j in range(pos - 1, len(encoded_data), pos + 2):

for k in range(j, min(j + pos, len(encoded_data))):

if encoded_data[k] != int(encoded_data):

error_position |= (xor_sum << i)

if error_position:

Print("Error found at position : " + str(error_position))

corrected_data = list(encoded_data)

corrected_data[error_position - 1] = '0' if .

encoded_data[error_position - 1] ==
else '1'

Print("Corrected data : " + ''.join(corrected_data))

else:

Print("No error found.")

corrected_data = list(encoded_data)

return ''.join(corrected_data), error_position

def extract_original_data(corrected_data, r):

m = len(corrected_data) - r

data_bits = []

for i in range(1, m + 1):

if (i & (i - 1)) == 0:

data_bits.append(corrected_data[i - 1])

binary_str = ''.join(data_bits)

return ''.join([chr(int(binary_str[i:i + 8], 2)) for
range(0, len(binary_str), 8)])

```
def main():
    text = input("Enter the text : ")
    encoded_data = apply_hamming_code(text)

    change = input("Do you want to change any bit in encoded-data : ")
    if change == 'Yes':
        position = int(input("Enter the bit position to change (1 - len(encoded-data)): "))

        if 1 <= position <= len(encoded_data):
            is_redundant = position in [1 < i < 10 for i in range(len(encoded_data))]
            encoded_data = list(encoded_data)
            encoded_data[position - 1] = '0' if encoded_data[position - 1] == '1' else '1'
            encoded_data = ''.join(encoded_data)
            print(f"Data after bit change, {encoded_data}")

            if is_redundant:
                print(f"Note: you changed a redundant bit at Position {position}.")
            else:
                print("Invalid position. No changes made.")

            corrected_data, error_position = check_and_correct(encoded_data, r)
            if change == 'Yes' and is_redundant and error_position == position:
                print("The change in the redundant bit was corrected successfully.")
            elif change == 'Yes' and is_redundant and error_position != position:
                print(f"The change in the redundant bit was not detected")
                original_text = extract_original_data(corrected_data, r)
                print(f"Decoded text: {original_text}")
        else:
            print("Position is out of bounds")

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

Output : Enter the text : k
Binary representation : 01101011
Redundant bit of Encoded data : 100111011011
Redundant bit positions : [1, 2, 4, 8]
Do you want to change bit in encoded data? : Yes

Enter the bit position (1-12) : 11

Error found at position : 11
(corrected data) : 100111011011

Decoded text : k.

Program finished

Code - bit stuffing, no error detected

Error detected - no error detected

Program finished

No redundant bits in the binary

(Error detected)

Column register has a value of 0000

Transmission starts - without errors, data detected

(corrected data)

no error detected - no errors found

(corrected data)

RESULT :

20/8/29. The program of Hamming Code has been successfully executed and the output is verified.

6.9.24

Practical - 7

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

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

Create a sender program with following features :-

1. Input window size from the user.
2. Input a text message from the user.
3. Consider 1 character per frame.
4. Create a frame with following field [Frame no, DATA]
5. Send the frames.
6. Wait for the ACK from the Receiver.
7. Read a file called Receiver-Buffer.
8. Check ACK field for the ACK number.
9. If the ACK number is as expected, send new set of frames accordingly.

Create a receiver file with following features.

C - Create a file called sender-Buffer.

1. Reader a file called sender-Buffer.
2. Check the Frame No.
3. 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.

(stubsommit) bspapp.cwmp

{ static const P::tns2 "t" tning }

"1 = 4 on_error . fls2

Student Observation:

Code :

```
import time
```

```
import threading
```

```
import os
```

```
class SlidingWindowProtocol:
```

```
    def __init__(self, window_size, message):
```

```
        self.window_size = window_size
```

```
        self.message = message
```

```
        self.frame_no = 0
```

```
        self.expected_ack_no = 0
```

```
        self.expected_frame_no = 0
```

```
        self.expected = None
```

```
        self.sender_buffer = "sender_Buffer.txt"
```

```
        self.receiver_buffer = "Receiving_Buffer.txt"
```

```
        self.sender_done = False
```

```
        self.receiver_done = False
```

```
def sender(self):
```

```
    while not self.sender_done:
```

```
        frames = []
```

```
        print(f"\n---SENDING FRAMES(window size: {self.window_size})---")
```

```
        for i in range(self.window_size):
```

```
            if self.frame_no < len(self.message):
```

~~```
 frame_data = f"Frame NO: {self.frame_no}, DATA: {self.message[self.frame_no]}
```~~~~```
                frames.append(frame_data)
```~~~~```
 print(f"sent: {frame_data}")
```~~~~```
                self.frame_no += 1
```~~

```

if frames:
    with open(self.sender_buffer, "w") as f:
        f.write(frames + "\n")
if self.frame_no >= len(self.message):
    self.sender_done = True
    print("All frames sent")
    print("Waiting for ACK...")
    time.sleep(2)
    self.wait_for_ack()
    ("def wait_for_ack(self):"
     ("try:" "file = open(receiver_buffer, "r") as f:")
     ("with open(self.receiver_buffer, "r") as f:")
     ("ack_data = f.readlines()")
     ("except FileNotFoundError:")
     ("print(f'Receiver buffer file {self.receiver_buffer} not found.'")
     ("return")
     ("for ack in ack_data:")
     ("    ack_type, ack_no = ack.strip().split(':')")
     ("    ack_no = int(ack_no)")
     ("    if ack_type == "ACK":"
     ("        if ack_no == self.expected_ack_no + 1:"
     ("            print(f'ACK {ack_no} received. Sending next frames.')
            self.frame_expected_ack_no = ack_no
        else:
            print(f'Unexpected ACK received. Expected {self.expected_ack_no + 1}, got {ack_no}. Resending frames.')
    else:
        print(f'NACK {ack_no} received. Resending frames.')
        self.frame_no -= self.window_size")
    def receiver(self):
        while not self.receiver_done:
            time.sleep(2)

```

```

if not os.path.exists(self.sender_buffer):
    print("Sender buffer " + self.sender_buffer + " does not exist")
    continue
with open(self.sender_buffer, 'r') as f:
    frames = f.readlines()
    if not frames:
        continue
    print("In -- RECEIVING FRAMES ---")
    ack_to_send = [0] * len(frames)
    for frame in frames:
        frame_no, data = frame.strip().split(',')
        frame_no = int(frame_no.split(':')[1])
        data = data.split(':')[1]
        with open(self.receiver_buffer, 'w') as f:
            f.write("\n".join(ack_to_send) + "\n")
        if self.expected_frame_no > len(self.message):
            print("All frames received. Stopping receiver.")
            time.sleep(2)
        if __name__ == "main":
            window_size = int(input("Enter window size"))
            message = input("Enter text Message")
            Protocol = SlidingWindowProtocol(window_size, message)
            sender_thread = threading.Thread(target=Protocol.send)
            receiver_thread = threading.Thread(target=Protocol.receive)
            sender_thread.start()
            receiver_thread.start()
            sender_thread.join()
            receiver_thread.join()
            print("\nTransmission Completed")

```

Output: P - 41511CA9E

18-07-1

Enter window size : 4

Enter Text message: cute

-- SENDING FRAMES (window size: 4) --

sent: [Frame No: 0, Data: c]

sent: [Frame No: 1, Data: u]

sent: [Frame No: 2, Data: t]

sent: [Frame No: 3, Data: e]

All frames sent.

Waiting for ACK...

Unexpected ACK received. Expected 1, got 4. Resending ...

-- RECEIVING FRAMES --

Received Frame No: 0, DATA: c]

Received Frame No: 1, DATA: u]

Received Frame No: 2, DATA: t]

Received Frame No: 3, DATA: e]

All frames received. Stopping receiver.

Transmission completed.

sender-Buffer Receiver-Buffer

[Frame No: 0, DATA: c]

ACK: 1

[Frame No: 1, DATA: u]

ACK: 2

[Frame No: 2, DATA: t]

ACK: 3

[Frame No: 3, DATA: e]

ACK: 4

RESULT:

The Program has been successfully executed
and the output is verified.

Classfull Subnetting

AIM: Implementation of subnetting in Cisco
Packet tracer

PROCEDURE:

1) Create the network using switches, router and PCs

2) The IP address will be as 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 giga Ethernet on the page.

→ Switch S1

* no IP assigned

→ LAN - 1

- PC 0

IP Address : 192.168.1.11

Gateway : 192.168.1.1

- PC 1

IP : 192.168.1.12

Gateway : 192.168.1.1

- PC 2

IP : 192.168.1.13

Gateway : 192.168.1.1

→ Switch S2

* NO IP

→ LAN - 2

- PC 5

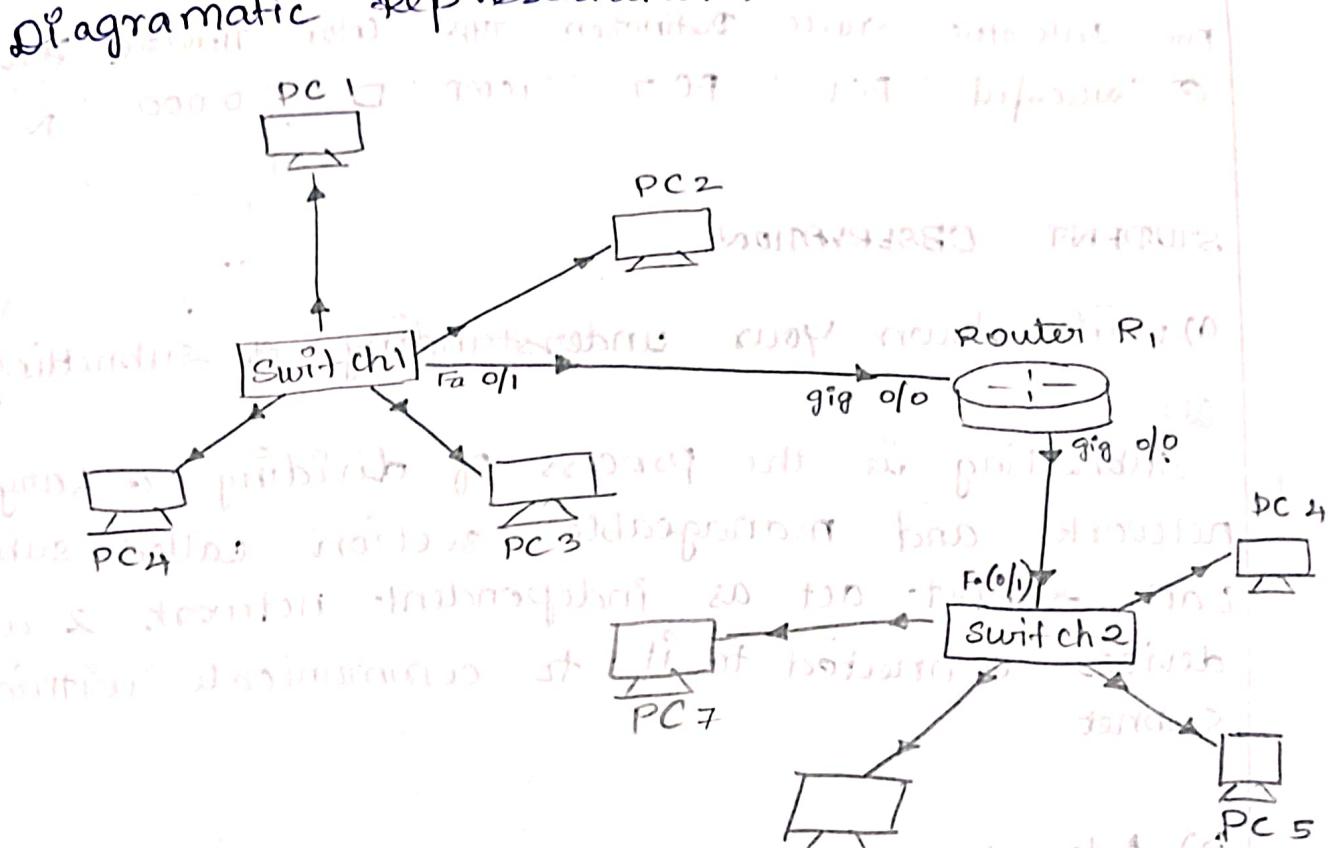
IP address : 192.168.2.11

Gateway : 192.168.2.1

- PC 6

IP : 192.168.2.12

Gateway : 192.168.2.1



OUTPUT:

Now lets assume the switch sender is PC1 and receiver is PC6 while simulating & observing we get the simulation panel.

| Simulation Panel | | |
|------------------|-------|-------------|
| Event List | | |
| VIS | time | Last device |
| | 0.000 | -- |
| | 0.003 | PC1 |
| | 0.005 | switch 1 |
| | 0.008 | Router 1 |
| | 0.010 | switch 2 |
| | 0.013 | PC7 |
| | 0.015 | switch 2 |
| | 0.018 | Router 1 |
| | 0.020 | switch |
| | 0.023 | -- |

Reset Simulation constant delay

Play controls

[Stop] [Next] [Run]

| Time | Last status | Source | Destination | Type | Color | Time(sec) | Periodic num. |
|------|-------------|--------|-------------|------|-------|-----------|---------------|
| ② | successful | PC 1 | PC 7 | ICMP | □ | 0.000 | N |

STUDENT OBSERVATION

a) write down your understanding of subnetting

Ans:-

Subnetting is the process of dividing a large IP network into manageable sections called subnets. Each subnet acts as an independent network & allows devices connected to it to communicate within the subnet.

b) Advantages of implementing subnetting within a network.

Ans:- Efficient IP management based on requirement of organization.

- ↳ efficient IP management - based on requirement of organization.
- ↳ Reduce Network congestion by limiting broadcast traffic to individual subnets.

RESULTS:

The implementation of subnetting in Cisco has been done successfully.

a) VIRTUAL LAN

AIM: simulate virtual configuration using CISCO Packet Tracer simulation.

PROCEDURE :

- 1) Create the network using switch & 4 PC as shown figure.

- 2) Assign IP address to the PCs

PC 0 - 10.0.0.1

PC 1 - 10.0.0.2

PC 2 - 10.0.0.3

PC 3 - 10.0.0.4

- 3) Check the network by sending packets.

- 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 0/2

#switchport access vlan 2

#exit

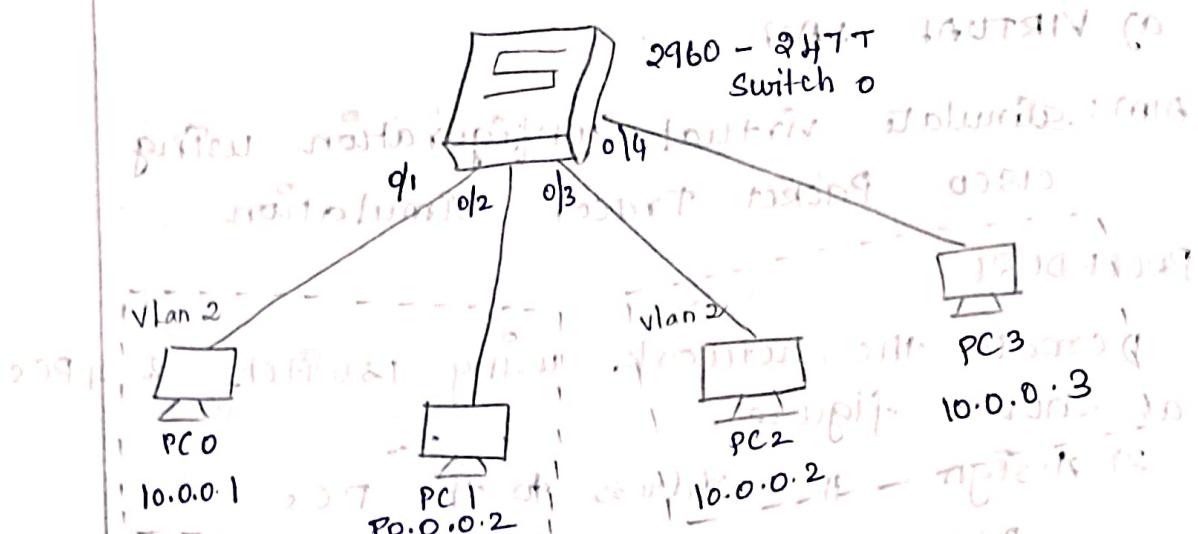
#interface fastEthernet 0/3

#switchport access vlan 3

#exit

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

Diagrammatic Representation :-



Observation :

when packet transferred from - PC0 to PC1 the packets are successfully transferred. Similarly for PCs within same VLAN packet are transferred successfully.

whereas for PCs can't transfer packets out of its VLAN.

Output :

| fire | laststatus | source | destination | type | colour | time | periodic | Num |
|------|------------|--------|-------------|------|--------|-------|----------|-----|
| ① | Successful | PC0 | PC1 | ICMP | □ | 0.000 | N | 0 |
| ② | failed | PC1 | PC2 | ICMP | □ | 0.655 | N | 1 |

RESULT :

Thus the virtual LAN has been simulated using Cisco packet tracer.

Wireless LAN

AIM: To configure wireless LAN using cisco packet tracer.

PROCEDURE:

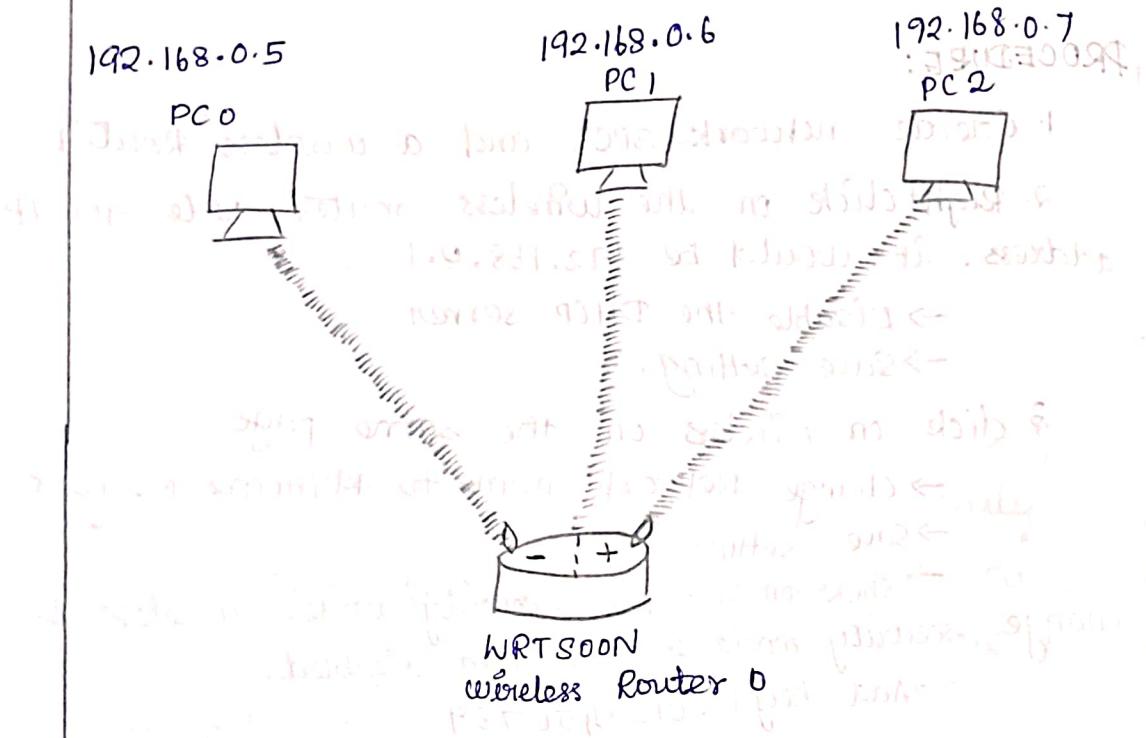
1. Create network, 3 PCs and a wireless Router.
2. Right click on the wireless router. Note the IP address. It would be 192.168.0.1
 - Disable the DHCP server
 - Save setting.
3. Click on wireless on the same page
 - change Network name to MYHome Network
 - Save setting
 - click on wireless security under wireless & change security mode to WEP from disabled.
 - Add key 1 : 0123456789
 - save settings
 - then close the window
4. Now assign IP to the PCs
 - 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 port in the diagram to left menu
 - Drag the 'WMP300N' to the port from the bg
 - Turn on that
 - Now click on desktop → PC wireless
 - click connect → site info → connect.

→ Type '0123456789' in the WEP key,
→ Connect.

6. Repeat step 5 for the PC1 and PC2

Then you can see the below connection in the network.
Diagrammatic Representation:



Output:

Thus the packets could be transmitted via wireless router.

| fire | last status | source | destination | Type | colour | time | period | Nurr |
|------|-------------|--------|-------------|------|--------|-------|--------|------|
| ○ | Successful | PC0 | PC2 | ICMP | □ | 0.000 | N | 0 |
| ○ | Successful | PC1 | PC0 | ICMP | □ | 0.066 | N | 1 |

| Simulation Panel | | |
|------------------|------------|-------------------|
| Event List | | |
| v8 | Time (sec) | Last Device |
| | 0.000 | -- |
| | 0.001 | PC0 |
| | 0.003 | -- |
| | 0.004 | wireless router 0 |
| | 0.005 | -- |
| | 0.006 | PC2 |
| | 0.008 | -- |
| | 0.009 | wireless router 0 |
| viseble | 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 the same area.

b) What is security key in wireless router.

Ans:-

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

Result:

thus wireless LAN has been configured successfully using cisco packet tracer.

Jitendra

AIM : To simulate RIP using Cisco packet tracer.

PROCEDURE :

1) Create network using 3 PCs & 4 Router as shown in image.

2) Design IP address for the PCs & router port

PC 0

IP - 10.1.1.1

Gateway - 10.1.1.2

PC 1

IP - 200.1.1.1

Gateway - 200.1.1.2

PC 2

IP - 222.2.2.2

Gateway - 222.2.2.12

Router 3

gig 0/0 : 20.1.1.1

0/1 : 192.168.1.1

0/2 : 10.1.1.1

Router 2

gig 0/0 - 20.1.1.2

0/1 - 172.1.1.1

0/2 - 200.1.1.2

Router 1

gig 0/0 - 192.168.1.3

0/1 - 172.1.1.2

0/2 - 217.1.1.1

Router 4

gig 0/0 - 217.1.1.2

0/1 - 222.2.2.12

3. click on router 3

→ click config → RIP

→ Enter Network 10.0.0.0 → Add

→ " " 20.0.0.0 → Add

→ " " 192.168.1.0 → Add

Thus step is done in order to add the neighbouring network address for router 3

4. Do same for Router 2, 1 & 4

Router 2 → config → RIP

→ 20.0.0.0 - add

→ 172.1.0.0 - add

→ 200.1.1.0 - add

Router 1 → config → RIP

→ 172.1.0.0 - add

→ 192.168.1.0 - add

→ 217.1.1.0 - add

Router 4 → config → RIP

→ 217.1.1.0 - add

→ 222.2.2.0 - add

5. Now to display the routing table

click on router (say router 1)

→ then on CLI & type the command

exit

exit

show route

Output:

R - 10.0.0.0/8 via 192.168.1.1 gig 0/0

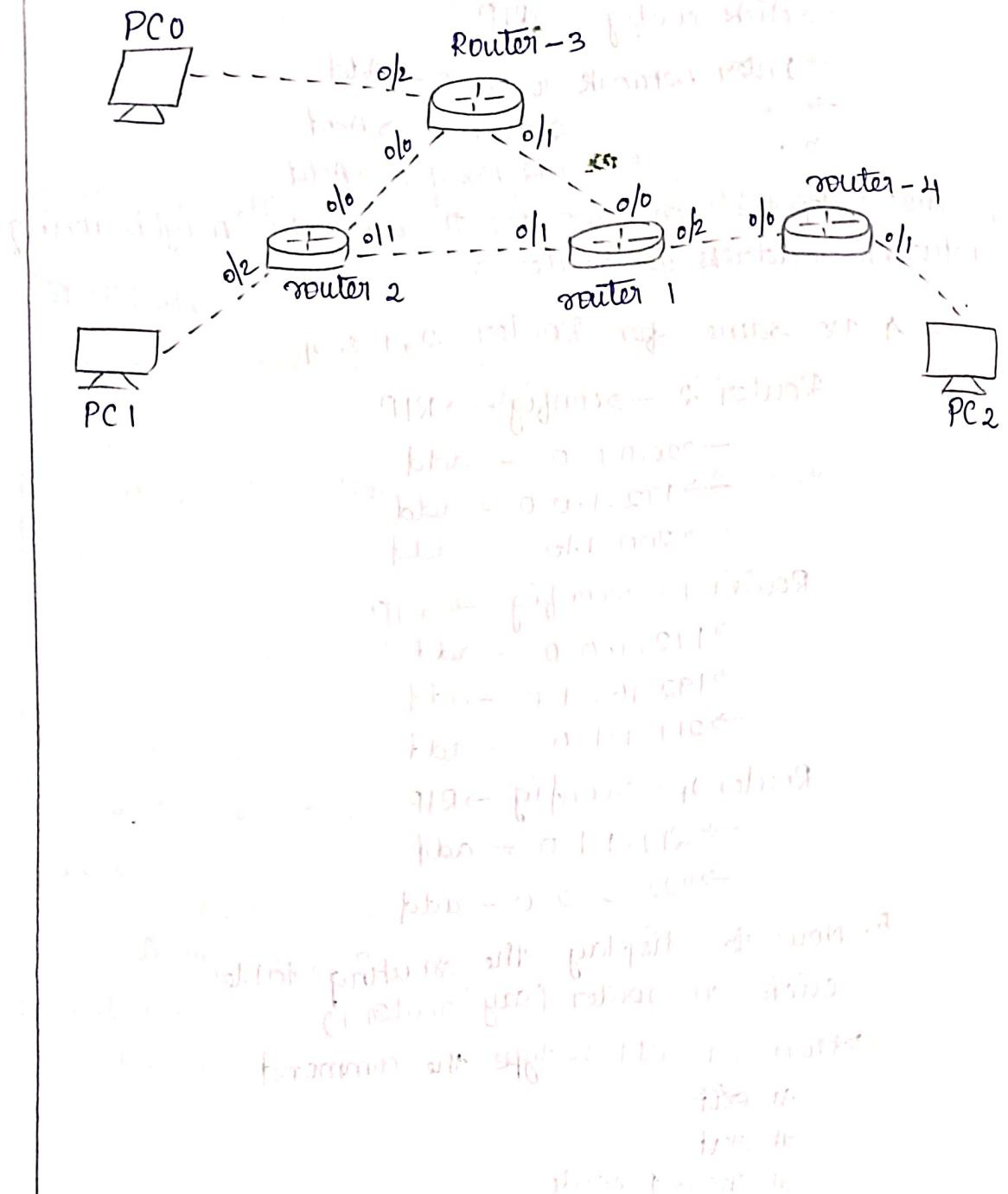
R - 20.0.0.0/8 via 192.168.1.1 gig 0/0

R - 172.0.0.0/16 is variable connected 2 subnet 2 mask

R - 172.1.0.0/16 is directly connected gig 0/1

L - 172.1.1.2/32 is directly connected gig 0/1

Diagrammatic Representation:



RESULT :

Thus, RIP is simulated using Cisco packet tracer successfully

Practice - 11 a

Static Routing Configuration

AIM: To stimulate static routing configuration using cisco packet tracer.

Procedure:

1) Arrange pc and Router in the manner as the image below

2) Assign IP address for router and PCs

Router 0 [2811]

→ IP FA 0/0 - 192.168.1.2
FA 0/1 - 192.168.2.3

Router 1 [2811]

→ IP FA 0/0 - 192.168.1.4
FA 0/1 - 192.168.3.3

PC 0

IP - 192.168.2.7
gateway - 192.168.2.3

PC 1

IP - 192.168.2.3
gateway - 192.168.2.3

PC 2

IP - 192.168.3.5
gateway - 192.168.3.3

PC 3

IP - 192.168.3.7

gateway - 192.168.3.3

3) Now click on router 0

then config → static

Network - 192.168.2.0

Subnet - 255.255.255.0

next-hop - 192.168.1.4

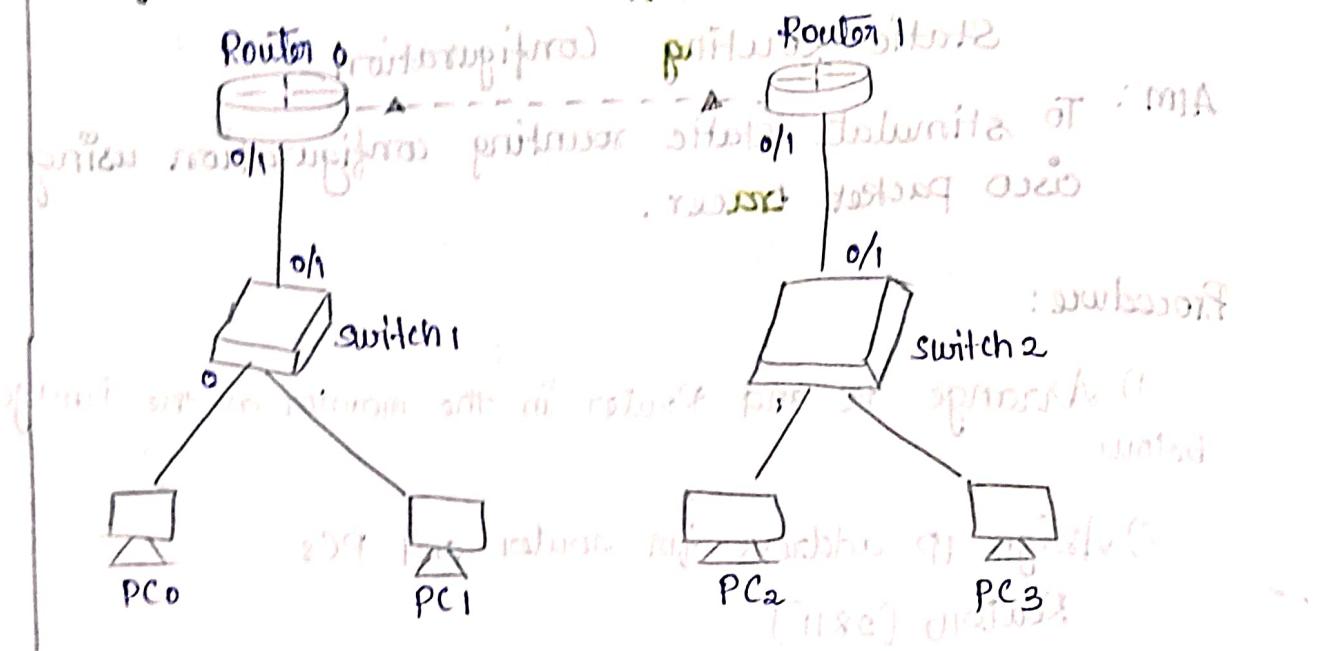
4) Now click on router 1

then config → static

Network - 192.168.2.0

Subnet - 255.255.255.0

Diagram:



Output

Ping from PC0 to PC2

| Vie | time (sec) | Last device |
|-------|------------|--------------------|
| 0.000 | | PC0 |
| 0.001 | | switch0 |
| 0.002 | | Router0 |
| 0.003 | | Router0 - pingpong |
| 0.004 | | Router1 |
| 0.005 | | switch1 |
| 0.006 | | PC2 |
| 0.007 | | switch1 |
| 0.008 | | Router1 |
| 0.009 | | Router0 |
| 0.010 | | switch0 |

| Fixe | laststatus | source | destination | type | color | type(sec) | period |
|------|------------|--------|-------------|-------|-------|-----------|--------|
| ① | successful | PC0 | PC2 | TCPMP | □ | 0.000 | N |

RESULT

Thus static routing configuration has been successfully run and execute.

Practicle - 10A

a) TCP / UDP sockets

Aim : Implement echo client server using TCP / UDP sockets.

ALGORITHM :

1. Create a socket and bind to an Address and port.
 2. Listen for connection.
 3. Accept a connection.
 4. Receive and Echo Data and close a connection.
-
1. Create a socket.
 2. Connect to the Server.
 3. Send data to the Server.
 4. Receive data from the server and close the socket.

PROGRAM :

```
import socket  
def tcp_echo_server():  
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server:  
        server.bind("localhost", 12345)  
        server.listen(1)  
        print("TCP server Listening...")  
        conn, addr = server.accept()  
        with conn:  
            print(f"Connected by {addr}")  
            while data := conn.recv(1024):  
                conn.sendall(data)  
echo_server()
```

TCP Echo Client

```
import socket  
def echo_client():  
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
        as client:  
            client.connect("localhost", 12345)  
            client.sendall(b"Hello, TCP server!")  
            print("Received:", client.recv(1024))  
echo_client()
```

Digitized by srujanika@gmail.com

RESULT :

The client server using TCP/UDP program has been successfully executed and output is verified.

PRACTICAL 12b

Practical 12b: Chat Server

AIM: Implement chat client server using TCP/UDP
SOCKETS.

SOURCE CODE:

```
TCP Chat Server
[ctrl] main.py [ctrl] client.py [ctrl] server.py [ctrl]
import socket, threading
def chat_server():
    server = socket.socket(socket.AF_INET,
                           socket.SOCK_STREAM)
    server.bind(("localhost", 12345))
    server.listen()
    clients = []
    def handle(client):
        while True:
            try:
                msg = client.recv(1024)
                for c in clients:
                    if c != client:
                        c.send(msg)
            except:
                clients.remove(client)
                break
        while True:
            client, _ = server.accept()
            clients.append(client)
            threading.Thread(target=handle, args=(client, server)).start()
chat_server()
```

TCP Chat Client: del JASITAR

Import socket, threading

```
def chat_client():
    client = socket.socket(socket.AF_INET,
                           socket.SOCK_STREAM)
    client.connect(("localhost", 12345))
    threading.Thread(target=lambda : print(client.recv(1024).decode())
                     for _ in iter(int, 1)).start()
chat_client()
```

RESULT:

The Chat Client Server using TCP/UDP
Program can be successfully executed and
output is verified.

AIM: Implement your own ping program.

ALGORITHM :

1. Import Required Modules for raw socket.
2. Define Checksum function for ICMP packet.
3. Create Ping function.
4. Receive and Handle reply and request timeout.
5. call the ping function and pass IP address.

Source code:

```

import socket
import struct
import time
import os

def checksum(data):
    S = sum((data[i] << 8) + (data[i+1] if i+1 < len(data)
                                else 0) for i in range(0, len(data), 2))
    S = (S >> 16) + (S & 0xFFFF)
    return S & 0xFFFF

def Ping(dest_addr):
    with socket.socket(socket.AF_INET, socket.SOCK_RAW,
                      socket.IPPROTO_ICMP) as sock:
        sock.settimeout(1)
        packet_id = os.getpid() & 0xFFFF
        header = struct.pack("!BBHHH", 8, 0, 0, packet_id, 1)
        data = struct.pack("d", time.time())

```

```
packet = header[:2] + struct.pack('!H', checksum(header+data)) + header[4:] + data
sock.sendto(packet, (dest_addr, 1))
```

import socket time
Start = time.time()
try:
 sock.recvfrom(1024) : MHT190101A
 print(f"reply from {dest_addr} : time =
 {time.time() - Start} * 1000 : .2f ms"
 except socket.timeout:
 Print("Request timed out")

Ping("8.8.8.8")

• گھریلے کاموں

RESULT:

The ping program has successfully executed and output is verified.

Aim: Write a code using RAW sockets to implement packet sniffing.

Algorithm:

1. Import Necessary modules to create raw Sockets.
2. Define packet sniffer Function
3. Starting capturing packets
4. Receive and process packets.
5. Run the packet sniffer Function

Source code:

```
import socket
def packet_sniffer():
    with socket.socket(socket.AF_PACKET, socket.SOCK_RAW,
                      socket.ntohs(0x0003)) as sniffer:
        print("Sniffer started . Capturing packets...")
        while True:
            packet, _ = sniffer.recvfrom(65565)
            print(f"Packet : {packet[64].hex()}")

```

packet_sniffer()

: Output

RESULT:

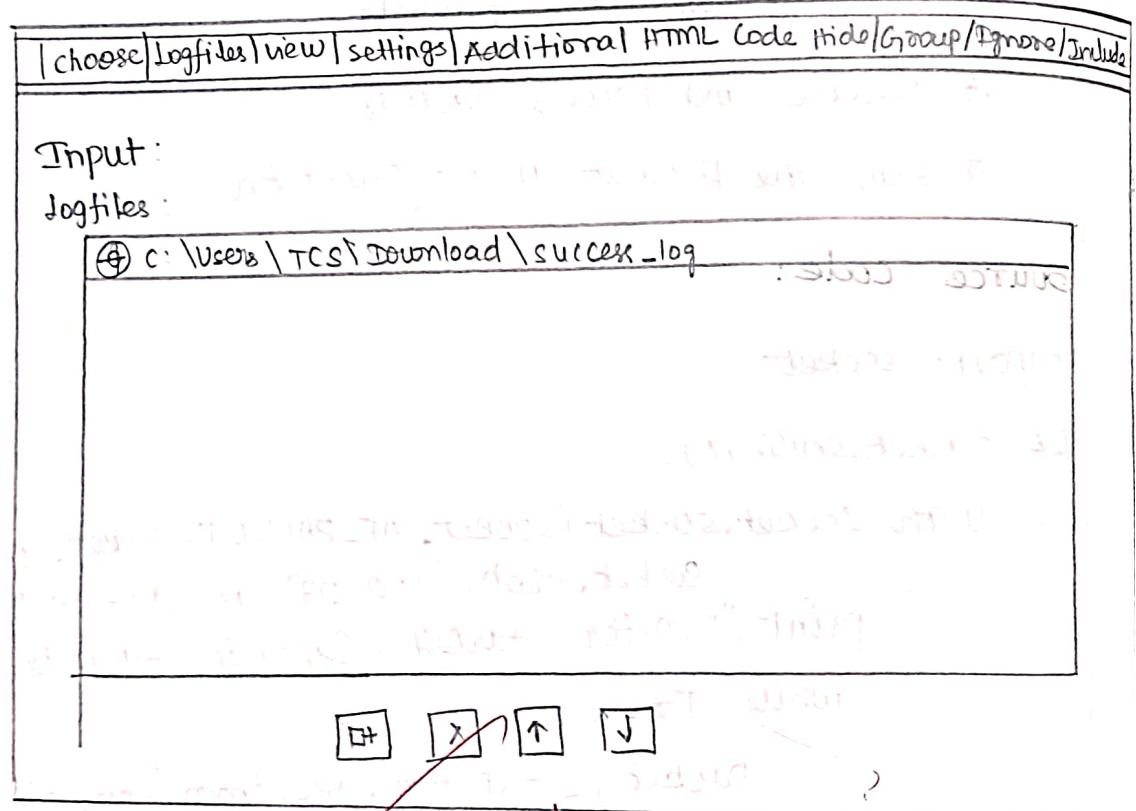
The RAW sockets program has been successfully executed and output is verified.

✓
Signature

~~AIM: To analyze the different types of web logs using webalizer tool.~~

PROCEDURE:

1. Run webalizer windows version.
2. Input web log file (download from web)
3. Press Run webalizer.



Output:

all 8

RESULTS: