THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment No. 1

Aim: Study of RJ45 and CAT16 Cabling and connections using crimping tool.

Theory:

- **RJ45 Connectors**

⇒ RJ45 is a standard type of connector for network cables. RJ45 connector due most commonly seen with ethernet cables featuring 8 pins to which the wise standard of a cable interface electrically.

- Standard of RJ45 pinouts defines the arrangement of the individual wires needed when attaching connectors to a cable.
- An 8-pin | 8-position plug or jack is commonly used to connect computers onto Ethernet-based local area networks (LAN).
- Two wiring schemes - T568A and T568B - are used to terminate the twisted-pair cable onto the connector interface.

- **Standard (straight-through) connections**

⇒ A standard ethernet cable is wired in a way that the pins at one end of the cable are connected to the same pin numbers at the other end.

- In other words, the wire at pin 1 on one end of the cable is connected to pin 1 at the other end, the wire at pin 2 is

connected to pin 2, and so on.
- This type of connection is especially used to connect different types of devices, such as a computer to a switch or a router, where the transmit (TX) pin of one device is connected to the receiver (RX) pin of the other device.
- The standard ethernet cable follows the T568B or T568A wiring standard, which determines the color coding of the wires.

- **Crossed-wire connection:**

⇒ A crossed-wire ethernet cable, on the other hand, is wired in a way that certain pins at one end are swapped with corresponding pins at other end.
- The purpose of this wiring is to allow direct communication between two similar devices without the need for an intermediate device like a switch or a hub.
- In past, crossover cables were commonly used to connect two computers directly together without a switch or router.
- In this scenerio, the transmit pin of one computer is connected to the receive pin of other and vice-versa.
- The proper sequence of color codes: Orange/white, Orange, Green/white, Blue, Blue/White, Green, brown/white, brown.

- **Cat5 Cable**

⇒ Cat5 stands for category 5, and it was one of the earlier twisted-pair Ethernet cable standards.

- It supports data transmission up to 100Mbps and can handle network speeds up to 100 MHz.
- Cat5 cables consist of four twisted pairs of copper wires and use the 8P8C modular connector to plug into network devices.

- **Cat 6 Cable**

⇒ Cat 6 cable stands for category 6, and is an improved version of cat5 cable.

- It offers higher performance and bandwidth compared to cat 5.
- Cat 6 cables can support data transmission speeds up to 10Gbps over a maximum distance of 55 meters, and they can handle network speeds up to 250MHz.
- Cat 6 cables are built with tighter specifications and better shielding to reduce crosstalk and interference, which allows for higher data rates and better performance in noisy environment.

• Crimping Process

⇒

1) Strip the cable back 1 inch (25mm) from the end. Insert the cable into stripper section of the crimper and squeeze it tight. Then, rotate the crimping tool around the cable in a smooth and even motion to create a clean cut. Keep the tool clamped and pull away towards the end of the wire to remove sheating.
2) Untwist and straighthen the wires inside the cable cut off the small plastic wire separator or core so its out of the way.
3) Arrange the wires into right order. Use your fingers to put the wires in the correct order so they can be properly crimped.
4) Cut the wire into an even line 1/2 inch from the sheathing. Hold the wire with your thumb and index finger to keep them in order. Thus, use the cutting section of crimper to cut the wires into an even line.
5) Insert the wires into RJ45 connector. Hold the RJ45 connector so the clip of it is on the underside and the small metal pins are facing up. Insert the cable into the connector so that each of the small wires fit in the small grooves.

Om Shete
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
TE C32
2103163

Experiment 2

Aim: Implementation of hamming code for error detection and correction.

Theory:

- Hamming code is a set of error-detection codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver.
- **Redundant Bits**

⇒ Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer.

- The number of redundant bits can be calculated using the following formula:

$$2^r \geq m + r + 1$$

where, r = redundant bits,
m = data bits

- **Even Parity bit**: In the case of even parity, for a given set of bits, the number of 1's are counted.
- If that count is odd, the parity bit value is set to 1, making the total count of occurences of 1's even number.

- If the total number of 1's in a given set of bits is already even, the parity bit value is 0.
- Odd Parity bit : In the case of odd parity, for a given sets of bits, the number of 1's are counted.
- If that count is even, the parity bit value is set to 1, making the total count of occurences of 1's odd number.
- If the total number of 1's in a given set of bits is already odd, the parity bit's value is 0.

- Algorithm:

1) Write the bit position starting from 1 in binary form (1, 10, 11, 100, etc).
2) All the bits positions that are a power of 2 are marked as parity bits.
3) All the other bit positions are marked as data bits.
4) Each data bit is included in a unique set of parity bits as determined its bit position in binary form.
- Parity bit 1 covers all the bits position in binary representation includes a 1 in the least significant position (1, 3, 5, 7, etc)
- Parity bit 2 covers all the bits positions whose binary representation include a 1

in the second position from the least significant bit (2, 3, 6, 7, etc).

- Parity bit 4 covers all the bits position whose binary representation includes a 1 in the third position from the least significant bit (4-7, 12-15, etc).
- In general, each parity bits covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

5) Since we check for even parity set a parity bit to 1 if the total number of ones in the position it checks is odd.

6) Set a parity bit to 0 if the total number of ones in the position it checks is even.

- For e.g.,
  data bit = 1011

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	P_4	1	P_2	P_1

- For P_1, sections to be considered are 1, 3, 5, 7.
  Here, we have to set $P_1 = 1$ as 3, 5, 7 = 111 in order to have even parity.

- **Decide P_2**

⇒ For P_2, sections to be considered 2,3,6,7
Here we have to set $P_2 = 0$ as 3,6,7 = 101 in order to have even parity.

- **Decide P_4**

⇒ For P_4, sections to be considered are 4,5,6,7.
Here we have to set $P_4 = 0$ as 5,6,7 = 101 in order to have the even parity.

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	0	1	0	1

Thus, the code word which is transmitted to the receiver = 1010101.

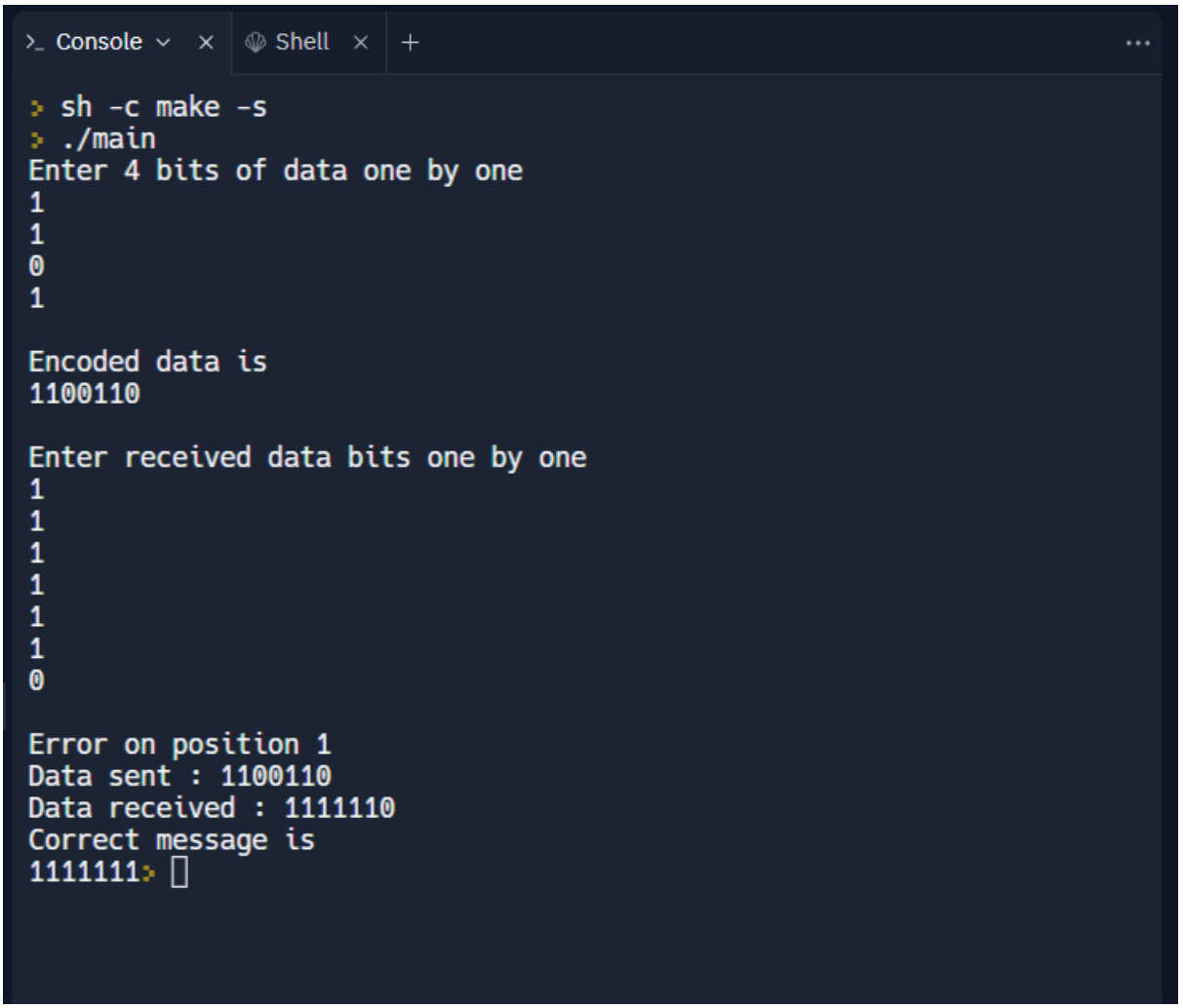
22/08/17

Code:

```
#include <iostream>
using namespace std;

int main() {
  int data[10];
  int dataatrec[10], c, c1, c2, c3, i;

  cout << "Enter 4 bits of data one by one\n";
  cin >> data[0];
  cin >> data[1];
  cin >> data[2];
  cin >> data[4];

  data[6] = data[0] ^ data[2] ^ data[4];
  data[5] = data[0] ^ data[1] ^ data[4];
  data[3] = data[0] ^ data[1] ^ data[2];

  cout << "\nEncoded data is\n";
  for (i = 0; i < 7; i++)
    cout << data[i];

  cout << "\n\nEnter received data bits one by one\n";
  for (i = 0; i < 7; i++)
    cin >> dataatrec[i];

  c1 = dataatrec[6] ^ dataatrec[4] ^ dataatrec[2] ^ dataatrec[0];
  c2 = dataatrec[5] ^ dataatrec[4] ^ dataatrec[1] ^ dataatrec[0];
  c3 = dataatrec[3] ^ dataatrec[2] ^ dataatrec[1] ^ dataatrec[0];
  c = c3 * 4 + c2 * 2 + c1;

  if (c == 0) {
    cout << "\nNo error while transmission of data\n";
  } else {
    cout << "\nError on position " << c;
    cout << "\nData sent : ";
    for (i = 0; i < 7; i++)
      cout << data[i];

    cout << "\nData received : ";
    for (i = 0; i < 7; i++)
      cout << dataatrec[i];
```

```
    cout << "\nCorrect message is\n";

    if (dataatrec[7 - c] == 0)
      dataatrec[7 - c] = 1;
    else
      dataatrec[7 - c] = 0;
    for (i = 0; i < 7; i++) {
      cout << dataatrec[i];
    }
  }
  return 0;
}
```

Output:



Om Shete
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
TE C32
2103163

Experiment 3

Aim : Implementation of CRC for error detection.

Theory :

- **Cyclic Redundancy Check (CRC)**

⇒ An error detection mechanism in which is a special number is appended to a block of data in order to detect any changes introduced during storage.

- The CRC is a method of detecting accidental changes/errors in the communication channel.
- CRC uses generator polynomial which is available on both sender and receiver side.
- An example generator polynomial is of the form like x^3+x+1. This generator polynomial represents key 1011.
- Another example is x^2+1 that represents key 101.
- n : Number of bits in data to be sent from sender side.
  k : Number of bits in the key obtained from generator polynomial.
- **Sender side**

⇒

1. The binary data is first argumented by adding k−1 zeros in end of the data.

2. Use modulo - 2 binary division to divide binary data by the key and store remainder of division.
3. Append the remainder at the end of the data to form the encoded data and send the same.

- Receiver Side

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

- Modulo 2 Division

⇒ The process of modulo-2 binary division is the same as the familiar division process use for decimal number. Instead of substraction, we use XOR here.

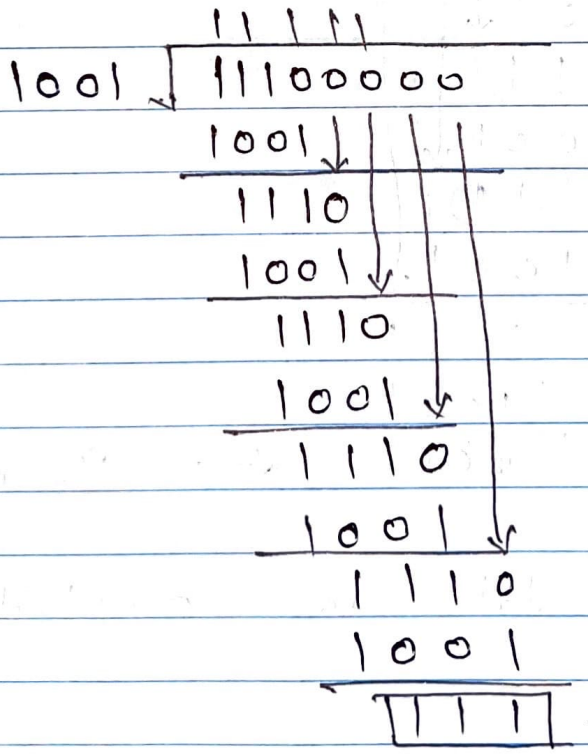
- In each step, a copy of the divisor is XORed with the k bits of the dividend.
- The result of the XOR operation is (n-i) bits, which is used for the next step after 1 extra bit is pulled down to make it n bits long.
- When there are no bits left to pull down, we have a result. The (n-i) bit remainder which is appended at the sender side.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

- **CRC Generator**

⇒

- A CRC generator uses a modulo-2 division. Firstly ~~three zeros~~ (n-1) zeros are appended at the end of the data as the length of divisor is n.
- Now, the consider the string 11100 and divisor 1001.
- Now, the string becomes 11100000, and the resultant string is divided by divisor 1001.
- The remainder generated from the binary division is known as CRC remainder.



- CRC remainder replaces the appended string of 0s at the end of the data unit, and the final string would be 11100111 which is sent across the network.

- **CRC Checker**

⇒ The functionality of the CRC checker is similar to the CRC generator.

- When the string 111001111 is received at the receiving end, the CRC checker performs the modulo-2 division.
- A string is divided by the same divisor, i.e 1001.

```
          11111
      ┌──────────
1001  │ 111001111
        1001↓
        ──────
          1110
          1001↓
          ────
           1111
           1001↓
           ────
            1101
            1001↓
            ────
             1001
             1001
             ────
             0000   ← Remainder = 0
```

- Therefore, the data is accepted.

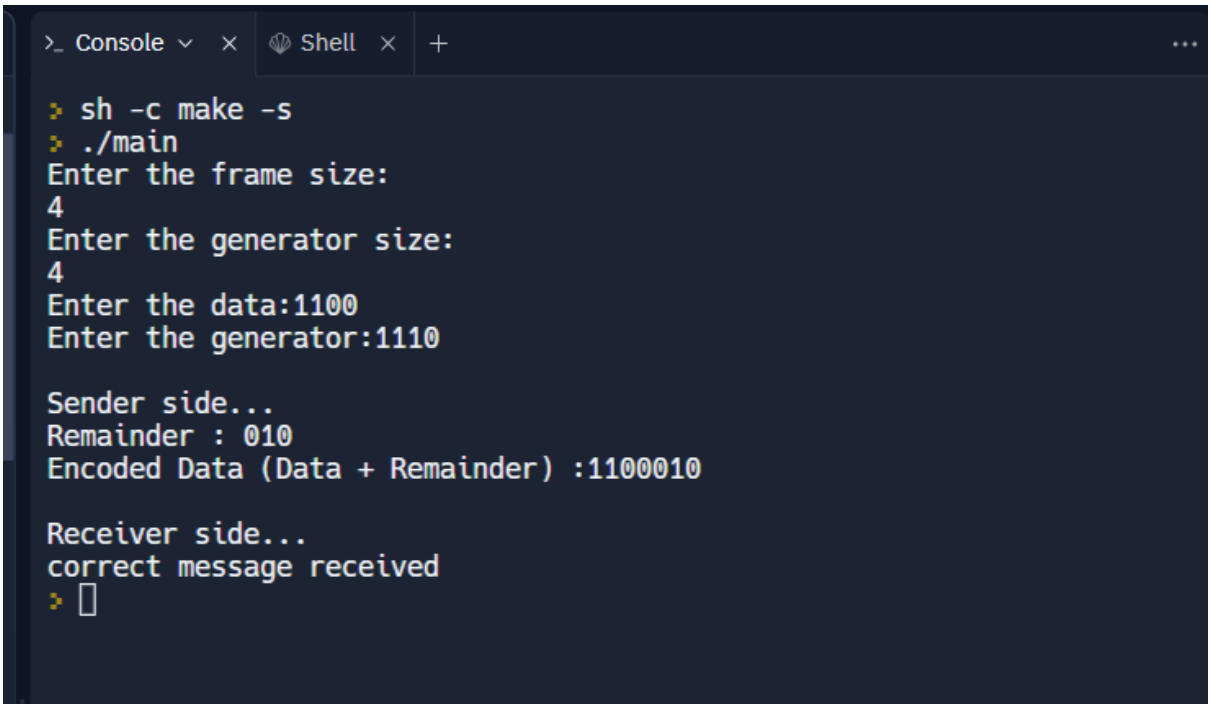
Code:

```
#include <bits/stdc++.h>
using namespace std;

string xor1(string a, string b) {
  string result = "";
  int n = b.length();
  for (int i = 1; i < n; i++) {
    if (a[i] == b[i])
      result += "0";
    else
      result += "1";
  }
  return result;
}

string mod2div(string dividend, string divisor) {
  int pick = divisor.length();
  string tmp = dividend.substr(0, pick);

  int n = dividend.length();

  while (pick < n) {
    if (tmp[0] == '1')
      tmp = xor1(divisor, tmp) + dividend[pick];
    else
      tmp = xor1(std::string(pick, '0'), tmp) + dividend[pick];

    pick += 1;
  }
  if (tmp[0] == '1')
    tmp = xor1(divisor, tmp);
  else
    tmp = xor1(std::string(pick, '0'), tmp);

  return tmp;
}

void encodeData(string data, string key) {
  int l_key = key.length();
  string appended_data = (data + std::string(l_key - 1, '0'));
```

```
  string remainder = mod2div(appended_data, key);

  string codeword = data + remainder;
  cout << "Remainder : " << remainder << "\n";
  cout << "Encoded Data (Data + Remainder) :" << codeword << "\n";
}

void receiver(string data, string key) {
  string currxor = mod2div(data.substr(0, key.size()), key);
  int curr = key.size();
  while (curr != data.size()) {
    if (currxor.size() != key.size()) {
      currxor.push_back(data[curr++]);
    } else {
      currxor = mod2div(currxor, key);
    }
  }
  if (currxor.size() == key.size()) {
    currxor = mod2div(currxor, key);
  }
  if (currxor.find('1') != string::npos) {
    cout << "there is some error in data" << endl;
  } else {
    cout << "correct message received" << endl;
  }
}

int main() {
  string data;
  string generator;
  int frameSize, generatorSize;
  cout << "Enter the frame size: " << endl;
  cin >> frameSize;
  cout << "Enter the generator size: " << endl;
  cin >> generatorSize;
  cout << "Enter the data:";
  cin >> data;
  cout << "Enter the generator:";
  cin >> generator;
  cout << "\nSender side..." << endl;
  encodeData(data, generator);

  cout << "\nReceiver side..." << endl;
  receiver(
```

```
    data + mod2div(data + std::string(generator.size() - 1, '0'), generator),
    generator);

  return 0;
}
```

Output:



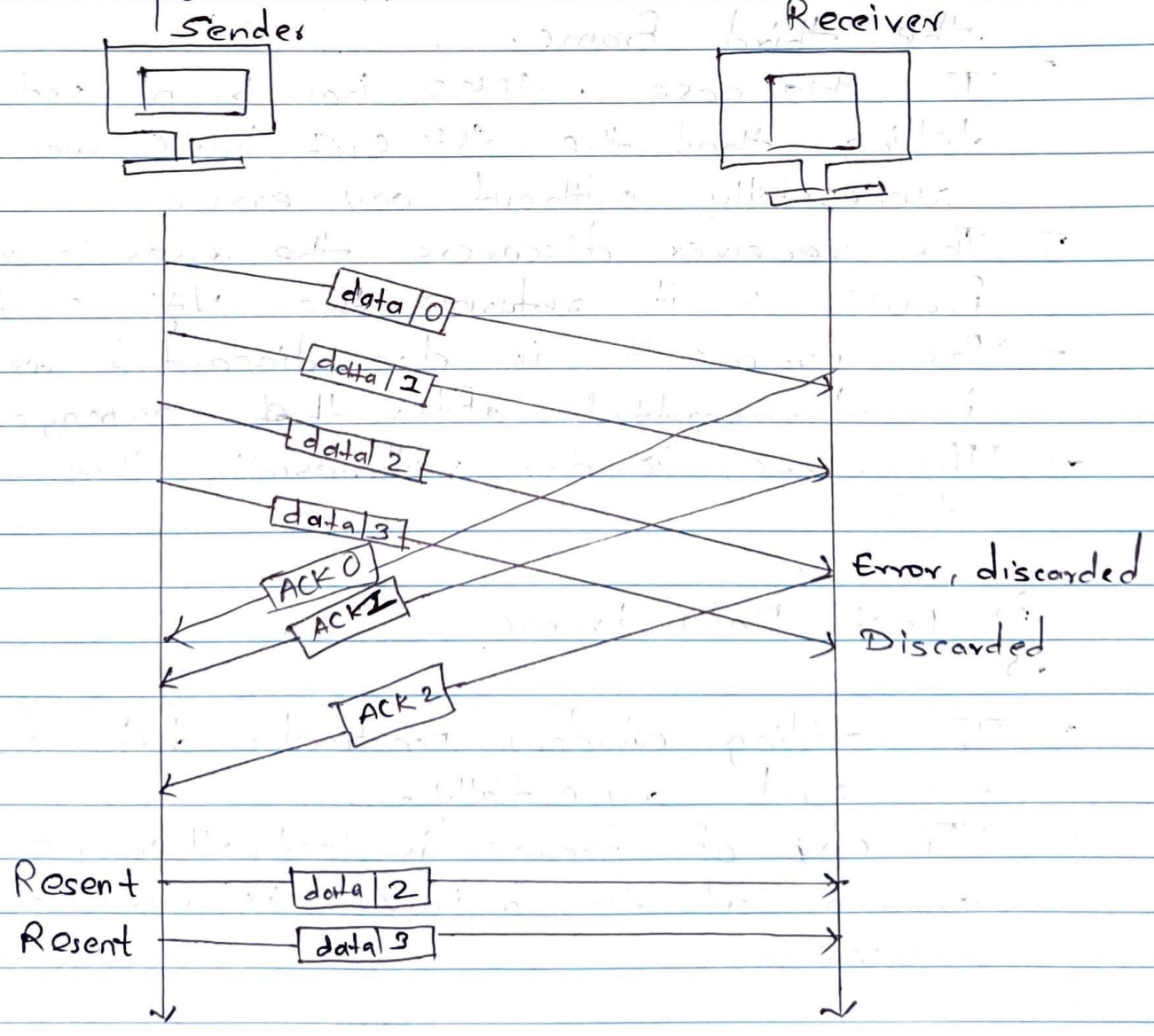
Om Shete
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
TE C32
2103163

Experiment No. 4

Aim: Simulation of Go Back N Flow control Algorithm.

Theory:

- In Go-Back-N ARQ protocol, if one frame is lost or damaged, then it retransmits all the frames after which it does not receive the positive ACK.



- There are three possilibities can occur for retransmission -

1) Damaged Frames

⇒

- When the frame is damaged, the receiver sends a NAK Frame.
- In above figure, three frames have been transmitted before an error discovered in the third frame.
- In this case, ACK 2 has been returned telling that the ACK 0, 1 have been received successfully without any error.
- The receiver discovers the error in data 2 frame, so it returns the NAK 2 frame.
- The frame 3 is also discarded as it is transmitted after that damaged frame.
- Therefore, sender retransmits the frame 2, 3.

2) Lost Data Frame

⇒

- In sliding window protocol, data frames are sent sequentially.
- If any of frame is lost, then next frame arrives at the receiver is out of sequence.
-

- The receiver checks the sequence number of each of the frame, discovers the frame that has been skipped and returns the NAK for the missing frames.
- The sending device retransmits the frame indicated by NAK as well as the frames transmitted after the lost frame.

3) Lost Acknowledgement

⇒

- The sender can send as many frames as the windows allows before waiting for any acknowledgement.
- Once the limit of the window is reached and the sender has no more frames to send; he must wait for the acknowledgement.
- Problem: If the acknowledgemant is lost, then the sender could wait forever.
- Solution: To avoid such situation, the sender is equipped with the timer that start counting whenever the window capacity is reached.
- If the acknowledgenent has not been receeived within the time limit, then the sender retransmits the frame since the last ACK.

02/08/23.

Code:

```
#include <iostream>
using namespace std;

int main() {
  int count;
  cout << "Enter window size : ";
  int windowSize;
  cin >> windowSize;
  cout << "Enter total frames to be sent : ";
  int totalFrames;
  cin >> totalFrames;

  int senderFrames[totalFrames];
  for (int i = 0; i < totalFrames; i++) {
    senderFrames[i] = i;
  }

  for (int i = 0; i < totalFrames; i++) {
    cout << senderFrames[i] << " | ";
  }

  cout << endl;
  cout << "Do you want to start sending frames (0/1) : ";
  int choice;
  cin >> choice;
  cout << endl;

  if (choice == 1) {
    int ptrOnWindowLeftSender = 0;
    int ptrOnWindowLeftReceiver = 0;
    int totalSentFrames = 0;

    while (ptrOnWindowLeftSender < totalFrames) {
      count = 0;
      cout << "At Sender End:" << endl;
      for (int i = ptrOnWindowLeftSender;
           (i < totalFrames && count < windowSize); i++) {
        cout << "Sent frame[" << (i + 1) << "]" << endl;
        ptrOnWindowLeftSender++;
        totalSentFrames++;
        count++;
      }
      cout << endl;

      // Receiver side
      cout << "At Receiver end: " << endl;
      int j = 0;
      count = 0;
      for (int i = ptrOnWindowLeftReceiver;
           (i < totalFrames && count < windowSize); i++) {
```

```
        char yN;
        cout << "Did you receive frame[" << (i + 1) << "] (y/n) : ";
        cin >> yN;
        if (yN == 'n') {
          cout << "Frames will be sent again from frame no. " << (i + 1)
               << endl;
          cout << endl;
          ptrOnWindowLeftSender = i;
          break;
        } else {
          j++;
          ptrOnWindowLeftReceiver++;
        }
        count++;
      }

      if (j == windowSize) {
        cout << "All Frames from this window sent without errors. Sending next "
                "frames..."
             << endl;
        cout << endl;
      }
    }

    cout << endl;
    cout << "All frames are sent." << endl;
    cout << "Total no. of frames sent including retransmission is "
         << totalSentFrames << endl;
  }

  return 0;
}
```

Output:

```
Enter window size : 4
Enter total frames to be sent : 10
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
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) : n
Frames will be sent again from frame no. 2

At Sender End:
Sent frame[2]
Sent frame[3]
Sent frame[4]
Sent frame[5]

At Receiver end:
Did you receive frame[2] (y/n) : y
Did you receive frame[3] (y/n) : y
Did you receive frame[4] (y/n) : y
Did you receive frame[5] (y/n) : y
All Frames from this window sent without errors. Sending next frames...

At Sender End:
Sent frame[6]
Sent frame[7]
Sent frame[8]
Sent frame[9]
```

```
At Receiver end:
Did you receive frame[6] (y/n) : y
Did you receive frame[7] (y/n) : y
Did you receive frame[8] (y/n) : y
Did you receive frame[9] (y/n) : y
All Frames from this window sent without errors. Sending next frames...

At Sender End:
Sent frame[10]

At Receiver end:
Did you receive frame[10] (y/n) : y

All frames are sent.
Total no. of frames sent including retransmission is 13
```

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment No. 5

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

Theory:

- CISCO Packet Tracer as the name suggests, is a tool built by CISCO. This tool provides a network simulation to practice simple and complex networks.
- The main purpose of CISCO packet tracer is to help students learn the principles of networking with hands-on experience as well as develop CISCO technology specific skills.

- WORKSPACE for CISCO Packet Tracer

⇒

1. **Logical**

⇒ The logical workspace shows the logical network topology that is built by the user.
- It displays the connecting, planning and clustering of virtual network devices.

2. **Physical**

⇒ In the physical workspace, we can see the physical implementation of the logical network.

- It also shows how the network devices such as switches, routers and hosts are connected in a real network topology.

- Features of CISCO Packet Tracer.

⇒

1) CISCO packet tracer supports the multi-user system that allows any user to connect in different topologies accross different computer networks. By using this, the teacher assigns different tasks to different students.
2) We can also remove the capabilities of the CISCO packet tracer with the help of an API. This feature is also provided by the CISCO packet tracer.
We can also remove the special feature like accesebility, gaming, assement delivery and interaction with the real-world from the CISCO packet tracer.
3) We can also simulate the configuration related to routers, and this can be accessed anywhere.
4) We can access this configuration with unlimited devices.
5) It also provides a self-placed and interactive environment.

6) The enhanced physical mode transports you to a virtual lab where you can simulate calling devices on a rack. Refresh key skills such as device placement, on-device power switching, device port-to-port cabling, troubleshoot, and more.

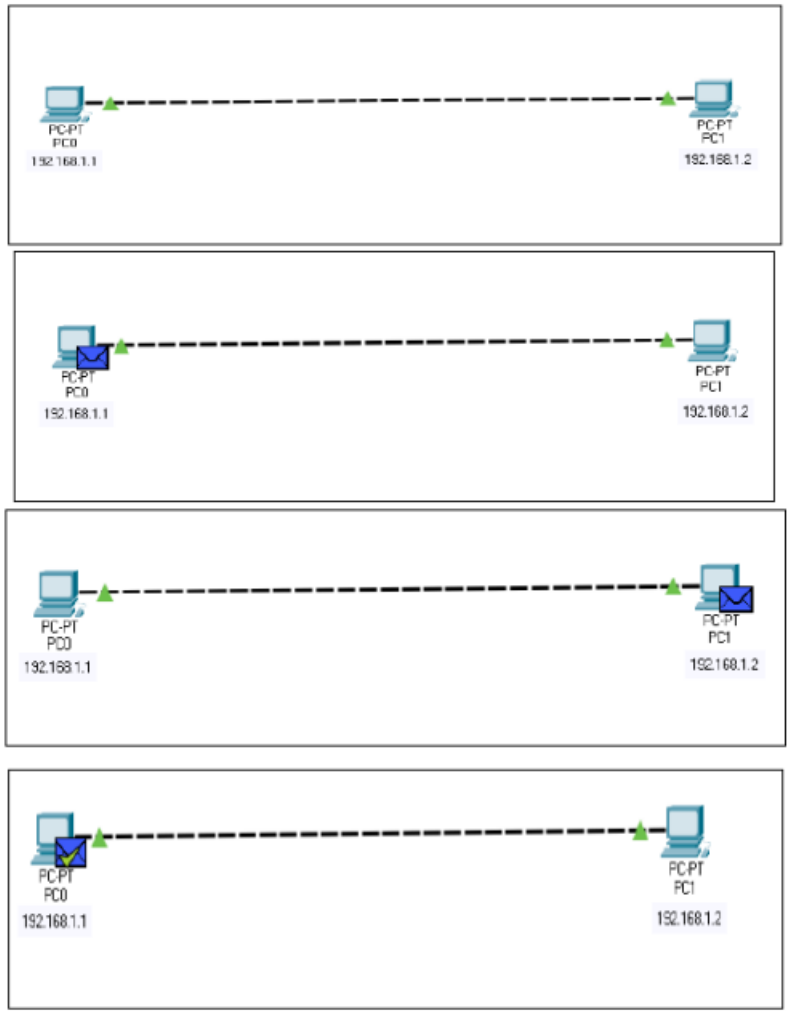
7) The network controller allows you a centralized dashboard to see the network's state, instantly discover and diagnose issue and push configuration changes to all managed devices at once, whether you use its Web GUI or its APIs.

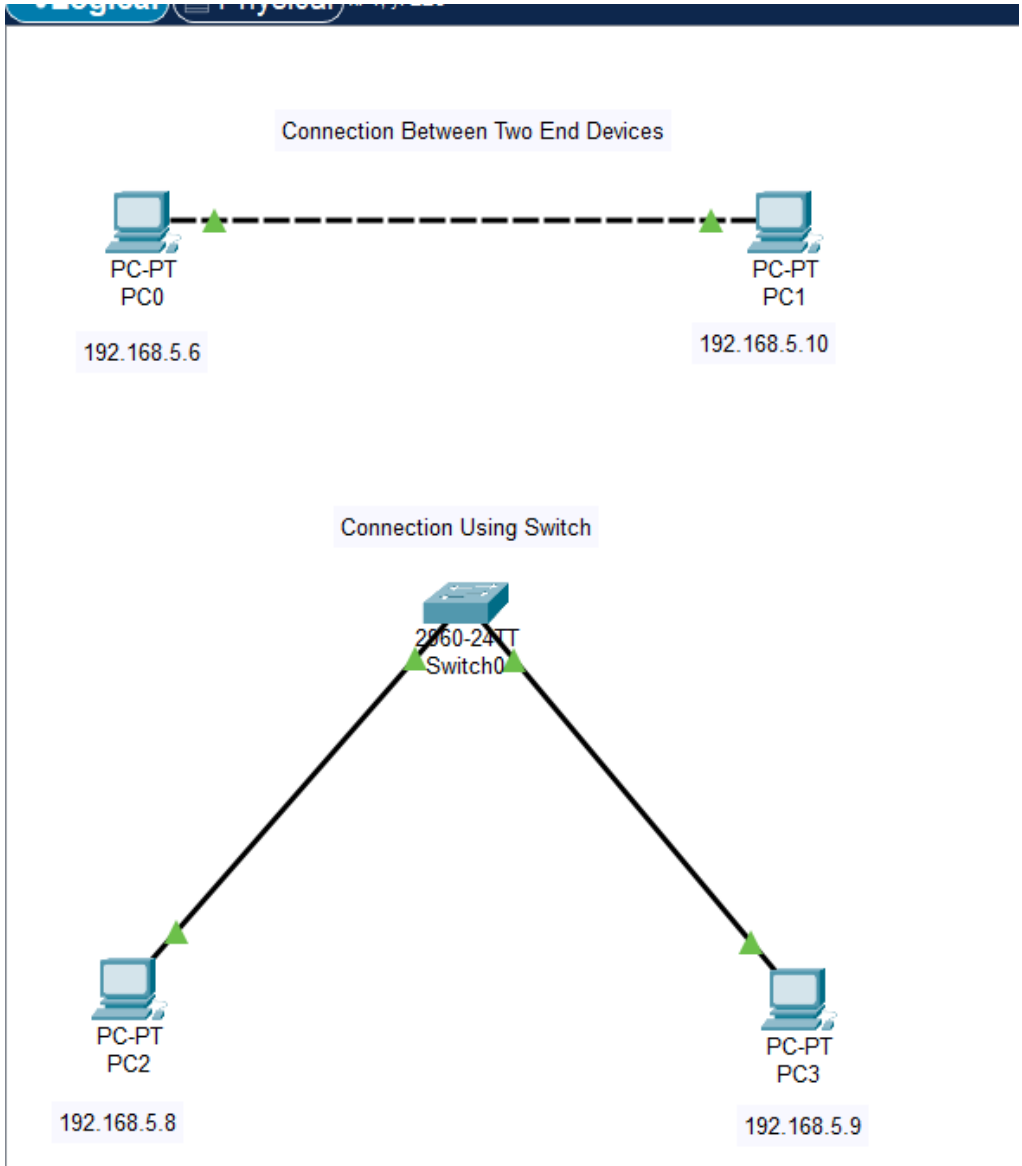
26/09/23

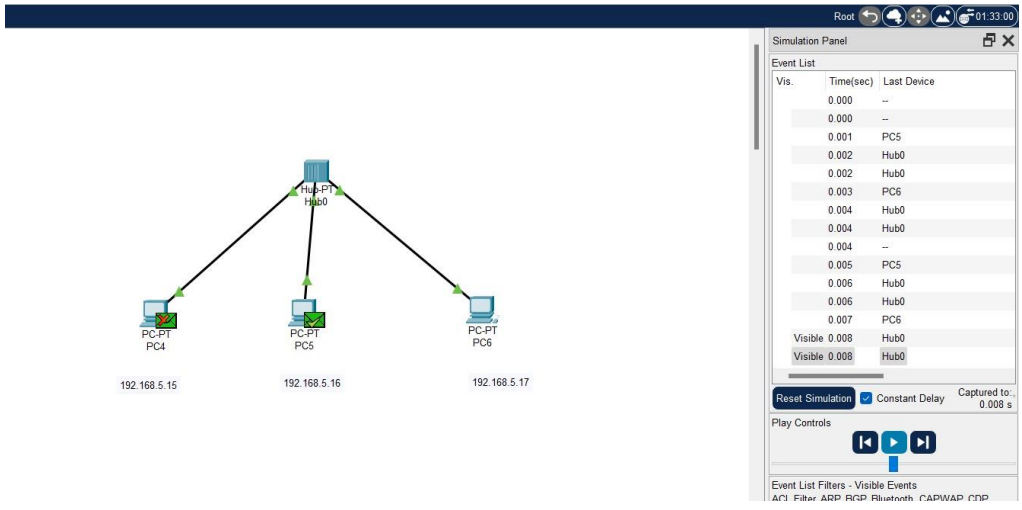
(A)

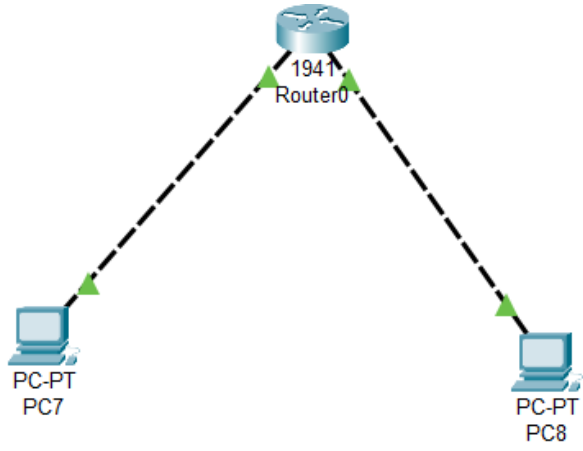
Output:

PC to PC



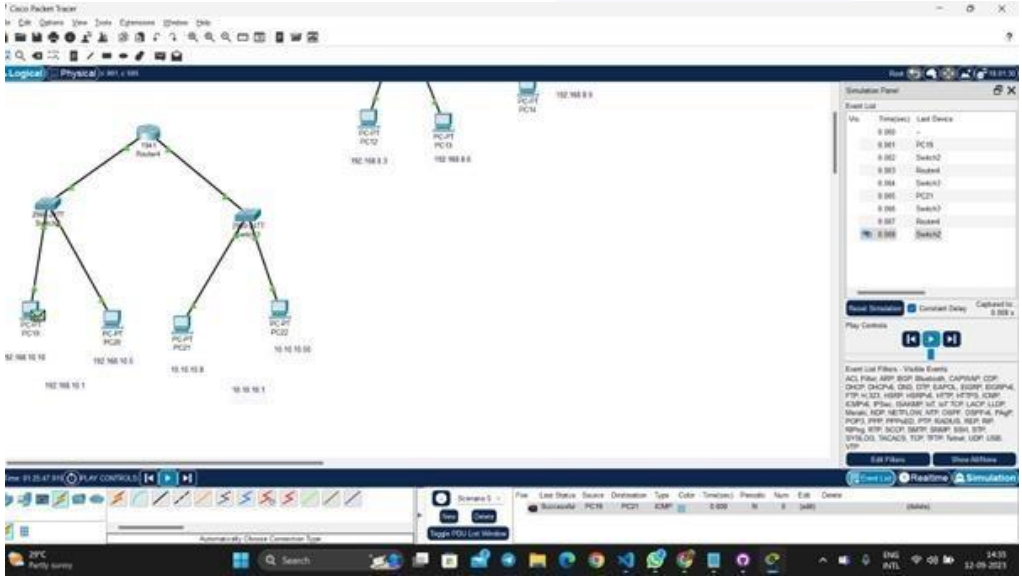

Connection Between Two End Devices
PC-PT
PC0
192.168.5.6
PC-PT
PC1
192.168.5.10
Connection Using Switch
2960-24TT
Switch0
PC-PT
PC2
192.168.5.8
PC-PT
PC3
192.168.5.9





10.10.10.5
Gateway address: 10.10.10.1

192.168.5.10
gateway address: 192.168.5.1



Om Shete
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
TE C32
2103163

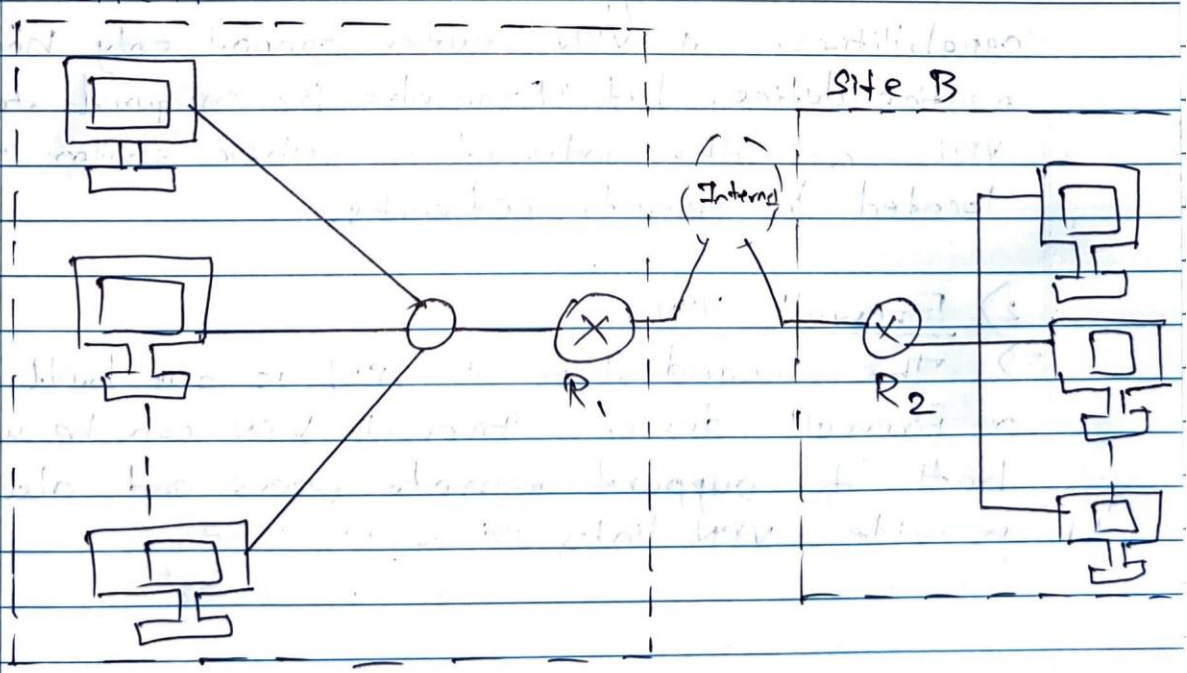
Experiment No: 6

Aim: Design VPN and Configure RIP/OSPF using packet tracer.

Theory:

- **Virtual Private Network (VPN)**:

- VPN stands for virtual private network.
- It allows you to connect your computer to a private network, creating an encrypted connections that masks your IP address to securely share data and surf the web, protecting your identity online.
- A VPN connection is shown in the figure below:



- In this figure, Routers R_1 and R_2 use VPN technology to guarantee privacy for the organization.
- A VPN is an encrypted connection over the Internet from a device to a network.
- The encrypted connection helps ensure that sensitive data is safely transmitted.
- It prevents unauthorized people from eavesdropping on traffic and allows the user to conduct work remotely.
- VPN technology widely used in corporate environments.
- Types of VPNs :

1) Router VPN

⇒ The first type uses a router with added VPN capabilities. A VPN router cannot only handle normal routine duties, but it can also be configured to form VPNs over the internet to other similar routers located in remote networks.

2) Firewall VPN

⇒ The second type of VPN is one build into a firewall device. Firewall VPN can be used both to support remote users and also to provide VPN links

- **Routing Information Protocol (RIP)**

⇒ RIP is dynamic routing protocol that uses hop count as a routing metric to find the best path between the source and the destination network.

- It is a distance-vector routing protocol that has an AD value of 120 and works on the Network layer of OSI model.
- RIP uses port number 520.
- **Hop Count**

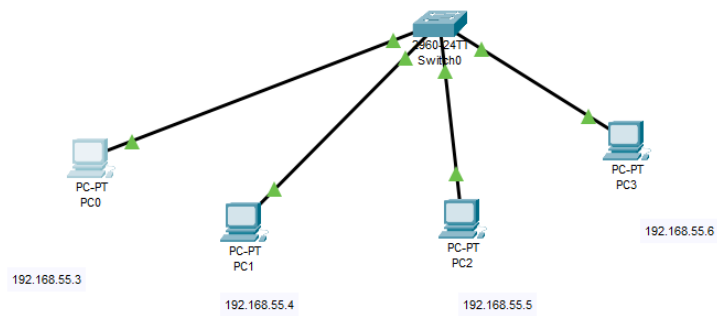
⇒ Hop count is the no. of routers occuring in between the source and destination network.

- The path with the lowest hop count is considered as the best route to reach a network and therefore placed in routing table.
- RIP prevents routing loops by limiting the number of loops allowed in a path from source and destination.
- **Features of RIP :**

1) Updates of the network are exchanged periodically.
2) Updates are always broadcast.
3) Full routing tables are sent in updates.
4) Routers always trust routing information received from neighbour routers. This is also known as Routing on rumours.

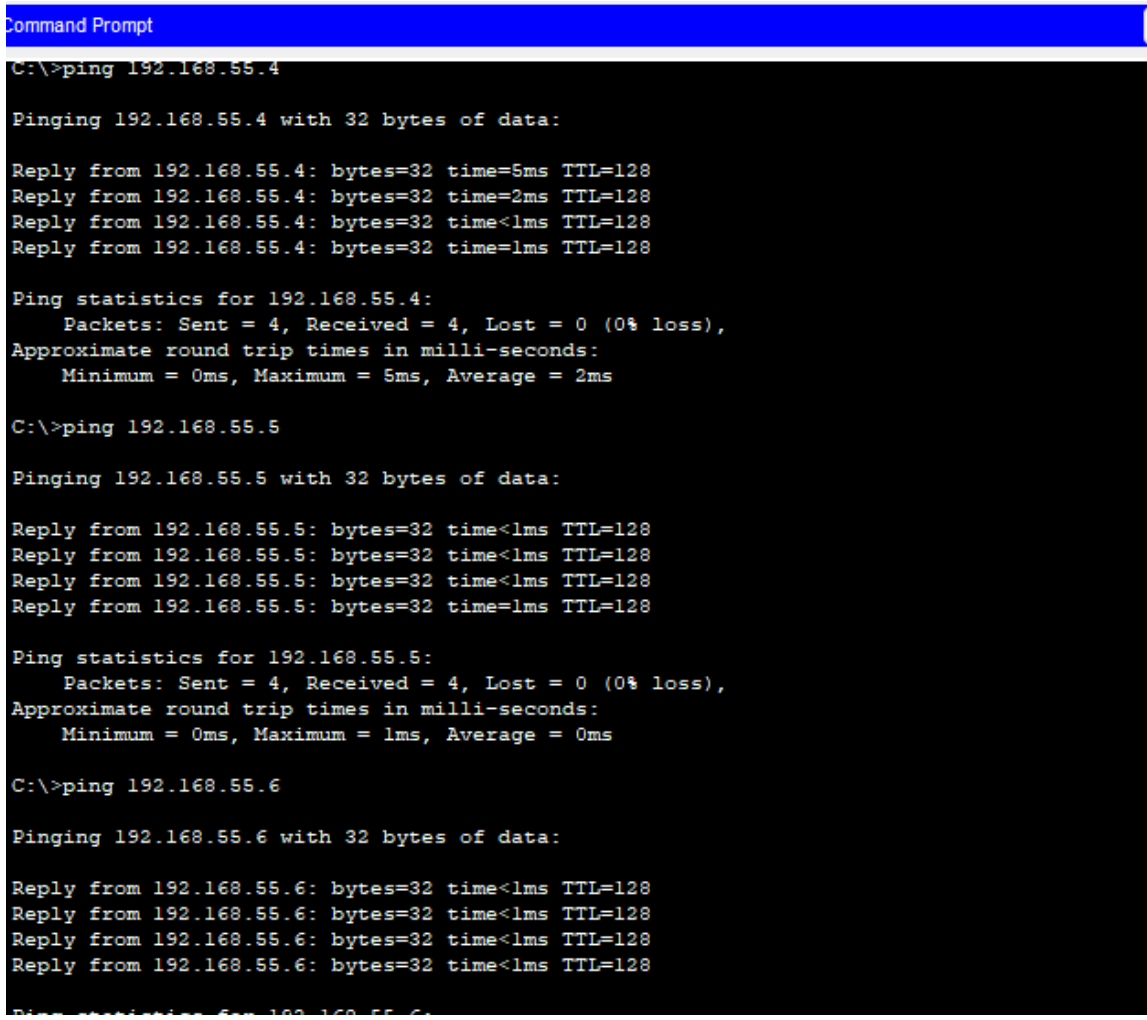
Output:

VPN

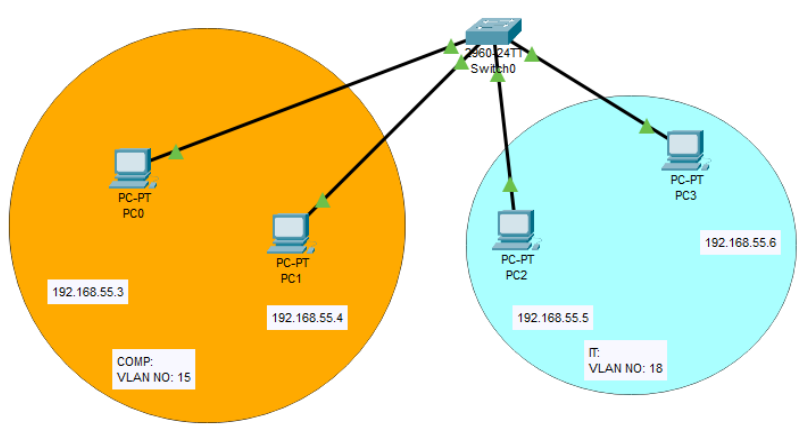


Command Prompt

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

FastEthernet0 Connection:(default port)

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.....: FE80::260:5CFF:FE90:477B
   IPv6 Address.....: ::
   IPv4 Address.....: 192.168.55.3
   Subnet Mask.....: 255.255.255.0
   Default Gateway.....: ::
                                 0.0.0.0

Bluetooth Connection:

   Connection-specific DNS Suffix..:
   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 192.168.55.3

Pinging 192.168.55.3 with 32 bytes of data:

Reply from 192.168.55.3: bytes=32 time=1ms TTL=128
Reply from 192.168.55.3: bytes=32 time=5ms TTL=128
Reply from 192.168.55.3: bytes=32 time=4ms TTL=128
Reply from 192.168.55.3: bytes=32 time=3ms TTL=128

Ping statistics for 192.168.55.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 5ms, Average = 3ms
```

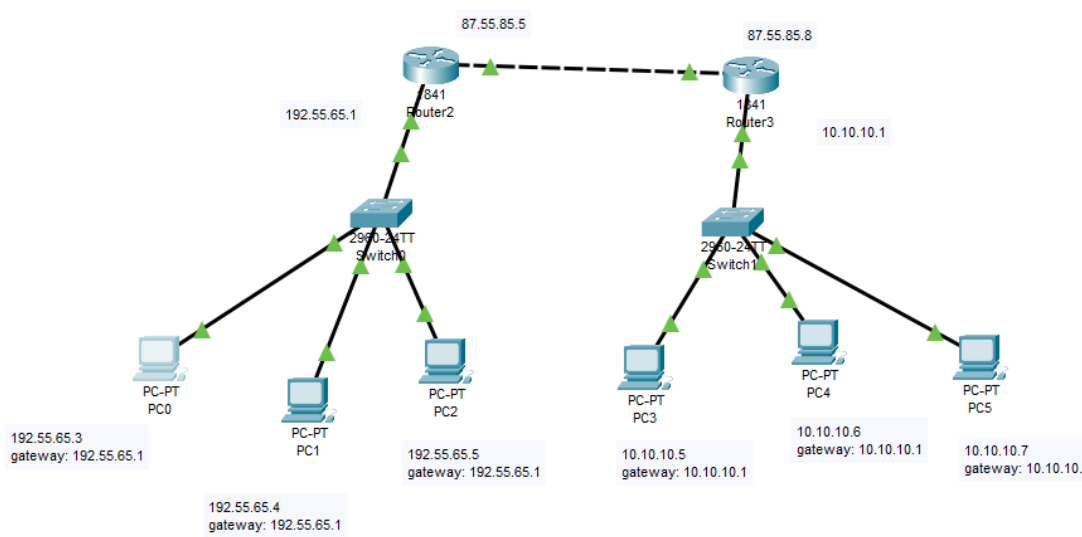

Command Prompt
C:\>ping 192.168.55.4
Pinging 192.168.55.4 with 32 bytes of data:
Reply from 192.168.55.4: bytes=32 time=5ms TTL=128
Reply from 192.168.55.4: bytes=32 time=2ms TTL=128
Reply from 192.168.55.4: bytes=32 time<1ms TTL=128
Reply from 192.168.55.4: bytes=32 time=1ms TTL=128
Ping statistics for 192.168.55.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 5ms, Average = 2ms
C:\>ping 192.168.55.5
Pinging 192.168.55.5 with 32 bytes of data:
Reply from 192.168.55.5: bytes=32 time<1ms TTL=128
Reply from 192.168.55.5: bytes=32 time<1ms TTL=128
Reply from 192.168.55.5: bytes=32 time<1ms TTL=128
Reply from 192.168.55.5: bytes=32 time=1ms TTL=128
Ping statistics for 192.168.55.5:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
C:\>ping 192.168.55.6
Pinging 192.168.55.6 with 32 bytes of data:
Reply from 192.168.55.6: bytes=32 time<1ms TTL=128
Reply from 192.168.55.6: bytes=32 time<1ms TTL=128
Reply from 192.168.55.6: bytes=32 time<1ms TTL=128
Reply from 192.168.55.6: bytes=32 time<1ms TTL=128

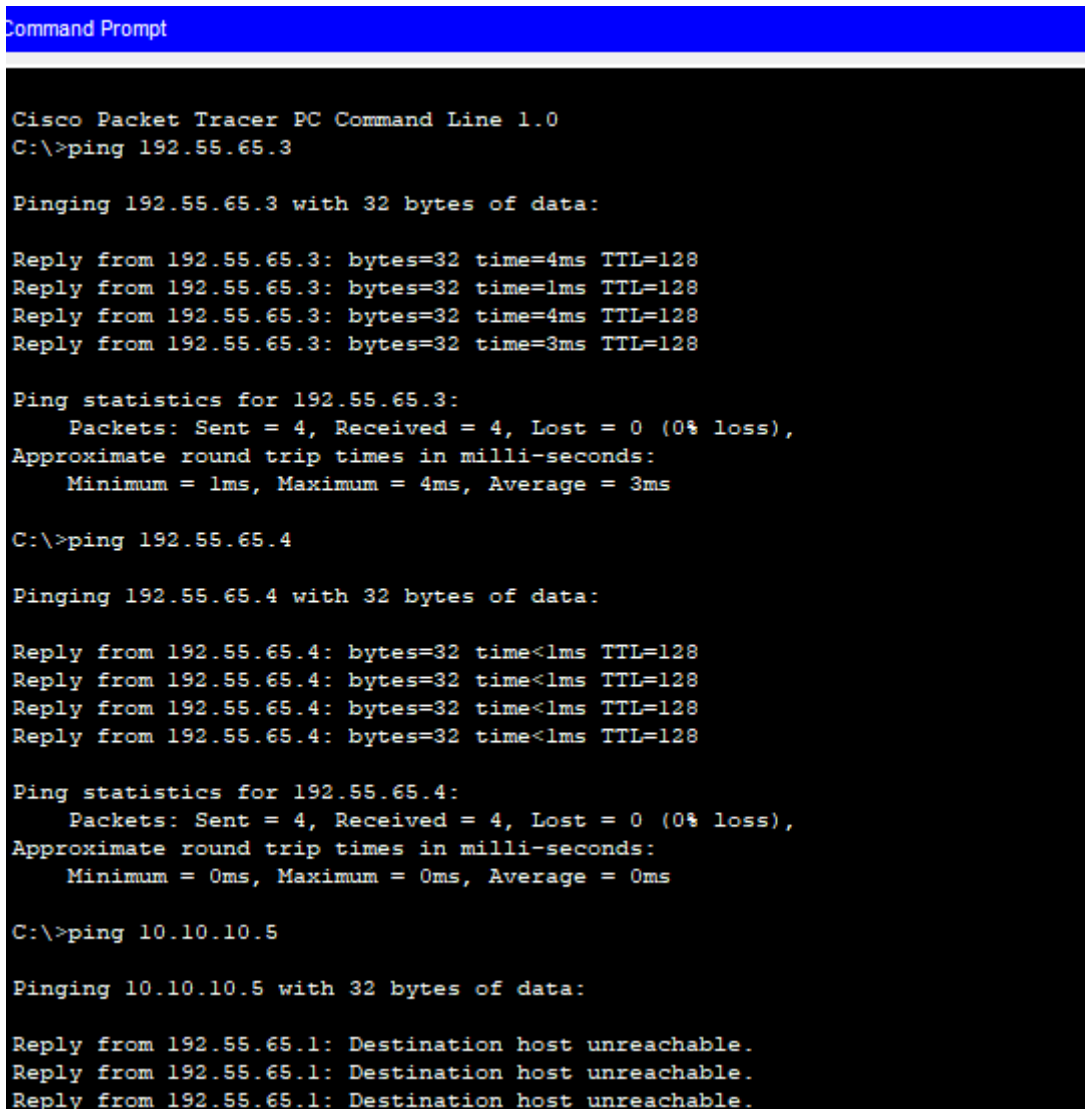
```
Command Prompt

Reply from 192.168.55.4: bytes=32 time<1ms TTL=128
Reply from 192.168.55.4: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.55.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 5ms, Average = 2ms

C:\>ping 192.168.55.5

Pinging 192.168.55.5 with 32 bytes of data:

Reply from 192.168.55.5: bytes=32 time<1ms TTL=128
Reply from 192.168.55.5: bytes=32 time<1ms TTL=128
Reply from 192.168.55.5: bytes=32 time<1ms TTL=128
Reply from 192.168.55.5: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.55.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>ping 192.168.55.6

Pinging 192.168.55.6 with 32 bytes of data:

Reply from 192.168.55.6: bytes=32 time<1ms TTL=128
Reply from 192.168.55.6: bytes=32 time<1ms TTL=128
Reply from 192.168.55.6: bytes=32 time<1ms TTL=128
Reply from 192.168.55.6: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.55.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

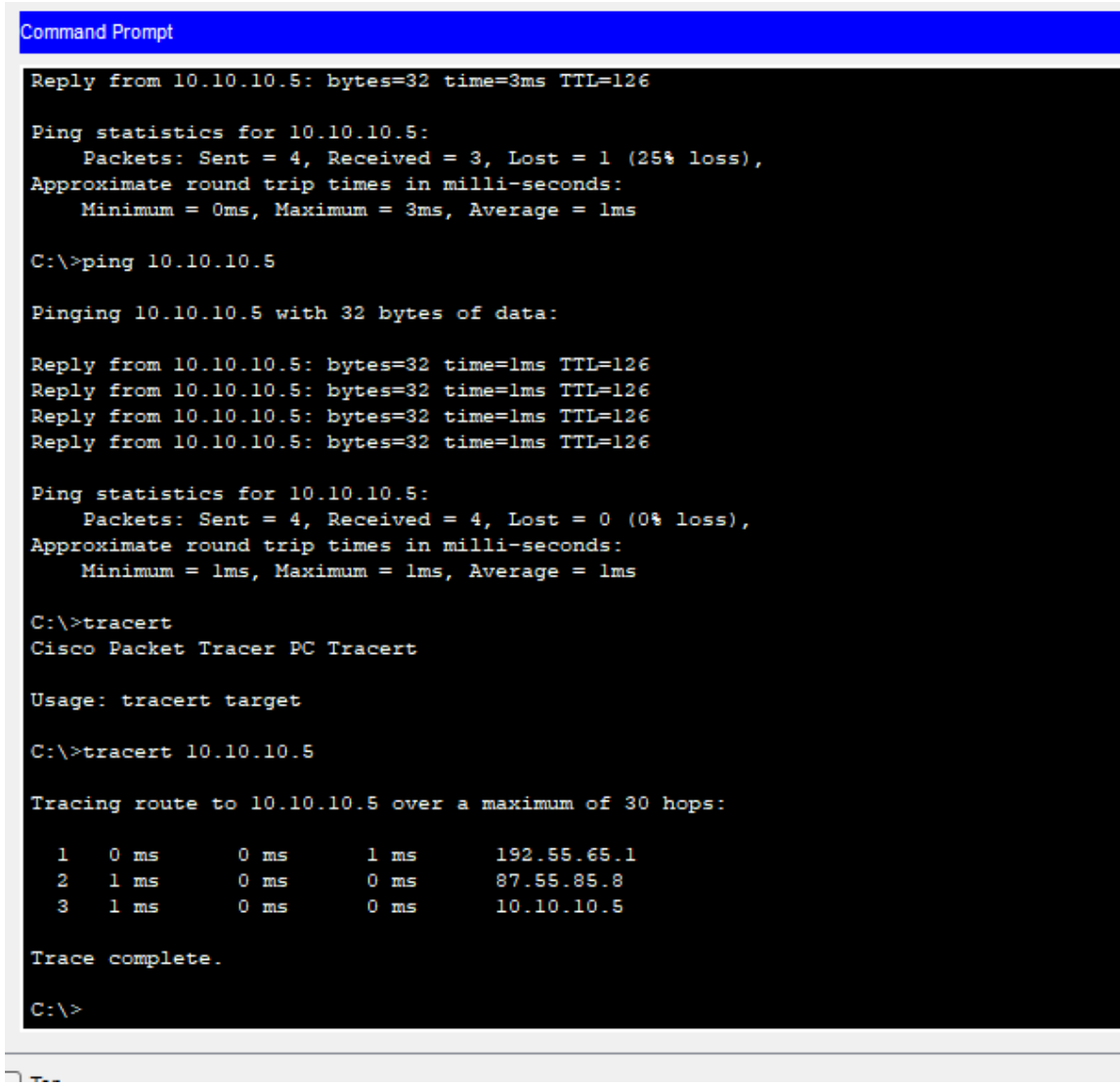


```
C:\>ping 192.168.55.4

Pinging 192.168.55.4 with 32 bytes of data:

Reply from 192.168.55.4: bytes=32 time<1ms TTL=128
Reply from 192.168.55.4: bytes=32 time=1ms TTL=128
Reply from 192.168.55.4: bytes=32 time<1ms TTL=128
Reply from 192.168.55.4: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.55.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>ping 192.168.55.5

Pinging 192.168.55.5 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.55.5:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>
```

RIP:




Command Prompt
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.55.65.3
Pinging 192.55.65.3 with 32 bytes of data:
Reply from 192.55.65.3: bytes=32 time=4ms TTL=128
Reply from 192.55.65.3: bytes=32 time=1ms TTL=128
Reply from 192.55.65.3: bytes=32 time=4ms TTL=128
Reply from 192.55.65.3: bytes=32 time=3ms TTL=128
Ping statistics for 192.55.65.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 1ms, Maximum = 4ms, Average = 3ms
C:\>ping 192.55.65.4
Pinging 192.55.65.4 with 32 bytes of data:
Reply from 192.55.65.4: bytes=32 time<1ms TTL=128
Reply from 192.55.65.4: bytes=32 time<1ms TTL=128
Reply from 192.55.65.4: bytes=32 time<1ms TTL=128
Reply from 192.55.65.4: bytes=32 time<1ms TTL=128
Ping statistics for 192.55.65.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\>ping 10.10.10.5
Pinging 10.10.10.5 with 32 bytes of data:
Reply from 192.55.65.1: Destination host unreachable.
Reply from 192.55.65.1: Destination host unreachable.
Reply from 192.55.65.1: Destination host unreachable.


Command Prompt
Reply from 10.10.10.5: bytes=32 time=3ms TTL=126
Ping statistics for 10.10.10.5:
Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 3ms, Average = 1ms
C:\>ping 10.10.10.5
Pinging 10.10.10.5 with 32 bytes of data:
Reply from 10.10.10.5: bytes=32 time=1ms TTL=126
Reply from 10.10.10.5: bytes=32 time=1ms TTL=126
Reply from 10.10.10.5: bytes=32 time=1ms TTL=126
Reply from 10.10.10.5: bytes=32 time=1ms TTL=126
Ping statistics for 10.10.10.5:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 1ms, Maximum = 1ms, Average = 1ms
C:\>tracert
Cisco Packet Tracer PC Tracert
Usage: tracert target
C:\>tracert 10.10.10.5
Tracing route to 10.10.10.5 over a maximum of 30 hops:
1 0 ms 0 ms 1 ms 192.55.65.1
2 1 ms 0 ms 0 ms 87.55.85.8
3 1 ms 0 ms 0 ms 10.10.10.5
Trace complete.
C:\>

Om Shete
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
TE C32
2103163

Experiment No. 7

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

Theory :

- **IPv4 Addressing**

⇒

- Every host and router on the Internet has an IP address that can be used in the source address and destination address fidds of IP packet.
- IP address is assigned to each and every host and router on the Internet which is referred in the source as well as destination address fields of IP packets.
- One significant aspect is that an IP address does not refer to host in reality.
- It actually refers to a network interface, hence if presence of a host is on two networks, then there must be two IP addresses assigned to it.
- However, in general, the presence of most of the hosts is on single network and accordingly has one IP address.
- Whereas, there are multiple interfaces of router and accordingly have multiple IP addresses.
- The address space of IPv4 is 2^{32}.

For e.g.,
binary notation to decimal dotted notation
⇒ 11000011 10000011 00011010 11111111

We replace each group of 8 bits with its equivalent decimal number and add dots for separation.
∴ 193.131.26.255

- **Subnet**

⇒ A subnet is simply a subdivision of a network address that can be used to represent one LAN on the internetwork.

- **Subnet Mask**

⇒
- An IP address has two components,
  - The network address and
  - The host address.
- Subnet mask is a mask used to determine what subnet an IP address belongs to.
- The subnet mask is the network address plus the bits reserved for identifying the subnetwork -- by convention, bits for the network address are all set to 1, though it would also work if the bits were set exactly as in the network address.

• Characterstics of IPv4.

⇒
- IPV4 could be a 32-bit IP address.
- IPv4 could be a numeric address, and its bits are separated by a dot.
- The number of header fields is twelve and the length of the header field is twenty.
- IPv4 supports VLSM (Virtual Length Subnet Mask).
- IPv4 uses the Port Address Resolution Protocol to map to the MAC address.

• Advantages of IPv4

⇒
- IPv4 security permits encryption to keep up privacy and security.
- IPv4 network allocation is significant and presently has quite 85000 practical routers.
- It becomes easy to attach multiple devices accross an outsized network while not NAT.
- IPv4 addresses are redefined and permit flawless encoding.

- In IPV4, we have 5 classes A, B, C, D, E.
- It is clear that first byte / Octet tells us to which class address belongs and values for class A, B, C, D and E. HOB decides the class.
- Addresses are denoted by 4 decimal numbers from 0 to 255, separated by dot
- for e.g., 192.168.3.39

	1st octet range	1st octet value/HOB
class A	1 - 126*	0
class B	128 - 191	10
class C	192 - 233	110
class D	224 - 239	1110
class E	240 - 255	1111

*127 is reserved for loopback.

26/09/23

```
import ipaddress

# Input the IP address
ip_address = input("Enter the IP address (e.g., 192.168.1.0): ")
print(f"Net ID: {ip_address}")
print(f"Class C")
subnet_mask = "255.255.255.0"
print(f"Network Mask: {subnet_mask}")

# Create an IPv4 network object
network = ipaddress.IPv4Network(f"{ip_address}/{subnet_mask}", strict=False)

# Set the number of subnets to 4
num_subnets = int(input("Enter the no. of subnets: "))

# Calculate the custom prefix length with 2 bits for subnet ID
custom_prefix_length = network.prefixlen + 2

# Calculate the subnet size
subnet_size = 2 ** (32 - custom_prefix_length)

print(f"Total number of IP addresses possible in each subnet: {subnet_size} addresses")

# Iterate and print each of the 4 subnets with a custom prefix length of 30 bits
for i, subnet in enumerate(network.subnets(new_prefix=custom_prefix_length)):
    if i >= num_subnets:
        break
    print(f"For Subnet {i + 1}:")
    print(f"Subnet Address: {subnet.network_address}")
    print(f"Broadcast Address: {subnet.broadcast_address}")
    print(f"Valid Range: {subnet.network_address + 1} to {subnet.broadcast_address - 1}")
    print()
```

```
Enter the IP address (e.g., 192.168.1.0): 193.121.100.0
Net ID: 193.121.100.0
Class C
Network Mask: 255.255.255.0
Enter the no. of subnets: 4
Total number of IP addresses possible in each subnet: 64 addresses
For Subnet 1:
Subnet Address: 193.121.100.0
Broadcast Address: 193.121.100.63
Valid Range: 193.121.100.1 to 193.121.100.62

For Subnet 2:
Subnet Address: 193.121.100.64
Broadcast Address: 193.121.100.127
Valid Range: 193.121.100.65 to 193.121.100.126

For Subnet 3:
Subnet Address: 193.121.100.128
Broadcast Address: 193.121.100.191
Valid Range: 193.121.100.129 to 193.121.100.190

For Subnet 4:
Subnet Address: 193.121.100.192
Broadcast Address: 193.121.100.255
Valid Range: 193.121.100.193 to 193.121.100.254
```

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
Om Shete
TE C32
2103163

Experiment No. 8

Aim: Use basic networking commands in Linux (Ping, track, nslookup, netstat, ARP, RARP, ip, ifconfig, dig, route).

Theory:

- **Ping**

⇒ It is used to test network connectivity to a host.

For e.g., ping 192.168.32.31

- **Traceroute**

⇒ To trace the route packets take to reach a destination.

For e.g., traceroute www.google.com

- **nslookup**

⇒ To query DNS servers to resolve domain names.

For e.g., nslookup www.google.com

- **netstat**

⇒ To display network statistics and active connections.

For e.g., netstat -a
netstat -l
netstat -at
netstat -au

- **arp**

⇒ To display or manipulate the ARP cache.

For e.g., arp -a

- **rarp**

⇒ Reverse ARP, used to map MAC addresses to IP addresses.

For e.g., rarp -a

- **ip**

⇒ A versatile command for configuring and managing network interfaces.

For e.g., ip addr

- **if config**

⇒ An older command for ~~int~~ network interface configuration.

For e.g., ifconfig.

- **dig**

⇒ A tool for querying DNS servers for domain information.

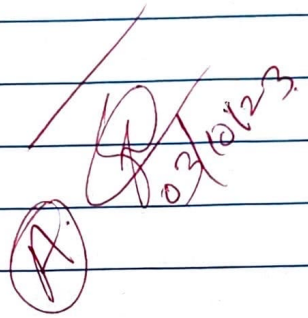
For e.g., dig www.google.com

- **route**

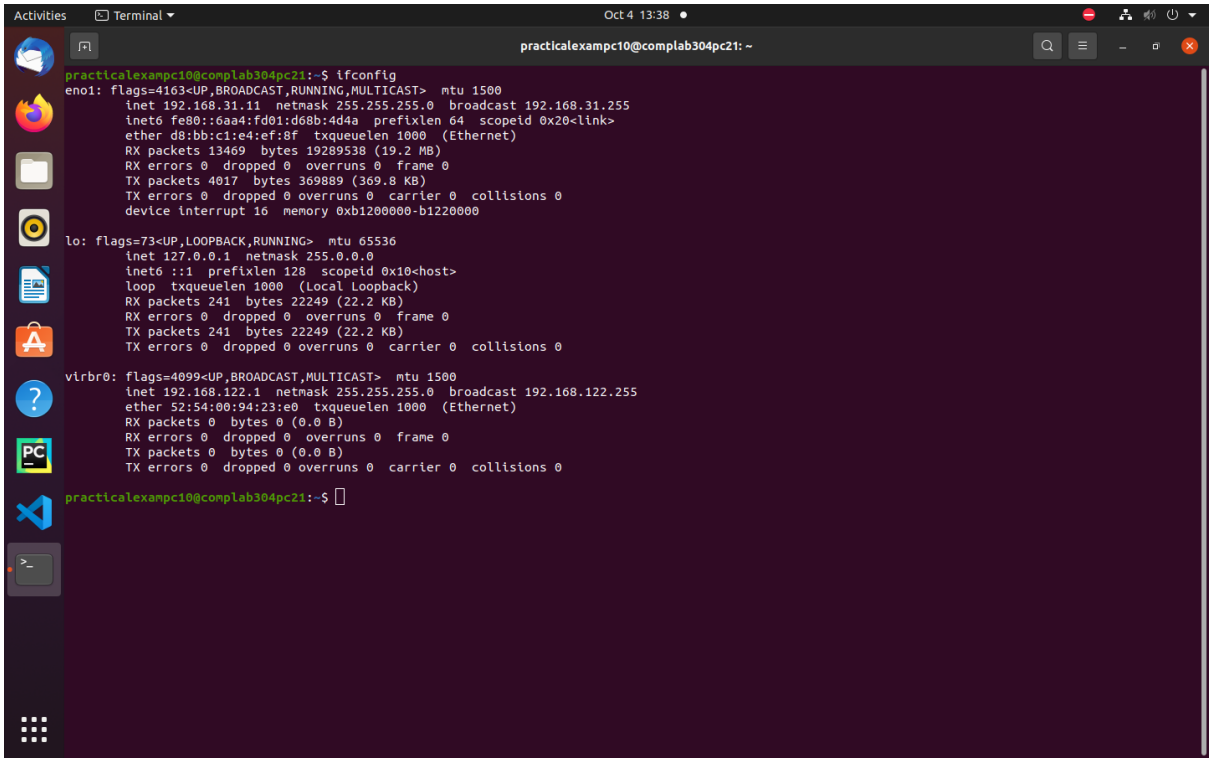
⇒ To view and manipulate the IP routing table.

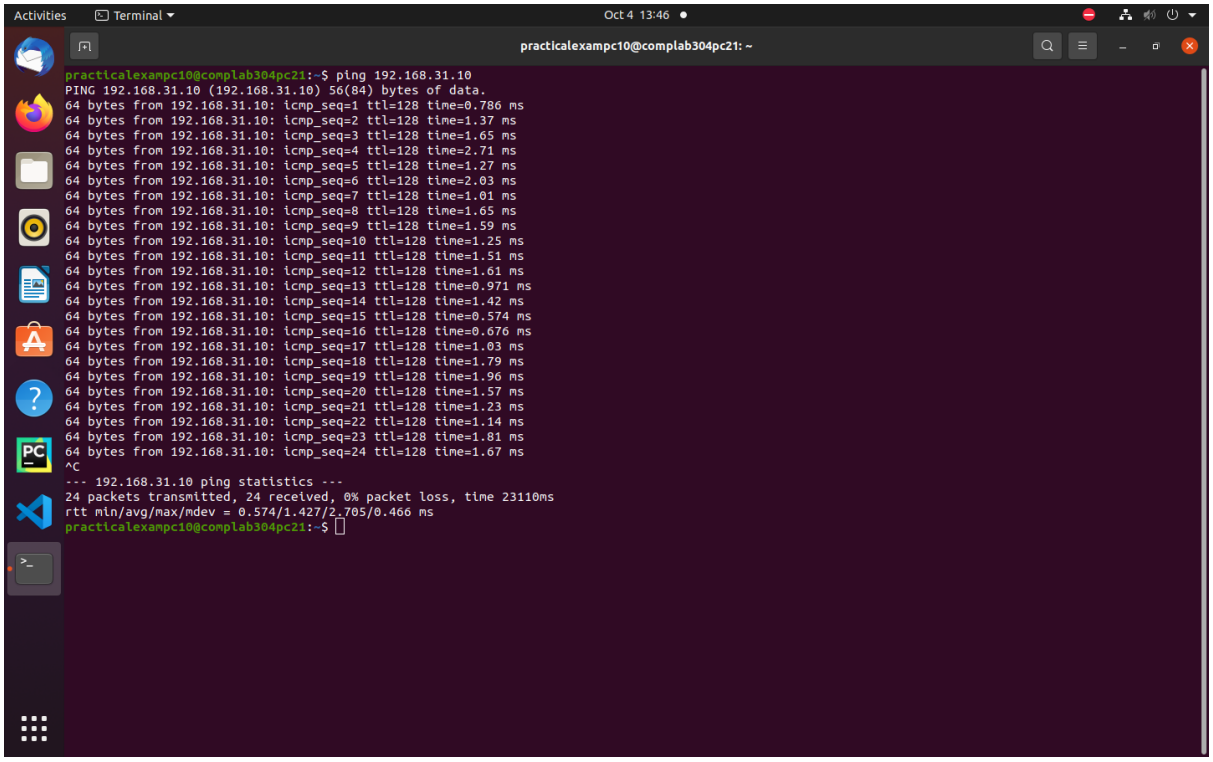
For e.g.; route -n

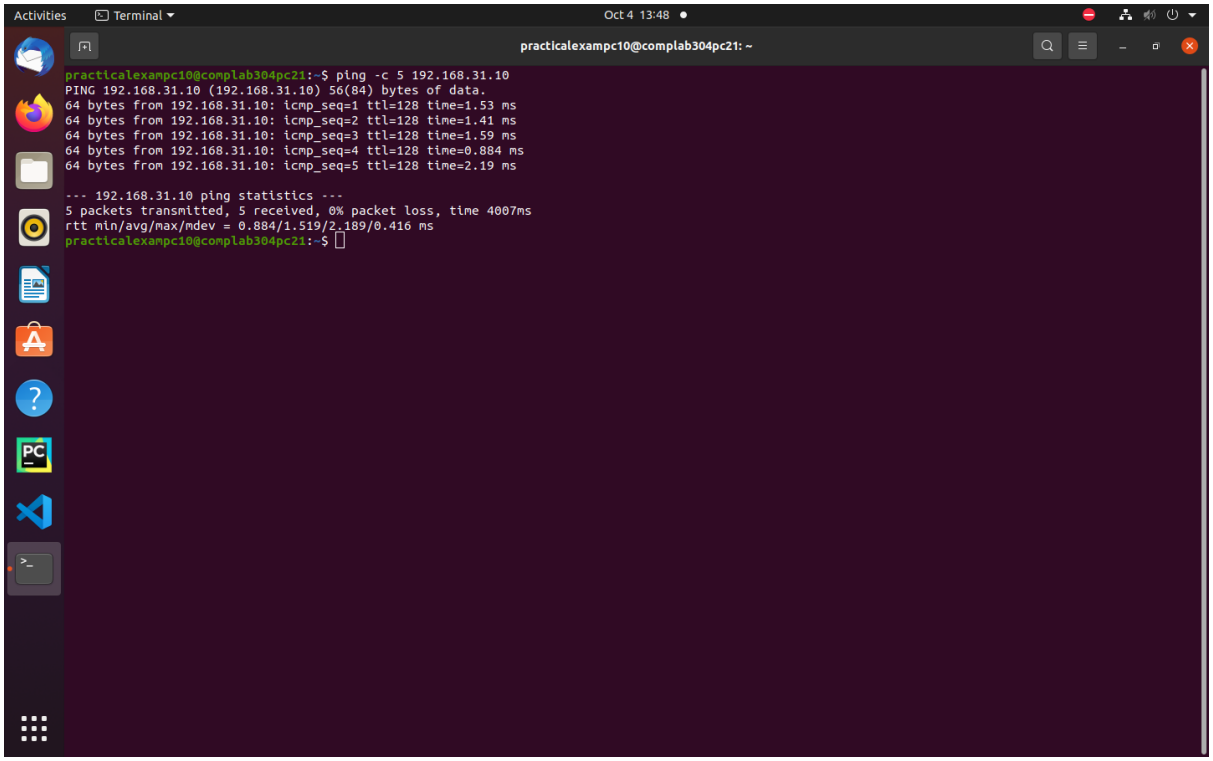
- These commands are essential for diagnosing and managing network-related tasks on a Linux system.

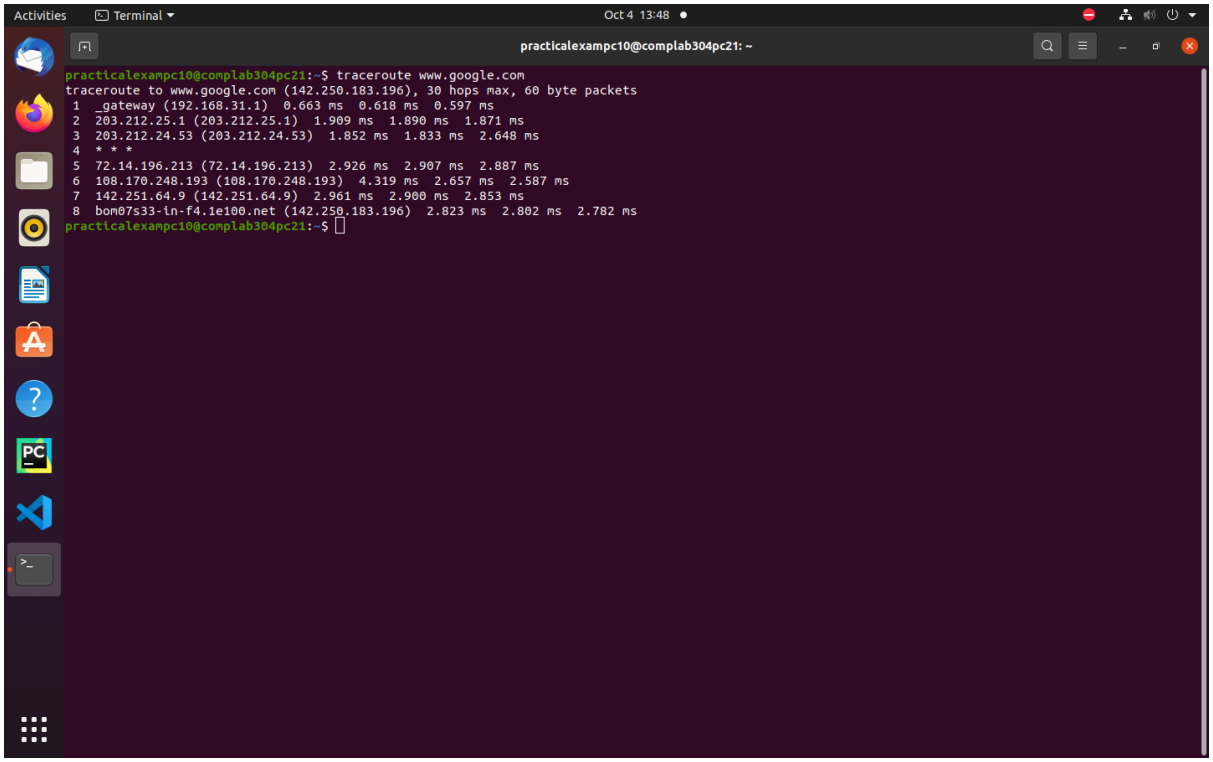


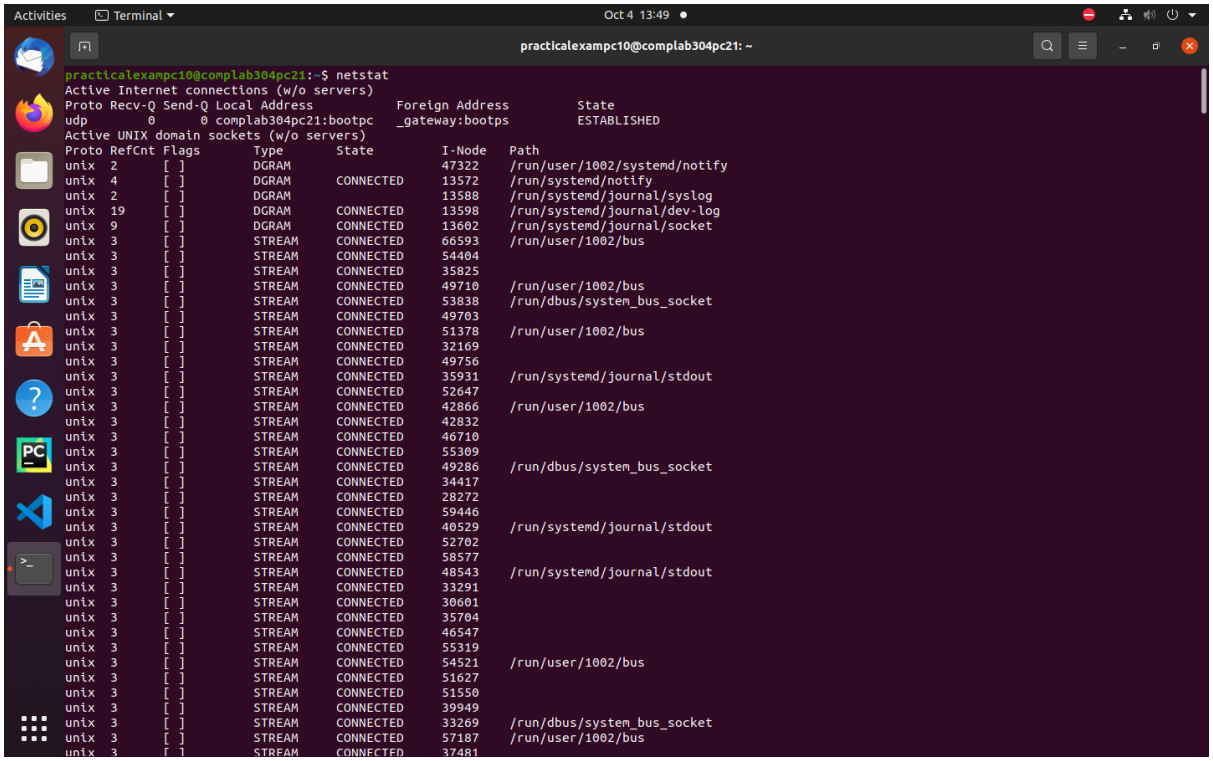
EXP 8

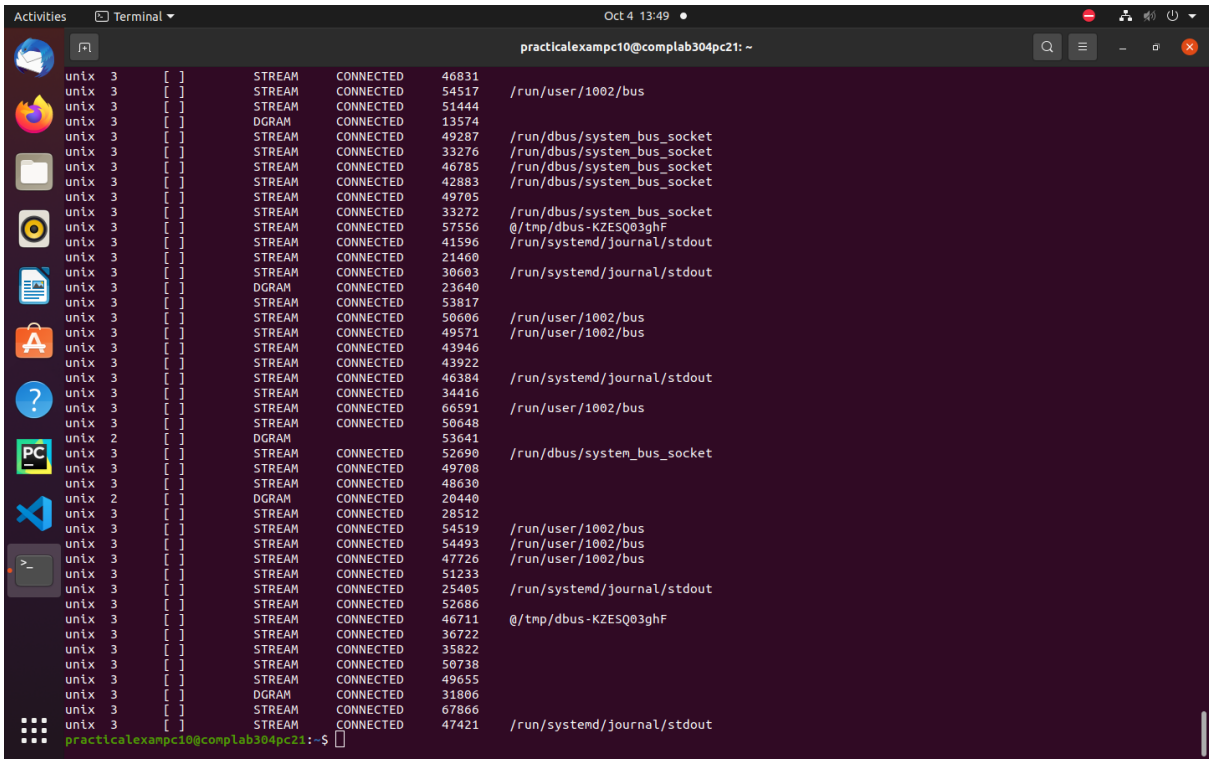

Activities
Terminal
Oct 4 13:38
practicalexampc10@complab304pc21: ~
practicalexampc10@complab304pc21:~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.31.11 netmask 255.255.255.0 broadcast 192.168.31.255
inet6 fe80::6aa4:fd01:d68b:4d4a prefixlen 64 scopeid 0x20<link>
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
practicalexampc10@complab304pc21:~$

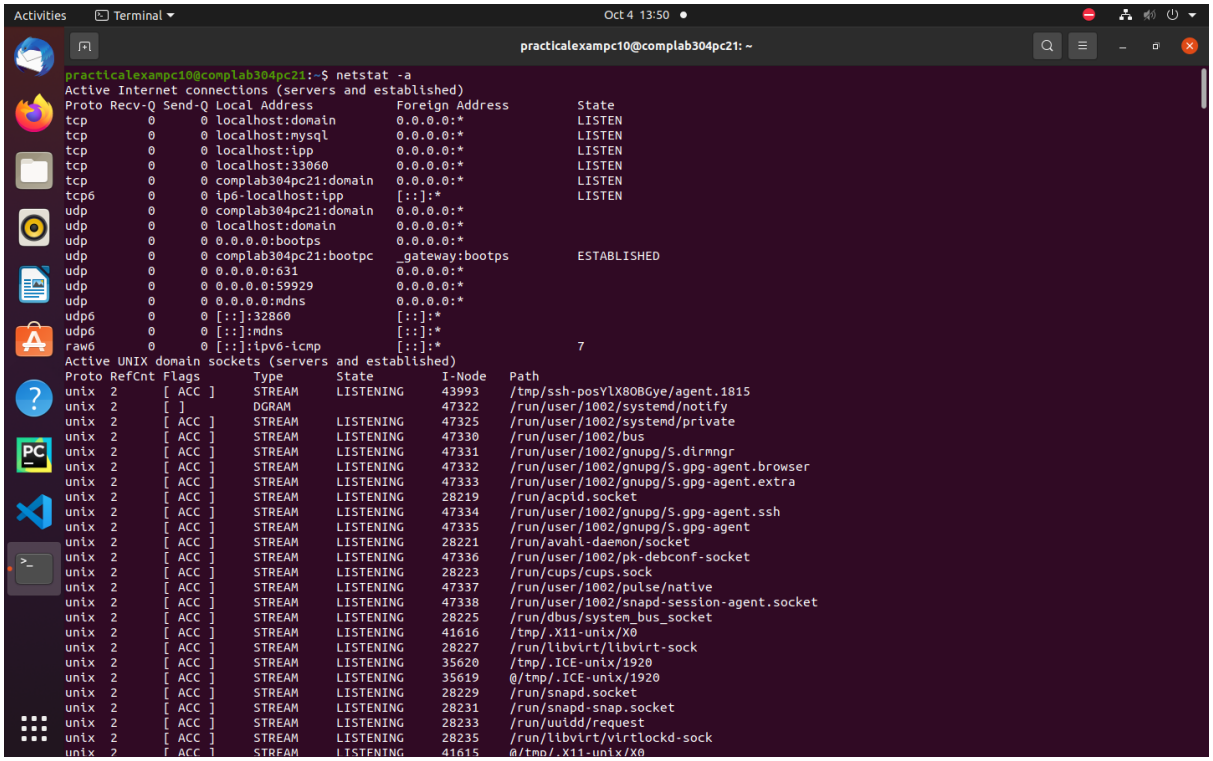

Activities
Terminal
Oct 4 13:46
practicalexampc10@complab304pc21: ~
practicalexampc10@complab304pc21:~$ ping 192.168.31.10
PING 192.168.31.10 (192.168.31.10) 56(84) bytes of data.
64 bytes from 192.168.31.10: icmp_seq=1 ttl=128 time=0.786 ms
64 bytes from 192.168.31.10: icmp_seq=2 ttl=128 time=1.37 ms
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
practicalexampc10@complab304pc21:~$

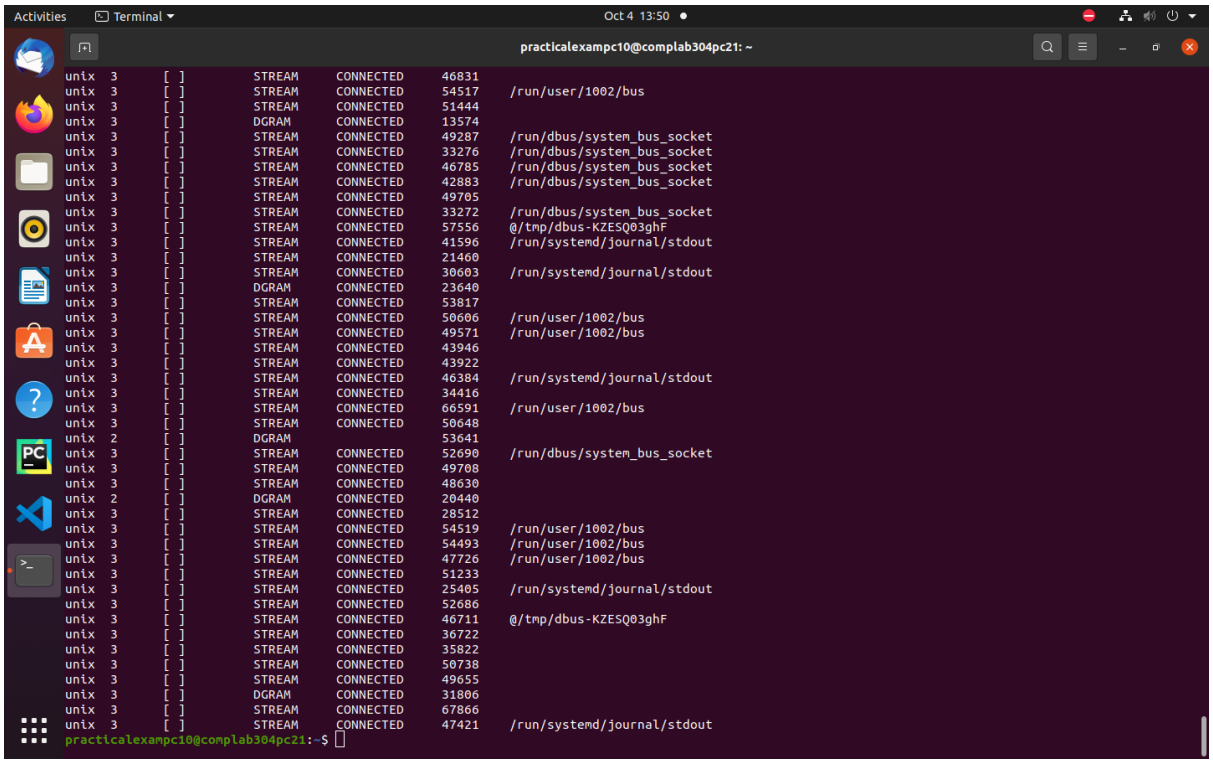

Activities
Terminal
Oct 4 13:48
practicalexampc10@complab304pc21: ~
practicalexampc10@complab304pc21:~$ ping -c 5 192.168.31.10
PING 192.168.31.10 (192.168.31.10) 56(84) bytes of data.
64 bytes from 192.168.31.10: icmp_seq=1 ttl=128 time=1.53 ms
64 bytes from 192.168.31.10: icmp_seq=2 ttl=128 time=1.41 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
practicalexampc10@complab304pc21:~$

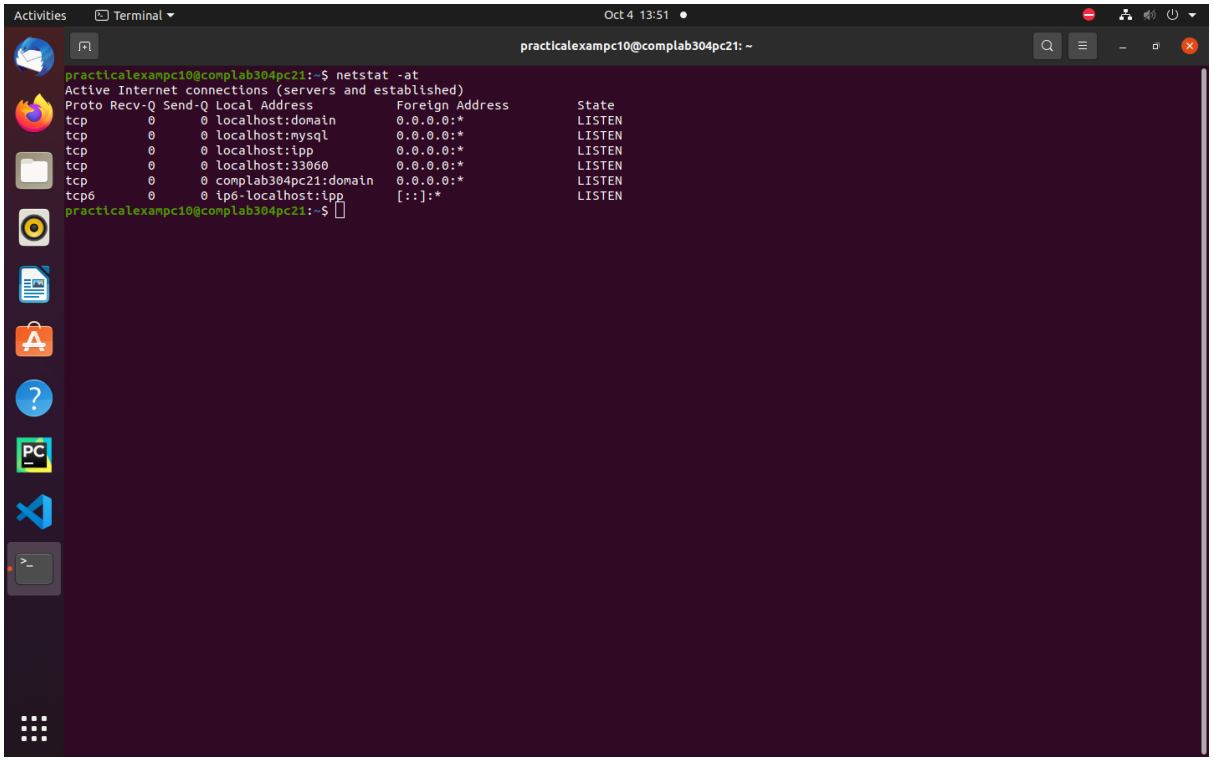

Activities
Terminal
Oct 4 13:48
practicalexampc10@complab304pc21: ~
practicalexampc10@complab304pc21:~$ traceroute www.google.com
traceroute to www.google.com (142.250.183.196), 30 hops max, 60 byte packets
1 _gateway (192.168.31.1) 0.663 ms 0.618 ms 0.597 ms
2 203.212.25.1 (203.212.25.1) 1.909 ms 1.890 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 2.900 ms 2.853 ms
8 bom07s33-in-f4.1e100.net (142.250.183.196) 2.823 ms 2.802 ms 2.782 ms
practicalexampc10@complab304pc21:~$

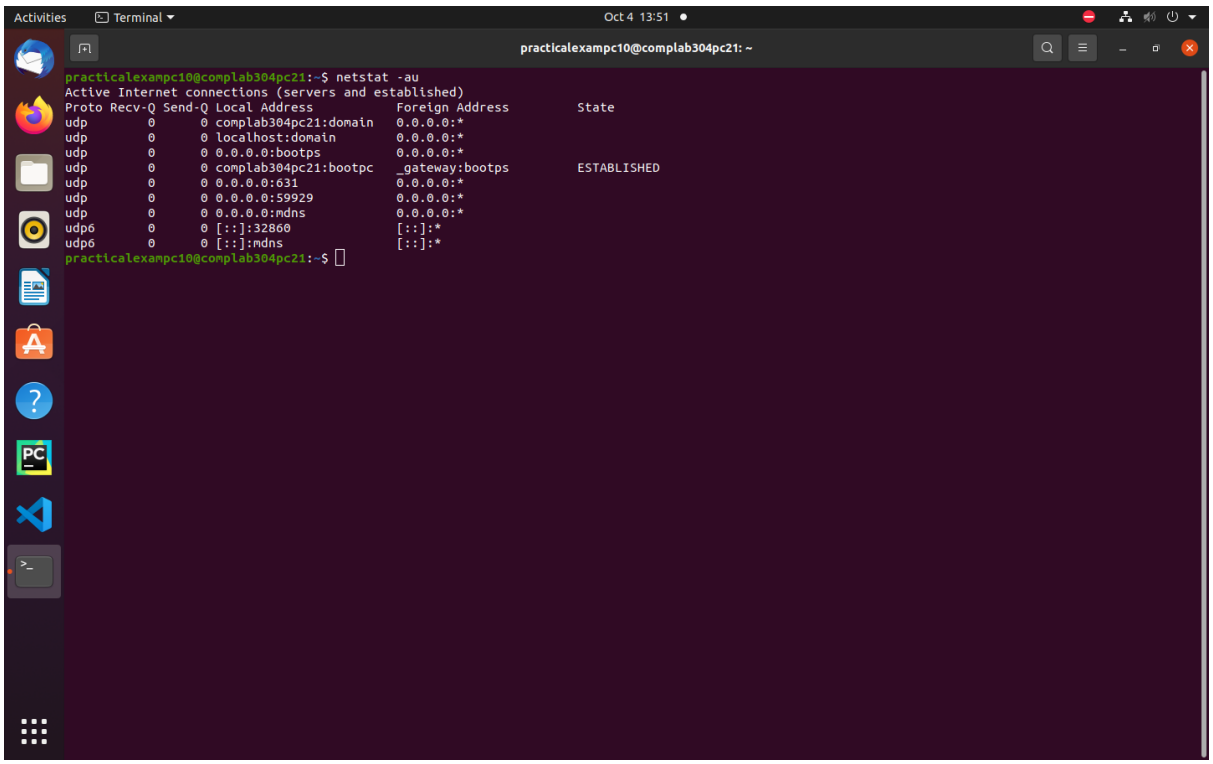

Activities
Terminal
Oct 4 13:49
practicalexampc10@complab304pc21: ~
practicalexampc10@complab304pc21:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
udp 0 0 complab304pc21:bootpc _gateway:bootps ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags Type State I-Node Path
unix 2 [ ] DGRAM 47322 /run/user/1002/systemd/notify
unix 4 [ ] DGRAM CONNECTED 13572 /run/systemd/notify
unix 2 [ ] DGRAM 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
unix 3 [ ] STREAM CONNECTED 35825
unix 3 [ ] STREAM CONNECTED 49710 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 53838 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 49703
unix 3 [ ] STREAM CONNECTED 51378 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 32169
unix 3 [ ] STREAM CONNECTED 49756
unix 3 [ ] STREAM CONNECTED 35931 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 52647
unix 3 [ ] STREAM CONNECTED 42866 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 42832
unix 3 [ ] STREAM CONNECTED 46710
unix 3 [ ] STREAM CONNECTED 55309
unix 3 [ ] STREAM CONNECTED 49286 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 34417
unix 3 [ ] STREAM CONNECTED 28272
unix 3 [ ] STREAM CONNECTED 59446
unix 3 [ ] STREAM CONNECTED 40529 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 52702
unix 3 [ ] STREAM CONNECTED 58577
unix 3 [ ] STREAM CONNECTED 48543 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 33291
unix 3 [ ] STREAM CONNECTED 30601
unix 3 [ ] STREAM CONNECTED 35704
unix 3 [ ] STREAM CONNECTED 46547
unix 3 [ ] STREAM CONNECTED 55319
unix 3 [ ] STREAM CONNECTED 54521 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 51627
unix 3 [ ] STREAM CONNECTED 51550
unix 3 [ ] STREAM CONNECTED 39949
unix 3 [ ] STREAM CONNECTED 33269 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 57187 /run/user/1002/bus

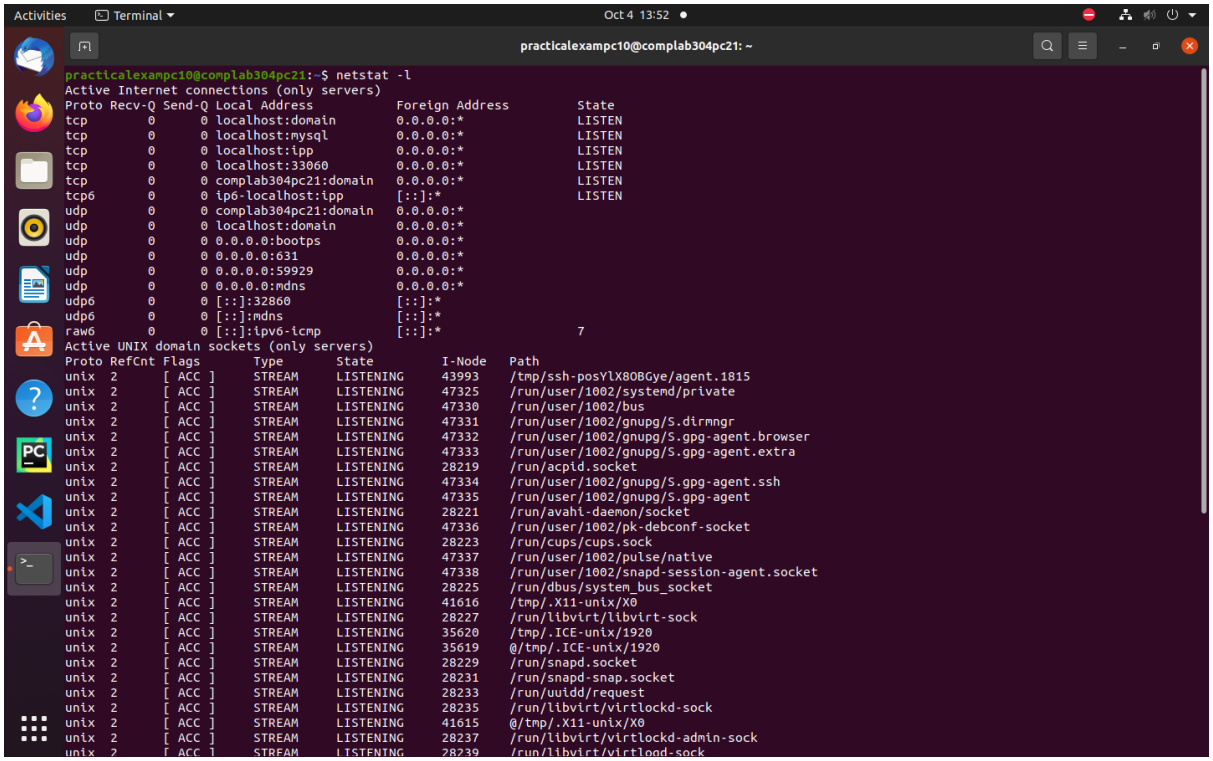

Activities
Terminal
Oct 4 13:49
practicalexampc10@complab304pc21: ~
unix 3 [ ] STREAM CONNECTED 46831
unix 3 [ ] STREAM CONNECTED 54517 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 51444
unix 3 [ ] DGRAM CONNECTED 13574
unix 3 [ ] STREAM CONNECTED 49287 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 33276 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 46785 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 42883 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 49705
unix 3 [ ] STREAM CONNECTED 33272 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 57556 @/tmp/dbus-KZESQ03ghF
unix 3 [ ] STREAM CONNECTED 41596 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 21460
unix 3 [ ] STREAM CONNECTED 30603 /run/systemd/journal/stdout
unix 3 [ ] DGRAM CONNECTED 23640
unix 3 [ ] STREAM CONNECTED 53817
unix 3 [ ] STREAM CONNECTED 50606 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 49571 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 43946
unix 3 [ ] STREAM CONNECTED 43922
unix 3 [ ] STREAM CONNECTED 46384 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 34416
unix 3 [ ] STREAM CONNECTED 66591 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 50648
unix 2 [ ] DGRAM 53641
unix 3 [ ] STREAM CONNECTED 52690 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 49708
unix 3 [ ] STREAM CONNECTED 48630
unix 2 [ ] DGRAM CONNECTED 20440
unix 3 [ ] STREAM CONNECTED 28512
unix 3 [ ] STREAM CONNECTED 54519 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 54493 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 47726 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 51233
unix 3 [ ] STREAM CONNECTED 25405 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 52686
unix 3 [ ] STREAM CONNECTED 46711 @/tmp/dbus-KZESQ03ghF
unix 3 [ ] STREAM CONNECTED 36722
unix 3 [ ] STREAM CONNECTED 35822
unix 3 [ ] STREAM CONNECTED 50738
unix 3 [ ] STREAM CONNECTED 49655
unix 3 [ ] DGRAM CONNECTED 31806
unix 3 [ ] STREAM CONNECTED 67866
unix 3 [ ] STREAM CONNECTED 47421 /run/systemd/journal/stdout
practicalexampc10@complab304pc21:~$

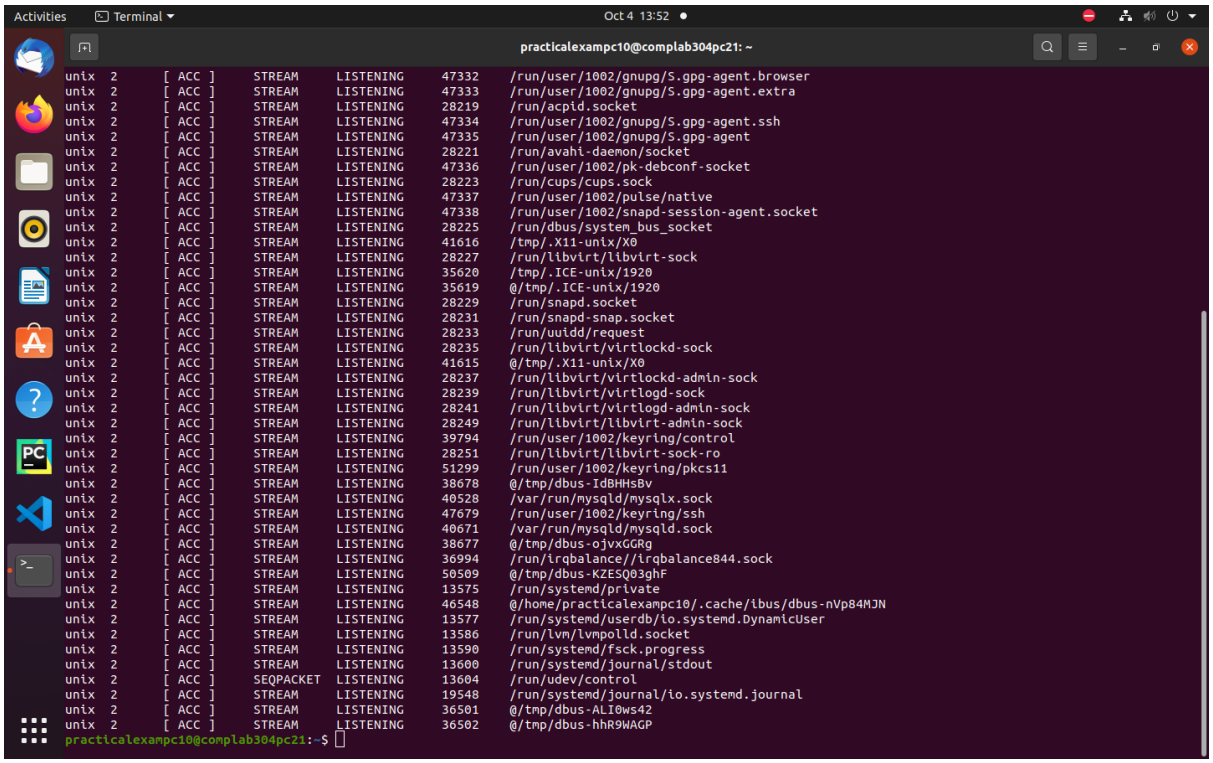

Activities
Terminal
Oct 4 13:50
practicalexampc10@complab304pc21: ~
practicalexampc10@complab304pc21:~$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 localhost:domain 0.0.0.0:* LISTEN
tcp 0 0 localhost:mysql 0.0.0.0:* LISTEN
tcp 0 0 localhost:ipp 0.0.0.0:* LISTEN
tcp 0 0 localhost:33060 0.0.0.0:* LISTEN
tcp 0 0 complab304pc21:domain 0.0.0.0:* LISTEN
tcp6 0 0 ip6-localhost:ipp [::]:* LISTEN
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:mdns 0.0.0.0:*
udp6 0 0 [::]:32860 [::]:*
udp6 0 0 [::]:mdns [::]:*
raw6 0 0 [::]:ipv6-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-posYlX8OBGye/agent.1815
unix 2 [ ] DGRAM 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.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/uuidd/request
unix 2 [ACC] STREAM LISTENING 28235 /run/libvirt/virtlockd-sock

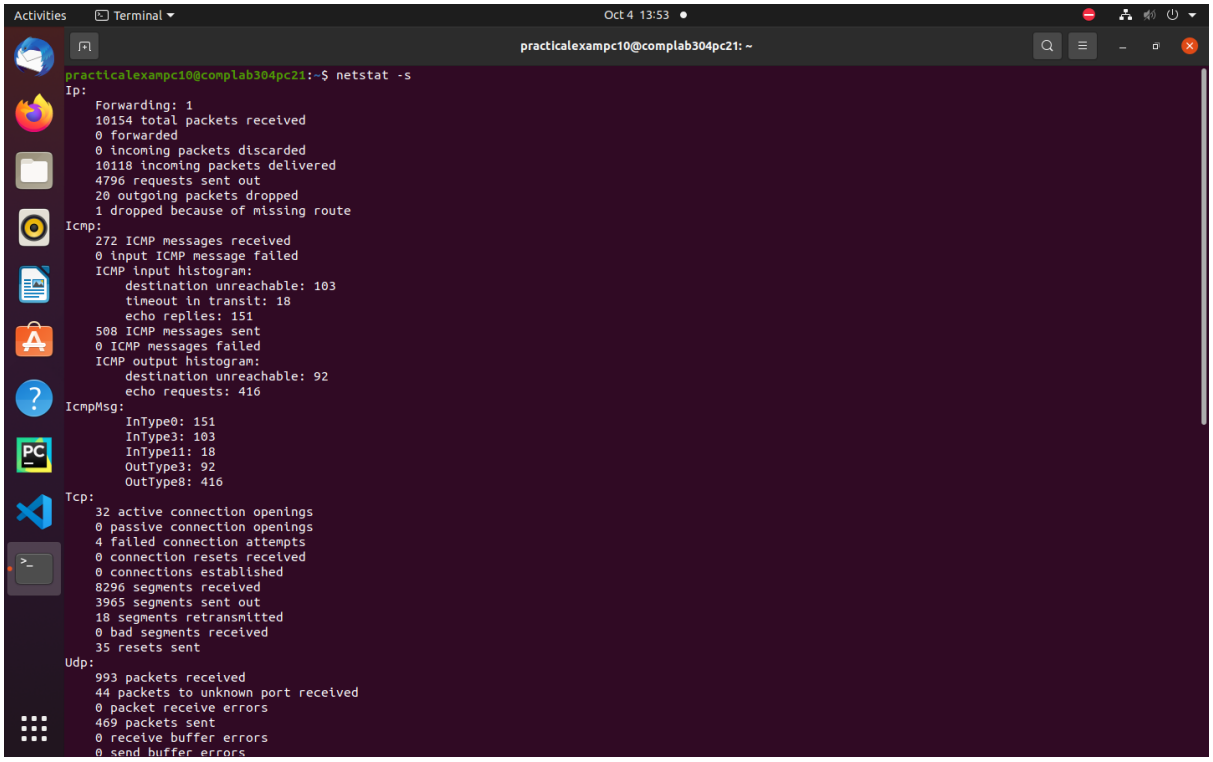

Activities
Terminal
Oct 4 13:50
practicalexampc10@complab304pc21: ~
unix 3 [ ] STREAM CONNECTED 46831
unix 3 [ ] STREAM CONNECTED 54517 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 51444
unix 3 [ ] DGRAM CONNECTED 13574
unix 3 [ ] STREAM CONNECTED 49287 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 33276 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 46785 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 42883 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 49705
unix 3 [ ] STREAM CONNECTED 33272 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 57556 @/tmp/dbus-KZESQ03ghF
unix 3 [ ] STREAM CONNECTED 41596 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 21460
unix 3 [ ] STREAM CONNECTED 30603 /run/systemd/journal/stdout
unix 3 [ ] DGRAM CONNECTED 23640
unix 3 [ ] STREAM CONNECTED 53817
unix 3 [ ] STREAM CONNECTED 50606 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 49571 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 43946
unix 3 [ ] STREAM CONNECTED 43922
unix 3 [ ] STREAM CONNECTED 46384 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 34416
unix 3 [ ] STREAM CONNECTED 66591 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 50648
unix 2 [ ] DGRAM 53641
unix 3 [ ] STREAM CONNECTED 52690 /run/dbus/system_bus_socket
unix 3 [ ] STREAM CONNECTED 49708
unix 3 [ ] STREAM CONNECTED 48630
unix 2 [ ] DGRAM CONNECTED 20440
unix 3 [ ] STREAM CONNECTED 28512
unix 3 [ ] STREAM CONNECTED 54519 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 54493 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 47726 /run/user/1002/bus
unix 3 [ ] STREAM CONNECTED 51233
unix 3 [ ] STREAM CONNECTED 25405 /run/systemd/journal/stdout
unix 3 [ ] STREAM CONNECTED 52686
unix 3 [ ] STREAM CONNECTED 46711 @/tmp/dbus-KZESQ03ghF
unix 3 [ ] STREAM CONNECTED 36722
unix 3 [ ] STREAM CONNECTED 35822
unix 3 [ ] STREAM CONNECTED 50738
unix 3 [ ] STREAM CONNECTED 49655
unix 3 [ ] DGRAM CONNECTED 31806
unix 3 [ ] STREAM CONNECTED 67866
unix 3 [ ] STREAM CONNECTED 47421 /run/systemd/journal/stdout
practicalexampc10@complab304pc21:~$

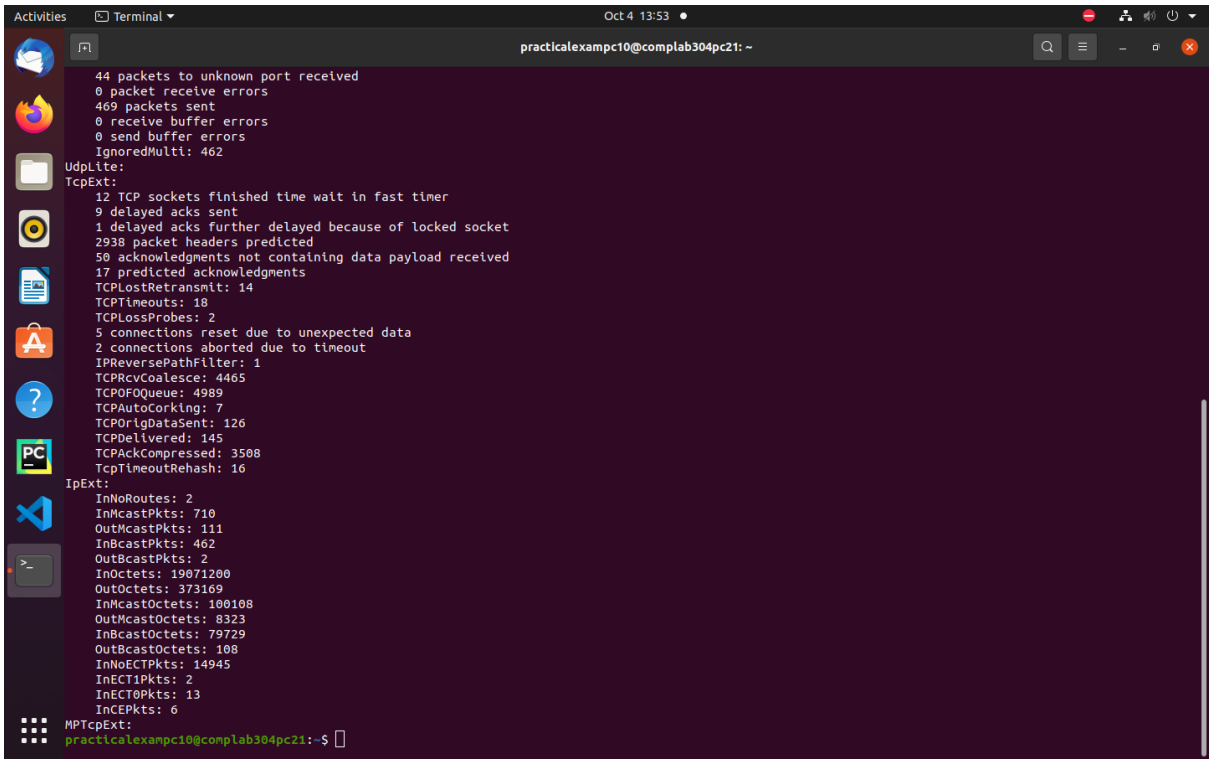

Activities
Terminal
Oct 4 13:51
practicalexampc10@complab304pc21: ~
practicalexampc10@complab304pc21:~$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 localhost:domain 0.0.0.0:* LISTEN
tcp 0 0 localhost:mysql 0.0.0.0:* LISTEN
tcp 0 0 localhost:ipp 0.0.0.0:* LISTEN
tcp 0 0 localhost:33060 0.0.0.0:* LISTEN
tcp 0 0 complab304pc21:domain 0.0.0.0:* LISTEN
tcp6 0 0 ip6-localhost:ipp [::]:* LISTEN
practicalexampc10@complab304pc21:~$

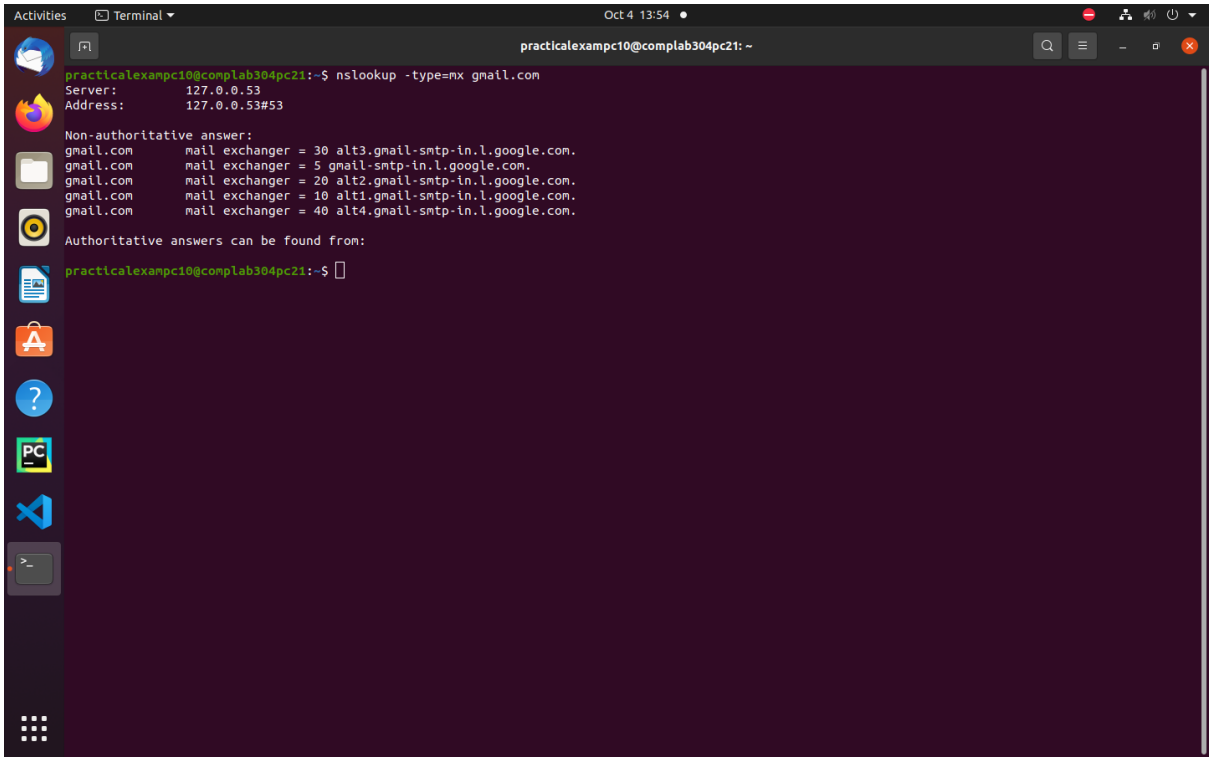

Activities
Terminal
Oct 4 13:51
practicalexampc10@complab304pc21: ~
practicalexampc10@complab304pc21:~$ netstat -au
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
udp 0 0 complab304pc21:domain 0.0.0.0:*
udp 0 0 localhost:domain 0.0.0.0:*
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:mdns 0.0.0.0:*
udp6 0 0 [::]:32860 [::]:*
udp6 0 0 [::]:mdns [::]:*
practicalexampc10@complab304pc21:~$

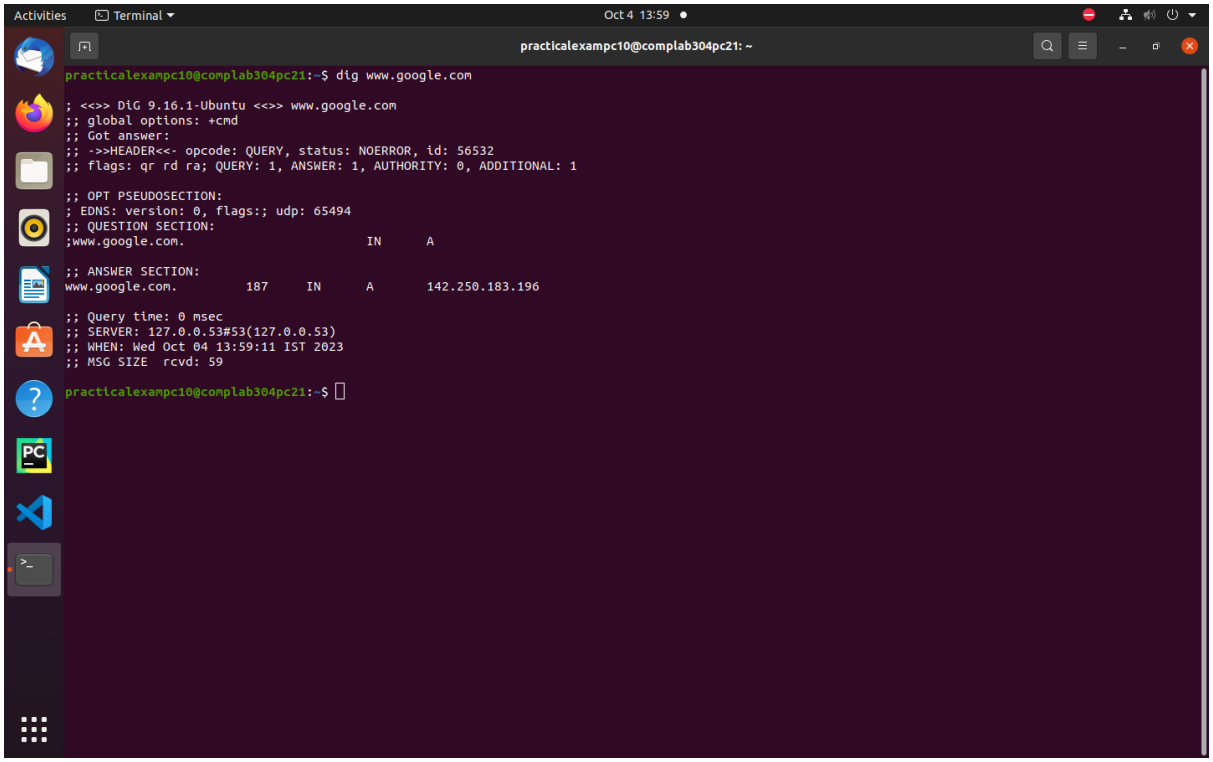

Activities
Terminal
Oct 4 13:52
practicalexampc10@complab304pc21: ~
practicalexampc10@complab304pc21:~$ netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 localhost:domain 0.0.0.0:* LISTEN
tcp 0 0 localhost:mysql 0.0.0.0:* LISTEN
tcp 0 0 localhost:ipp 0.0.0.0:* LISTEN
tcp 0 0 localhost:33060 0.0.0.0:* LISTEN
tcp 0 0 complab304pc21:domain 0.0.0.0:* LISTEN
tcp6 0 0 ip6-localhost:ipp [::]:* LISTEN
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-posYlX8OBGye/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/uuidd/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

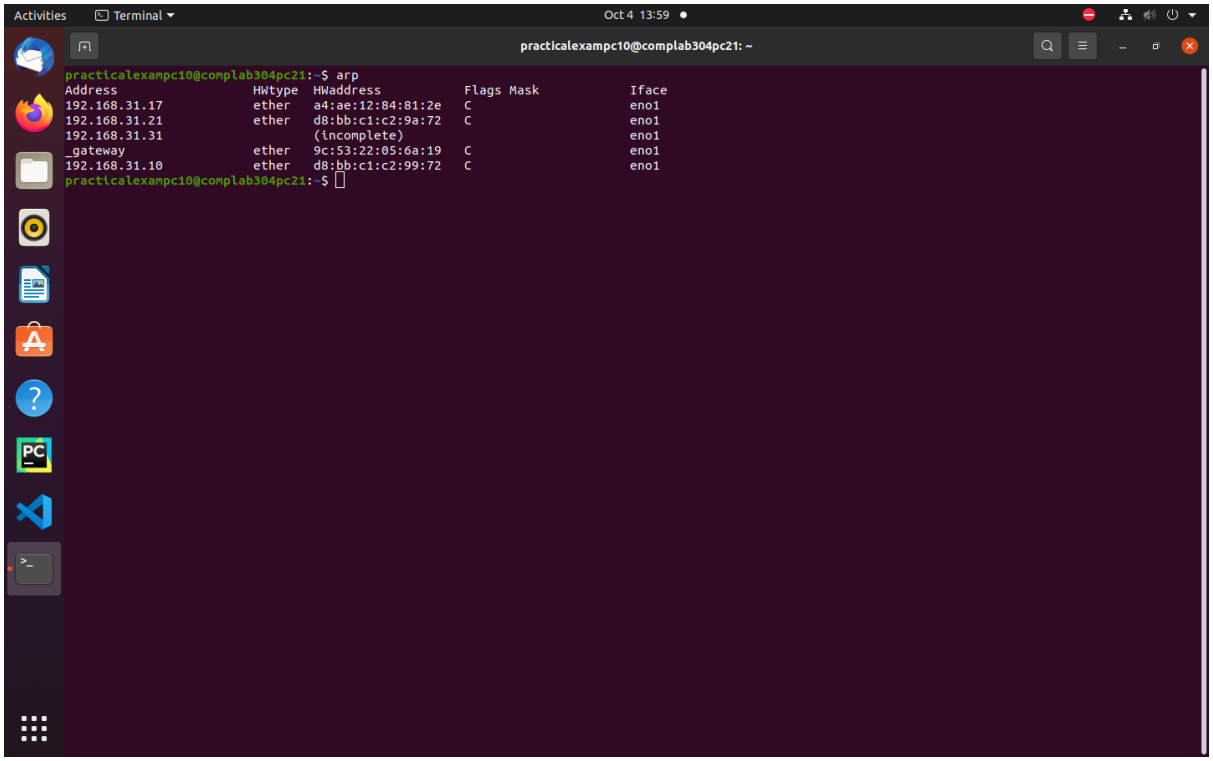

Activities
Terminal
Oct 4 13:52
practicalexampc10@complab304pc21: ~
unix 2 [ACC] STREAM LISTENING 47332 /run/user/1002/gnupg/S.gpg-agent.browser
unix 2 [ACC] STREAM LISTENING 47333 /run/user/1002/gnupg/S.gpg-agent.extra
unix 2 [ACC] STREAM LISTENING 28219 /run/acpid.socket
unix 2 [ACC] STREAM LISTENING 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/uuidd/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/virtlogd-sock
unix 2 [ACC] STREAM LISTENING 28241 /run/libvirt/virtlogd-admin-sock
unix 2 [ACC] STREAM LISTENING 28249 /run/libvirt/libvirt-admin-sock
unix 2 [ACC] STREAM LISTENING 39794 /run/user/1002/keyring/control
unix 2 [ACC] STREAM LISTENING 28251 /run/libvirt/libvirt-sock-ro
unix 2 [ACC] STREAM LISTENING 51299 /run/user/1002/keyring/pkcs11
unix 2 [ACC] STREAM LISTENING 38678 @/tmp/dbus-IdBHHsBv
unix 2 [ACC] STREAM LISTENING 40528 /var/run/mysqld/mysqlx.sock
unix 2 [ACC] STREAM LISTENING 47679 /run/user/1002/keyring/ssh
unix 2 [ACC] STREAM LISTENING 40671 /var/run/mysqld/mysqld.sock
unix 2 [ACC] STREAM LISTENING 38677 @/tmp/dbus-ojvxGGRg
unix 2 [ACC] STREAM LISTENING 36994 /run/irqbalance//irqbalance844.sock
unix 2 [ACC] STREAM LISTENING 50509 @/tmp/dbus-KZESQ03ghF
unix 2 [ACC] STREAM LISTENING 13575 /run/systemd/private
unix 2 [ACC] STREAM LISTENING 46548 @/home/practicalexampc10/.cache/ibus/dbus-nVp84MJN
unix 2 [ACC] STREAM LISTENING 13577 /run/systemd/userdb/io.systemd.DynamicUser
unix 2 [ACC] STREAM LISTENING 13586 /run/lvm/lvmpolld.socket
unix 2 [ACC] STREAM LISTENING 13590 /run/systemd/fsck.progress
unix 2 [ACC] STREAM LISTENING 13600 /run/systemd/journal/stdout
unix 2 [ACC] SEQPACKET LISTENING 13604 /run/udev/control
unix 2 [ACC] STREAM LISTENING 19548 /run/systemd/journal/io.systemd.journal
unix 2 [ACC] STREAM LISTENING 36501 @/tmp/dbus-ALI0ws42
unix 2 [ACC] STREAM LISTENING 36502 @/tmp/dbus-hhR9WAGP
practicalexampc10@complab304pc21:~$

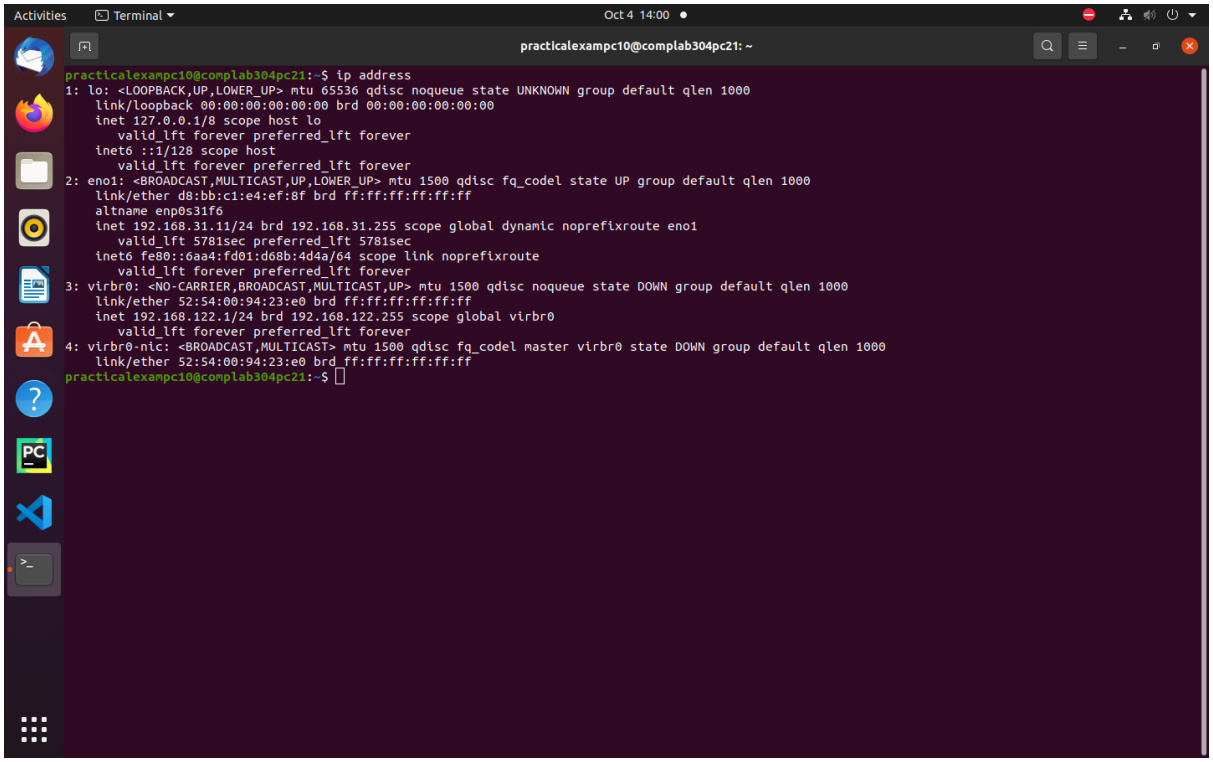

Activities
Terminal
Oct 4 13:53
practicalexampc10@complab304pc21: ~
practicalexampc10@complab304pc21:~$ netstat -s
Ip:
Forwarding: 1
10154 total packets received
0 forwarded
0 incoming packets discarded
10118 incoming packets delivered
4796 requests sent out
20 outgoing packets dropped
1 dropped because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


Activities
Terminal
Oct 4 13:53
practicalexampc10@complab304pc21: ~
44 packets to unknown port received
0 packet receive errors
469 packets sent
0 receive buffer errors
0 send buffer errors
IgnoredMulti: 462
UdpLite:
TcpExt:
12 TCP sockets finished time wait in fast timer
9 delayed acks sent
1 delayed acks further delayed because of locked socket
2938 packet headers predicted
50 acknowledgments not containing data payload received
17 predicted acknowledgments
TCPLostRetransmit: 14
TCPTimeouts: 18
TCPLossProbes: 2
5 connections reset due to unexpected data
2 connections aborted due to timeout
IPReversePathFilter: 1
TCPRcvCoalesce: 4465
TCPOFOQueue: 4989
TCPAutoCorking: 7
TCPOrigDataSent: 126
TCPDelivered: 145
TCPAckCompressed: 3508
TcpTimeoutRehash: 16
IpExt:
InNoRoutes: 2
InMcastPkts: 710
OutMcastPkts: 111
InBcastPkts: 462
OutBcastPkts: 2
InOctets: 19071200
OutOctets: 373169
InMcastOctets: 100108
OutMcastOctets: 8323
InBcastOctets: 79729
OutBcastOctets: 108
InNoECTPkts: 14945
InECT1Pkts: 2
InECT0Pkts: 13
InCEPkts: 6
MPTcpExt:
practicalexampc10@complab304pc21:~$


Activities
Terminal
Oct 4 13:54
practicalexampc10@complab304pc21: ~
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:~$


Activities
Terminal
Oct 4 13:59
practicalexampc10@complab304pc21: ~
practicalexampc10@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
practicalexampc10@complab304pc21:~$


Activities
Terminal
Oct 4 13:59
practicalexampc10@complab304pc21: ~
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) eno1
_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:~$


Activities
Terminal
Oct 4 14:00
practicalexampc10@complab304pc21: ~
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:~$

Experiment No. 9

Aim : Use shark to understand the operation of TCP/IP layers :

- Ethernet Layer : Frame header, Frame size, etc.
- Data link Layer : MAC address, ARP (IP and MAC address binding)
- Network Layer : IP Packet (header, Configuration), ICMP (Query and Echo)
- Transport Layer : TCP Ports, TCP handshake segments, etc.

Application Layer : DHCP, FTP, HTTP header formats.

Theory :

- Wireshark, a network analysis tool mainly known as etherreal capwes packets in real time and display them in the human readable formats.
- Wireshark includes filters, colours, coding and other features that let you keep netwak traffic and inspect individual packet.
- **Capturing Packets**

⇒ After downloding and installing wireshark you can launch it and double click the capture traffic on your wireless network, click your wireless interface you can configure advanced features by clicking capture option but is not necessary for now.

- As soon as you click on interface name, you will see the packet shark appear in real time.
- Wireshark capture each packet sent to or from your server.

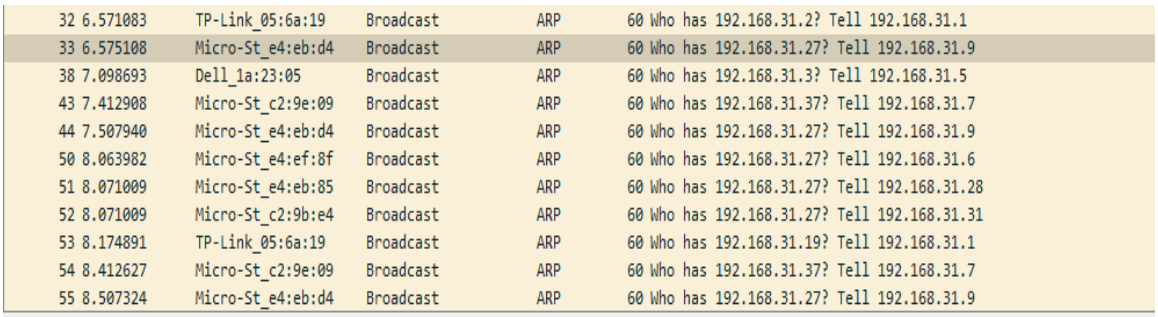
- Colour Coding

⇒ You will probably see packet highlighted in a variety of different colours.
- Wireshark that type of a glance uses colours to help you identify that type of glance.
- By default light purple is TCP traffic light blue is USP traffic and black identify packets with errors.
- For e.g., they could have blue delivered out of order. To view exactly what colour mean, click view colouring rules, you can also customize and modify colouring rules.

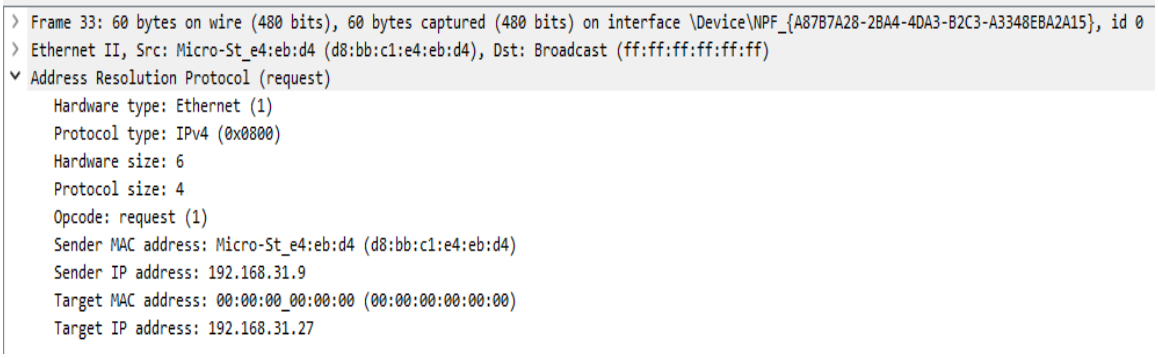
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Colours in Wireshark	Packet type
1) Light purple	TCP
2) Light blue	UDP
3) Black	Packets with errors
4) light Green	HTTP traffic
5) light yellow	windows-specific traffic handling server message blocks.
6) Dark yellow	Routing
7) Dark Grey	TCP, SYN, FIN & Ack traffic

19/10/23

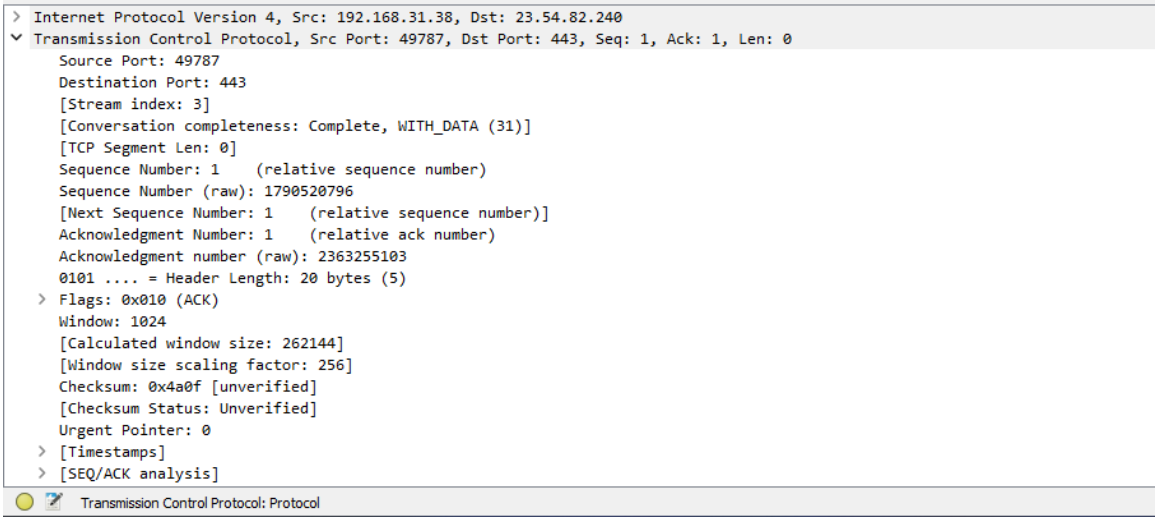


32	6.571083	TP-Link_05:6a:19	Broadcast	ARP	60 Who has 192.168.31.2? Tell 192.168.31.1
33	6.575108	Micro-St_e4:eb:d4	Broadcast	ARP	60 Who has 192.168.31.27? Tell 192.168.31.9
38	7.098693	Dell_1a:23:05	Broadcast	ARP	60 Who has 192.168.31.3? Tell 192.168.31.5
43	7.412908	Micro-St_c2:9e:09	Broadcast	ARP	60 Who has 192.168.31.37? Tell 192.168.31.7
44	7.507940	Micro-St_e4:eb:d4	Broadcast	ARP	60 Who has 192.168.31.27? Tell 192.168.31.9
50	8.063982	Micro-St_e4:ef:8f	Broadcast	ARP	60 Who has 192.168.31.27? Tell 192.168.31.6
51	8.071009	Micro-St_e4:eb:85	Broadcast	ARP	60 Who has 192.168.31.27? Tell 192.168.31.28
52	8.071009	Micro-St_c2:9b:e4	Broadcast	ARP	60 Who has 192.168.31.27? Tell 192.168.31.31
53	8.174891	TP-Link_05:6a:19	Broadcast	ARP	60 Who has 192.168.31.19? Tell 192.168.31.1
54	8.412627	Micro-St_c2:9e:09	Broadcast	ARP	60 Who has 192.168.31.37? Tell 192.168.31.7
55	8.507324	Micro-St_e4:eb:d4	Broadcast	ARP	60 Who has 192.168.31.27? Tell 192.168.31.9



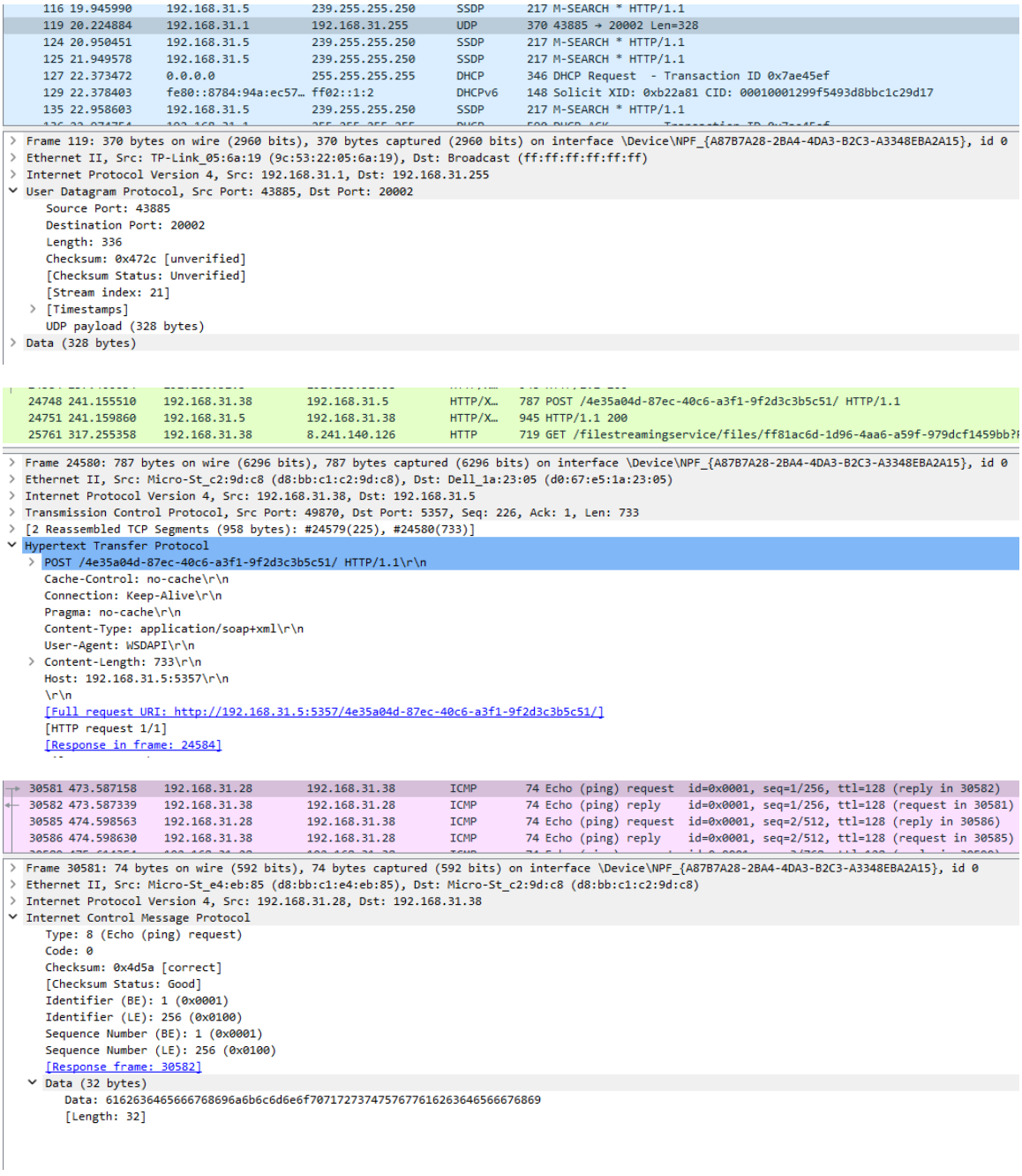
```
> Frame 33: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{A87B7A28-2BA4-4DA3-B2C3-A3348EBA2A15}, id 0
> Ethernet II, Src: Micro-St_e4:eb:d4 (d8:bb:c1:e4:eb:d4), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
v Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Micro-St_e4:eb:d4 (d8:bb:c1:e4:eb:d4)
    Sender IP address: 192.168.31.9
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.31.27
```

167	28.021182	192.168.31.38	23.54.82.240	TCP	54 49787 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
168	28.021628	192.168.31.38	23.54.82.240	TLSv1.3	627 Client Hello
169	28.024939	23.54.82.240	192.168.31.38	TCP	60 443 → 49787 [ACK] Seq=1 Ack=574 Win=64128 Len=0



```
> Internet Protocol Version 4, Src: 192.168.31.38, Dst: 23.54.82.240
v Transmission Control Protocol, Src Port: 49787, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
    Source Port: 49787
    Destination Port: 443
    [Stream index: 3]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 1790520796
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 2363255103
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 1024
    [Calculated window size: 262144]
    [Window size scaling factor: 256]
    Checksum: 0x4a0f [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
```

Transmission Control Protocol: Protocol


116 19.945990 192.168.31.5 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
119 20.224884 192.168.31.1 192.168.31.255 UDP 370 43885 → 20002 Len=328
124 20.950451 192.168.31.5 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
125 21.949578 192.168.31.5 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
127 22.373472 0.0.0.0 255.255.255.255 DHCP 346 DHCP Request - Transaction ID 0x7ae45ef
129 22.378403 fe80::8784:94a:ec57… ff02::1:2 DHCPv6 148 Solicit XID: 0xb22a81 CID: 00010001299f5493d8bbc1c29d17
135 22.958603 192.168.31.5 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
Frame 119: 370 bytes on wire (2960 bits), 370 bytes captured (2960 bits) on interface \Device\NPF_{A87B7A28-2BA4-4DA3-B2C3-A3348EBA2A15}, id 0
Ethernet II, Src: TP-Link_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: 192.168.31.255
User Datagram Protocol, Src Port: 43885, Dst Port: 20002
Source Port: 43885
Destination Port: 20002
Length: 336
Checksum: 0x472c [unverified]
[Checksum Status: Unverified]
[Stream index: 21]
[Timestamps]
UDP payload (328 bytes)
Data (328 bytes)
24748 241.155510 192.168.31.38 192.168.31.5 HTTP/X… 787 POST /4e35a04d-87ec-40c6-a3f1-9f2d3c3b5c51/ HTTP/1.1
24751 241.159860 192.168.31.5 192.168.31.38 HTTP/X… 945 HTTP/1.1 200
25761 317.255358 192.168.31.38 8.241.140.126 HTTP 719 GET /filestreamingservice/files/ff81ac6d-1d96-4aa6-a59f-979dcf1459bb?
Frame 24580: 787 bytes on wire (6296 bits), 787 bytes captured (6296 bits) on interface \Device\NPF_{A87B7A28-2BA4-4DA3-B2C3-A3348EBA2A15}, id 0
Ethernet II, Src: Micro-St_c2:9d:c8 (d8:bb:c1:c2:9d:c8), Dst: Dell_1a:23:05 (d0:67:e5:1a:23:05)
Internet Protocol Version 4, Src: 192.168.31.38, Dst: 192.168.31.5
Transmission Control Protocol, Src Port: 49870, Dst Port: 5357, Seq: 226, Ack: 1, Len: 733
[2 Reassembled TCP Segments (958 bytes): #24579(225), #24580(733)]
Hypertext Transfer Protocol
POST /4e35a04d-87ec-40c6-a3f1-9f2d3c3b5c51/ HTTP/1.1\r\n
Cache-Control: no-cache\r\n
Connection: Keep-Alive\r\n
Pragma: no-cache\r\n
Content-Type: application/soap+xml\r\n
User-Agent: WSDAPI\r\n
Content-Length: 733\r\n
Host: 192.168.31.5:5357\r\n
\r\n
[Full request URI: http://192.168.31.5:5357/4e35a04d-87ec-40c6-a3f1-9f2d3c3b5c51/]
[HTTP request 1/1]
[Response in frame: 24584]
30581 473.587158 192.168.31.28 192.168.31.38 ICMP 74 Echo (ping) request id=0x0001, seq=1/256, ttl=128 (reply in 30582)
30582 473.587339 192.168.31.38 192.168.31.28 ICMP 74 Echo (ping) reply id=0x0001, seq=1/256, ttl=128 (request in 30581)
30585 474.598563 192.168.31.28 192.168.31.38 ICMP 74 Echo (ping) request id=0x0001, seq=2/512, ttl=128 (reply in 30586)
30586 474.598630 192.168.31.38 192.168.31.28 ICMP 74 Echo (ping) reply id=0x0001, seq=2/512, ttl=128 (request in 30585)
Frame 30581: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{A87B7A28-2BA4-4DA3-B2C3-A3348EBA2A15}, id 0
Ethernet II, Src: Micro-St_e4:eb:85 (d8:bb:c1:e4:eb:85), Dst: Micro-St_c2:9d:c8 (d8:bb:c1:c2:9d:c8)
Internet Protocol Version 4, Src: 192.168.31.28, Dst: 192.168.31.38
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0x4d5a [correct]
[Checksum Status: Good]
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence Number (BE): 1 (0x0001)
Sequence Number (LE): 256 (0x0100)
[Response frame: 30582]
Data (32 bytes)
Data: 6162636465666768696a6b6c6d6e6f707172737475767761626364656667686 9
[Length: 32]

```
Ping statistics for 192.168.31.28:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PS C:\Users\complab301pc13> ping 192.168.31.31

Pinging 192.168.31.31 with 32 bytes of data:
Reply from 192.168.31.31: bytes=32 time=2ms TTL=128
Reply from 192.168.31.31: bytes=32 time=2ms TTL=128
Reply from 192.168.31.31: bytes=32 time=3ms TTL=128
Reply from 192.168.31.31: bytes=32 time=2ms TTL=128

Ping statistics for 192.168.31.31:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 3ms, Average = 2ms
PS C:\Users\complab301pc13>
```

No.	Time	Source	Destination	Protocol	Length	Info
29225	364.936033	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.251 for any sources
29234	365.030537	192.168.31.20	224.0.0.22	IGMPv3	60	Membership Report / Join group 239.255.255.250 for any sources

```
> Frame 29225: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{A87B7A28-2BA4-4DA3-B2C3-A3348EBA2A15}, id 0
> Ethernet II, Src: Micro-St_c2:9b:e4 (d8:bb:c1:c2:9b:e4), Dst: IPv4mcast_16 (01:00:5e:00:00:16)
> Internet Protocol Version 4, Src: 192.168.31.31, Dst: 224.0.0.22
v Internet Group Management Protocol
    [IGMP Version: 3]
    Type: Membership Report (0x22)
    Reserved: 00
    Checksum: 0xfb02 [correct]
    [Checksum Status: Good]
    Reserved: 0000
    Num Group Records: 1
  v Group Record : 224.0.0.251  Mode Is Exclude
      Record Type: Mode Is Exclude (2)
      Aux Data Len: 0
      Num Src: 0
      Multicast Address: 224.0.0.251
```

dhcp

No.	Time	Source	Destination	Protocol	Length	Info
29018	359.950677	192.168.31.5	255.255.255.255	DHCP	342	DHCP Inform - Transaction ID 0x8950e73e
30013	416.916189	192.168.31.1	255.255.255.255	DHCP	590	DHCP ACK - Transaction ID 0x93b9ceff

```
> Frame 29018: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface \Device\NPF_{A87B7A28-2BA4-4DA3-B2C3-A3348EBA2A15}, id 0
> Ethernet II, Src: Dell_1a:23:05 (d0:67:e5:1a:23:05), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 192.168.31.5, Dst: 255.255.255.255
> User Datagram Protocol, Src Port: 68, Dst Port: 67
v Dynamic Host Configuration Protocol (Inform)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x8950e73e
    Seconds elapsed: 0
  > Bootp flags: 0x0000 (Unicast)
    Client IP address: 192.168.31.5
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: Dell_1a:23:05 (d0:67:e5:1a:23:05)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  > Option: (53) DHCP Message Type (Inform)
  > Option: (61) Client identifier
  > Option: (12) Host Name
  > Option: (60) Vendor class identifier
  > Option: (55) Parameter Request List
  > Option: (255) End
    Padding: 0000000000
```

3	0.743426	192.168.31.31	224.0.0.251	MDNS	85	Standard query 0x0000 PTR _microsoft_mcc._tcp.local, "QM" question
416	29.441569	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources
507	30.434737	192.168.31.31	224.0.0.22	IGMPv3	62	Membership Report / Join group 224.0.0.252 for any sources / Join group 224.0.0.251 for any sources
526	31.443043	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.251 for any sources
618	40.449998	192.168.31.31	224.0.0.22	IGMPv3	62	Membership Report / Join group 224.0.0.252 for any sources / Join group 224.0.0.251 for any sources
632	40.951671	192.168.31.31	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
641	41.448736	192.168.31.31	224.0.0.22	IGMPv3	62	Membership Report / Join group 224.0.0.252 for any sources / Join group 224.0.0.251 for any sources
680	43.952881	192.168.31.31	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
699	45.443200	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 239.255.255.250 for any sources
890	46.959592	192.168.31.31	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
941	49.966735	192.168.31.31	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
946	50.971886	192.168.31.31	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
953	51.980358	192.168.31.31	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
958	52.989679	192.168.31.31	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
7946	112.448477	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.113 for any sources
8071	113.936270	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources
12451	116.948126	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.113 for any sources
13069	117.441341	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 239.255.255.250 for any sources
13289	117.936501	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.251 for any sources
23775	156.436510	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.251 for any sources
23817	157.446721	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.251 for any sources
23880	166.439286	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources
23900	169.981694	192.168.31.31	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
23913	170.397379	192.168.31.31	224.0.0.251	MDNS	85	Standard query 0x0000 PTR _microsoft_mcc._tcp.local, "QU" question
23916	170.436411	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 239.255.255.250 for any sources
23935	170.993740	192.168.31.31	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
23943	171.401019	192.168.31.31	224.0.0.251	MDNS	85	Standard query 0x0000 PTR _microsoft_mcc._tcp.local, "QM" question
23947	171.998585	192.168.31.31	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
23982	173.003518	192.168.31.31	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
24135	181.443385	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 239.255.255.250 for any sources
24153	182.438096	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 239.255.255.250 for any sources
24472	234.447153	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources
24497	235.444095	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources
24532	236.435313	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 239.255.255.250 for any sources
24547	236.949871	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources
24592	237.436695	192.168.31.31	224.0.0.22	IGMPv3	62	Membership Report / Join group 224.0.0.252 for any sources / Join group 224.0.0.113 for any sources
24643	238.436361	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources
24768	241.447938	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.251 for any sources
24769	241.447938	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 239.255.255.250 for any sources
24803	242.443141	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 239.255.255.250 for any sources
24831	243.447883	192.168.31.31	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.113 for any sources

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment No. 10

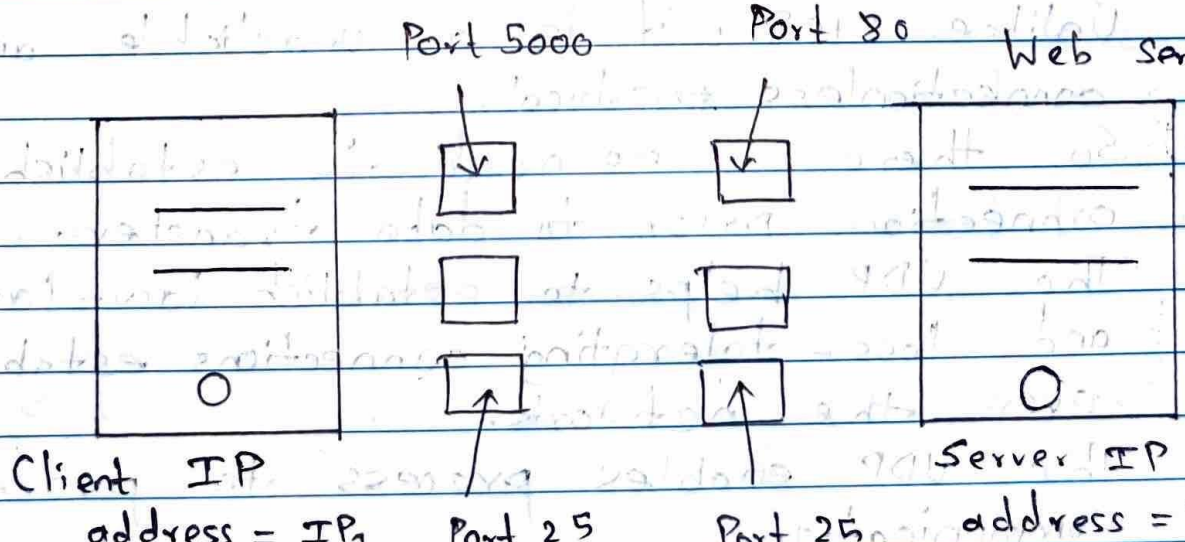
Aim: Socket Programming using TCP or UDP

Theory:

- **Socket**

⇒

- A computer network is a set of devices connected to exchange information and resource such as files, data, images, etc.
- The communication between two or more devices is the communication between the processes present on the different nodes or different computers in a network.
- The communication between different processes on the same nodes or different nodes is done using the concept of a socket.
- A socket is an end-point structure that allows the communication between processes i.e. sending and receiving data over a network.



IP address + Port Number = Socket

- **Transmission Control Protocol (TCP)**

⇒

- TCP is one of the main protocols of the internet protocol suite.
- It lies between the application and network layers which are used in providing reliable delivery services.
- It is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network.
- The IP which establishes the technique for sending data packets between computers, works with TCP.

- **User Datagram Protocol (UDP)**

⇒

- UDP is a transport layer protocol. UDP is a part of IP suite, referred to as UDP / IP suite.
- Unlike TCP, it is an unreliable and connectionless protocol.
- So there is no need to establish a connection prior to data transfer.
- The UDP helps to establish low-latency and loss-tolerating connections establish over the network.
- The UDP enables process to process communication.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

- **Creating Server :**

- To create the server application, we need to create the instance of ServerSocket class.
- Here, we are using 6666 port number for the communication between the client and server.
- The accept() method waits for the client. If the clients connects with the given port number, it returns an instance of socket.

```
ServerSocket ss = ServerSocket(6666);
Socket s = ss.accept();
```

- **Creating client:**

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

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

9/1/19/23.

Code:

(Server.java)

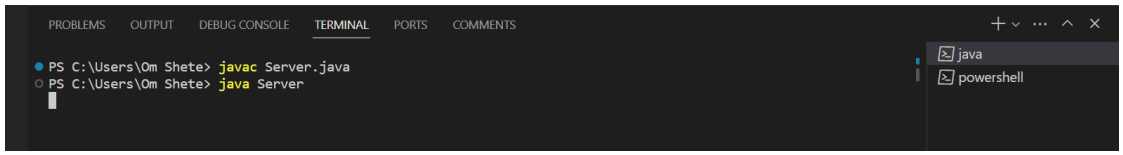
```java
import java.io.*;
import java.net.*;

public class Server {
    public static void main(String[] args) {
        try {
            ServerSocket ss = new ServerSocket(6666);
            Socket s = ss.accept(); // establishes connection
            DataInputStream dis = new
DataInputStream(s.getInputStream());
            String str = (String) dis.readUTF();
            System.out.println("Message= " + str);
            ss.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

(Client.java)

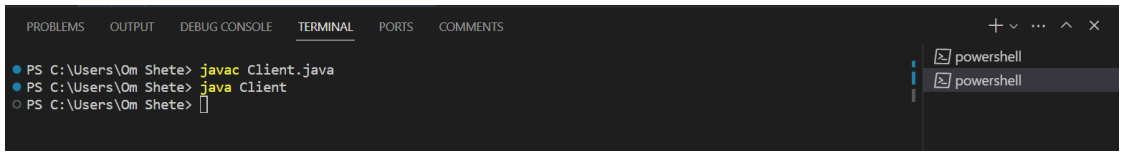
```java
import java.io.*;
import java.net.*;

public class Client {
    public static void main(String[] args) {
        try {
            Socket s = new Socket("localhost", 6666);
            DataOutputStream dout = new
DataOutputStream(s.getOutputStream());
            dout.writeUTF("Hello Server");
            dout.flush();
            dout.close();
            s.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:

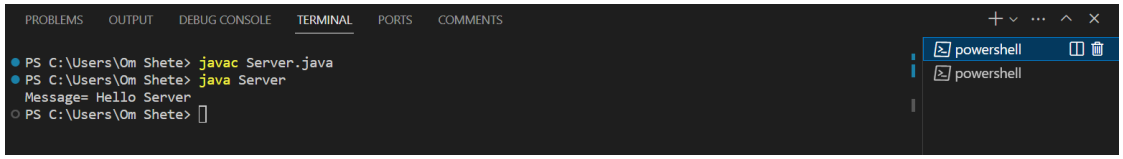
To execute this program open two command prompts and execute each program at each command prompt as displayed in the below figures. First, run the Server.java file in terminal/cmd,



Running Server.java
Then in the new terminal/cmd run the Client.java file



Running Client.java
As soon as you run the Client program a message is sent to the server and displayed in the Server Terminal/CMD as shown below,



Message displayed in Server after running Client

CONCLUSION: So, in this experiment, we have successfully understood the concept of Socket Programming and implemented it using Java Programming

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment No - I

Q.1 Write short note on :

1) Repeater

⇒
- A repeater is a dynamic network device used to reproduce the signals when they transmit over a greater distance so that signal's strength remains equal.
- It can be used to create an Ethernet network.
- A repeater that occurs as the first layer of the OSI layer is the physical layer.
- The significant point to be noted regarding these devices is that they do not strengthen the signal.
- Whenever the signal gets weak, they reproduce it at the actual strength.
- A repeater is a two-port device.

2) Hub

⇒
- A Hub is common connection point, also known as network hub, which is used for connections of devices in a network.
- It works as a central connection for all devices that are connected through a hub.
- The hub has numerous ports.
- IF a packet reaches at one port, it is able to see all the segments of the network due to a packet is copied to the other ports.

- Network Hubs are classified as -

1) Active Hub

⇒ These hubs have their own power supply and these hubs are used to clean, increase and transmit the signal using the network.

2) Passive Hub

⇒ These hubs collect wiring from the power supply and different nodes of an active hub.

3) gent Hub

⇒ It works like active hubs and include remote management capabilities.

3) Bridges

⇒
- A bridge in the computer network is used to unite two or more network segments.
- The main function of bridge in network architecture is to store as well as transmit Frames among the various segments.
- Bridges use MAC hardware for transferring Frames.
- In the OSI model, bridges work at the data link and physical layers to divide the networks from larger to smaller by controlling the data flow between the two.
- These are also used for connecting two

physical local area networks to a larger logical local area network.

4) Switches

⇒
- Similar to a hub, this also works at the layer in the LAN and a switch is more clever compare with a hub.
- The hub is used for data transferring, whereas a switch is used for filtering and forwarding the data.
- So this is the more clever technique to deal with the data packets.
- Whenever a data packet is obtained from the interfaces in the switch, then the data packet can be filtered and transmits to the interface of the proposed receiver.
- Due to this reason, a switch maintains a content addressable memory table to maintain system configuration as well as memory. This table is also named as FIB (Forwarding Information Base)

5) Router

⇒
- A network router is one kind of network device in a computer network and it is used for routing traffic from one network to another.
- These two networks could be private to public company networks.

- Example: A router is considered as traffic police at the junction, he directs dissimilar traffic networks to dissimilar directions.

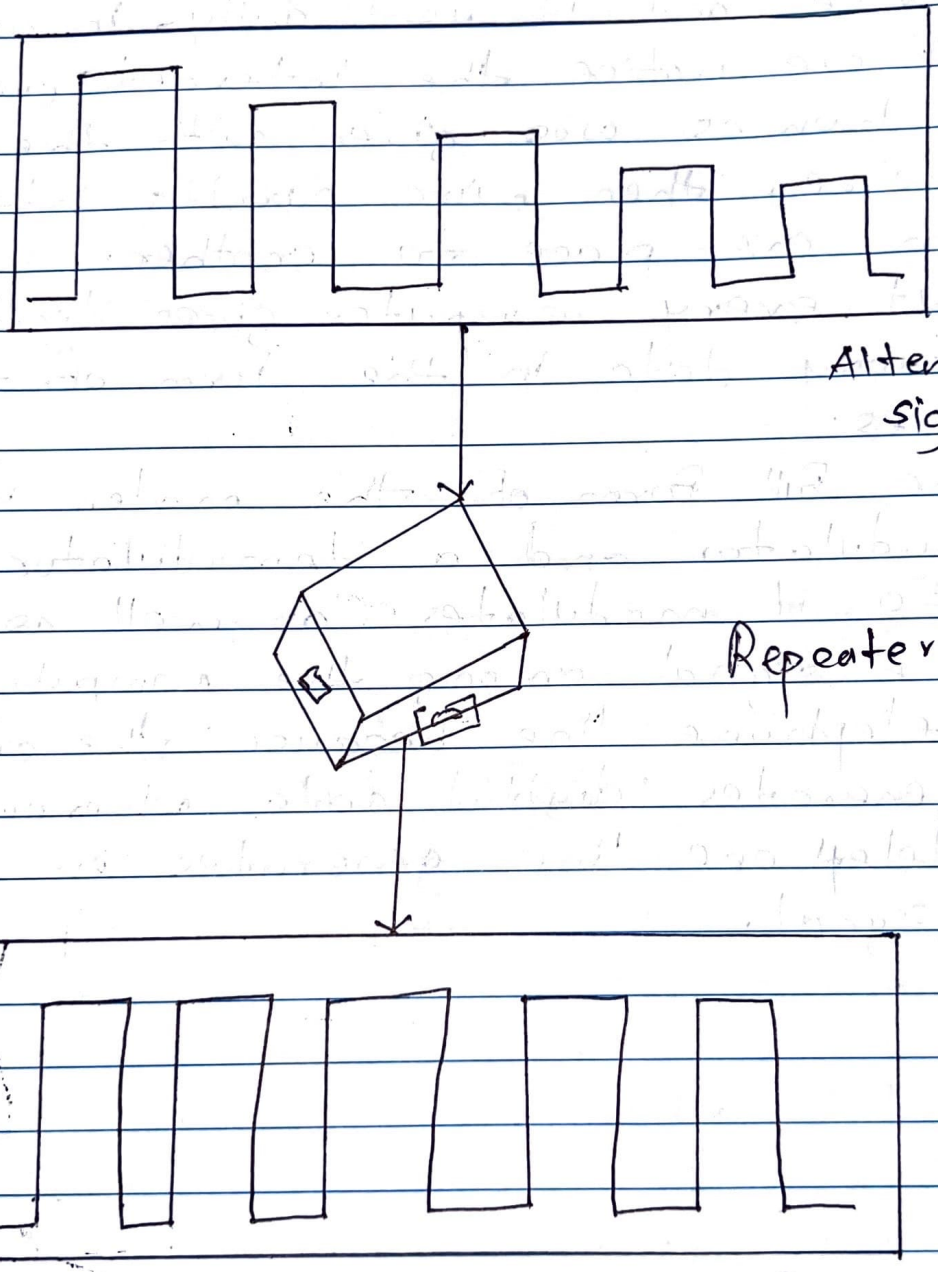
6) Gateway

⇒
- Generally, a gateway performs at the session and transport layers in the OSI model.
- Gateways offer conversion between networking technologies like OSI and TCP/IP.
- Because of this, these are connected to two or many autonomous networks, where each network has its own domain name service, routing algorithm, topology, protocols, and procedures of network administration and policies.
- Gateways execute all the functions of routers.
- Actually, a router with additional conversion functionality is a gateway, so the conversion between various network technologies is known as a protocol converter.

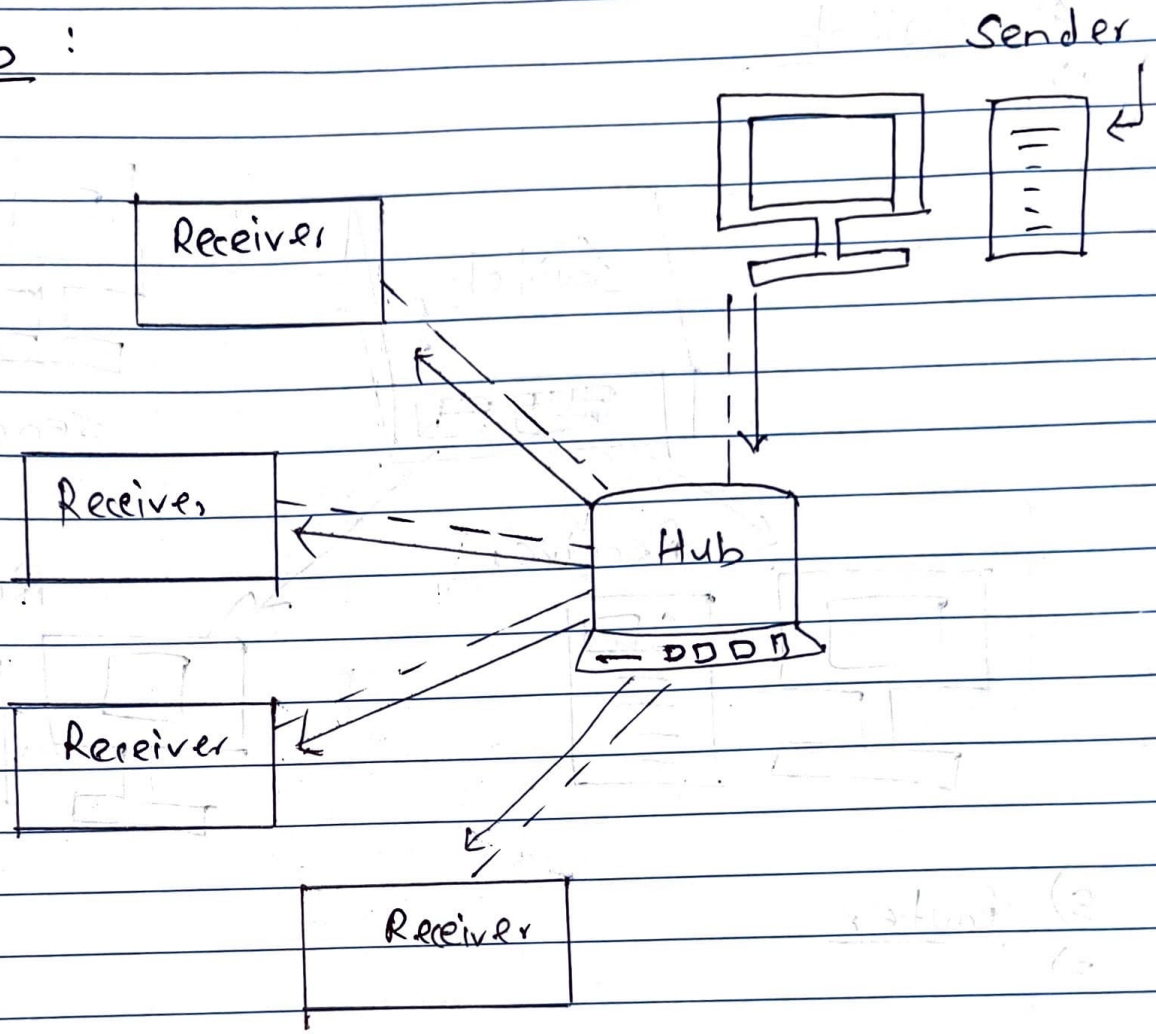
7) Modem

=> • A modem is the most important network device and it used daily in our life.
• IF we notice the internet connections to homes was given with the help of a wire, then wire carries internet data from one place to another.
• But, every computer gives digital or binary data in the form of zeros and ones.
• The full form of the modem is a modulator and a demodulator.
• So it modulates as well as demodulates the signal among the computer and a telephone line because the computer generates digital data whereas the telephone line generates an analog signal.

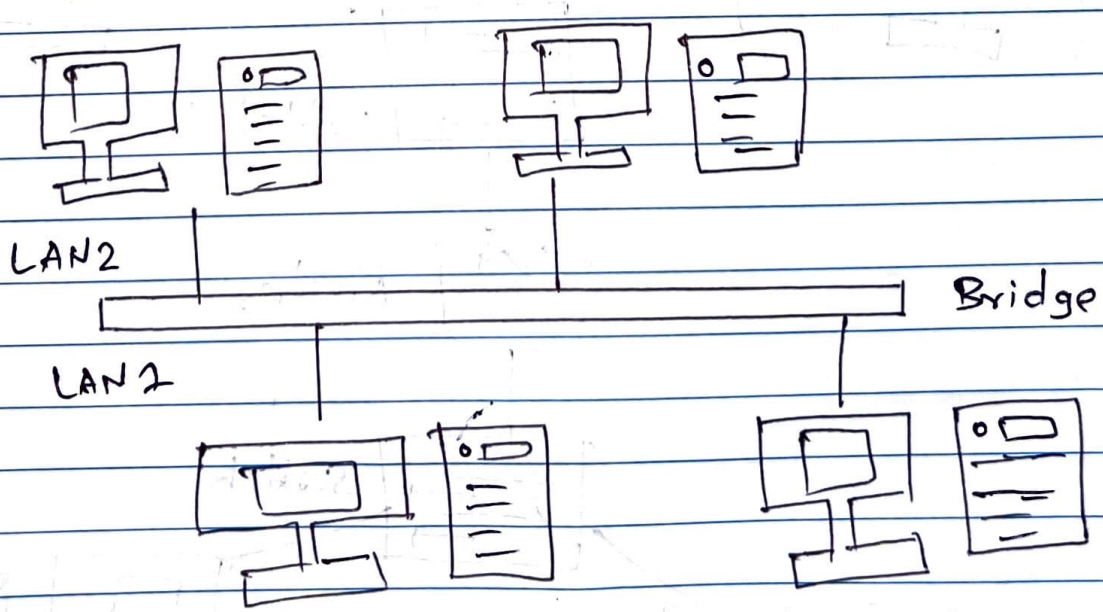
- Repeater



2) Hub :

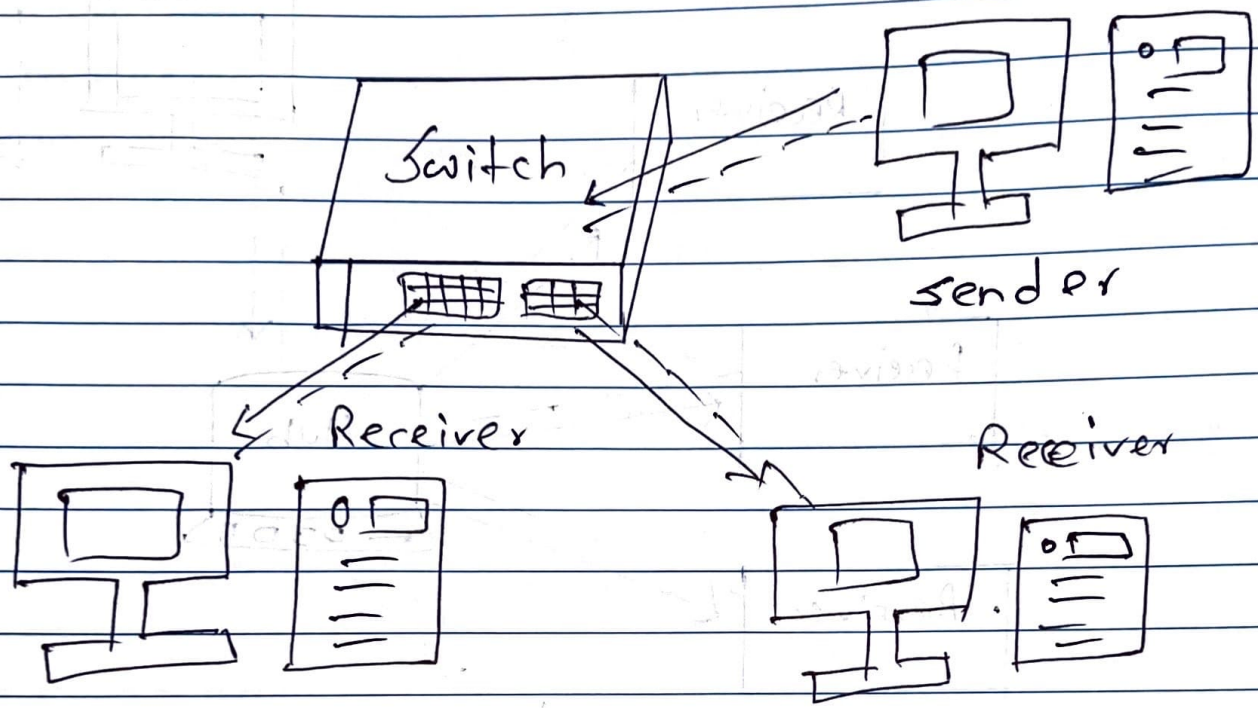


3) Bridge



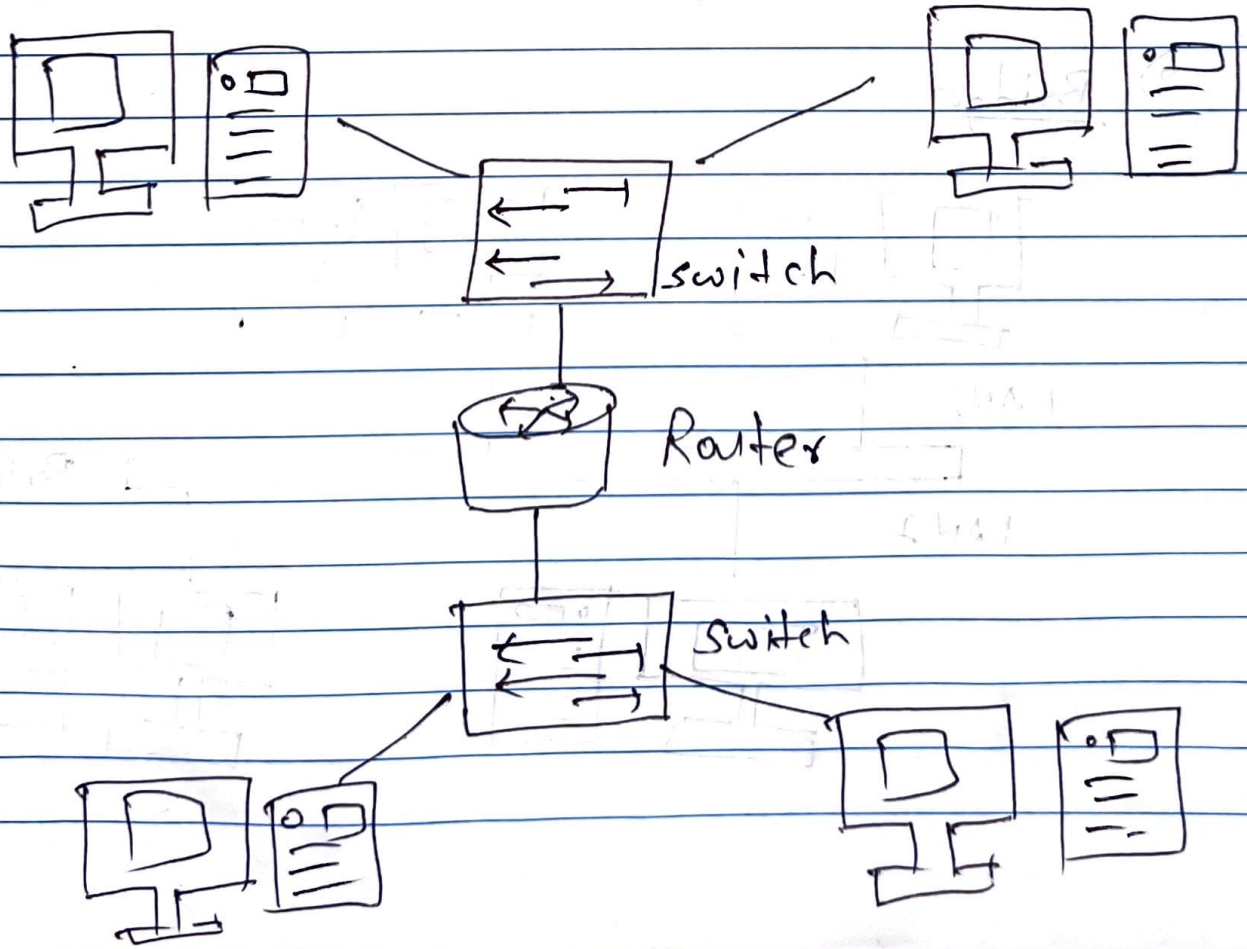
4) Switch

⇒



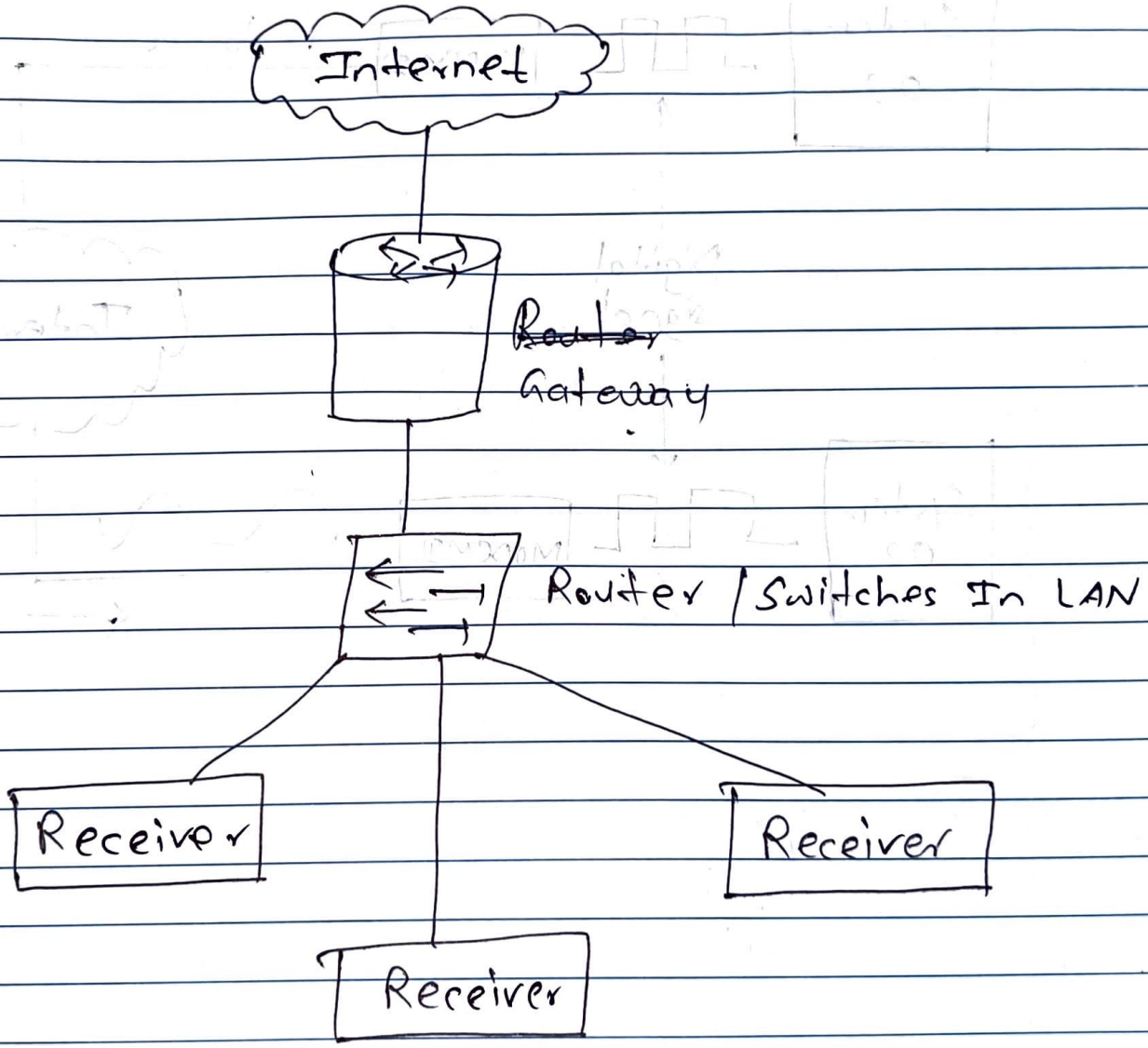
5) Router

⇒

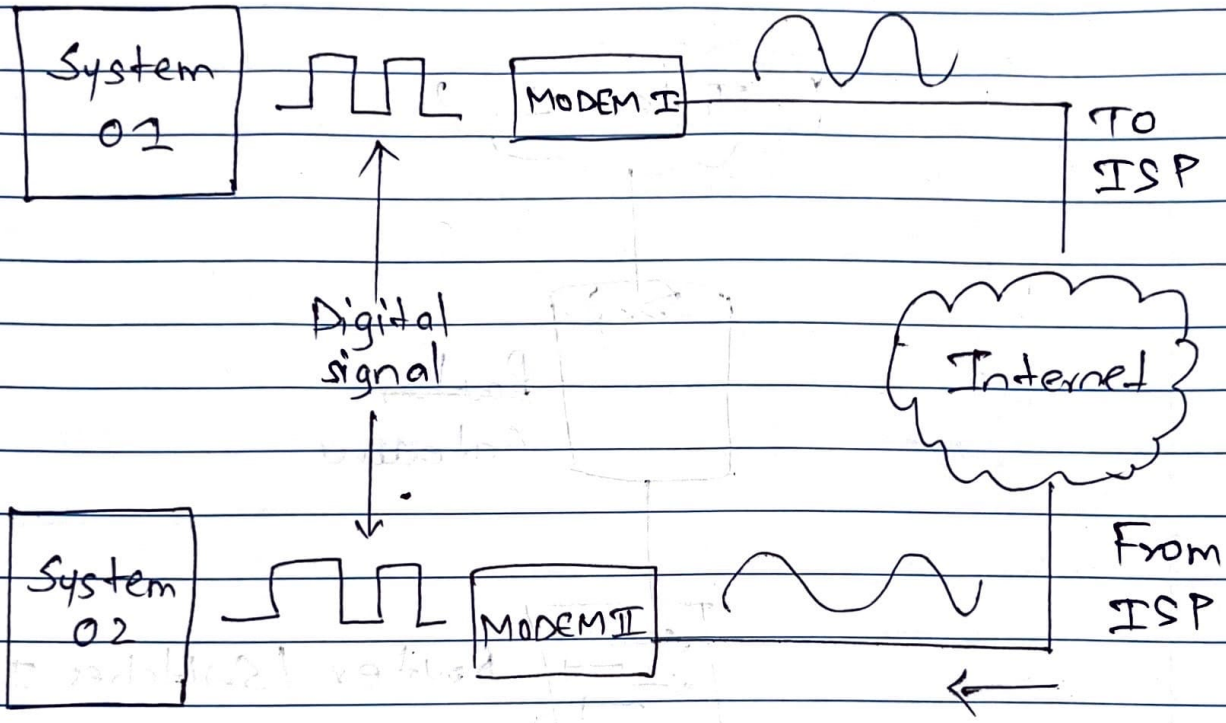


6) Gateways

⇒


Internet
Router
Gateway
Router / Switches In LAN
Receiver
Receiver
Receiver

7) Modem


System 01
MODEM I
TO ISP
Digital signal
Internet
System 02
MODEM II
From ISP

Om Shete
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
TE C32
2103163

Assignment 2

Q.1 Write short notes on the
i) Ethernet ii) IPv6 iii) ssh

⇒

i) Ethernet

⇒

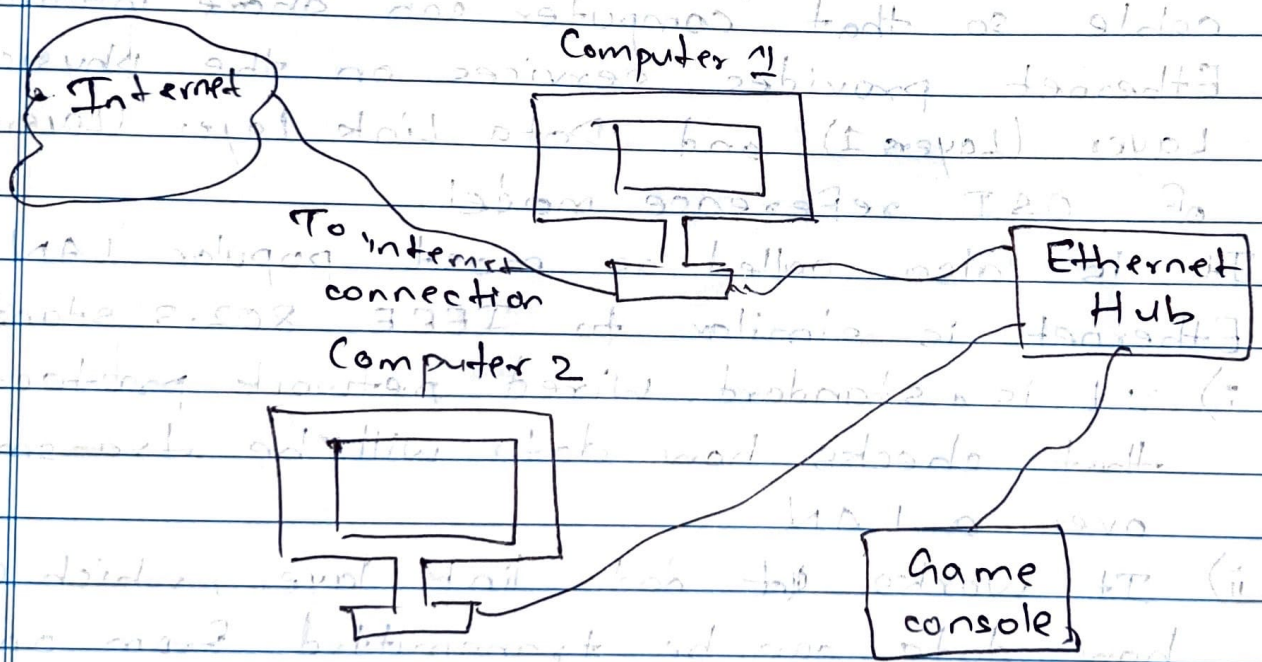
- Ethernet is a computer network technology which is used for connecting number of computers to which form a LAN.
- Ethernet connects computers together with cable so that computer can share information.
- Ethernet provides services on the physical Layer (Layers 1) and Data Link Layer (Layers 2) of OSI reference model.
- It is also called as most popular LAN.
- Ethernet is similar to IEEE 802.3 standard
  i) It is a standard wired network protocol that checks how data will be transmitted over a LAN.
  ii) It works at data link layer, which checks how data can be transmitted from one device to other device over same network segment.
- For connection Ethernet cable is inserted into the Ethernet port of our system. It can travel over speed of one Gigabyte per second.
- Ethernet is standard communication protocol embedded in software and hardware devices.

- There are two types of Ethernet Networks:
  i) Wired Ethernet Network
  ii) Wireless Ethernet.

i) Wired Ethernet

⇒

1) The ethernet technology mainly works with the fibre optic cables that connects devices within a distance of 10 km. The Ethernet supports 10 Mbps.



2) A computer network interface card (NIC) is installed in each computer, and is assigned to a unique address.
3) An Ethernet cable runs from each NIC to the central switch or hub.

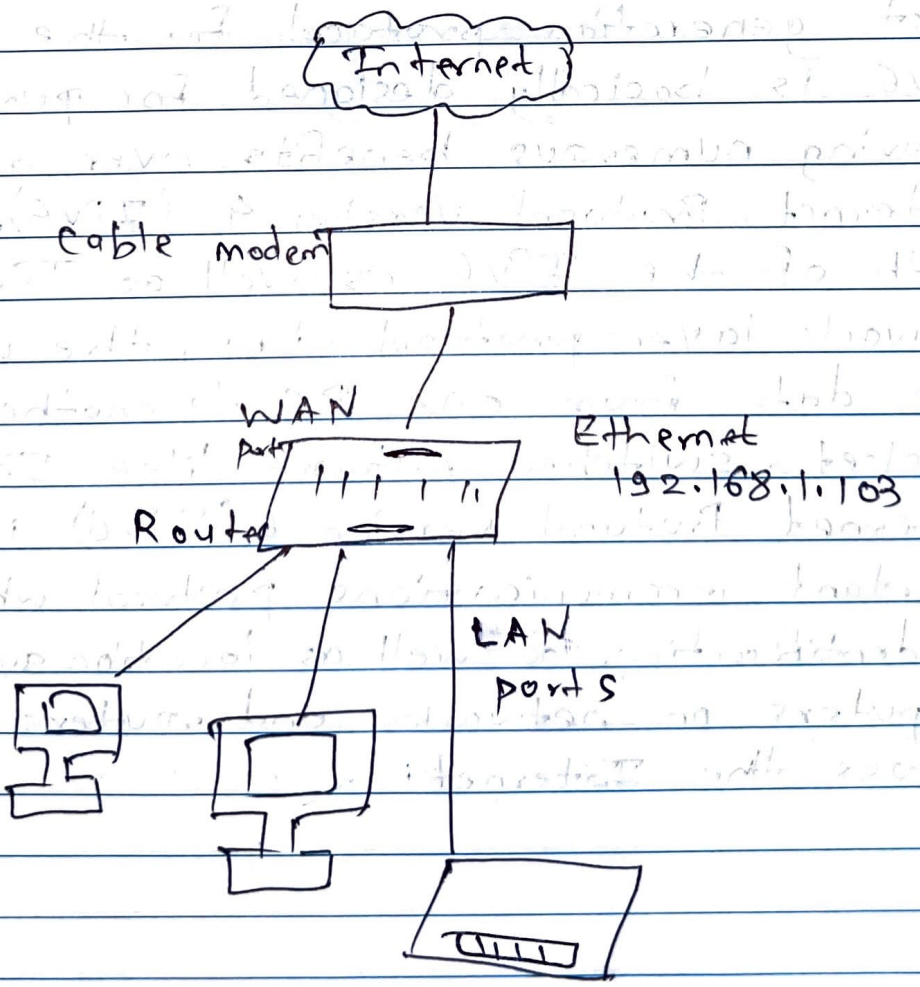
THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

?i) Wireless Ethernet

⇒

1) Ethernet networks can also be wireless. Rather than using ethernet cable to connect the computers, wireless NICs use radio waves for two-way communication with a wireless switch or hub.

2) It consists of Ethernet ports, wireless NICs switches and hubs. Wireless network technology can be more flexible to use, but also requires extra care in configuring security.



ii) IPV6

⇒

- Shortcomings of IPV4 like address depletion prompted a new version of IP in the early 1990s. The next or new version, which is called Internet Protocol version 6 (IPV6). It is also called as IP new generation (IPng) protocol.
- IPV6 has increased the address space immensely and also redesign the format of IP packet. It has also revise some of the auxillary protocol for e.g., ICMP
- IP V6 or Internet protocol version 6 is the next generation protocol for the Interned.
- IPV6 is basically designed for purpose of proving numerous benefits over current Internet Protocol Version 4 (IPV4).
- Both of the IPV6 as well as IPV4 define network layer protocol, i.e., the way to send the data from one PC to another PC over packet-switched network like Internet.
- Internet Protocol version 6 (IPV6) is an important communications protocol which provides an identification as well as location system for computers on networks and routers traffic across the Internet.

- IETF has developed IPv6 with the very old problem regarding IPv4 address exhaustion. The basic intention of IPv6 is to replace IPv4. Devices which are present on the Internet are assigned a unique IP address for the purpose of identification as well as location definition.
- Advantages of IpV6 :

1) Increased address space.
2) More efficient routing
3) Reduced management requirements
4) Improved methods to change ISP
5) Better mobility support.
6) Multi-homing
7) Security.
8) Scoped address : link-local, site-local and global-address space.

iii) ssh

⇒

- SSH (Secure Shell) is access credential that is used in the SSH protocol.
- In other words, it is a cryptographic network protocol that is used in for transferring encrypted data over network.
- It allows you to connect to a server, or multiple servers, without having you to remember or enter your password for each system that is to login remotely from one system into another.
- It always comes in "key pair":

1) Publickey ⇒ Everyone can see it, no need to protect it
2) Private key ⇒ Stays in computer, must be protected.

- Key pairs can be of following types:

1) User key ⇒ If public key and private key remain with the user.
2) Host key ⇒ If public key and private key are on a remote system.
3) Session key ⇒ Used when large amount of data is to be transmitted.

Q.2 Explain the purpose of following protocols with their header format.

i) ARP

⇒

- Address Resoultion Protocol (ARP) is used to map an IP address to a physical (MAC) address on a local network.
- ARP operates at the link layer of the OSI model.
- ARP is primilary used to resolve the layer 3 (IP) addresses to layer 2 (MAC) addresses. When a local device on a local network needs to communicate with another device on the same network, it must know the MAC address of the target device. ARP helps in finding the MAC address that corresponds to given IP address within the local network.
- In many local networks, device obtain IP addresses dynamically using protocols like DHCP. ARP plays crucial role in the dynamic allocation of IP addresses by helping devices find the MAC addresses corresponding to their IP address.

- ARP Header Format :

Octet offset	0	1
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 byte)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 byte)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 byte)	
20	(last 2 byte)	
22	(last 2 byte)	
24	Target protocol address (TPA) (first 2)	
26	(last 2 byte)	

ii) ICMP

=>

- Internet Control Message Protocol (ICMP), is another important network protocol that operates at the network layer (layers 3) of the OSI model.
- ICMP is primilary used for reporting errors and conditions related to the IP packet delivery process.
- ICMP includes a function for network troubleshooting known as the "ping" utility.
- A device can send an ICMP Echo Request message to another device, and if the target device is reachable and operational, it will reply with an ICMP Echo Reply Message.
- The header format :

Type (8 bit)	Code (8 bit)	Checksum (16bit)
Extended Header (32 bit)		
Data / Payload (Variable length)		

- ICMP can be used for network management and control.

iii) DNS

⇒

- Domain Name system (DNS) is a critical protocol used in computer networks to translate human-readable domain names into IP addresses that computers use to identify each other on the internet.
- DNS is primilary used to resolve domain names to IP addresses. When you enter a website's URL in a web browser or send an email to a domain, DNS is responsible for translating the human-readable domain name into the corresponding IP address so that data can be routed to correct destination.
- DNS can be configured with redundant servers and failover mechanisms.
- If one DNS server becomes unavailable, DNS can automatically direct traffic to an alternative server to ensure service continuity.
- DNS is also used for reverse lookups, where an IP address is translated back into a domain name.
- This is often used for security and logging purposes.

Q.3 Discuss Persistent and Non-persistant protocols used in Transport and Application layer of TCP/IP protocol suite.

⇒

- **Persistent HTTP:**

- With persistent connections, the servers leaves the TCP connection open after sending responses and hence the subsequent requests and responses between the same client and server can be sent.
- Multiple objects can be sent over a single TCP connection between client and server.
- Fewer RTTs and less slow start.
- HTTP 1.1 uses persistent HTTP in default mode.

- **Non-Persistent HTTP:**

- A non-persistent connection is the one that is closed after the server sends the requested object to the client.
- At most one object is sent over a TCP connection.
- 2 RTTs to fetch each object.
- HTTP 1.0 uses non-persistent HTTP.

02/10/23