

Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

© CERTIFICATE ©

Certify that Mr./Miss Parth Sandeep Dabholkar of Computer Department, Semester II with Roll No. 2103032 has completed a course of the necessary experiments in the subject Computer Networks under my supervision in the **Thadomal Shahani Engineering College** Laboratory in the year 2023 - 2024

The 19/10/23
Teacher In-Charge

Head of the Department

Date 19/10/23

Principal

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1.	Study of RJ45 and CAT6 cabling and connection using crimping.	1 - 4	19/07/23	3
2.	Implementation of Hamming Code for Error Detection and Correction.	5 - 11	26/07/23	
3.	Implementation of CRC for Error detection	12 - 18	02/08/23	
4.	Simulation of Go-Back N Algorithm	19 - 23	09/08/23	
5.	Build simple n/w topology and configure it for static routing using packet tracer. Configure IP, subnetting and masking.	24 - 30	23/08/23	
6.	Design VPN and configure RIP using packet tracer.	31 - 40	06/09/23	Tej 10/09/23
7.	WAP to implement IPv4 addressing along with subnet masking.	41 - 46	13/09/23	
8.	Use basic networking commands in Linux. (ping, tracert, nslookup, ip, ifconfig, dig)	47 - 55	27/09/23	
9.	Use Wireshark to understand QoS of TCP/UDP (Ethernet, Data-link, Network, Transport and Application Layer)	56 - 61	27/09/23	
10.	Socket programming using TCP or UDP	62 - 65	04/10/23	
11.	Assignment 1	66 - 77	17/10/23	
12.	Assignment 2	78 - 83	17/10/23	

EXPERIMENT : 1

AIM: Study of RJ45 and CAT6 cabling and connection using crimping tool.

THEORY:

Tools and materials required are:

- i) Ethernet cable
- ii) RJ45 connectors
- iii) Wire stripper
- iv) Cable tester
- v) Crimping tool

* Steps involved in crimping RJ45 are:

- 1) Prepare the cable: Start by cutting the ethernet cable to the desired length, making sure it's long enough for your needs but not too long. Use the wire stripper to carefully remove about 1.5 inches of the outer jacket from the end of the cable. Be cautious not to cut into the inner wires while stripping.
- 2) Untwist and arrange wire: Inside the cable, you'll find four twisted pairs of wires. Untwist them gently without overstretching. Arrange the wires in the correct order for your chosen wiring standard. For a standard ethernet cable using the T568B wiring scheme, the order from left to right should be: white-orange, orange, white-green, blue, white-blue, green, white-brown, brown.
- 3) Trim and align wires: Trim the wires so they are all even and flush with the front of the RJ45 connector.

Use a wire cutter or scissors for this. Make sure the wires are in the correct order and properly align.

- 4) Insert wires into connector: Carefully insert the trimmed wires into the RJ45, ensuring they go all the way to the end of the connector and make contact with the metal pins inside.
- 5) Crimp the connector: Place the RJ45 connector, with the wires inserted, into the crimping tool. Ensure the connector is seated properly in the tool's crimping tool's crimping chamber. Squeeze the crimping tool's chamber firmly and evenly.
- 6) Inspect the connection: Visually inspect the RJ45 connector to ensure all wires are still in the correct order and properly secured.
- 7) Repeat for the other end: If you are completing a ethernet cable, repeat the above steps on the other end of the cable.
- 8) Test the cable: To ensure your cable is properly crimped and functional, you can use a cable tester.

- * Cat 6 cable: Category 6 is a standardized cable and connector type used in networking to transmit data at higher speeds and with better performance compared to its predecessors, such as Cat5.

Features of CAT6 :

- 1) Higher Data Transfer Rates
- 2) Enhanced Performance
- 3) Backward compatibility
- 4) Quality and shielding
- 5) Connector compatibility

6)

- * The T568A and T568B are two different wiring schemes used for Ethernet cables, specifying the arrangement of wires within an RJ45 connector.

T568A :

- 1) white/Green
- 2) Green
- 3) white/orange
- 4) Blue
- 5) white/blue
- 6) orange
- 7) white/Brown
- 8) Brown

T568B :

- 1) white/orange
- 2) orange
- 3) white/green
- 4) blue
- 5) white/blue
- 6) green
- 7) white/Brown
- 8) Brown.

Standard Ethernet Connection :

- 1) Standard Ethernet cables are used for connecting different types of network devices, such as computers to switches or routers to switches.
- 2) In a standard connection, both the ends of the cable use the same wiring scheme, either T568A or T568B.
- 3) The pin configuration on one end matches the pin configuration on other end.
- 4) They are used for regular data transmission.

* Crossover Ethernet connection:

- 1) Crossover cables are used for connecting similar types of networks devices directly to each other.
- 2) In a crossover Ethernet connection, the wiring scheme on one end is different from wiring scheme on the other end.
- 3) the pin configuration is crossover i.e. pin 1 on one end connects to pin 3 on the other end and vice versa.
- 4) these cables are used to establish a direct network connection between two devices of the same type.

Q2
Date: 10/10/18



EXPERIMENT : 2

AIM : Implementation of Hamming code for detection and correction.

THEORY : Hamming codes are a type of error correcting code used in digital communication systems to detect and correct transmitted errors in transmitted data. They are particularly useful for single bit error correction and error detection. Here's an overview of the Hamming code detection and correction method.

Hamming codes add redundant bits (parity bits) to the original data to create a codeword.

The number of parity bits is determined by the formula $2^p \geq p + m + 1$, where m is the number of data bits, p is the number of parity bits.

- 1) Data encoding : For input message of k bits, r parity bits are added to the positions that are powers of 2 (1, 2, 4, 8, etc). The positions of these parity bits are calculated by using hamming representation. Each parity bit is responsible for checking results of data bit.
- 2) Parity calculation : Each parity bit calculates its value based on data bits that it carries. For eg: parity bit at position 1 checks all data bits that have a 1 in their least significant bits and so on. Parity bits are calculated using even parity, which means total numbers of 1s in group of bits should be even.
- 3) Error detection - During transmission if a bit gets flipped

due to noise , the parity bits will no longer match their calculated values. This discrepancy indicates an error.

4) Error correction : If an error is detected , the erroneous bits position can be determined by looking at parity bits that cover that position . By using the pattern of incorrect parity bits, the erroneous bits can be identified and corrected.

5) Decoding : The received data is compared to parity bits . If any discrepancies are found , error detection and correction are performed based on parity bit positions.

It's limitations, is it cannot perform correction for error in more than two bits or detect error in parity bits.

Problem:- 7 bit hamming code is received as 1011011 . Assume even parity and state whether the received code is correct or wrong , if wrong locate the bit in error.

Soln:- Received hc: D₇ D₆ D₅ P₄ D₃ P₂ P₁
1 0 1 1 0 1 1

Detecting error: Analyzing bits 1,3,5,7
we have P₁ D₃ D₅ D₇ \Rightarrow 1011 odd parity, error exists.
we put P₁ = 1

Step 2: Analyzing 2,3,6,7
P₂ D₃ D₆ D₇

1 0 0 1 \times even parity , no error exists.
 $\Rightarrow P_2 = 0$

Step 3 : Analyzing bits 4, 5, 6, 7

We have P_4, D_5, D_6, D_7

1 1 0 1 of odd parity
 error exists

We put $P_4 = 1$

Correcting error : error word $E = \begin{matrix} P_4 & P_2 & P_1 \\ 1 & 0 & 1 \end{matrix}$

decimal value $\epsilon = 5$ which shows that 5th bit
 is in error.

we write the correct word by simply inverting
 the 5th bit:-

\therefore correct-word = 1001011

✓
 Q 27/01/23
 A

```

option=int(input('Press 1 for generating hamming code \nPress 2 for finding error in hamming code\n\tEnter your choice:--\n'))

while(option>0 and option<3): # GENERATE HAMMING CODE

    if(option == 1):
        print('Enter the data bits')
        d=input()
        data=list(d)
        data.reverse()
        c,ch,j,r,h=0,0,0,0,[]

        while ((len(d)+r+1)>(pow(2,r))):
            r=r+1

        for i in range(0,(r+len(data))): 
            p=(2**c)

            if(p==(i+1)):
                h.append(0)
                c=c+1

            else:
                h.append(int(data[j]))
                j=j+1

        for parity in range(0,(len(h))): 
            ph=(2**ch)

            if(ph==(parity+1)):

                startIndex=ph-1
                i=startIndex
                toXor=[]

                while(i<len(h)):
```

```

block=h[i:i+ph]
toXor.extend(block)
i+=2*ph

for z in range(1,len(toXor)):
    h[startIndex]=h[startIndex]^toXor[z]
    ch+=1

h.reverse()
print('Hamming code generated would be:- ', end="")
print(int("".join(map(str, h))))


elif(option==2): # DETECT ERROR IN RECEIVED HAMMING CODE
    print('Enter the hamming code received')
    d=input()
    data=list(d)
    data.reverse()
    c,ch,j,r,error,h,parity_list,h_copy=0,0,0,0,[],[],[]

    for k in range(0,len(data)):
        p=(2**c)
        h.append(int(data[k]))
        h_copy.append(data[k])
        if(p==(k+1)):

            c=c+1

    for parity in range(0,(len(h))):
        ph=(2**ch)
        if(ph==(parity+1)):

            startIndex=ph-1

```

```

i=startIndex
toXor=[]

while(i<len(h)):
    block=h[i:i+ph]
    toXor.extend(block)
    i+=2*ph

for z in range(1,len(toXor)):
    h[startIndex]=h[startIndex]^toXor[z]
    parity_list.append(h[parity])
    ch+=1

parity_list.reverse()
error=sum(int(parity_list) * (2 ** i) for i, parity_list in enumerate(parity_list[::-1]))

if((error)==0):
    print('There is no error in the hamming code received')

elif((error)>=len(h_copy)):
    print('Error cannot be detected')

else:
    print('Error is in',error,'bit')

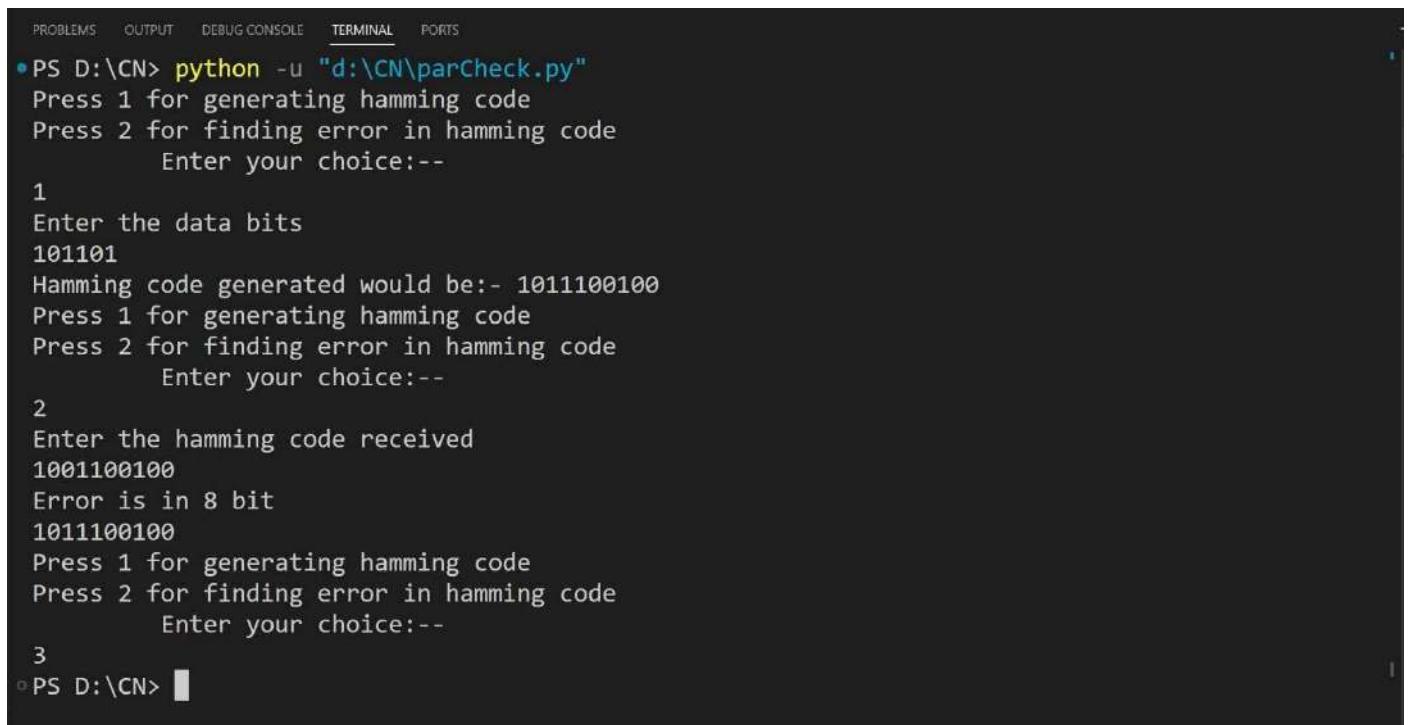
    if(h_copy[error-1]=='0'):
        h_copy[error-1]='1'

    elif(h_copy[error-1]=='1'):
        h_copy[error-1]='0'
        print('After correction hamming code is:- ')
        h_copy.reverse()
        print(int("".join(map(str, h_copy))))

```

```
else:  
    print('Option entered does not exist')  
  
option=int(input('Press 1 for generating hamming code \nPress 2 for finding error in hamming code\n\tEnter your choice:--\n'))
```

OUTPUT



The screenshot shows a terminal window with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS D:\CN> python -u "d:\CN\parCheck.py"  
Press 1 for generating hamming code  
Press 2 for finding error in hamming code  
Enter your choice:--  
1  
Enter the data bits  
101101  
Hamming code generated would be:- 1011100100  
Press 1 for generating hamming code  
Press 2 for finding error in hamming code  
Enter your choice:--  
2  
Enter the hamming code received  
1001100100  
Error is in 8 bit  
1011100100  
Press 1 for generating hamming code  
Press 2 for finding error in hamming code  
Enter your choice:--  
3  
PS D:\CN>
```

EXPERIMENT : 3

AIM: Implementation of CRC (Cyclic Redundancy Check)

THEORY: CRC is method of detecting error in communication channel. Given k bit frame or message, the transmitter generates a n -bit sequence known as frame check sequence (FCS), so that resulting frame, consisting of $(k+n)$ bits.

- Bit sequences can be written as polynomials with coefficient 0 and 1.
- Frame with k bits is considered as polynomial of degree $k-1$.
- The most significant bit is coefficient of x^{k-1} . The next bit is coefficient of x^{k-2} . Eg: The bit sequence 10011010 corresponds to the polynomial
- Sending and receiving message can be imagined as exchange of polynomials.
$$\begin{aligned}M(x) &= 1 \cdot x^7 + 0 \cdot x^6 + 2 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 0 \cdot x^0 \\&= x^7 + x^5 + x^4 + x^3\end{aligned}$$
- The Data Link Layer protocol specifies a generator polynomial $G(x)$. Generator polynomial is available for both sender and receiver side.

$G(x)$ is a polynomial of degree k .
Eg: $G(x) = x^3 + x^2 + x$

= 1101, then $x=3$

The degree of generator polynomial is degree 3.

The degree of generator polynomial is equal to number of bits minus one.

If for a frame, the CRC need to be calculated, no. of bits are appended to the frame. 'n' corresponds to degree of generator polynomial.

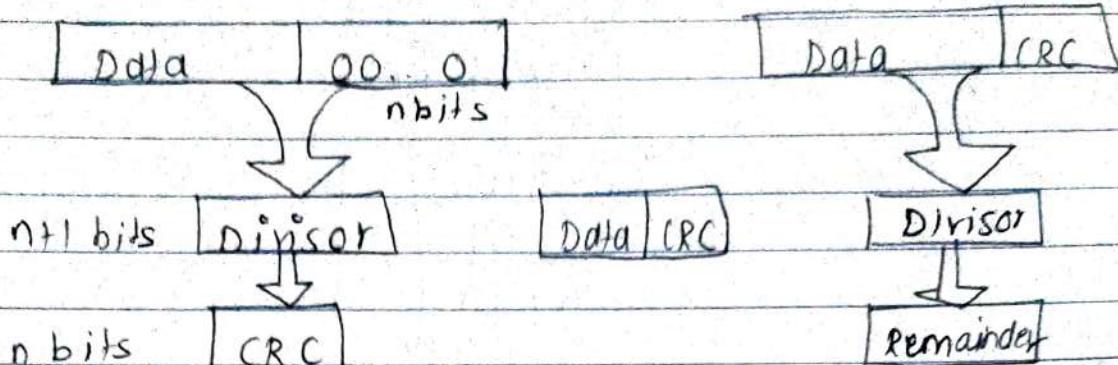
Generator polynomial 100110

Generator polynomial has 6 digits. There are zero bits are appended.

Frame with appended 0 bit 1010100000

- * Sender side (Generation of encoded Data from Data and Generator Polynomial (or key))
 - Binary data is first arranged by adding n zeroes in end of the data (n degree of generator polynomial).
 - Use modulo-2 binary division to divide binary data by key and store remainder of division.
 - Append remainder at end of the data to form encoded data and send the same.
- * Receiver side (check if there is any error introduced in transmission).
 - Perform modulo-2 division again if the remainder is 0, then there are no errors.

Diagram:



zero, accept,
non-zero, reject

SENDER

RECEIVER

Eg:-

$$\begin{array}{r}
 111101 \\
 1101) 100100000 \\
 \underline{1101} \downarrow | | | \\
 01000 | | | \\
 \underline{1101} \downarrow | | | \\
 01010 | | | \\
 \underline{1101} \downarrow | | | \\
 01110 | | | \\
 \underline{1101} \downarrow | | | \\
 001100 \\
 \underline{1101} \\
 0001 \rightarrow \text{CRC}
 \end{array}$$

∴ Transmitted code: 100100001

Consider error in bit 5.

Received code: 100110001

$$\begin{array}{r}
& \underline{\underline{11111}} \\
1101 & \underline{\underline{100110001}} \\
& \underline{\underline{1101}} | \quad | \quad | \\
& 01001 \\
& \underline{\underline{1101}} \downarrow \\
& 01001 \\
& \underline{\underline{1101}} \downarrow \\
& 01000 \\
& \underline{\underline{1101}} \downarrow \\
& 01010 \\
& \underline{\underline{1101}} \downarrow \\
& 01111 \\
& \underline{\underline{1101}} \\
& 0010
\end{array}$$

→ Remainder
not zero.

∴ Error has occurred in received bits.

Reception

```

def XOR(x, y):
    if x == y:
        return 0
    return 1

def flip(x):
    if x == 0:
        return 1
    return 0

def moduloDivision(data, dividend, divisor):
    for i in range(len(data)):
        if dividend[i] == 1:
            for j in range(len(divisor)):
                dividend[i + j] = XOR(dividend[i + j], divisor[j])
    return dividend

def displayCRC(data, dividend):
    print("CRC is :", end=" ")
    for i in range(len(data), len(dividend)):
        print(dividend[i], end="")
    print()

def displayCheckSum(data, dividend):
    print("Checksum code is :", end=" ")
    for i in range(len(data)):
        dividend[i] = int(data[i])
        print(dividend[i], end="")
    print()

def main():
    print("Enter data bits : ", end="")

```

```

data = input()
print("Enter check bits : ", end="")
check = input()
dividend = [0] * (len(data) + len(check) - 1)
divisor = [0] * len(check)

for i in range(len(data)):
    dividend[i] = int(data[i])

for i in range(len(check)):
    divisor[i] = int(check[i])

# Calculating remainder (CRC)
dividend = moduloDivision(data, dividend, divisor)

# Display remainder
displayCRC(data, dividend)

# Display checksum
displayCheckSum(data, dividend)

# Asking for a change in checksum
print("Do you want to put error bit(0/1) : ", end="")
choice = int(input())

if choice == 1:
    print("How many error bits do you want to change : ", end="")
    select = int(input())
    print("Enter the bit number you want to change : ", end="")
    change = input()
    for i in range(select):
        dividend[int(change[i])] = flip(dividend[int(change[i])])

```

```
dividend = moduloDivision(data, dividend, divisor)

displayCRC(data, dividend)

print("We see that the remainder is not 0. Hence data is corrupted!!")

else:

    print("CRC obtained at the receiver side is zero")

    print("Data sent without corruption")

if __name__ == "__main__":

    main()
```

OUTPUT

The screenshot shows a terminal window with the following interface elements:

- Top bar: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), PORTS.
- Terminal content:
 - PS D:\CN> python -u "d:\CN\CRC.py"
 - Enter data bits : 4
 - Enter check bits : 1011
 - CRC is : 000
 - Checksum code is : 4
 - Do you want to put error bit(0/1) : 1
 - How many error bits do you want to change : 1
 - Enter the bit number you want to change : 2
 - CRC is : 010
 - We see that the remainder is not 0. Hence data is corrupted!!
 - PS D:\CN>

EXPERIMENT : 4

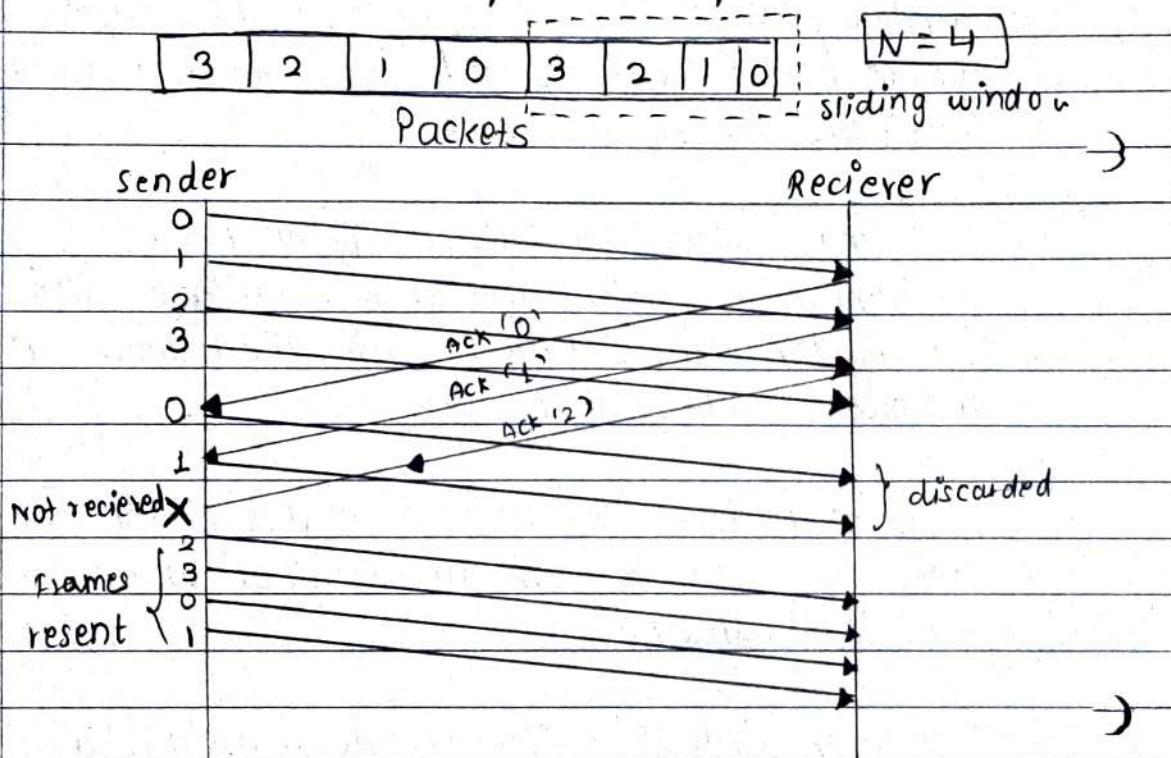
ATM: Simulation of Go-Back-N flow control algorithm

THEORY: The Go-Back-N protocol is a network protocol used in the data link layer of the OSI model to ensure reliable communication over a network. It is a sliding window protocol. Go-Back-N is designed for use in situations where packet loss is expected, such as in wireless or unreliable network environments.

1. Sliding Window: Go-Back-N employs a sliding window mechanism. The sender can transmit multiple packets before waiting for acknowledgements, while the receiver can accept and process packets out of order.
2. Sender Window: The sender maintains a fixed-size window that determines the maximum number of unacknowledged packets it can send.
3. Receiver Window: The receiver also maintains a window, indicating the expected sequence number of incoming packets.
4. Sequence Numbers: Each packet is assigned a sequence number, which helps in tracking and ordering packets.
5. Timeouts: The sender sets timeout for each transmitted packet. If an acknowledgement is known not received within the timeout period, the sender assumes the packet was lost and retransmits it.

Example: consider the Go Back N protocol where sender's window size is given as 4, which means sender can send up to 4 packets unacknowledged at a time.

Since the window size is 4, the sequence number given to the packets will be 0,1,2,3,0,1,2,3 and so on. Consider there are 8 packets, represented below:



In the above example, we can see that the sender sends 4 unacknowledged packets (0, 1, 2, 3). The sliding window shifts only if the sender receives an acknowledgement from receiver. We can see that sender receives acknowledgement for frame 0. Window slides further by sending next packet to receiver. Now, sender receives another acknowledgement for packet 1. Window slides further. This time sender does not receive any acknowledgement. Therefore, the packets present in the sliding window are retransmitted again. Whereas, the previously sent packets are discarded. Go back to Packet 2. This is Go-Back-N protocol.

```

def main():

    print("Enter window size: ")

    window_size = int(input())

    print("Enter total frames to be sent: ")

    total_frames = int(input())


    # Initializing array with data frames

    sender_frames = [i for i in range(total_frames)]


    # Displaying data frames

    for frame in sender_frames:

        print(frame, " | ", end="")

    print()


    # Displaying sender window

    print("Do you want to start sending frames (0/1): ")

    choice = int(input())

    print()


if choice == 1:

    ptr_on_window_left_sender = 0

    ptr_on_window_left_receiver = 0

    total_sent_frames = 0


    while ptr_on_window_left_sender < total_frames:

        # Sender side

        count = 0

        print("At Sender End:")

        for i in range(ptr_on_window_left_sender, total_frames):

            if count < window_size:

                print("Sent frame[", i + 1, "]")

                ptr_on_window_left_sender += 1

                total_sent_frames += 1

                count += 1

```

```

else:
    break

print()

# Receiver side
print("At Receiver end:")
j = 0
count = 0
for i in range(ptr_on_window_left_receiver, total_frames):
    if count < window_size:
        print("Did you receive frame[", i + 1, "] (y/n) : ")
        y_n = input()
        if y_n == 'n':
            print("Frames will again be sent from frame no.", i + 1)
            print()
            ptr_on_window_left_sender = i
            break
    else:
        j += 1
        ptr_on_window_left_receiver += 1
        count += 1
    else:
        break

if j == window_size:
    print("All Frames from this window sent without errors. Sending next frames...")
    print()

print()
print("All frames are sent.")
print("Total no. of frames sent including retransmission is", total_sent_frames)

if __name__ == "__main__":

```

main()

OUTPUT

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

• PS D:\CN> python -u "d:\CN\GoBackN.py"
Enter window size:
4
Enter total frames to be sent:
8
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
Do you want to start sending frames (0/1):
1

At Sender End:
Sent frame[ 1 ]
Sent frame[ 2 ]
Sent frame[ 3 ]
Sent frame[ 4 ]

At Receiver end:
Did you receive frame[ 1 ] (y/n) :
y
Did you receive frame[ 2 ] (y/n) :
y
Did you receive frame[ 3 ] (y/n) :
y
Did you receive frame[ 4 ] (y/n) :
y
All Frames from this window sent without errors. Sending next frames...
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

At Sender End:
Sent frame[ 5 ]
Sent frame[ 6 ]
Sent frame[ 7 ]
Sent frame[ 8 ]

At Receiver end:
Did you receive frame[ 5 ] (y/n) :
y
Did you receive frame[ 6 ] (y/n) :
y
Did you receive frame[ 7 ] (y/n) :
n
Frames will again be sent from frame no. 7

At Sender End:
Sent frame[ 7 ]
Sent frame[ 8 ]

At Receiver end:
Did you receive frame[ 7 ] (y/n) :
y
Did you receive frame[ 8 ] (y/n) :
y

All frames are sent.
Total no. of frames sent including retransmission is 10
• PS D:\CN> █
```

EXPERIMENT : 5

AIM: Build a simple network topology and configure it for static routing protocol using packet tracer. Setup a network and configure IP addressing, subnetting, masking.

THEORY: Network topology refers to the physical or logical layout of devices and connections within a network. Different networks topologies are suited for various applications and have unique advantages and disadvantages. Here are a few different network topologies along with small examples of their applications.

Star Topology: In a star topology, all devices are connected to a central hub or switch. Devices do not connect directly to each other.

Star topology is commonly used in home networks where all devices (computers, printers, smart devices) connect to a central router or switch. This topology is straightforward to set up and manage.

Bus Topology: In a bus topology, all devices share a single communication line (bus). Data is transmitted down the bus, and devices can "tap-in" to access the data.

Bus topologies were historically used in early ethernet networks. While they are less common today, they can still be found in applications like industrial control systems.

Ring Topology: In a ring topology, each device is connected to exactly two other devices, forming a closed loop.

Data travels in one direction around the ring.

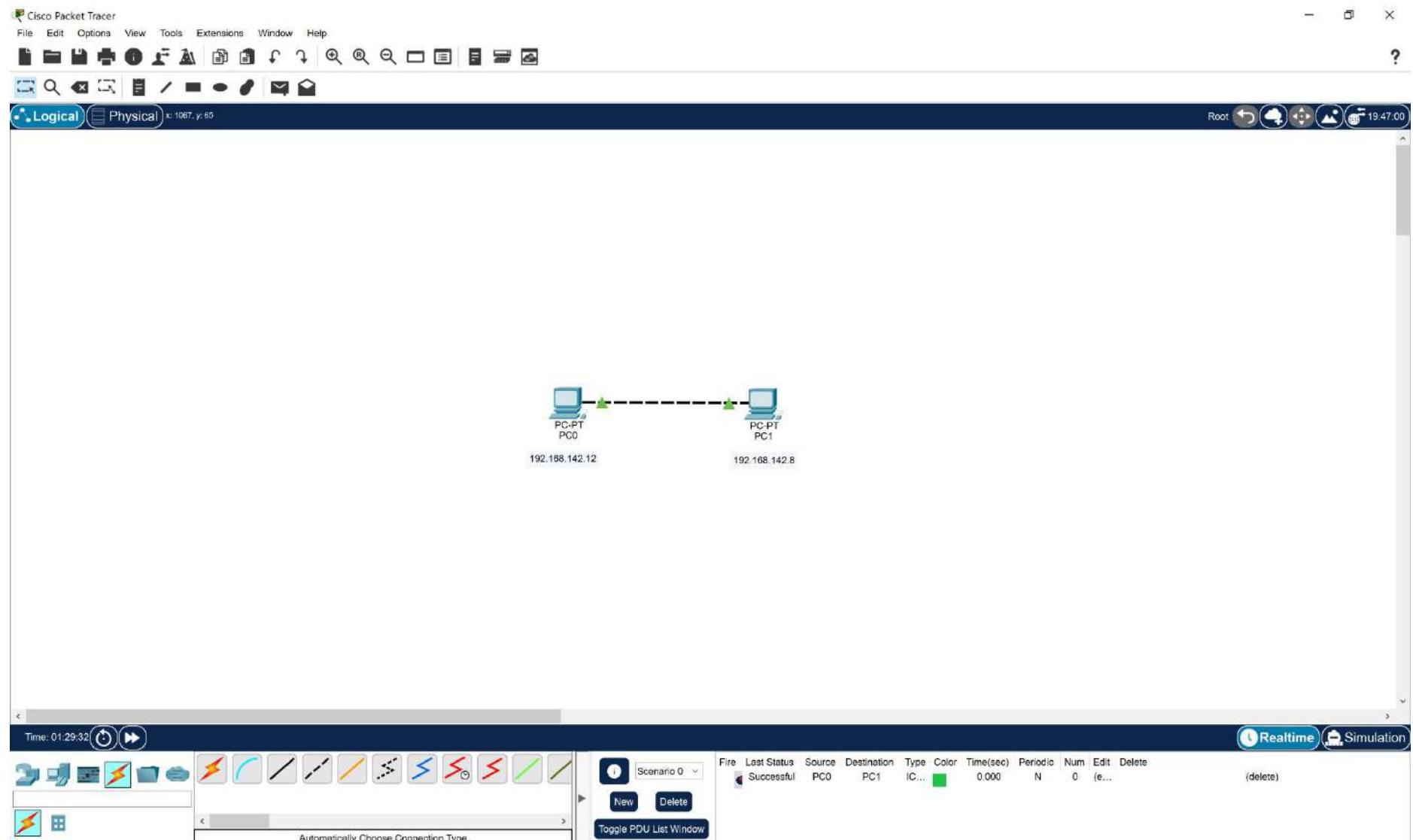
Ring topologies are less common in modern LANs but are used in specific applications like fibre optic networks. Token ring networks used this topology in the past.

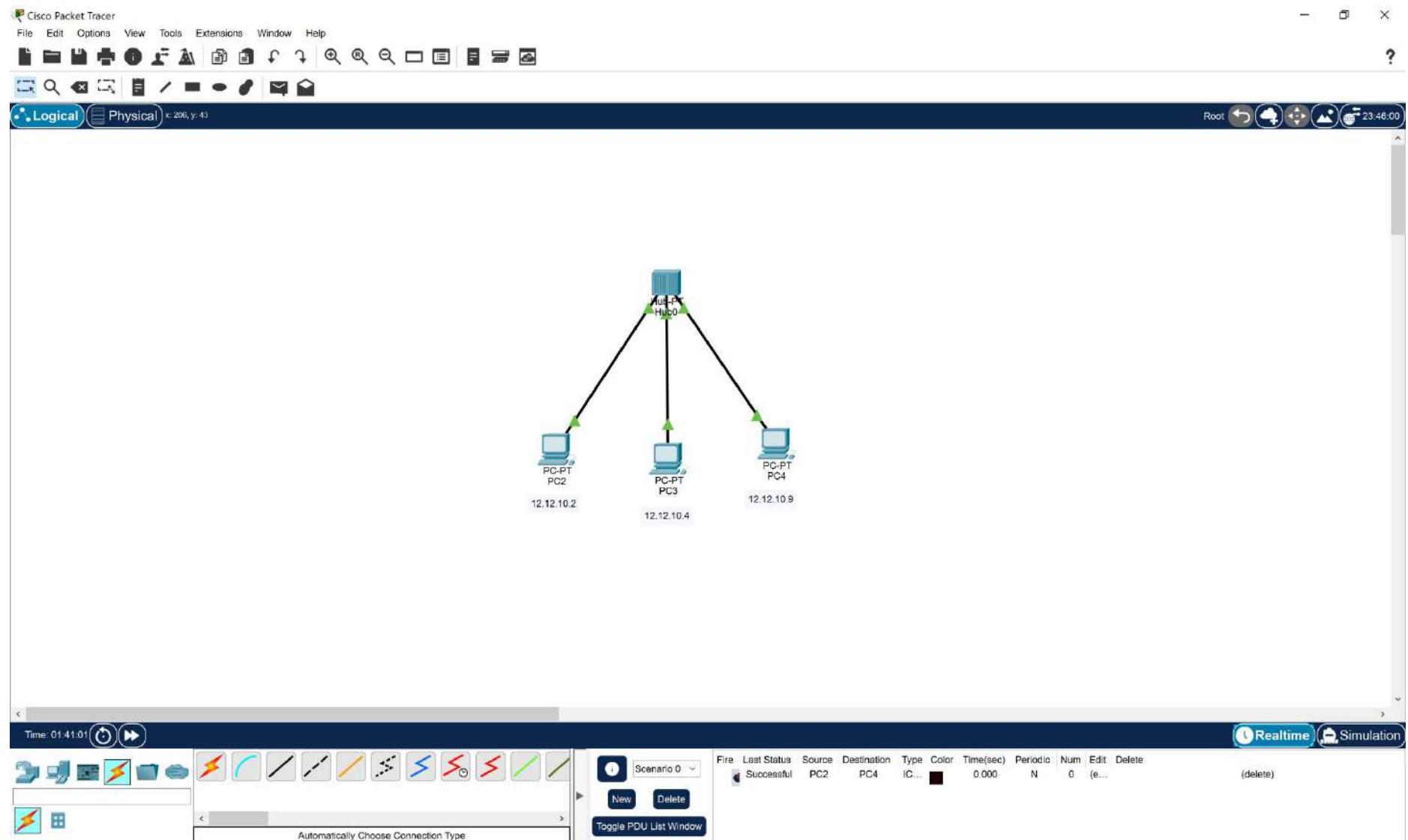
Mesh topology: In a mesh topology, every device is connected to every other device, creating multiple plans/ paths for data to travel.

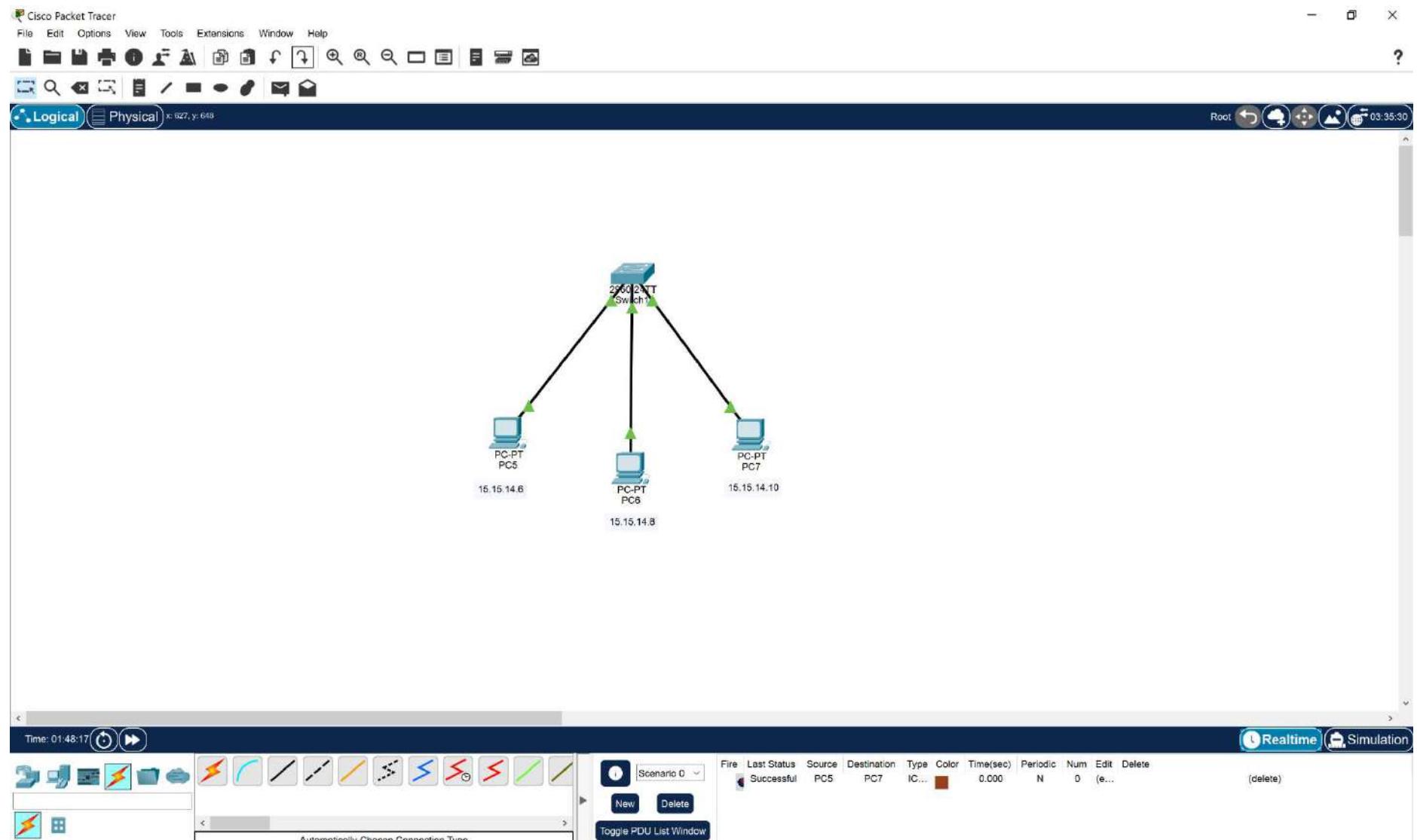
Mesh topologies are often used in critical applications, like data centres, where redundancy and fault tolerance are crucial. Each device can communicate directly with any other, enhancing reliability.

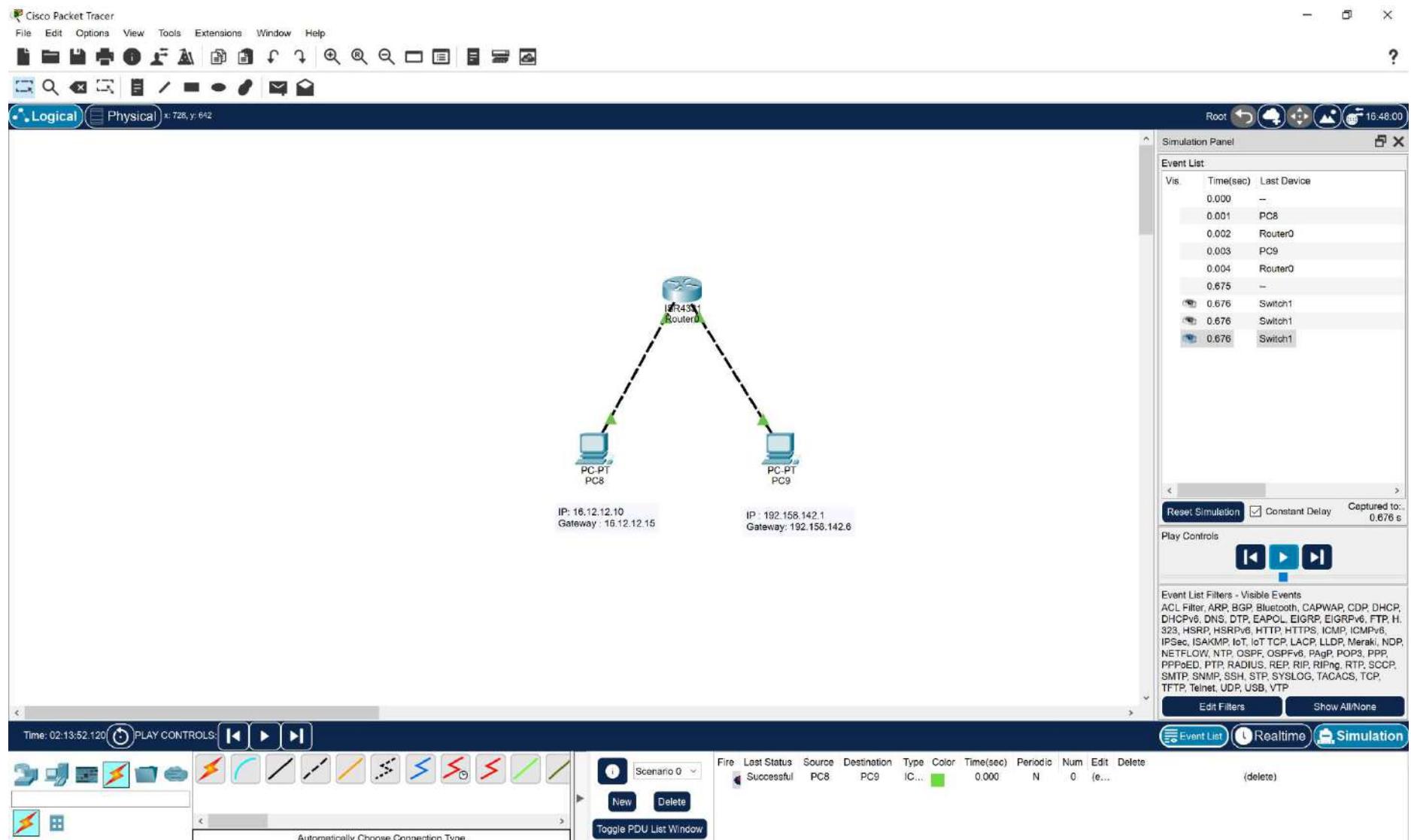
Hybrid topology: A hybrid topology combines two or more different topologies into a single network. For example, a network might have a star topology in one office and a ring topology in another office connected through a router. Hybrid topologies allow organizations to tailor their network to specific needs. They are commonly found in large enterprises with diverse networking requirements.

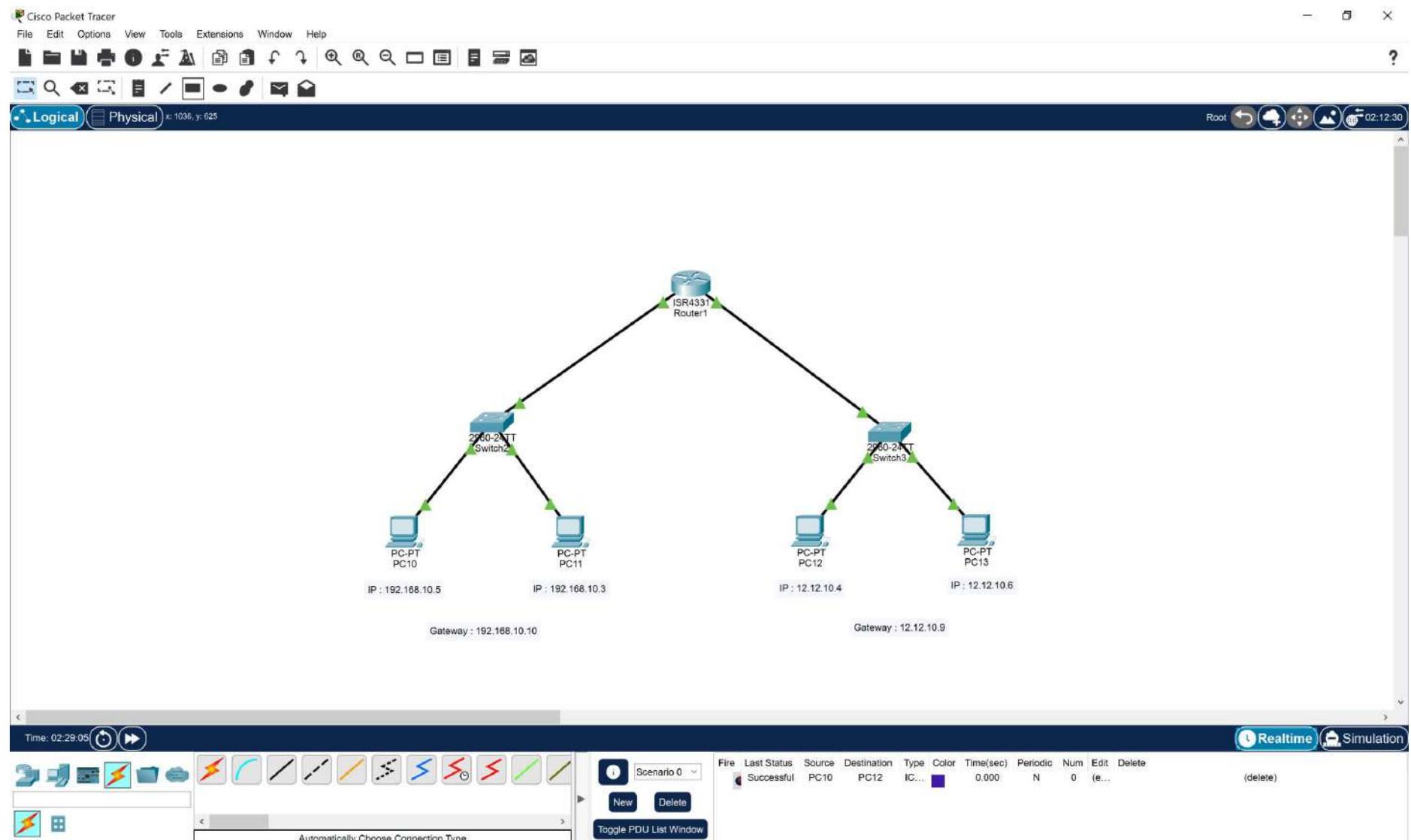
Tree (Hierarchical) topology: In a tree topology, multiple stars are connected. A central hub from each star connects to a main central hub. Tree topologies are often used in large-scale networks, such as corporate networks with multiple branches. Each branch office may have a star topology and they all connect to the main office using a hierarchical structure.











EXPERIMENT NO: 6

AIM: Design VPN and configure RIP/OSPF using Packet Tracer.

THEORY:

Routing Information Protocol (RIP) is one of the oldest distance-vector routing protocols used in computer networks. Its primary purpose is to determine the best path for data packets to travel from one network segment to another within an IP based network.

Routing: Refers to the process of forwarding data packets from one network to another. Routers are devices responsible for this task.

Distance-Vector-Protocol: RIP uses a distance-vector algorithm to determine the best path for packet transmission. Each router in the network maintains a routing table, which contains information about the distance (number of hops) and direction to reach various network destinations.

Hop Count: RIP uses hop count as a metric to determine the best path. A hop is a passage through one router to another. RIP routers exchange routing information periodically, updating their routing tables based on the number of hops to reach different networks.

Routing Updates: RIP routers send routing updates at regular intervals to share information about the networks they know. These updates contain the network address and associated hop counts.

Use Case: RIP is typically used in small to medium-size networks where simplicity is more important than rapid convergence.

Virtual Local Area Networks (VLANs) :

VLANs are a method of segmenting a physical network into multiple logical networks, improving networks efficiency and security.

Logical Segmentation : VLAN's allow you to logically divide a physically LAN into multiple isolated networks. These networks behave as if they are on separate physical switches, even if they share the same physical infrastructure.

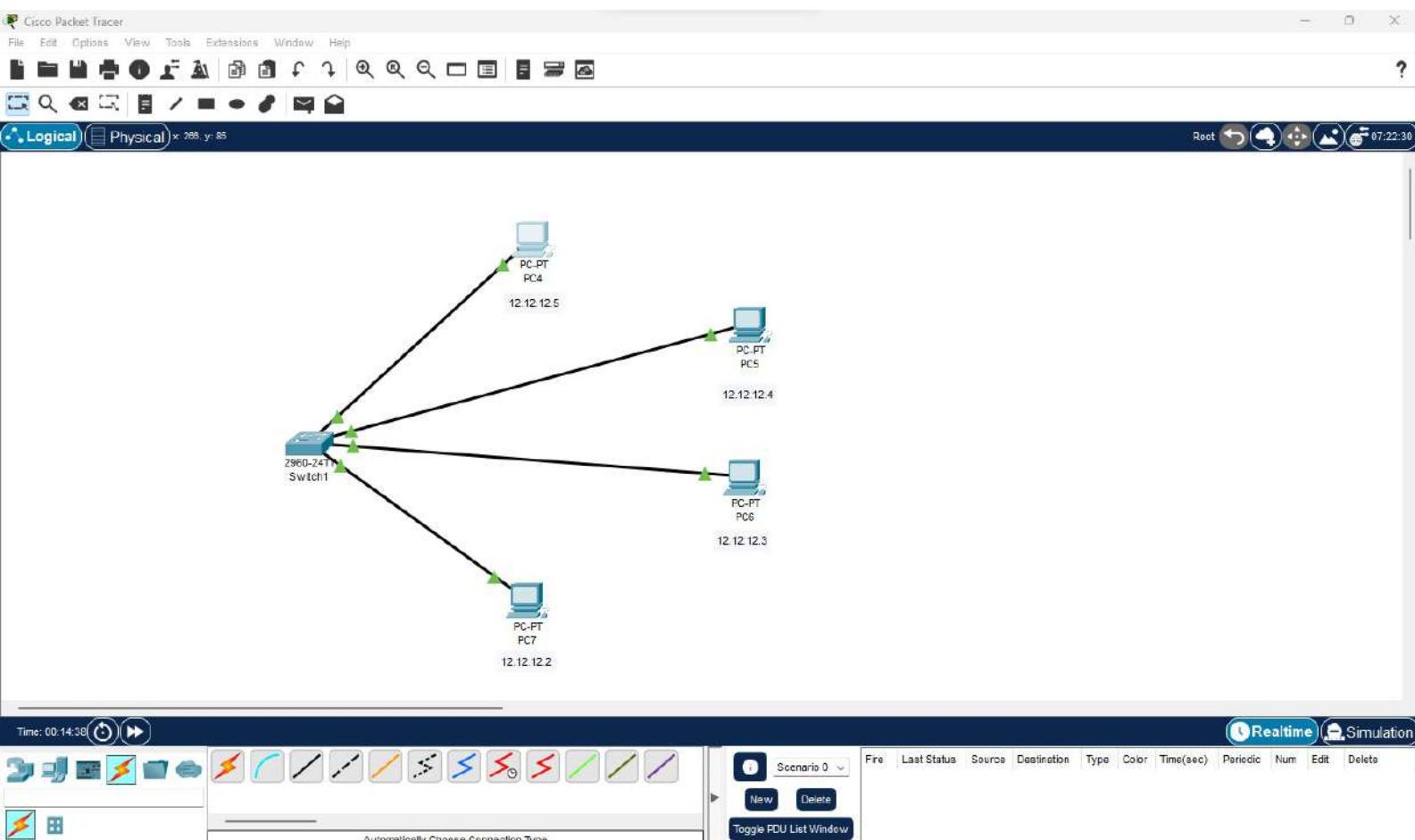
Flexibility : VLAN's are flexible and can be configured based on different criteria, such as department, function, or security requirements. VLAN configuration is typically done at the switch level.

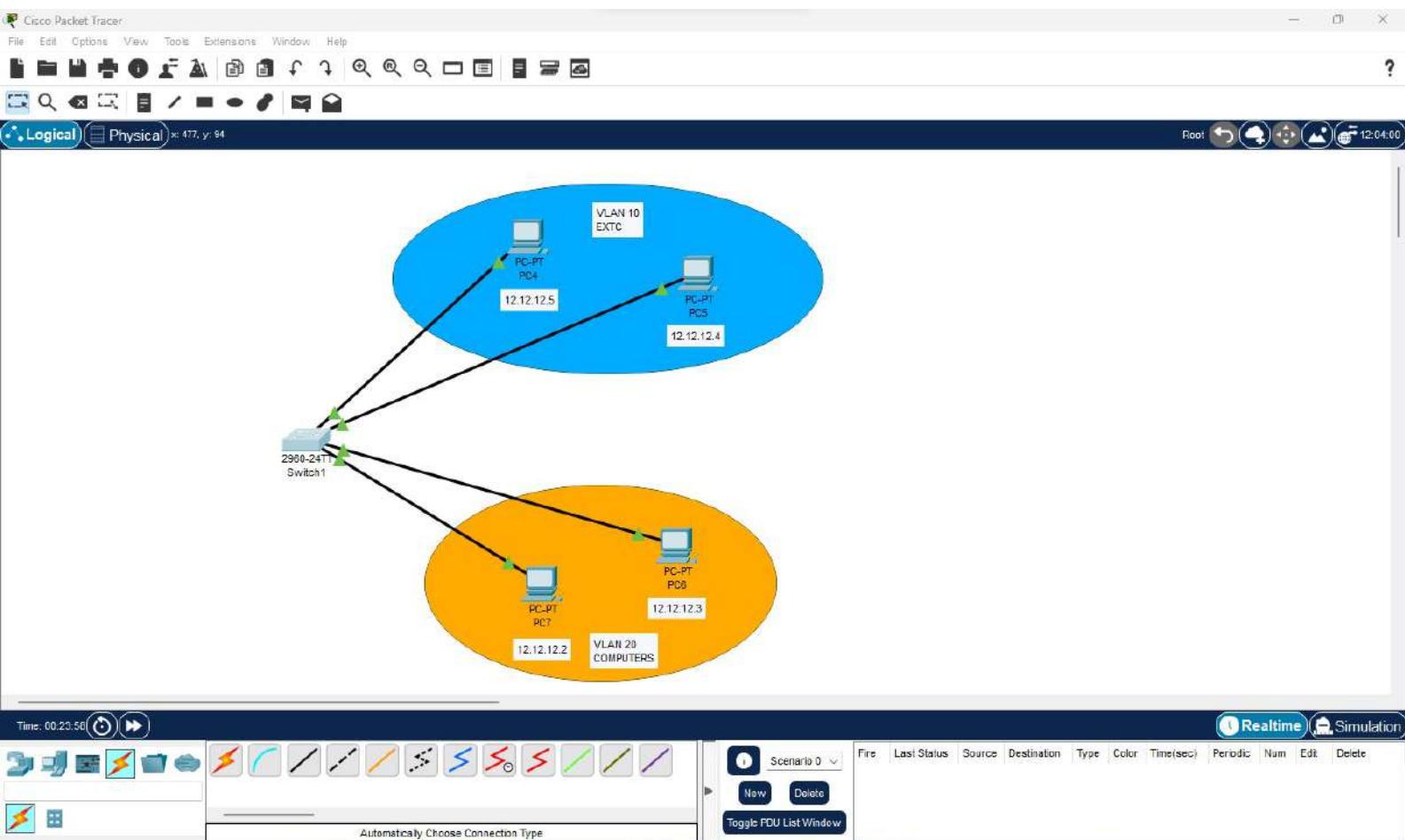
Security : VLANs enhance network security by isolating sensitive data or services. For example, a company might create separate VLANs for guest WiFi, employee workstations, and servers to isolate traffic and enhance security.

Scalability : VLANs are highly scalable, making them suitable for large and complex network environments.

Use Cases : VLANs are commonly used in enterprise networks, data centers, and environments where network segmentation is essential.

Inter-VLAN Routing : For communication between VLANs a router or Layer 3 switch is required. This allows controlled communication between different VLANs while maintaining isolation.





Physical Config CLI Attributes

GLOBAL
Settings
Algorithm Settings
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/1
FastEthernet0/2
FastEthernet0/3
FastEthernet0/4
FastEthernet0/5
FastEthernet0/6
FastEthernet0/7
FastEthernet0/8
FastEthernet0/9
FastEthernet0/10
FastEthernet0/11
FastEthernet0/12

VLAN Configuration	
VLAN Number	20
VLAN Name	COMPUTERS
	<input type="button" value="Add"/> <input type="button" value="Remove"/>
VLAN No	VLAN Name
1	default
10	EXTC
20	COMPUTERS
1002	fddi-default
1003	token-ring-default
1004	fddinet-default
1005	trnet-default

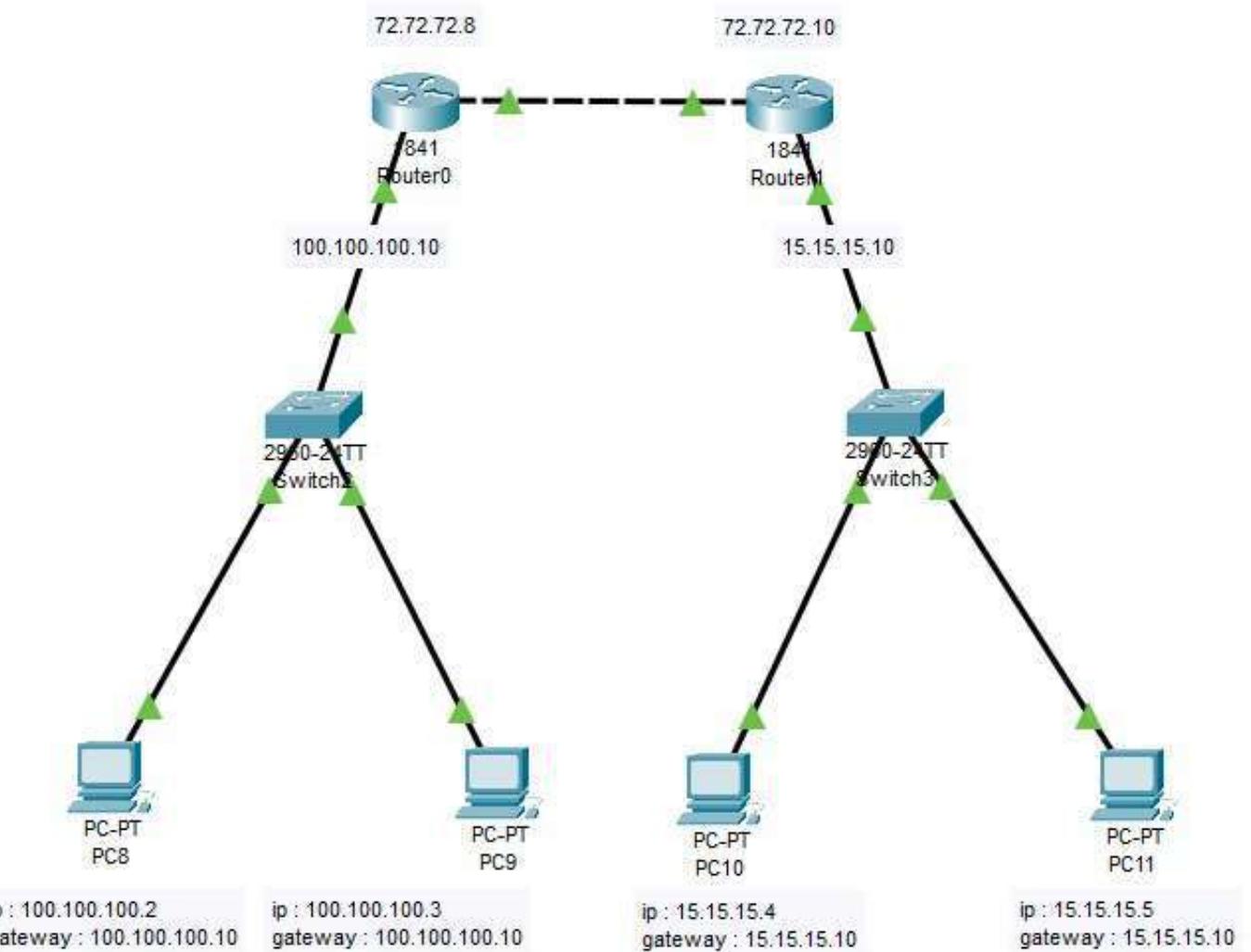
Equivalent IOS Commands

```

Switch>enable
Switch#
Switch#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#
Switch(config)#
Switch(config)#
Switch(config)#vlan 10
Switch(config-vlan)# name EXTC
Switch(config-vlan)#vlan 20
Switch(config-vlan)# name COMPUTERS
Switch(config-vlan)#

```

 Top



```
C:\>ping 15.15.15.4

Pinging 15.15.15.4 with 32 bytes of data:

Reply from 100.100.100.10: Destination host unreachable.

Ping statistics for 15.15.15.4:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>ping 15.15.15.4
```

PC6

Physical Config Desktop Programming Attributes

Command Prompt

```
Link-Local IPv6 Address ..... ::  
IPv6 Address..... ::  
IPv4 Address..... 0.0.0.0  
Subnet Mask..... 0.0.0.0  
Default Gateway..... 0.0.0.0  
C:\>ping 12.12.12.5  
  
Pinging 12.12.12.5 with 32 bytes of data:  
  
Reply from 12.12.12.5: bytes=32 time<1ms TTL=128  
Reply from 12.12.12.5: bytes=32 time=1ms TTL=128  
Reply from 12.12.12.5: bytes=32 time<1ms TTL=128  
Reply from 12.12.12.5: bytes=32 time<1ms TTL=128  
  
Ping statistics for 12.12.12.5:  
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 0ms, Maximum = 6ms, Average = 1ms  
  
C:\>ping 12.12.12.4  
  
Pinging 12.12.12.4 with 32 bytes of data:  
  
Reply from 12.12.12.4: bytes=32 time<1ms TTL=128  
Reply from 12.12.12.4: bytes=32 time=1ms TTL=128  
Reply from 12.12.12.4: bytes=32 time<1ms TTL=128  
Reply from 12.12.12.4: bytes=32 time<1ms TTL=128  
  
Ping statistics for 12.12.12.4:  
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 0ms, Maximum = 1ms, Average = 0ms  
  
C:\>ping 12.12.12.2  
  
Pinging 12.12.12.2 with 32 bytes of data:  
  
Reply from 12.12.12.2: bytes=32 time<1ms TTL=128  
Reply from 12.12.12.2: bytes=32 time=1ms TTL=128  
Reply from 12.12.12.2: bytes=32 time<1ms TTL=128  
Reply from 12.12.12.2: bytes=32 time<1ms TTL=128  
  
Ping statistics for 12.12.12.2:  
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 0ms, Maximum = 1ms, Average = 0ms  
  
C:\>
```

PC6

Physical Config Desktop Programming Attributes

Command Prompt

```
Cisco Packet Tracer PC Command Line 1.0
C:>ipconfig

FastEthernet0 Connection (default port)

Connection-specific DNS Suffix...:
Link-local IPv6 Address.....:: FE80::202:4AFF:FE0A:7B1
IPv6 Address.....:: ::1
IPv4 Address.....: 12.12.12.3
Subnet Mask.....: 255.0.0.0
Default Gateway.....: 0.0.0.0

Bluetooth Connection:

Connection-specific DNS Suffix...:
Link-local IPv6 Address.....:: ::1
IPv6 Address.....:: ::1
IPv4 Address.....: 0.0.0.0
Subnet Mask.....: 0.0.0.0
Default Gateway.....: ::1
0.0.0.0

C:>ping 12.12.12.5

Pinging 12.12.12.5 with 32 bytes of data:
Reply from 12.12.12.5: bytes=32 time=6ms TTL=128
Reply from 12.12.12.5: bytes=32 time=1ms TTL=128
Reply from 12.12.12.5: bytes=32 time<1ms TTL=128
Reply from 12.12.12.5: bytes=32 time<1ms TTL=128

Ping statistics for 12.12.12.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 6ms, Average = 1ms

C:>ping 12.12.12.4

Pinging 12.12.12.4 with 32 bytes of data:
Reply from 12.12.12.4: bytes=32 time<1ms TTL=128
Reply from 12.12.12.4: bytes=32 time=1ms TTL=128
Reply from 12.12.12.4: bytes=32 time=1ms TTL=128
Reply from 12.12.12.4: bytes=32 time=1ms TTL=128

Ping statistics for 12.12.12.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
```

Top

Physical Config Desktop Programming Attributes

Command Prompt X

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 100.100.100.3

Pinging 100.100.100.3 with 32 bytes of data:

Reply from 100.100.100.3: bytes=32 time<1ms TTL=128

Ping statistics for 100.100.100.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 15.15.15.4

Pinging 15.15.15.4 with 32 bytes of data:

Reply from 100.100.100.10: Destination host unreachable.

Ping statistics for 15.15.15.4:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

C:\>

EXPERIMENT : 7

AIM : Write a program to implement IPv4 addressing concept along with subnet masking.

THEORY : IPv4 (Internet Protocol version 4) addressing is a fundamental concept in network that involves assigning numerical labels to devices on a network. These addresses are used for identification and communication between devices in the IP network. IPv4 addresses are represented as a series of 4 decimal numbers, each ranging from 0 to 255, separated by periods (.). Each of these numbers is an octet, representing 8 bits of the 32 bit IPv4 address.

IP address of a system is a 32 bit unique address. It is divided into two parts: The network portion and the host portion. The boundary between these two portions is determined by a subnet mask.

Subnet Mask : The subnet mask is a 32-bit number that defines the boundary between the network and host portions of IPv4 address. It consists of a series of contiguous 1s followed by a series of contiguous 0s. The 1s indicate the network bits and the 0s indicate the host bits.

Network ID and Host ID : The network ID portion of the address identifies the network to which a device belongs, while the host portion identifies the specific device within that network. Devices within the same network share the same Net ID.

Classes of IPv4 address:

		8 bits		
A	0	Network	Host	31 1.0.0.0 to 127.255.255.255
B	10	Network	Host	31 128.0.0.0 to 191.255.255.255
C	110	Network	Host	31 192.0.0.0 to 223.255.255.255
D	1110	Multicast address		31 224.0.0.0 to 239.255.255.255
E	1111	Reserved for future		31 240.0.0.0 to 255.255.255.255

Example:- Give the class as well as subnet mask for IP: 200.1.2.10 divided into 2 subnets.

Given: 200.1.2.10 : Network address: 200.1.2.0
The given IP is Class C IP address.

Network ID: 200.1.2.0

Host ID: 10.

To get the subnet mask for above IP, make all bits of network address as 1 and make all bits of Host ID as 0.

Subnet mask : 255.255.255.0

1st Subnet: Address: 200.1.2.0

Range: 200.1.2.0 - 200.1.2.127

2nd Subnet: Address: 200.1.2.128

Range: 200.1.2.128 - 200.1.2.255

```

import math

def findClass(ip):
    if 0 <= ip[0] <= 127:
        print("Network Address is : ", ip[0])
        print('No. of IP addresses possible : ', 2 ** 24)
        return "A", '255.0.0.0'
    elif 128 <= ip[0] <= 191:
        ip = [str(i) for i in ip]
        print("Network Address is : ", ".".join(ip[0:2]))
        print('No. of IP addresses possible : ', 2 ** 16)
        return "B", '255.255.0.0'
    elif 192 <= ip[0] <= 223:
        ip = [str(i) for i in ip]
        print("Network Id is : ", ".".join(ip[0:3]))
        print('No. of IP addresses possible : ', 2 ** 8)
        return "C", '255.255.255.0'
    elif 224 <= ip[0] <= 239:
        print("In this Class, IP address is not divided into Network and Host ID")
        return "D"
    else:
        print("In this Class, IP address is not divided into Network and Host ID")
        return "E"

def Subnetting(ip, num, className, ip_addresses):
    temp = 0
    if className == "A":
        place2 = ip_addresses / (256 ** 2)
        for i in range(num):

```

```

print(f"Subnet {i} => ")
print(temp)
print("Subnet Address : ", ip[0] + '.' + str(temp) + '.0' + '.0')
temp += int(place2)
print("Broadcast address : ", ip[0] + '.' + str(temp - 1) + '.255' + '.255')
print("Valid range of host IP address : ", ip[0] + '.' + str(temp - int(place2)) + '.' + '0' +
'.1' + '\t-\t' + ip[0] + '.' + str(temp - 1) + '.254' + '.254')
print()

elif className == "B":
    place2 = ip_addresses / 256
    for i in range(num):
        print(f"\nSubnet {i} => ")
        print("Subnet Address : ", ".".join(ip[0:2]) + '.' + str(temp) + '.0')
        temp += int(place2)
        print("Broadcast address : ", ".".join(ip[0:2]) + '.' + str(temp - 1) + '.255')
        print("Valid range of host IP address : ",".".join(ip[0:2]) + '.' + str(temp - int(place2)) +
'.1\t-\t' + ".".join(ip[0:2]) + '.' + str(temp - 1) + '.254')
        print()

elif className == "C":
    for i in range(num):
        print(f"\nSubnet {i} => ")
        print("Subnet Address : ", ".".join(ip[0:3]) + '.' + str(temp))
        temp += int(ip_addresses)
        print("Broadcast address : ", ".".join(ip[0:3]) + '.' + str(temp - 1))
        print("Valid range of host IP address : ",".".join(ip[0:3]) + '.' + str(temp -
int(ip_addresses) + 1) + '\t-\t' + ".".join(ip[0:3]) + '.' + str(temp - 2))
        print()

else:
    print("In this Class, IP address is not divided into Network and Host ID")

```

```

def subnetmask(num, network_mask):
    var = '1' * int(math.log(num, 2))
    var1 = '0' * (8 - int(math.log(num, 2)))
    binary_num = var + var1
    network_mask = network_mask.split('.')
    network_mask = [i for i in network_mask if i != '0']
    network_mask.append(str(int(binary_num, 2)))
    while len(network_mask) < 5:
        network_mask.append('0')
    print('Subnet Mask - ', '.'.join(network_mask[0:4]))

```

```

ip = input("Enter the IP address : ")
ip = ip.split(".")
ip = [int(i) for i in ip]
lst = findClass(ip)
networkClass = lst[0]
print("Given IP address belongs to class : ", networkClass)
ip = [str(i) for i in ip]
network_mask = lst[1]
print('Network Mask : ', network_mask)
num_subnet = int(input('\nNo. of subnets (power of 2) : '))
num_ip = int(2 ** (8 * (68 - ord(networkClass))) / num_subnet)
print('The no. of bits in the subnet id : ', int(math.log(num_subnet, 2)))
if ord(networkClass) < 68:
    print('Total no. of IP addresses possible in each subnet : ', num_ip)
Subnetting(ip, num_subnet, networkClass, num_ip)
subnetmask(num_subnet, network_mask)

```

OUTPUT:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- PS D:\CN> `python -u "d:\CN\ipv4.py"`
Enter the IP address : 192.37.58.12
Network Id is : 192.37.58
No. of IP addresses possible : 256
Given IP address belongs to class : C
Network Mask : 255.255.255.0

No. of subnets (power of 2) : 4
The no. of bits in the subnet id : 2
Total no. of IP addresses possible in each subnet : 64

Subnet 0 =>
Subnet Address : 192.37.58.0
Broadcast address : 192.37.58.63
Valid range of host IP address : 192.37.58.1 - 192.37.58.62

Subnet 1 =>
Subnet Address : 192.37.58.64
Broadcast address : 192.37.58.127
Valid range of host IP address : 192.37.58.65 - 192.37.58.126

Subnet 2 =>
Subnet Address : 192.37.58.128
Broadcast address : 192.37.58.191
Valid range of host IP address : 192.37.58.129 - 192.37.58.190

Subnet 3 =>
Subnet Address : 192.37.58.192
Broadcast address : 192.37.58.255
Valid range of host IP address : 192.37.58.193 - 192.37.58.254

Subnet Mask - 255.255.255.192

- PS D:\CN> █

EXPERIMENT - 8

AIM: Use basic networking commands in linux (ping, traceroute, nslookup, netstat, ARP, RARP, ip, ipconfig, dig, route)

THEORY :

- ifconfig: This command is utilized in network inspection, initializing the interface, enabling disabling ip address and configuring an interface with an IP address.

Syntax: ifconfig

- ping: It is used for network troubleshooting . It checks network connectivity between two different nodes.

Syntax: ping <destination>

- traceroute: It identifies the delay and identifies or decides pathway to our target.

Syntax: traceroute <destinations>

- netstat: It is a short network statistics . It displays various network related information such as network connections, routing tables, interface statistics, multicast membership, etc.

→ netstat -a : Shows the state of all the sockets.

→ netstat -at : List all TCP ports.

→ netstat -au : List all UDP ports.

→ netstat -l : Lists only the listening ports.

→ netstat -s : Lists statistics for all ports.

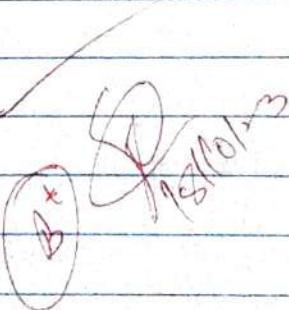
• nslookup: It is an old edition of dig command. It is useful for getting information from DNS server. It is also used to troubleshoot DNS related problems.
Syntax: nslookup <domain-name>

• dig: The Domain Information Groper is the improved version of nslookup. It is used for verifying and troubleshooting DNS problems and to perform DNS lookup.

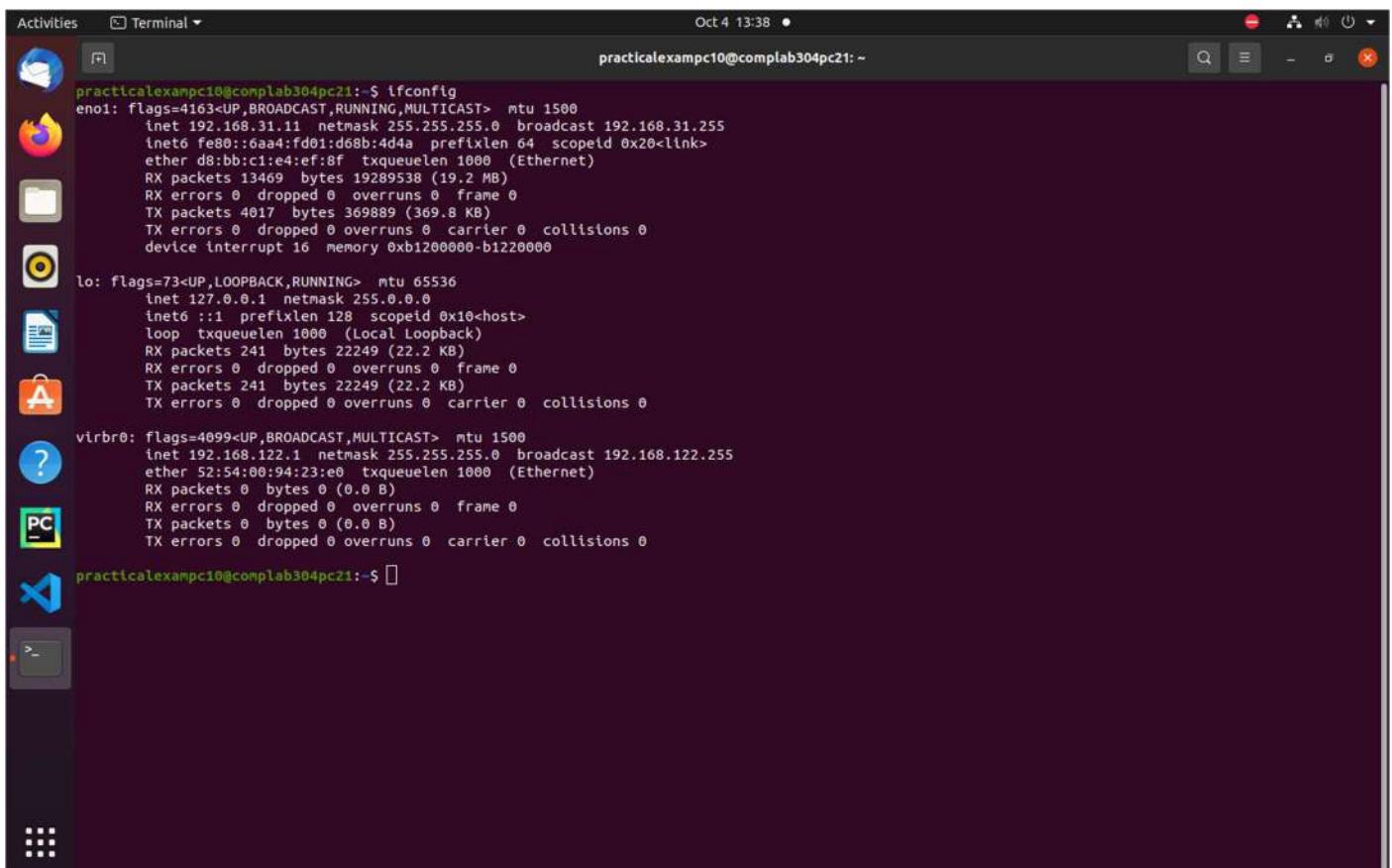
• arp: Address Resolution Protocol is used to restore/resolve the ip address of a system to mac address and hence it works between Data Link Layer and Network Layer.

• rarp: Reverse Address Resolution Protocol is used to resolve the mac address of a system to ip address and hence it works between Data-Link layer and Network Layer.

• ip: It is the updated and latest version of ifconfig command. The command provides the information of every network, such as ifconfig. It can also be used to get information about a particular interface. It is also used to show or manipulate routing devices and tunnels.



1. ifconfig



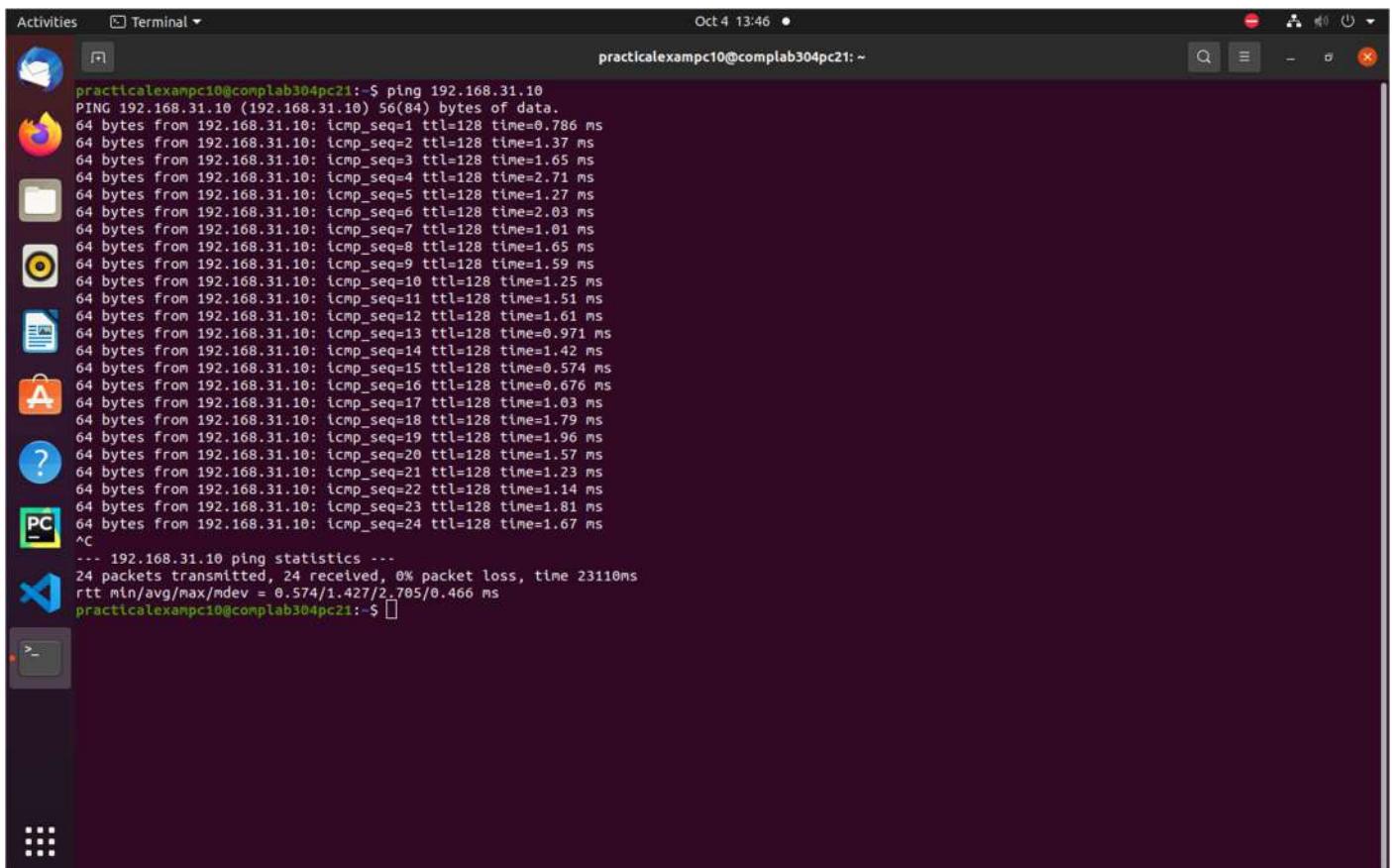
```
practicalexamplepc10@complab304pc21: ~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.31.11 netmask 255.255.255.0 broadcast 192.168.31.255
        inet6 fe80::6aa4:fd01:d68b:4d4a prefixlen 64 scopeid 0x20<link>
          ether d8:bb:c1:e4:ef:8f txqueuelen 1000 (Ethernet)
            RX packets 13469 bytes 19289538 (19.2 MB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 4017 bytes 369889 (369.8 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
          device interrupt 16 memory 0xb1200000-b1220000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
            RX packets 241 bytes 22249 (22.2 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 241 bytes 22249 (22.2 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255
          ether 52:54:00:94:23:e0 txqueuelen 1000 (Ethernet)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

practicalexamplepc10@complab304pc21: ~$
```

2. ping without count



```
practicalexamplepc10@complab304pc21: ~$ ping 192.168.31.10
PING 192.168.31.10 (192.168.31.10) 56(84) bytes of data.
64 bytes from 192.168.31.10: icmp_seq=1 ttl=128 time=0.786 ms
64 bytes from 192.168.31.10: icmp_seq=2 ttl=128 time=1.37 ms
64 bytes from 192.168.31.10: icmp_seq=3 ttl=128 time=1.65 ms
64 bytes from 192.168.31.10: icmp_seq=4 ttl=128 time=2.71 ms
64 bytes from 192.168.31.10: icmp_seq=5 ttl=128 time=1.27 ms
64 bytes from 192.168.31.10: icmp_seq=6 ttl=128 time=2.03 ms
64 bytes from 192.168.31.10: icmp_seq=7 ttl=128 time=1.01 ms
64 bytes from 192.168.31.10: icmp_seq=8 ttl=128 time=1.65 ms
64 bytes from 192.168.31.10: icmp_seq=9 ttl=128 time=1.59 ms
64 bytes from 192.168.31.10: icmp_seq=10 ttl=128 time=1.25 ms
64 bytes from 192.168.31.10: icmp_seq=11 ttl=128 time=1.51 ms
64 bytes from 192.168.31.10: icmp_seq=12 ttl=128 time=1.61 ms
64 bytes from 192.168.31.10: icmp_seq=13 ttl=128 time=0.971 ms
64 bytes from 192.168.31.10: icmp_seq=14 ttl=128 time=1.42 ms
64 bytes from 192.168.31.10: icmp_seq=15 ttl=128 time=0.574 ms
64 bytes from 192.168.31.10: icmp_seq=16 ttl=128 time=0.676 ms
64 bytes from 192.168.31.10: icmp_seq=17 ttl=128 time=1.03 ms
64 bytes from 192.168.31.10: icmp_seq=18 ttl=128 time=1.79 ms
64 bytes from 192.168.31.10: icmp_seq=19 ttl=128 time=1.96 ms
64 bytes from 192.168.31.10: icmp_seq=20 ttl=128 time=1.57 ms
64 bytes from 192.168.31.10: icmp_seq=21 ttl=128 time=1.23 ms
64 bytes from 192.168.31.10: icmp_seq=22 ttl=128 time=1.14 ms
64 bytes from 192.168.31.10: icmp_seq=23 ttl=128 time=1.81 ms
64 bytes from 192.168.31.10: icmp_seq=24 ttl=128 time=1.67 ms
^C
--- 192.168.31.10 ping statistics ---
24 packets transmitted, 24 received, 0% packet loss, time 23110ms
rtt min/avg/max/mdev = 0.574/1.427/2.705/0.466 ms
practicalexamplepc10@complab304pc21: ~$
```

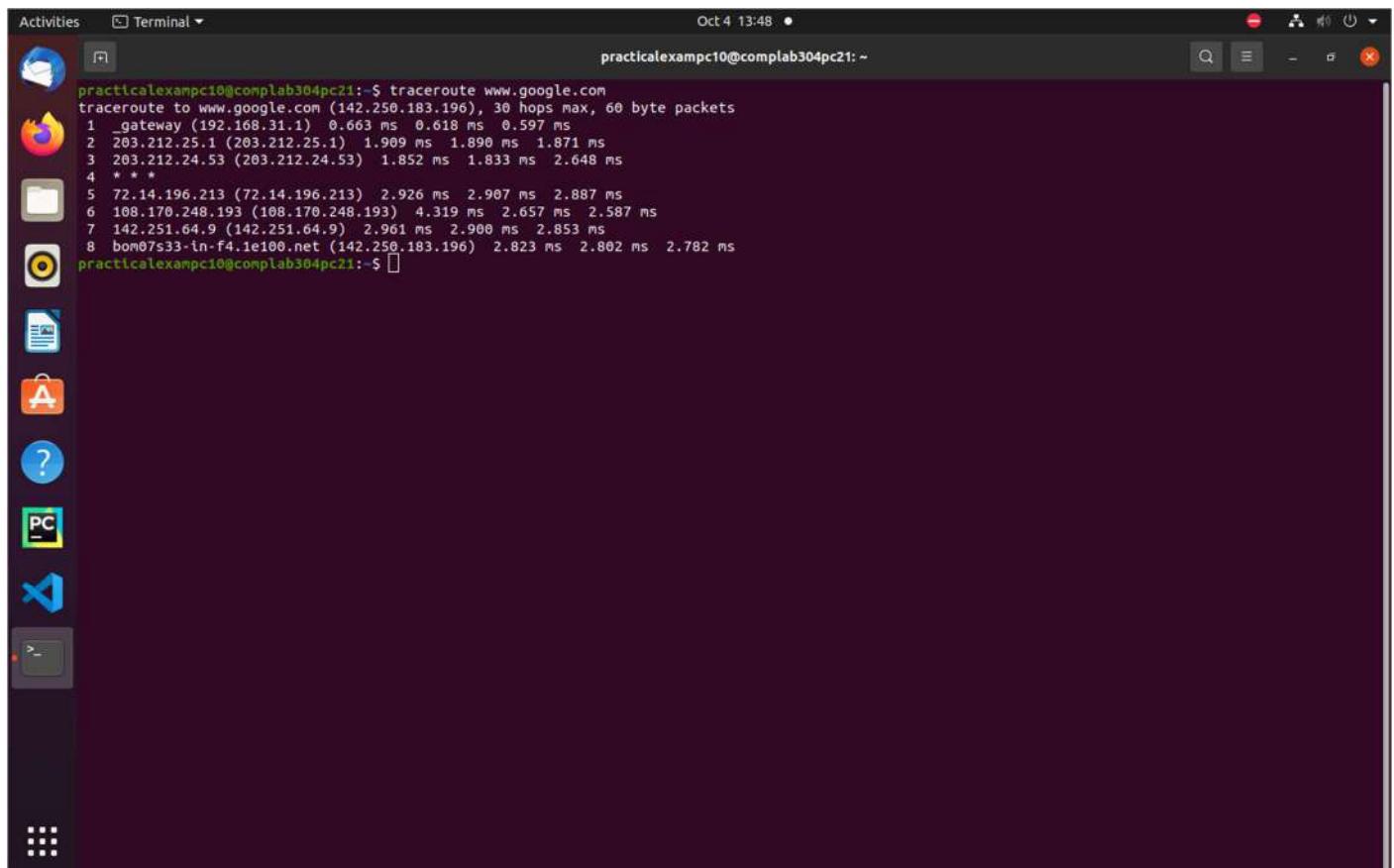
3. ping with count

```
Activities Terminal Oct 4 13:48 • practicalexamplepc10@complab304pc21:~  
practicalexamplepc10@complab304pc21:~$ ping -c 5 192.168.31.10  
PING 192.168.31.10 (192.168.31.10) 56(84) bytes of data.  
64 bytes from 192.168.31.10: icmp_seq=1 ttl=128 time=1.53 ms  
64 bytes from 192.168.31.10: icmp_seq=2 ttl=128 time=1.41 ms  
64 bytes from 192.168.31.10: icmp_seq=3 ttl=128 time=1.59 ms  
64 bytes from 192.168.31.10: icmp_seq=4 ttl=128 time=0.884 ms  
64 bytes from 192.168.31.10: icmp_seq=5 ttl=128 time=2.19 ms  
--- 192.168.31.10 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4007ms  
rtt min/avg/max/mdev = 0.884/1.519/2.189/0.416 ms  
practicalexamplepc10@complab304pc21:~$
```

4. dig

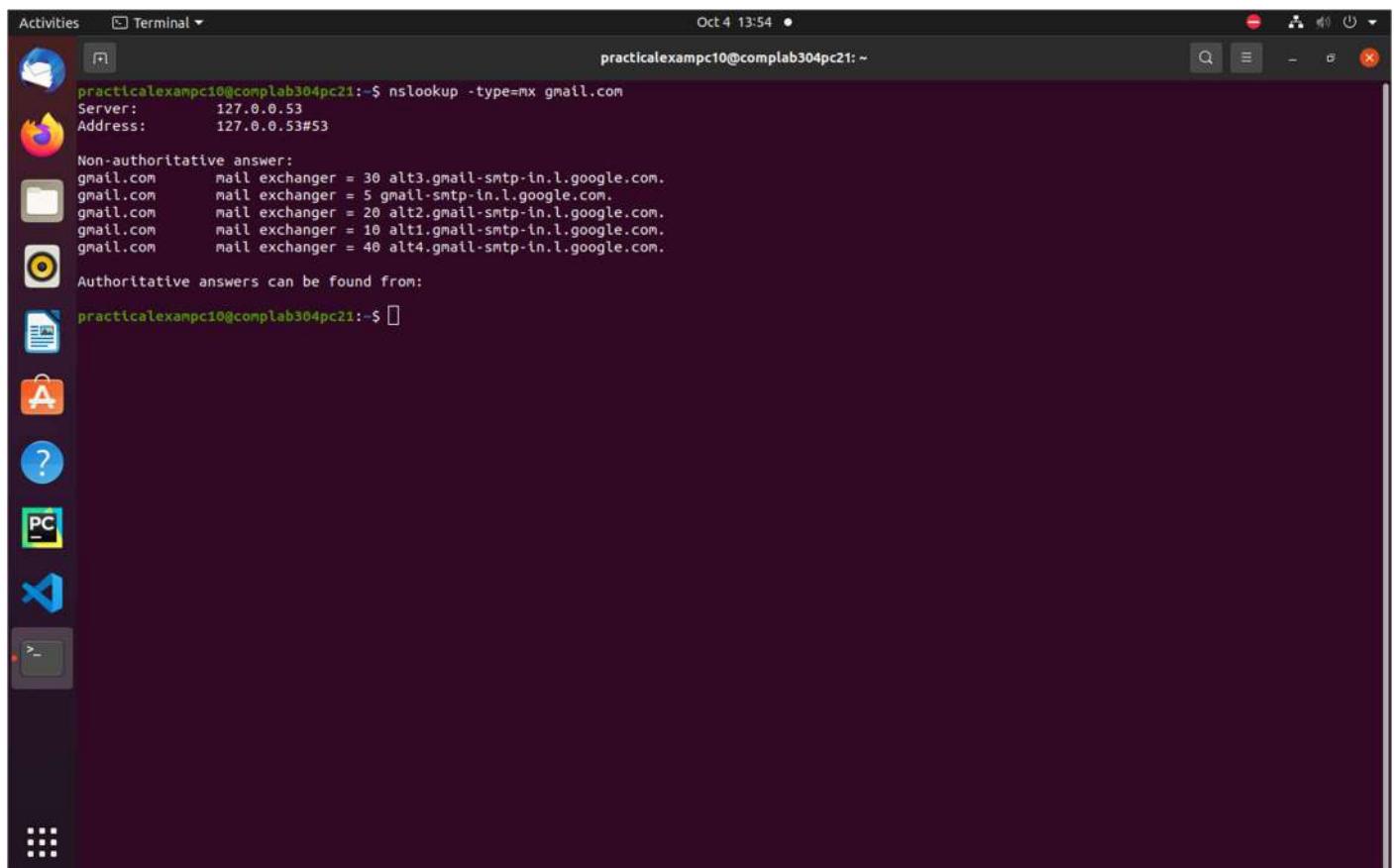
```
Activities Terminal Oct 4 13:59 • practicalexamplepc10@complab304pc21:~  
practicalexamplepc10@complab304pc21:~$ dig www.google.com  
; <>> DiG 9.16.1-Ubuntu <>> www.google.com  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 56532  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 65494  
;; QUESTION SECTION:  
;www.google.com. IN A  
  
;; ANSWER SECTION:  
www.google.com. 187 IN A 142.250.183.196  
  
;; Query time: 0 msec  
;; SERVER: 127.0.0.53#53(127.0.0.53)  
;; WHEN: Wed Oct 04 13:59:11 IST 2023  
;; MSG SIZE rcvd: 59  
practicalexamplepc10@complab304pc21:~$
```

5. traceroute



```
practicalexampc10@complab304pc21:~$ traceroute www.google.com
traceroute to www.google.com (142.250.183.196), 30 hops max, 60 byte packets
 1  _gateway (192.168.31.1)  0.663 ms  0.618 ms  0.597 ms
 2  203.212.25.1 (203.212.25.1)  1.909 ms  1.890 ms  1.871 ms
 3  203.212.24.53 (203.212.24.53)  1.852 ms  1.833 ms  2.648 ms
 4  * * *
 5  72.14.196.213 (72.14.196.213)  2.926 ms  2.907 ms  2.887 ms
 6  108.170.248.193 (108.170.248.193)  4.319 ms  2.657 ms  2.587 ms
 7  142.251.64.9 (142.251.64.9)  2.961 ms  2.908 ms  2.853 ms
 8  bon07s33-in-f4.1e100.net (142.250.183.196)  2.823 ms  2.802 ms  2.782 ms
practicalexampc10@complab304pc21:~$
```

6. nslookup

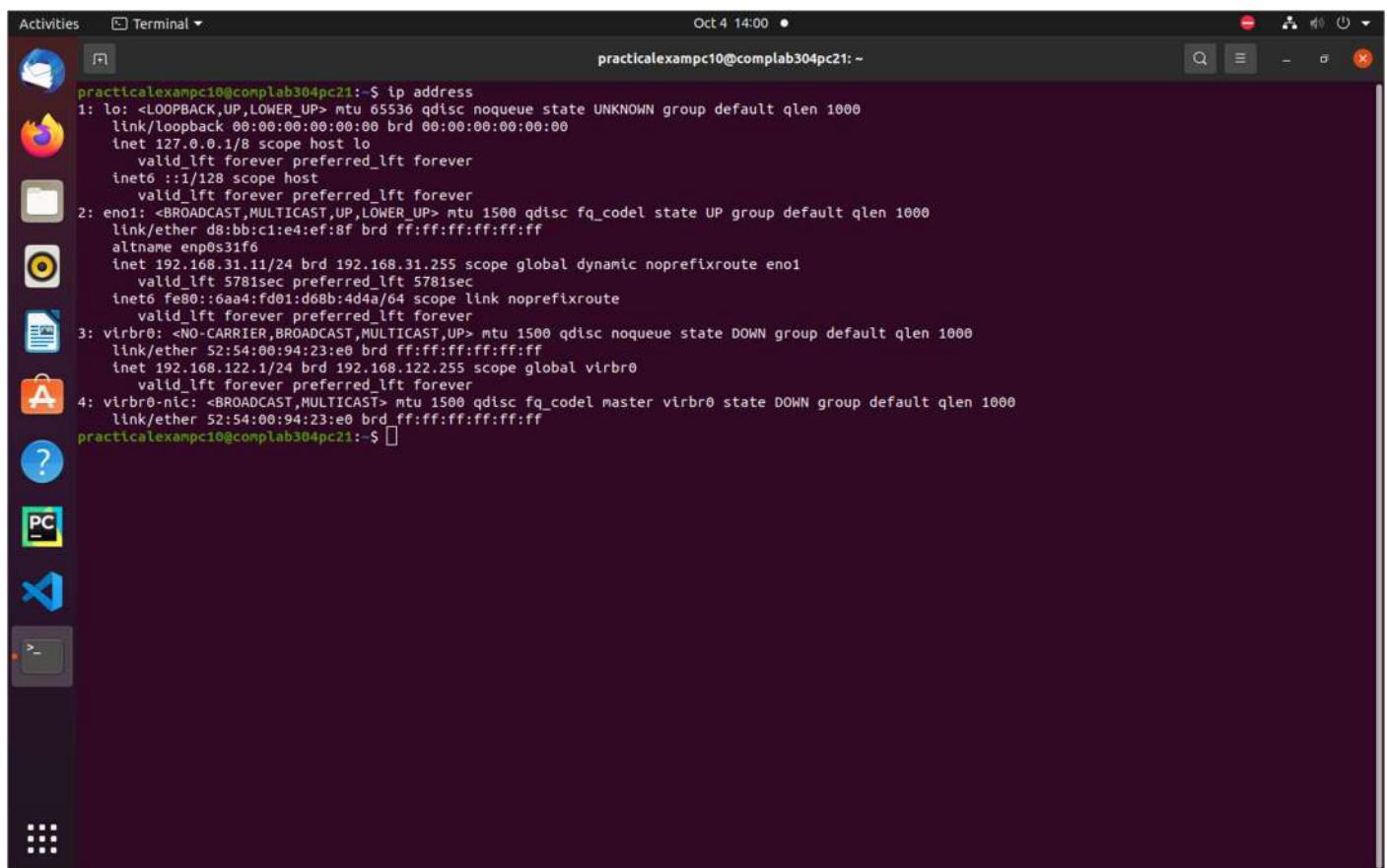


```
practicalexampc10@complab304pc21:~$ nslookup -type=mx gmail.com
Server:  127.0.0.53
Address: 127.0.0.53#53

Non-authoritative answer:
gmail.com      mail exchanger = 30 alt3.gmail-smtp-in.l.google.com.
gmail.com      mail exchanger = 5 gmail-smtp-in.l.google.com.
gmail.com      mail exchanger = 20 alt2.gmail-smtp-in.l.google.com.
gmail.com      mail exchanger = 10 alt1.gmail-smtp-in.l.google.com.
gmail.com      mail exchanger = 40 alt4.gmail-smtp-in.l.google.com.

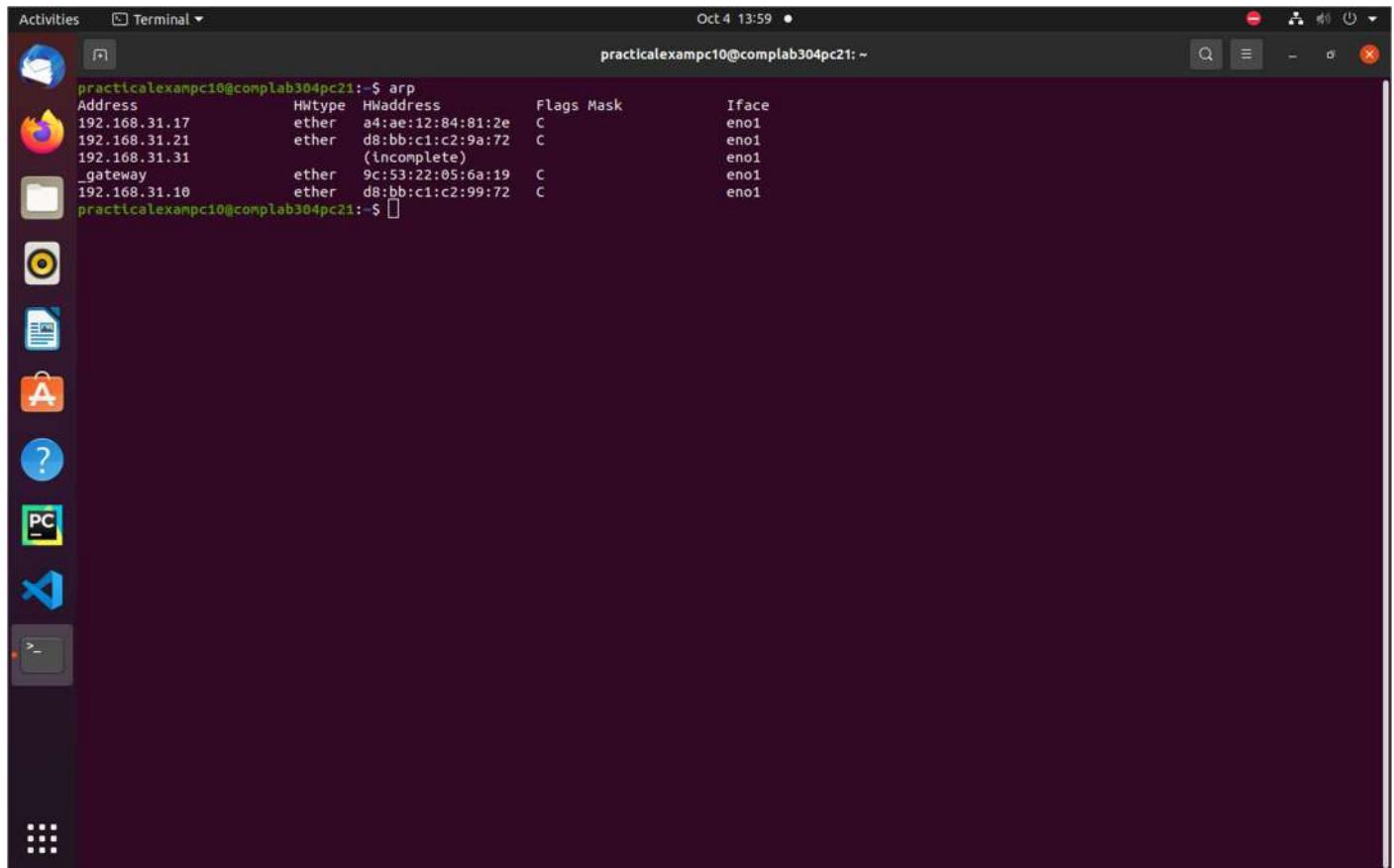
Authoritative answers can be found from:
practicalexampc10@complab304pc21:~$
```

7. ip address



```
practicalexampc10@complab304pc21:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
            inet6 ::1/128 scope host
                valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether d8:bb:c1:e4:ef:8f brd ff:ff:ff:ff:ff:ff
        altname enp0s31f6
        inet 192.168.31.11/24 brd 192.168.31.255 scope global dynamic noprefixroute eno1
            valid_lft 5781sec preferred_lft 5781sec
            inet6 fe80::6aa4:fd01:d68b:4d4a/64 scope link noprefixroute
                valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:94:23:e0 brd ff:ff:ff:ff:ff:ff
        inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
            valid_lft forever preferred_lft forever
4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:94:23:e0 brd ff:ff:ff:ff:ff:ff
practicalexampc10@complab304pc21:~$
```

8. arp



```
practicalexampc10@complab304pc21:~$ arp
Address      HWtype  HWaddress          Flags Mask   Iface
192.168.31.17  ether   a4:a1:12:84:81:2e  C      eno1
192.168.31.21  ether   d8:bb:c1:c2:9a:72  C      eno1
192.168.31.31          (incomplete)
_gateway       ether   9c:53:22:05:6a:19  C      eno1
192.168.31.10  ether   d8:bb:c1:c2:99:72  C      eno1
practicalexampc10@complab304pc21:~$
```

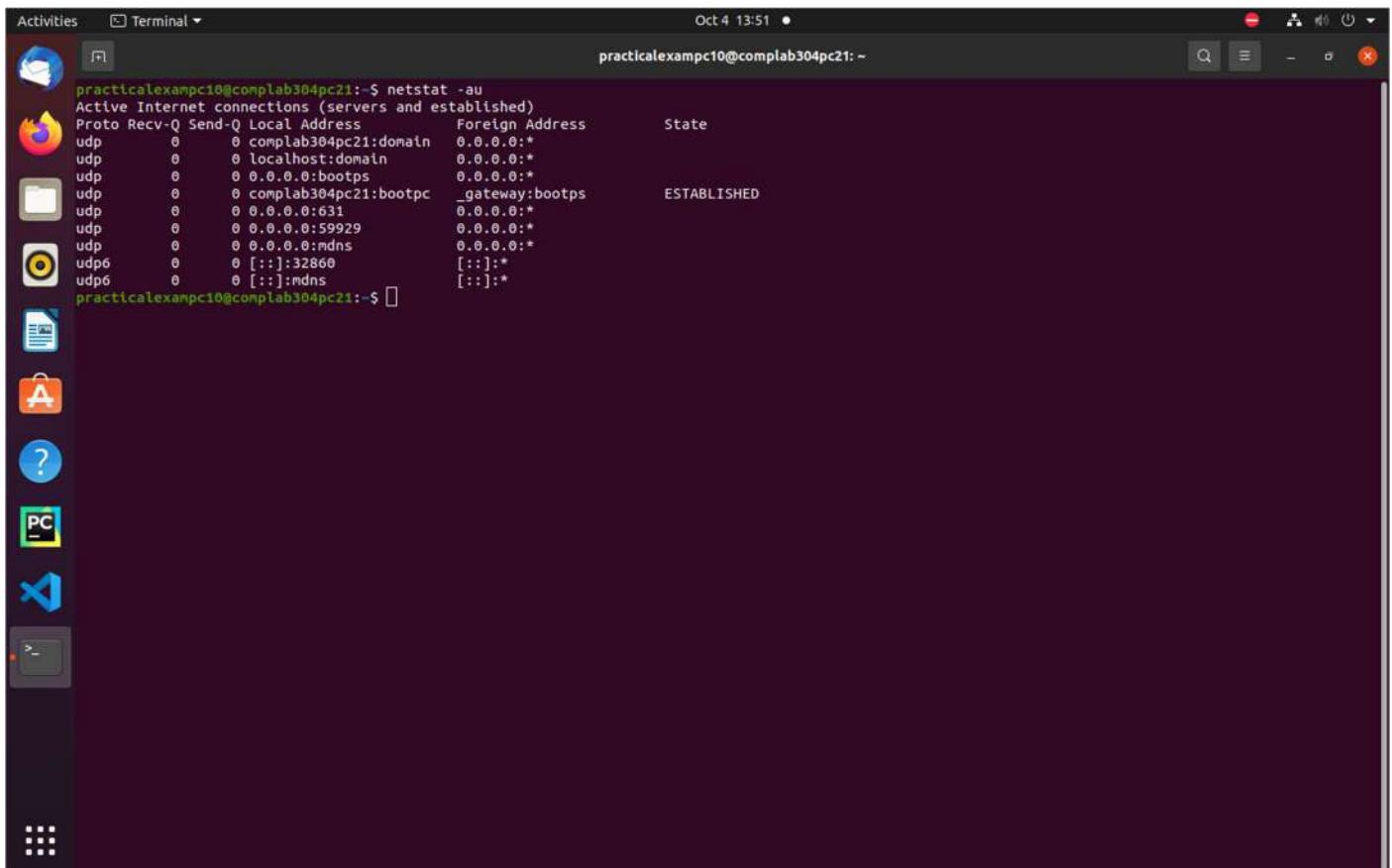
9. netstat

```
Activities Terminal Oct 4 13:49 • prakticalexampc10@complab304pc21:~  
practicallexampc10@complab304pc21:~$ netstat  
Active Internet connections (w/o servers)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State  
tcp        0      0 localhost:domain        0.0.0.0:*              LISTEN  
tcp        0      0 localhost:mysql        0.0.0.0:*              LISTEN  
tcp        0      0 localhost:ipp          0.0.0.0:*              LISTEN  
tcp        0      0 localhost:33060         0.0.0.0:*              LISTEN  
tcp        0      0 ip6-localhost:ipp       [::]:*                LISTEN  
tcp6       0      0 complab304pc21:domain  0.0.0.0:*              LISTEN  
tcp6       0      0 complab304pc21:domain  [::]:*                LISTEN  
udp        0      0 localhost:domain        0.0.0.0:*              LISTEN  
udp        0      0 0.0.0.0:bootps        0.0.0.0:*              LISTEN  
udp        0      0 complab304pc21:bootpc   _gateway:bootps        ESTABLISHED  
udp        0      0 0.0.0.0:631          0.0.0.0:*              LISTEN  
udp        0      0 0.0.0.0:59929        0.0.0.0:*              LISTEN  
udp        0      0 0.0.0.0:ndns         0.0.0.0:*              LISTEN  
udp6       0      0 [::]:32860           [::]:*                LISTEN  
udp6       0      0 [::]:ndns            [::]:*                LISTEN  
raw6       0      0 [::]:tpv6-icmp       [::]:*                7  
Active UNIX domain sockets (w/o servers)  
Proto RefCnt Flags Type      State      I-Node Path  
unix  2      [ ]  DGRAM    CONNECTED  47322  /run/user/1002/systemd/notify  
unix  4      [ ]  DGRAM    CONNECTED  13572  /run/systemd/notify  
unix  2      [ ]  DGRAM    CONNECTED  13588  /run/systemd/journal/syslog  
unix 19     [ ]  DGRAM    CONNECTED  13598  /run/systemd/journal/dev-log  
unix  9      [ ]  DGRAM    CONNECTED  13602  /run/systemd/journal/socket  
unix  3      [ ]  STREAM   CONNECTED  66593  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  54404  /run/user/1002/systemd/notify  
unix  3      [ ]  STREAM   CONNECTED  35825  /run/systemd/journal/syslog  
unix  3      [ ]  STREAM   CONNECTED  49710  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  53838  /run/dbus/system_bus_socket  
unix  3      [ ]  STREAM   CONNECTED  49703  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  51378  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  32169  /run/user/1002/systemd/notify  
unix  3      [ ]  STREAM   CONNECTED  49756  /run/systemd/journal/stdout  
unix  3      [ ]  STREAM   CONNECTED  35931  /run/systemd/journal/stdout  
unix  3      [ ]  STREAM   CONNECTED  52647  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  42866  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  42832  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  46710  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  55309  /run/dbus/system_bus_socket  
unix  3      [ ]  STREAM   CONNECTED  49286  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  34417  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  28272  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  59446  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  40529  /run/systemd/journal/stdout  
unix  3      [ ]  STREAM   CONNECTED  52702  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  58577  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  48543  /run/systemd/journal/stdout  
unix  3      [ ]  STREAM   CONNECTED  33291  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  30601  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  35704  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  46547  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  55319  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  54521  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  51627  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  51550  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  39949  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  33269  /run/dbus/system_bus_socket  
unix  3      [ ]  STREAM   CONNECTED  57187  /run/user/1002/bus  
unix  3      [ ]  STREAM   CONNECTED  37481  /run/user/1002/bus
```

10. netstat -a

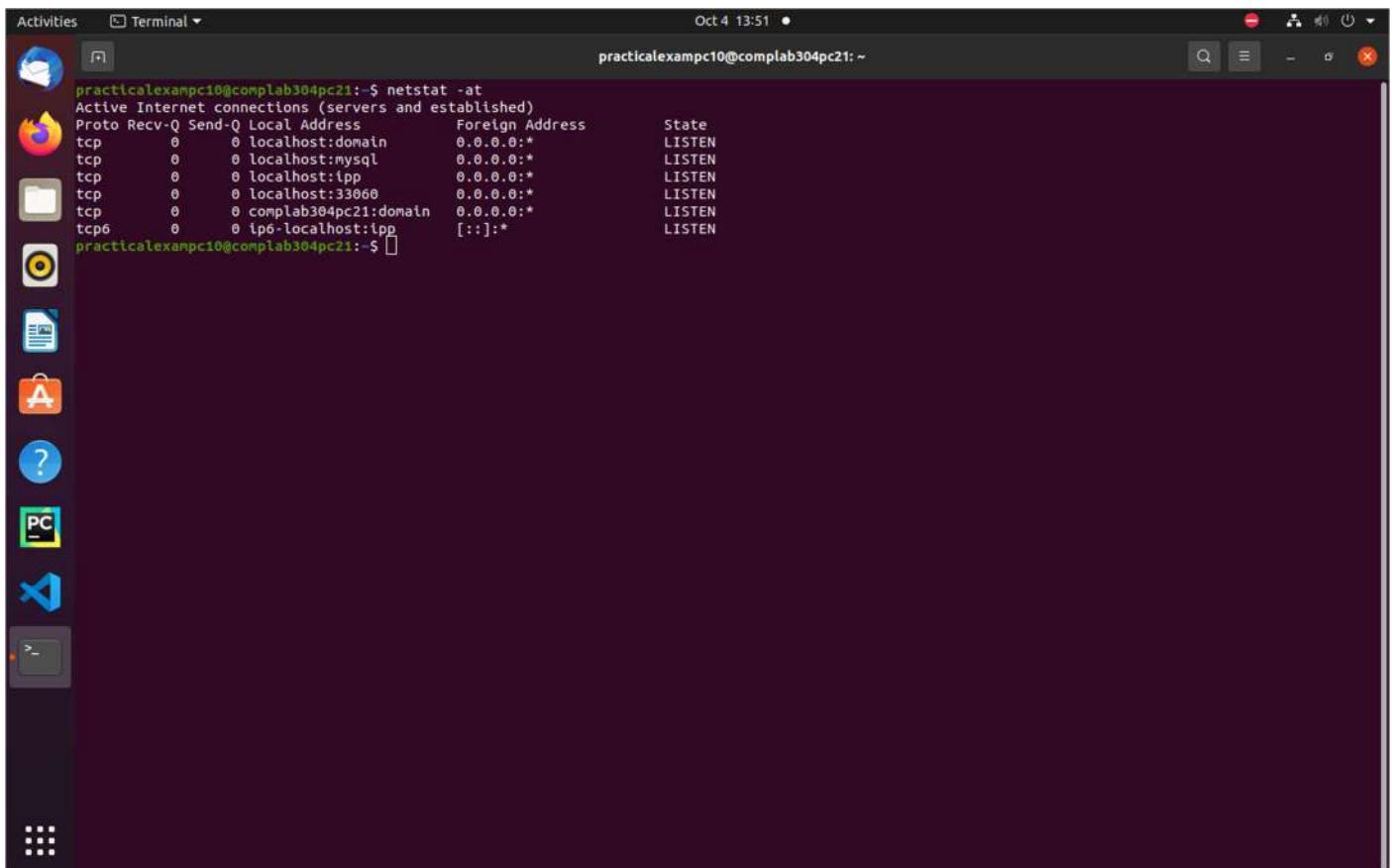
```
Activities Terminal Oct 4 13:50 • prakticalexampc10@complab304pc21:~  
practicallexampc10@complab304pc21:~$ netstat -a  
Active Internet connections (servers and established)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State  
tcp        0      0 localhost:domain        0.0.0.0:*              LISTEN  
tcp        0      0 localhost:mysql        0.0.0.0:*              LISTEN  
tcp        0      0 localhost:ipp          0.0.0.0:*              LISTEN  
tcp        0      0 localhost:33060         0.0.0.0:*              LISTEN  
tcp6       0      0 ip6-localhost:ipp       [::]:*                LISTEN  
tcp6       0      0 complab304pc21:domain  0.0.0.0:*              LISTEN  
tcp6       0      0 complab304pc21:domain  [::]:*                LISTEN  
tcp6       0      0 localhost:domain        0.0.0.0:*              LISTEN  
tcp6       0      0 0.0.0.0:bootps        0.0.0.0:*              LISTEN  
tcp6       0      0 complab304pc21:bootpc   _gateway:bootps        ESTABLISHED  
tcp6       0      0 0.0.0.0:631          0.0.0.0:*              LISTEN  
tcp6       0      0 0.0.0.0:59929        0.0.0.0:*              LISTEN  
tcp6       0      0 0.0.0.0:ndns         0.0.0.0:*              LISTEN  
tcp6       0      0 [::]:32860           [::]:*                LISTEN  
tcp6       0      0 [::]:ndns            [::]:*                LISTEN  
raw6       0      0 [::]:tpv6-icmp       [::]:*                7  
Active UNIX domain sockets (servers and established)  
Proto RefCnt Flags Type      State      I-Node Path  
unix  2      [ ACC ]  STREAM   LISTENING  43993  /tmp/ssh-posYlx80BGye/agent.1815  
unix  2      [ ]  DGRAM    LISTENING  47322  /run/user/1002/systemd/notify  
unix  2      [ ACC ]  STREAM   LISTENING  47325  /run/user/1002/systemd/private  
unix  2      [ ACC ]  STREAM   LISTENING  47330  /run/user/1002/bus  
unix  2      [ ACC ]  STREAM   LISTENING  47331  /run/user/1002/gnupg/S.dirmng  
unix  2      [ ACC ]  STREAM   LISTENING  47332  /run/user/1002/gnupg/S.gpg-agent.browser  
unix  2      [ ACC ]  STREAM   LISTENING  47333  /run/user/1002/gnupg/S.gpg-agent.extra  
unix  2      [ ACC ]  STREAM   LISTENING  28219  /run/acpid.socket  
unix  2      [ ACC ]  STREAM   LISTENING  47334  /run/user/1002/gnupg/S.gpg-agent.ssh  
unix  2      [ ACC ]  STREAM   LISTENING  47335  /run/user/1002/gnupg/S.gpg-agent  
unix  2      [ ACC ]  STREAM   LISTENING  28221  /run/avahi-daemon/socket  
unix  2      [ ACC ]  STREAM   LISTENING  47336  /run/user/1002/pk-debconf-socket  
unix  2      [ ACC ]  STREAM   LISTENING  28223  /run/cups/cups.sock  
unix  2      [ ACC ]  STREAM   LISTENING  47337  /run/user/1002/pulse/native  
unix  2      [ ACC ]  STREAM   LISTENING  47338  /run/user/1002/snapd-session-agent.socket  
unix  2      [ ACC ]  STREAM   LISTENING  28225  /run/dbus/system_bus_socket  
unix  2      [ ACC ]  STREAM   LISTENING  41616  /tmp/X11-unix/X0  
unix  2      [ ACC ]  STREAM   LISTENING  28227  /run/libvirt/libvirt-sock  
unix  2      [ ACC ]  STREAM   LISTENING  35620  /tmp/.ICE-unix/1920  
unix  2      [ ACC ]  STREAM   LISTENING  35619  @/tmp/.ICE-unix/1920  
unix  2      [ ACC ]  STREAM   LISTENING  28229  /run/snapd.socket  
unix  2      [ ACC ]  STREAM   LISTENING  28231  /run/snapd-snap.socket  
unix  2      [ ACC ]  STREAM   LISTENING  28233  /run/uuid/request  
unix  2      [ ACC ]  STREAM   LISTENING  28235  /run/libvirt/virtlockd-sock  
unix  2      [ F ACC ]  STREAM   LISTENING  41615  @/tmp/.X11-unix/X0
```

11. netstat -au



```
practicalexampc10@complab304pc21:~$ netstat -au
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 complab304pc21:domain    0.0.0.0:*
udp        0      0 localhost:domain        0.0.0.0:*
udp        0      0 0.0.0.0:bootps          0.0.0.0:*
udp        0      0 complab304pc21:bootpc   _gateway:bootps       ESTABLISHED
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:59929            0.0.0.0:*
udp        0      0 0.0.0.0:ndns            0.0.0.0:*
udp6       0      0 [::]:32860              [::]:*
udp6       0      0 [::]:ndns              [::]:*
```

12. netstat -at



```
practicalexampc10@complab304pc21:~$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:domain        0.0.0.0:*
tcp        0      0 localhost:mysql          0.0.0.0:*
tcp        0      0 localhost:ipp           0.0.0.0:*
tcp        0      0 localhost:33060          0.0.0.0:*
tcp        0      0 complab304pc21:domain  0.0.0.0:*
tcp6       0      0 ip6-localhost:ipp     [::]:*
practicalexampc10@complab304pc21:~$
```

13. netstat -l

```
Activities Terminal Oct 4 13:52 ●
practicalexampc10@complab304pc21:~$ netstat -l
Active Internet connections (only servers)
Proto Recv-Q Local Address           Foreign Address         State
tcp      0    0 localhost:domain     0.0.0.0:*
tcp      0    0 localhost:mysql      0.0.0.0:*
tcp      0    0 localhost:ipp       0.0.0.0:*
tcp      0    0 localhost:33060      0.0.0.0:*
tcp      0    0 complab304pc21:domain 0.0.0.0:*
tcp6     0    0 ip6-localhost:ipp    [::]:*
udp      0    0 complab304pc21:domain 0.0.0.0:*
udp      0    0 localhost:domain     0.0.0.0:*
udp      0    0 0.0.0.0:bootps      0.0.0.0:*
udp      0    0 0.0.0.0:631        0.0.0.0:*
udp      0    0 0.0.0.0:59929      0.0.0.0:*
udp      0    0 0.0.0.0:mdns       0.0.0.0:*
udp6     0    0 [::]:32860          [::]:*
udp6     0    0 [::]:mdns          [::]:*
raw6     0    0 [::]:ipv6-icmp     [::]:*               7

Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type      State      I-Node  Path
unix  2      [ ACC ]     STREAM   LISTENING  43993   /tmp/ssh-posYlx80BGye/agent.1815
unix  2      [ ACC ]     STREAM   LISTENING  47325   /run/user/1002/systemd/private
unix  2      [ ACC ]     STREAM   LISTENING  47330   /run/user/1002/bus
unix  2      [ ACC ]     STREAM   LISTENING  47331   /run/user/1002/gnupg/S.dirmngr
unix  2      [ ACC ]     STREAM   LISTENING  47332   /run/user/1002/gnupg/S.gpg-agent.browser
unix  2      [ ACC ]     STREAM   LISTENING  47333   /run/user/1002/gnupg/S.gpg-agent.extra
unix  2      [ ACC ]     STREAM   LISTENING  28219   /run/acpid.socket
unix  2      [ ACC ]     STREAM   LISTENING  47334   /run/user/1002/gnupg/S.gpg-agent.ssh
unix  2      [ ACC ]     STREAM   LISTENING  47335   /run/user/1002/gnupg/S.gpg-agent
unix  2      [ ACC ]     STREAM   LISTENING  28221   /run/avahi-daemon/socket
unix  2      [ ACC ]     STREAM   LISTENING  47336   /run/user/1002/pk-debconf-socket
unix  2      [ ACC ]     STREAM   LISTENING  28223   /run/cups/cups.sock
unix  2      [ ACC ]     STREAM   LISTENING  47337   /run/user/1002/pulse/native
unix  2      [ ACC ]     STREAM   LISTENING  47338   /run/user/1002/snapd-session-agent.socket
unix  2      [ ACC ]     STREAM   LISTENING  28225   /run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM   LISTENING  41616   /tmp/.X11-unix/X0
unix  2      [ ACC ]     STREAM   LISTENING  28227   /run/libvirt/libvirt-sock
unix  2      [ ACC ]     STREAM   LISTENING  35620   /tmp/.ICE-unix/1920
unix  2      [ ACC ]     STREAM   LISTENING  35619   @/tmp/.ICE-unix/1920
unix  2      [ ACC ]     STREAM   LISTENING  28229   /run/snapd.socket
unix  2      [ ACC ]     STREAM   LISTENING  28231   /run/snapd-snap.socket
unix  2      [ ACC ]     STREAM   LISTENING  28233   /run/uuid/request
unix  2      [ ACC ]     STREAM   LISTENING  28235   /run/libvirt/virtlockd-sock
unix  2      [ ACC ]     STREAM   LISTENING  41615   @/tmp/.X11-unix/X0
unix  2      [ ACC ]     STREAM   LISTENING  28237   /run/libvirt/virtlockd-admin-sock
unix  2      [ ACC ]     STREAM   LISTENING  28239   /run/libvirt/virtlockd-sock
```

14. netstat -s

```
Activities Terminal Oct 4 13:53 ●
practicalexampc10@complab304pc21:~$ netstat -s
Ip:
Forwarding: 1
10154 total packets received
0 forwarded
0 incoming packets discarded
10118 incoming packets delivered
4796 requests sent out
20 outgoing packets dropped
1 dropped because of missing route

Icmp:
272 ICMP messages received
0 input ICMP message failed
ICMP input histogram:
destination unreachable: 103
timeout in transit: 18
echo replies: 151
508 ICMP messages sent
0 ICMP messages failed
ICMP output histogram:
destination unreachable: 92
echo requests: 416

IcmpMsg:
InType0: 151
InType3: 103
InType11: 18
OutType3: 92
OutType8: 416

Tcp:
32 active connection openings
0 passive connection openings
4 failed connection attempts
0 connection resets received
0 connections established
8296 segments received
3965 segments sent out
18 segments retransmitted
0 bad segments received
35 resets sent

Udp:
993 packets received
44 packets to unknown port received
0 packet receive errors
469 packets sent
0 receive buffer errors
0 send buffer errors
```

EXPERIMENT - 9

AIM: Use wireshark to understand the operation of TCP/IP layers:

Ethernet Layer: Frame header, frame size etc.

Data Link Layer: MAC address, ARP (IP and MAC)

Network Layer: IP Packet (header, fragmentation),
ICMP (Query and Echo)

Transport Layer: TCP Ports, TCP handshake segments etc.

Application Layer: DHCP, FTP, HTTP header formats.

THEORY:

wireshark is a popular open-source network protocol analyser used for capturing and inspecting packets on a network. It is a powerful tool for network administrators, security professionals and students.

The objective of this experiment is to use the wireshark, a network protocol analyser, to capture and analyze network traffic to understand the operations and various layers and TCP/IP protocol suite.

1. Ethernet Layer: This layer deals with frames which are packets of data at the ethernet layer. One can observe the size of the ethernet frames in wireshark. Ethernet frames have headers containing information like source and destination MAC address.

2. Data Link layer: the data link layer deals with MAC

addresser. Wireshark will show you the MAC address of the device communicating in the network. User can capture ARP packets to see how device map IP addresses to MAC addresses.

3. Network Layer: This layer involves a lot of processes. It involves IP packets. Wireshark displays IP header with information like source and destination IP addresses.

TCP packets can be captured to observe network troubleshooting messages, like ping requests and replies.

4. Transport Layer: User can see which ports are used by applications (Eg: web browsers, email clients) for communication. By capturing TCP traffic, you can see the three way handshake process, which is how two devices establish a connection.

Protocol Name	color in Wireshark
Error in Packets	Black / Dark shades
TCP	Light Purple
UDP	Light Blue
HTTP	Light Green
ARP	Peach
ICMP	Pink
SMB	Light yellow
Routing	Dark yellow
TCP SYN, FIN, ACK	Dark Gray
TCP RST	Red

1. Internet Protocol (IP)

ip.addr==192.168.31.10						
Source	No.	Time	Destination	Protocol	Length	Info
192.168.31.10	266	12.719752	204.79.197.283	TCP	54	49700 + 443 [ACK] Seq=518 Ack=2881 Win=132352 Len=0
204.79.197.283	267	12.719826	192.168.31.10	TCP	1494	443 + 49700 [ACK] Seq=2881 Ack=518 Win=4194048 Len=1440 [TCP segment of a reassembled PDU]
204.79.197.283	268	12.719957	192.168.31.10	TCP	1494	443 + 49700 [ACK] Seq=4321 Ack=518 Win=4194048 Len=1440 [TCP segment of a reassembled PDU]
192.168.31.10	269	12.719984	204.79.197.283	TCP	54	49700 + 443 [ACK] Seq=518 Ack=5761 Win=132352 Len=0
192.168.31.10	271	12.765667	204.79.197.283	TCP	54	49700 + 443 [ACK] Seq=518 Ack=5984 Win=132096 Len=0
192.168.31.10	272	12.769267	203.212.24.46	DNS	71	Standard query 0xdead A ntp.msn.com
203.212.24.46	273	12.771526	192.168.31.10	DNS	146	1 standard query response 0xdead A http.msn.com CNAME a-0003.a-msedge.net CNAME a-0003.a-msedge.net A 204.79.197..
192.168.31.10	274	12.775489	204.79.197.283	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
192.168.31.10	275	12.776589	204.79.197.283	TLSv1.2	153	Application Data
192.168.31.10	276	12.777066	204.79.197.283	TLSv1.2	2098	Application Data
204.79.197.283	277	12.777365	192.168.31.10	TCP	66	443 + 49700 [ACK] Seq=5958 Ack=676 Win=4193792 Len=0
204.79.197.283	278	12.778875	192.168.31.10	TLSv1.2	396	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
204.79.197.283	279	12.778875	192.168.31.10	TLSv1.2	123	Application Data
192.168.31.10	280	12.778912	204.79.197.283	TCP	54	49700 + 443 [ACK] Seq=2820 Ack=6369 Win=1318496 Len=0
204.79.197.283	281	12.778926	192.168.31.10	TCP	66	443 + 49700 [ACK] Seq=6369 Ack=775 Win=4193792 Len=0
192.168.31.10	282	12.779039	204.79.197.283	TLSv1.2	92	Application Data
204.79.197.283	283	12.779451	192.168.31.10	TLSv1.2	92	Application Data
204.79.197.283	284	12.779451	192.168.31.10	TCP	66	443 + 49700 [ACK] Seq=6487 Ack=2828 Win=4194048 Len=0
204.79.197.283	285	12.780827	192.168.31.10	TCP	66	443 + 49700 [ACK] Seq=6487 Ack=2858 Win=4194048 Len=0
192.168.31.10	286	12.828175	204.79.197.283	TCP	54	49700 + 443 [ACK] Seq=2858 Ack=6407 Win=131584 Len=0
204.79.197.283	288	12.926652	192.168.31.10	TCP	1494	443 + 49700 [ACK] Seq=6487 Ack=2858 Win=4194048 Len=1440 [TCP segment of a reassembled PDU]
204.79.197.283	289	12.926731	192.168.31.10	TCP	1494	443 + 49700 [ACK] Seq=7847 Ack=2858 Win=4194048 Len=1440 [TCP segment of a reassembled PDU]
192.168.31.10	290	12.926744	204.79.197.283	TCP	54	49700 + 443 [ACK] Seq=2858 Ack=9287 Win=132352 Len=0
204.79.197.283	291	12.926857	192.168.31.10	TCP	1494	443 + 49700 [ACK] Seq=9287 Ack=2858 Win=4194048 Len=1440 [TCP segment of a reassembled PDU]
204.79.197.283	292	12.927009	192.168.31.10	TCP	1494	443 + 49700 [ACK] Seq=10727 Ack=2658 Win=4194048 Len=1440 [TCP segment of a reassembled PDU]
192.168.31.10	293	12.927021	204.79.197.283	TCP	54	49700 + 443 [ACK] Seq=2858 Ack=12167 Win=132352 Len=0
204.79.197.283	294	12.927101	192.168.31.10	TCP	1494	443 + 49700 [ACK] Seq=2858 Ack=12167 Win=4194048 Len=1440 [TCP segment of a reassembled PDU]

> Frame 221: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\WPF_{704644D1-9B3F-44CB-8D76-142FFB438987}, id 0
> Ethernet II, Src: Micro-St_c2:99:72 (d8:b0:c1:c2:99:72), Dst: IPv4mcast_16 (01:00:5e:00:00:16)
> Internet Protocol Version 4, Src: 192.168.31.10, Dst: 224.0.0.22
> Internet Group Management Protocol

2. Internet Group Management Protocol (IGMP)

igmp						
Source	No.	Time	Destination	Protocol	Length	Info
169.254.190.1	282270	35.653162	224.0.0.22	IGMPv3	68	Membership Report / Join group 224.0.0.251 for any sources
169.254.190.1	282271	35.653162	224.0.0.22	IGMPv3	68	Membership Report / Join group 224.0.0.252 for any sources
169.254.190.1	282290	35.655593	224.0.0.22	IGMPv3	68	Membership Report / Join group 239.255.255.256 for any sources
169.254.190.1	285588	36.156458	224.0.0.22	IGMPv3	78	Membership Report / Join group 224.0.0.252 for any sources / Join group 239.255.255.256
192.168.31.11	234053	40.552535	224.0.0.22	IGMPv3	68	Membership Report / Leave group 224.0.0.252
192.168.31.11	234184	40.561403	224.0.0.22	IGMPv3	68	Membership Report / Leave group 239.255.255.256

> Frame 285508: 78 bytes on wire (560 bits), 78 bytes captured (560 bits) on interface \Device\WPF_{704644D1-9B3F-44CB-8D76-142FFB438987}, id 0
> Ethernet II, Src: Micro-St_e4:ef:8f (d8:b0:c1:e4:ef:8f), Dst: IPv4mcast_16 (01:00:5e:00:00:16)
> Internet Protocol Version 4, Src: 169.254.190.1, Dst: 224.0.0.22
> Internet Group Management Protocol
[IGMP Version: 3]
Type: Membership Report (0x22)
Reserved: 00
Checksum: 0x2009 [correct]
[Checksum Status: Good]
Reserved: 0000
Num Group Records: 3
> Group Record : 224.0.0.252 Change To Exclude Mode
> Group Record : 224.0.0.251 Change To Exclude Mode
> Group Record : 239.255.255.256 Change To Exclude Mode

3. Internet Control Message Protocol (ICMP)

icmp						
Source	No.	Time	Destination	Protocol	Length	Info
192.168.31.10	1136	6.937252	203.212.24.46	ICMP	347	Destination unreachable (Port unreachable)
192.168.31.10	2406	11.541414	203.212.24.46	ICMP	169	Destination unreachable (Port unreachable)
192.168.31.10	4084	13.093202	203.212.24.46	ICMP	176	Destination unreachable (Port unreachable)
192.168.31.10	4385	13.926584	203.212.24.46	ICMP	239	Destination unreachable (Port unreachable)
192.168.31.10	165592	31.196969	203.212.24.46	ICMP	347	Destination unreachable (Port unreachable)
192.168.31.10	208982	66.235611	203.212.24.46	ICMP	180	Destination unreachable (Port unreachable)

> Frame 165592: 347 bytes on wire (2776 bits), 347 bytes captured (2776 bits) on interface \Device\WPF_{704644D1-9B3F-44CB-8D76-142FFB438987}, id 0
> Ethernet II, Src: Micro-St_c2:99:72 (d8:b0:c1:c2:99:72), Dst: 9c:53:22:05:6a:19 (9c:53:22:05:6a:19)
> Internet Protocol Version 4, Src: 192.168.31.10, Dst: 203.212.24.46
> Internet Control Message Protocol
Type: 3 (Destination unreachable)
Code: 3 (Port unreachable)
Checksum: 0x1c10 [correct]
[Checksum Status: Good]
Unused: 00000000
> Internet Protocol Version 4, Src: 203.212.24.46, Dst: 192.168.31.10
> User Datagram Protocol, Src Port: 53, Dst Port: 53854
> Domain Name System (response)

4. Address Resolution Protocol (ARP)

The screenshot shows the Wireshark interface with the following details:

- Selected Frame:** Frame 276786 (ARP request from Dell_a5:05:b6 to Broadcast).
- Frame Details:**
 - Source: Dell_a5:05:b6
 - No.: 276786
 - Time: 47.419387
 - Destination: Broadcast
 - Protocol: ARP
 - Length: 60
 - Info: 60 Who has 192.168.31.37? Tell 192.168.31.1
- Selected Hex View:**

```

0000  ff ff ff ff ff ff 54 bf 64 a5 05 b6 08 06 00 81  ....T d....
0010  08 00 06 04 00 01 54 bf 64 a5 05 b6 c0 a8 1f 25  ....T d....%
0020  00 00 00 00 00 00 c0 a8 1f 01 00 00 00 00 00 00 00  .....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
    
```
- Selected ASCII View:**

```

0000  ff ff ff ff ff ff 54 bf 64 a5 05 b6 08 06 00 81  ....T d....
0010  08 00 06 04 00 01 54 bf 64 a5 05 b6 c0 a8 1f 25  ....T d....%
0020  00 00 00 00 00 00 c0 a8 1f 01 00 00 00 00 00 00 00  .....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
    
```
- Selected Bytes View:**

```

0000  ff ff ff ff ff ff 54 bf 64 a5 05 b6 08 06 00 81  ....T d....
0010  08 00 06 04 00 01 54 bf 64 a5 05 b6 c0 a8 1f 25  ....T d....%
0020  00 00 00 00 00 00 c0 a8 1f 01 00 00 00 00 00 00 00  .....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
    
```
- Selected Preview View:**

```

0000  ff ff ff ff ff ff 54 bf 64 a5 05 b6 08 06 00 81  ....T d....
0010  08 00 06 04 00 01 54 bf 64 a5 05 b6 c0 a8 1f 25  ....T d....%
0020  00 00 00 00 00 00 c0 a8 1f 01 00 00 00 00 00 00 00  .....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
    
```
- Selected Information View:**

```

Frame 276786: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{704644D1-983F-44C6-8076-142FFB438987}, id 0
Ethernet II, Src: Dell_a5:05:b6 (54:bf:64:a5:05:b6), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Dell_a5:05:b6 (54:bf:64:a5:05:b6)
    Sender IP address: 192.168.31.37
    Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.31.1
  
```

5. Dynamic Host Configuration Protocol (DHCP)

6. Domain Name System (DNS)

Source	No.	Time	Destination	Protocol	Length	Info
203.212.24.46	4788	14.792727	192.168.31.18	DNS	273	Standard query response 0x7e0a A speedtest.bigventuresmedia.com.prod.hosts.ooklaserver.net CNAME speedtest.bigventuresmedia.com
192.168.31.18	4961	17.714059	203.212.24.46	DNS	82	Standard query 0x93fa A ipv6-api.speedtest.net
192.168.31.18	4962	17.714155	203.212.24.46	DNS	82	Standard query 0xd6bc HTTPS ipv6-api.speedtest.net
203.212.24.46	4963	17.718900	192.168.31.18	DNS	169	Standard query response 0xd6bc HTTPS ipv6-api.speedtest.net SOA ns-1643.awdns-13.co.uk
203.212.24.46	4965	17.730163	192.168.31.18	DNS	169	Standard query response 0x93fa A ipv6-api.speedtest.net SOA ns-1643.awdns-13.co.uk
192.168.31.18	4968	17.730744	203.212.24.46	DNS	82	Standard query 0xc7f4 A ipv6-api.speedtest.net
203.212.24.46	4967	17.732393	192.168.31.18	DNS	169	Standard query response 0xc7f4 A ipv6-api.speedtest.net SOA ns-1643.awdns-13.co.uk
192.168.31.18	32845	20.498282	203.212.24.46	DNS	83	Standard query 0x2fb0 A ctldl.windowsupdate.com
203.212.24.46	33034	20.513474	192.168.31.18	DNS	242	Standard query response 0x2fb0 A ctldl.windowsupdate.com CNAME wu-bg-shim.trafficmanager.net CNAME download.windowsupdate.com
192.168.31.18	61693	22.811315	203.212.24.46	DNS	79	Standard query 0x714e A download.lenovo.com

```

> Frame 4967: 169 bytes on wire (1352 bits), 169 bytes captured (1352 bits) on interface \Device\NPF_{704644D1-983F-44C8-8D76-142FFB438987}, id 0
> Ethernet II, Src: 9c:53:22:05:6a:19 (9c:53:22:05:6a:19), Dst: Micro-St_c2:99:72 (d8:bb:c1:c2:99:72)
> Internet Protocol Version 4, Src: 203.212.24.46, Dst: 192.168.31.18
> User Datagram Protocol, Src Port: 53, Dst Port: 53854
> Domain Name System (response)
    Transaction ID: 0xc7f4
    Flags: 0x8100 Standard query response, No error
    Questions: 1
    Answer RRs: 0
    Authority RRs: 1
    Additional RRs: 0
    Queries:
    > Authoritative nameservers
        [Request_In: 4968]
        [Time: 0.001649000 seconds]

```

7. Hypertext Transfer Protocol (HTTP)

Source	No.	Time	Destination	Protocol	Length	Info
192.168.31.18	298059	88.556072	192.168.31.1	HTTP/X-	363	POST /ifc HTTP/1.1
192.168.31.1	298104	88.556849	192.168.31.18	HTTP/X-	408	HTTP/1.1 200 OK
192.168.31.18	298155	88.562941	192.168.31.1	HTTP/X-	367	POST /ifc HTTP/1.1
192.168.31.1	298217	98.567233	192.168.31.18	HTTP/X-	422	HTTP/1.1 200 OK
14.104.35.123	298410	98.582486	192.168.31.18	HTTP	1084	HTTP/1.1 206 Partial Content
192.168.31.18	298415	98.584567	192.168.31.1	HTTP/X-	363	POST /ifc HTTP/1.1

```

> Frame 298410: 1084 bytes on wire (8672 bits), 1084 bytes captured (8672 bits) on interface \Device\NPF_{704644D1-983F-44C8-8D76-142FFB438987}, id 0
> ethernet II, Src: 9c:53:22:05:6a:19 (9c:53:22:05:6a:19), Dst: Micro-St_c2:99:72 (d8:bb:c1:c2:99:72)
> Internet Protocol Version 4, Src: 34.104.35.123, Dst: 192.168.31.18
> Transmission Control Protocol, Src Port: 80, Dst Port: 51270, Seq: 5419490, Ack: 6104, Len: 1030
[247] Reassembled TCP Segments (355270 bytes): #298017(1440), #298018(1440), #298019(1440), #298020(1440), #298021(1440), #298022(1440), #298023(1440), #298025(1440), #298027(1440), #298028(1440), #2980
> Hypertext Transfer Protocol
    > HTTP/1.1 206 Partial Content\r\n
        accept-ranges: bytes\r\n
        content-disposition: attachment\r\n
        content-security-policy: default-src 'none'\r\n
        server: Google-Edge-Cache\r\n
        x-content-type-options: nosniff\r\n
        x-frame-options: SAMEORIGIN\r\n
        x-xss-protection: 1\r\n
        date: Tue, 03 Oct 2023 16:44:14 GMT\r\n
        age: 53590\r\n
        last-modified: Fri, 29 Sep 2023 16:33:39 GMT\r\n
        etag: "1bd5d84"\r\n
        content-type: application/octet-stream\r\n
        > content-length: 354659\r\n
        x-request-id: d3a06f18-5db6-4314-a0cc-f0f606eb92d\r\n
        content-range: bytes 505455-5409163/5409164\r\n
        alt-svc: h3="443"; ma=2592000, h3-29="443"; ma=2592000\r\n
        cache-control: public, max-age=86400\r\n
        \r\n
        [HTTP response 15/15]
        [Time since request: 0.037555000 seconds]
        [Prev request in frame: 294731]
        [Prev response in frame: 297931]
        [Request In frame: 298018]
        [Request URL: http://edged1.me.gvt1.com/edged1/release2/chrome_component/adhicj45hzjkfunn7ccrbqyyhu5q_20230916.567854687.14/cbedbbhbpmojnkanicoggnmeloomoc_20230916.567854667.14_all_ENU580000_lr74]
        File Data: 354659 bytes
        <a href="javascript: void(0)">
```

8. Transmission Control Protocol (TCP)

Screenshot of Wireshark showing TCP traffic on interface "Ethernet".

Summary:

- Frame 33813: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{A8787A28-2B44-4DA3-B2C3-A3348E8A2A15}, id 0
- Ethernet II, Src: Micro-St_C2:99:e5 (d8:b1:c1:21:99:e5), Dst: 0c:53:22:05:6a:19 (9c:53:22:05:6a:19)
- Internet Protocol Version 4, Src: 192.168.31.13, Dst: 0.241.131.254
- Transmission Control Protocol, Src Port: 57571, Dst Port: 80, Seq: 2671, Ack: 7039441, Len: 0

Details:

- Source Port: 57571
- Destination Port: 80
- [Stream index: 0]
- [Conversation completeness: Incomplete (12)]
- [TCP Segment Len: 0]
- Sequence Number: 2671 (relative sequence number)
- Sequence Number (raw): 1179984078
- [Next Sequence Number: 2671 (relative sequence number)]
- Acknowledgment Number: 7039441 (relative ack number)
- Acknowledgment number (raw): 3066583598
- 1000 - Header Length: 32 bytes (8)
- Flags: 0x10 (ACK)
- Window: 481
- [Calculated window size: 401]
- [Window size scaling factor: -1 (unknown)]
- Csum: 0xbccb [unverified]
- [Checksum Status: Unverified]
- Urgent Pointer: 0
- Options: (12 bytes), No-operation (NOP), No-operation (NOP), Timestamps
- [Timestamps]
- [SEQ/ACK analysis]



9. User Datagram Protocol (UDP)

Screenshot of Wireshark showing UDP traffic on interface "Ethernet".

Summary:

- Frame 405: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface \Device\NPF_{A8787A28-2B44-4DA3-B2C3-A3348E8A2A15}, id 0
- Ethernet II, Src: Micro-St_C2:99:e5 (d8:b1:c1:21:99:e5), Dst: 0c:53:22:05:6a:19 (9c:53:22:05:6a:19)
- Internet Protocol Version 4, Src: 192.168.31.13, Dst: 203.212.24.46
- User Datagram Protocol, Src Port: 58779, Dst Port: 53

Details:

- Source Port: 58779
- Destination Port: 53
- Length: 52
- Csum: 0xc3fd [unverified]
- [Checksum Status: Unverified]
- [Stream index: 3]
- [Timestamps]
- UDP payload (44 bytes)
- > Domain Name System (query)



EXPERIMENT: 10

AIM: Implement Socket Programming using Java.

THEORY: Java socket programming is used for communication between the applications running on different JRE.

Java socket programming can be connection-oriented or connection-less.

Socket and ServerSocket class are used for connection-oriented socket programming and Datagram Packet classes are used for connection-less socket programming.

The client in socket programming must know two information:

- IP Address of server
- Port Number

Here, we are going to implement two-way client and server information communication. In this application, client sends a message to server, server reads the message. In the same way, server sends a message to the client and the client can read the message. Here, two classes are being used : Socket and ServerSocket.

The socket class is used to communicate client and server. Through this class, we can read and write messages.

The ServerSocket class is used at server side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at the server-side.

Socket Class :

A socket is simply an endpoint for communication between the machines. The socket class can be used to create socket.

Server Socket class:

The server socket class can be used to create a server socket. This object is used to establish communication with the clients.

Creating Server: To create the server application, we need to create the instance of server socket class. Here, we are using 6666 port number for the communication between the client and the server. You may also choose any port number. The accept() method waits for the client. If the client connects with the given port number, it returns an instance of socket.

ServerSocket ss = new ServerSocket(6666);
Socket s = ss.accept(); (Establishes connection)

Creating Client: To create the client application, we need to create the instance of a socket class. Here, we need to pass the IP address or hostname of the server and a port number. Here, we are using "localhost" because our server is running on the same computer system.

Socket s = new Socket("localhost", 6666);

To use these classes we need to import :

import java.io.*;
import java.net.*;

SERVER PROGRAM:

```
import java.io.*;
import java.net.*;

public class MyServer1
{
    public static void main(String args[])throws IOException
    {
        System.out.println("Server started at port 3333");
        ServerSocket ss = new ServerSocket(3333);
        Socket s = ss.accept(); //This will establish connection between client and server
        System.out.println("Connected");
        BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the data to be sent to the client");
        String nickname = br1.readLine();
        BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream()));
        String msg = br.readLine();
        System.out.println("Client Data: "+msg);
        OutputStreamWriter os = new OutputStreamWriter(s.getOutputStream());
        PrintWriter out = new PrintWriter(os);
        out.println(nickname);
        out.flush();
        s.close();
        ss.close();
    }
}
```

CLIENT PROGRAM:

```
import java.net.*;
import java.io.*;
public class MyClient1
{
    public static void main(String args[])throws IOException
    {
        Socket s = new Socket("localhost",3333);
        System.out.println("Enter the msg to be sent: ");
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str = br.readLine();
        OutputStreamWriter os = new OutputStreamWriter(s.getOutputStream());
        PrintWriter out = new PrintWriter(os);
        out.println(str);
        os.flush();
        BufferedReader br1 = new BufferedReader(new InputStreamReader(s.getInputStream()));
        String nickname = br1.readLine();
        System.out.println("Data from server : "+nickname);
        s.close();
    }
}
```

OUTPUT:

The image shows two Microsoft Command Prompt windows running on Windows 10. The left window represents the server (MyServer1.java) and the right window represents the client (MyClient1.java).

Server (Left Window):

```
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\parth>cd javaprogram
C:\Users\parth\JavaProgram>javac MyServer1.java

C:\Users\parth\JavaProgram>java MyServer1
Server started at port 3333
Connected
Enter the data to be sent to the client
Hello Client I am Parth the Server
Client Data: Hello Parth Server, I am Client
C:\Users\parth\JavaProgram>
```

Client (Right Window):

```
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\parth>cd javaprogram
C:\Users\parth\JavaProgram>javac MyClient1.java

C:\Users\parth\JavaProgram>java MyClient1
Enter the msg to be sent:
Hello Parth Server, I am Client
Data from server : Hello Client I am Parth the Server
C:\Users\parth\JavaProgram>
```

Assignment no: 1

1. Write a short note on following devices:
- | | | |
|-------------|------------|----------|
| a. Repeater | d. Switch | g. Modem |
| b. Hub | e. Router | |
| c. Bridge | f. Gateway | |

o. REPEATER: A repeater is a powerful network device that regenerates an incoming signal from the sender before retransmitting it to the receiver. It is also known as a signal booster, and it helps in extending the coverage area of network. Since it is a hardware device, it only works at the physical layer of the OSI (Open System Interconnection) model. Therefore, it is also known as a layer-1 device. It is a two port device.

Functions:

- i) Repeaters do not amplify the signal. They strengthen the signal by regenerating it.
- ii) When the incoming signals are attenuated, it copies them bit by bit and retransmits them at their original strength.
- iii) It is used for longer distance data transmission without compromising data security and quality.

Types:

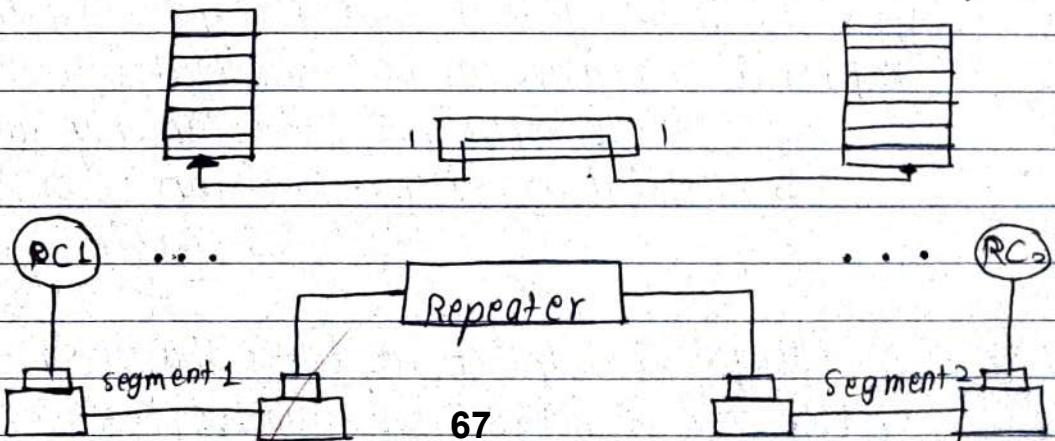
1. According to the type of signals that they regenerate, repeaters can be classified into two categories:
 - a. Analog Repeaters: They can only amplify signal.
 - b. Digital Repeaters: They can reconstruct a distorted signal.

2. According to the types of networks that they connect, repeaters can be categorised into two types:

- a. wired Repeaters :- They are used in wired LANs.
 - b. wireless Repeaters :- They are used in wireless LANs and cellular networks.
3. According to the domain of LANs they connect, repeaters can divided into two categories :-
- a. Local Repeaters :- They connect LAN segments separated by small distance.
 - b. Remote Repeaters :- They connect LANs that are far from each other.

Advantages :- i) Repeaters are simple to install and can easily extend the coverage area of networks.
 ii) They are cost effective.
 iii) They can connect signals using different types of cables.

Disadvantages :- i) Repeaters cannot connect dissimilar networks.
 ii) They cannot differentiate between actual signal and noise.
 iii) They cannot reduce network traffic or congestion.
 iv) Most networks have limitations upon the number of repeaters that can be deployed.



B. Hub : A hub is a physical layer networking devices which is used to connect multiple devices in a network. They are generally used to connect computers in a LAN. Since it works as physical layer, it is also known as Layer 1 device. It is a multi port repeater. Therefore, it can connect multiple systems to a network.

Functions :- i) A signal received at any port on the hub is retransmitted on all other ports.

ii) Network segments that employ hubs are often described as having a star topology, in which the hub forms the wiring centre of the star.

Types of Hub :- i) Active Hub :- They have a power supply for regenerating, and amplify the signals.

ii) Passive Hub :- Used to connect signals of different network cables as they do not have any computerized elements.

iii) Intelligent Hub :- These are smarter hubs which consists of Management Information Base (MIB) software that helps in analyzing and troubleshooting networks.

Advantages :- i) Less Expensive

ii) Does not impact network performance

iii) Support different network media.

iv) Easily connects with different media.

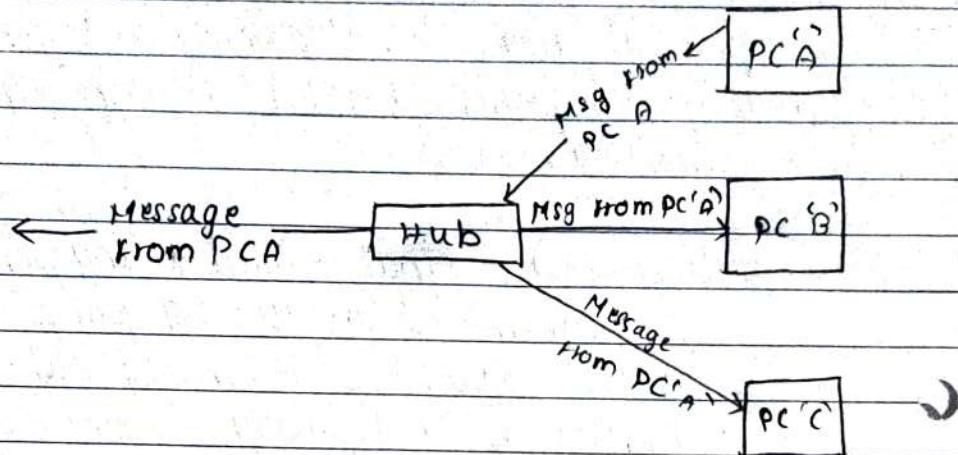
Disadvantages :- i) It cannot find the best/shortest path of the network.

ii) No mechanism for traffic detection.

iii) No mechanism for data filtration.

iv) Not capable of connecting to different

network topologies like token, ring, ethernet, etc.



C. Bridge: A bridge in a computer network is a device used to connect multiple LANs together with a Local Area Network (LAN). The mechanism of network aggregation is known as bridging. The bridge is a physical or hardware device but operates at the OSI model's data link layer and is also known as layer two device.

Functions:

- i) Bridge can be used to connect LANs of different layers.
- ii) Bridges are also used to connect 2 LANs but they have to be identical.
- iii) Bridges only have two parts. Thus, they connect only two LAN networks.
- iv) Bridges perform the forwarding functions. If a packet is received on a bridge, it forwards it.

Types of Bridges:

- i) Static Bridge: static bridges are that type of bridges which maintains a table that consists of two columns - MAC Address and Port Number

This table is manually handled by Network Admin.

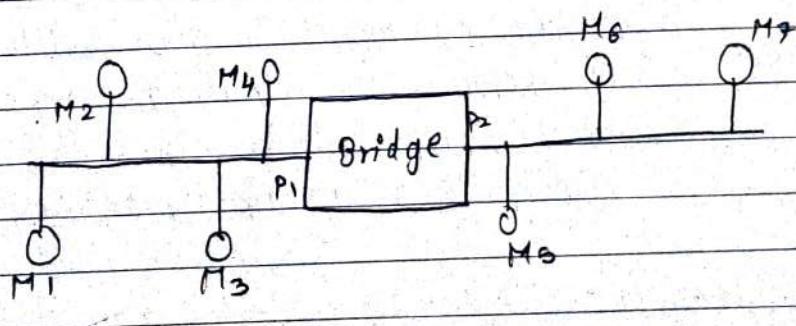
i) Dynamic-Bridge: Dynamic bridges are also known as self learning bridges or transparent bridges. In this type of the bridge the table is initially empty and is made using its intelligence.

Advantages :-

- i) Bridges can be used as a network extension like they can connect two network topologies network.
- ii) It has a separate collision domain, which results in increased bandwidth.
- iii) It can create a buffer when different MAC protocols are there for different segments.
- iv) Highly reliable and maintainable. The network can be divided into multiple LAN segments.

Disadvantages :-

- i) Expensive as compared to hubs and repeaters.
- ii) Slow in speed.
- iii) Poor performance as additional processing is required to view the MAC address of the device on the network.
- iv) As the traffic received is in bulk or is broadcasted traffic, individual filtering of data is not possible.



D. switch: A network device that connects multiple LAN networks irrespective of their types. It is a multiport bridge. It works at data link layer in the OSI model. Thus, it is also known as Layer 2 device. Different devices can be connected to its port including a router. The switch is a network device that is used to segment the networks into different subnetworks called subnets or LAN segments.

Functions : i) It performs error checking before forwarding data.
 ii) It transfers the data only to the device that has been addressed.
 iii) Switches have a more significant number of ports.
 iv) It operates in full duplex mode.

Types of switches : i) Virtual switches : Virtual switches are the switches that are inside virtual machine hosting environment.

ii) Routing switches : These are the switches that are used to connect LANs. They also have the work of performing functions in network layer of the OSI model.

Advantages : i) Prevents traffic loading over in a network by segmenting the network into smaller subnets.
 ii) Increases the bandwidth of the networks.
 iii) Reduces frame collision as the switches creates the collision domain for each connection.

Disadvantages : i) It cannot stop traffic designed for a different LAN segment from travelling to

all other LAN segments.

ii) switches are more expensive.

E. ROUTER: A router is a networking device that forward data packets between computer networks. One or more packet-switched networks or subnetworks can be connected using a router. By sending data packets to their internal IP addresses, it manages traffic between different networks and permits several devices to share an Internet connection.

Functions : i) Forwarding : The router receives the packets from its input ports, checks its header, performs some basic functions like checksum, and then looks up to the routing table to find the appropriate output port to dump the packets onto, and forwards the packets onto that output port. It transfers the data only to that device that has been addressed.

ii) Routing : Routing is the process by which the router ascertains what is the best path for the packets to reach the destination, it maintains a routing table that is made using different algorithms by the router only.

iii) Filtering : Decides whether to forward/block the packet.

Types of Routers : i) Broadband Routers : These are one of the important kinds of routers. It is used to do different ways types of things. It is used to connect computers or it is also used to connect to the internet.

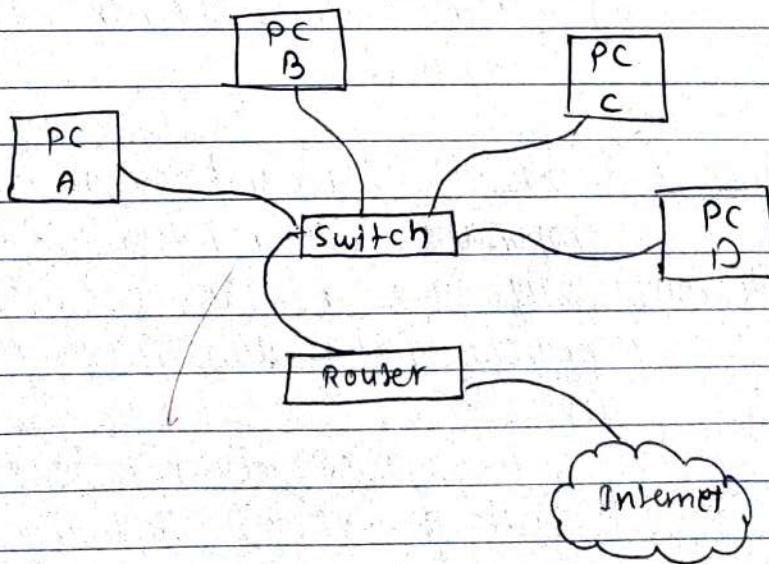
ii) - Wireless Routers : These routers are used to create a wireless signal in your office or home.

Core Routers: Core routers distribute packets within the same networks. The main task is to carry heavy data transfers.

- Advantages :
- i) Easier connection & sharing a single network connection among numerous machines is the router's main job.
 - ii) Security: Undoubtedly, installing a router is the first step in securing a network connection.
 - iii) Filtering Packets: Switching between packets and filtering packets are two more router services.

Disadvantages:

- i) slower: Routers are analyze multiple layers of information, from the physical layer to the network layer which slows down connection.
- ii) High Cost: They are more expensive than some other tools for systems administration. This includes security, extension and the focal point. As a result, routers are typically not the greatest option for issues.



F. Gateway: A gateway is a network node that forms a passage between two networks operating with different transmission protocols. The most common type of gateways, the network gateway operates at layer 3, i.e. the network layer of the OSI Open systems Interconnection model.

Functions :- i) Gateway is located at the boundary of a network and manages all data that inflows or outflows from that network.

- ii) It forms a passage between two different networks operating with different transmission protocols.
- iii) A gateway operates as a protocol converter, providing compatibility between different protocols used in the two different networks.
- iv) The feature that differentiates a gateway from other network devices is that it can operate at any layer of the OSI model.
- v) It also stores information about the routing paths of the communicating networks.

Types of Gateways :- i) Unidirectional-Gateway:

They allow data to flow in only one direction. Changes made in the source node are replicated in the destination node, but not vice-versa. They can be used as archiving tools.

ii) Bidirectional-Gateways: They allow data to flow in both directions. They can be used as synchronization tools.

Advantages : i) Connectivity: As mentioned earlier, the main benefit of gateway is the connectivity it provides. A gateway can expand the network by connecting computers with different systems together. Through this, same kind of information will be able to be accessed by different computers.

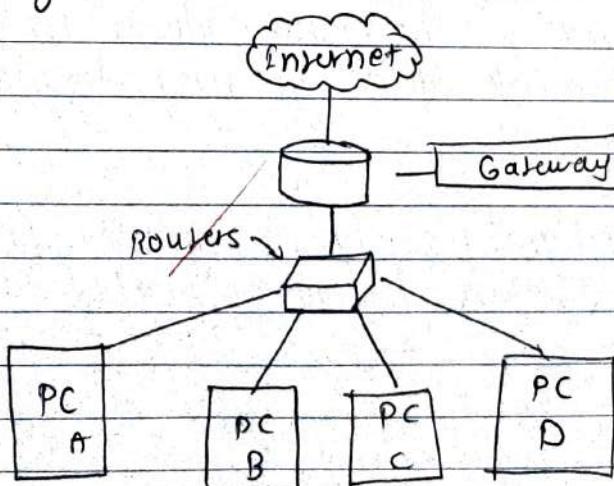
ii) Security: Gateways are known to possess improved security since they allow user authentication. Forms of security such as user ID and password can be imposed on gateway so that all the unwanted access will be prevented.

iii) Filtering: Filtering process is another important capability of a gateway. Without them whatever the services that arrives at the gateway carries the risk of threat.

Disadvantages :- i)- Gateway networks always causes time delay since information must be translated.

There is no gateway can a instant transfer take place.

ii) If there are possibilities of failure causing occurring at the gateway ,it can lead to communication loss.



6. MODEM: Modem stands for Modulation Demodulation. A modem converts the digital data signals into analogue data signals. They can be installed within the computer in a development slot applicable to it.

- Functions :-
- i) when a computer sends digital signals, it is converted into analog signals by modem. Once it reaches the destination computer, the signals are converted back into its original form.
 - ii) Modem possess high transmission rates since telephones lines are used here. Usually these rates are measured in a unit called baud. Despite offering high speeds, it can cost more too.
 - iii) Modems are considerably expensive. A typical 4G modem costs more than a router. However the prices may vary depending upon the router's battery capacity. And also modems offers flexible internet plans in terms of pricing.

Types of Modems :-

- i) standard Modem : The standard modems use generic device drivers, and they can be internal and external versions.

- ii) window Modem : A window modem is a private plug and plays tool. It requires a particular device driver supported by the window operating framework to function properly/correctly.

Advantages :-

- i) Signal Conversion :- When a computer sends digital signals, it is converted into analog signals by a modem. Once it reaches the destination computer, the signals are converted back into its original form.

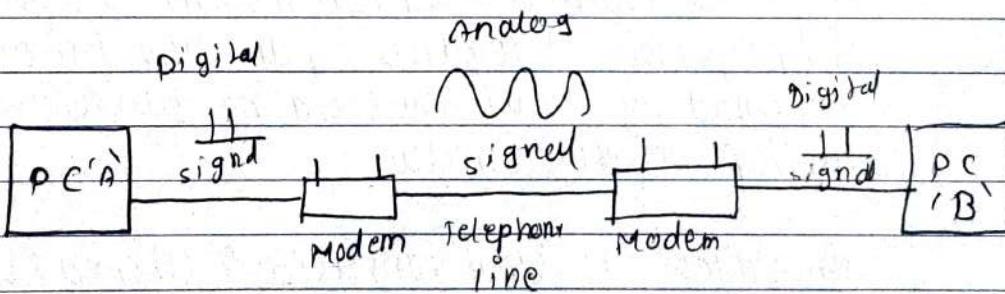
ii) Speed: Modem possesses high transmission rates since telephone lines are used here. Usually these rates are measured by in a unit called baud. Despite offering high speeds, it can cost more too.

iii) Cost: Modems are considerably expensive. A typical 4G modem costs more than a router. However the prices may vary depending upon the router's capacity. And also modems offer flexible internet plans in terms of pricing.

Disadvantages: i). One major drawback of connecting a modem is that it can make your computer vulnerable to hackers and malwares.

ii)- While external modems lack mobility, all the internal modems do support them. Users can use internal modems to benefit from wireless internet.

iii)- Modem especially of DSL type has a problem in availability. Services are not available in rural and remote locations. It solely depends upon the local phone company.



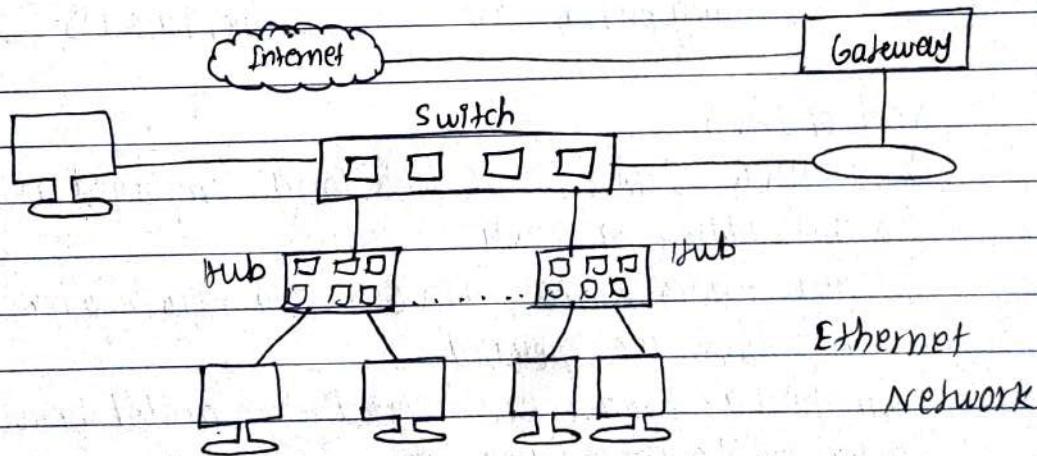
Assignment no : 2

i) Write a short note on:

i) Ethernet:

Ethernet is a type of communication protocol that connects computers on a network over a wired communication. It is a widely used LAN protocol, which is also known as CSMA/CD. It connects computers within the local area network and wide area network. Numerous devices like printers and laptops can be connected by LAN and WAN within buildings, homes and even small neighbours.

The wireless networks replaced ethernet in many areas; however Ethernet is still more common for wired networking. WiFi reduces the need for cabling as it allows users to connect smartphones or laptops to a network without the required cable.



Advantages of Ethernet:

- i) It is costly, but still inexpensive than other options.
- ii) Provides high security.
- iii) Quality of data is maintained.

Disadvantages of Ethernet:

- i) Distance covered is less.
- ii) No acknowledgement from receiver side.
- iii) Cannot determine which node is faulty.

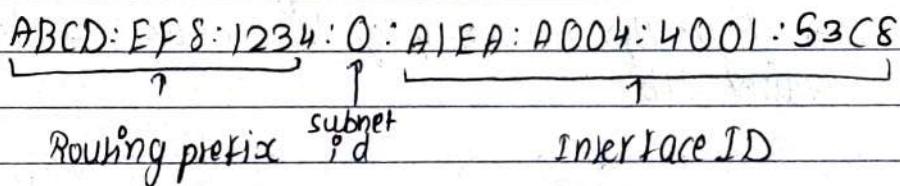
ii) IPv6 :

IPv6 was developed by Internet Engineering Task Force (IETF) to deal with the problem of IPv4 exhaustion.

IPv6 is a 128 bit address having an address space of 2^{128} , which is way bigger than IPv4. IPv6 uses hexadecimal format separated by colon(:).

Components of IPv6 address:

- i) 8-groups, each group represents 2 bytes (16 bits)
- ii) Each Hex-digit is of 4 bits (1 nibble)
- iii) Delimiter used - colon(:).



Need of IPv6 :

- i) An IPv6 address is 128 bits long compared with the 32-bit address of IPv4.
- ii) Better header format; IPv6 uses a new header format in which options are separated from the base header and inserted when needed between the base header and the upper layer data.

This simplifies and speeds up the routing process.

- iii) IPv6 has new options to allow for functionalities.
- iv) IPv6 allows extension of the protocol if required by new technologies.

Advantages of IPv6 :

- i) Real time data transmission
- ii) Supports authentication.
- iii) Performs encryption
- iv) Faster processing at router side.

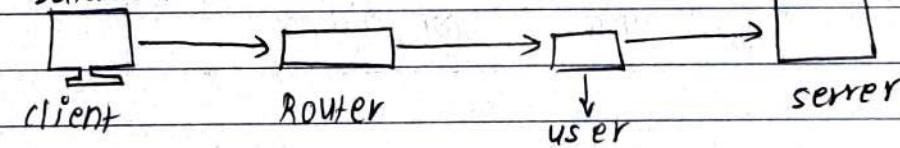
iii) SSH :

SSH stands for ~~secure~~ shell or secure socket shell. It is a cryptographic network protocol that allows two communicate and share the data over an insecure network such as internet. It is used to login to a remote server to execute commands and data transfer from one machine to another machine.

It provides a strong password encryption and password authentication communication with a public key over an insecure channel. It is used to replace unprotected remote login protocols such as telnet, login etc.

Before SSH :

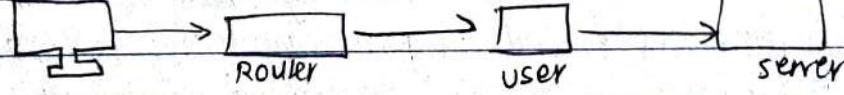
Data sent



Data received.

After SSH :

Data sent



Data received.

Advantages of SSH :

- i) It prevents malicious activities like DNS spoofing, Data manipulation, eavesdropping or sniffing of transmitted data, IP address routing.
- ii) It uses strong authentication methods which provide more security.
- iii) SSH enables port forwarding.

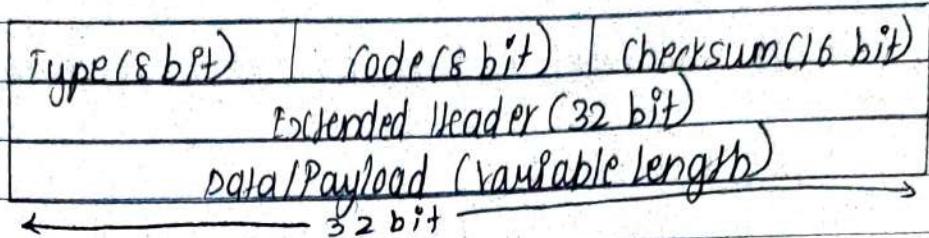
Q2) Explain the purpose of following protocols with their header:

i) ARP: The address resolution protocol (ARP) is a fundamental networking protocol that plays a pivotal role in local network communication by mapping known IP address to their corresponding physical MAC address. It enables devices on a network to discover and associate MAC addresses with IP addresses.

32 bits	
Hardware type	Protocol type
Hardware length	Protocol operation
length	1-request 2-Reply
sender Hardware Address	
sender Protocol Address	
target Hardware Address	
target Protocol address	

ii) ICMP: ICMP is used for error reporting; if two devices connect over the internet and some error occurs, so router sends an ICMP error message to the source informing about the error. for example, whenever a device sends any message which is large enough for the receiver, in that case, the receiver will drop the message and replace ICMP message to the source.

Another important use of ICMP protocol is used to perform network diagnosis by making use of trace route and ping utility.



Type: Basic description of message

Code: Additional information

Checksum: Checksum of ICMP.

Extended header: Points out problem in IP message.

Data or Payload of variable length.

iii) DNS: The Domain Name System (DNS) serves the vital role of translating user friendly domain names into numerical IP addresses enabling seamless internet communication. It ensures user can access websites and services using easily memorable names, simplifying web browsing and resource location. It also provides data packets addressing and routing.

Identification	Flags
No. of Questions	No. of answers RRs
No. of Authority RRs	No. of additional RRs

Identification: Used to match the response with the request.

Flags: 16 bit flags for indication of values. No. of questions, Number of authority RRs, No. of answers RRs and No. of additional RRs are self-explanatory.

Q3)- Discuss the persistent and non persistent protocols used in transport and application layers of TCP/IP protocol suite.

Persistent and Non Persistent protocol are two different approaches to handling connections in the transport and application layers of TCP/IP protocol suite. These protocols

govern how data is exchanged between two devices over a network.

Persistent Protocol:

- i) Connection oriented: Persistent protocols are connection oriented which means they establish a communication between the sender and receiver before data exchange begins. This connection remains open for multiple transactions.
- ii) Example: The most common example of a persistent protocol in TCP/IP suite is the transmission control protocol (TCP). TCP ensures reliable, ordered and error checked delivery of data. It maintains a continuous until explicitly closed by either the sender or receiver.
- iii) Use cases: Persistent protocols are suitable for applications that require reliable and ordered data transfer such as web browsing, email, file transfer and most client server instructions.
- iv) Overhead: They typically have overhead due to the need for connection setup & maintenance, which involves three way handshake, acknowledgement message and error recieving mechanisms.

Non Persistent Protocol:

- i) Connectionless: Non persistent protocols are connection-less meaning they don't establish a continuous connection between sender and receiver. Instead a new communication is established for each data exchange and closed after that exchange is complete.
- ii) Example: A common example is User Datagram Protocol (UDP).
- iii) Use cases: Non persistent protocols are suitable for applications that prioritize speed and efficiency over reliability. Commonly used in real-time data sharing.
- iv) Overhead: Lesser overhead as compared to persistent protocols because they skip communication setup.

Persistent Protocols: App: HTTP, SMTP, Transport Layer: TCP

Non-persistent protocols: App: DNS, SNMP, DHCP, Transport Layer: UDP, SCTP