# AIRPORT MANAGEMENT SYSTEM

## CASE STUDY

*19CSE301 - Computer Networks*

***Date:*** *October 26, 2021*

**Group Details:**

| S.No | Name of the Student | Roll No. |
|---|---|---|
| 1. | RAVELLA ABHINAV | CB.EN.U4CSE19453 |
| 2. | SINGADI SHANTHAN REDDY | CB.EN.U4CSE19459 |
| 3. | PENUGONDA KOUSHIK | CB.EN.U4CSE19449 |
| 4. | P S KALAIARASAN | CB.EN.U4CSE19446 |

## Description:

The project is to design a network in an airport. The airport has four departments :

1. Airport authority (Ticket Mgmt.)
2. Flight service providers (Flight Mgmt.)
3. Help Desk
4. Security

The Flight service provider maintains a server which handles the flight management controls. Airport authority should have access only to the specific server in the airport authority network and not to any other systems. The Help Desk should have access to a high-speed internet connection, which should be shared among all the users in all the departments.

The airport authority has 15 clients, the flight service providers have 10 clients and the help desk have 10 clients.

## Why do we need Network?

The data network is undoubtedly a key asset in any airport. It underpins everything from crucial day-to-day operations such as managing bag drops and security queues, to improving passenger experience, with optimised traffic flows and proactive management. It is used by multiple groups within the airport – passengers, third-party businesses and staff.

## Network Type:

**VLAN:** A virtual LAN (VLAN) address issues such as scalability, security, and network management. Network architects set up VLANs to provide network segmentation. Routers between VLANs filter broadcast traffic, enhance network security, perform address summarization, and mitigate network congestion. In a network utilizing broadcasts for service discovery, address assignment and resolution and other services, as the number of peers on a network grows, the frequency of broadcasts also increases.

## Servers Types:

- ***Client Servers:*** In the client/server programming model, a server is a program that awaits and fulfils requests from client programs in the same or other computers. A given application in a computer may function as a client with requests for services from other programs and also as a server of requests from other programs.

- ***FTP Servers:*** One of the oldest of the Internet services, File Transfer Protocol, makes it possible to move one or more files securely between computers while providing file security and organization as well as transfer control.

- ***Real-Time Communication Servers:*** Real-time communication servers, formerly known as chat servers or IRC Servers, and still sometimes referred to as instant messaging (IM) servers, enable large numbers users to exchange information near instantaneously.

- ***Proxy Servers:*** Proxy servers sit between a client program (typically a Web browser) and an external server (typically another server on the Web) to filter requests, improve performance, and share connections.

## Software/ Operating System:

- ***Operating System:***

  IBM AIX
  Linux (RedHat and Novell SuSE)
  Microsoft Windows Server
  Oracle Microsystems Server
  Real Time OS - uCOS
  VxWorks
  LynxOS

- ***Cloud Services:***
  Amazon AWS

## Hardware/ Devices:

- ***Servers:***
  Base Server – HPE ProLiant DL380 Gen10 2U Rack Server

  Intel Xeon Silver 4208 (2nd Gen.,2.1GHz/ 8Core) Processor with 16GB RAM / Open Bay P/N P02462-B21

  Additional 1.2TB 10K RPM SFF DS Hot Plug SAS HDD (2.5") P/N 872479-B21

- ***Routers:***
  Cisco SG350X-24 Stackable Switch with 24 Ports

- ***Printers:***
  CUSTOM TK302III
  Canon Laser Compact All-in-One Printer, image CLASS MF241D

## *Why do we need Performance Parameters?*

The demands on networks are increasing every day, and the need for proper network performance measurement is more important than ever before. Effective network performance translates into improved user satisfaction, whether that be internal employee efficiencies, or customer-facing network components such as an e-commerce website, making the business rationale for performance testing and monitoring self-evident.

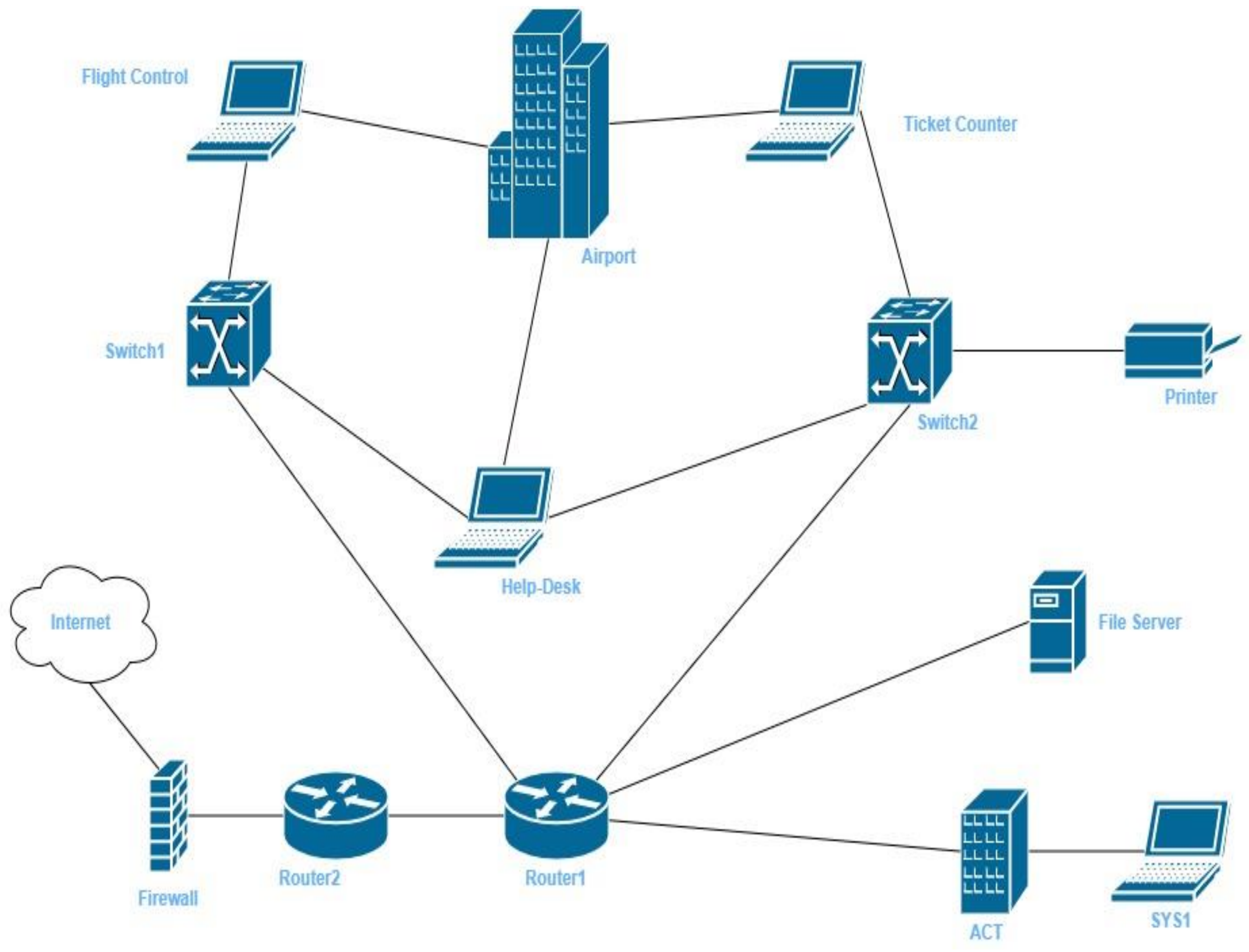
***Performance Parameters:***

| ***Parameter*** | ***Meaning*** | ***Formula*** |
|---|---|---|
| Bandwidth | Bandwidth is the capacity of a network communications link to transmit the maximum amount of data from one point to another in a given amount of time | Expressed as bits per second (bps), millions of bits per second (Mbps) or billions of bits per second ( Gbps). |
| Throughput | Throughput measures the percentage of data packets that are successfully being sent; | |
| Packet Loss | Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination. Due to network congestion | Efficiency = 100% * (transferred - retransmitted) / transferred<br><br>Network Loss = 100 - Efficiency |
| Transmission time | This depends on the size of the message and the bandwidth of the channel. | Transmission time=Message size / Bandwidth |
| Propagation Time | Propagation time measures the time required for a bit to travel from the source to the destination. | Propagation time = Distance /Propagation speed |

## *Departments in Airport Management:*

| ***Roll No.*** | ***Department*** | ***Description*** |
|---|---|---|
| CB.EN.U4CSE19446 | Flight Management | Helps to add new flights, remove flights, or view flights. |
| CB.EN.U4CSE19449 | Ticket Management | A passenger can view his ticket using PNR number, book a new ticket or cancel his ticket with PNR number. |
| CB.EN.U4CSE19453 | Help Desk | You can query, view, update data here. |
| CB.EN.U4CSE19459 | Security | Authorization, verification. |

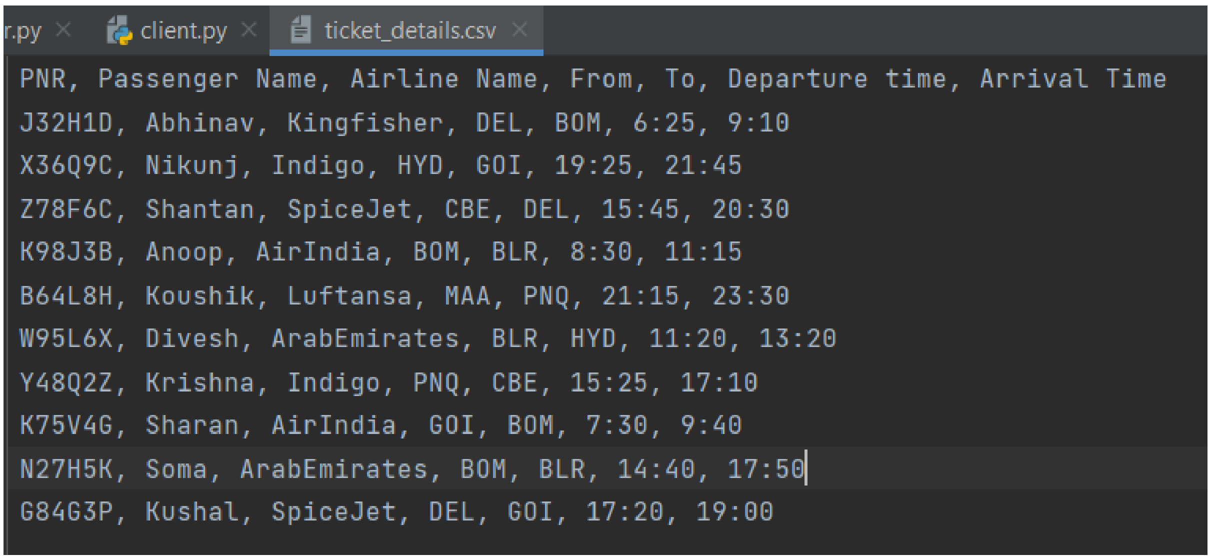
***Network Architecture diagram:***



## *Socket Programming:*

### ***Csv file Details:***

| ***Column Name*** | ***Description*** |
|---|---|
| PNR | It is the key or unique id for a ticket booking |
| Passenger Name | Name of the passenger |
| Airline Name | Airlines used by passenger to fly from source to destination |
| From location | Location from where passenger wants to fly |
| To location | Location where passenger wants to fly to |
| Departure time | Time at which flight takes off from airport |
| Arrival Time | Time at which passenger arrives at the desired airport |



```
PNR, Passenger Name, Airline Name, From, To, Departure time, Arrival Time
J32H1D, Abhinav, Kingfisher, DEL, BOM, 6:25, 9:10
X36Q9C, Nikunj, Indigo, HYD, GOI, 19:25, 21:45
Z78F6C, Shantan, SpiceJet, CBE, DEL, 15:45, 20:30
K98J3B, Anoop, AirIndia, BOM, BLR, 8:30, 11:15
B64L8H, Koushik, Luftansa, MAA, PNQ, 21:15, 23:30
W95L6X, Divesh, ArabEmirates, BLR, HYD, 11:20, 13:20
Y48Q2Z, Krishna, Indigo, PNQ, CBE, 15:25, 17:10
K75V4G, Sharan, AirIndia, GOI, BOM, 7:30, 9:40
N27H5K, Soma, ArabEmirates, BOM, BLR, 14:40, 17:50
G84G3P, Kushal, SpiceJet, DEL, GOI, 17:20, 19:00
```

*Server.py:*

```python
import socket
import csv
from _thread import *

IP = socket.gethostbyname(socket.gethostname())
PORT = 4455
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"


def new_client(conn, addr):
    """ Getting Filename from Client. """
    filename = conn.recv(SIZE).decode(FORMAT)

    if filename != "exit":
        print(f"\n[CLIENT] Filename sent.")
        """ Opening the file. """
        file = open("data/" + filename, 'rb')
        line = file.read(1024)

        """ Reading the file and sending it. """
        print("\n[CLIENT] Receiving the data.")
        while (line):
            conn.send(line)
            line = file.read(1024)

        """ Closing the file"""
        file.close()
        print("\n[CLIENT] Data received.")

        """ Closing the connection from the client. """
        conn.close()
        print(f"\n[DISCONNECTED] {addr} disconnected.")

    else:
        """ Closing the connection from the client. """
        conn.close()
        print(f"\n[DISCONNECTED] {addr} disconnected.")


def main():
    """ To check how many clients are connected to the server.
"""
    No_of_Clients = 0

    """ Staring a TCP socket. """
    print("\n[STARTING] Server is starting.")
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    """ Bind the IP and PORT to the server. """
    try:
        server.bind(ADDR)
    except socket.error as error:
        """ Error message."""
        print("Error has occured! Please try again " +
str(error))

    """ Server is listening, i.e., server is now waiting for
the client to connected. """
    server.listen(5)
    print("\n[LISTENING] Server is listening.")

    while True:
        """ Server has accepted the connection from the
clients. """
        conn, addr = server.accept()
        print(f"\n[NEW CONNECTION] {addr} connected.")

        """ Creating new threads to process multiple clients.
"""
        start_new_thread(new_client, (conn, addr))

        """ To print No of clients connected to server. """
        # No_of_Clients += 1
        # print("\n!!!!!! Clients Details !!!!!! ")
        # print("Clients online : " + str(No_of_Clients))


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

***Client.py:***

```
import socket
import csv
import string
import random


def id_generator(size=5, chars=string.ascii_uppercase +
string.digits):
    return ''.join(random.choice(chars) for _ in range(size))


IP = socket.gethostbyname(socket.gethostname())
PORT = 4455
ADDR = (IP, PORT)
SIZE = 1024
```

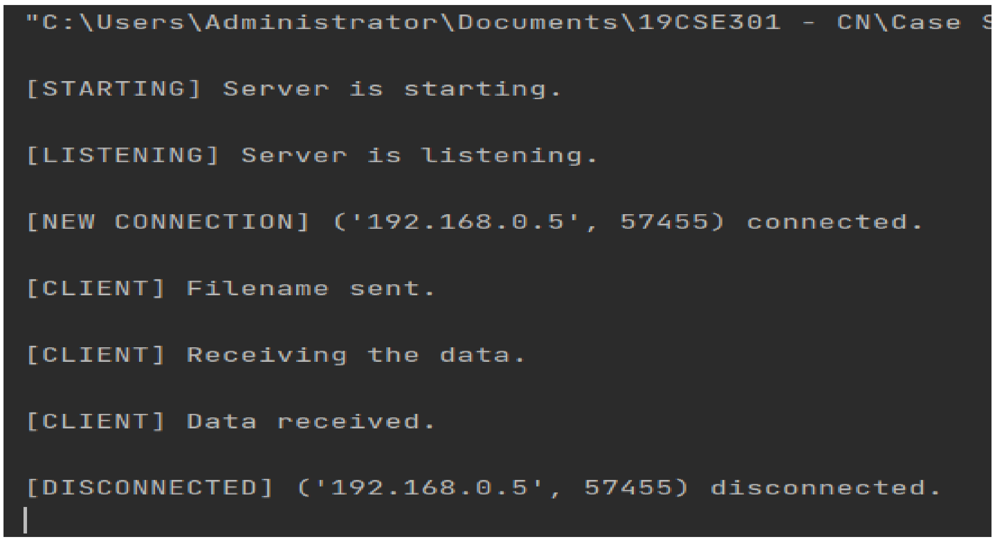
```
FORMAT = "utf-8"


def main():
    """ Staring a TCP socket. """
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    """ Connecting to the server. """
    client.connect(ADDR)
    print("\n[SERVER] Connected.")

    """ Displaying the menu and getting the option. """
    print("\n")
    operation = input("""************ Welcome to Airlines
**************

            A: Ticket Details
            B: Booking tickets
            C: Ticket Cancellation
            Q: Disconnect

            Please enter your choice: """)

    """ A. Ticket details wrt to PNR number. """
    if (operation == "A"):

        """ Sending filename to server. """
        client.send("ticket_details.txt".encode(FORMAT))
        print(f"\n[SERVER] Filename Received.")

        """ Opening a .csv file to write the data. """
        with open('ticket_details.csv', 'wb') as file:

            """ Writing data sent by the server to the file. """
            print("\n[SERVER] Data is being sent.")
            while True:
                data = client.recv(SIZE)
                if not data:
                    break
                file.write(data)

        """ Closing the file. """
        file.close()
        print(f"\n[SERVER] Process completed.")

        """ Closing the connection from the server. """
        client.close()
        print("\n[SERVER] Disconnected.")

        """ Getting PNR number from the user. """
        print("\n")
        print("\n")
        PNR = input("Enter your PNR number: ")
        """ Opening the csv file and creating iterative obj. """
        csv_file = open('ticket_details.csv')
        csv_obj = csv.reader(csv_file)

        """ Getting fields of the file. """
        fields = next(csv_obj)

        """ Printing the ticket Details. """
        print("\nYour ticket details are: ")
        print("[ " + ', '.join(field for field in fields) + "]")
        for row in csv_obj:
            if (row[0] == PNR):
                print(row)
        print("\n")

    """ B. Ticket Booking. """
    if (operation == "B"):
        """ Sending filename to server. """
        client.send("ticket_details.txt".encode(FORMAT))
        print(f"\n[SERVER] Filename Received.")

        """ Opening a .csv file to write the data. """
        with open('ticket_details.csv', 'wb') as file:

            """ Writing data sent by the server to the file. """
            print("\n[SERVER] Data is being sent.")
            while True:
                data = client.recv(SIZE)
                if not data:
                    break
                file.write(data)

        """ Closing the file. """
        file.close()
        print(f"\n[SERVER] Process completed.")

        """ Closing the connection from the server. """
        client.close()
        print("\n[SERVER] Disconnected.")

        """ Getting PNR number from the user. """
        print("\n")
        print("\n")
        rows = []
        name = input("Enter your name: ")
        des_from = input("\nEnter your SOURCE: ")
        des_to = input("\nEnter your DESTINATION: ")
        print(f"\n[SERVER] Ticket Confirmed.")
        print("\n")

    """ C. Ticket Cancellation wrt to PNR number. """
    if (operation == "C"):
```

```python
        """ Sending filename to server. """
        client.send("ticket_details.txt".encode(FORMAT))
        print(f"\n[SERVER] Filename Received.")

        """ Opening a .csv file to write the data. """
        with open('ticket_details.csv', 'wb') as file:

            """ Writing data sent by the server to the file. """
            print("\n[SERVER] Data is being sent.")
            while True:
                data = client.recv(SIZE)
                if not data:
                    break
                file.write(data)

        """ Closing the file. """
        file.close()
        print(f"\n[SERVER] Process completed.")

        """ Closing the connection from the server. """
        client.close()
        print("\n[SERVER] Disconnected.")

        print("\n")
        print("\n")
        PNR = input("Enter your PNR number: ")
        lines = list()
        with open('ticket_details.csv', 'r') as readFile:
            reader = csv.reader(readFile)
            for row in reader:
                lines.append(row)
                for field in row:
                    if field == PNR:
                        lines.remove(row)
        with open('ticket_details.csv', 'w') as writeFile:
            writer = csv.writer(writeFile)
            writer.writerows(lines)
        print(f"\n[SERVER] Ticket Cancelled.")
        print("\n")

    """ Q. Disconnect"""
    if (operation == "Q"):
        """ Sending filename to server. """
        client.send("exit".encode(FORMAT))

        """ Closing the connection from the server. """
        client.close()
        print("\n[SERVER] Disconnected.\n")


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

## ***Screenshots:***

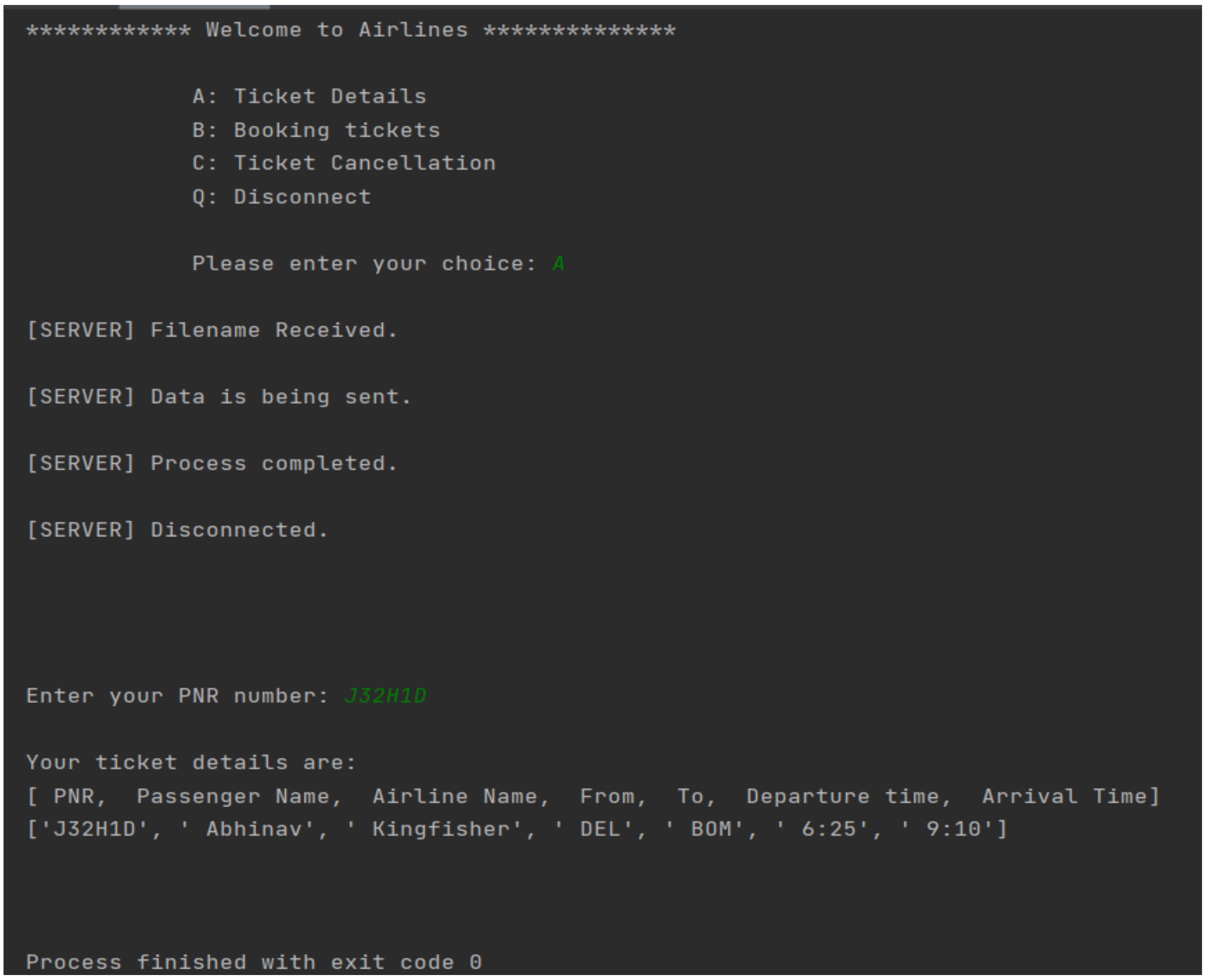
### 1. *Ticket Details:*

*Server POV:*



```
"C:\Users\Administrator\Documents\19CSE301 - CN\Case S

[STARTING] Server is starting.

[LISTENING] Server is listening.

[NEW CONNECTION] ('192.168.0.5', 57455) connected.

[CLIENT] Filename sent.

[CLIENT] Receiving the data.

[CLIENT] Data received.

[DISCONNECTED] ('192.168.0.5', 57455) disconnected.
```

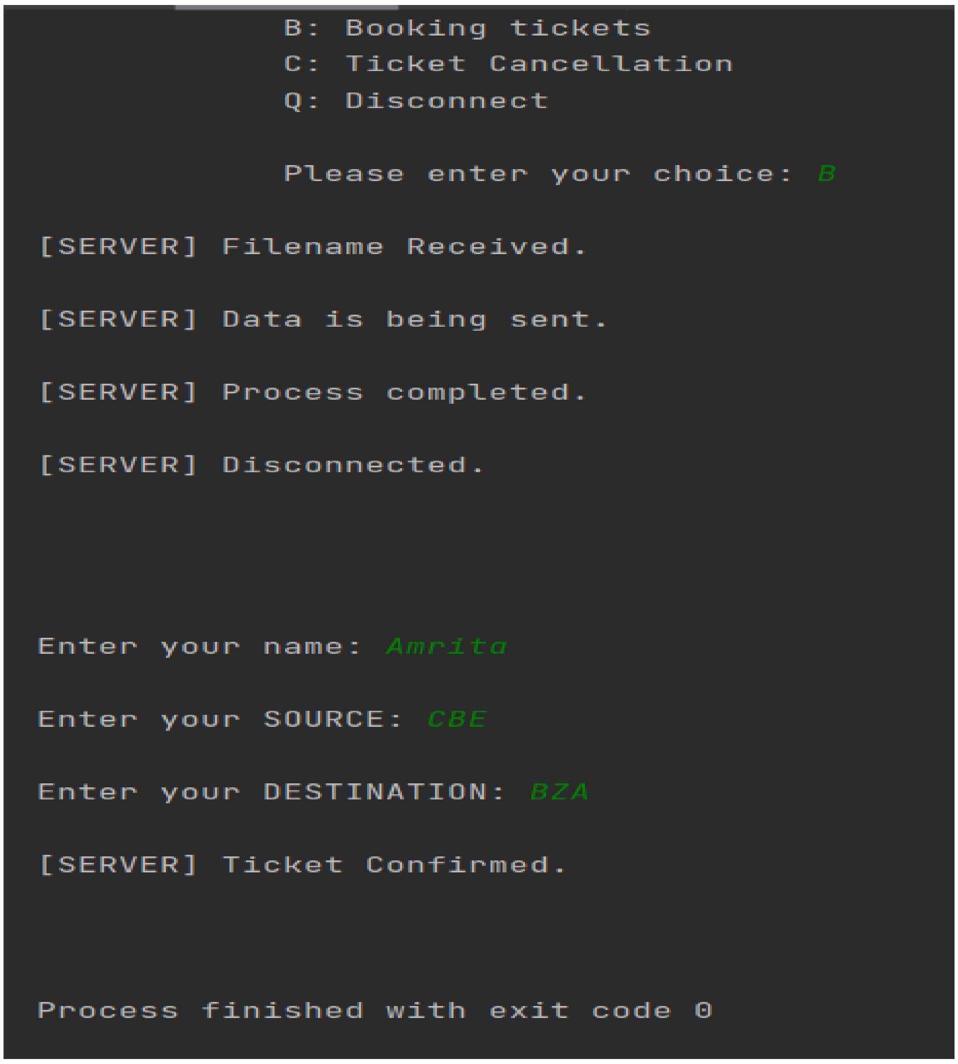
*Client POV:*



```
************ Welcome to Airlines **************

            A: Ticket Details
            B: Booking tickets
            C: Ticket Cancellation
            Q: Disconnect

            Please enter your choice: A

[SERVER] Filename Received.

[SERVER] Data is being sent.

[SERVER] Process completed.

[SERVER] Disconnected.



Enter your PNR number: J32H1D

Your ticket details are:
[ PNR,  Passenger Name,  Airline Name,  From,  To,  Departure time,  Arrival Time]
['J32H1D', ' Abhinav', ' Kingfisher', ' DEL', ' BOM', ' 6:25', ' 9:10']



Process finished with exit code 0
```

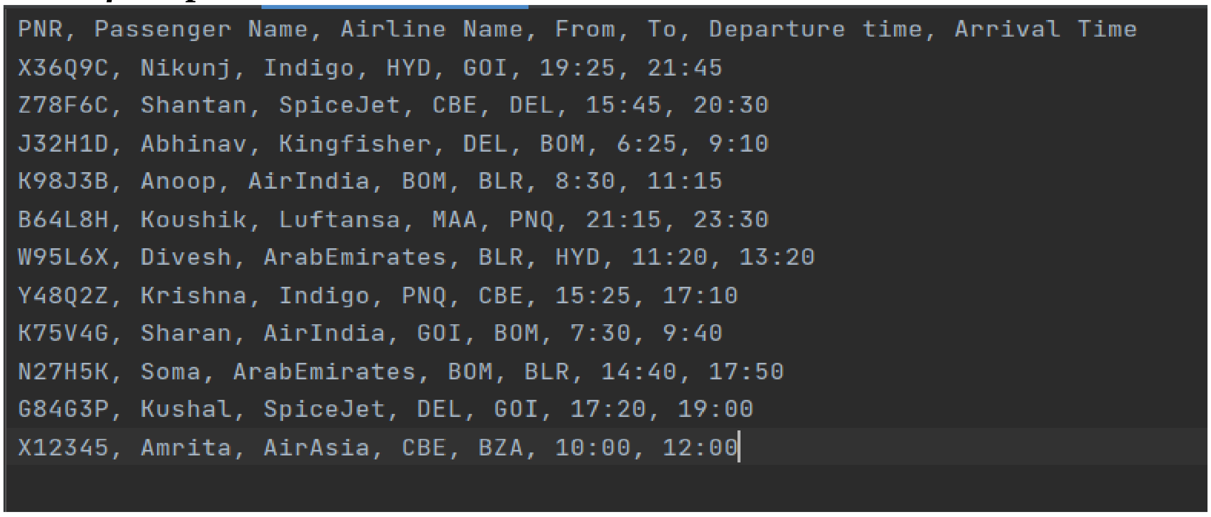
2. ***Ticket Booking:***

*Client POV:*



```
                B: Booking tickets
                C: Ticket Cancellation
                Q: Disconnect

                Please enter your choice: B

  [SERVER] Filename Received.

  [SERVER] Data is being sent.

  [SERVER] Process completed.

  [SERVER] Disconnected.



  Enter your name: Amrita

  Enter your SOURCE: CBE

  Enter your DESTINATION: BZA

  [SERVER] Ticket Confirmed.


  Process finished with exit code 0
```

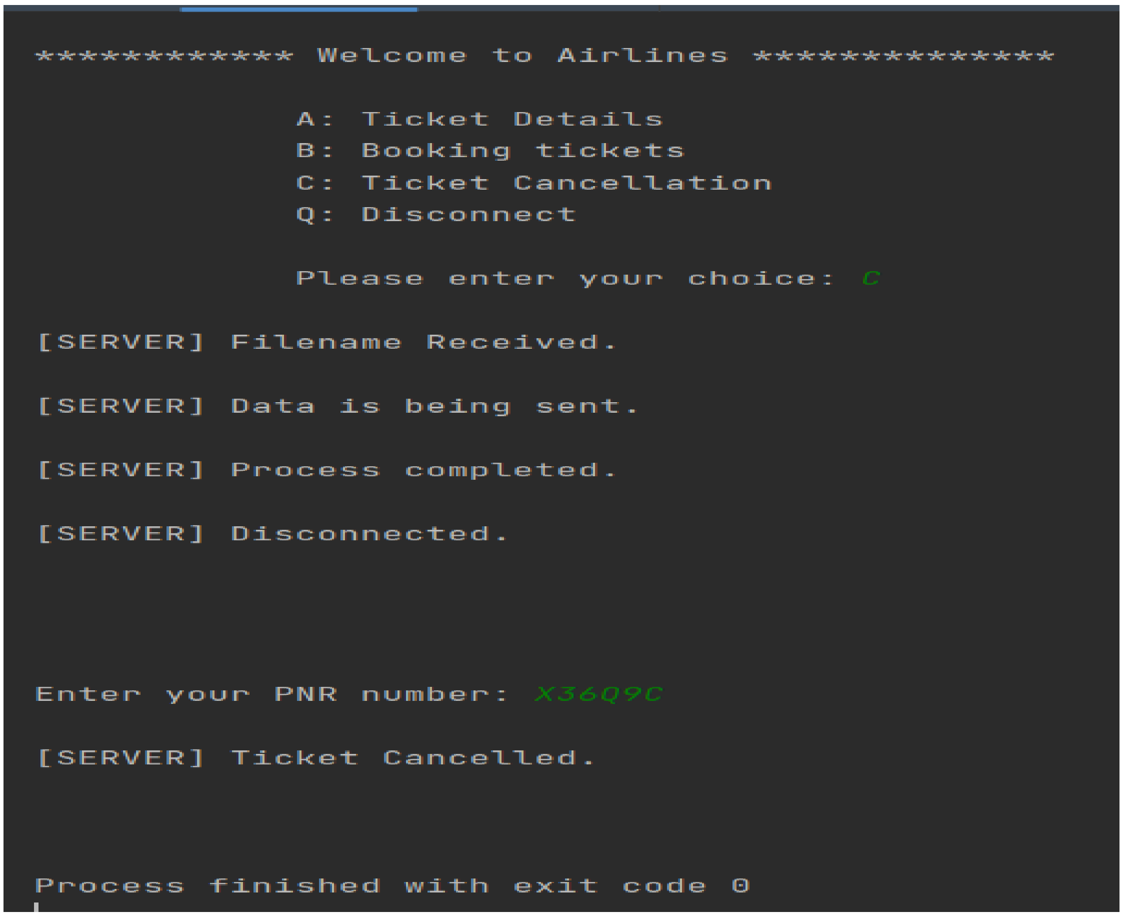
***Data after operation:***



```
PNR, Passenger Name, Airline Name, From, To, Departure time, Arrival Time
X36Q9C, Nikunj, Indigo, HYD, GOI, 19:25, 21:45
Z78F6C, Shantan, SpiceJet, CBE, DEL, 15:45, 20:30
J32H1D, Abhinav, Kingfisher, DEL, BOM, 6:25, 9:10
K98J3B, Anoop, AirIndia, BOM, BLR, 8:30, 11:15
B64L8H, Koushik, Luftansa, MAA, PNQ, 21:15, 23:30
W95L6X, Divesh, ArabEmirates, BLR, HYD, 11:20, 13:20
Y48Q2Z, Krishna, Indigo, PNQ, CBE, 15:25, 17:10
K75V4G, Sharan, AirIndia, GOI, BOM, 7:30, 9:40
N27H5K, Soma, ArabEmirates, BOM, BLR, 14:40, 17:50
G84G3P, Kushal, SpiceJet, DEL, GOI, 17:20, 19:00
X12345, Amrita, AirAsia, CBE, BZA, 10:00, 12:00
```

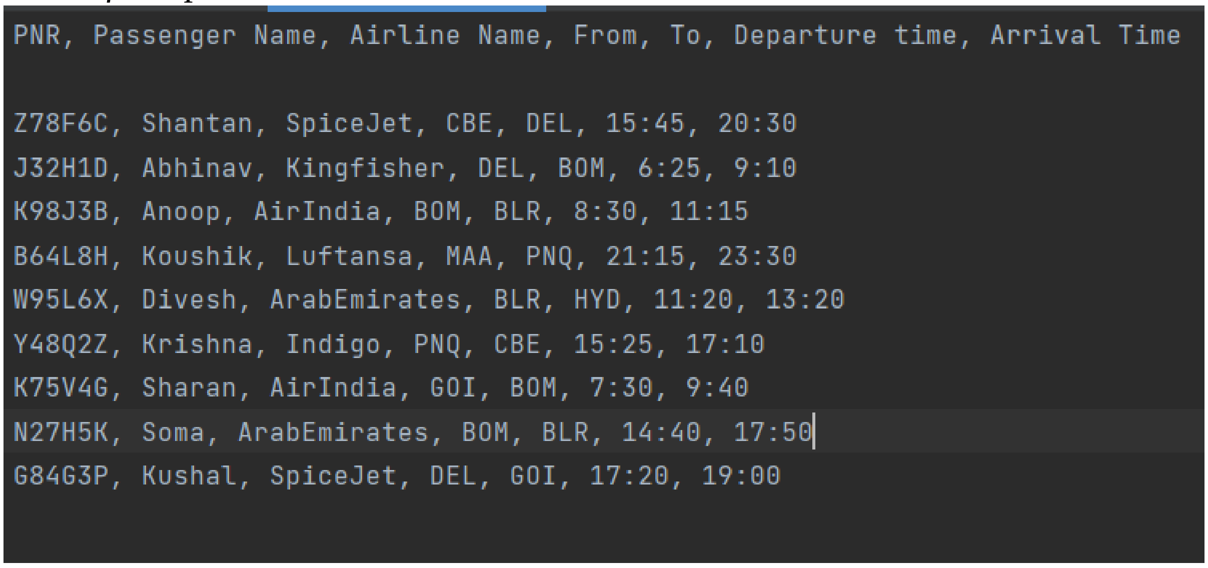
## 3. *Ticket Cancellation:*

*Client POV:*



```
************* Welcome to Airlines ***************

             A: Ticket Details
             B: Booking tickets
             C: Ticket Cancellation
             Q: Disconnect

             Please enter your choice: C

[SERVER] Filename Received.

[SERVER] Data is being sent.

[SERVER] Process completed.

[SERVER] Disconnected.



Enter your PNR number: X36Q9C

[SERVER] Ticket Cancelled.


Process finished with exit code 0
```

*Data after operation:*



```
PNR, Passenger Name, Airline Name, From, To, Departure time, Arrival Time

Z78F6C, Shantan, SpiceJet, CBE, DEL, 15:45, 20:30
J32H1D, Abhinav, Kingfisher, DEL, BOM, 6:25, 9:10
K98J3B, Anoop, AirIndia, BOM, BLR, 8:30, 11:15
B64L8H, Koushik, Luftansa, MAA, PNQ, 21:15, 23:30
W95L6X, Divesh, ArabEmirates, BLR, HYD, 11:20, 13:20
Y48Q2Z, Krishna, Indigo, PNQ, CBE, 15:25, 17:10
K75V4G, Sharan, AirIndia, GOI, BOM, 7:30, 9:40
N27H5K, Soma, ArabEmirates, BOM, BLR, 14:40, 17:50
G84G3P, Kushal, SpiceJet, DEL, GOI, 17:20, 19:00
```

## Cisco Packet Tracer Network Design:

- In this section we stimulated network design for airport management system.

Table below shows the IP address distribution for the subnets of address block 128.39.22.0/26

| Network Address | Usable host range | Broadcast Address |
|---|---|---|
| **128.39.22.0** | 128.39.22.1 - 128.39.22.62 | 128.39.22.63 |
| **128.39.22.64** | 128.39.22.65 - 128.39.22.126 | 128.39.22.127 |
| **128.39.22.128** | 128.39.22.129 - 128.39.22.190 | 128.39.22.191 |
| **128.39.22.192** | 128.39.22.193 - 128.39.22.254 | 128.39.22.255 |

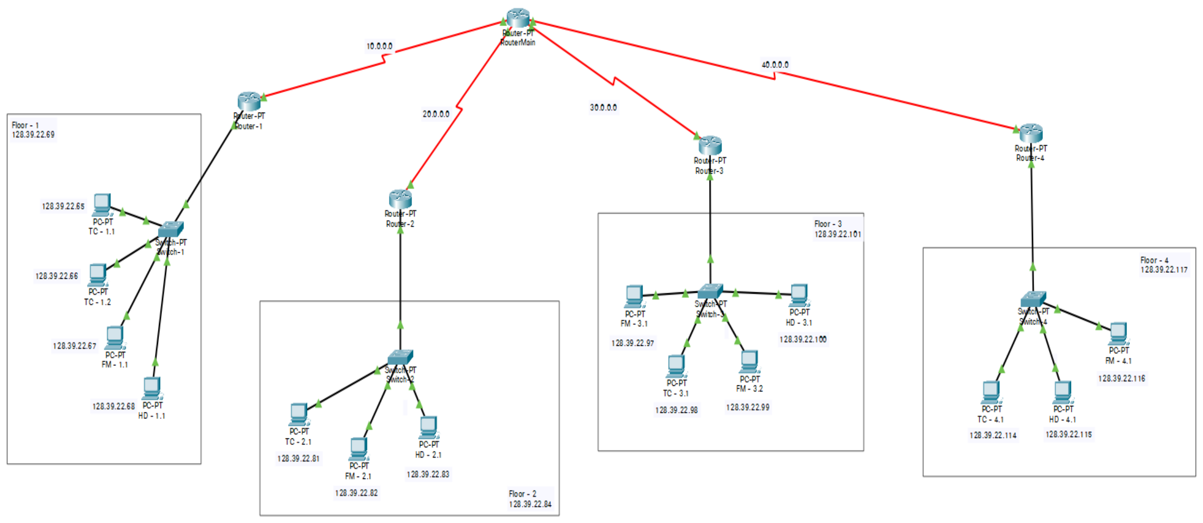
## Implementation in Case Study:

Each floor is a subnet with different departments in it and each department is a vlan.

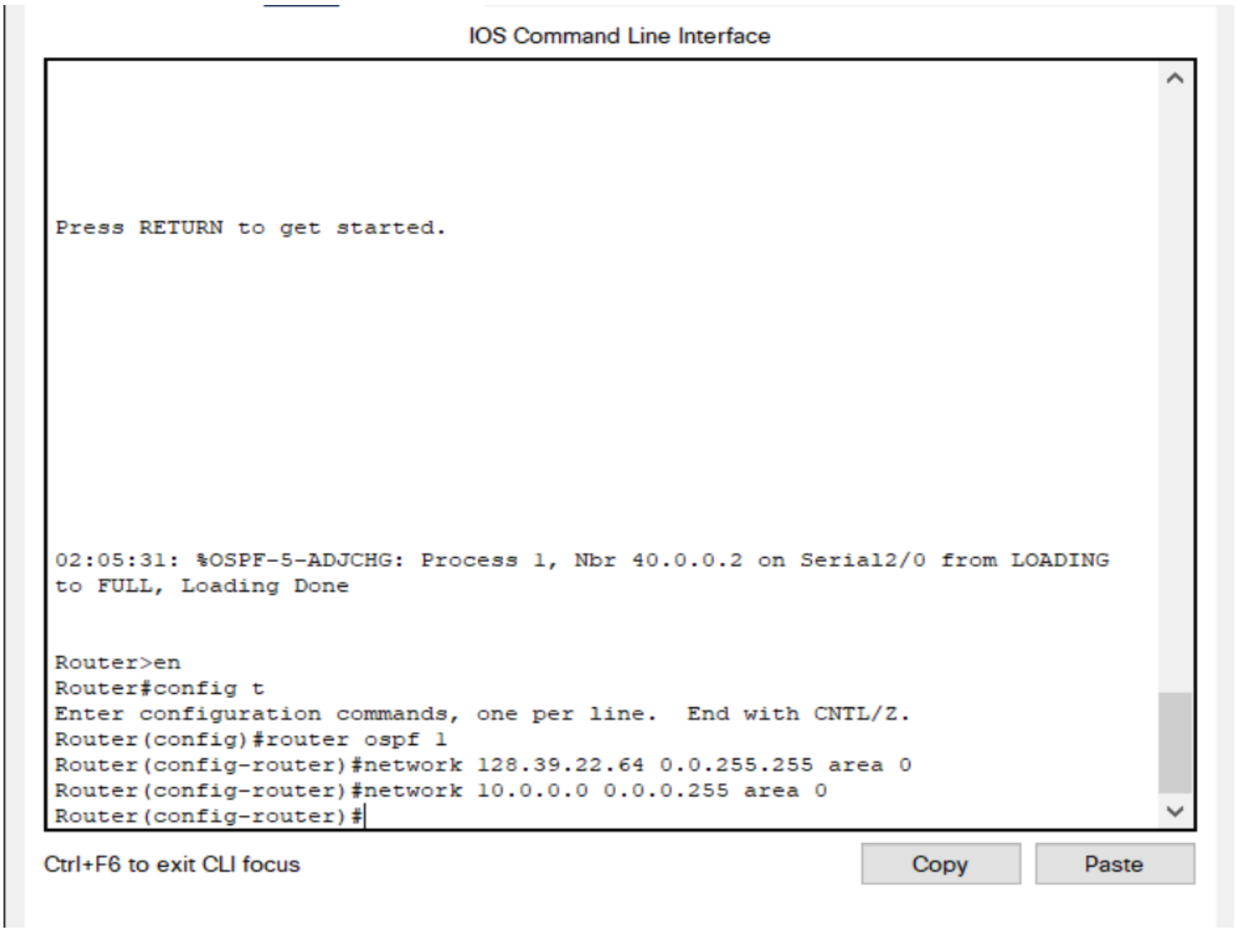
## ***<u>Routing Protocols:</u>***

1. **OSPF:**

### Network Design:



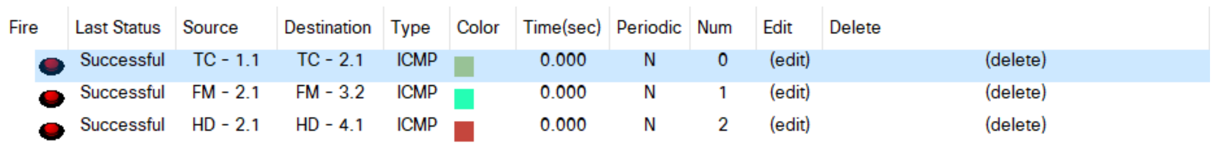
**Router Config:**



```
Press RETURN to get started.






02:05:31: %OSPF-5-ADJCHG: Process 1, Nbr 40.0.0.2 on Serial2/0 from LOADING
to FULL, Loading Done


Router>en
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#router ospf 1
Router(config-router)#network 128.39.22.64 0.0.255.255 area 0
Router(config-router)#network 10.0.0.0 0.0.0.255 area 0
Router(config-router)#
```

## Working Screenshot:

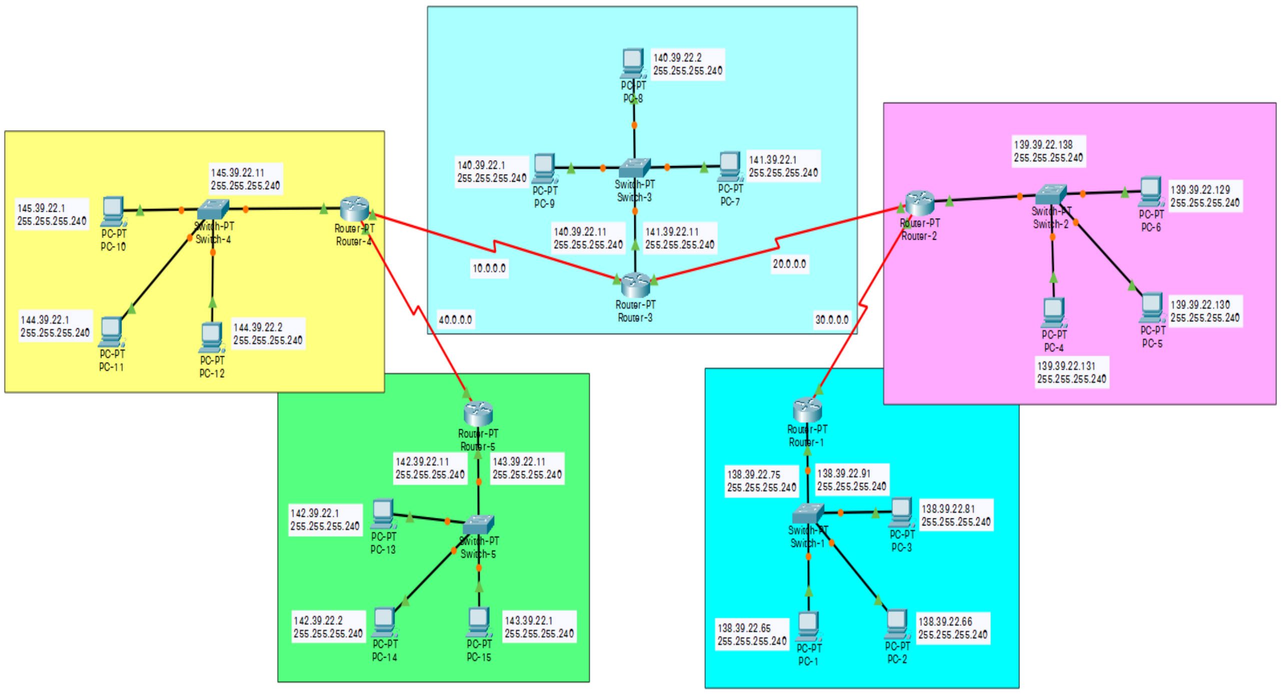
- TC-1.1 in subnet 1 and TC-2.1 in subnet 2
- FM-2.1 in subnet 2 and FM-3.2 in subnet 3
- HD-3.1 in subnet 3 and HD-4.1 in subnet 4



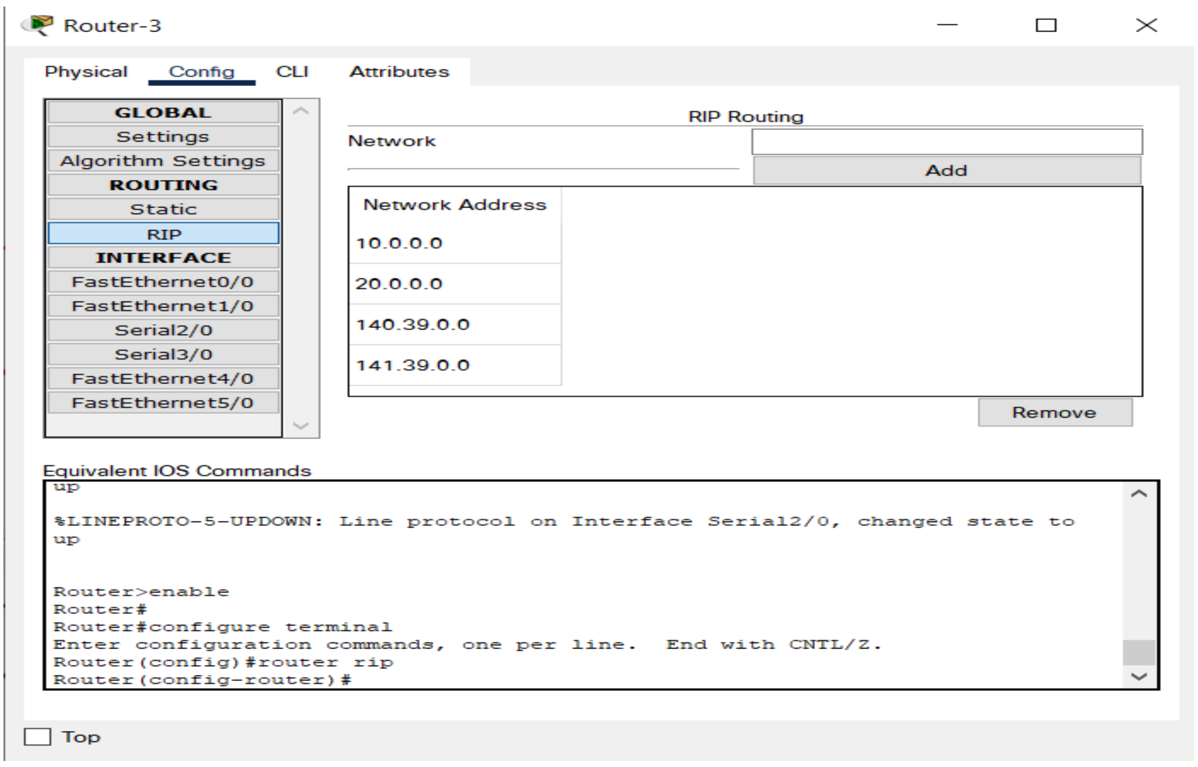
| Fire | Last Status | Source | Destination | Type | Color | Time(sec) | Periodic | Num | Edit | Delete |
|---|---|---|---|---|---|---|---|---|---|---|
| ● | Successful | TC - 1.1 | TC - 2.1 | ICMP | 🟩 | 0.000 | N | 0 | (edit) | (delete) |
| ● | Successful | FM - 2.1 | FM - 3.2 | ICMP | 🟩 | 0.000 | N | 1 | (edit) | (delete) |
| ● | Successful | HD - 2.1 | HD - 4.1 | ICMP | 🟥 | 0.000 | N | 2 | (edit) | (delete) |

## 2. RIP + VLAN:

### Network Design:



### Router Config:



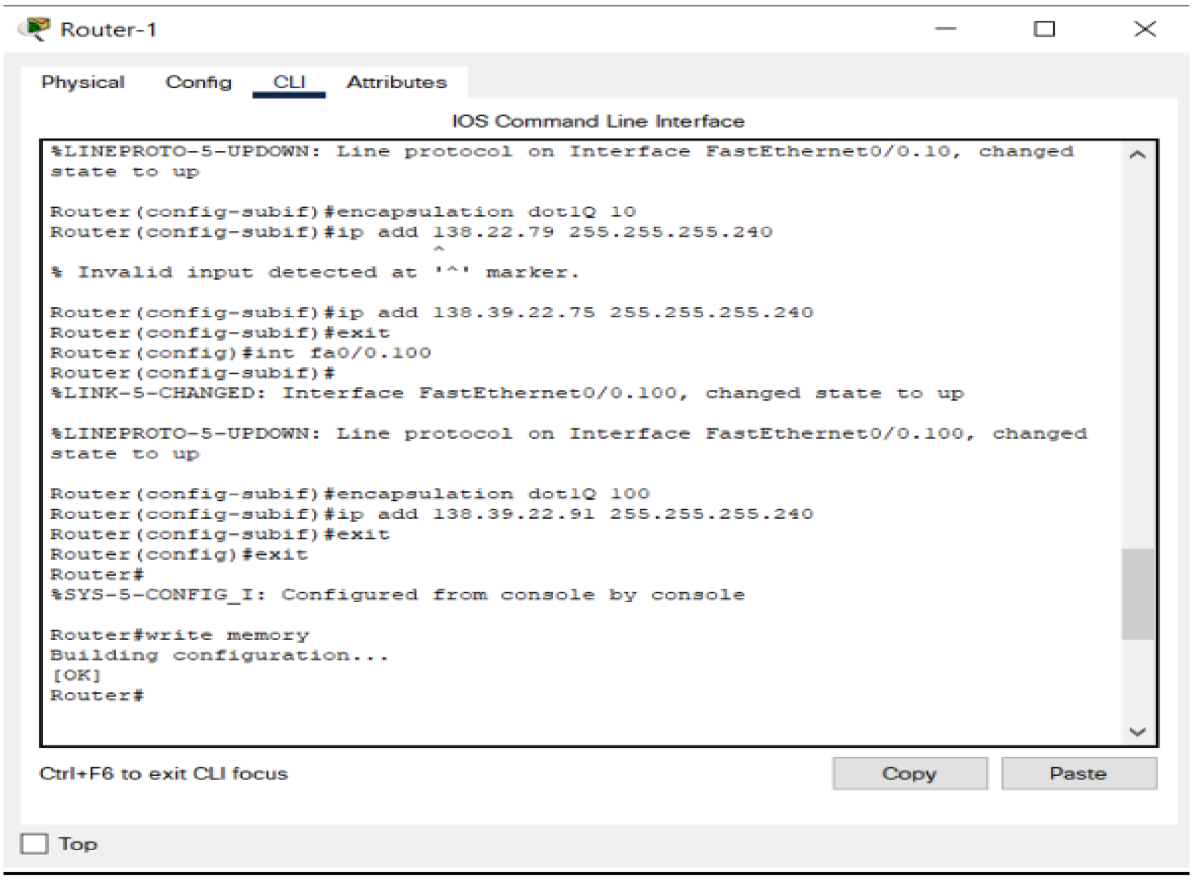
RIP Routing

Network

| Network Address |
|---|
| 10.0.0.0 |
| 20.0.0.0 |
| 140.39.0.0 |
| 141.39.0.0 |

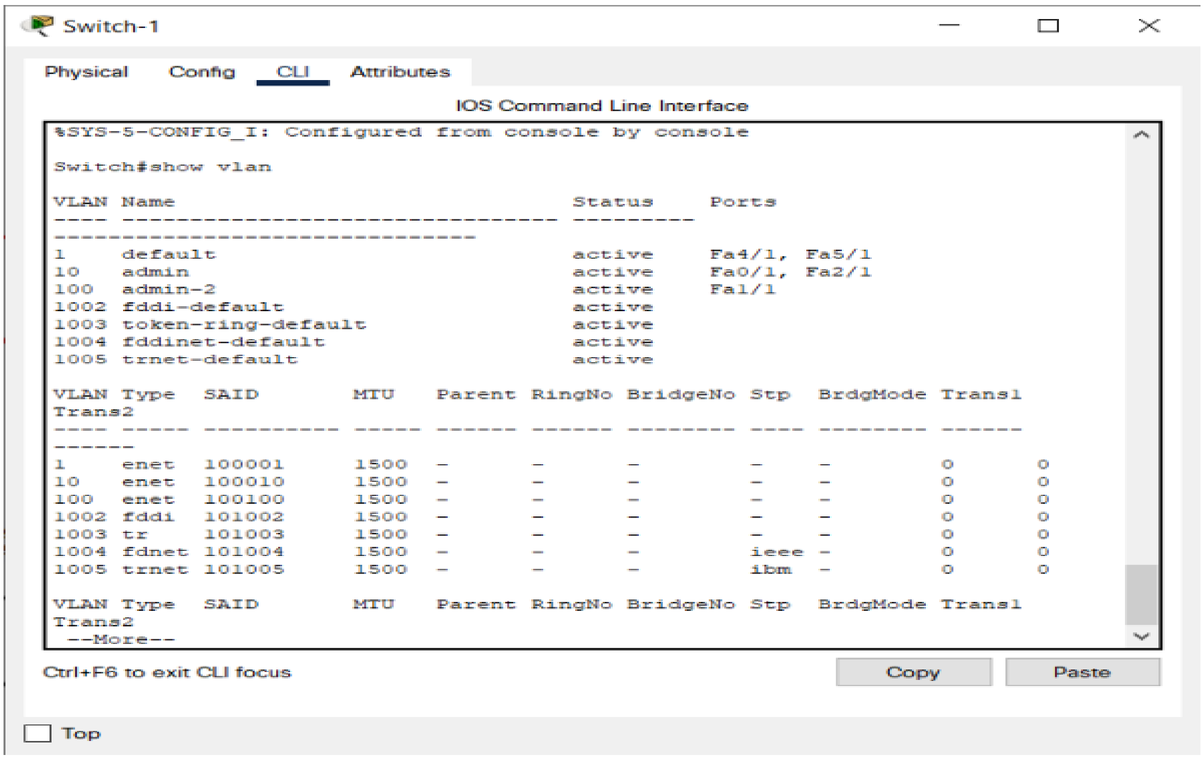
Equivalent IOS Commands

```
up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to
up


Router>enable
Router#
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#router rip
Router(config-router)#
```



```
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0.10, changed
state to up

Router(config-subif)#encapsulation dot1Q 10
Router(config-subif)#ip add 138.22.79 255.255.255.240
                                        ^
% Invalid input detected at '^' marker.

Router(config-subif)#ip add 138.39.22.75 255.255.255.240
Router(config-subif)#exit
Router(config)#int fa0/0.100
Router(config-subif)#
%LINK-5-CHANGED: Interface FastEthernet0/0.100, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0.100, changed
state to up

Router(config-subif)#encapsulation dot1Q 100
Router(config-subif)#ip add 138.39.22.91 255.255.255.240
Router(config-subif)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#write memory
Building configuration...
[OK]
Router#
```

## Switch Config:

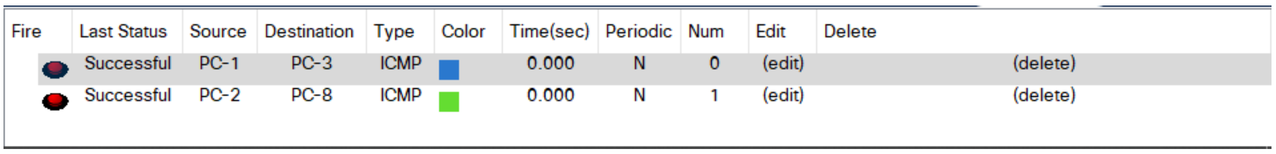


```
%SYS-5-CONFIG_I: Configured from console by console

Switch#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- 
-------------------------------
1    default                          active    Fa4/1, Fa5/1
10   admin                            active    Fa0/1, Fa2/1
100  admin-2                          active    Fa1/1
1002 fddi-default                     active    
1003 token-ring-default               active    
1004 fddinet-default                  active    
1005 trnet-default                    active    

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------
------
1    enet  100001     1500  -      -      -        -    -        0      0
10   enet  100010     1500  -      -      -        -    -        0      0
100  enet  100100     1500  -      -      -        -    -        0      0
1002 fddi  101002     1500  -      -      -        -    -        0      0
1003 tr    101003     1500  -      -      -        -    -        0      0
1004 fdnet 101004     1500  -      -      -        ieee -        0      0
1005 trnet 101005     1500  -      -      -        ibm  -        0      0

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1
Trans2
 --More--
```

**Note:** Both Inter and Intra VLAN communication applied.

## Working Screenshot:

- PC-1 and PC-3 are in same subnet but different vlan
- PC-2 in subnet 1 and PC-8 in subnet 3 both in different vlans



| Fire | Last Status | Source | Destination | Type | Color | Time(sec) | Periodic | Num | Edit | Delete |
|---|---|---|---|---|---|---|---|---|---|---|
| ● | Successful | PC-1 | PC-3 | ICMP | 🟦 | 0.000 | N | 0 | (edit) | (delete) |
| ● | Successful | PC-2 | PC-8 | ICMP | 🟩 | 0.000 | N | 1 | (edit) | (delete) |

## GoBack-N:

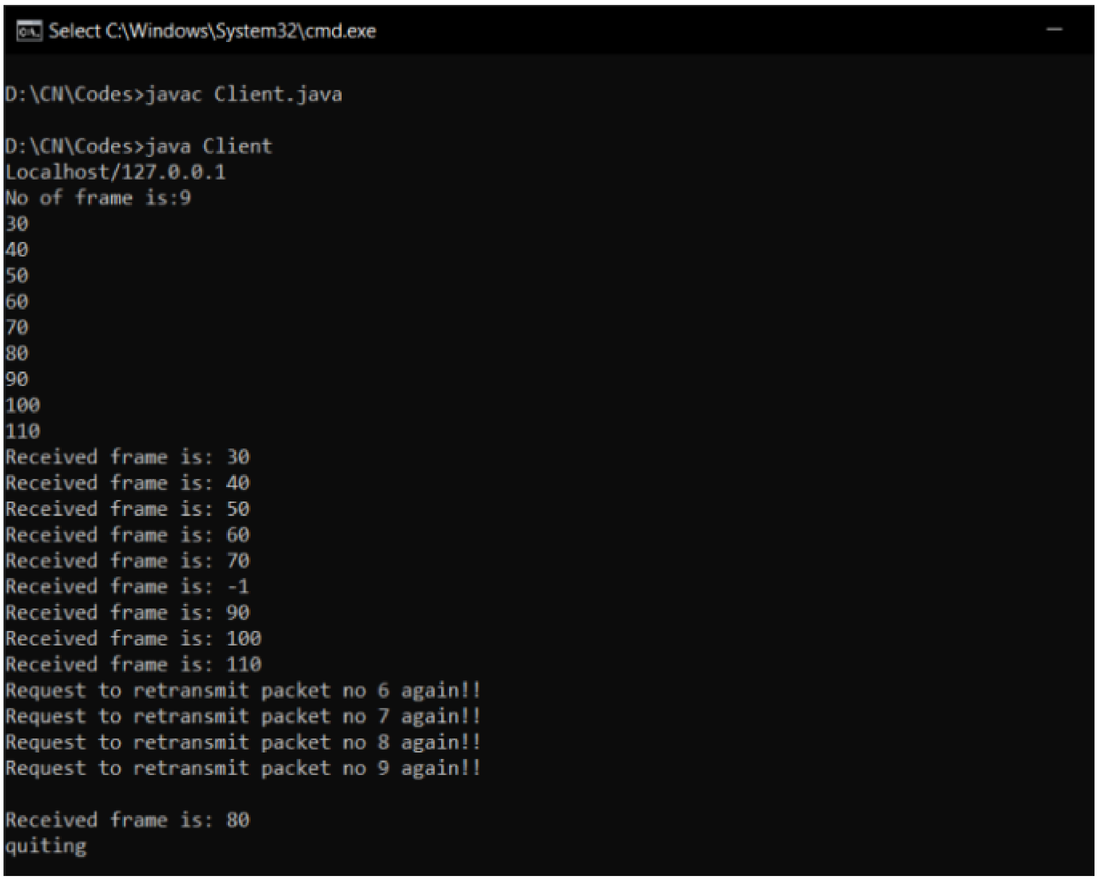
## Server:

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;

public class Server {
    static ServerSocket Serversocket;
    static DataInputStream dis;
    static DataOutputStream dos;

    public static void main(String[] args) throws SocketException {

    try {
        int a[] = { 30, 40, 50, 60, 70, 80, 90, 100, 110 };
        Serversocket = new ServerSocket(8011);
        System.out.println("waiting for connection");
        Socket client = Serversocket.accept();
        dis = new DataInputStream(client.getInputStream());
        dos = new DataOutputStream(client.getOutputStream());
        System.out.println("The number of packets sent is:" + a.length);
        int y = a.length;
        dos.write(y);
        dos.flush();

        for (int i = 0; i < a.length; i++) {
            dos.write(a[i]);
            dos.flush();
        }

        int k = dis.read();

        dos.write(a[k]);
        dos.flush();

        } catch (IOException e) {
            System.out.println(e);
        } finally {
            try {
                dis.close();
                dos.close();
            } catch (IOException e) {
                e.printStackTrace();
            }

        }
    }
}
```

## Client:

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

public class Client {
    static Socket connection;

    public static void main(String a[]) throws SocketException {
        try {
            int v[] = new int[9];
            //int g[] = new int[8];
            int n = 0;
            InetAddress addr = InetAddress.getByName("Localhost");
            System.out.println(addr);
            connection = new Socket(addr, 8011);
            DataOutputStream out = new DataOutputStream(
                    connection.getOutputStream());
            DataInputStream in = new DataInputStream(
                    connection.getInputStream());
            int p = in.read();
            System.out.println("No of frame is:" + p);

            for (int i = 0; i < p; i++) {
                v[i] = in.read();
                System.out.println(v[i]);
                //g[i] = v[i];
            }
            v[5] = -1;
            for (int i = 0; i < p; i++)
             {
                System.out.println("Received frame is: " + v[i]);

                }
            for (int i = 0; i < p; i++)
                if (v[i] == -1) {
                    // comment this section for selective repeat
                    if(i+1<p){
                        v[i+1]=-1;
                    }
                    // till here
            System.out.println("Request to retransmit packet no "
                                + (i+1) + " again!!");
                    n = i;
                    out.write(n);
                    out.flush();
                    }

               System.out.println();

                    v[n] = in.read();
               System.out.println("Received frame is: " + v[n]);



               System.out.println("quiting");
          } catch (Exception e) {
               System.out.println(e);
          }

     }
}
```

## Output:



```
Select C:\Windows\System32\cmd.exe

D:\CN\Codes>javac Client.java

D:\CN\Codes>java Client
Localhost/127.0.0.1
No of frame is:9
30
40
50
60
70
80
90
100
110
Received frame is: 30
Received frame is: 40
Received frame is: 50
Received frame is: 60
Received frame is: 70
Received frame is: -1
Received frame is: 90
Received frame is: 100
Received frame is: 110
Request to retransmit packet no 6 again!!
Request to retransmit packet no 7 again!!
Request to retransmit packet no 8 again!!
Request to retransmit packet no 9 again!!

Received frame is: 80
quiting
```

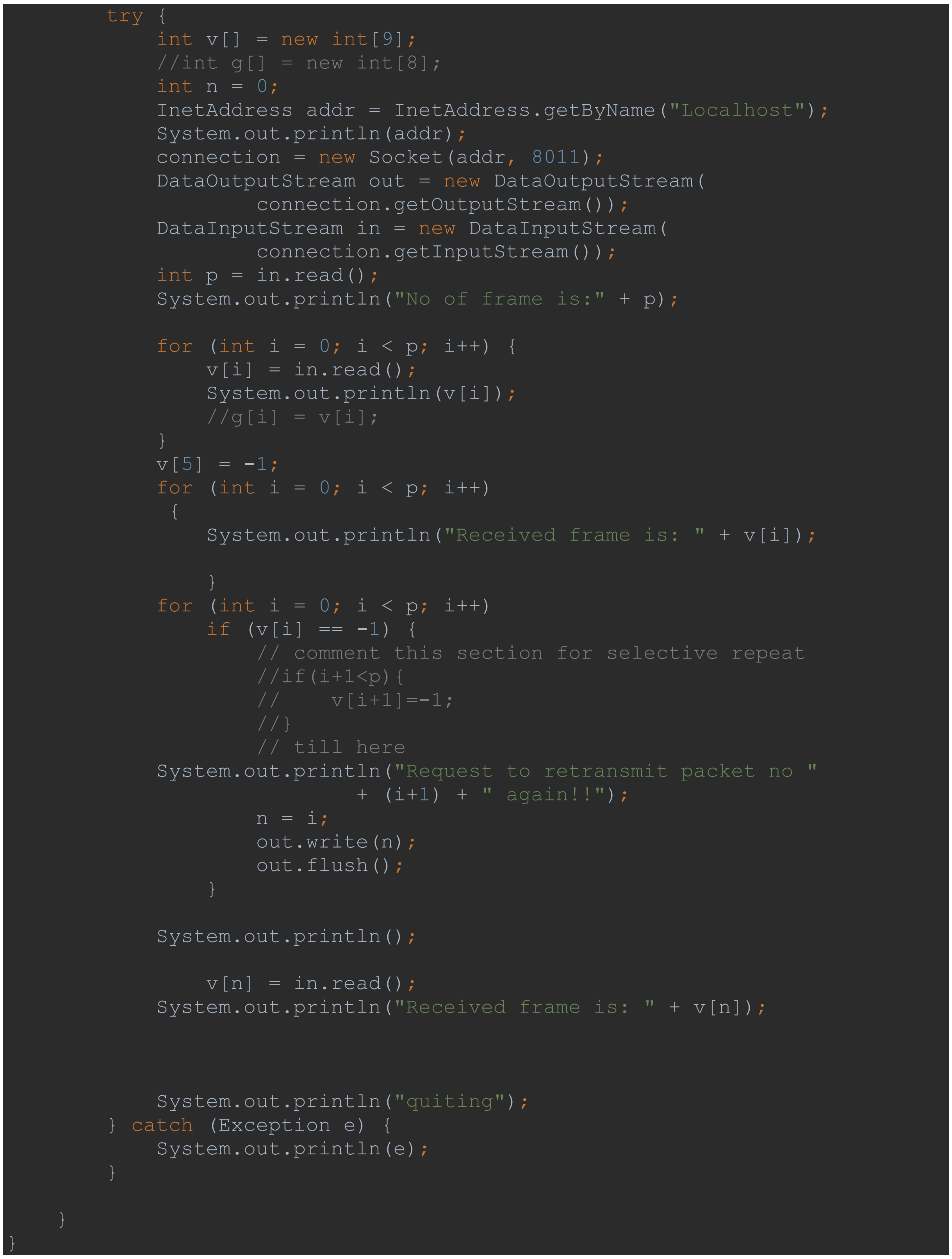
## Selective Repeat:

## Server:

```java
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;

public class Server {
    static ServerSocket Serversocket;
    static DataInputStream dis;
    static DataOutputStream dos;

    public static void main(String[] args) throws SocketException {

    try {
        int a[] = { 30, 40, 50, 60, 70, 80, 90, 100, 110 };
        Serversocket = new ServerSocket(8011);
        System.out.println("waiting for connection");
        Socket client = Serversocket.accept();
        dis = new DataInputStream(client.getInputStream());
        dos = new DataOutputStream(client.getOutputStream());
        System.out.println("The number of packets sent is:" + a.length);
        int y = a.length;
        dos.write(y);
        dos.flush();

        for (int i = 0; i < a.length; i++) {
            dos.write(a[i]);
            dos.flush();
        }

        int k = dis.read();

        dos.write(a[k]);
        dos.flush();

        } catch (IOException e) {
            System.out.println(e);
        } finally {
            try {
                dis.close();
                dos.close();
            } catch (IOException e) {
                e.printStackTrace();
            }

        }
    }
}
```

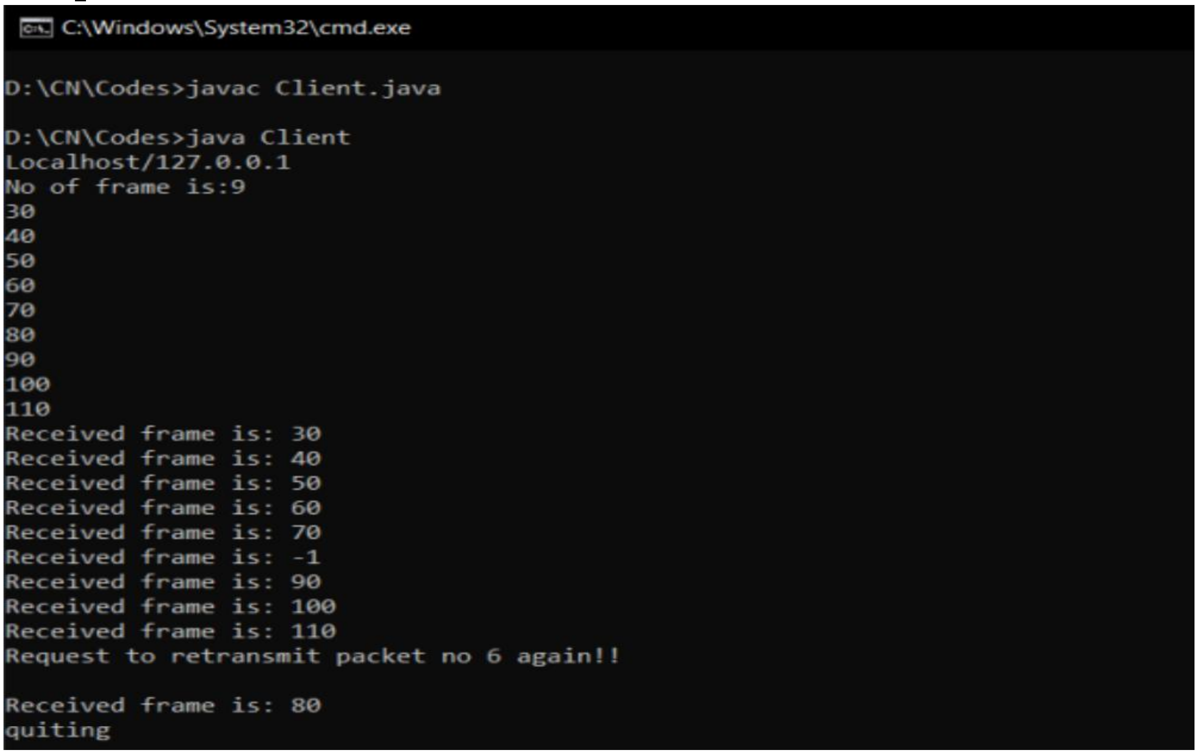
**Client:**

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

public class Client {
    static Socket connection;

    public static void main(String a[]) throws SocketException {
```



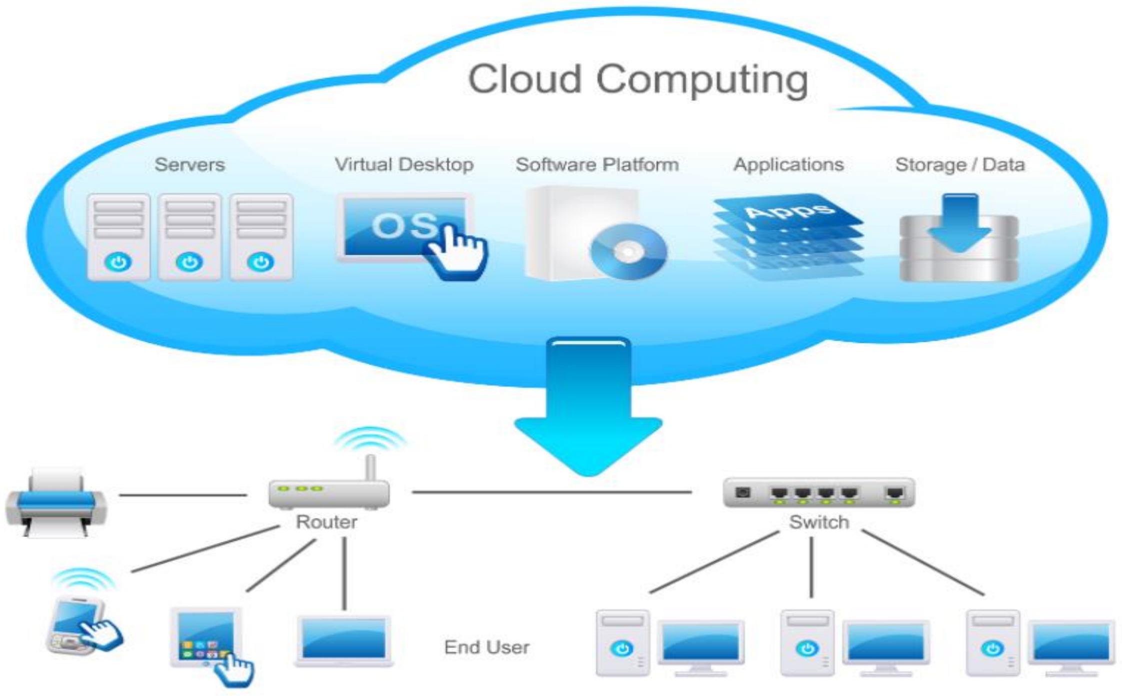
```java
        try {
            int v[] = new int[9];
            //int g[] = new int[8];
            int n = 0;
            InetAddress addr = InetAddress.getByName("Localhost");
            System.out.println(addr);
            connection = new Socket(addr, 8011);
            DataOutputStream out = new DataOutputStream(
                    connection.getOutputStream());
            DataInputStream in = new DataInputStream(
                    connection.getInputStream());
            int p = in.read();
            System.out.println("No of frame is:" + p);

            for (int i = 0; i < p; i++) {
                v[i] = in.read();
                System.out.println(v[i]);
                //g[i] = v[i];
            }
            v[5] = -1;
            for (int i = 0; i < p; i++)
             {
                System.out.println("Received frame is: " + v[i]);

                }
            for (int i = 0; i < p; i++)
                if (v[i] == -1) {
                    // comment this section for selective repeat
                    //if(i+1<p){
                    //    v[i+1]=-1;
                    //}
                    // till here
            System.out.println("Request to retransmit packet no "
                            + (i+1) + " again!!");
                    n = i;
                    out.write(n);
                    out.flush();
                }

            System.out.println();

                v[n] = in.read();
            System.out.println("Received frame is: " + v[n]);



            System.out.println("quiting");
        } catch (Exception e) {
            System.out.println(e);
        }

    }
}
```

## Output:

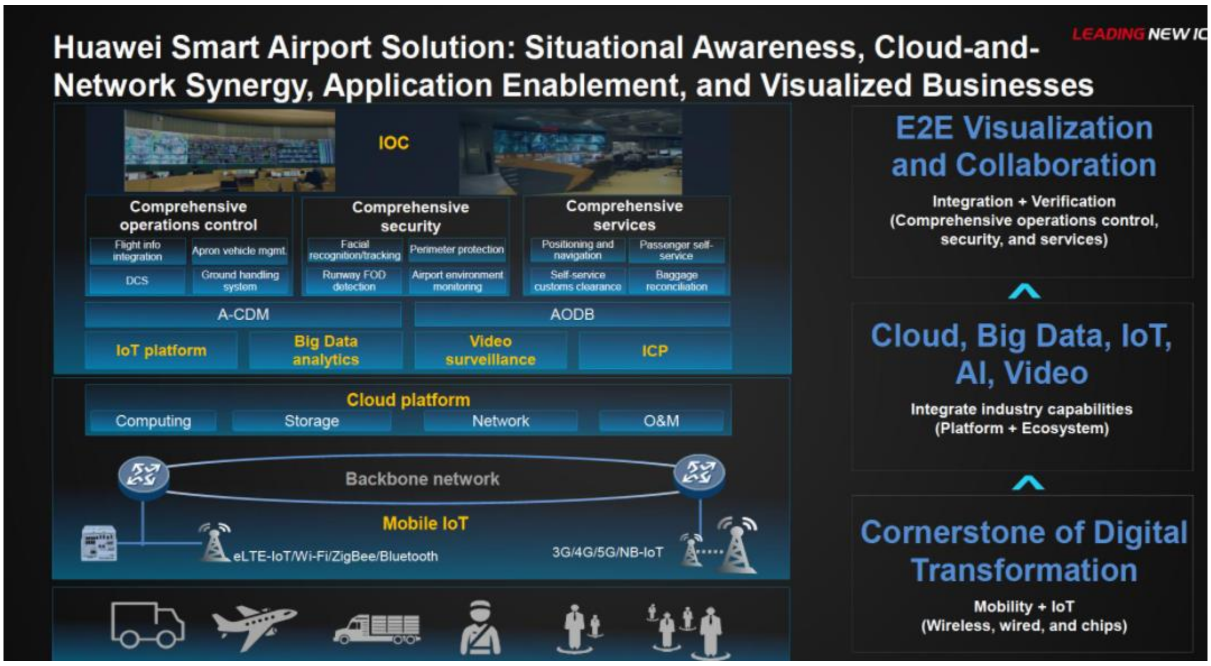


```
C:\Windows\System32\cmd.exe

D:\CN\Codes>javac Client.java

D:\CN\Codes>java Client
Localhost/127.0.0.1
No of frame is:9
30
40
50
60
70
80
90
100
110
Received frame is: 30
Received frame is: 40
Received frame is: 50
Received frame is: 60
Received frame is: 70
Received frame is: -1
Received frame is: 90
Received frame is: 100
Received frame is: 110
Request to retransmit packet no 6 again!!

Received frame is: 80
quiting
```

## Cloud Computing:



Virtualization is a technique of how to separate a service from the underlying physical delivery of that service. It is the process of creating a virtual version of something like computer hardware. It was initially developed during the mainframe era. It involves using specialized software to create a virtual or software-created version of a computing resource rather than the actual version of the same resource. With the help of Virtualization, multiple operating systems and applications can run on same machine and its same hardware at the same time, increasing the utilization and flexibility of hardware.
In other words, one of the main cost effective, hardware reducing, and energy saving techniques used by cloud providers is virtualization. Virtualization allows to share a single physical instance of a resource or an application among multiple customers and organizations at one time. It does this by assigning a logical name to a physical storage and providing a pointer to that physical resource on demand. The term virtualization is often synonymous with hardware virtualization, which plays a fundamental role in efficiently delivering Infrastructure-as-a-Service (IaaS) solutions for cloud computing. Moreover, virtualization technologies provide a virtual environment for not only executing applications but also for storage, memory, and networking.



Huawei Smart Airport Solution: Situational Awareness, Cloud-and-Network Synergy, Application Enablement, and Visualized Businesses
LEADING NEW IC
IOC
Comprehensive operations control
Flight info integration
Apron vehicle mgmt.
DCS
Ground handling system
Comprehensive security
Facial recognition/tracking
Perimeter protection
Runway FOD detection
Airport environment monitoring
Comprehensive services
Positioning and navigation
Passenger self-service
Self-service customs clearance
Baggage reconciliation
A-CDM
AODB
IoT platform
Big Data analytics
Video surveillance
ICP
Cloud platform
Computing
Storage
Network
O&M
Backbone network
Mobile IoT
eLTE-IoT/Wi-Fi/ZigBee/Bluetooth
3G/4G/5G/NB-IoT
E2E Visualization and Collaboration
Integration + Verification (Comprehensive operations control, security, and services)
Cloud, Big Data, IoT, AI, Video
Integrate industry capabilities (Platform + Ecosystem)
Cornerstone of Digital Transformation
Mobility + IoT (Wireless, wired, and chips)

**BENEFITS OF VIRTUALIZATION:**

1. More flexible and efficient allocation of resources.
2. Enhance development productivity.
3. It lowers the cost of IT infrastructure.
4. Remote access and rapid scalability.

5. High availability and disaster recovery.
6. Pay peruse of the IT infrastructure on demand.
7. Enables running multiple operating systems.

**Types of Virtualizations:**

1. **Application Virtualization:**
   Application virtualization helps a user to have remote access of an application from a server. The server stores all personal information and other characteristics of the application but can still run on a local workstation through the internet. Example of this would be a user who needs to run two different versions of the same software. Technologies that use application virtualization are hosted applications and packaged applications.

2. **Network Virtualization:**
   The ability to run multiple virtual networks with each has a separate control and data plan. It co-exists together on top of one physical network. It can be managed by individual parties that potentially confidential to each other. Network virtualization provides a facility to create and provision virtual networks—logical switches, routers, firewalls, load balancer, Virtual Private Network (VPN), and workload security within days or even in weeks.

3. **Desktop Virtualization:**
   Desktop virtualization allows the users' OS to be remotely stored on a server in the data centre. It allows the user to access their desktop virtually, from any location by a different machine. Users who want specific operating systems other than Windows Server will need to have a virtual desktop. Main benefits of desktop virtualization are user mobility, portability, easy management of software installation, updates, and patches.

4. **Storage Virtualization:**
   Storage virtualization is an array of servers that are managed by a virtual storage system. The servers are not aware of exactly where their data is stored, and instead function more like worker bees in a hive. It makes managing storage from multiple sources to be managed and utilized as a single repository. storage virtualization software maintains smooth operations, consistent performance, and a continuous suite of advanced

functions despite changes, break down and differences in the underlying equipment.

5. **Server Virtualization:**
   This is a kind of virtualization in which masking of server resources takes place. Here, the central-server (physical server) is divided into multiple different virtual servers by changing the identity number, processors. So, each system can operate its own operating systems in isolate manner. Where each sub-server knows the identity of the central server. It causes an increase in the performance and reduces the operating cost by the deployment of main server resources into a sub-server resource. It is beneficial in virtual migration, reduce energy consumption, reduce infrastructural cost, etc.

6. **Data virtualization:**
   This is the kind of virtualization in which the data is collected from various sources and managed that at a single place without knowing more about the technical information like how data is collected, stored & formatted then arranged that data logically so that its virtual view can be accessed by its interested people and stakeholders, and users through the various cloud services remotely. Many big giant companies are providing their services like Oracle, IBM, At scale, Cdata, etc.

**Use of Cloud:**

Main use of cloud in the Ticket Management department and Flight Department is highly dependent in our case study. This is where the significance of cloud kicks in, the frequent updating of flight timings and ticket booking to verify accordingly to keep tract of flight times. so, the whole updating process can be taken place in cloud storage which can reduce the bare-metal servers to some extent and results in seamless experience because bare-metal servers can have downtimes whereas cloud storage (ex. Azure, GCP) has no down times at all because they create multiple instances of data in different location so that if one instance is experiencing down time due to some technical reasons the other instances will full-fill our request. so, it is highly available. The other scope for cloud is to store all the bills and make it publicly available for customers this is closely impossible when we maintain bare-metal servers because storing data at that scale needs lot of bare metal servers and then making it publicly accessible is highly risky.