INDEX

NAME: R. Karthiga STD.: _____ SEC.: 'B' ROLL NO.: 119 SUB.: CN Observation

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	16.7.24	Study of various Network Commands used in Linux and Windows		
2.	23.7.24	Study of different types of Network		
3	30.7.24	Study of packet tracer tool Installation & User Interface overview		
4.	6.8.24	Setup and config a LAN using switch and Ethernet cables.		
5.	9.8.24	Experiments on packet capture tool: Wireshark		
6.	20.8.24	Error Correction at Data Link Layer.		
7.	6.9.24	Sliding window protocol		
8.a	18.10.24	Virtual LAN		
8b	18.1024	Wireless LAN		
9.	8.10.24	Classfull Subnetting		
10.	18.10.24	Internetworking		

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
11b	18.10.24	RIP		
11a		Static Routing Config		
12 a)		TCP/UDP Sockets		
12 b)		Chat client server using TCP/UDP		
13.		Ping program		
14.		RAW socket packet sniffing.		
15.		Webalizer tool		

Completed

16.07.2024

PRACTICAL - 1.

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
172.16.8.1	7c-5a-1c-cf-be-45	dynamic
224.0.0.2	01-00-5e-00-00-02	static
239.255.255.250	01-00-5e-7f-ff-fa	static

2) Hostname

DESKTOP-HCVQANO

3) ipconfig /all

Windows IP configuration

Host Name . . . . . : DESKTOP-HCVQANO
Primary Dns Suffix . . . . . :
Node Type . . . . . : Mixed
IP Routing Enabled . . . . . : No
WINS Proxy Enabled . . . . . : No

Ethernet adapter Ethernet :

Connection-specific DNS Suffix . . :
Description . . . . . : Realtek PCIe GbE Family Controller
DNS servers . . . . . : 172.16.8.1
NetBIOS over Tcpip. . . . . : Enabled

4) nbtstat -a

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 addr
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	HDC1017152:38881	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),
Min = 0ms, max = 1ms, 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 COMMANS:

1) arp -a

gateway (172.16.8.1) at 7C:5A:1C:CF:BE:45 [ether] on enp2s0

2) Hostname

localhost.localdomain

3) ifconfig

enp2s0 : flag=4163<UP, BROADCAST, RUNNING, MULTICAST> mtu 1500

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

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

4) nmblookup -A <ip address>

nmblookup -A 14.99.10.282

Looking up status of 14.99.10.232

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

DESKTOP-BQ498VC <00> - B <ACTIVE>

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

5) nslookup www.google.com

Server : 172.16.8.1

Address : 172.16.8.1#53

Non-authorised answer:

Name : www.google.com

Address : 142.250.183.228

6) Ping

i) Ping localhost

PING localhost (localhost (::1)) 56 data bytes

64 bytes from localhost (::1) : icmp_seq=1 ttl=64 time=0.017ms

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=35.2 ms

iii) Ping www.facebook.com

```
PING star-mini.c10r.facebook.com (157.240.192.35) 56(84)
                                          - bytes of data.
64 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 lo
   valid_lft forever
inet6 ::1/128
2: enp2s0: <BROADCAST, MULTICAST, UP, LOWER_UP> mtu 1500
link/ether 50:9a:4c:34:d8:85
3: wlp3s0: <BROADCAST, MULTICAST, UP, LOWER_UP> mtu 1500
link/ether d4:6a:6a:82:ca:fb
```

b) # ip address add 192.168.1.254/24 dev enp803

c) # ip address del 192.168.1.254/24 dev enps03

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 enp2s0 src 172.16.8.84 uid 0 cache

2. ifconfig

enp2s0 : 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> hostname/IP

a) #mtr google.com

Host	Packets		Pings				
	Loss%	Snt	Last	Avg	Best	Wrst	StDev
172.16.8.1	52.2%	160	0.2	0.2	0.2	0.3	0.0
142.250.171.161	48.6%	181	2.9	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

4. tcpdump

tcpdump : 287 packets captured
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.i686 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 or -vv

listening on enp2s0, link-type EN10MB (ethernet), capture

11:31:24.517943 IP 172.16.9.164.. 5101 > 239.255.255.250.

11:31:24.518748 IP localhost.localdomain.localdomain.4515

18 packets captured, 328 packets received, 328 dropped.

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 enp2s0 -c 4 host 8.8.8.8

tcpdump: verbose output suppressed, use -v

listening on enp2s0, link-type EN10MB (Ethernet)

0 packets captured

0 packets received by filter

0 packets dropped by kernel.

f) #tcpdump -i enp2s0 src host 8.8.8.8

tcpdump: verbose output suppressed, use -v or -w

0 packets captured

0 packets received by filter

0 packets dropped by kernel.

nmcli connection show

NAME	UUID	TYPE	DEVICE
wired connection 1	54fbdda8-3af4-3001-8551-defed4f2909e	ethernet	enp2s:

#nmcli connection add con-name enp0s2 type ethernet

connection. enp0s2 (6406790-6702-4761-9fb3-048090aeb74d)

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

#nmcli connection modify "wired connection 1" ipv6.method auto

#ip address show enp0s2

2: enp0s2: <BROADCAST, MULTICAST, PROMISE, UP, LOWER-UP> mtu 1500 qdisc fq-codel state UP group default qlen 1000

link/ether 08:00:27:1f:01:65 brd ff:ff:ff:ff:ff:ff.

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 edns0 trust-ad
search mshome.net

16/7/24

Student Observation:

1. Which command is used to find the reachability of a host machine from your device:
   * ping cmd.
2. Which command will give the details of hope taken by a packet to reach its destination?
   * mtr (Matt's traceroute)
3. Which command display the IP config of your machine?
   * IP <options> <object> <command>
4. Which command displays the TCP port status in your machine?
   * netstat
5. Write the modify ip config in a Linux machine.
   * address add 192.168.1.254/24 dev enp0s3
   * ip address del 192.168.1.254/24 dev enp0s4

RESULT:

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

23.07.2024

PRACTICE - 2

AIM:- Study of different types of Network cables.

Different type of cables of network:

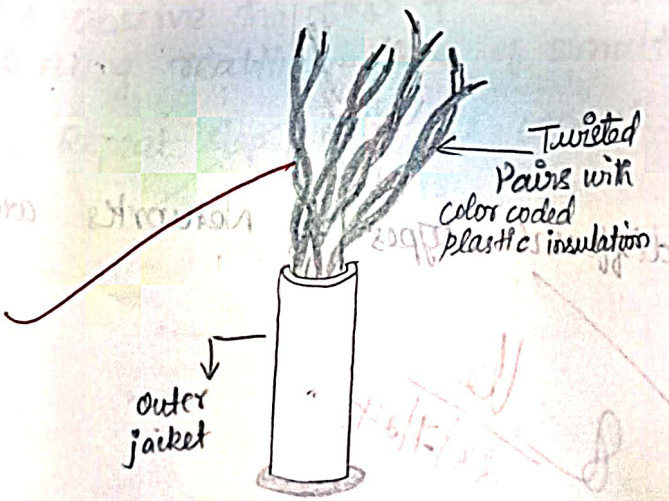
1. Unshielded Twisted Pair (UTP) cable
2. Shielded Twisted Pair (STP) cable
3. Coaxial Cable
4. Fibre Optic cable.

Cable Type	Category	Maximum Data Transmission	Advantages / Disadvantages	Application / Use
UTP	Category 3	10 bps	**Advantages** • cheaper in cost • Easy to install as they have a smaller overall diameter. **Disadvantages** • More prone to (EMI) Electromagnetic Interference and noise.	10Base-T Ethernet
	Category 5	Up to 100 Mbps		Fast Ethernet, Gigabit Ethernet
	Category 5e	1 Gbps		Fast Ethernet, Gigabit Ethernet.
STP	Category 6, 6a	10 Gbps	**Advantages** • shielded • Faster than UTP • less susceptible to noise and interference **Disadvantages** • Expensive • Greater Installation effort	Gigabit Ethernet, 10G Ethernet (50m) widely used in data centres
SSTP	Category 7	10 Gbps		Gigabit Ethernet, 10G Ethernet (100 m)

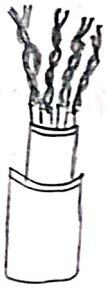
Cable Type	Category	Data Transmission	Advantages / Disadvantages	Application / Use
Coaxial cable	RG1-6 RG1-59 RG1-11	10-100 Mbps	**Advantages** • High bandwidth • Immune to interference • Low loss bandwidth • versatile **Disadvantages** • Limited distance • Cost • Size is bulky	Speed of signa is 500m Television networ High speed internet connectior
fibre optics cable	Single mode Multi-mode	100 Gps	**Advantages** • High speeds • High bandwidth • High security • Long distance **Disadvantages** • Expensive • Requires skilled installers	• Maximum distance of fibre optics cable is around 100 mete

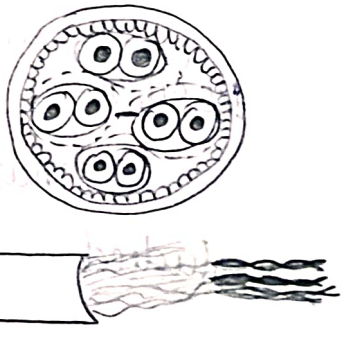
Images:

a) UTP



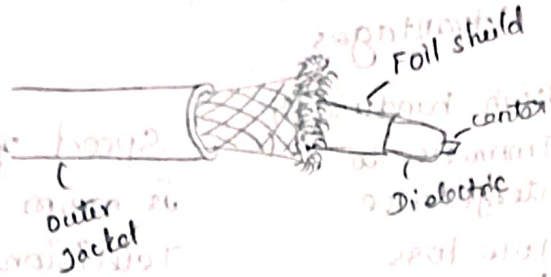
STP



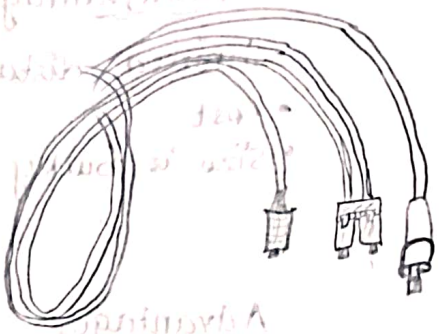


SSTP

Coaxial cable



Fibre optics cable.



RESULT:-

Thus, the different types of Networks are studied.

23/7/24

30.7.2024

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:
   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 cabl The link LED should glow in green, indicating that the link is u
   c. Similarly connect 4 PCs to the switch using copper straight-through cable.

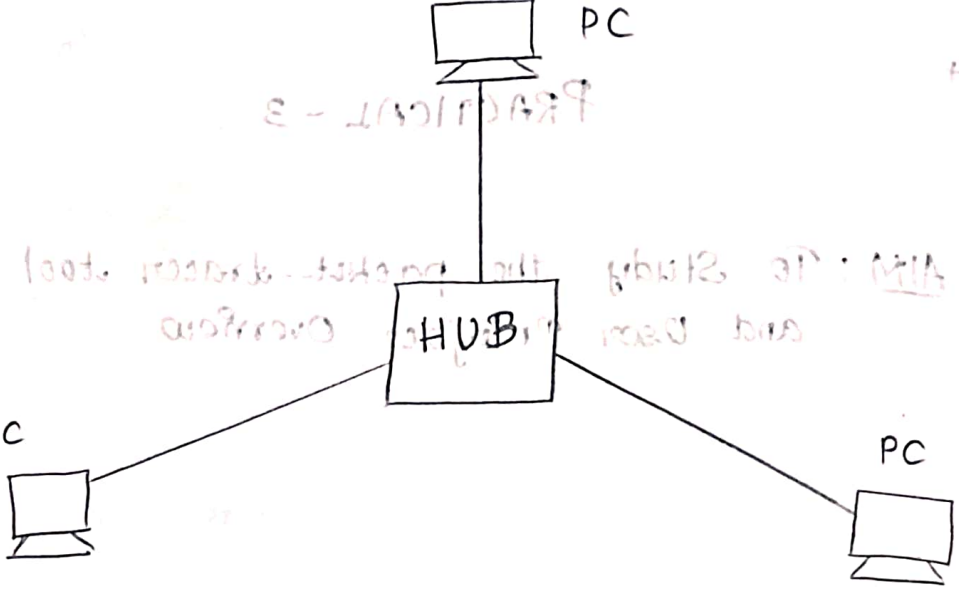
3. click and PC's connected to hub, go to desktop tab, click IP Configuration and enter IP address.

   click message Icon from common tool bar,

   a) Drag and Drop it once of PC and drop it on anothe PC connected HUB.

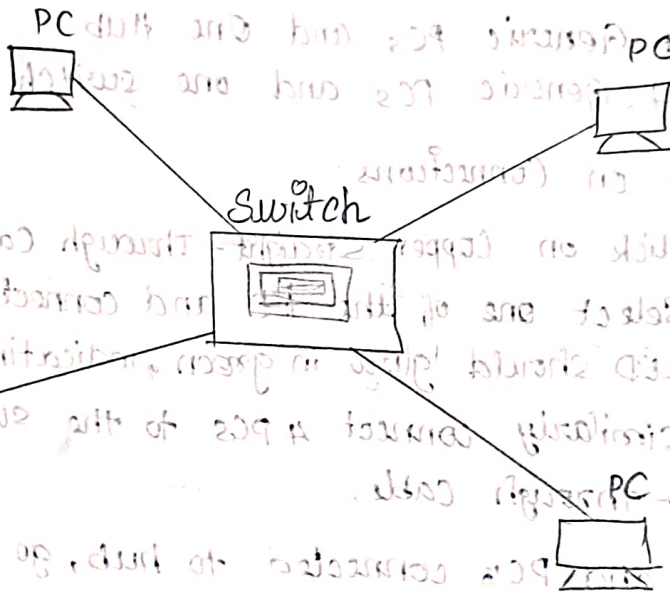
4. Observe the flow of PDU from sender to receiver by selecting realtime mode of simultation.

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



HUB:

*



Observation:

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 link 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 business broadcasts incoming data to all connected devices, regardless of the destination.

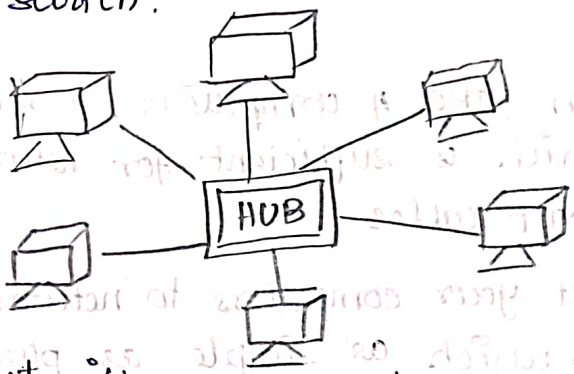
b) Network topology implementation in your college and draw and label that topology is your observation book.

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

characteristics of star

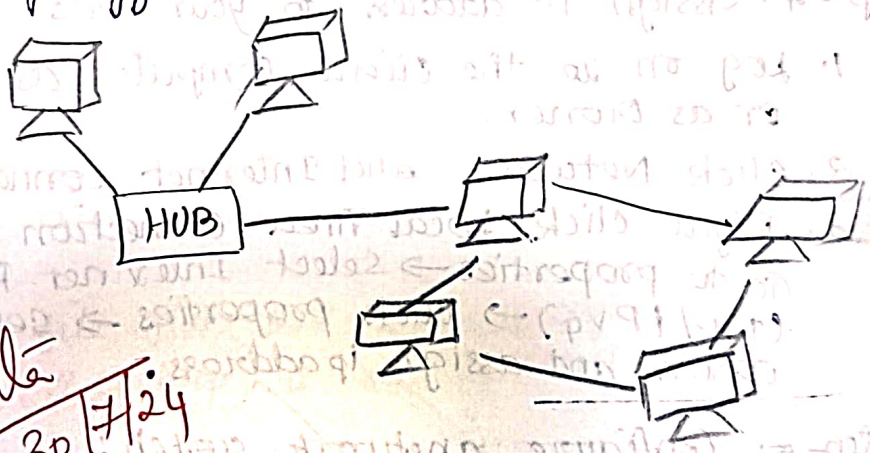
→ all cable run to a central connection point

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



characteristics of Hybrid

→ Hybrid topology is combination of more than one topology



30/7/24

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 Architecture 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

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

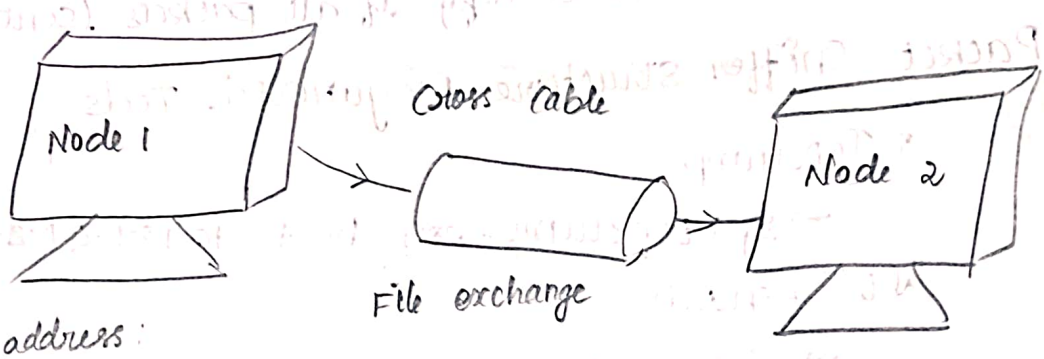
Step - 7 : Select a folder, → go to properties → click sharing tag → share it with everyone on the same LAN.

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 & energy device. write the outcome and challege faced while configuring the LAN.

Diagram of the LAN :



IP address :
172.16.8.82

IP address :
172.16.8.83

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 : [//172.16.8.83 → users//REC//Desktop] is sent successfully.

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

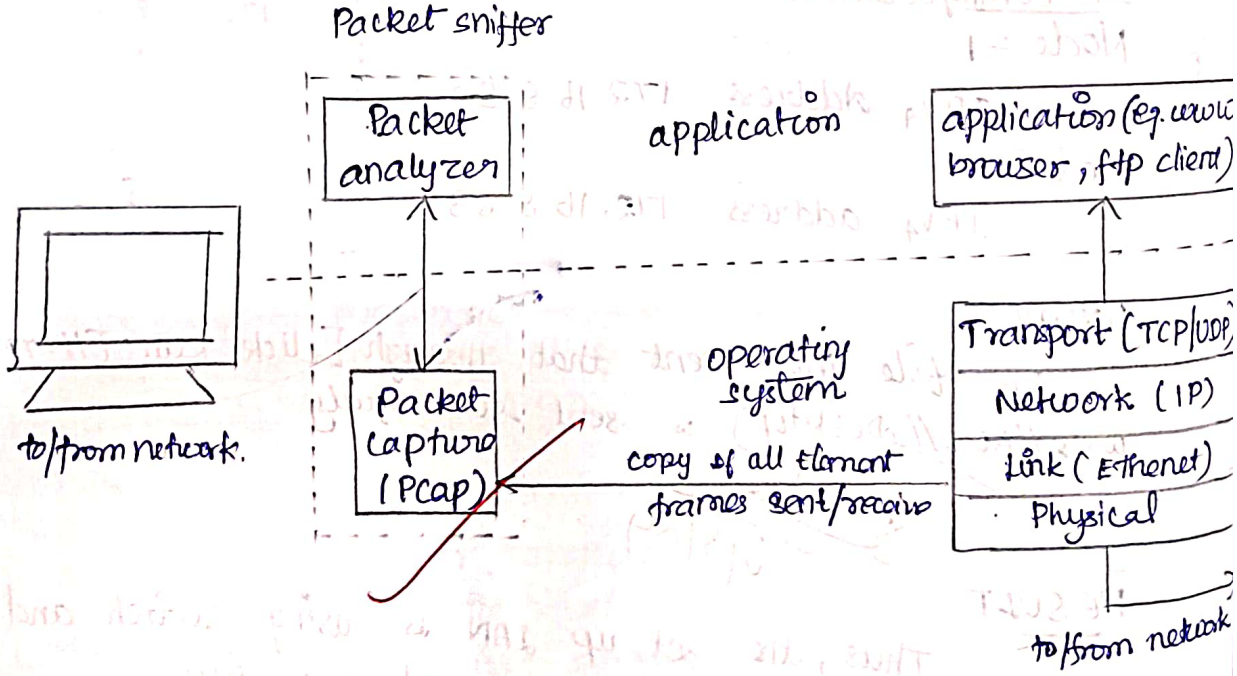
AIM: Experiments on Packet capture tool: Wireshark

Packet Sniffer:

* Sniffs messages being sent/received from/by your computer.
* store and display the contents of the various protocal fields in the messages.
* Passive program.
  - ⇒ never sends packets intself
  - ⇒ no packets addressed to it
  - ⇒ receives a copy of all packets (sent/received)

Packet Sniffer Structure Diaganostic Tools:

* Tcpdump
  - ⇒ Eg: tcpdump -exn host 10.129.41.2-10.
* Wireshark
  - ⇒ wireshark -r exe3.out.



Wireshark

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

What we can do with Wireshark?

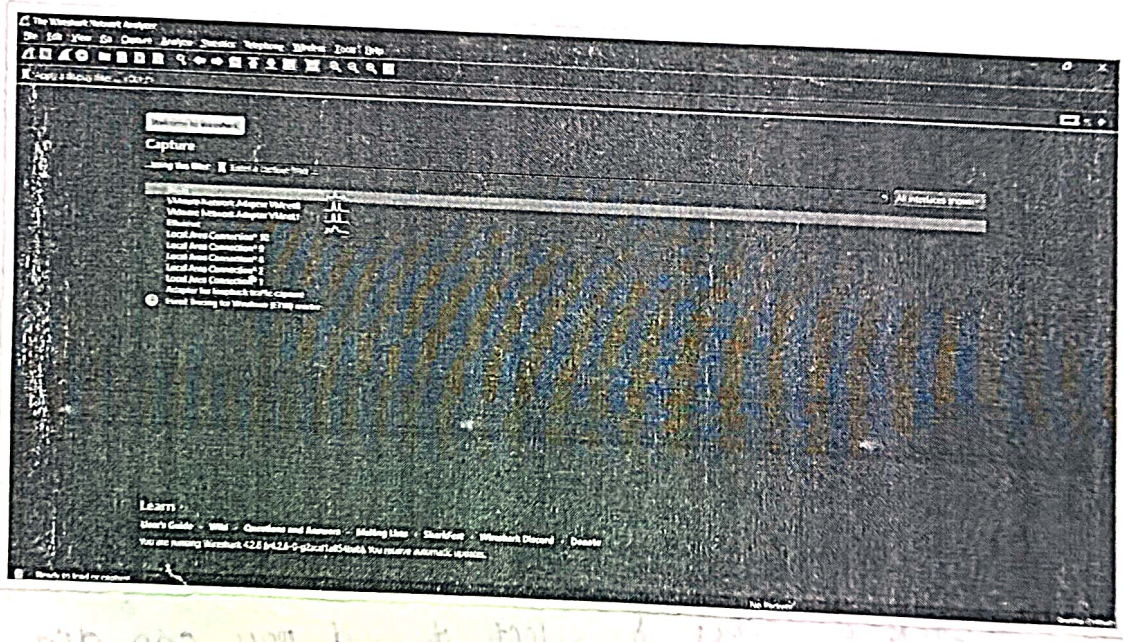
* Capture network traffic
* Decode packet protocals using dissectors.

Wireshark used for:

* Network administrators: troubleshoot network probb
* Network security engineers: examine security problem

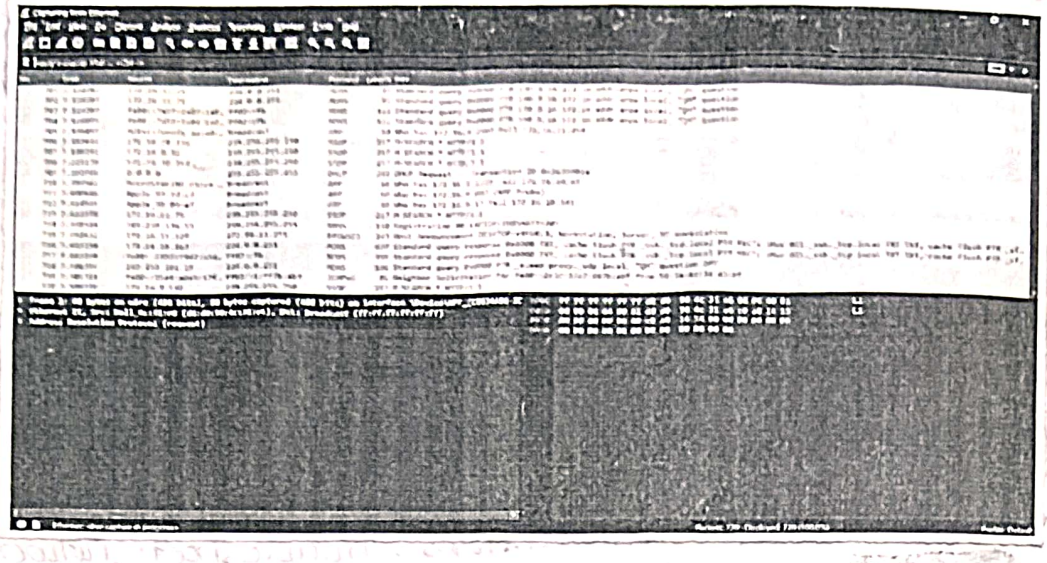
Capturing Packets.

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



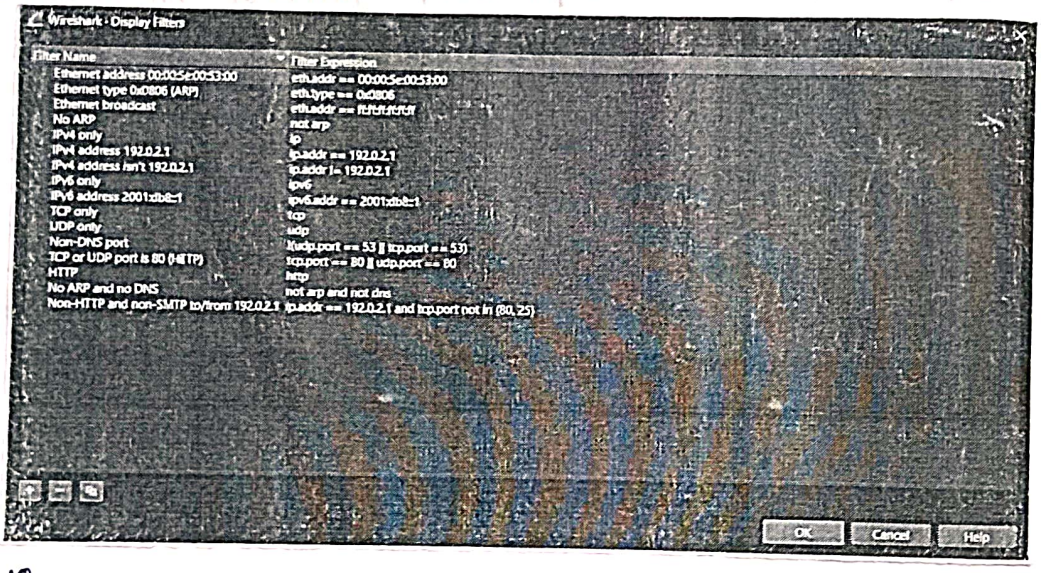
Color coding

* 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



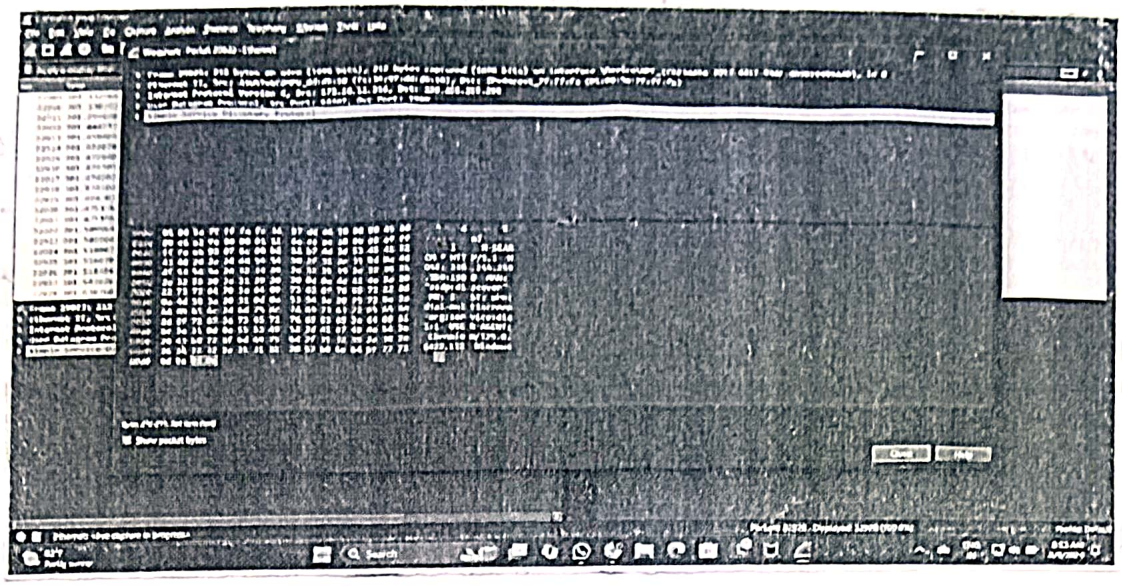
Filtering packets

* the boost basic way to apply a fibre 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.



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 do 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 protocal is used DNS?

⇒ uses UDP as its transport layer protocal on port 53

4. What is the port number used by http protocal?

⇒ HTTP protocal 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.

RESULT :-

Thus, Wireshark tool has been experiments on packet captured and studied.

9/8/24

20.8.2024

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 this 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

Code :-

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

def calculate_redundant_bits(m):
    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(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(f"Binary representation: {binary_data}")
    m = len(binary_data)
    r = calculate_redundant_bits(m)
    encoded_data = insert_redundant_bits(binary_da
                                         , m, r)
    Print(f"Encoded data with redundat bits:
                                {encoded_data}")
```

```
    Print(f"Redundant bit positions: {[1<<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 = (1<<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))):
                xor_sum ^= int(encoded_data)
        error_position |= (xor_sum<<i)

    if error_position:
        Print(f"Error found at position: {error_position}")
        corrected_data = list(encoded_data)
        corrected_data[error_position-1] = '0' if
                encoded_data[error_position-1]=='
                else '1'
        Print(f"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+r+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 i in
                    range(0, len(binary_str), 8)])
```

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

    change = input("Do you want to change any bit in encoc
                                    data: ").
                                    strip().lower()
    if change == 'yes':
        position = int(input(f"Enter the bit position to
                    change (1-{len(encoded_data)}): "))

        if 1 <= position <= len(encoded_data):
            is_redundant = position in [1<< i for i in rang
            encoded_data = list(encoded_data)
            encoded_data[position-1] = '0' if encoded_d
                -[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_correc
                                (encoded_data, r)
        if change == 'yes' and is_redundant and error_posit
                                        == position:
            print("The change in the redundent bit was
                                corrected successfully.")
        elif change == 'yes' and is_redundant and
                            error_position != position:
            print("The change in the redundant bit was
                                not detected")
        original_text = extract_original_data(corrected_d
                                        , r)
        print(f"Decoded text: {original_text}")
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.
```

RESULT:

20/8/24. The program of Hamming Code has been succesfully 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 acheive 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. Reader 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.

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.

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.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.fram
                        , DATA: {self.message[self.frame
                    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("\n".join(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:
            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 {se
                        expected_ack_no + 1}, got {ack_no}. Resending
            elif ack_type == "NACK":
                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(f"sender buffer {self.sender_buffer} does not exi
                waiting for frames...")
    continue.
with open(self.sender_buffer, 'r') as f:
    frames = f.readlines().
if not frames:
    continue.
Print("\n---RECEIVING FRAMES---")
ack_to_send = []
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 = protocal.rece
sender_thread.start()
receiver_thread.start()
sender_thread.join()
receiver_thread.join()
Print("\n Transmission Completed")
```

Output:

```
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.

Tramission 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

6/9/24

RESULT:

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

8-10-24

PRACTICLE - 9

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.

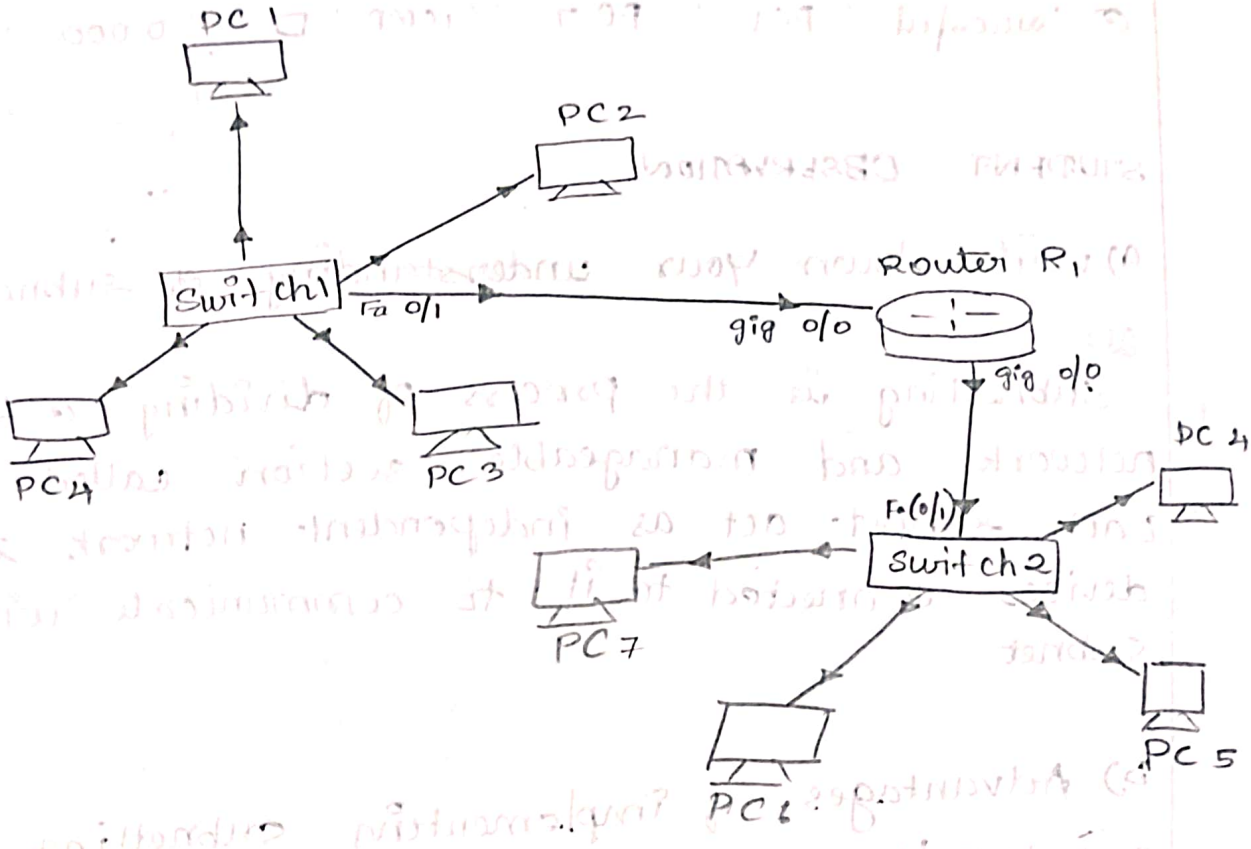
→ Switch S1
* no IP

→ 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

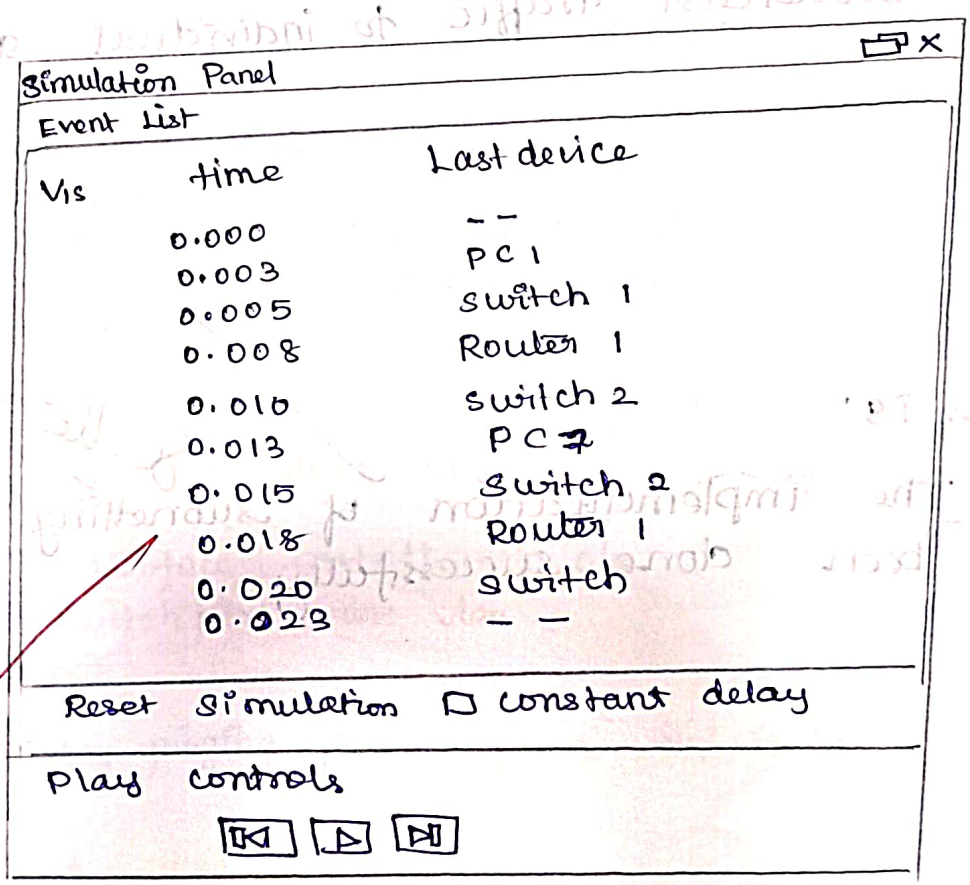
Diagramatic



OUTPUT:

Now lets assume the switch sender is PC 1 and receiver is PC6

While simulating & observing we get the simulation panel.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num
⊙	successful	PC1	PC 7	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 section called subnets. Each subnet act as independent network & allow devices connected to it to communicate within the subnet

b) Advantages of Implementing subnetting within a network.

Ans:-

↳ efficient IP management - based on require -ment.

↳ Reduce Network congestion - Limit broadcast traffic to individual subnets.

RESULTS :

The Implementation of subnetting in cisco has been done successfully.

18-10-24

PRACTICLE - 8a

a) VIRTUAL LAN

AIM: Simulate virtual configuration using CISCO Packet Tracer simulation.

PROCEDURE:

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

2) 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

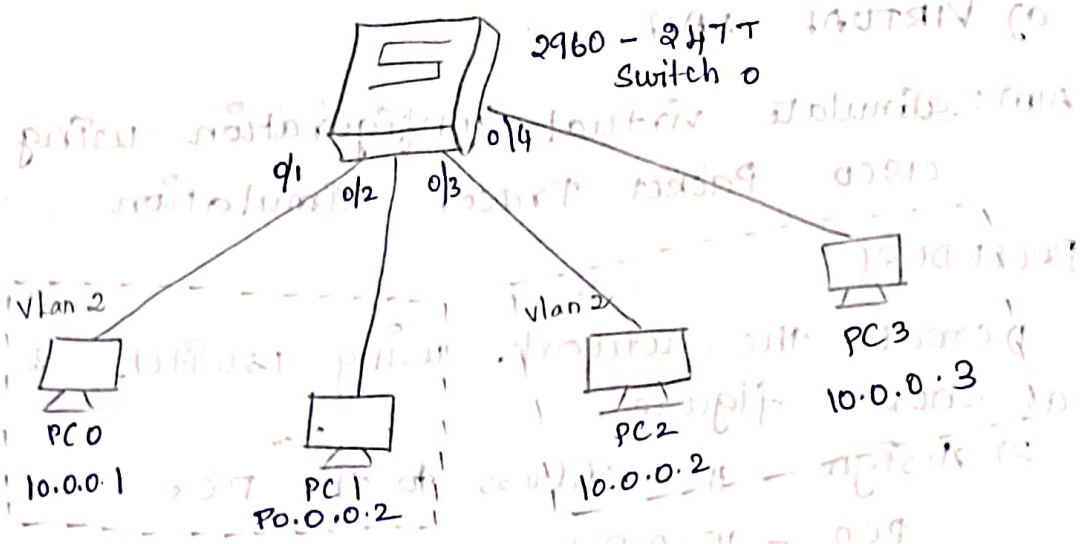
3) Check the network by sending packets.

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
#switch port access vlan 2
#exit
#interface fastEthernet 0/2
#switchport access vlan 2
#exit
#interface fastethernet 0/3
#switch port access vlan 3
#exit.
```

5) Now try pinging packet from PCs 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	Last Status	Source	Destination	type	colour	Time	periodic	Num
⊙	Successfull	PC0	PC1	ICMP	□	0.000	N	0
⊙	failed	PC1	PC2	ICMP	□	0.655	N	1

RESULT :

Thus the virtual LAN has been stimulated using Cisco packet traces.

18-10-24

PRACTICLE - 8b

Wireless LAN

AIM: To configure wireless LAN using cisco packet tracer.

PROCEDURE:

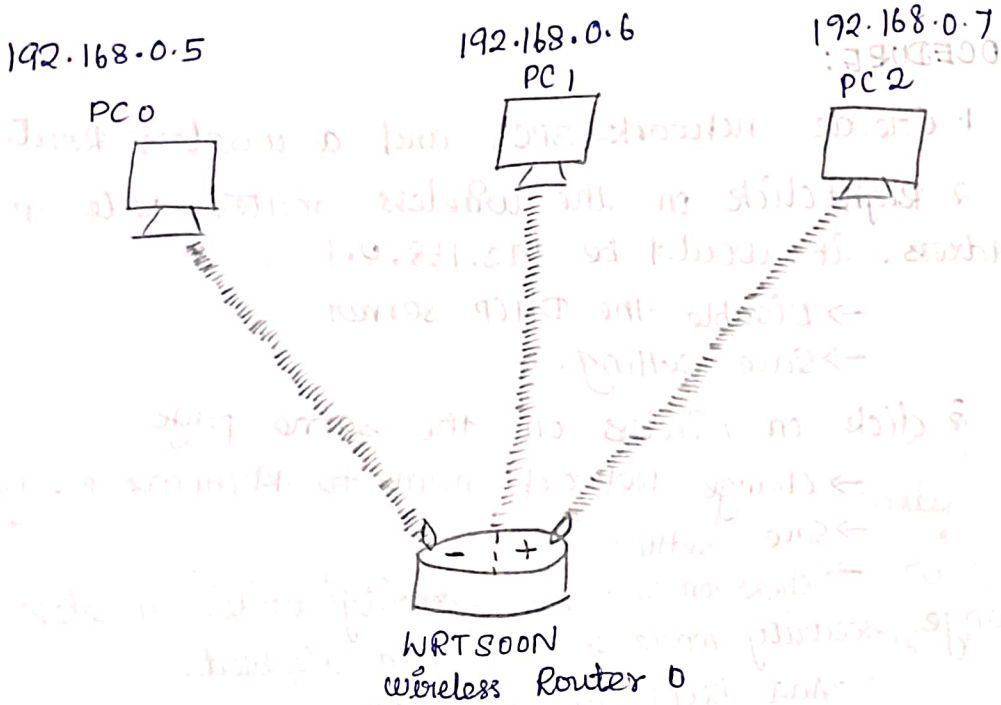
1. Create network 3PCs and a wireless Router
2. Right click on the wireless router. Note the IP address. it would be 192.168.0.1
   → Disable the DHCP server
   → Save setting.
3. click on wireless on the same page
   → change Network name to MYHome Networ
   → Save setting
   → click on wireless security 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 le
   → Turn on that
   → Now click on desktop → PC wireless
   → click connect → site info → connect

→ Type '0123456789' in the WEP key 1
→ Connect

6. Repeat step 5 for the PC1 and PC2

Then you can see the below connection in the network

Diagramatic Representation:



Output:

Thus the packets could be transmitted via wireless router.

fire	last Status	Source	Destination	Type	colour	Time	Period	Num
	Successful	PC0	PC2	ICMP		0.000	N	0
	Successful	PC1	PC0	ICMP		0.766	N	1

Simulation Panel

Event List

Vis	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.
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 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 authorised users can connect. Common types are WEP, WPA etc.

Result:

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

18-10-24

PRACTICLE - 11b

RIP

AIM: To stimulate RIP using cisco packet tracer.

PROCEDURE:

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

2) Assign IP address for the PCs & router port

PC0
IP - 10.1.1.1
Gateway - 10.1.1.2

PC1
IP - 200.1.1.1
Gateway - 200.1.1.2

PC2
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 inorder 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 commond

exit

exit

Show ip route

Output :

R - 10.0.0.0/8 via 192.168.1.1 gig 0/0

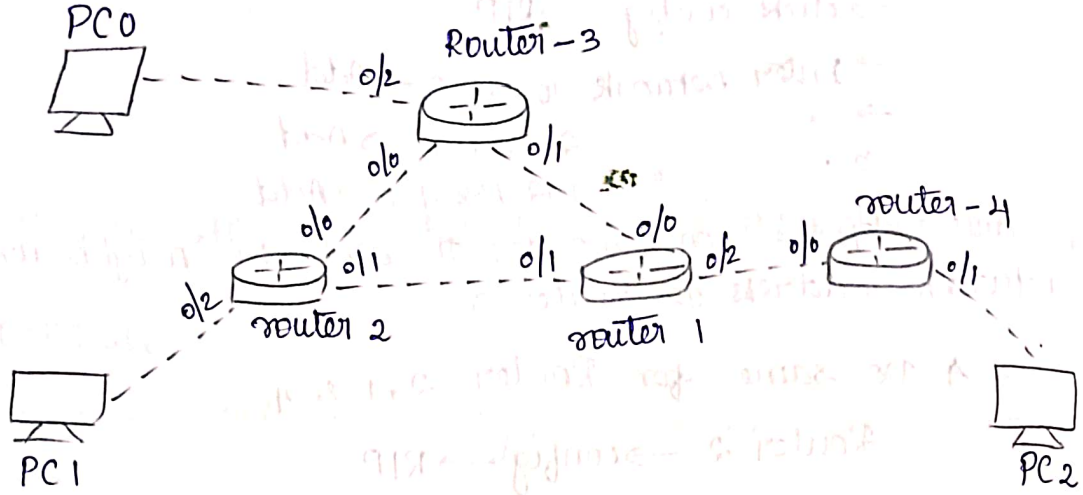
R - 20.0.0.0/8 via 192.168.1.1 gig 0/0

172.1.0.0/16 is variable connected 2 subnet 2 mask

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

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

Diagramatic Representation:



RESULT:

Thus, RIP is simulated using cisco packet tracer successfully

Practicle - 11 a

Static Routing Configuration

AIM: To stimulate static rounting 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

Router0 [2811]

→ IP FA 0/0 - 192.168.1.2

FA 0/1 - 192.168.2.3

Router1 2811

→ IP FA 0/0 - 192.168.1.4

FA 0/1 - 192.168.3.3

PC0

IP - 192.168.2.7

gateway - 192.168.2.3

PC1

IP - 192.168.2.3

gateway - 192.168.2.3

PC2

IP - 192.168.3.5

gateway - 192.168.3.3

PC3

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

nexthop - 192.168.1.4

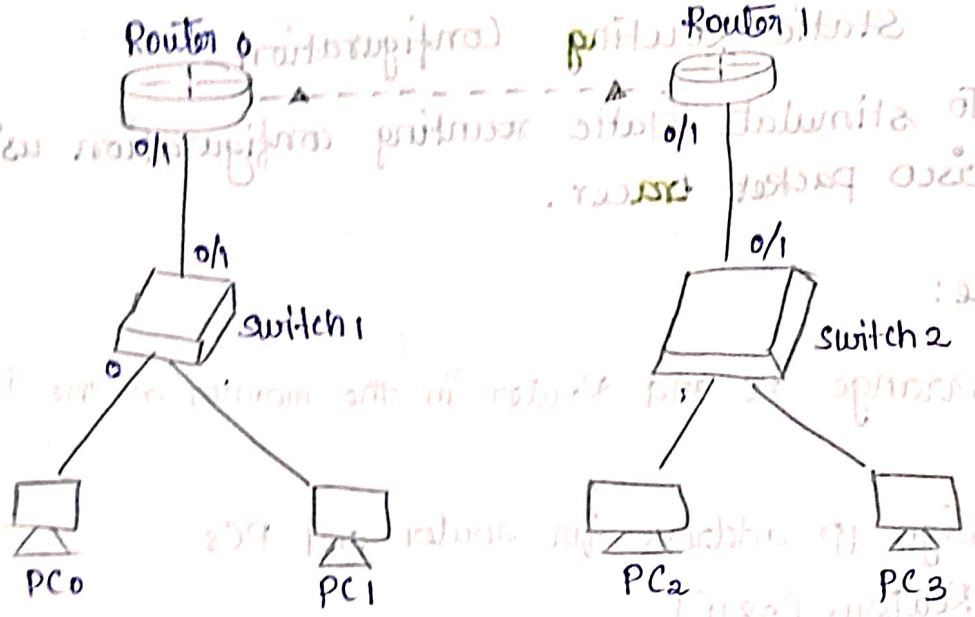
4) Now click on router 1

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	PC 0
	0.001	switch 0
	0.002	Router 0
	0.003	
	0.004	Raiter 1
	0.005	switch 1
	0.006	PC2
	0.007	Switch 1
	0.008	Router 1
	0.009	Router 0
	0.010	Switch 0

Fire	Last Status	Source	destination	type	color	Time(sec)	period
●	successful	PC0	PC2	ICMP	■	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 por
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 socke

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

RESULT:

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

PRACTICAL 12b

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

SOURCE CODE:

TCP Chat Server

```
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, 21
                                                 server)
chat server()
```

TCP chat Client:

```
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.

PRACTICLE - 13a

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(da
                                                        ,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))

    start = time.time()

    try:
        sock.recvfrom(1024)
        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")
```

RESULT:

The ping program has successfully executed and output is verified.

PRACTICLE - 14

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)
            Paint(f"packet: {packet[64].hex()}")
Pnacket_sniffer()
```

RESULT:

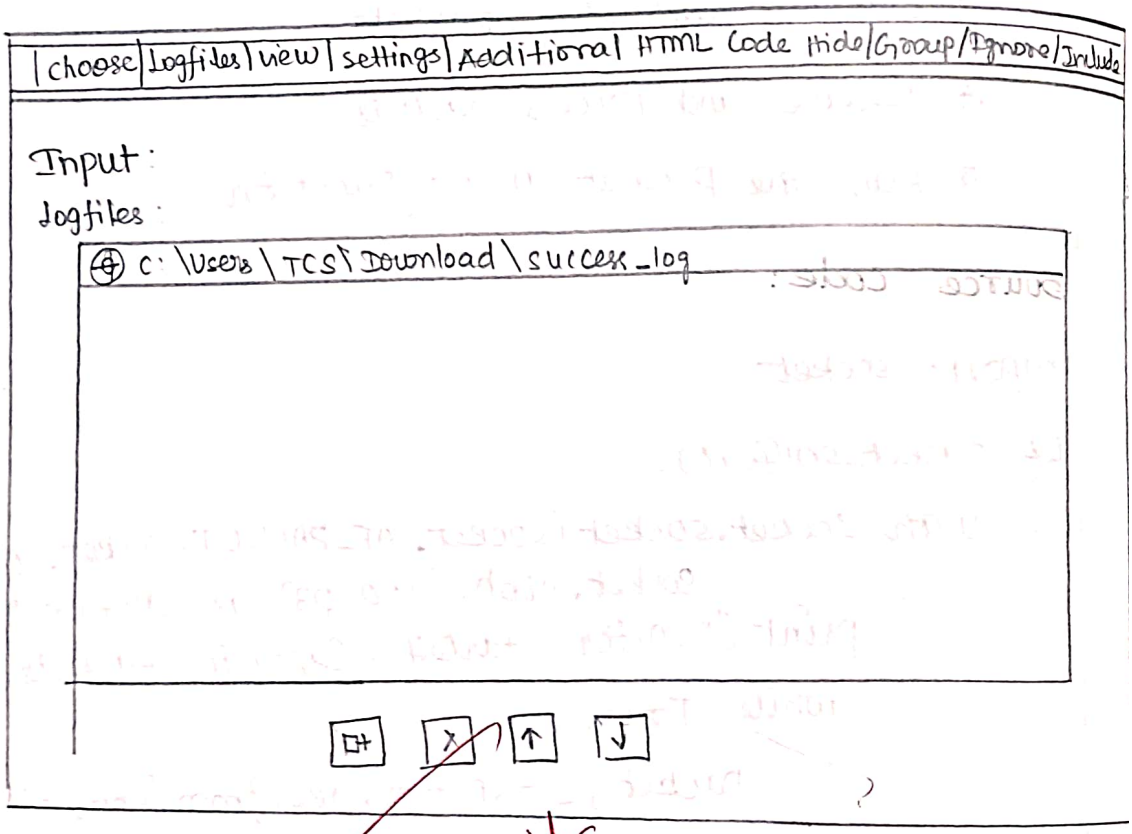
The RAW sockets program has been successfully executed and output is verified.

PRACTICAL – 15

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

PROCEDURE:

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



Output: