

Experiment-1

Title :- Study of RJ45 & CAT6 cabling & connection using crimping tool.

Theory :-

RJ45 & CAT6 cabling are widely used in networking to connect various devices such as computer, router, switches and access point to create LAN and establish internet connectivity. The process of creating these connection involves the use of crimping tool to terminate the cable with RJ45 connector.

What is Straight-Through Connection?

Straight-through cable is a type of CAT5 with RJ-45 connection at each end, and each has the same pin out. It is in accordance with either the same color code throughout the LAN. For consistency. This type of twisted-pair cable is used in LAN to connect a computer or a network hub such as a router. It is one of most common types of network cable.

What is crossover connection?

The crossover cable is a type of CAT5 where one end is T568A configuration and other end as T568B configuration. In this type of

cable connection, pin 1 is crossed with Pin 3, and Pin 2 is crossed with Pin 6. The internal wiring of cross over cable reverses the transmission and receive signals. It is widely used to connect two devices of the same type. e.g. two computers or two switches to each other.

RJ45 Connector (Registered Jack 45) :-

RJ45 is most common type of Ethernet connector used for wired network connection. It is an 8P8C modular plug that can accommodate eight wires allowing for data transmission and communication between network devices. The connector name indicates that it has eight position & eight contacts.

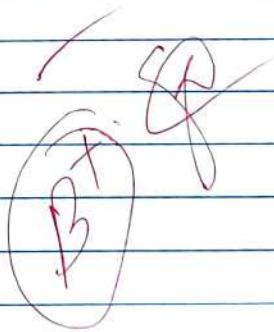
CAT6 Cable (category 6 cable)

CAT6 is type of twisted Ethernet cable which means it consists of pair of copper wires twisted together to reduce interference & crosstalk. It is an improvement over its predecessor, CAT5e, offering higher data transfer speeds and improved performance.

Procedure of experiment (crimping process)

- 1.) Strip the cable back 1 inch (25mm) from the end.
- 2.) Untwist and straighten the wires inside of the cable

- 3.) Arrange the wires into right order. The proper sequence is as follows from left to right.
Orange/white, Orange, Green/white, Blue, Blue/white, Green, Brown/white, Brown
- 4.) Cut the wires into even line $1/2$ inch [13mm] from sheathing.
- 5.) Insert the wires into the RJ45 connector. If any small wires bend or don't fit into a groove correctly, take the cable out and straighten the wires with your finger before trying again.
- 6.) Stick the connector into the crimping part of the tool and squeeze twice.
- 7.) Remove the cable from tool and check that all of pins are down.



EXP - 2

CN - C32 - 2103164

AIM - Implementation of Hamming Code for error correction and detection

Theory :- The Hamming Code is a widely used for error correction and detection in digital communication.

The theory behind the Hamming Code involves adding parity bits to data stream to detect and correct error. It works as follows:-

Data Encoding :- For input message of ' K ' bits, (r) parity bits are added at positions that are powers of 2 (1, 2, 4, 8 etc). The positions of these parity bits are calculated by using binary representation.

Each parity bit is responsible for checking subset of data bits.

Parity Calculation :- Each parity bit calculates its value based on data bits that it covers. For eg, parity bit at position 1 checks all data bits that have a 1 in their least significant bit and so on. Parity bits are calculated using even parity, which means total numbers of 1's in group of bits should be even.

Error Detection :- During transmission, if a bit get flipped due to noise, the parity bits will no longer match their calculated values. This discrepancy indicates an error.

Error Correction :- If an error is detected, the erroneous bit position can be determined by looking at parity bits

CN-C32-2103164
 that cover that position. By using the pattern of incorrect parity bits, the erroneous bits position can be identified and corrected.

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

Its limitations, is it cannot correct more than two errors or detect error in parity bits themselves.

Problem :- A 7 bit Hamming code is received as D₇ D₆ D₅ P₄ D₃ P₂ P₁. Assume even parity & state whether the received code is correct or wrong, if wrong locate the bit in error.

Solution:- Received HC :-

D ₇	D ₆	D ₅	P ₄	D ₃	P ₂	P ₁
1	0	1	1	0	1	1

Detecting errors :-

Step 1: Analyzing bits 1, 3, 5, 7

We have P₁ D₃ D₅ D₇ \Rightarrow 1011 { odd parity error }

We put P₁ = 1

SN-C32-2103164

Step 2: analyzing bit 2, 3, 6 & 7

$P_2 D_3 D_6 D_7$

$\Rightarrow 1 \ 0 \ 0 \ 1$

{ Even parity }
 No errors }

$\Rightarrow P_2 = 0$.

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

We have $P_4 D_5 D_6 D_7$

$\Rightarrow 1 \ 1 \ 0 \ 1$ { odd parity }
 { error exists }

We put $P_4 = 1$

Correcting error

error word $E = \boxed{P_4 \ P_2 \ P_1}$
 $E = \boxed{1 \ 0 \ 1}$

decimal value $e = 5$ which shows that 5th bit
 is in error

We write the correct word by simply inverting
 the 5th bit

\therefore Correct word = 1 0 0 1 0 1 0 ~~1~~

✓ 27/08/2013
 (A)

```
option=int(input('Press 1 for generating hamming code \nPress 2 for finding error in hamming code\n\tEnter your choice:--\n'))\n\nwhile(option>0 and option<3): # GENERATE HAMMING CODE\n\n    if(option == 1):\n        print('Enter the data bits')\n        d=input()\n        data=list(d)\n        data.reverse()\n        c,ch,j,r,h=0,0,0,0,[]\n\n        while ((len(d)+r+1)>(pow(2,r))):\n            r=r+1\n\n            for i in range(0,(r+len(data))):\n                p=(2**c)\n\n                if(p==(i+1)):\n                    h.append(0)\n                    c=c+1\n\n                else:\n                    h.append(int(data[j]))\n                    j=j+1\n\n        for parity in range(0,(len(h))):\n            ph=(2**ch)\n\n            if(ph==(parity+1)):\n                startIndex=ph-1\n                i=startIndex\n                toXor=[]\n\n                while(i<len(h)):\n\n                    toXor.append(h[i])\n                    i=i+1\n\n                result=reduce(lambda x,y:x^y,toXor)\n                if(result==0):\n                    h[ph-1]=0\n                else:\n                    h[ph-1]=1\n\n            ch=ch+1\n\n    else:\n        print('Enter the hamming code')\n        h=input()\n        data=list(h)\n        data.reverse()\n\n        for i in range(0,(len(h))):\n            p=(2**c)\n\n            if(p==(i+1)):\n                h.append(0)\n                c=c+1\n\n            else:\n                h.append(int(data[j]))\n                j=j+1\n\n        for parity in range(0,(len(h))):\n            ph=(2**ch)\n\n            if(ph==(parity+1)):\n                startIndex=ph-1\n                i=startIndex\n                toXor=[]\n\n                while(i<len(h)):\n\n                    toXor.append(h[i])\n                    i=i+1\n\n                result=reduce(lambda x,y:x^y,toXor)\n                if(result==0):\n                    h[ph-1]=0\n                else:\n                    h[ph-1]=1\n\n            ch=ch+1\n\n        print(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')

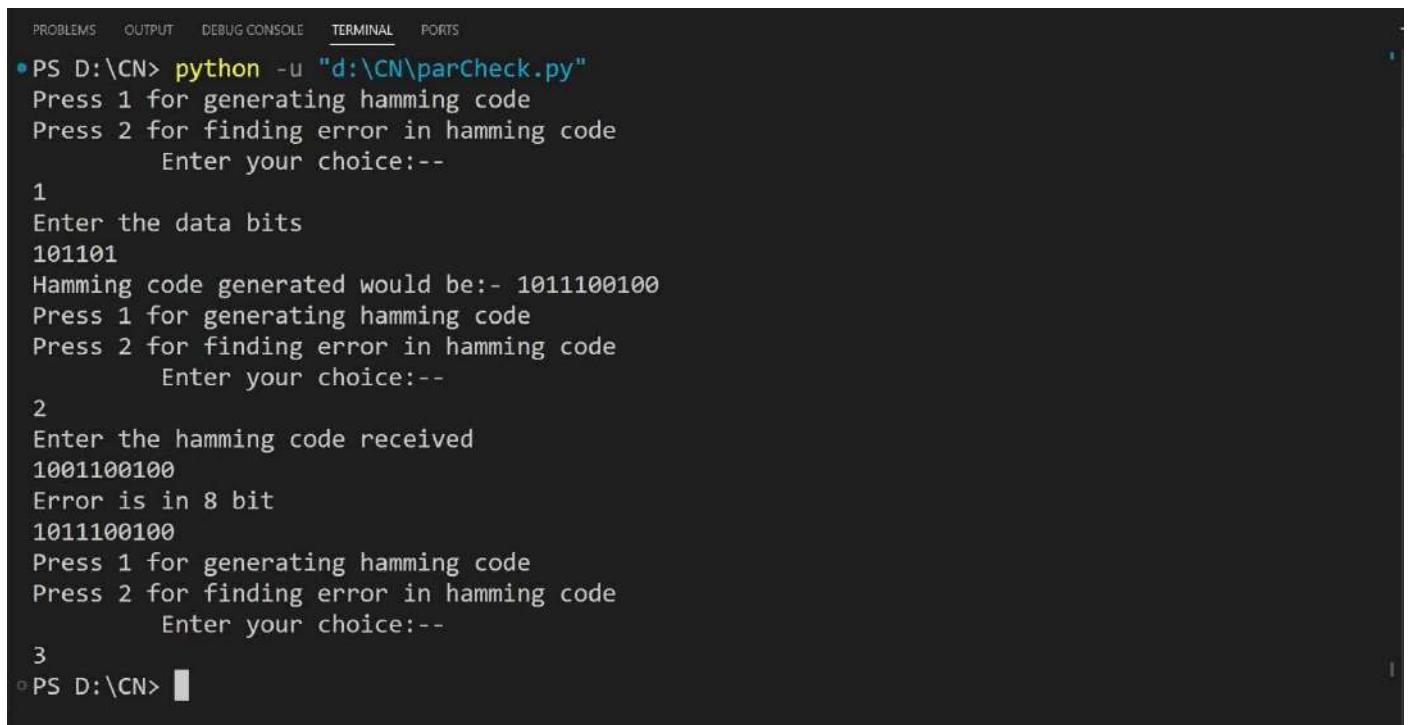
    if(h_copy[error-1]=='0'):
        h_copy[error-1]='1'

    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



```
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>
```

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.

Bits 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} . Example : The bit sequence 10011010 corresponds to this polynomial :

Sending and receiving message can be imagined as exchange of polynomials

$$M(x) = 1 * x^7 + 0 * x^6 + 0 * x^5 + 1 * x^4 + 1 * x^3 + \\ 0 * x^2 + 1 * x^1 + 0 * x^0 \\ = x^7 + x^4 + x^3 + x^1$$

The Data Link Layer Protocol specifies a generator polynomial $G(x)$. Generator polynomial is available for both sender and receiver side.

CN-C32-2103164

- $C(x)$ is a polynomial of degree K
- If e.g.

$$C(x) = x^3 + x^2 + x^0 \\ = 1101, \text{ then } K = 3$$

- Therefore, generator polynomial is degree 3.
- The degree of generator polynomial is equal to, of bits minus one.
- If for a frame, the CRC need to be calculated, are appended to the frame.
- n corresponds to degree of generator polynomial

generator polynomial	10110
----------------------	-------

- The generator polynomial has 6 digits
 - Therefore, five 0 bits are appended.

Frame (payload)	10101
Frame with appended bit	101010000

Sender Side (Generation of Encoded Data from Data and Generator Polynomial (or Key)):

binary data is first augmented by adding n zeroes in end of the data (n degree of generator polynomial)

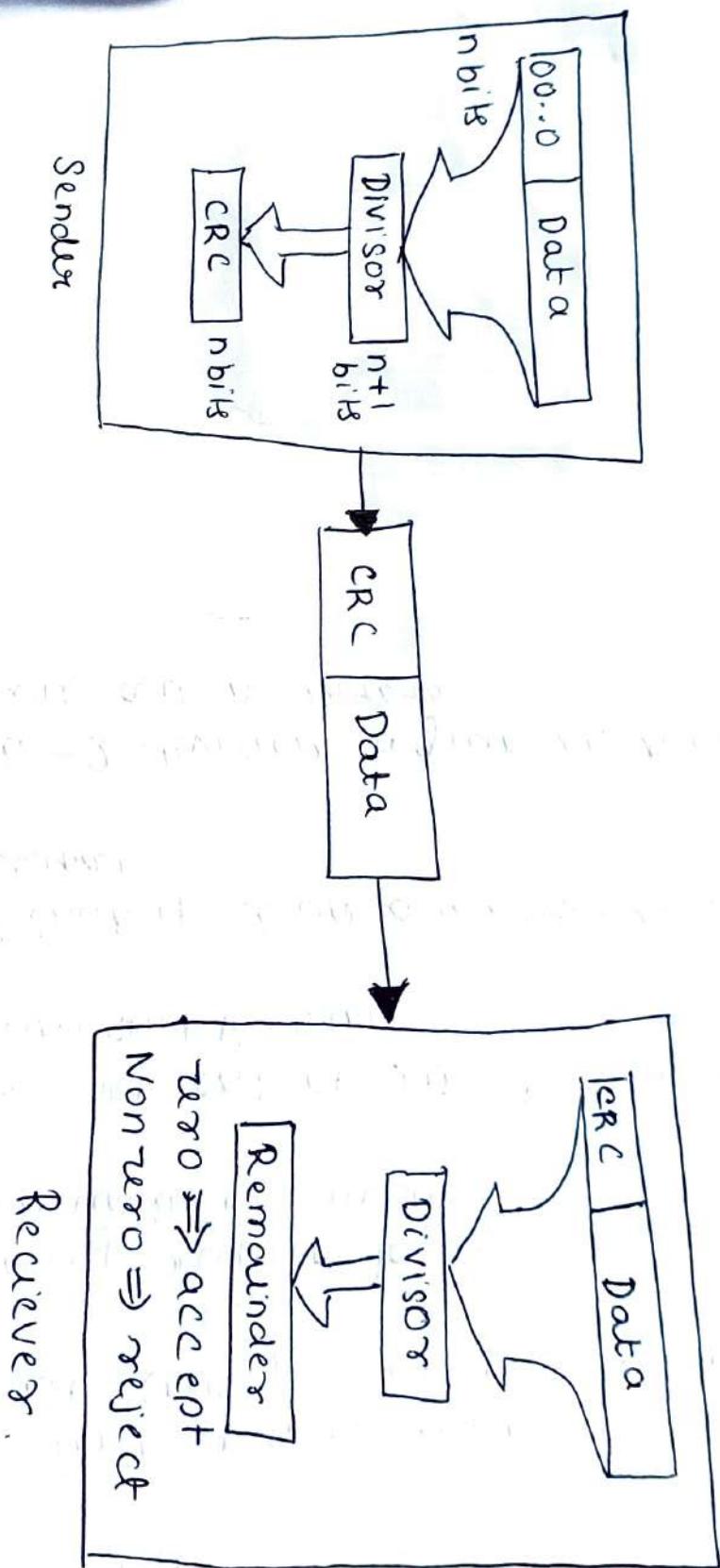
use modulo-2 binary division to divide binary data by key and store remainder of division.

Append remainder at end of data to form the encoded data and send the same.

Receiver Side (Check if there are errors introduced in transmission)

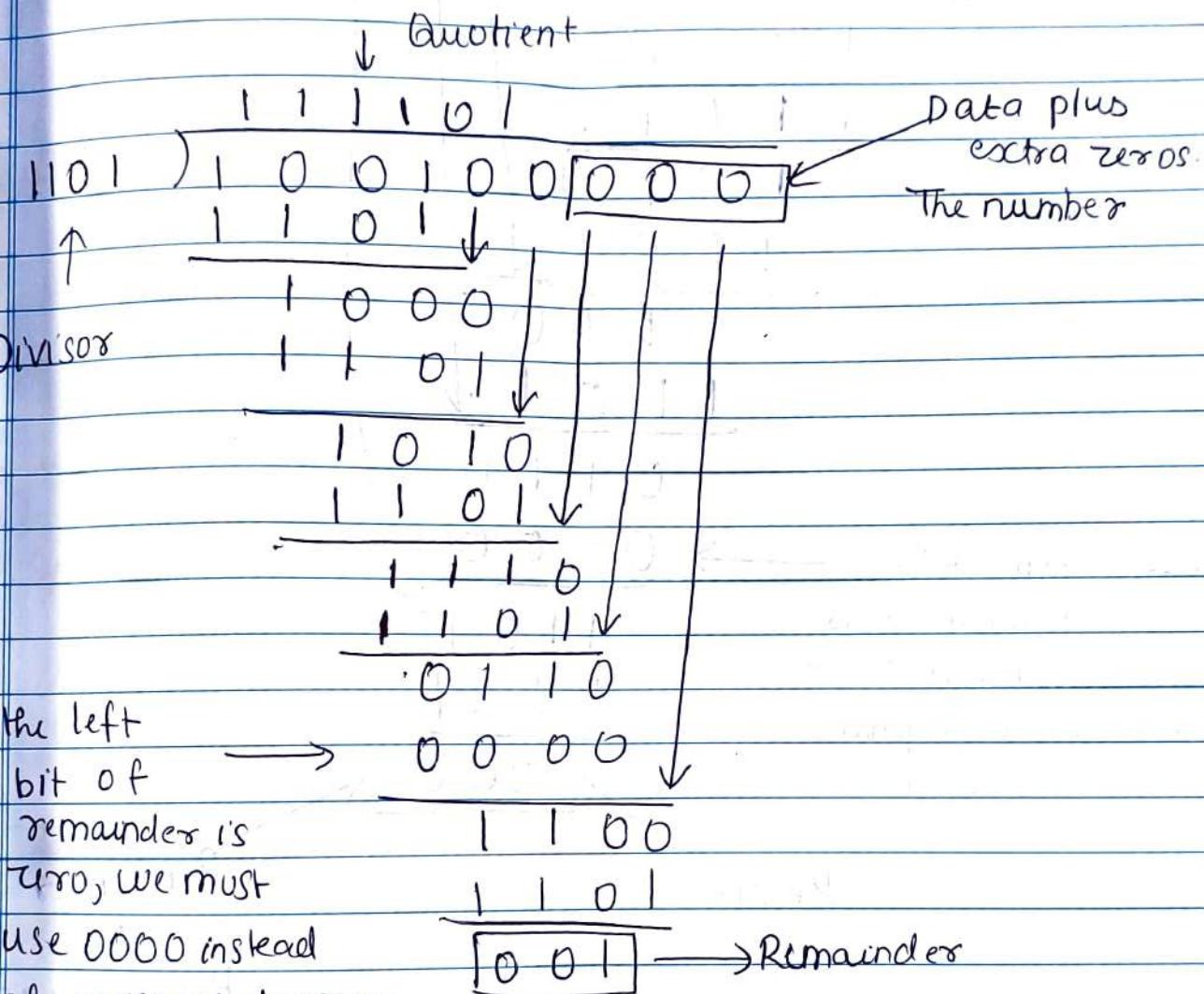
Perform modulo-2 division again if the remainder is 0, then there are no errors

CRC Cyclic Redundancy



Eg:-

→ CRC generator
uses modular-2 division



Divisor Quotient

$$\begin{array}{r}
 \text{Quotient} \\
 \downarrow \\
 1101) \overline{111101} \\
 1101 \quad \downarrow \\
 \hline
 1000 \\
 1101 \quad \downarrow \\
 \hline
 1010 \\
 1101 \quad \downarrow \\
 \hline
 1110 \\
 1101 \quad \downarrow \\
 \hline
 0110 \\
 0000 \\
 1101 \quad \downarrow \\
 \hline
 0000
 \end{array}$$

Data
CR
reqd

When leftmost bit of remainder is zero, we must 0000 instead of original divisor.

Result

Remainder

000

Hence acc

X

Program

```
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 content:

- PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
- 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>

EXP 4

CN-C32-2103164

Aim :- Implementation of Go back n sliding window.

Theory :-

- Discard all subsequent frames following the damaged frames sending no ACKs.

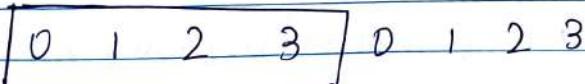
- Eventually the sender times out and retransmits all the unacknowledged frames in order starting with the damaged or lost one.

Operation :-

Assume no of bits in sequence no 2.

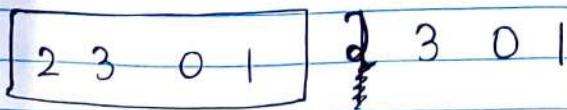
Window size :- 4 frames.

Sending window



- Sender has sent 4 frames 0 to 3 and waits for an ACK.

- ACK 0 and ACK 1 received by the sender.



- Now sent 0 1
- Frame 2 lost not received but received 3
- Receiver discards 3 0 and 1

Note

A receiving station does not acknowledge each received frame explicitly. If a sending station received an ack for frame j and receives an ack for frame K it assumes frames between j and K have been received correctly.

- Advantages :- Reduces no of acks and lessens n/w traffic.
- What should be max sender's and receiver's window size?
- Assume $K \rightarrow$ no of bits in sequence no.
- Frames are numbered from 0 to $2^K - 1$.
- Window size cannot be larger than 2^K .

Case 1: Window size larger than 2^K .

Let $K = 3$

Assume window size = 9.

Sender window →

0	1	2	3	4	5	6	7	0	1	2	3
---	---	---	---	---	---	---	---	---	---	---	---

Problem →

Sender receives an ACK 0. Does not know if it was for first / last frame.

Window size must be less than or equal to 2^K

Case 2: Window size = 2^K

- 1) At time t_1 A sends frames 0-7 to B
- 2) B receives each one recently.
- 3) At time t_2 , B sends ACK for the most received frame ACK 7.
- 4) ACK get lost
- 5) Next frame expected by B is frame numbered 0.
- 6) It does not receive ACK and hence re-transmits frames 0 through 7 at time t_3 .
- 7) At t_4 B receives frame 0. Sequence no matches with the one it is expecting. Hence B accepts it

Protocol fails (since B has accepted a duplicate and not a new frame)

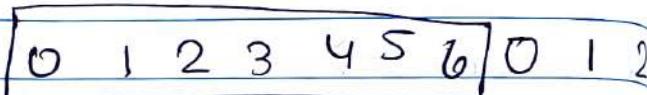
CN - C32 - 2103164

Reason

- 2 consecutive windows contain the same sequence no.
- Solution reduce the sender's window size

Case 3 :- Window size less than 2^K .

Sender's window →



- 1.) A sends frame 0 through 6 at time t1.
- 2.) B receives each one correctly.
- 3.) At time t2, B sends ACK for the most recently received frame ACK 6.
- 4.) ACK gets lost.
- 5.) Next frame expected by B is frame number 7.
- 6.) A does not receive ACK and hence retransmits frame 0 through 6 at time t3.
- 7.) B receives frame 0 through 6 at time t4.
- 8.) B expects frame 7.
- 9.) Hence ignores them.
- 10.) Eventually B sends another ACK 6 which A receives and A advances its window to include 7. 0 1 2 3 4 5 and protocol continues.

Operation performed

CN - C32-2103164

- Receiver sends a positive ACK if frame has arrived without error and in order (with accepted seq number)
- Receiver does not have to acknowledge each individual frame received correctly and in order.
- Receiver can send cumulative ACK for several frames (ACK 4 acknowledges frames (0, 1, 2, 3, 4))
- If frame is damaged or out of order, the receiver discards it and also discards all subsequent frames until it receives the one accepted.
- In this case, no ACK will be transmitted.

If the sender timer expires before receiving ACK, it will resend ALL frames beginning with one expired until last one set.

Drawback Go-back-n :-

- Receiver discards all correct frames transmitted after bad one.
- Channel bandwidth wasted on retransmitted frames.

Alternative Strategy :-

Allow the receiver to accept and buffer the frames following a damaged / lost one. - Called Selective Repeat

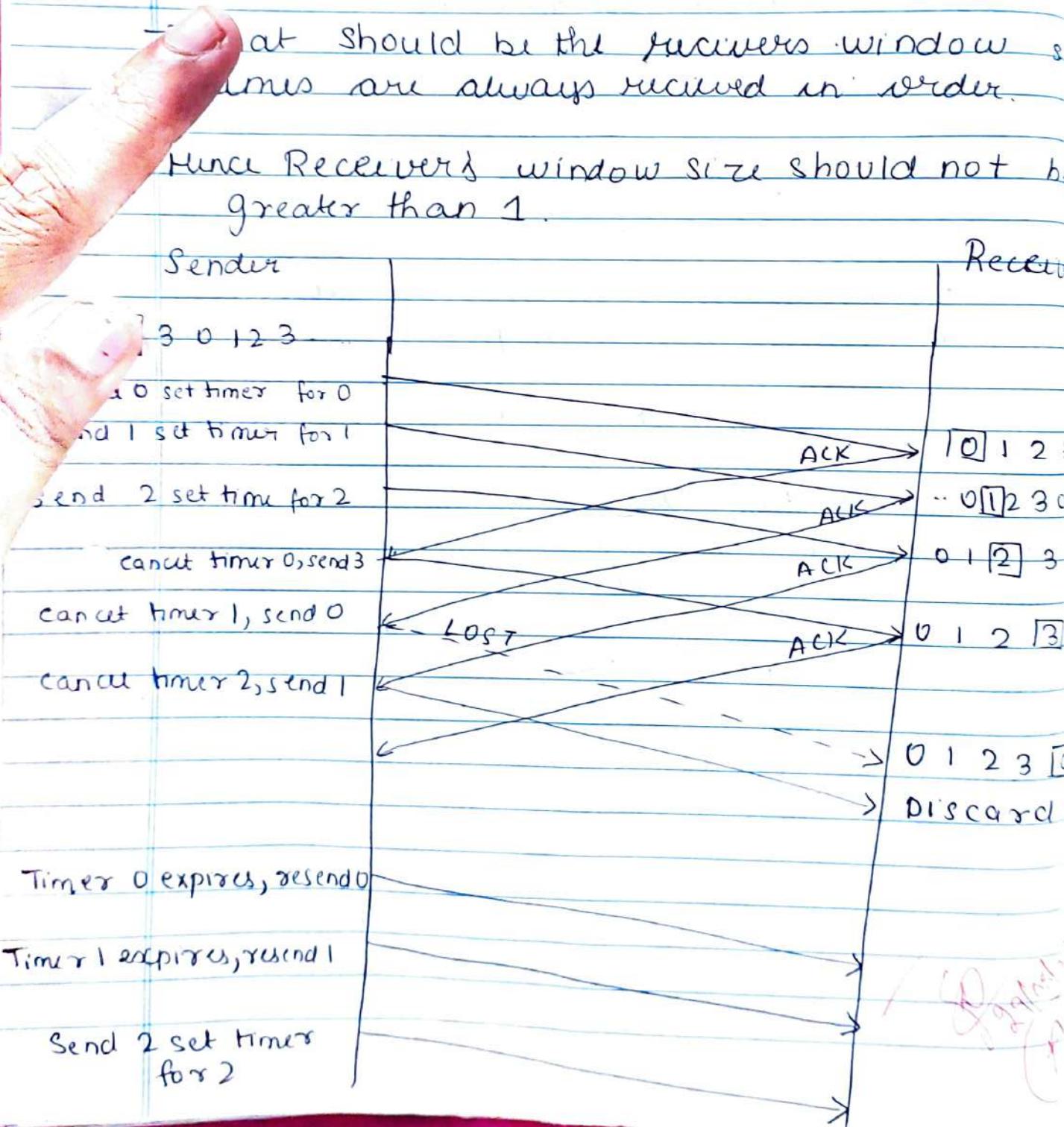
CN - C32 - 2103164

Summary

→ Sender window's size - strictly less than 2
(1 less than MAX SEQ)

→ That should be the receiver's window
frames are always received in order.

Hence Receiver's window size should not be greater than 1.



```
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 NO - 5

CN - C32 - 2103164

Aim :- Build a simple 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 physical or logical layout of devices and connection within a network. Different network topologies are suited to different applications and have unique advantages and disadvantages. Here are a few different network topologies along with small examples of their application.

Star Topology :-

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

Application : - Star topology is mainly used in home network where all devices connect to central router or switch. This topology is straightforward to set up and manage.

Bus Topology :-

Description : - In bus topology, all devices share a single communication line (bus). Data bounces back and forth between devices.

mited down the bus and devices can "tap" to access the data.

Application :- 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 :-

Description :- In ring topology, each device is connected to exactly two other devices, forming a closed loop. Data travels in one direction around the ring.

Application : Ring Topology are less common in modern LANs but are used in specific applications like fiber optic network. Token Ring Network used this topology in the past.

Mesh Topology :-

Description :- In mesh topology, every device is connected to exactly two or more devices, creating multiple paths for data to travel.

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

Hybrid Topology :-

Description:-

A hybrid topology combines two or more different topologies into single network. For example, a network might have star topology in one office and ring topology in another office connected through router.

Application: Hybrid topologies allow organization to tailor their network to specific needs. They are commonly found in larger enterprises with diverse networking requirements.

Tree Topology :-

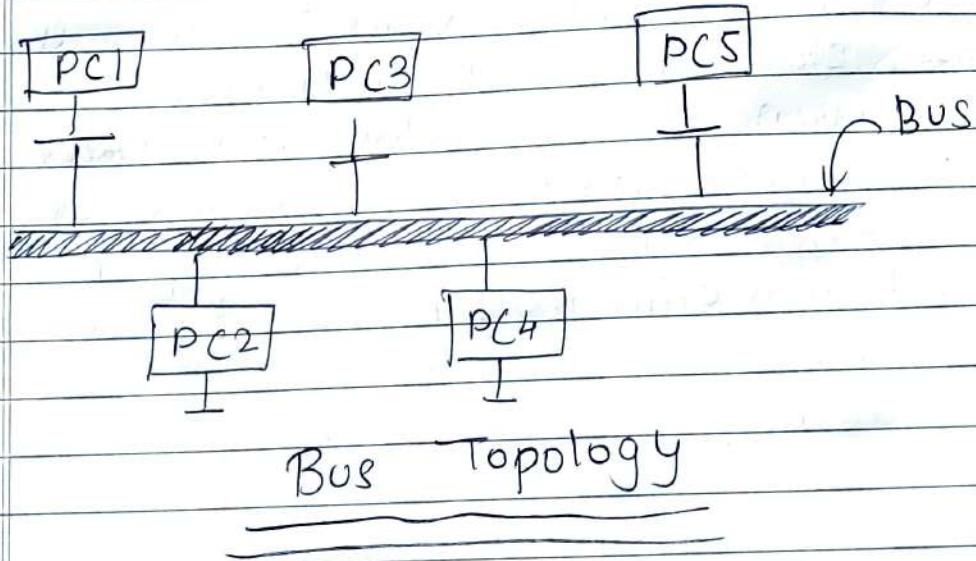
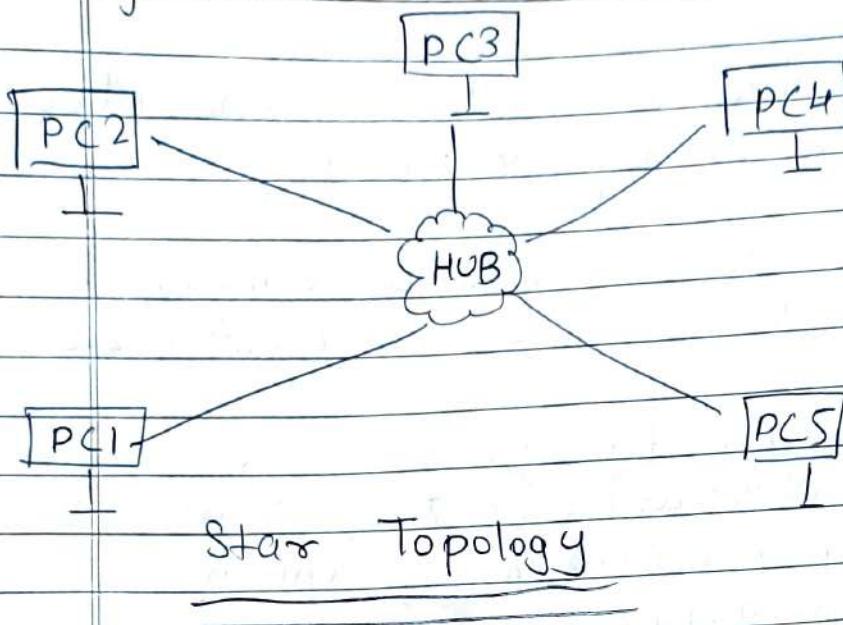
Description:-

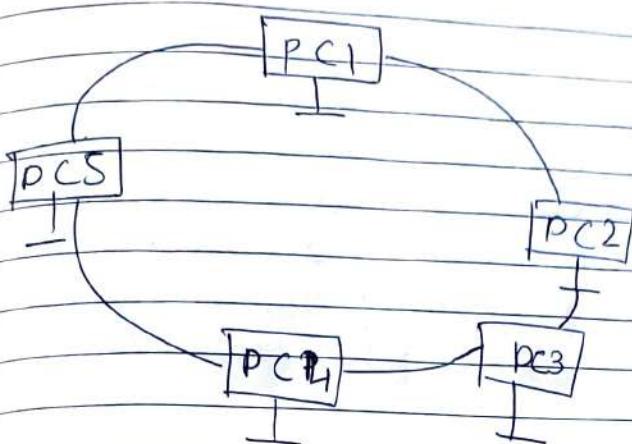
Also known as Hierarchical Topology, there are star connected to central hub.

Application:- Tree topologies are often used in large scale networks such as corporate networks with multiple branches. Each branch office may have star topology, they all connected to main office using a hierarchical structure.

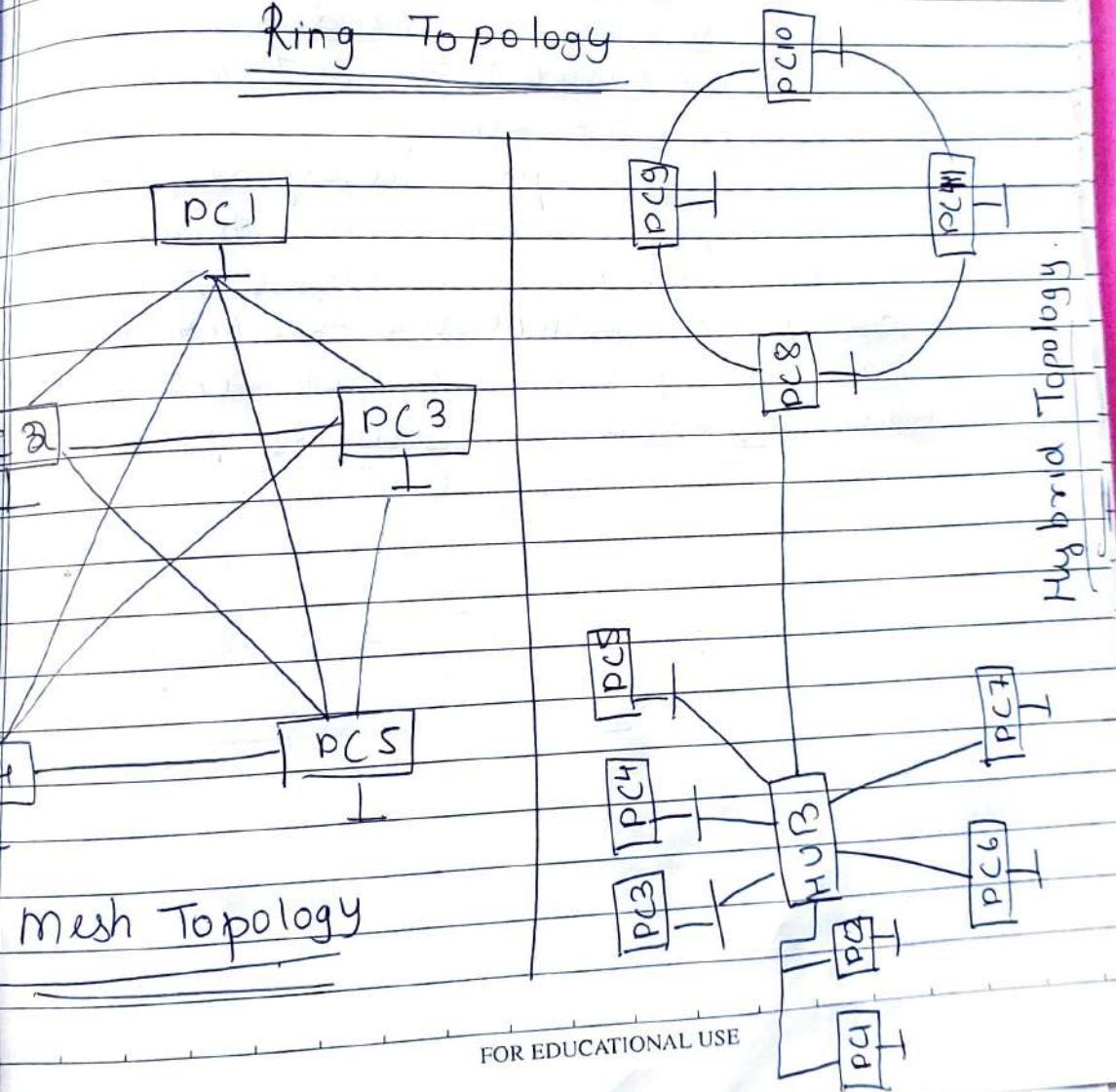


Diagrams :-





Ring Topology



FOR EDUCATIONAL USE

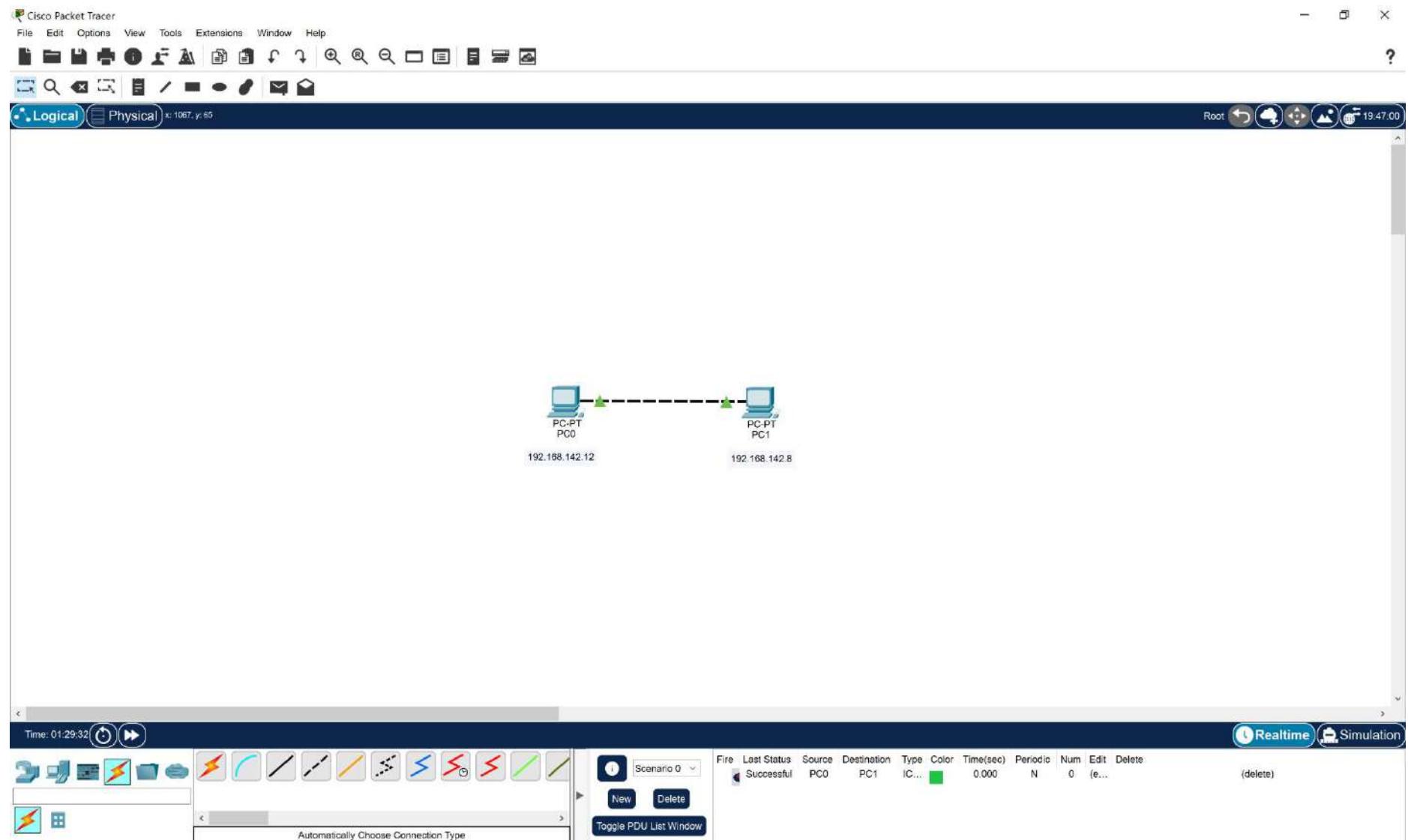
* Cisco packet Tracer :-

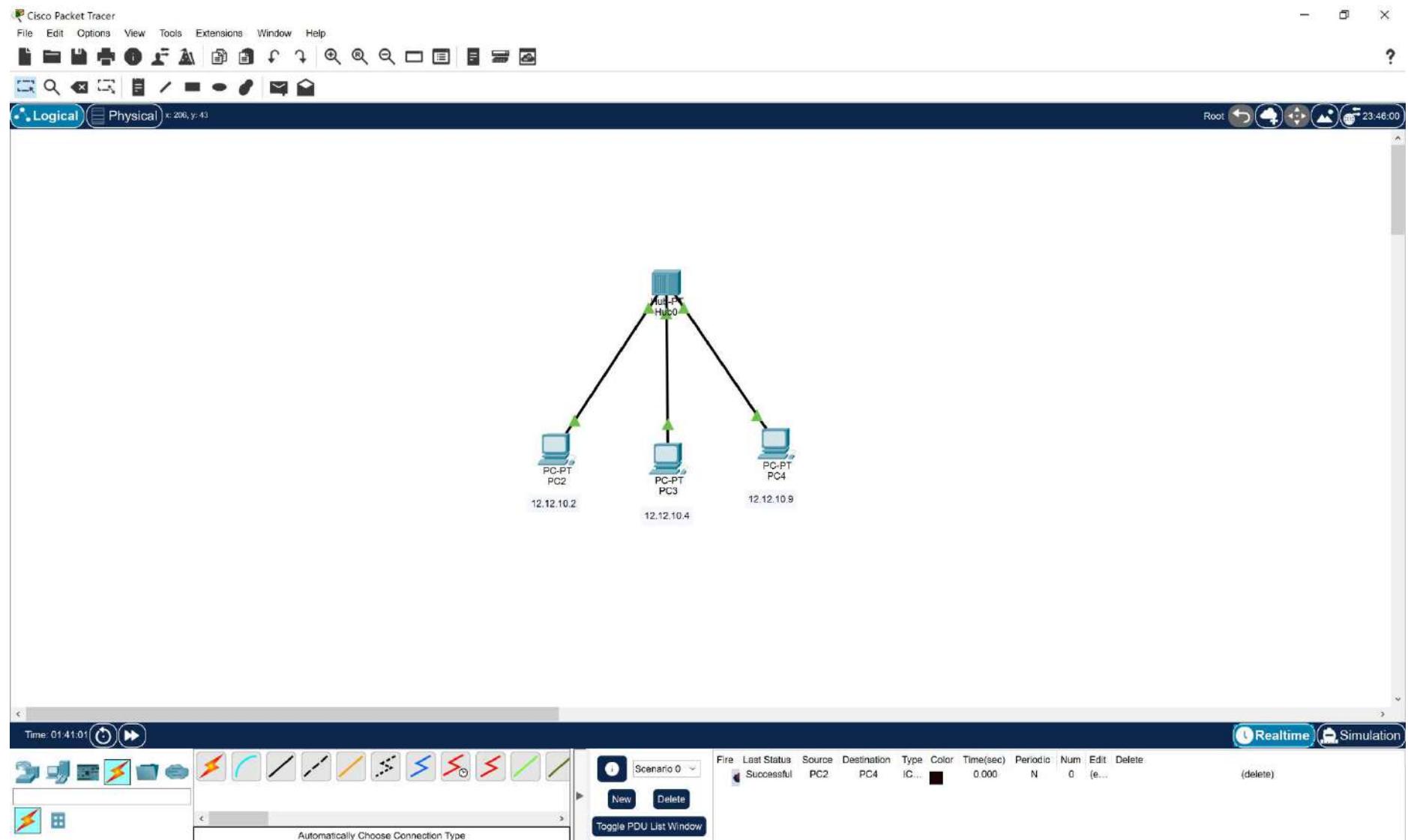
Cisco packet Tracer is a versatile network simulation tool developed by Cisco Systems. It serves as a valuable resource for both networking professionals and learners, enabling them to design, configure, and test network setups in virtual environment.

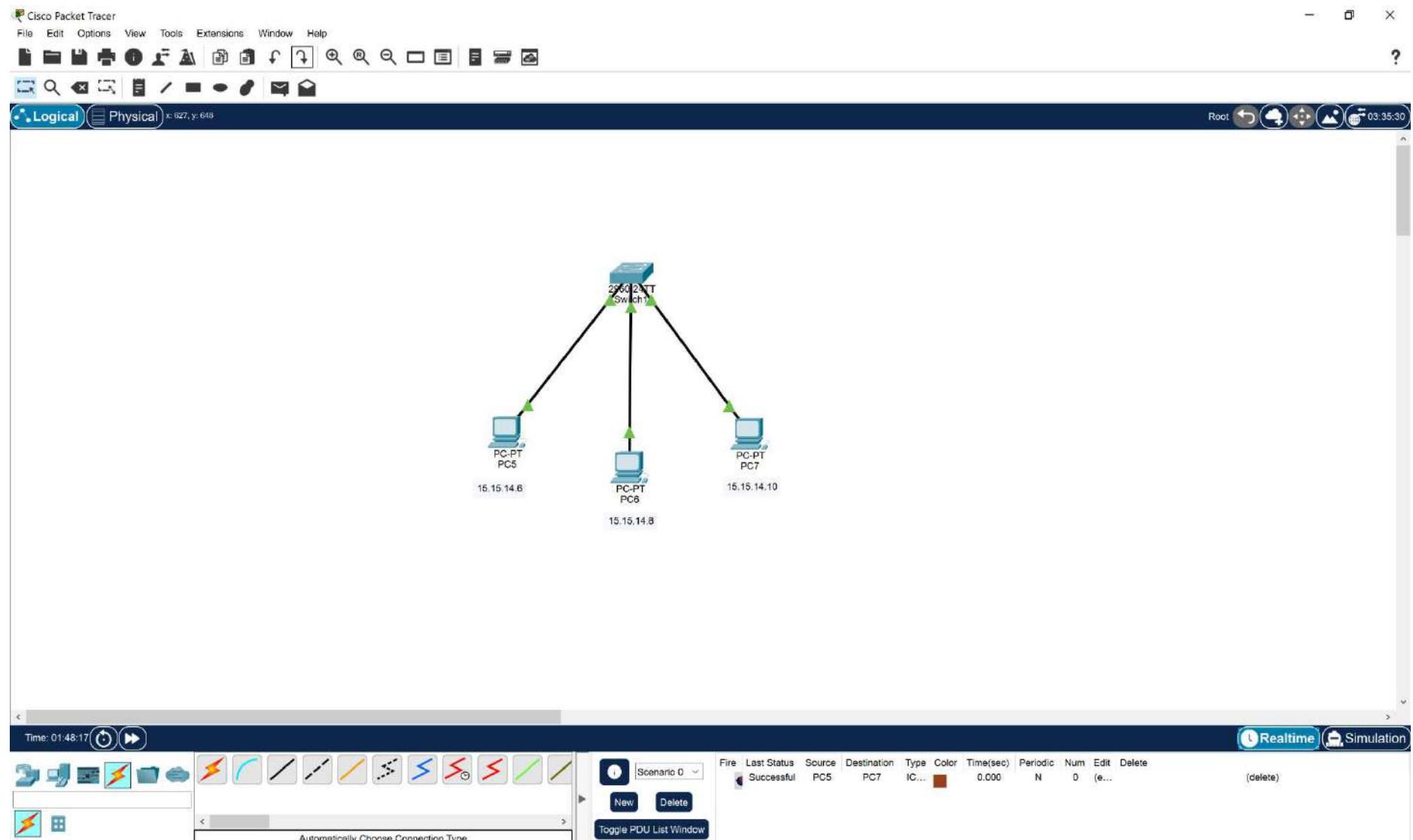
Packet Tracer offer a user friendly interface and wide range of Cisco devices and network components, allowing users to build and experiment with various network topologies, troubleshoot issues and gain practical experience without the need of physical hardware. It's commonly used in educational settings to teach network concepts and is valuable tool for gaining hands on experience in networking before working with real-world equipment.

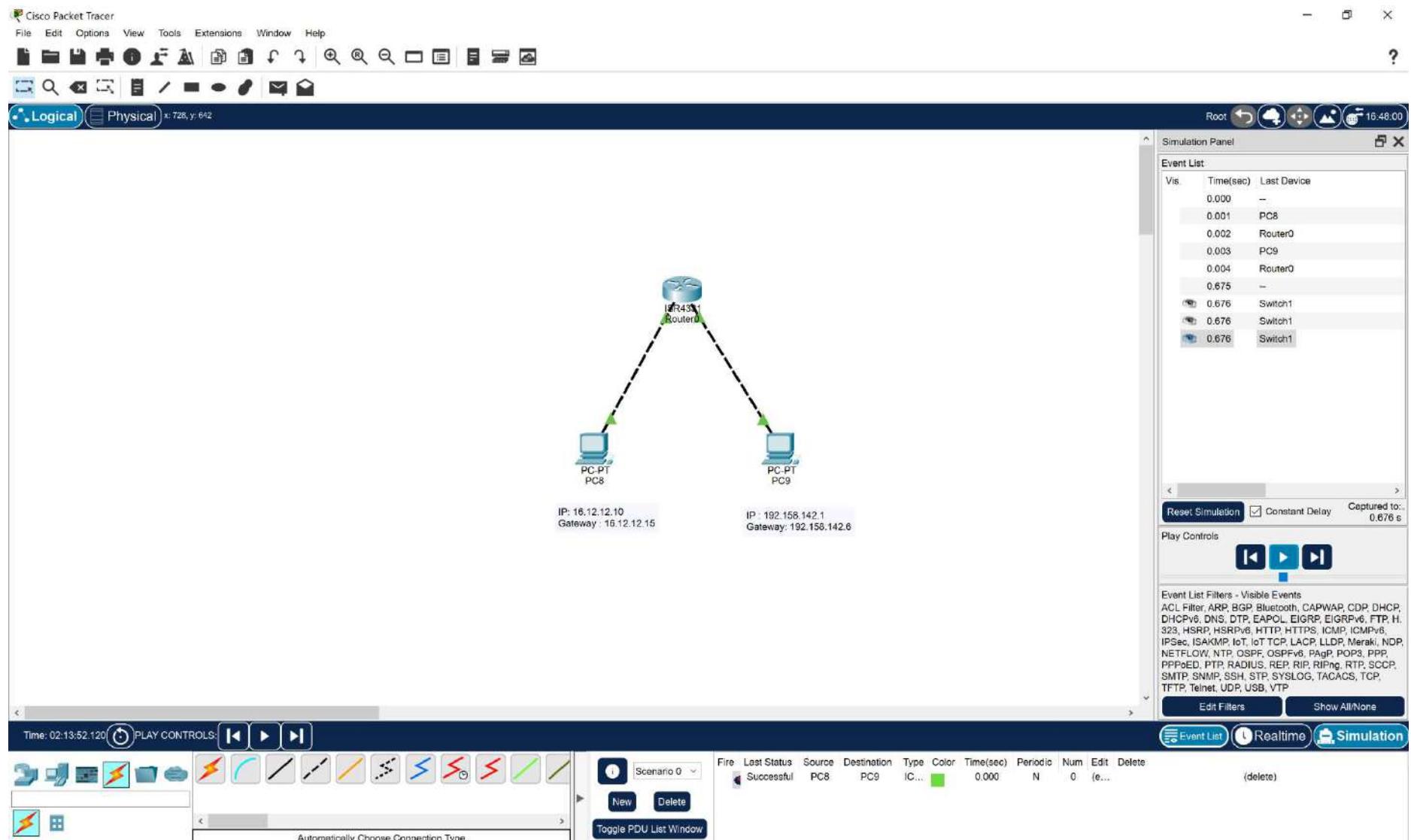
80
100
80

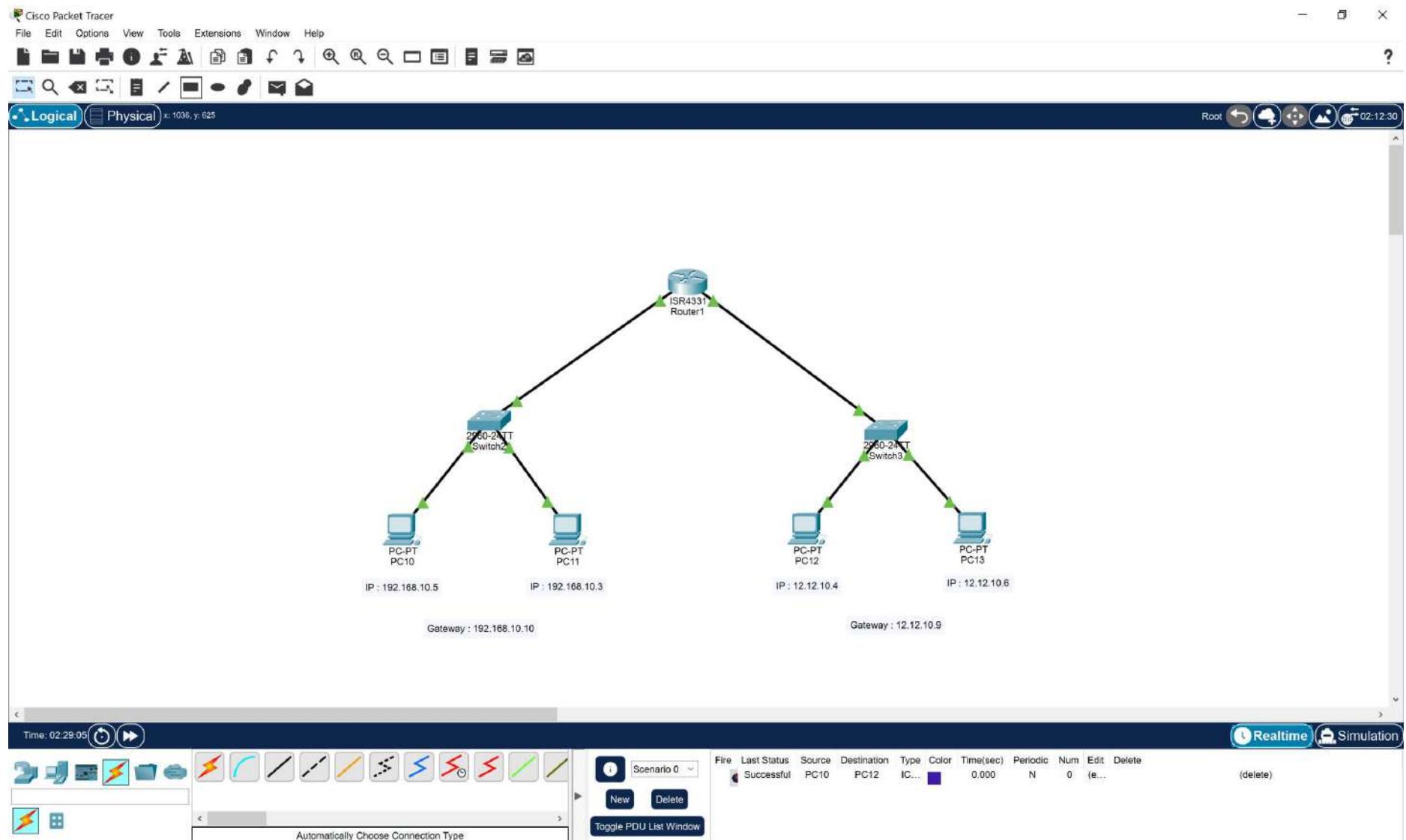
X
B











Experiment - 6

CN - C32 - 2103164

Aim :- Design VPN & configure RIP/OSPF using packet tracer.

Theory :-

A VPN or Virtual Private Network, is a technology that allows you to create a secure and encrypted connection over a less secure network, such as the internet. VPNs are commonly used to provide privacy and security for data transmission over the internet. Here's how a VPN works in context of computer networking.

1.) Encryption and Tunneling :- When you connect to VPN, your computer or device creates a secure "tunnel" over the internet to remote server operated by VPN service provider. The tunnel is encrypted, which means that all data travelling between your device and VPN server is encrypted and secure from eavesdropping.

2.) Hide Your IP Address :- Your IP address like your online identity, and it can reveal your location and other information about you. When you connect a VPN, your real IP address is hidden and VPN address is used. This helps protect your privacy & make it harder for websites and other online

services to track your location.

3) Secure Data Transmission :- Any data you send and receive while connected to VPN is encrypted. This includes web browsing, email, file transfer and any other online activities. This encryption makes it extremely difficult for hackers or malicious actors to intercept and read your data.

4.) Access to Restricted Content :- VPNs can be used to bypass geo-restrictions. For example, if a website or streaming service is only available in certain countries, you connect to VPN server in that country to access the content of if you were physically located there.

5.) Security and Privacy :- VPN provide an additional layer of privacy when using public Wi-Fi networks, such as those in coffee shops or anywhere. They protect your data from physically being seen here.

6.) Business Use :- VPNs are commonly used by businesses to provide secure remote access to their internal networks for employees working from home or while travelling. This ensures that sensitive company data remains secure even accessed from outside the office.

7.) Types Of VPNs :- There are several types of VPNs including remote access VPN, site-to-site VPNs, client-to-site VPNs. Each type has its own use case & implementation.

8.) VPN service Providers - To use VPN you typically need to subscribe VPN service provider. These providers maintain a network of VPN servers in various locations around the world.

RIP | OSPF

RIP stands for Routing Information Protocol and OSPF stands for Open Shortest Path First are two different routing protocols used in computer networks to determine how data packets should be forwarded from one network device to another. Both protocols are used to manage routing tables and facilitate efficient data transmission within a network, but they have significant differences in terms of operations and complexity.

RIP

① Distance-vector Protocol: RIP is a distance vector routing protocol. It determines the best path to destination by counting the number of hops (routers) between the source and destination. Each router periodically sends its routing table to its neighbours.

② Hop Count Metric: - RIP uses hop count as its metric for route selection. A route with fewer hops is considered better. However, RIP is limited to a maximum hop count of 15, which restricts its scalability.

⑥ Convergence Time :- RIP has relatively slow convergence times, when network topology changes occur, it can take a noticeable amount of time for routers to update their routing table & converge to a stable state.

* OSPF

⑦ Link-state protocol :- OSPF is a link state routing protocol. It builds and maintains a detailed database of the entire network topology, including information about the links & routers. This database is used to calculate the shortest path to reach any destination.

⑧ Cost metric :- OSPF uses a cost metric, typically based on bandwidth to determine the best path to destination. Lower costs represent better paths. This allows OSPF to take account factors like link bandwidth & latency when making routing decisions.

⑨ Convergence Time - OSPF generally has faster convergence times compared to RIP. When network changes occur, routers quickly make it suitable for large and dynamic networks.

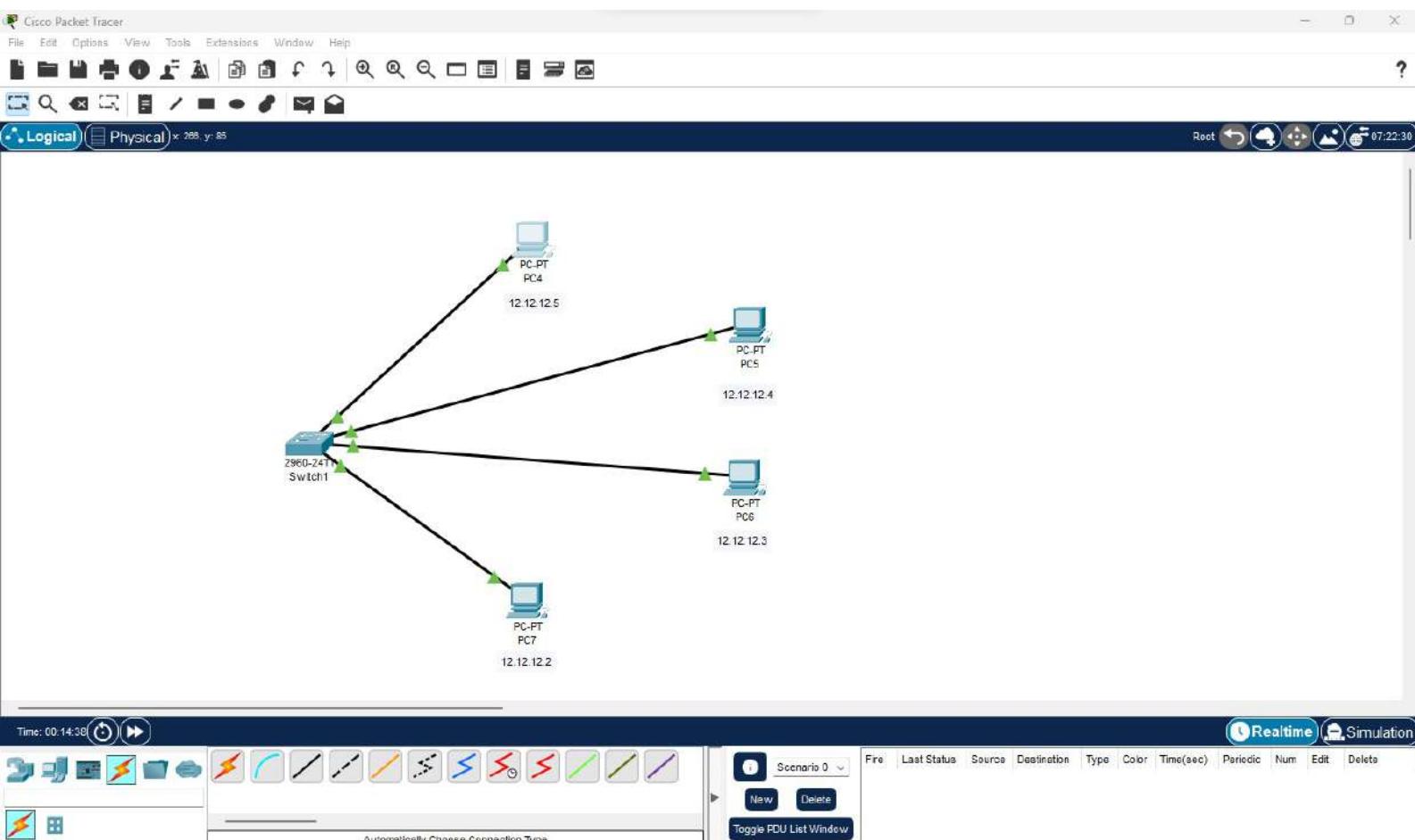
⑩ Commonly Used in Enterprise Network : OSPF is commonly used in enterprise networks and service provider environment for its flexibility and robustness.

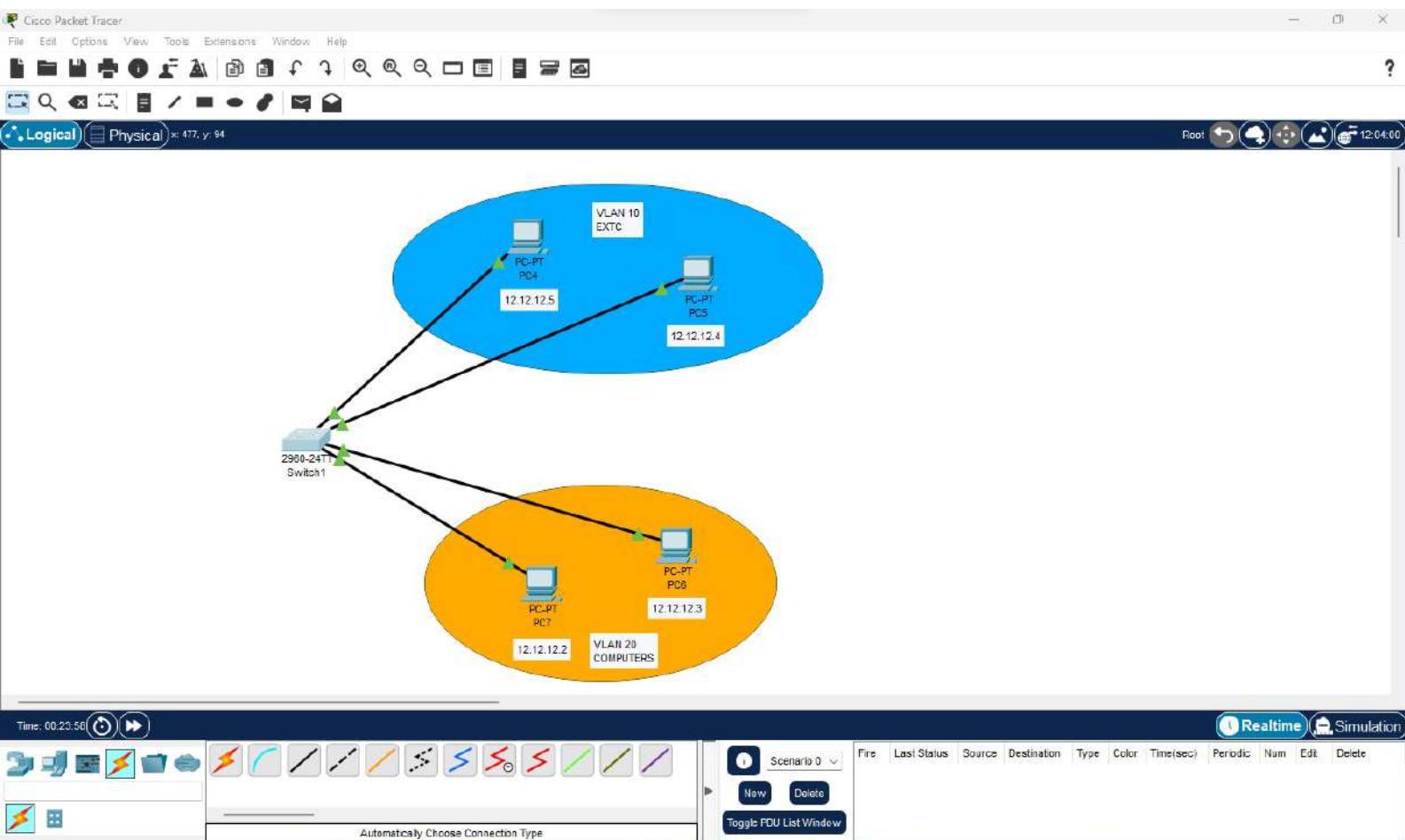
④ LSA (Link State Advertisement) :- Router exchange LSAs to share information about their link and network topology.

⑤ VLSM and CIDR :- OSPF is commonly used in enterprise networks and service provider environment for its flexibility and enhancing network scalability.

QUESTION

(Q)





Switch1

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

Add Remove

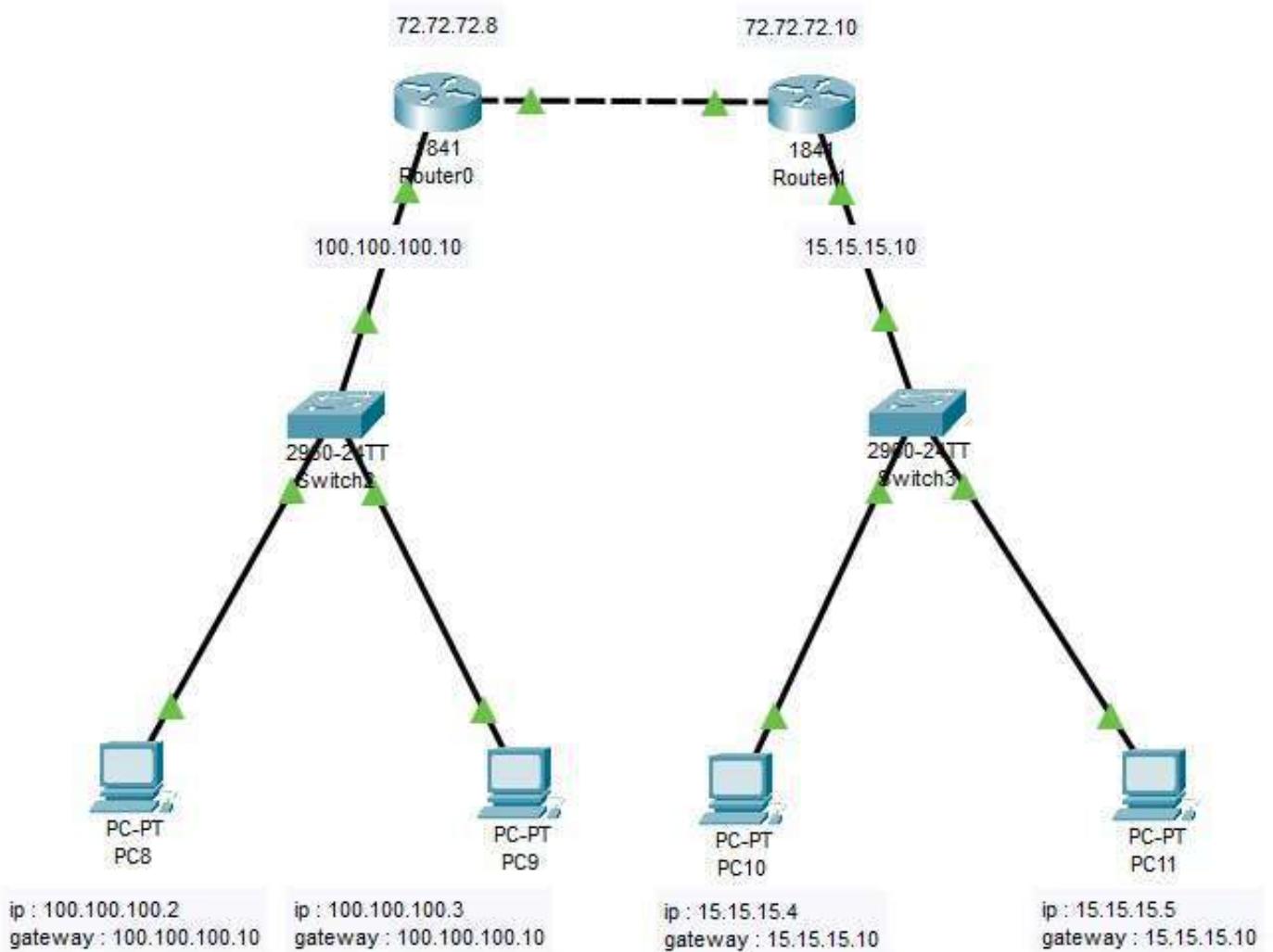
VLAN No	VLAN Name
1	default
10	EXTC
20	COMPUTERS
1002	fddi-default
1003	token-ring-default
1004	fddinet-default
1005	trnet-default

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:\>
```

Top

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:\>
```

Shrirish
C3 C32
2103164

EXPERIMENT - 7

CN - C32 - 2103164

Aim :- WAP to implement IPv4 addressing concept along with subnet masking.

Theory :-

IPv4 addressing is a fundamental concept in networking in that involves assigning unique numerical label to devices on a network. These addresses are used for identification and communication between devices in the IP network. IPv4 addresses are presented as a series of four decimal numbers, each ranging from 0 to 255, separated by periods. Each number is an octet representing 8 bits of 32 bit IPv4 address.

Here how IPv4 addressing works :-

32-bit Address :- IPv4 uses a 32-bit address space, allowing for approximately, 4.3 billion unique addresses.

Network and Host Portion :- An IPv4 address is divided into two parts : the network portion and the host portion. The boundary between two portions is determined by a subnet mask.

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

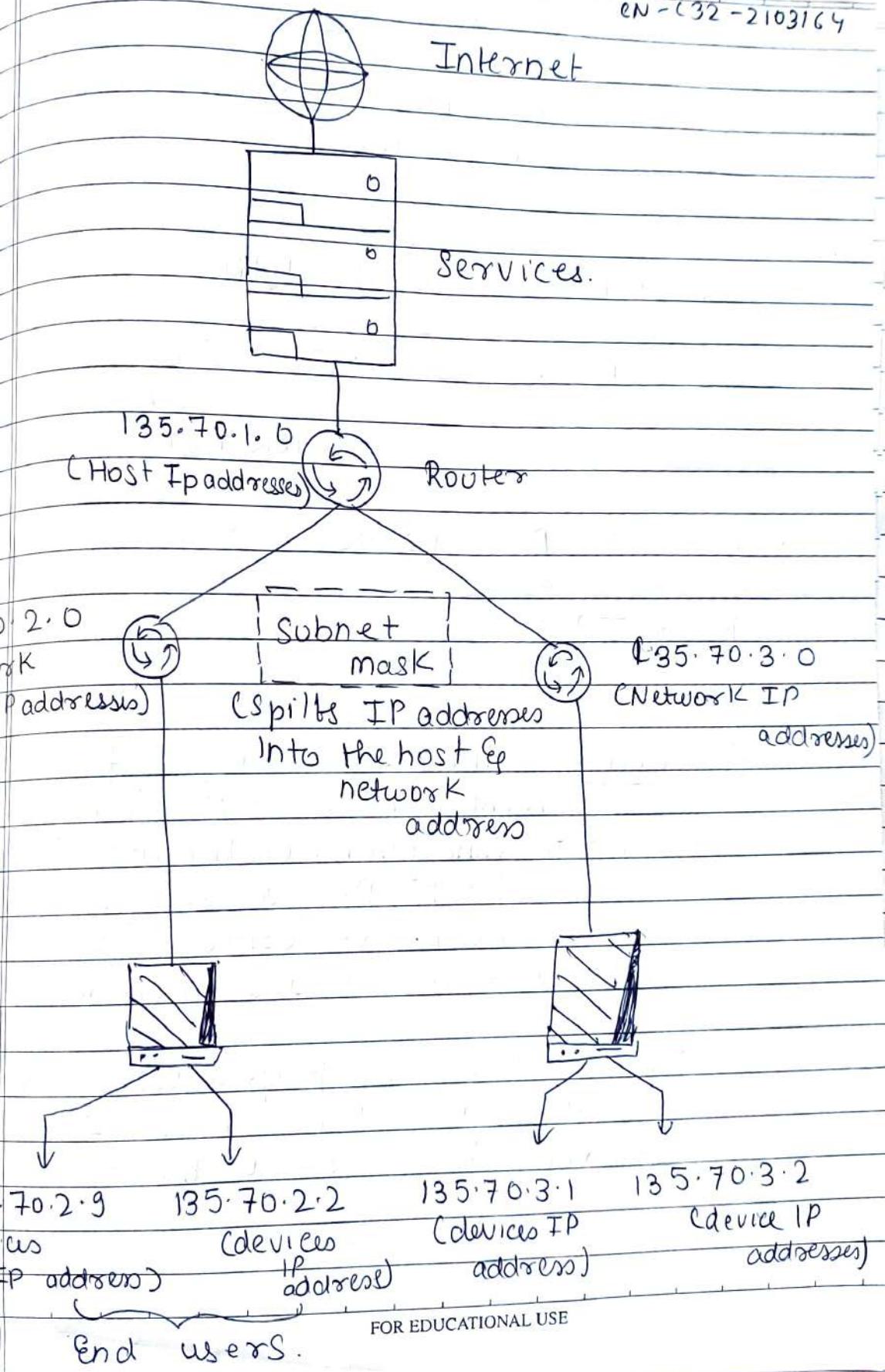
Network ID and Host ID :- The network portion of address identifies the network to which devices belongs, while the host portion is specific devices within that network. Devices within the same network share the same network ID.

Classes of IP addresses :- In past, IPv4 addresses were divided into five classes (A, B, C, D, E) based on their first octet range. However, classful addressing has largely been replaced by classless Inter-Domain Routing (CIDR) which allows for more flexible allocation of IP addresses space.

Private and Public Addresses :- IPv4 addresses are categorized into private and public address ranges. Private addresses are reserved for use within private network and are not routable on the public internet. Public addresses are globally unique and are used for devices accessible from the internet.

Subnet Masking

EN - C32 - 2103164



Subnet Masking

* Example of IPv4 addresses with their associated subnet mask

- IPv4 Address : 192.168.1.10
- Subnet Mask : 255.255.255.0 (also represented as 11111111.11111111.11111111.00000000) (CIDR notation.)
- Network ID : 192.168.1.0
- Host ID : 10.

* Conclusion :- IPv4 addressing is fundamental concept in networking & is essential for the functioning of Internet & local networks. Proper addresses allocation & subnetting are critical for efficient network, design and management.

This experiment helped us to shrink & built its logic which further enhanced the clarity of this concept

★ 30/03/2023

```

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 NO-8

CN-L32-2103164

Aim :- Use basic networking commands in Linux
(ping, traceroute, nslookup, netstat, ARP,
RARP, ip, config, dig, route).

Theory :-

1) Ping :-

The 'ping' command is used to test network connectivity between two devices. It sends ICMP Echo Request message to target host and wait for ICMP Echo Reply.

2) traceroute or tracert :-

'traceroute' is used to trace route taken by packet to destination. It shows the path packet follows through various routers and network to reach the target host.

Example :- 'traceroute google.com'

3) nslookup :-

'nslookup' is tool for querying DNS server to resolve domain name to corresponding IP address. It also provides information about DNS servers and domain records.

Example :- 'nslookup google.com'.

CN-132-210364

4.) 'netstat':- 'netstat' is command for displaying network statics, it can show information about active network connection, listening port, routing tables and various network interfaces.

Example:- 'netstat -tuln' to display listening TCP|UDP ports.

5.) 'arp':- The 'arp' command allows you to view and manipulate the Address Resolution protocol (ARP) cache. ARP maps IP address to MAC addresses on local network.

Example "arp -a" to list ARP cache entries.

6.) "rarp": Reverse Address Resolution is rarely used today and has been replaced by other method for IP address assignments.

7.) 'ip' :-

The 'ip' command is versatile tool for configuring and managing network interfaces, routing tables and other networking parameters. You can use to assign IP address, set up routes, and more.

Example :- 'ip address show' to display .

CN-C32-2103164

information about network interfaces.

'ifconfig' :- 'ifconfig' is used for configuring and displaying network interface while it is still present in many Linux distributions, it is considered somewhat deprecated and replaced by the 'ip' command.

Eg :- 'ifconfig eth0' to show information about the "eth0" interface.

'dig' :-

dig is powerful DNS query tool that allows you to retrieve detailed information from DNS servers. You can use it to query DNS records, find authoritative name servers and more.

Example 'dig google.com'

10.) 'route'

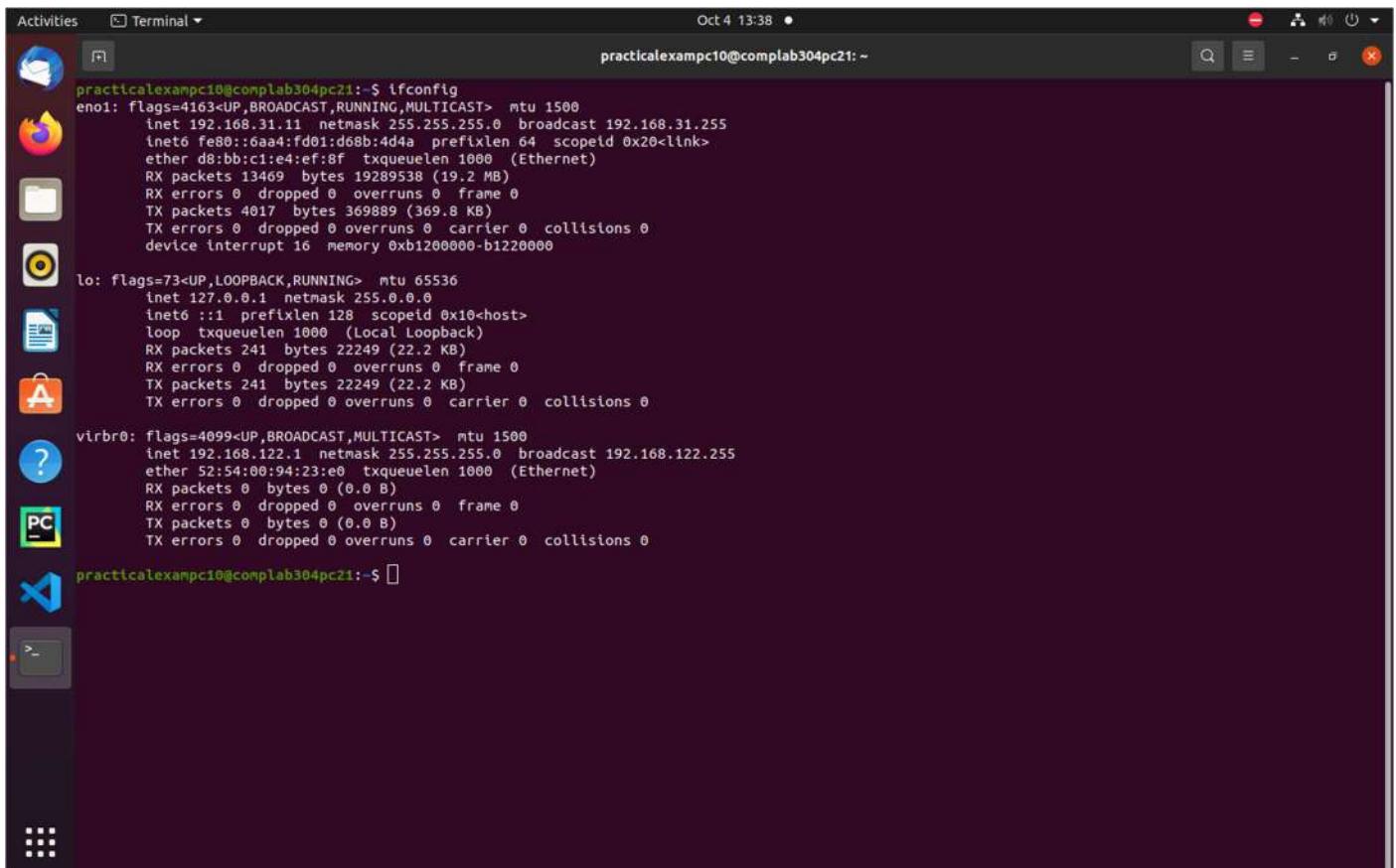
(Dated 20/10/2023)

The route command helps you view or manipulate the IP routing table. You can use it to add or remove routes, which determine how data packets are forwarded through a network.

Example :- 'route -n' to display the routing table in numeric format.

Conclusion :- These commands are essential for network diagnosis and configuration on Linux system, helping administrators & users troubleshoot & manage network related issues.

1. ifconfig



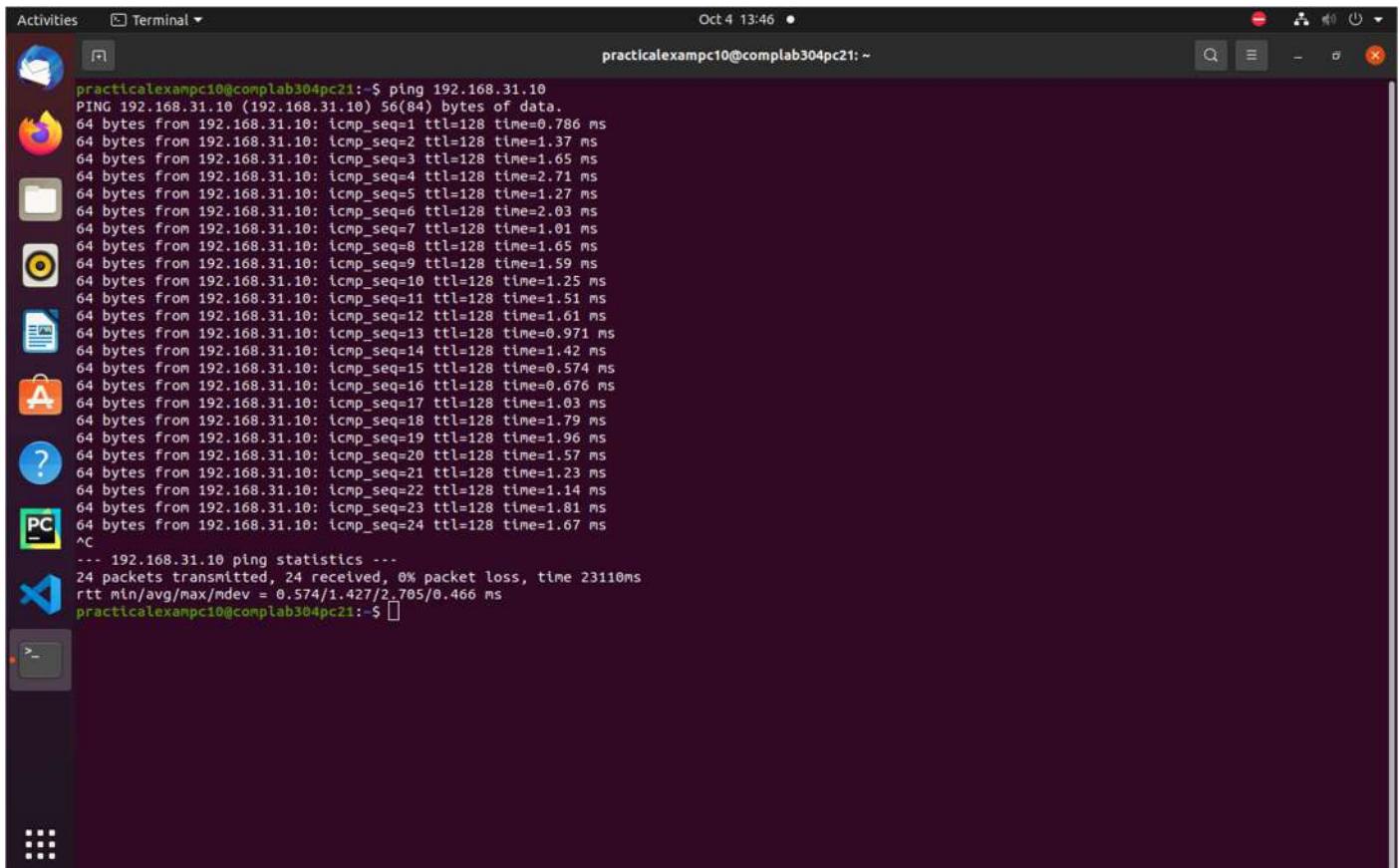
```
practicalexamplepc10@complab304pc21: ~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.31.11 netmask 255.255.255.0 broadcast 192.168.31.255
        inet6 fe80::6aa4:fd01:d68b:4d4a prefixlen 64 scopeid 0x20<link>
          ether d8:bb:c1:e4:ef:8f txqueuelen 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

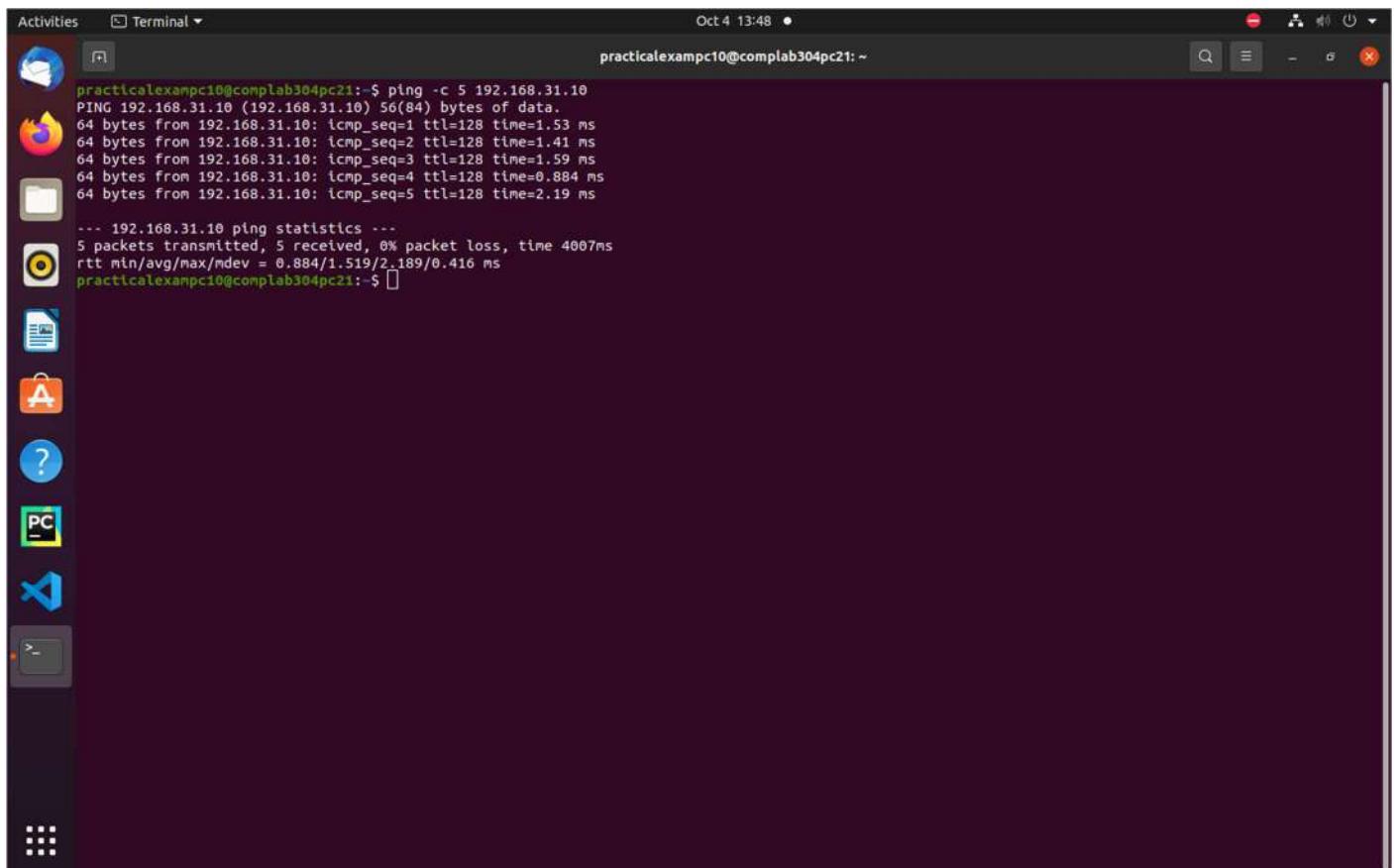
practicalexamplepc10@complab304pc21: ~$
```

2. ping without count



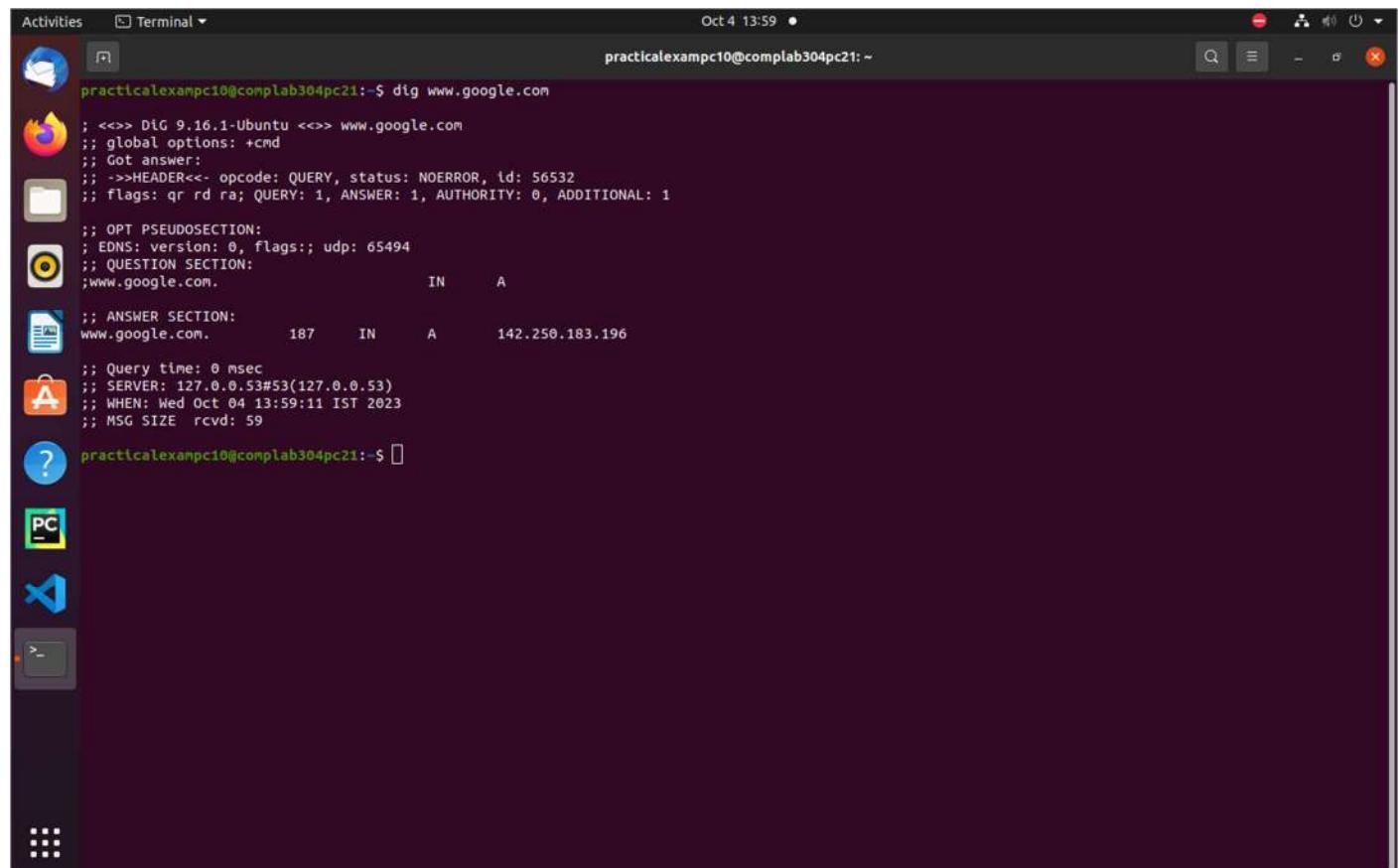
```
practicalexamplepc10@complab304pc21: ~$ ping 192.168.31.10
PING 192.168.31.10 (192.168.31.10) 56(84) bytes of data.
64 bytes from 192.168.31.10: icmp_seq=1 ttl=128 time=0.786 ms
64 bytes from 192.168.31.10: icmp_seq=2 ttl=128 time=1.37 ms
64 bytes from 192.168.31.10: icmp_seq=3 ttl=128 time=1.65 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



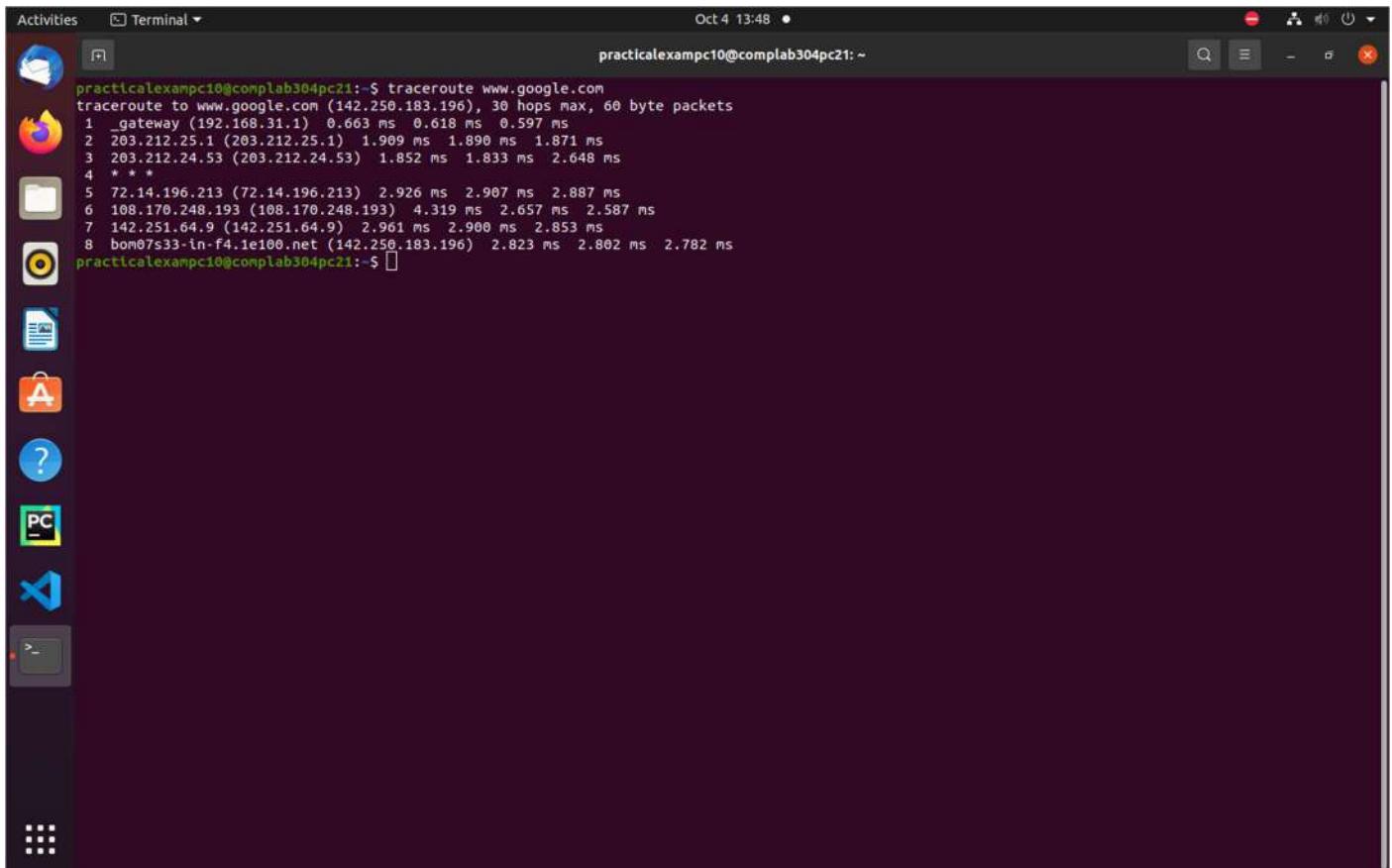
```
Activities Terminal Oct 4 13:48 ●
practicalexample10@complab304pc21:~$ ping -c 5 192.168.31.10
PING 192.168.31.10 (192.168.31.10) 56(84) bytes of data.
64 bytes from 192.168.31.10: icmp_seq=1 ttl=128 time=1.53 ms
64 bytes from 192.168.31.10: icmp_seq=2 ttl=128 time=1.41 ms
64 bytes from 192.168.31.10: 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
practicalexample10@complab304pc21:~$
```

4. dig



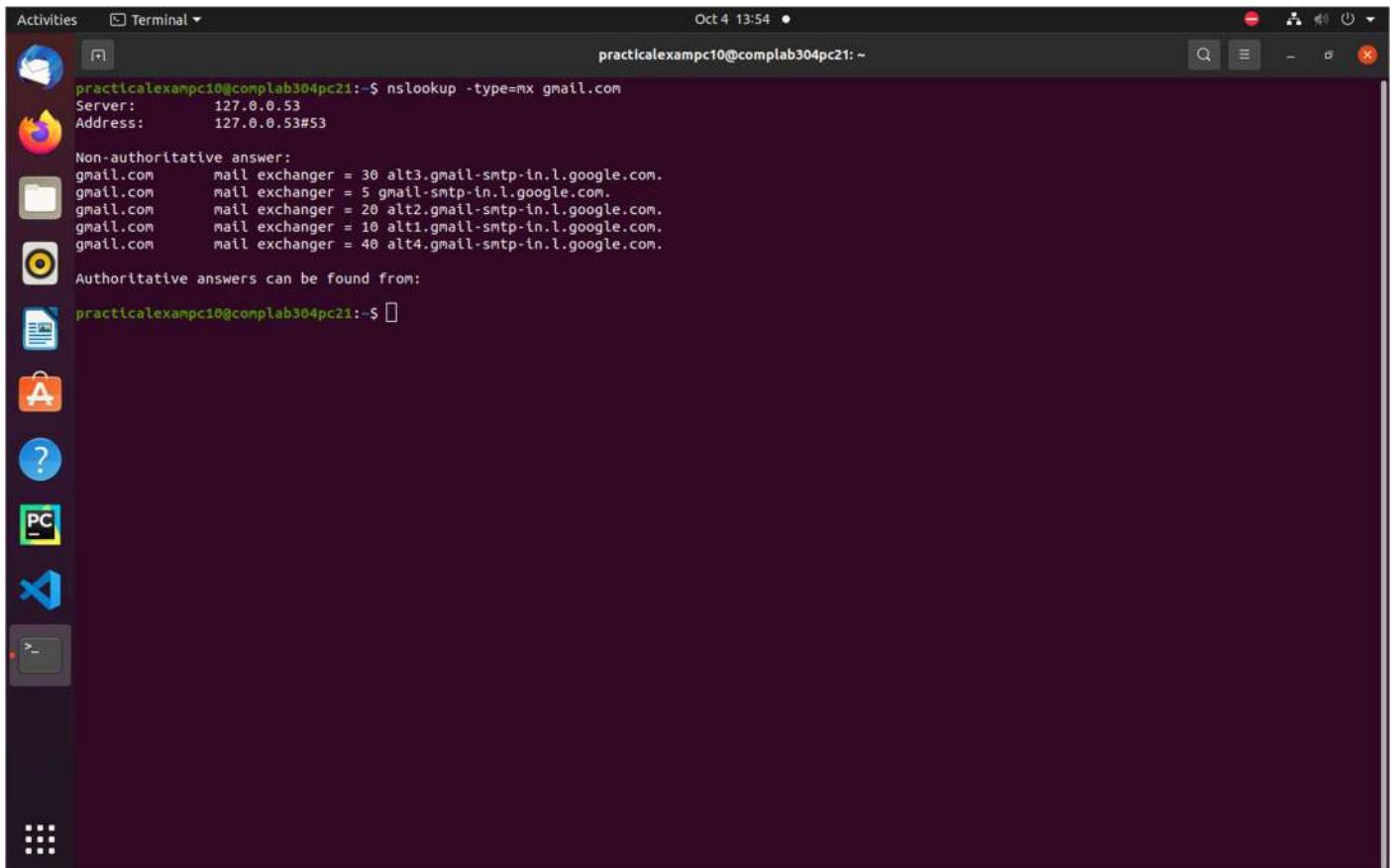
```
Activities Terminal Oct 4 13:59 ●
practicalexample10@complab304pc21:~$ dig www.google.com
; <>> DiG 9.16.1-Ubuntu <>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 56532
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; 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
practicalexample10@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

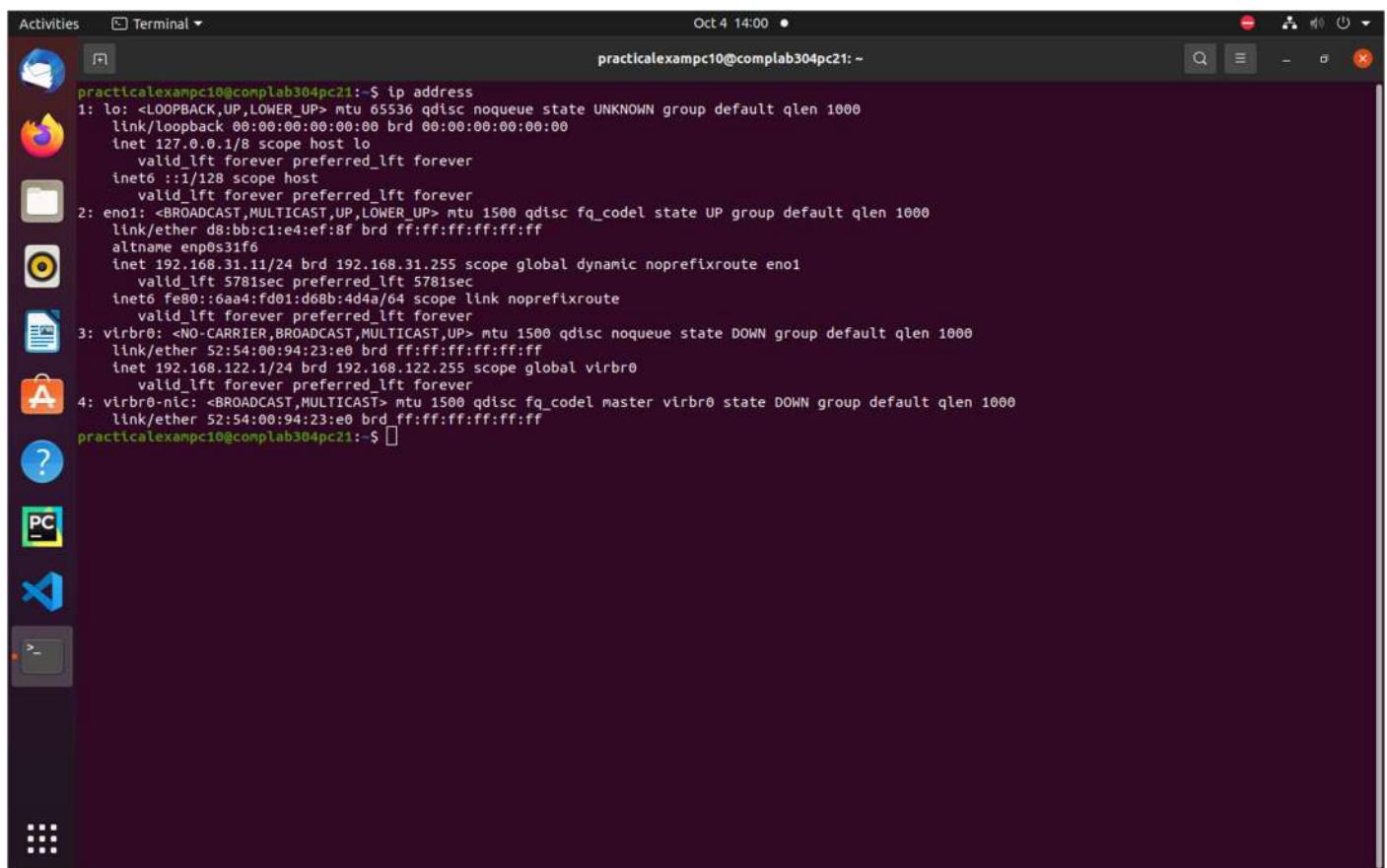


```
practicalexampc10@complab304pc21:~$ nslookup -type=mx gmail.com
Server:  127.0.0.53
Address: 127.0.0.53#53

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.

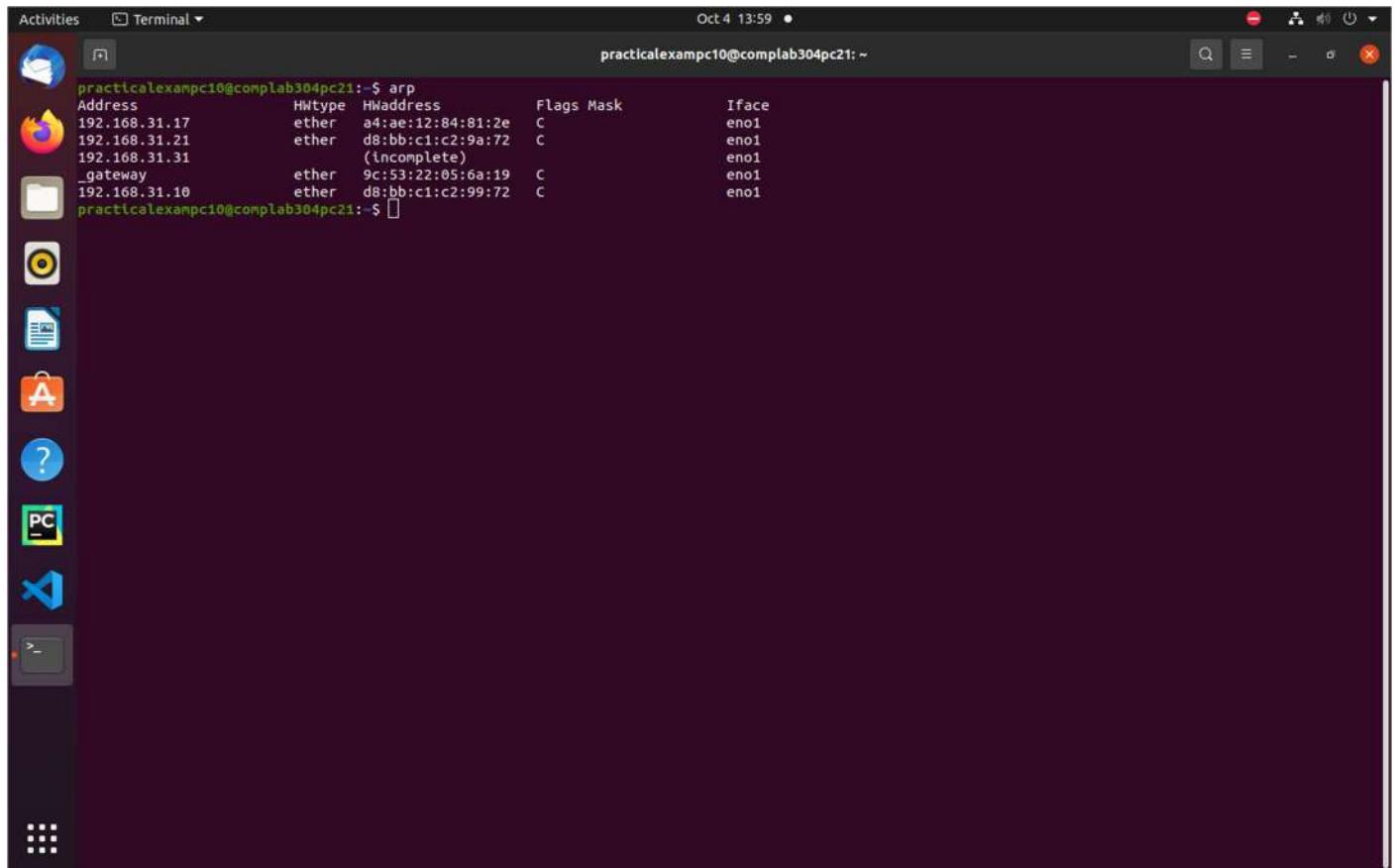
Authoritative answers can be found from:
practicalexampc10@complab304pc21:~$
```

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:ae: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:~$
```

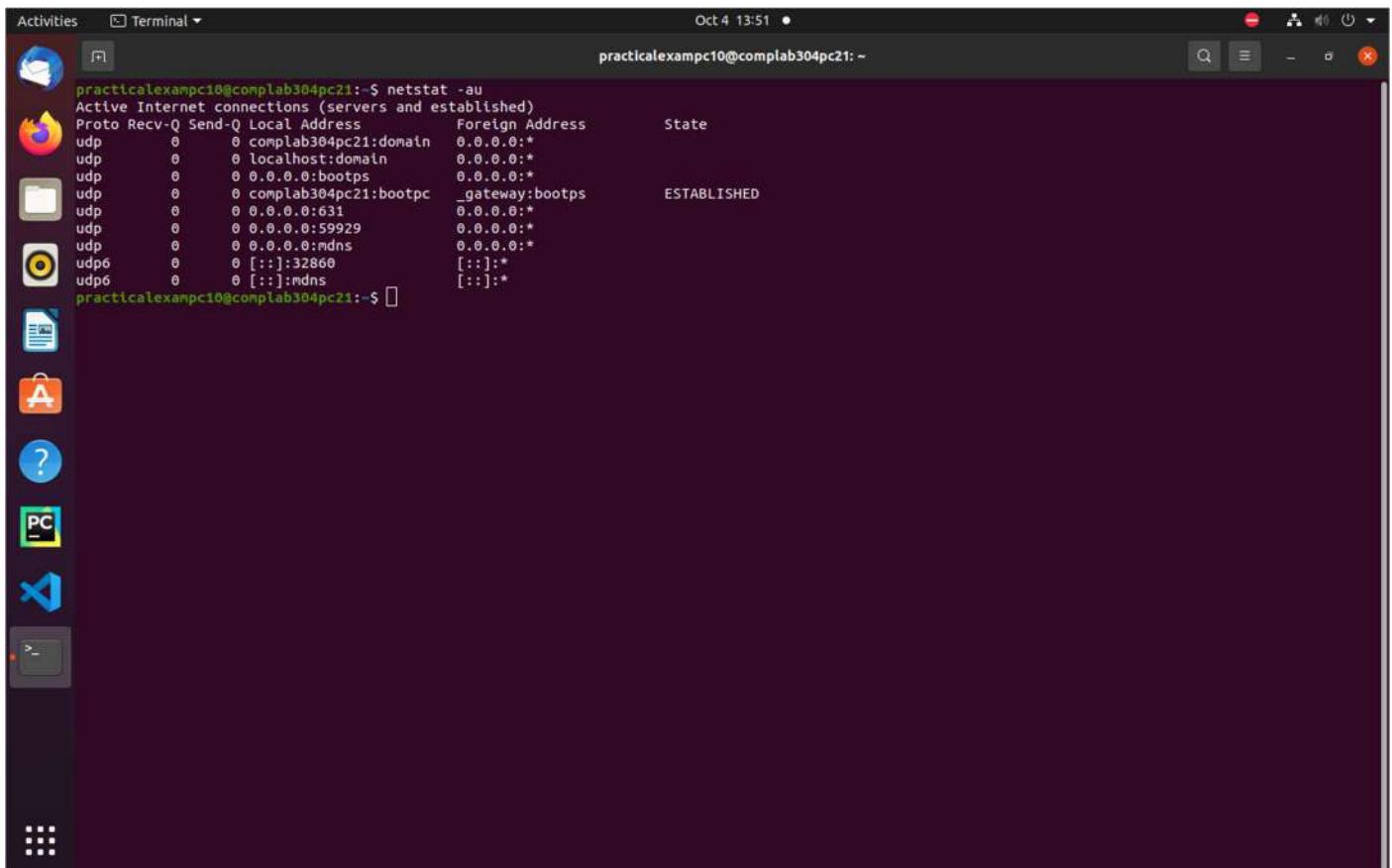
9. netstat

```
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/notify  
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/bus  
unix  3      [ ]  STREAM   CONNECTED  49756  /run/user/1002/bus  
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

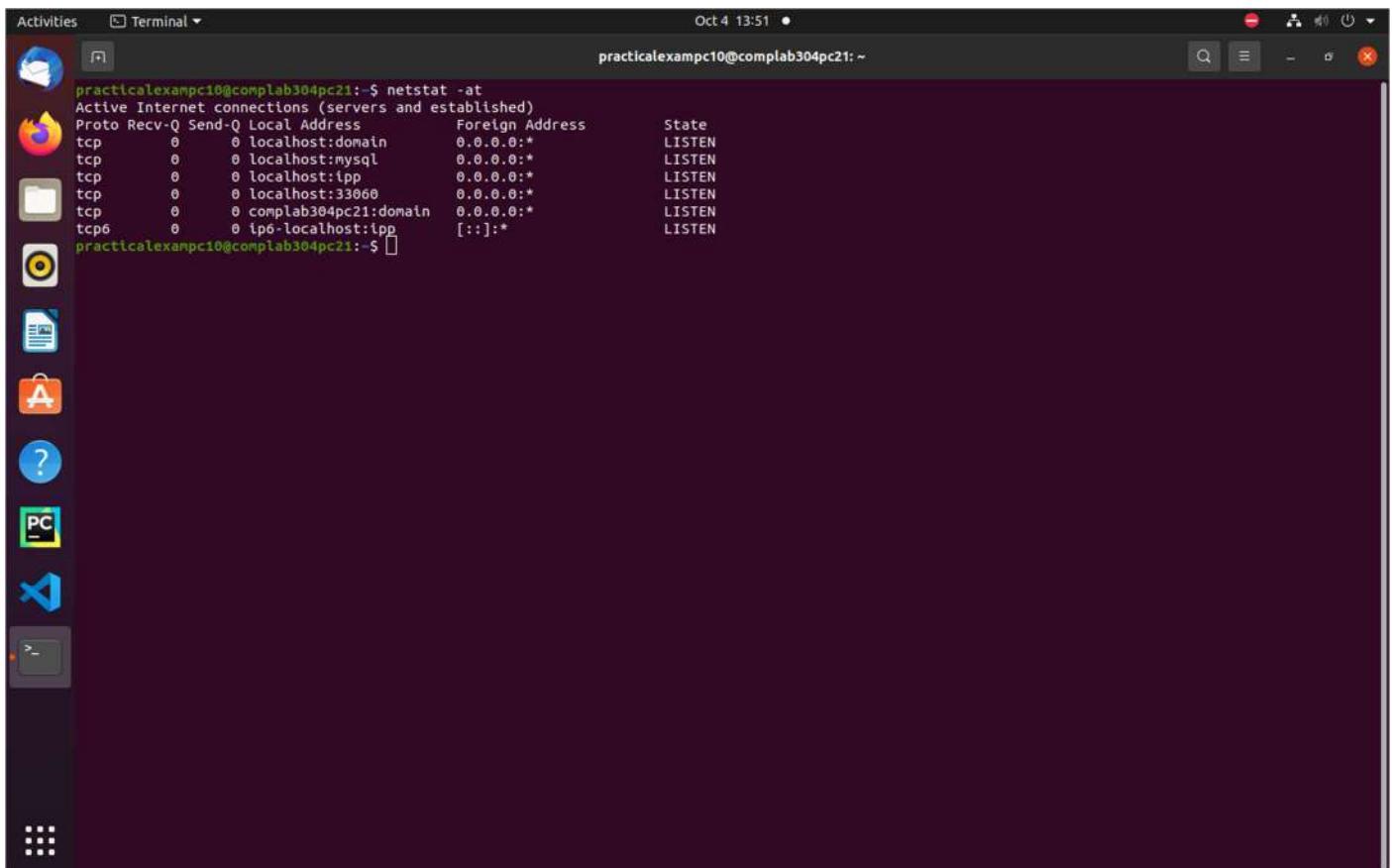
```
Activities Terminal Oct 4 13:50 • prakticalexampc10@complab304pc21:~  
practicallexampc10@complab304pc21:~$ netstat -a  
Active Internet connections (servers and established)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State  
tcp        0      0 localhost:domain        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  ::1:32860              ::*:*
udp6       0      0  ::1: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 NO-9

CN-C32-2103167

Aim:-

use wireshark to understand the operation of TCP/IP layers.

- Ethernet Layer : Frame header | Frame sized.
- Data Link Layer :- MAC address ; ARP (IP & MAC addressing)
- Network Layer : IP Packet
- Transport Layer : TCP Port, TCP, handshake segments etc.

Application Layer :- DHCP, FTP, HTTP. Header format.

Theory:-

Wireshark is a popular, open source network protocol analysis 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 wireshark, a network protocol analysis, to capture and analyze network traffic to understand the operation of various layer of TCP/IP protocol suite.

CN-182-2103164

1.) Ethernet layer.

This layer deals with frames, which are packets of data at the ethernet layer. Packets of data can be observed. One can observe the size of ethernet frames. One can observe the size of ethernet frames. One can observe the size of ethernet frames. Ethernet frames have headers containing information like source & destination MAC address.

2.) Data link layer

The data link layer deal with MAC address. Wireshark will show the MAC addresses of the addresses of the devices communicating in network. Users can capture ARP to packet to see how devices map IP addresses to MAC addresses.

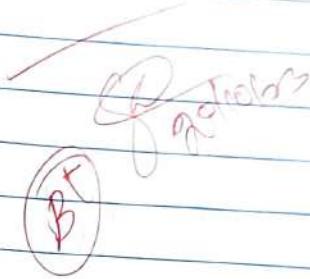
3.) Network layer

This layer involves a lot of processes. It involves IP packets. Wireshark displays IP headers with information like source & destination IP address. An ICMP packet can be captured to observe network troubleshooting, messages, like ping request & replies.



Transport layers :-

User can see which ports the used by application (eg:- web browsers, email clients) for communication. By capturing TCP traffic, you can see three-way handshake process, which is how two devices establish a connect.



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]
192.168.31.10	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	270	12.719991	192.168.31.10	TLSv1.2	251	Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
192.168.31.10	271	12.765667	204.79.197.283	TCP	54	49700 + 443 [ACK] Seq=518 Ack=5986 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	12.771626	192.168.31.10	DNS	148	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	12.775488	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=1318409 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=2820 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	12.926732	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=12167 Ack=2858 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 6
> 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: 0x8c1e0 [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)

Screenshot of Wireshark showing ARP traffic. The packet list shows several ARP requests and responses. The details pane shows the request for an IP address 192.168.31.1 from MAC address Dell_a5:05:b6. The bytes pane shows the raw hex and ASCII data of the ARP frame.

```

Frame 276786: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{704644D1-983F-44C8-8D76-142FFB438987}, 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
    
```

Source	No.	Time	Destination	Protocol	Length	Info
9c:53:22:05:6a:19	264869	45.548152	Broadcast	ARP	60	60 who has 192.168.31.37? Tell 192.168.31.1
Micro-St_c4:cf:8f	265469	45.650649	Broadcast	ARP	60	60 ARP Announcement for 192.168.31.11
Micro-St_c2:99:72	265644	45.678632	Micro-St_c2:9a:72	ARP	42	42 who has 192.168.31.21? Tell 192.168.31.10
Micro-St_c2:9a:72	268488	46.126583	Micro-St_c2:99:72	ARP	60	60 192.168.31.21 is at d8:bb:c1:c2:9a:72
9c:53:22:05:6a:19	268744	46.182324	Broadcast	ARP	60	60 who has 192.168.31.5? Tell 192.168.31.1
Dell_a5:05:b6	276786	47.419287	Broadcast	ARP	60	60 who has 192.168.31.17 Tell 192.168.31.37
Dell_a5:05:b6	277230	47.491897	Broadcast	ARP	60	60 who has 192.168.31.37? (ARP Probe)
Dell_a5:05:b6	278451	47.679292	Broadcast	ARP	60	60 who has 192.168.31.13 Tell 192.168.31.37
Dell_a5:05:b6	283428	48.491127	Broadcast	ARP	60	60 who has 192.168.31.37? (ARP Probe)
Dell_a5:05:b6	288875	49.498032	Broadcast	ARP	60	60 who has 192.168.31.37? (ARP Probe)

5. Dynamic Host Configuration Protocol (DHCP)

Screenshot of Wireshark showing DHCP traffic. The packet list shows a DHCP discover from 192.168.31.1, a DHCP offer from 192.168.31.1, and a DHCP request from 0.0.0.0. The details pane shows the configuration options sent by the server. The bytes pane shows the raw hex and ASCII data of the DHCP frames.

```

Frame 121144: 590 bytes on wire (4720 bits), 590 bytes captured (4720 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: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 192.168.31.1, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 67, Dst Port: 68
Dynamic Host Configuration Protocol (Offer)
    
```

Source	No.	Time	Destination	Protocol	Length	Info
0.0.0.0	113548	27.078915	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x128b3d80
0.0.0.0	116956	27.348086	255.255.255.255	DHCP	332	DHCP Discover - Transaction ID 0xb8ad7e84
192.168.31.1	121144	27.678937	255.255.255.255	DHCP	590	DHCP Offer - Transaction ID 0x128b3d80
0.0.0.0	121369	27.697072	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0xb8ad7e84
0.0.0.0	153957	31.866371	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x128b3d80
192.168.31.1	164172	31.885308	255.255.255.255	DHCP	590	DHCP Offer - Transaction ID 0x128b3d80

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 0xc7fa A ipv6-api.speedtest.net
203.212.24.46	4967	17.732393	192.168.31.18	DNS	169	Standard query response 0xc7fa 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: 0xc7fa
    Flags: 0x8100 Standard query response, No error
    Questions: 1
    Answer RRs: 0
    Authority RRs: 1
    Additional RRs: 0
    Queries: 1
    Authoritative nameservers
        [Request_Ini_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	88.567233	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 Resassembled 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: d3e06f18-5db6-4314-a0cc-f06f06eb92d\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/cbedbbhbpmojnkanicioggmelmoomoc_20230916.567854667.14_all_ENU580000_1r74]
        File Data: 354659 bytes
        <a href="javascript:;">...
```

8. Transmission Control Protocol (TCP)

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

Selected packet details:

- No. 33813 Time 12.098262 Source 192.168.31.13 Destination 8.241.131.254 Protocol TCP Length 66 Info: 66 57571 → 80 [ACK] Seq=2671 Ack=7036585 Win=481 Len=0 TSval=506375703 TSecr=3094558364
- No. 33814 Time 12.098694 Source 8.241.131.254 Destination 192.168.31.13 Protocol TCP Length 1494 Info: 1494 80 → 57571 [PSH, ACK] Seq=7036585 Ack=2671 Win=11 Len=1428 TSval=3094558364 TSecr=506375646 [TCP segment of a reassembled segment]
- No. 33815 Time 12.099902 Source 8.241.131.254 Destination 192.168.31.13 Protocol TCP Length 1494 Info: 1494 80 → 57571 [ACK] Seq=2671 Win=11 Len=1428 TSval=3094558365 TSecr=506375647 [TCP segment of a reassembled segment]
- No. 33816 Time 12.099912 Source 192.168.31.13 Destination 8.241.131.254 Protocol TCP Length 66 Info: 66 57571 → 80 [ACK] Seq=2671 Ack=7039441 Win=481 Len=0 TSval=506375705 TSecr=3094558364 [TCP segment of a reassembled segment]
- No. 33817 Time 12.100006 Source 8.241.131.254 Destination 192.168.31.13 Protocol TCP Length 1494 Info: 1494 80 → 57571 [ACK] Seq=7039441 Ack=2671 Win=11 Len=1428 TSval=3094558365 TSecr=506375647 [TCP segment of a reassembled segment]
- No. 33818 Time 12.100939 Source 8.241.131.254 Destination 192.168.31.13 Protocol TCP Length 1494 Info: 1494 80 → 57571 [ACK] Seq=70408884 Ack=2671 Win=11 Len=1428 TSval=3094558365 TSecr=506375647 [TCP segment of a reassembled segment]
- No. 33819 Time 12.100949 Source 192.168.31.13 Destination 8.241.131.254 Protocol TCP Length 66 Info: 66 57571 → 80 [ACK] Seq=2671 Ack=7041297 Win=481 Len=0 TSval=506375706 TSecr=3094558365 [TCP segment of a reassembled segment]
- No. 33820 Time 12.101043 Source 8.241.131.254 Destination 192.168.31.13 Protocol TCP Length 1494 Info: 1494 80 → 57571 [ACK] Seq=7042297 Ack=2671 Win=11 Len=1428 TSval=3094558365 TSecr=506375647 [TCP segment of a reassembled segment]
- No. 33821 Time 12.102036 Source 8.241.131.254 Destination 192.168.31.13 Protocol TCP Length 1494 Info: 1494 80 → 57571 [ACK] Seq=7043725 Ack=2671 Win=11 Len=1428 TSval=3094558366 TSecr=506375647 [TCP segment of a reassembled segment]
- No. 33822 Time 12.102046 Source 192.168.31.13 Destination 8.241.131.254 Protocol TCP Length 66 Info: 66 57571 → 80 [ACK] Seq=2671 Ack=7045153 Win=481 Len=0 TSval=506375707 TSecr=3094558366 [TCP segment of a reassembled segment]
- No. 33823 Time 12.102107 Source 8.241.131.254 Destination 192.168.31.13 Protocol TCP Length 1494 Info: 1494 80 → 57571 [ACK] Seq=7045153 Ack=2671 Win=11 Len=1428 TSval=3094558366 TSecr=506375647 [TCP segment of a reassembled segment]

Selected packet bytes:

```

> Frame 33816: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{A8787A28-2B44-4DA3-B2C3-A3348E8A215}, id 0
> Ethernet II, Src: Micro-St_C2:99:e5 (d8:b1:c1:c2:99:e5), Dst: 0c:53:22:05:6a:19 (0c:53:22:05:6a:19)
> Internet Protocol Version 4, Src: 192.168.31.13, Dst: 8.241.131.254
> Transmission Control Protocol, Src Port: 57571, Dst Port: 80, Seq: 2671, Ack: 7039441, Len: 0
  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): 1170984078
  [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)]
  Checksum: 0xccc [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".

Selected packet details:

- No. 47 Time 0.085612 Source 192.168.31.13 Destination 203.212.24.46 Protocol DNS Length 91 Info: 91 Standard query 0x1767 A teams.events.data.microsoft.com
- No. 136 Time 0.154171 Source 203.212.24.46 Destination 192.168.31.13 Protocol DNS Length 213 Info: 213 Standard query response 0x1767 A teams.events.data.microsoft.com CNAME teams-events-data.trafficmanager.net CNAME dns.
- No. 481 Time 0.849814 Source 192.168.31.31 Destination 224.0.0.251 Protocol DNS Length 180 Info: 180 Standard query response 0x0000 PTR, cache flush MUL105.local A, cache flush 192.168.31.31 NSEC, cache flush 31.31.168.
- No. 484 Time 0.868291 Source 192.168.31.13 Destination 203.212.24.46 Protocol DNS Length 86 Info: 86 Standard query 0x009c A config.teams.microsoft.com
- No. 495 Time 0.066333 Source 192.168.31.13 Destination 203.212.24.46 Protocol DNS Length 86 Info: 86 Standard query 0xd244 HTTPS config.teams.microsoft.com
- No. 553 Time 0.946408 Source 192.168.31.13 Destination 239.255.255.250 Protocol SSDP Length 217 Info: 217 M-SEARCH * HTTP/1.1
- No. 559 Time 0.959953 Source 203.212.24.46 Destination 192.168.31.13 Protocol DNS Length 244 Info: 244 Standard query response 0x609c A config.teams.microsoft.com CNAME config.teams.trafficmanager.net CNAME s-0005-teams..
- No. 567 Time 0.960987 Source 203.212.24.46 Destination 192.168.31.13 Protocol DNS Length 275 Info: 275 Standard query response 0xd244 HTTPS config.teams.microsoft.com CNAME config.teams.trafficmanager.net CNAME s-0005-te..
- No. 628 Time 1.049781 Source 192.168.31.13 Destination 203.212.24.46 Protocol DNS Length 76 Info: 76 Standard query 0x0568 A pool.minexmr.com
- No. 700 Time 1.078908 Source 203.212.24.46 Destination 192.168.31.13 Protocol DNS Length 136 Info: 136 Standard query response 0x0568 No such name A pool.minexmr.com SOA frank.ns.cloudflare.com
- No. 2618 Time 3.126206 Source 192.168.31.13 Destination 239.255.255.250 Protocol SSDP Length 217 Info: 217 M-SEARCH * HTTP/1.1
- No. 2621 Time 3.153589 Source 192.168.31.13 Destination 203.212.24.46 Protocol DNS Length 76 Info: 76 Standard query 0x6e49 A edge.microsoft.com
- No. 2622 Time 3.153753 Source 192.168.31.13 Destination 203.212.24.46 Protocol DNS Length 78 Info: 78 Standard query 0xd98e HTTPS edge.microsoft.com
- No. 2623 Time 3.180996 Source 203.212.24.46 Destination 192.168.31.13 Protocol DNS Length 182 Info: 182 Standard query response 0xd98e HTTPS edge.microsoft.com CNAME edge.microsoft.com.dual-a-0036.a-msedge.net SOA ns1.a-m..
- No. 2624 Time 3.171867 Source 203.212.24.46 Destination 192.168.31.13 Protocol DNS Length 181 Info: 181 Standard query response 0xc4e9 A edge.microsoft.com CNAME edge.microsoft.com.dual-a-0036.a-msedge.net CNAME dual-a-00..
- No. 2649 Time 3.230267 Source 0.0.0.0 Destination 255.255.255.255 Protocol DHCP Length 346 Info: 346 DHCP Request - Transaction ID 0x9e1d65f1
- No. 2654 Time 3.3212473 Source 0.0.0.0 Destination 255.255.255.255 Protocol DHCP Length 346 Info: 346 DHCP Request - Transaction ID 0x1cd2d0ca
- No. 2661 Time 3.367428 Source 192.168.31.1 Destination 255.255.255.255 Protocol DHCP Length 598 Info: 598 DHCP ACK - Transaction ID 0x9e1d65f1
- No. 2663 Time 3.381341 Source 192.168.31.1 Destination 255.255.255.255 Protocol DHCP Length 598 Info: 598 DHCP ACK - Transaction ID 0x1cd2d0ca

Selected packet bytes:

```

> Frame 405: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface \Device\NPF_{A8787A28-2B44-4DA3-B2C3-A3348E8A215}, id 0
> Ethernet II, Src: Micro-St_C2:99:e5 (d8:b1:c1:c2:99:e5), Dst: 0c:53:22:05:6a:19 (0c: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
  Source Port: 58779
  Destination Port: 53
  Length: 86
  Checksum: 0xc3fd [unverified]
  [Checksum Status: Unverified]
  [Stream index: 3]
  > [Timestamps]
  > [Domain Name System (query)]
  > Domain Name System (query)

```



Shivash

C3 C32

2103164

Experiment - 10.

CN - C32 - 2103164

Aim :- Socket programming using TCP or UDP.

Theory :-

Socket programming is a crucial aspect of network communication, enabling data exchange between computers over a network. TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are two commonly used transport layer protocols for socket programming.

I- TCP

TCP ensures reliable ordered and error checked delivery of data between two devices. It establishes a connection before data exchange.

Data is sent as a stream of bytes and TCP manages the packetization and reassembly of data.

TCP uses acknowledgements to confirm the receipt of data packets and retransmits lost packet if necessary.

TCP is suitable for application where data integrity and order are crucial such as web browsing, email and file transfer.

2.) ~~of~~ UDP

UDP don't establish a connection & doesn't guarantee delivery or order of data packets. Each UDP packet is treated as an independent unit of data.

UDP doesn't use acknowledgement, so it's faster but can result in last or out-of-order packets.

Socket connection consist of client side connection and server side connection.

Client - Side programming :-

To connect to another machine we need a socket connection. A socket connection means two machines have information about each other network location and TCP Port.

The ~~java.~~ net Socket class represents a socket. To open a socket,

`Socket socket = new Socket("127.0.0.1")`

The first argument - IP address of 5000 local host where code will run on the single stand alone machine. The second argument TCP port (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535).

To communicate over a socket connection, streams

CN - C32 - 2103164

used to both input and output the data.

Conclusion :- learnt about socket programming
and how to implement it in java using
TCP or UDP.

FRIDAY

(BT)

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:

```
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>

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>
```

CN Assignment-1

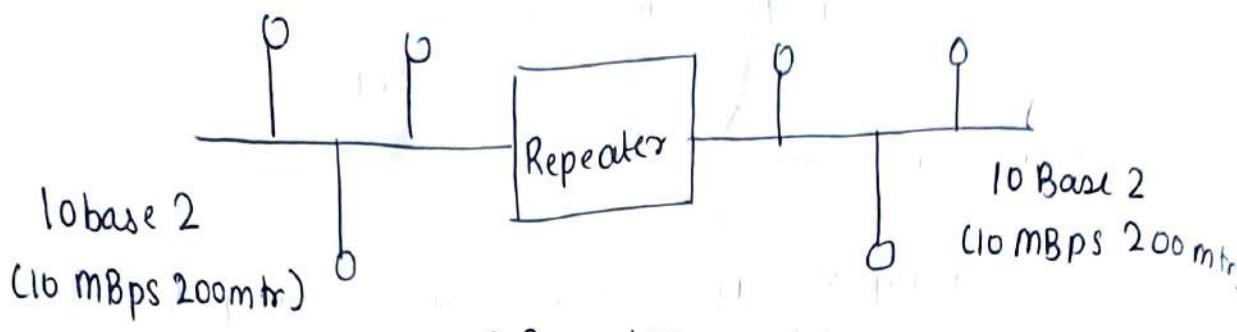
Write a short note on:-

CN-C32-2103164

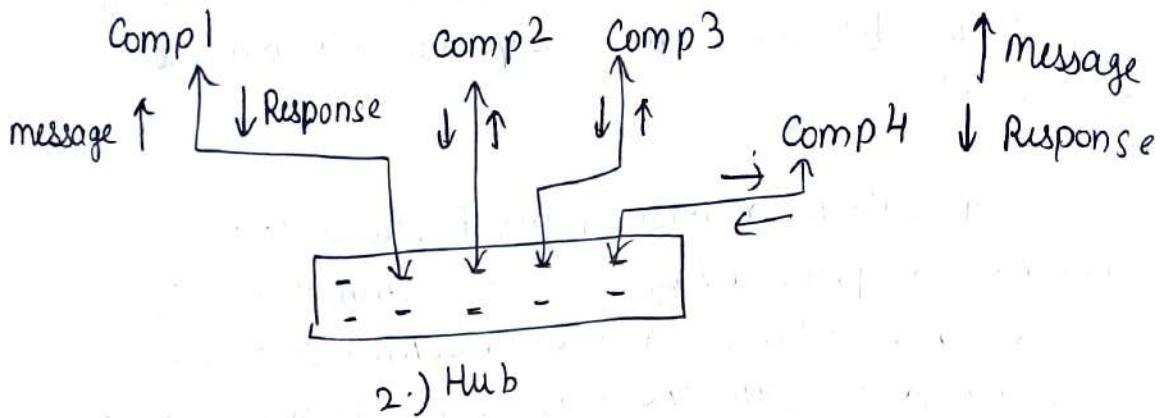
1) Repeater :- A repeater is a device used in computer networks to amplify and retransmit signals, extending the network's reach. It operates the physical layer, transparently regenerating signals without examining data content. However, advancement in networking technology have led to the decreased use of traditional repeaters in favour of more sophisticated devices like switches.

2) Hub :- A hub is a basic network device that operates at physical layer, used to connect multiple devices in a local area network (LAN). It receives incoming data from one device and broadcast it all connected devices. Hubs create single collision domain, leading to data collisions and reduced network efficiency. Due to their limitation and lack of intelligence, hubs have become outdated in modern networks and have been replaced by switches, which provide better performance and segmentation of collision domains for more efficient data transmission.

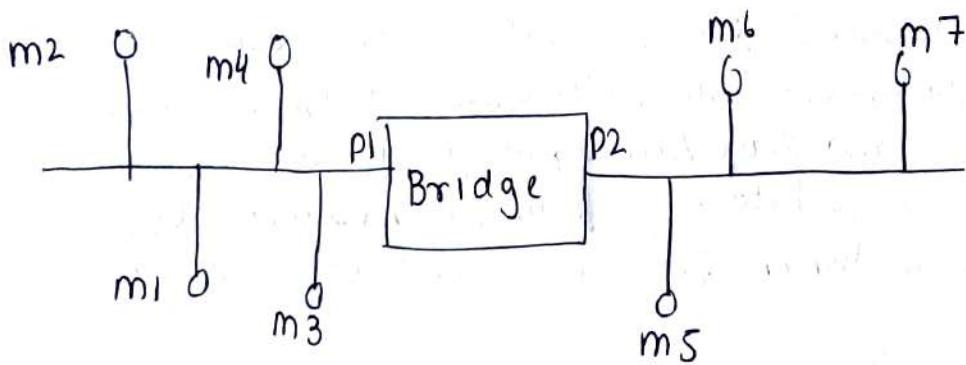
3) Bridges :- Bridges are network devices that operate at data link layer (Layer 2) of the OSI model, used to interconnect two or more LAN segments. They examine the MAC addresses of incoming data frames and maintain a table of MAC addresses and their associated segments. When a frame destination



1.) Repeater



2.) Hub

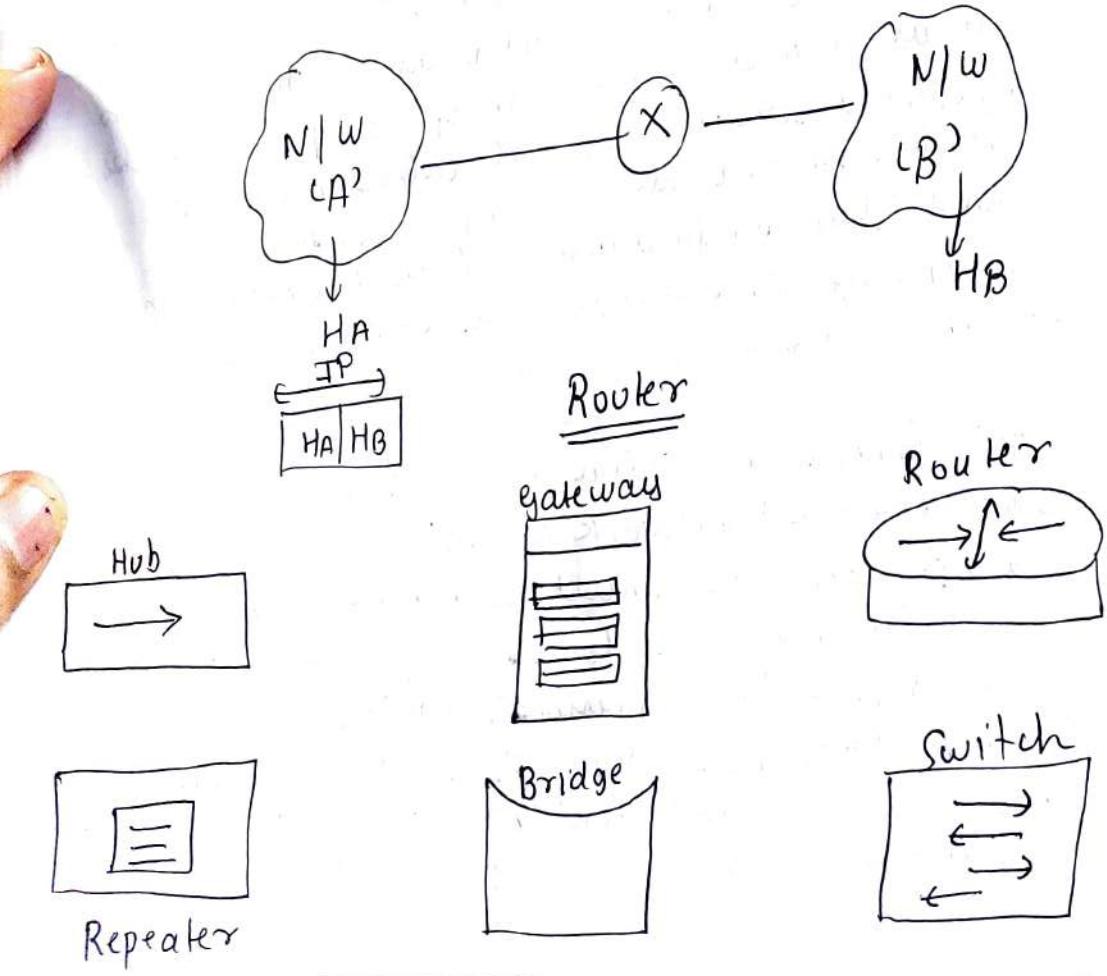
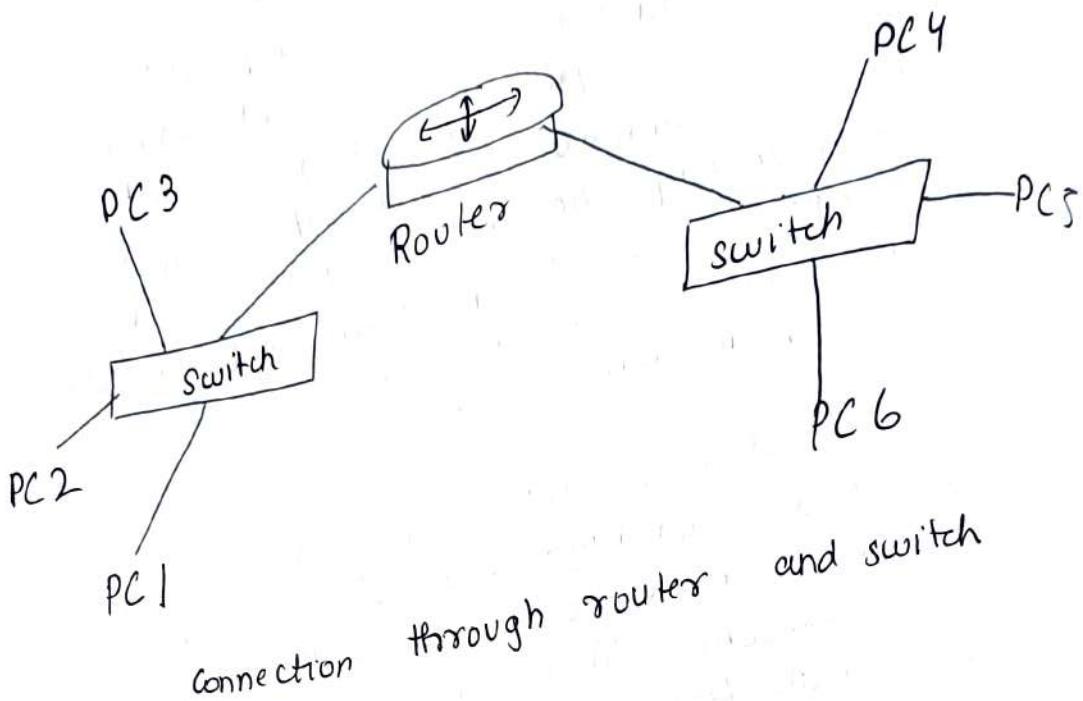


3.) Bridge

is on same segment as the source, the bridge filters the frame. If the destination is on different segment the bridge forwards the frame only to the relevant segment, reducing unnecessary network traffic. Bridges help in improving network traffic. Bridges help in improving network performance by dividing LANs into smaller collision domains and are predecessors to modern switches.

- 4) Switches : - Switches are integral network devices that operate at data link layer (Layer 2) of the OSI model, used to connect multiple devices within a local area network (LAN). They examine incoming data frames, make forwarding decisions based on the destination MAC address, and maintain a MAC address table to associate addresses with specific port. Unlike hubs, switches create separate collision domain for each connected device, improving network efficiency. They are fundamental to modern networking providing efficient and reliable communication between devices in a LAN.
- 5) Router : - A Router is a vital network device that operates at network layer (Layer 3) of the OSI model. It connects multiple networks and make intelligent data forwarding decisions based on destination IP addresses. They maintain routing table to determine best path for data transmission. By doing so, router enable efficient data exchange, ensure proper network segmentation, provide security by acting as

CN-C32-2103164



as a gateway between networks. Router plays a crucial role in directing data traffic and enabling communication across complex & interconnected networks.

Gateway :- A gateway is network devices that act as an interface between different networks, protocols, or communication technologies. They play a vital role in connecting LANs to the internet, making access to external networks. Gateways can be hardware or software based and are essential for establishing communication between networks with distinct characteristics. In essence, gateways serve as bridges, allowing data to flow freely between different network and ensuring interoperability in complex networking environments.

7) Modem :- A modem, short for modulator-demodulator, is a network device used to convert digital data from computers or digital devices into analog signal suitable for transmission over analog communication channels. It also performs reverse process i.e. converting incoming analog signal back to digital for receiving device. Modems are commonly used to provide internet access via dial-up connections and are essential for connecting to the internet over telephone lines. With the rise of broadband and digital communication technologies, traditional dial-up modems have become less common, but their role in early networking history was crucial for establishing internet connectivity.

Shirish
C32

CN Assignment No - 2



2103164
Shirish
C32

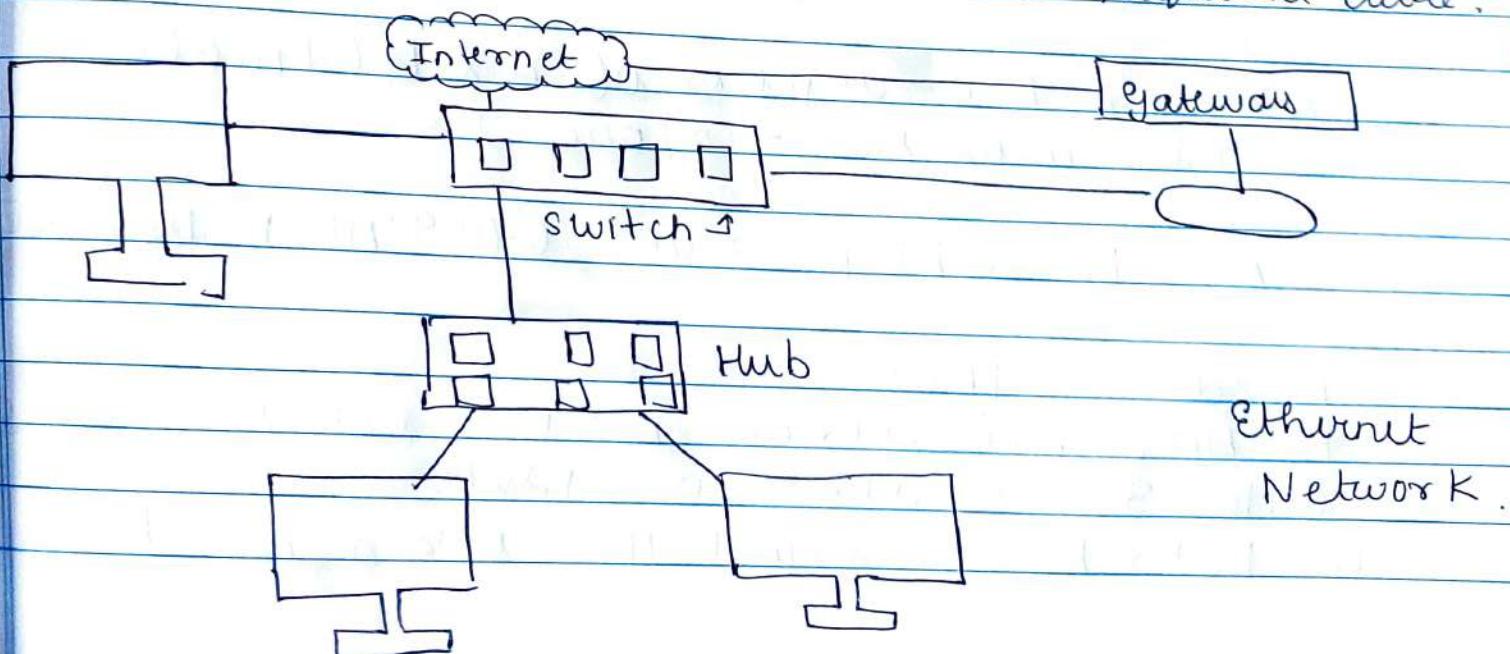
CN - C32 - 2103164

write short note on

Ethernet

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

The wireless networks replaced Ethernet in many areas, however Ethernet is still more common on bar wired networking. Wi-Fi reduces the need for cabling as it allows users to connect smartphones & laptops to a network without the required cable.



CN-C32-2103164

Advantages of Ethernet :-

- i) It is costlier, 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) Can't determine which node is faulty.

② 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 use Hexadecimal format separated by colonc :-)

Components of IPv6 address.

- i) 8-groups, each group represent 2 byte (16 bits).

ii) Each hex-digit is of 4 bits (1 nibble)

iii) Delimiter used - colonc :-)

ABCD : EFD1 : 2345 : 6789 : ABCD : D201 • 548

Need of IPv6

- i) An IPv6 address is 128 bits long compared with 32 bit address of IPv4.
- ii) Better header format, IPv6 uses a new header

format in which options are separated from the base header & inserted when needed between the base header & the upper layer data. This simplifies i.e. speed up the routing process.

(iii) IPv6 has new options to allow for functions not required by new technologies.

(iv) IPv6 allows extension of the protocol it

Advantages of IPv6

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

SSH

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

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

CN-C3T2103164
 Data received {username: "abc"
 password: "qabc"}

Before SSH

Data of Username: "one"
 sent Password: "abcd"

client

Router

man in
the middle

Data sniffed

{username: "one"
 password: "abcd" }

server

After SSH

Data {username: "clone"
 sent password: "abcd" }

client

Router

Data
sent

man in
the middle

Data :
snipped

Data
received

Advantages of SSH

- ⑥ prevent malicious activities like . DNS spoofing.
- Data manipulation, eavesdropping or sniffing of transmitted data; IP address routing.

(iii) SSH enables port forwarding

Explain purpose of following protocols with header format.

i) ARP ii) ICMP iii) DNS

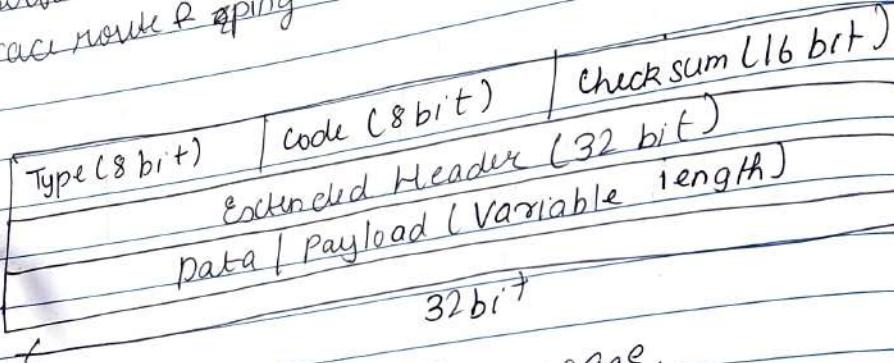
- i) ARP: The Address Resolution Protocol (ARP) is a fundamental networking protocol that plays a pivotal role in local network communication by mapping known IP addresses to their corresponding physical MAC addresses. It enables devices on a network to discover & associate MAC addresses with IP addresses.

32 bits	
Hardware Type	Protocol Type
Hardware length	Protocol length
	Operation 1-request 2-Reply.
Sender Hardware Address	
Sender Protocol Address	
Target Hardware Address	
Target Protocol Address	

CN-C32-210364

ii) ICMP: ICMP is used between routers reporting errors and occurs so, the router send 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 or that the receiver will drop the message & reply back ICMP message to the source.

Another important use of ICMP protocol is used to perform network diagnosis by making use of traceroute & ping utility. We will discuss one by one.



Type : Basic description of message.

Code : Additional information

Checksum : Checksum of ICMP.

Extended header : - Point out problem in IP message.

Data or Payload of variable length.

(1) DNS: The Domain Name System (DNS) serves vital role of translating user friendly domain names into numerical IP addresses, enabling seamless internet communication. It allows user to access website & services ~~by~~ using easily memorable names, simplifying web browsing & resources location. It also provides data packet addressing & routing.

0	15	16	31
Identification			Flags
No of questions			No of answers RRs (All DS in query message)
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, number of answer RRs & number of additional RRs are self explanatory.

CN - C32-2103164

A.3) Discuss the persistent & non persistent protocols used in transport & application layer of TCP/IP protocol suite.

Ans → Persistent & non persistent protocols are two different approaches to handling connections in the Transport & Application layer of the 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 connection between sender & receiver before data exchange begins. This connection remains open for multiple transactions.

(ii) Examples : - The most common example of persistent protocol in the TCP/IP suite is the Transmission Control Protocol (TCP). TCP ensures reliable, ordered, error check delivery of data. It maintains a connection until explicitly closed by either the sender or receiver.

(iii) Use cases : Persistent protocols are suitable for applications that require reliable & ordered data transfer such as web browsing, file transfer & most client interactions. They are ideal for situations where data integrity & correctness are crucial.

(iv) Overhead: They typically have more overhead due to the need for connection setup & maintenance which involves three-way handshakes, acknowledgement message & error recovery mechanism.

Non-persistent protocol:-

i) Connectionless: - Non persistent protocols are connectionless, meaning they do not establish a continuous connection between sender & receiver. Instead a new connection is established for each data exchange & it is closed after that exchange is complete.

ii) Examples: A common example is User Datagram Protocol (UDP)

iii) Use cases: - Non persistent protocols are suitable for applications that prioritise speed & efficiency over reliability.

Commonly used in real time data sharing.

(v) Overhead: - Lesser overhead as compared to persistent protocols because they skip connection setup.

Persistent protocols :-

Application: HTTP, FTP, SMTP

Transport: TCP

Non persistent protocols:-

Application Layer: - DNS, SNMP, DHCP

Transport: UDP, SCTP