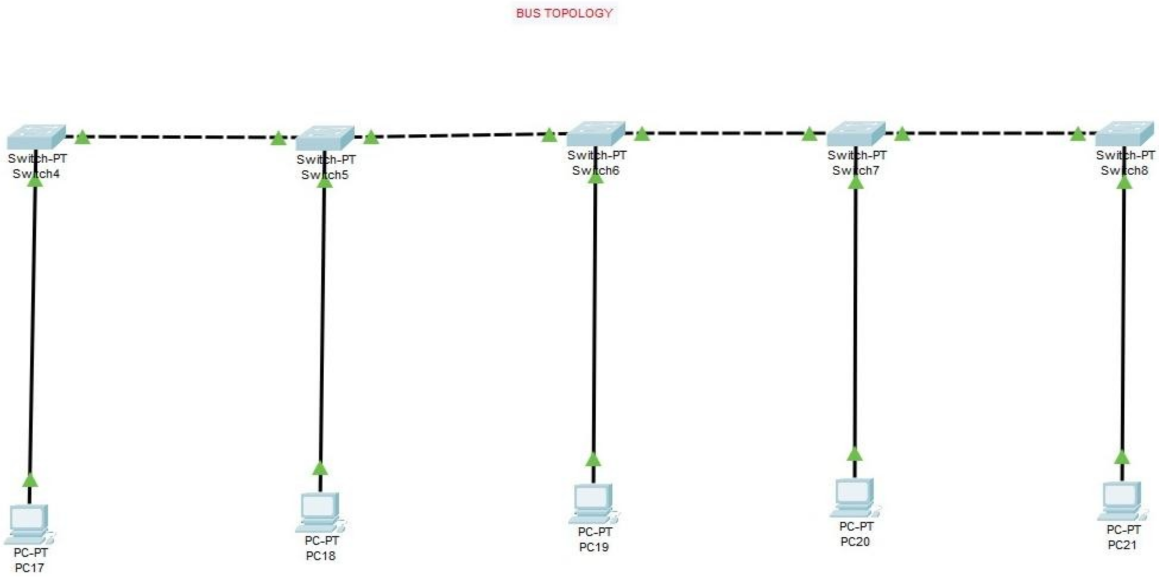
Name : Saish Baviskar
Roll No : TEAD23155
Division : A
Dept : TE (AI&DS)
Subject : Computer Networks Lab

Practical No :- 01

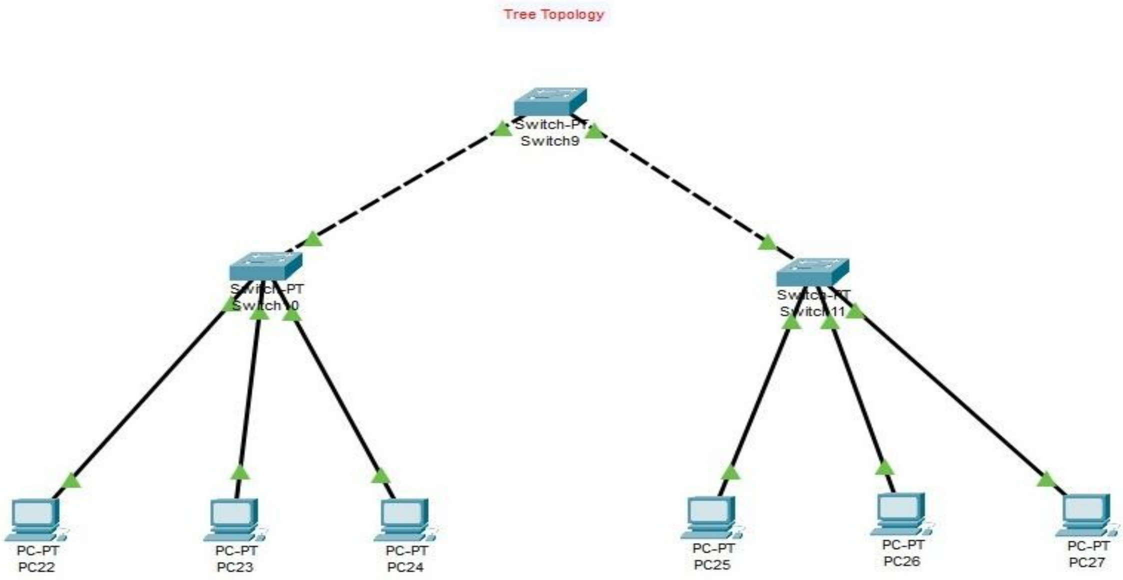
Problem Statement:

Demonstrate the different types of topologies and types of transmission media by using a packet tracer tool.

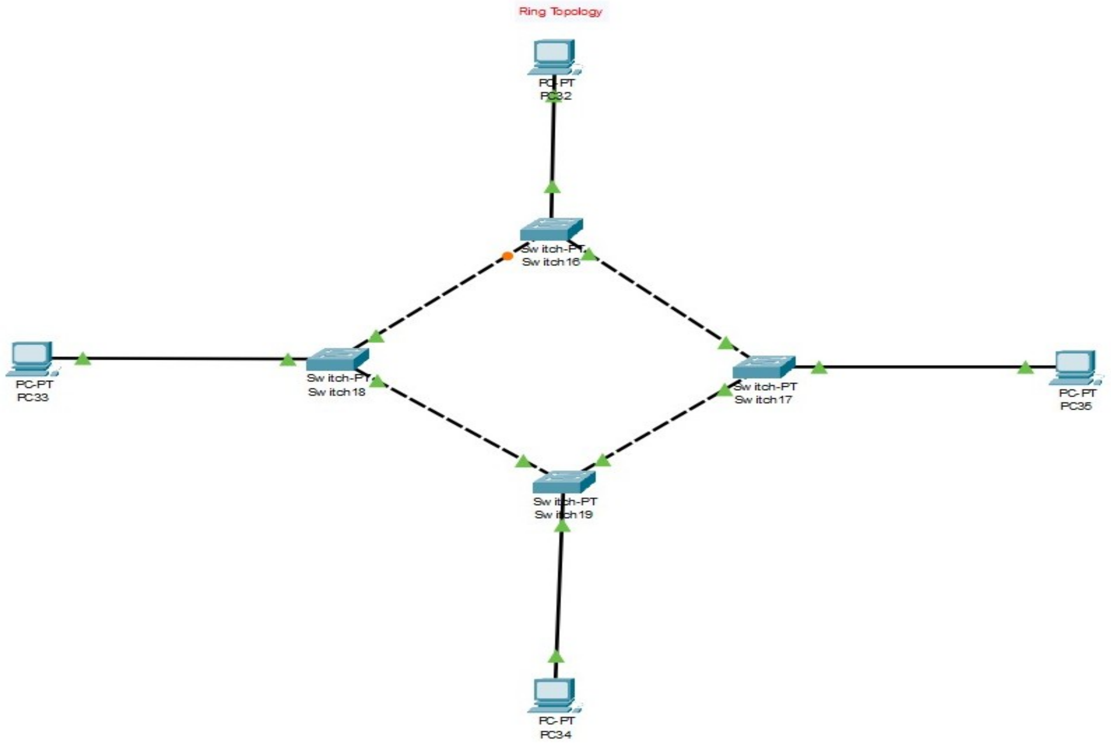
1. Bus Topology :



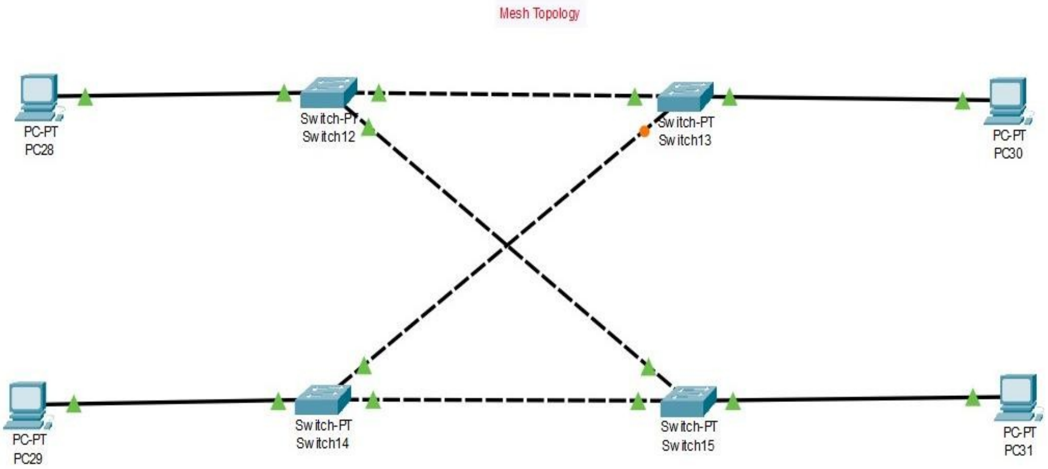
2. Tree Topology :



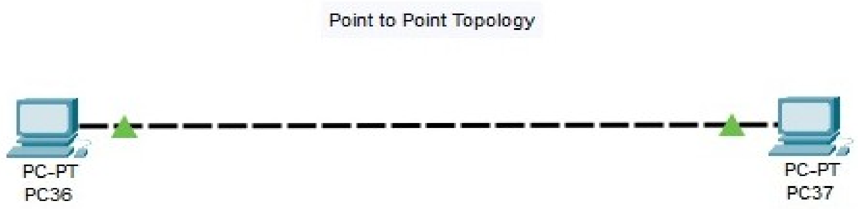
3.Ring Topology :



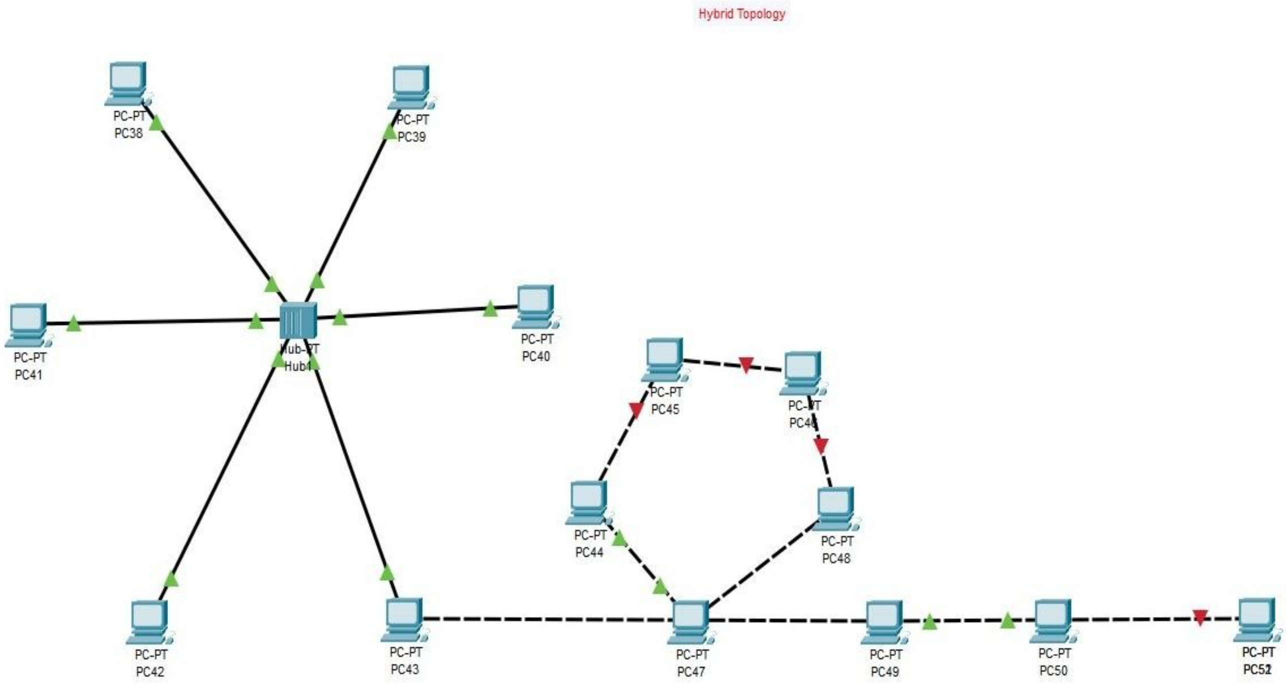
4. Mesh Topology :



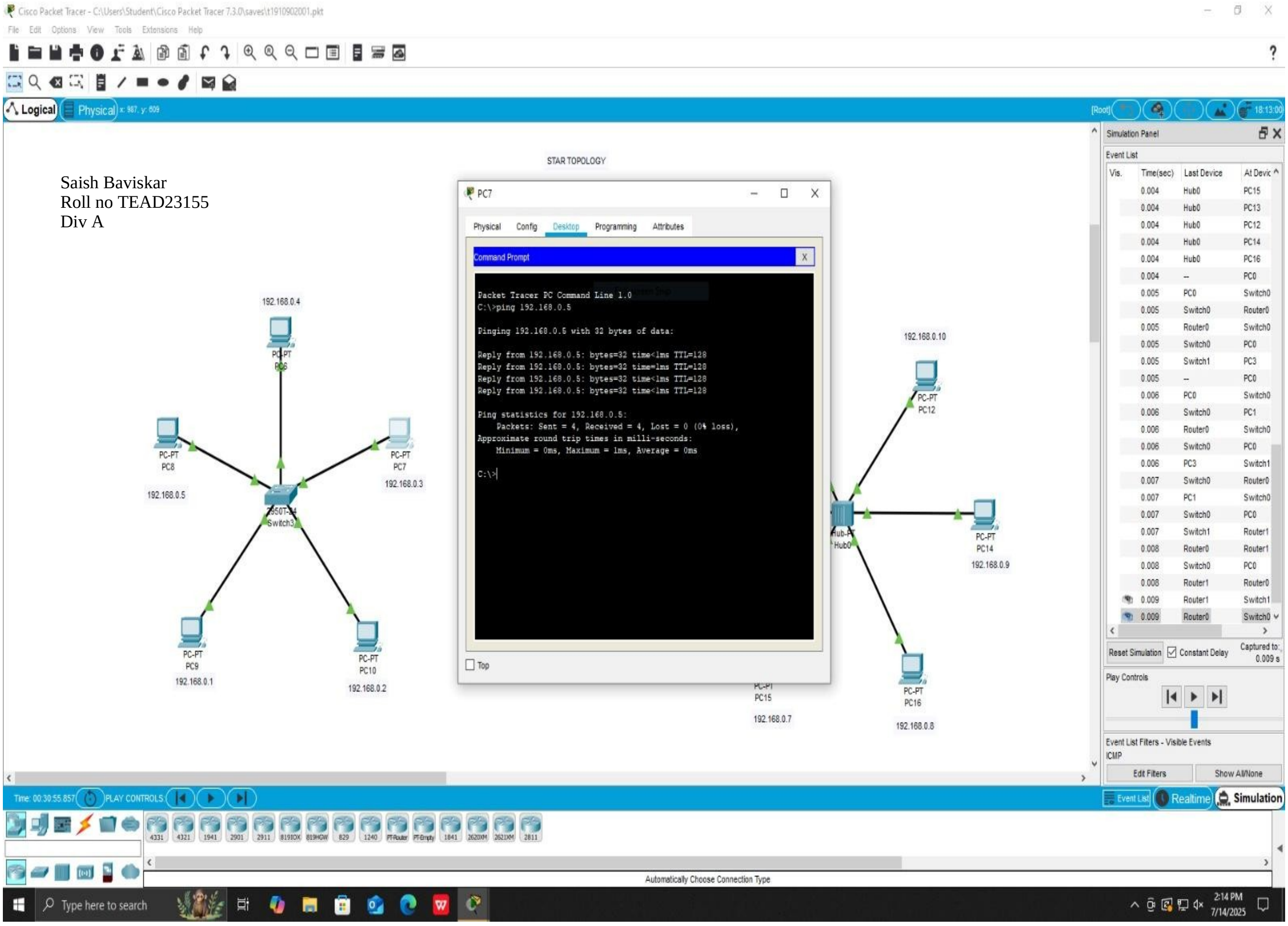
5.Point to Point Topology :



6.Hybrid Topology :



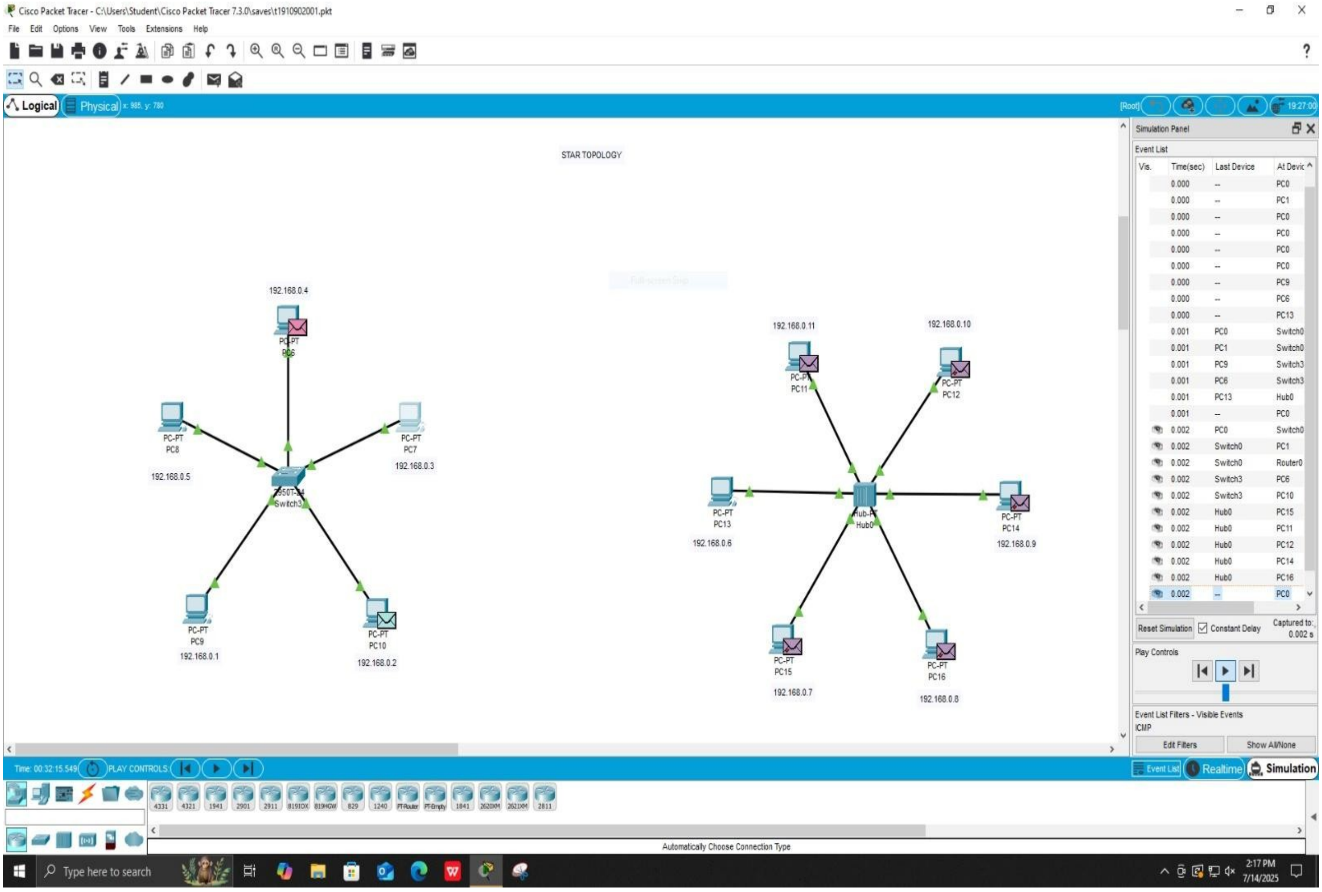
7.Star Topology :



Saish Baviskar
Roll no TEAD23155
Div A

```
Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.5

Pinging 192.168.0.5 with 32 bytes of data:

Reply from 192.168.0.5: bytes=32 time<1ms TTL=128
Reply from 192.168.0.5: bytes=32 time=1ms TTL=128
Reply from 192.168.0.5: bytes=32 time<1ms TTL=128
Reply from 192.168.0.5: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>
```

Vis.	Time(sec)	Last Device	At Devic
	0.004	Hub0	PC15
	0.004	Hub0	PC13
	0.004	Hub0	PC12
	0.004	Hub0	PC14
	0.004	Hub0	PC16
	0.004	--	PC0
	0.005	PC0	Switch0
	0.005	Switch0	Router0
	0.005	Router0	Switch0
	0.005	Switch0	PC0
	0.005	Switch1	PC3
	0.005	--	PC0
	0.006	PC0	Switch0
	0.006	Switch0	PC1
	0.006	Router0	Switch0
	0.006	Switch0	PC0
	0.006	PC3	Switch1
	0.007	Switch0	Router0
	0.007	PC1	Switch0
	0.007	Switch0	PC0
	0.007	Switch1	Router1
	0.008	Router0	Router1
	0.008	Switch0	PC0
	0.008	Router1	Router0
	0.009	Router1	Switch1
	0.009	Router0	Switch0



Event List

Vis.	Time(sec)	Last Device	At Devic
	0.000	--	PC0
	0.000	--	PC1
	0.000	--	PC0
	0.000	--	PC0
	0.000	--	PC0
	0.000	--	PC0
	0.000	--	PC9
	0.000	--	PC6
	0.000	--	PC13
	0.001	PC0	Switch0
	0.001	PC1	Switch0
	0.001	PC9	Switch3
	0.001	PC6	Switch3
	0.001	PC13	Hub0
	0.001	--	PC0
	0.002	PC0	Switch0
	0.002	Switch0	PC1
	0.002	Switch0	Router0
	0.002	Switch3	PC6
	0.002	Switch3	PC10
	0.002	Hub0	PC15
	0.002	Hub0	PC11
	0.002	Hub0	PC12
	0.002	Hub0	PC14
	0.002	Hub0	PC16
	0.002	--	PC0

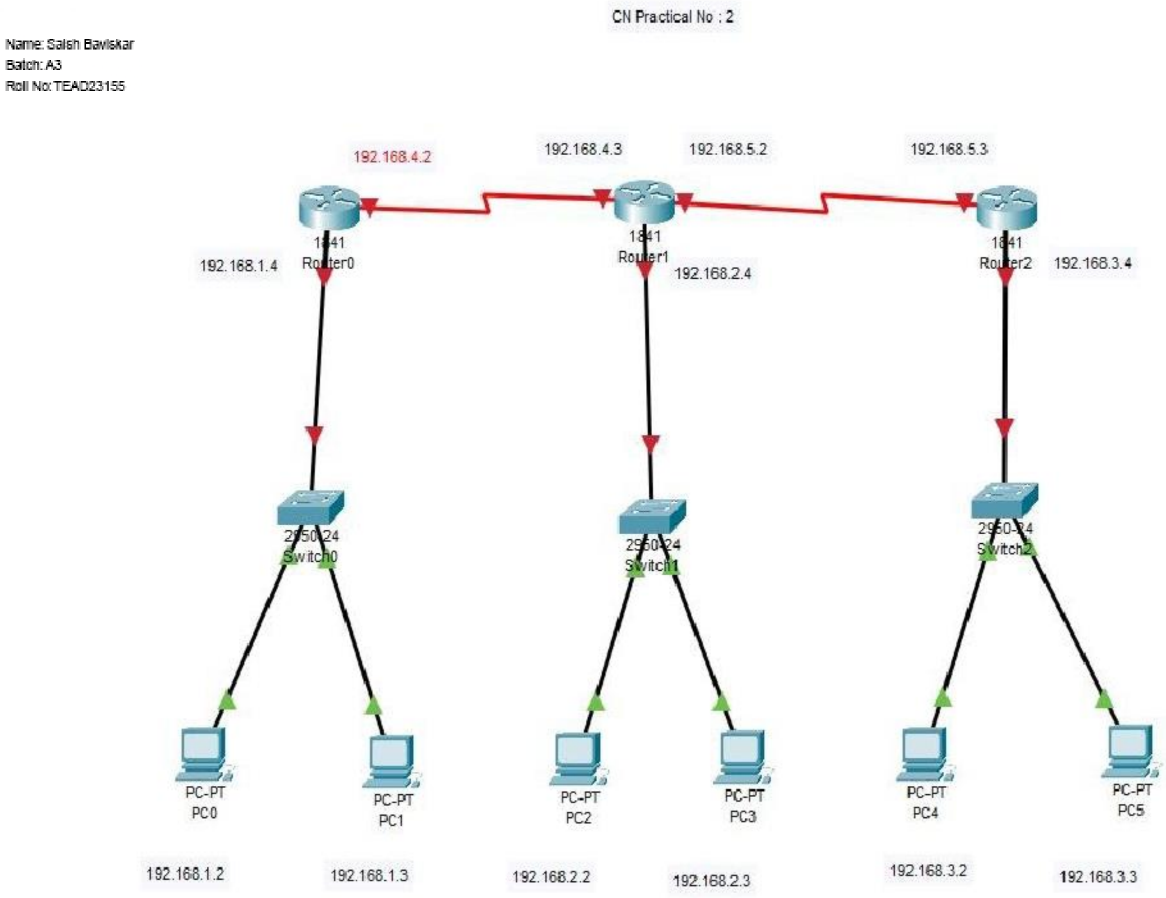
Name :Saish Baviskar
Roll No : TEAD23155
Division : B
Dept : TE (AI&DS)
Subject : Computer Networks Lab

Practical No :- 02

Problem Statement:

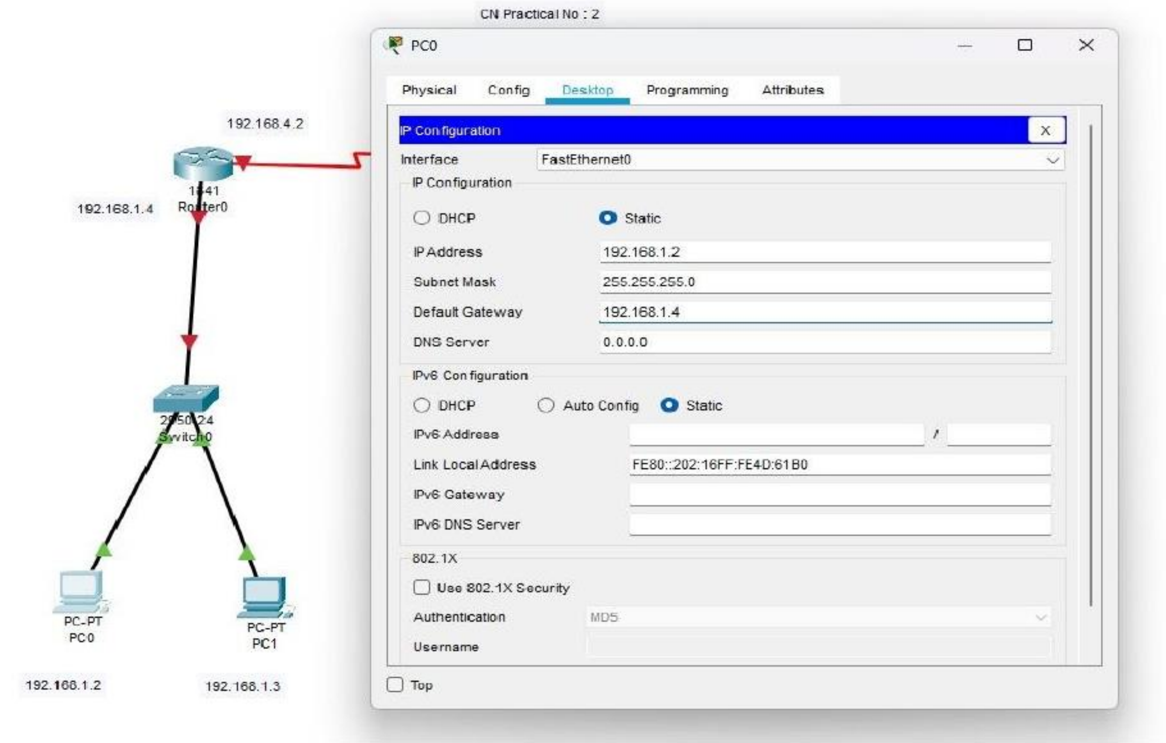
Use packet Tracer tool for confguraton of 3 router networks using one of the following protocols RIP/OSPF/BGP.

- Step 1 - Building Topology :



- Step 2 - Confgure the PCs :

1.IP Confguraton of PC-0:



PC0

Physical Config Desktop Programming Attributes

IP Configuration

Interface FastEthernet0

IP Configuration

○ DHCP ● Static

IP Address 192.168.1.2

Subnet Mask 255.255.255.0

Default Gateway 192.168.1.4

DNS Server 0.0.0.0

IPv6 Configuration

○ DHCP ○ Auto Config ● Static

IPv6 Address /

Link Local Address FE80::202:16FF:FE4D:61B0

IPv6 Gateway

IPv6 DNS Server

802.1X

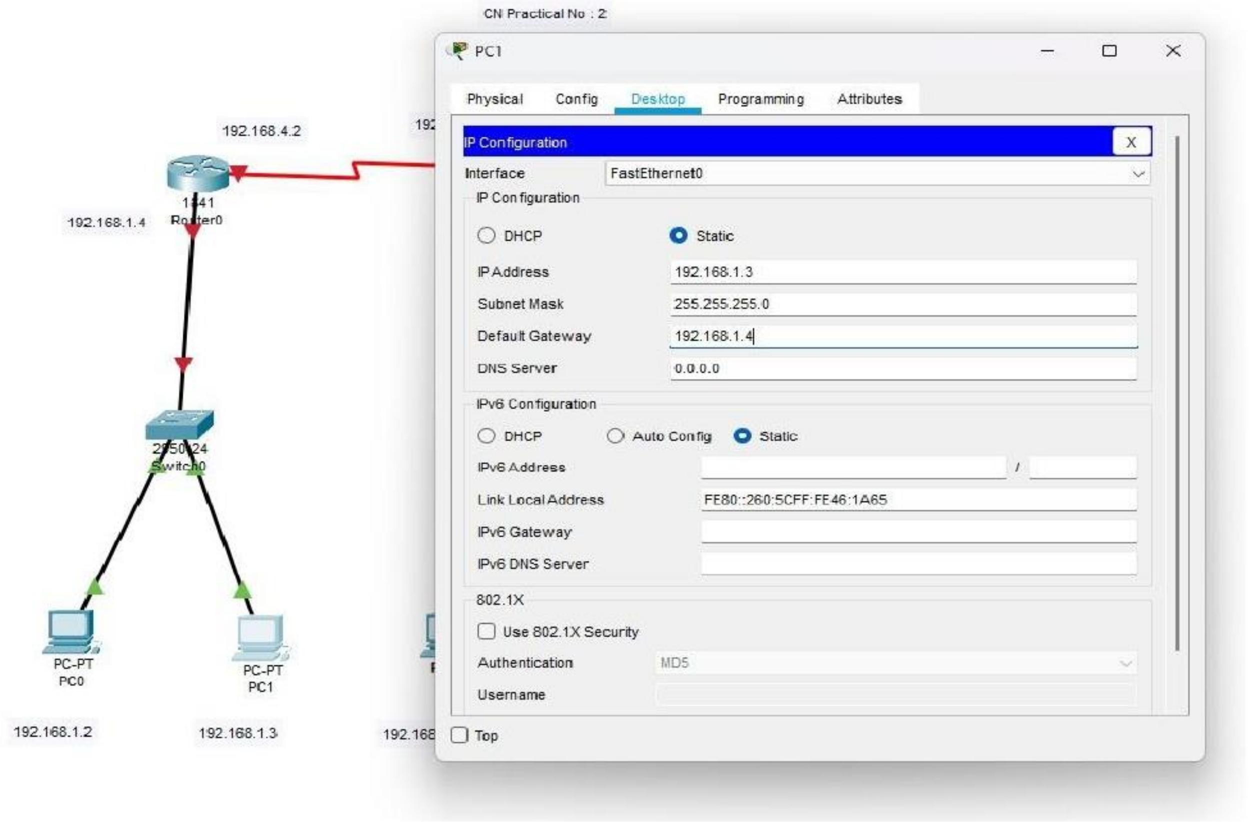
□ Use 802.1X Security

Authentication MD5

Username

□ Top

2.IP Confguraton of PC-1:



PC1

Physical Config Desktop Programming Attributes

IP Configuration

Interface FastEthernet0

IP Configuration

○ DHCP ● Static

IP Address 192.168.1.3

Subnet Mask 255.255.255.0

Default Gateway 192.168.1.4

DNS Server 0.0.0.0

IPv6 Configuration

○ DHCP ○ Auto Config ● Static

IPv6 Address /

Link Local Address FE80::260:5CFF:FE46:1A65

IPv6 Gateway

IPv6 DNS Server

802.1X

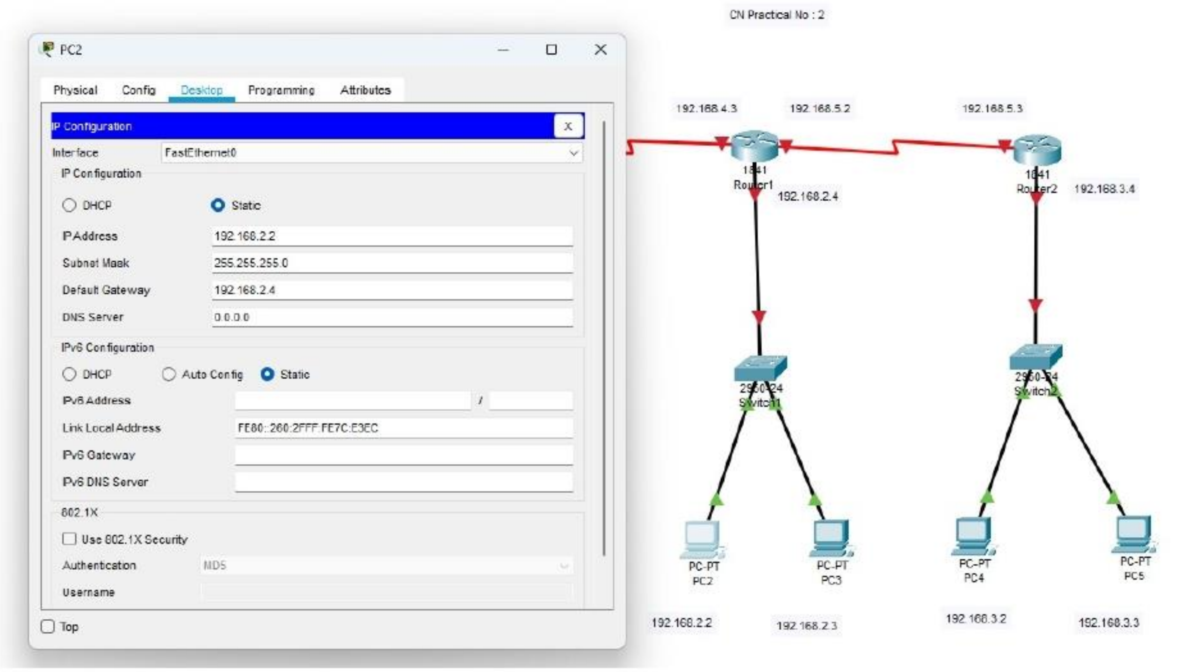
□ Use 802.1X Security

Authentication MD5

Username

□ Top

3.IP Confguraton of PC-2 :



Interface: FastEthernet0

IP Configuration

○ DHCP ● Static

IP Address: 192.168.2.2

Subnet Mask: 255.255.255.0

Default Gateway: 192.168.2.4

DNS Server: 0.0.0.0

IPv6 Configuration

○ DHCP ○ Auto Config ● Static

IPv6 Address: /

Link Local Address: FE80::260:2FFF:FE7C:E3EC

IPv6 Gateway:

IPv6 DNS Server:

802.1X

□ Use 802.1X Security

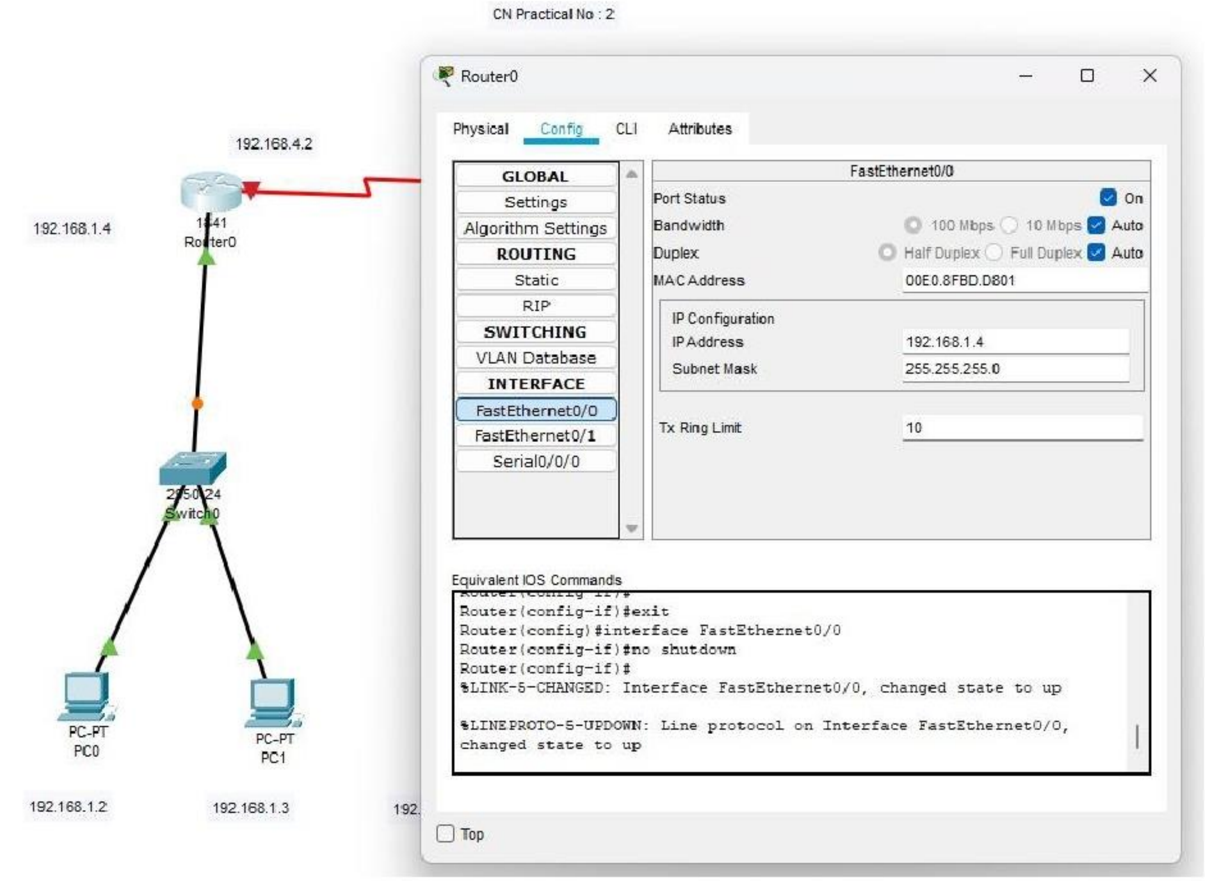
Authentication: MD5

Username:

□ Top

- Step 3 - Confgure the Routers :

1.IP Confguraton of Router-0 :



FastEthernet0/0

Port Status: ☑ On

Bandwidth: ○ 100 Mbps ○ 10 Mbps ☑ Auto

Duplex: ○ Half Duplex ○ Full Duplex ☑ Auto

MAC Address: 00E0.8FBD.D801

IP Configuration

IP Address: 192.168.1.4

Subnet Mask: 255.255.255.0

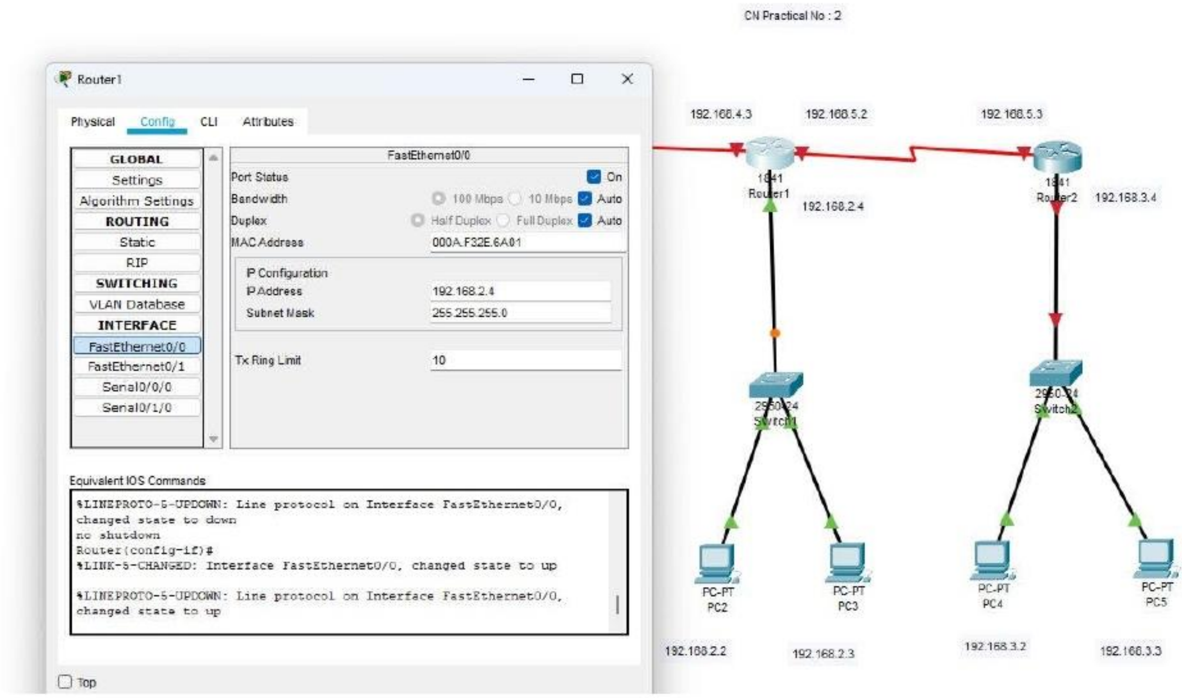
Tx Ring Limit: 10

Equivalent IOS Commands

```
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0,
changed state to up
```

□ Top

2.IP Confguraton of Router-1 :



Router1

FastEthernet0/0

Port Status ☑ On

Bandwidth ○ 100 Mbps ○ 10 Mbps ☑ Auto

Duplex ○ Half Duplex ○ Full Duplex ☑ Auto

MAC Address 000A.F32E.6A01

IP Configuration

IP Address 192.168.2.4

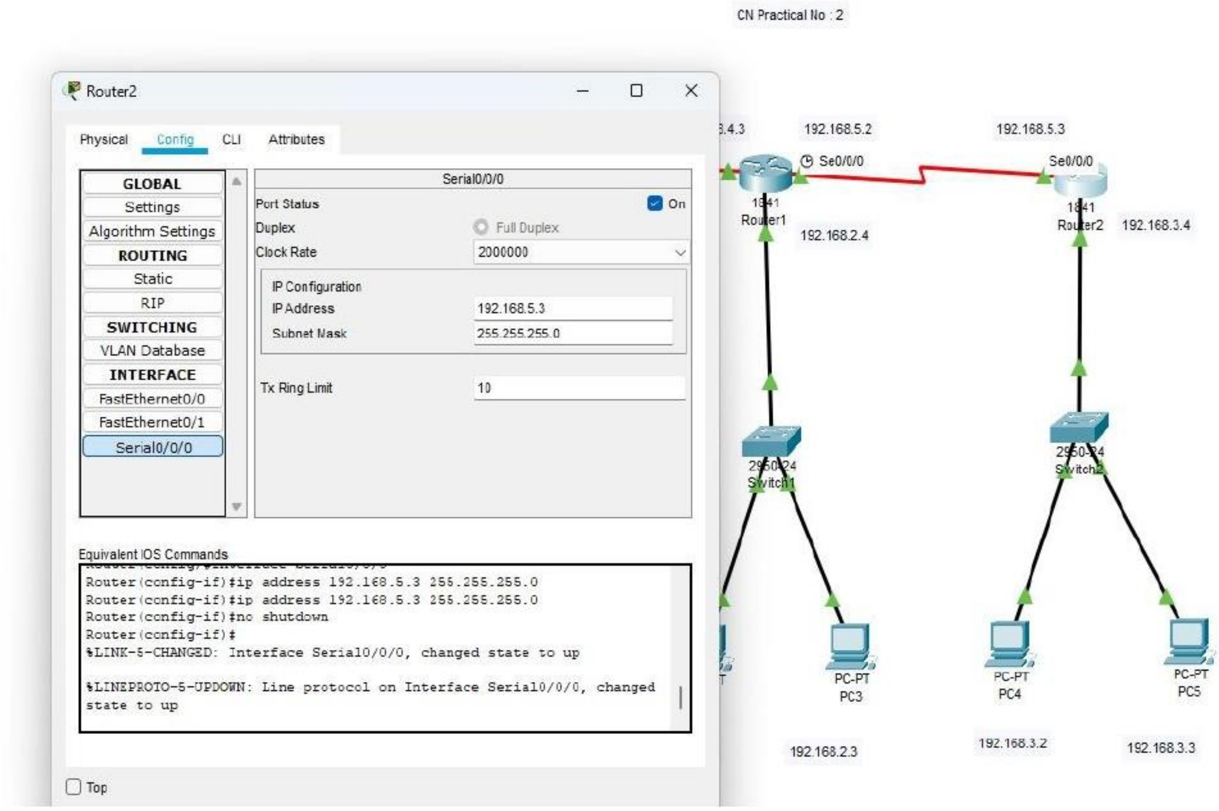
Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0,
changed state to down
no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0,
changed state to up
```

3.IP Confguraton of Router-2 :



Router2

Serial0/0/0

Port Status ☑ On

Duplex ○ Full Duplex

Clock Rate 2000000

IP Configuration

IP Address 192.168.5.3

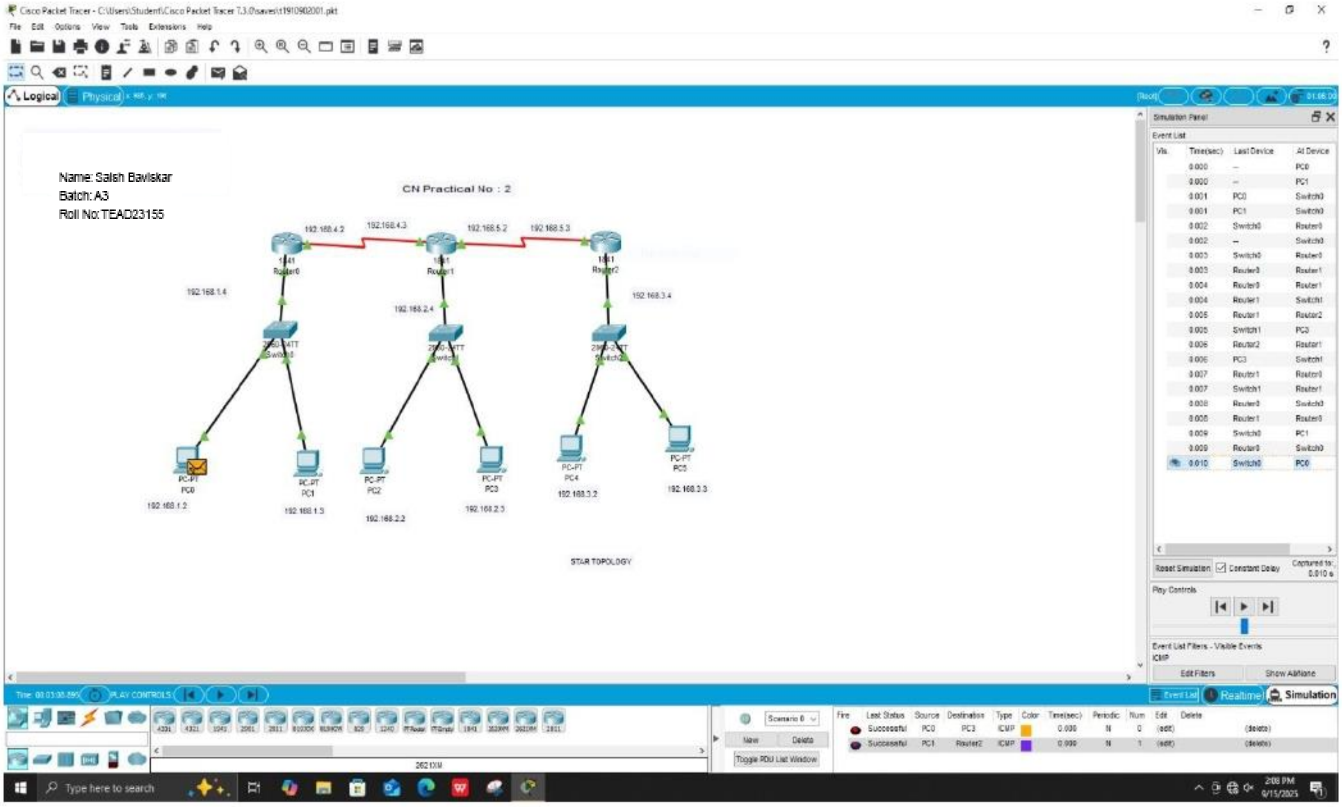
Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
Router(config-if)#ip address 192.168.5.3 255.255.255.0
Router(config-if)#ip address 192.168.5.3 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface Serial0/0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0/0, changed
state to up
```

- Step 4 - Send the packets from one PC to another :



Cisco Packet Tracer - C:\Users\Student\Cisco Packet Tracer 7.3.0\saves\t1910902001.pkt

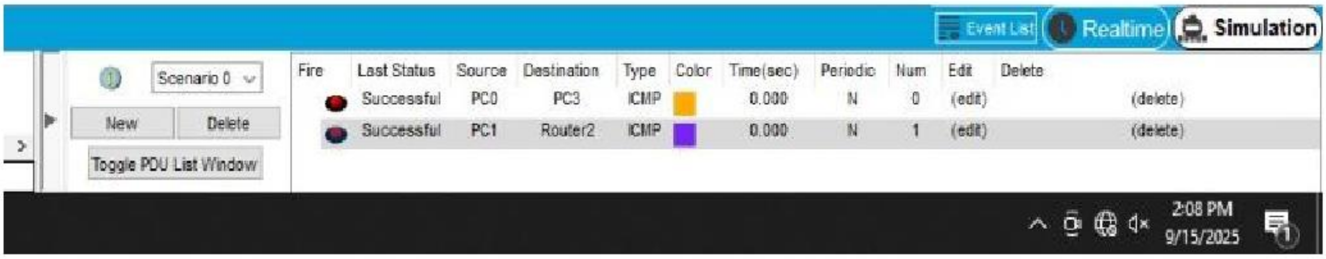
Name: Saish Bavlskar
Batch: A3
Roll No: TEAD23155

CN Practical No : 2

Vis.	Time(sec)	Last Device	At Device
	0.000	--	PC0
	0.000	--	PC1
	0.001	PC0	Switch0
	0.001	PC1	Switch0
	0.002	Switch0	Router0
	0.002	--	Switch0
	0.003	Switch0	Router0
	0.003	Router0	Router1
	0.004	Router0	Router1
	0.004	Router1	Switch1
	0.005	Router1	Router2
	0.005	Switch1	PC3
	0.006	Router2	Router1
	0.006	PC3	Switch1
	0.007	Router1	Router0
	0.007	Switch1	Router1
	0.008	Router0	Switch0
	0.008	Router1	Router0
	0.009	Switch0	PC1
	0.009	Router0	Switch0
	0.010	Switch0	PC0

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC3	ICMP	🟧	0.000	N	0	(edit)	(delete)
●	Successful	PC1	Router2	ICMP	🟪	0.000	N	1	(edit)	(delete)

- Result of packet transfer from PC-0 to PC-3 :



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC3	ICMP	🟧	0.000	N	0	(edit)	(delete)
●	Successful	PC1	Router2	ICMP	🟪	0.000	N	1	(edit)	(delete)

Name : SAISH BAVISKAR

Roll No : TEAD23155

Division : A

Dept : TE (AI&DS)

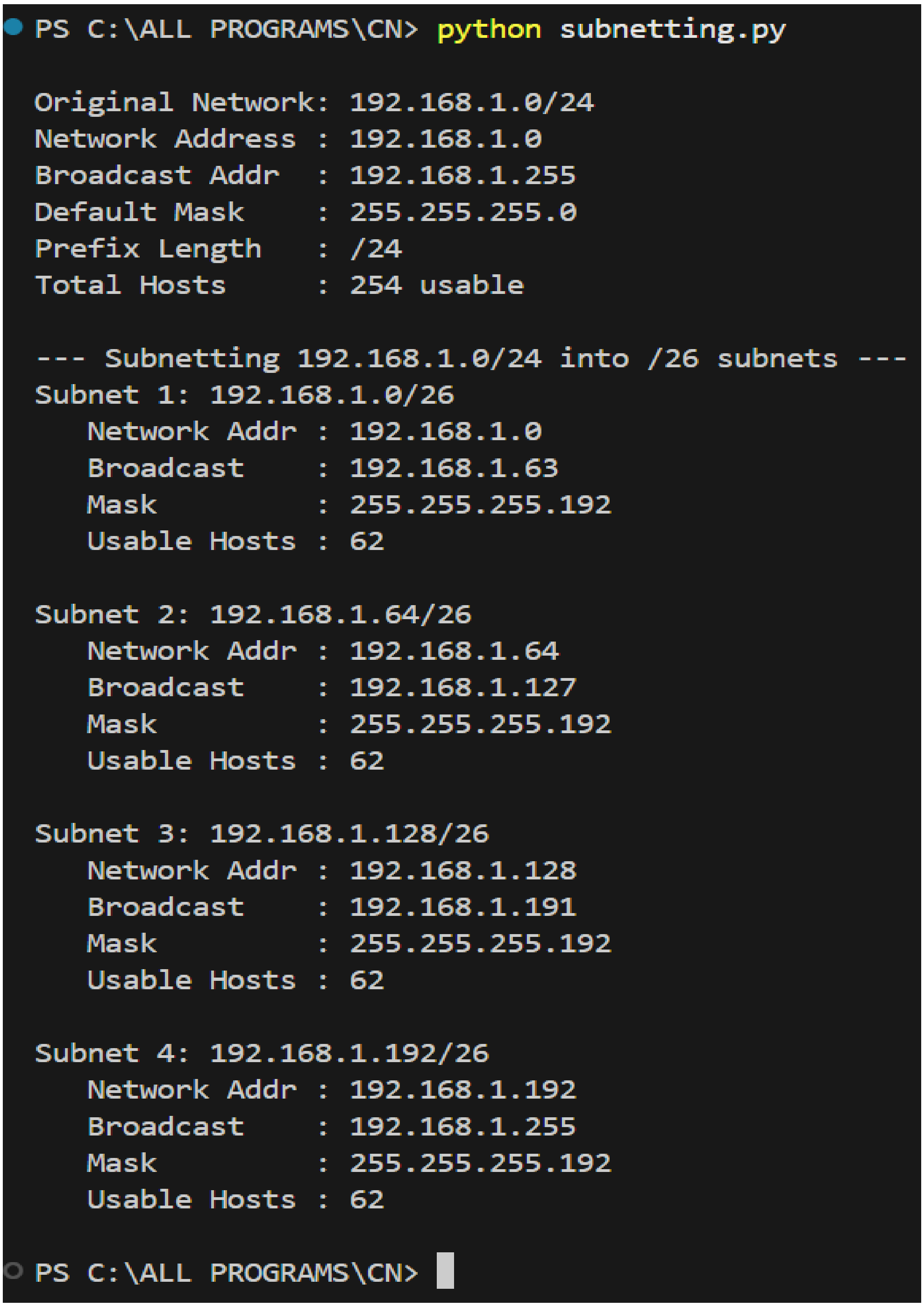
Subject : Computer Networks Lab

Problem Statement : 3 . Write a program to demonstrate Sub-netting and find subnet masks.

Code

```
import ipaddress

def demonstrate_subnetting(network_cidr, new_prefix):

    try:

        # Create network object

        network = ipaddress.ip_network(network_cidr, strict=False)

        print(f"\nOriginal Network: {network}")

        print(f"Network Address : {network.network_address}")

        print(f"Broadcast Addr  : {network.broadcast_address}")

        print(f"Default Mask    : {network.netmask}")

        print(f"Prefix Length   : /{network.prefixlen}")

        print(f"Total Hosts     : {network.num_addresses - 2} usable\n")


        print(f"--- Subnetting {network} into /{new_prefix} subnets ---")

        subnets = list(network.subnets(new_prefix=new_prefix))

        for i, subnet in enumerate(subnets, start=1):

            print(f"Subnet {i}: {subnet}")

            print(f"   Network Addr : {subnet.network_address}")

            print(f"   Broadcast    : {subnet.broadcast_address}")

            print(f"   Mask         : {subnet.netmask}")

            print(f"   Usable Hosts : {subnet.num_addresses - 2}\n")
    except Exception as e:
        print(f"Error: {e}")
if __name__ == "__main__":
    demonstrate_subnetting("192.168.1.0/24", 26)
```

Output



```
PS C:\ALL PROGRAMS\CN> python subnetting.py

Original Network: 192.168.1.0/24
Network Address : 192.168.1.0
Broadcast Addr  : 192.168.1.255
Default Mask    : 255.255.255.0
Prefix Length   : /24
Total Hosts     : 254 usable

--- Subnetting 192.168.1.0/24 into /26 subnets ---
Subnet 1: 192.168.1.0/26
   Network Addr : 192.168.1.0
   Broadcast    : 192.168.1.63
   Mask         : 255.255.255.192
   Usable Hosts : 62

Subnet 2: 192.168.1.64/26
   Network Addr : 192.168.1.64
   Broadcast    : 192.168.1.127
   Mask         : 255.255.255.192
   Usable Hosts : 62

Subnet 3: 192.168.1.128/26
   Network Addr : 192.168.1.128
   Broadcast    : 192.168.1.191
   Mask         : 255.255.255.192
   Usable Hosts : 62

Subnet 4: 192.168.1.192/26
   Network Addr : 192.168.1.192
   Broadcast    : 192.168.1.255
   Mask         : 255.255.255.192
   Usable Hosts : 62

PS C:\ALL PROGRAMS\CN>
```

Name : SAISH BAVISKAR

Roll No : TEAD23155

Division : A

Dept : TE (AI&DS)

Subject : Computer Networks Lab

Problem Statement : 4. Write a program to implement link state /Distance vector routing protocol to find a

suitable path for transmission.

CODE_1

```
# distance_vector.py


import json

import sys

from copy import deepcopy


INFINITY = 10**9


def load_topology(path):

    with open(path, "r") as f:

        topo = json.load(f)

    nodes = topo["nodes"]

    edges = topo["links"]

    graph = {n: {} for n in nodes}

    for u, v, w in edges:

        graph[u][v] = w

        graph[v][u] = w

    return nodes, graph


def initialize_tables(nodes, graph):

    # distance vector tables: dist[node][dest] = cost, next_hop[node][dest] = next-hop

    dist = {n: {m: INFINITY for m in nodes} for n in nodes}

    next_hop = {n: {m: None for m in nodes} for n in nodes}
    for n in nodes:
        dist[n][n] = 0
    for u in nodes:
        for v, w in graph[u].items():
            dist[u][v] = w
            next_hop[u][v] = v
    return dist, next_hop

def simulate_distance_vector(nodes, graph, max_rounds=100):
    dist, next_hop = initialize_tables(nodes, graph)
    round_no = 0
    while True:
        changed = False
        round_no += 1
        # simulate each node sending its vector to neighbors and neighbors updating
        for u in nodes:
            # u sends its dist[u] to all neighbors
            for neighbor in graph[u]:
                # neighbor updates its table using u's vector
                for dest in nodes:
                    via_u_cost = dist[u][dest] + graph[neighbor][u]  # cost neighbor->u + u->dest
                    if via_u_cost < dist[neighbor][dest]:
                        dist[neighbor][dest] = via_u_cost
                        next_hop[neighbor][dest] = u if next_hop[neighbor][u] is None else next_hop[neighbor][u]
                        changed = True
        if not changed or round_no >= max_rounds:
            break
    return dist, next_hop, round_no

def print_routing_tables(nodes, dist, next_hop):
    for n in nodes:
        print(f"\nRouting table for {n}")
        print(f"{'Destination':>12} {'Cost':>8} {'NextHop':>8}")
        for dest in sorted(nodes):
            cost = dist[n][dest]
            cost_str = "∞" if cost >= INFINITY else str(cost)
            nh = next_hop[n][dest] if next_hop[n][dest] is not None else "-"
            print(f"{dest:>12} {cost_str:>8} {nh:>8}")

def reconstruct_path_from_dv(source, dest, next_hop):
    if source == dest:
        return [source]
    path = [source]
    cur = source
    visited = set([cur])
    while cur != dest:
        nh = next_hop[cur][dest]
        if nh is None or nh in visited:
            return None  # no path or loop detected
        path.append(nh)
        visited.add(nh)
        cur = nh
    return path

def main():
    if len(sys.argv) < 2:
        print("Usage: python distance_vector.py topology/topology.json")
        return
    topo_file = sys.argv[1]
    nodes, graph = load_topology(topo_file)
    dist, next_hop, rounds = simulate_distance_vector(nodes, graph)
    print(f"Converged in {rounds} rounds.")
    print_routing_tables(nodes, dist, next_hop)

    # interactive path query
    while True:
        q = input("\nEnter source,destination to show path (like A D) or 'quit': ").strip()
        if q.lower() in ("q", "quit", "exit", ""):
            break
        try:
            s, d = q.split()
            if s not in nodes or d not in nodes:
                print("Unknown nodes. Try again.")
                continue
            path = reconstruct_path_from_dv(s, d, next_hop)
            if path:
                # compute cost
                cost = 0
                for i in range(len(path)-1):
                    cost += graph[path[i]][path[i+1]]
                print(f"Path {s} -> {d}: {' -> '.join(path)} (cost {cost})")
            else:
                print("No path (or loop) found according to DV tables.")
        except Exception:
            print("Invalid input. Example: A D")

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

Code_2

# link_state.py

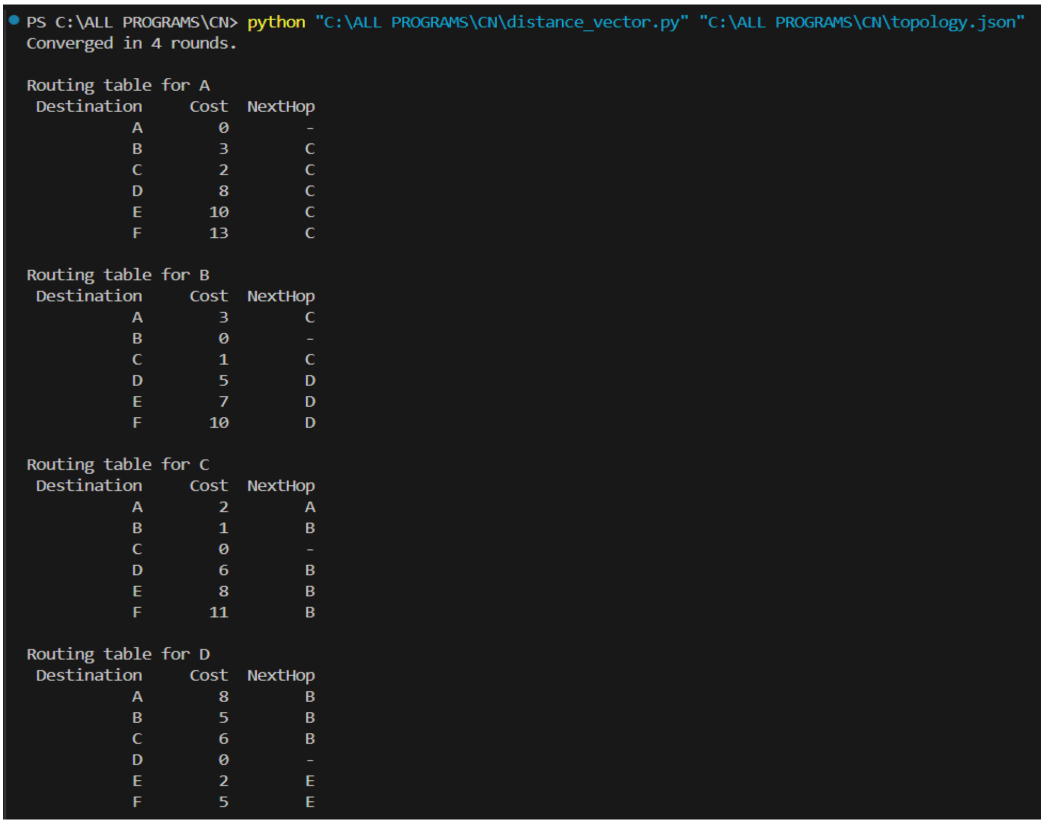
```
import json
import sys
import heapq
from collections import defaultdict

def load_topology(path):
    with open(path, "r") as f:
        topo = json.load(f)
    nodes = topo["nodes"]
    edges = topo["links"]
    graph = {n: {} for n in nodes}
    for u, v, w in edges:
        graph[u][v] = w
        graph[v][u] = w  # undirected
    return nodes, graph

def dijkstra(source, graph):
    # returns dist dict and parent dict (for path reconstruction)
    dist = {n: float('inf') for n in graph}
    parent = {n: None for n in graph}
    dist[source] = 0
    pq = [(0, source)]
    while pq:
        d, u = heapq.heappop(pq)
        if d > dist[u]:
            continue
        for v, w in graph[u].items():
            nd = d + w
            if nd < dist[v]:
                dist[v] = nd
                parent[v] = u
                heapq.heappush(pq, (nd, v))
    return dist, parent

def reconstruct_path(parent, src, dst):
    if parent[dst] is None and src != dst:
        if src == dst:
            return [src]
        return None
    path = []
    cur = dst
    while cur is not None:
        path.append(cur)
        if cur == src:
            break
        cur = parent[cur]
    path.reverse()
    if path[0] != src:
        return None
    return path

def next_hop_from_path(path):
    if not path or len(path) < 2:
        return None
    return path[1]

def build_routing_table(node, graph):
    dist, parent = dijkstra(node, graph)
    table = {}
    for dest in graph:
        if dest == node:
            table[dest] = (0, "-")
        else:
            path = reconstruct_path(parent, node, dest)
            if path is None:
                table[dest] = (float('inf'), None)
            else:
                nh = next_hop_from_path(path)
                table[dest] = (dist[dest], nh)
    return table

def print_table(node, table):
    print(f"\nRouting table for {node}")
    print(f"{'Destination':>12} {'Cost':>8} {'NextHop':>8}")
    for dest in sorted(table):
        cost, nh = table[dest]
        cost_str = "∞" if cost == float('inf') else str(cost)
        nh_str = "-" if nh is None else nh
        print(f"{dest:>12} {cost_str:>8} {nh_str:>8}")


def main():
    if len(sys.argv) < 2:
        print("Usage: python link_state.py topology/topology.json")
        return
    topo_file = sys.argv[1]
    nodes, graph = load_topology(topo_file)
    for n in nodes:
        table = build_routing_table(n, graph)
        print_table(n, table)


    # optionally allow path queries
    while True:
        q = input("\nEnter source,destination to show path (like A D) or 'quit': ").strip()
        if q.lower() in ("q", "quit", "exit", ""):
            break
        try:
            s, d = q.split()
            if s not in graph or d not in graph:
                print("Unknown nodes. Try again.")
                continue
            _, parent = dijkstra(s, graph)
            path = reconstruct_path(parent, s, d)
            if path:
                print(f"Path {s} -> {d}: {' -> '.join(path)} (cost {sum(graph[path[i]][path[i+1]] for i in range(len(path)-1))})")
            else:
                print("No path found.")
        except Exception as e:
            print("Invalid input. Example: A D")


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

Code_3

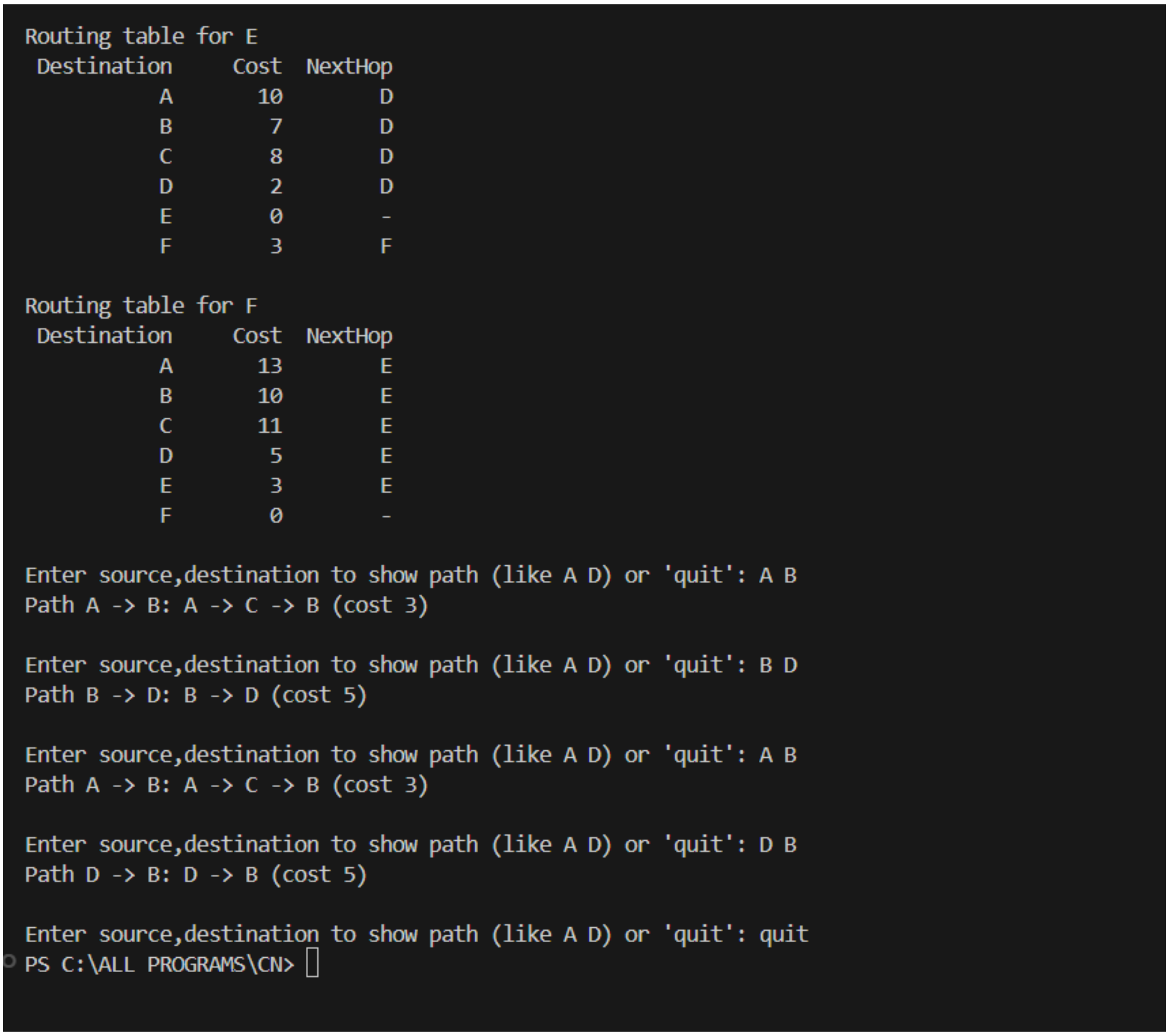
Ftopology.json

```json
{
  "nodes": ["A", "B", "C", "D", "E", "F"],
  "links": [
    ["A", "B", 4],
    ["A", "C", 2],
    ["B", "C", 1],
    ["B", "D", 5],
    ["C", "D", 8],
    ["C", "E", 10],
    ["D", "E", 2],
    ["D", "F", 6],
    ["E", "F", 3]
  ]
}
```

Output



```
PS C:\ALL PROGRAMS\CN> python "C:\ALL PROGRAMS\CN\distance_vector.py" "C:\ALL PROGRAMS\CN\topology.json"
Converged in 4 rounds.

Routing table for A
 Destination     Cost  NextHop
           A        0        -
           B        3        C
           C        2        C
           D        8        C
           E       10        C
           F       13        C

Routing table for B
 Destination     Cost  NextHop
           A        3        C
           B        0        -
           C        1        C
           D        5        D
           E        7        D
           F       10        D

Routing table for C
 Destination     Cost  NextHop
           A        2        A
           B        1        B
           C        0        -
           D        6        B
           E        8        B
           F       11        B

Routing table for D
 Destination     Cost  NextHop
           A        8        B
           B        5        B
           C        6        B
           D        0        -
           E        2        E
           F        5        E
```



```
Routing table for E
 Destination     Cost  NextHop
           A       10        D
           B        7        D
           C        8        D
           D        2        D
           E        0        -
           F        3        F

Routing table for F
 Destination     Cost  NextHop
           A       13        E
           B       10        E
           C       11        E
           D        5        E
           E        3        E
           F        0        -

Enter source,destination to show path (like A D) or 'quit': A B
Path A -> B: A -> C -> B (cost 3)

Enter source,destination to show path (like A D) or 'quit': B D
Path B -> D: B -> D (cost 5)

Enter source,destination to show path (like A D) or 'quit': A B
Path A -> B: A -> C -> B (cost 3)

Enter source,destination to show path (like A D) or 'quit': D B
Path D -> B: D -> B (cost 5)

Enter source,destination to show path (like A D) or 'quit': quit
PS C:\ALL PROGRAMS\CN> 
```

Name : SAISH BAVISKAR

Roll No : TEAD23155

Division : A

Dept : TE (AI&DS)

Subject : Computer Networks Lab

Problem Statement : 5. Socket Programming using C/C++/Java/python.

a. TCP Client, TCP Server.

b. UDP Client, UDP Server.

Code

# Tcp_client.py

```
import socket

def tcp_client(host='127.0.0.1', port=65432):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((host, port))
        message = "Hello TCP Server!"
        print(f"Sending: {message}")
        s.sendall(message.encode())
        data = s.recv(1024)
        print('Received from server:', data.decode())


if __name__ == '__main__':
    tcp_client()
```

# Tcp_server.py

```
import socket


def tcp_server(host='127.0.0.1', port=65432):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((host, port))
        s.listen()
        print(f"TCP Server listening on {host}:{port}...")
        conn, addr = s.accept()
        with conn:
            print(f"Connected by {addr}")
            while True:
                data = conn.recv(1024)
                if not data:
                    break
                print("Received:", data.decode())
                conn.sendall(data)  # Echo back
if __name__ == '__main__':
    tcp_server()
```

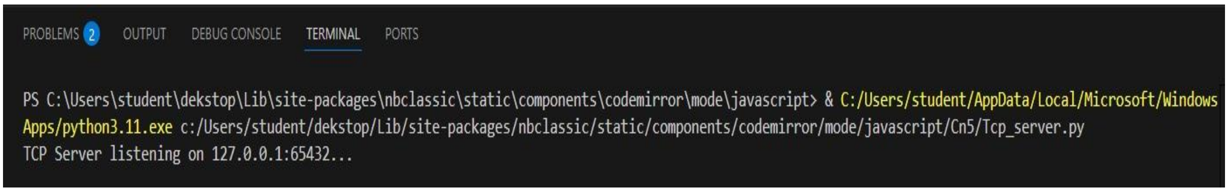
# udp_client.py

```
import socket
def udp_client(host='127.0.0.1', port=65433):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        message = "Hello UDP Server!"
        s.sendto(message.encode(), (host, port))
        print(f"Sent: {message}")

        data, server = s.recvfrom(1024)  # Wait for response
        print(f"Received from server: {data.decode()}")


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

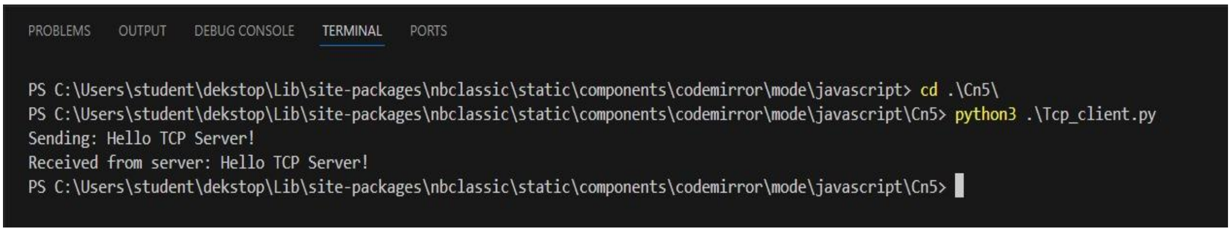
# udp_server.py

```
import socket

def udp_server(host='127.0.0.1', port=65433):

    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:

        s.bind((host, port))

        print(f"UDP server listening on {host}:{port}...")

        while True:

            data, addr = s.recvfrom(1024)  # Receive data from client

            print(f"Received from {addr}: {data.decode()}")

            s.sendto(data, addr)  # Echo back to client


if __name__ == "__main__":

    udp_server()
```

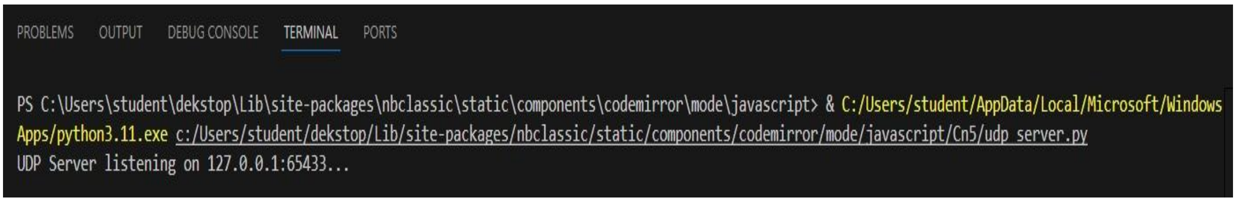
OUTPUT



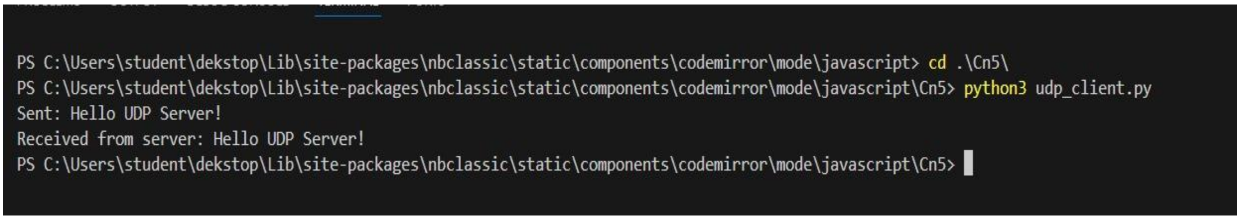
```
PS C:\Users\student\dekstop\Lib\site-packages\nbclassic\static\components\codemirror\mode\javascript> & C:/Users/student/AppData/Local/Microsoft/Windows
Apps/python3.11.exe c:/Users/student/dekstop/Lib/site-packages/nbclassic/static/components/codemirror/mode/javascript/Cn5/Tcp_server.py
TCP Server listening on 127.0.0.1:65432...
```



```
PS C:\Users\student\dekstop\Lib\site-packages\nbclassic\static\components\codemirror\mode\javascript> cd .\Cn5\
PS C:\Users\student\dekstop\Lib\site-packages\nbclassic\static\components\codemirror\mode\javascript\Cn5> python3 .\Tcp_client.py
Sending: Hello TCP Server!
Received from server: Hello TCP Server!
PS C:\Users\student\dekstop\Lib\site-packages\nbclassic\static\components\codemirror\mode\javascript\Cn5>
```



```
PS C:\Users\student\dekstop\Lib\site-packages\nbclassic\static\components\codemirror\mode\javascript> & C:/Users/student/AppData/Local/Microsoft/Windows
Apps/python3.11.exe c:/Users/student/dekstop/Lib/site-packages/nbclassic/static/components/codemirror/mode/javascript/Cn5/udp_server.py
UDP Server listening on 127.0.0.1:65433...
```



```
PS C:\Users\student\dekstop\Lib\site-packages\nbclassic\static\components\codemirror\mode\javascript> cd .\Cn5\
PS C:\Users\student\dekstop\Lib\site-packages\nbclassic\static\components\codemirror\mode\javascript\Cn5> python3 udp_client.py
Sent: Hello UDP Server!
Received from server: Hello UDP Server!
PS C:\Users\student\dekstop\Lib\site-packages\nbclassic\static\components\codemirror\mode\javascript\Cn5>
```

Name : SAISH BAVISKAR

Roll No : TEAD23155

Division : A

Dept : TE (AI&DS)

Subject : Computer Networks Lab

Problem Statement : 6. Write a program using TCP socket for wired network for following

a. Say Hello to Each other.

b. File transfer.

Code

# cn6_client.py

```
import socket
import os
def start_client(server_ip='127.0.0.1', server_port=12345, filename='testfile.txt'):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((server_ip, server_port))

        # 1. Send greeting
        s.sendall(b'Hello')

        # Receive server reply
        data = s.recv(1024)
        print('Server says:', data.decode())

        # 2. Send file info (filename, filesize)
        filesize = os.path.getsize(filename)
        file_info = f"{filename},{filesize}"
        s.sendall(file_info.encode())
        # Wait for server OK
        ack = s.recv(1024)
        if ack != b'OK':
            print('Server did not acknowledge file info.')
            return
        # 3. Send file data
        with open(filename, 'rb') as f:
            while True:
                bytes_read = f.read(4096)
                if not bytes_read:
                    break
                s.sendall(bytes_read)

        print(f'File {filename} sent successfully.')

if __name__ == '__main__':
    start_client()
```

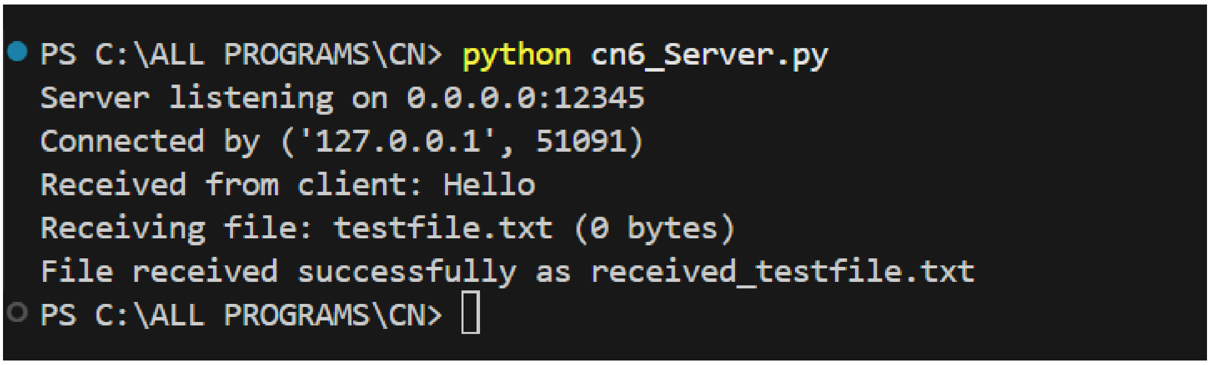
# cn6_Server.py

```
import socket
def start_server(host='0.0.0.0', port=12345):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((host, port))
        s.listen(1)
        print(f'Server listening on {host}:{port}')

        conn, addr = s.accept()
        with conn:
            print(f'Connected by {addr}')
```

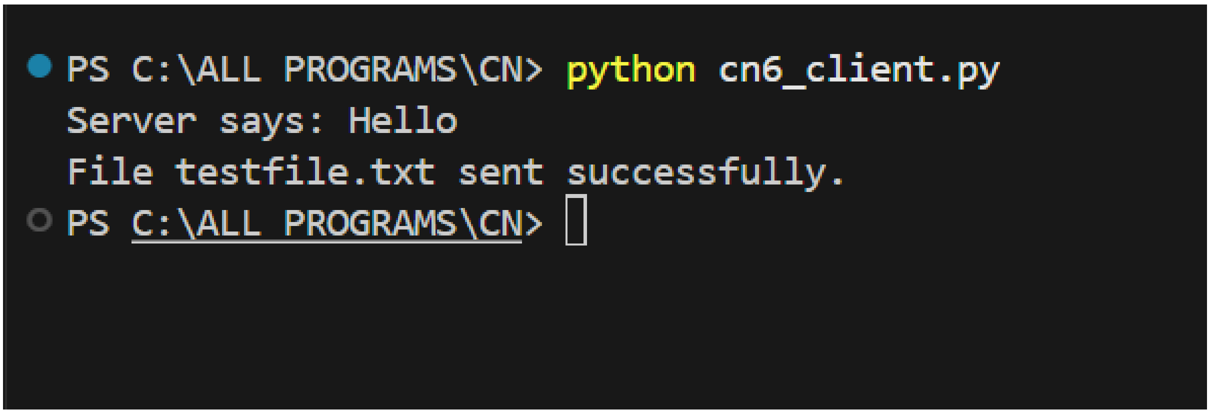
```
# 1. Receive greeting
data = conn.recv(1024).decode()
print(f'Received from client: {data}')

# Reply "Hello"
if data.strip().lower() == 'hello':
    conn.sendall(b'Hello')
else:
    conn.sendall(b'Unknown greeting')
    return

# 2. Receive file info (filename and size)
file_info = conn.recv(1024).decode()
filename, filesize = file_info.split(',')
filesize = int(filesize)
print(f'Receiving file: {filename} ({filesize} bytes)')

# Acknowledge file info receipt
conn.sendall(b'OK')
# 3. Receive the file data
with open('received_' + filename, 'wb') as f:
    received = 0
    while received < filesize:
        bytes_read = conn.recv(4096)
        if not bytes_read:
            break
        f.write(bytes_read)
        received += len(bytes_read)

print(f'File received successfully as received_{filename}')
```

```
if __name__ == '__main__':

start_server()
```

OUTPUT



```
PS C:\ALL PROGRAMS\CN> python cn6_Server.py
Server listening on 0.0.0.0:12345
Connected by ('127.0.0.1', 51091)
Received from client: Hello
Receiving file: testfile.txt (0 bytes)
File received successfully as received_testfile.txt
PS C:\ALL PROGRAMS\CN>
```



```
PS C:\ALL PROGRAMS\CN> python cn6_client.py
Server says: Hello
File testfile.txt sent successfully.
PS C:\ALL PROGRAMS\CN>
```

Name : SAISH BAVISKAR

Roll No : TEAD23155

Division : A

Dept : TE (AI&DS)

Subject : Computer Networks Lab

Problem Statement : 7. Write a program using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each) between two machines.

Code

# CN7_client.py

```
import socket

import os

# Server details

SERVER_IP = "127.0.0.1"   # Change to server machine IP

SERVER_PORT = 5001

BUFFER_SIZE = 4096


# Create UDP socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)


# Files to send

files = [

    "files_to_send/script.py",

    "files_to_send/notes.txt",

    "files_to_send/audio.mp3",

    "files_to_send/video.mp4"

]
```

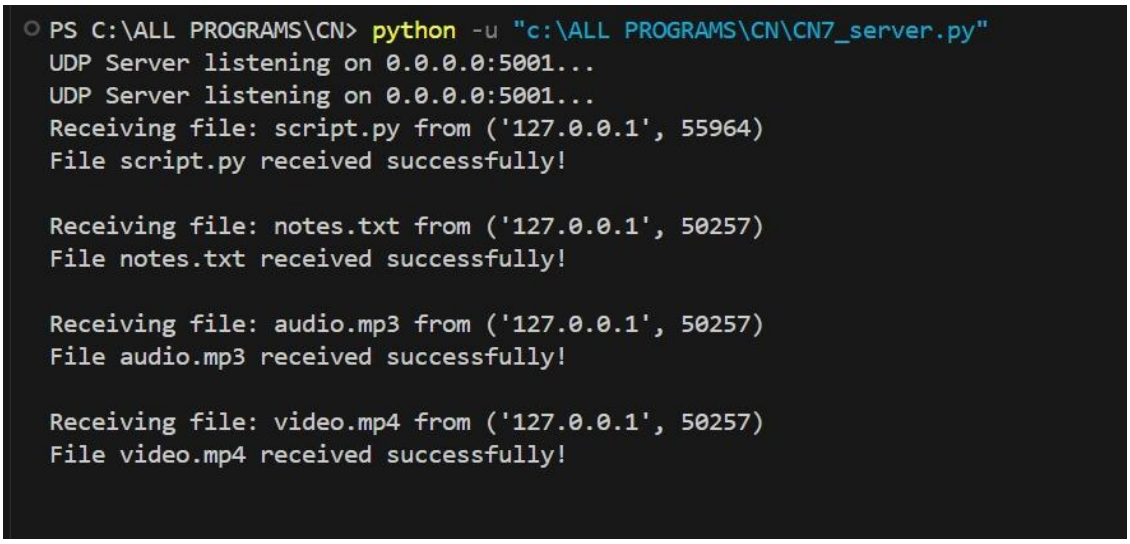
```
for filepath in files:
    filename = os.path.basename(filepath)

    # Send filename first
    client_socket.sendto(filename.encode(), (SERVER_IP, SERVER_PORT))

    # Send file content
    with open(filepath, "rb") as f:
        while True:
            bytes_read = f.read(BUFFER_SIZE)
            if not bytes_read:
                break
            client_socket.sendto(bytes_read, (SERVER_IP, SERVER_PORT))

    # Send end-of-file marker
    client_socket.sendto(b"EOF", (SERVER_IP, SERVER_PORT))

    print(f"File {filename} sent successfully!\n")
```

# CN7_server.py

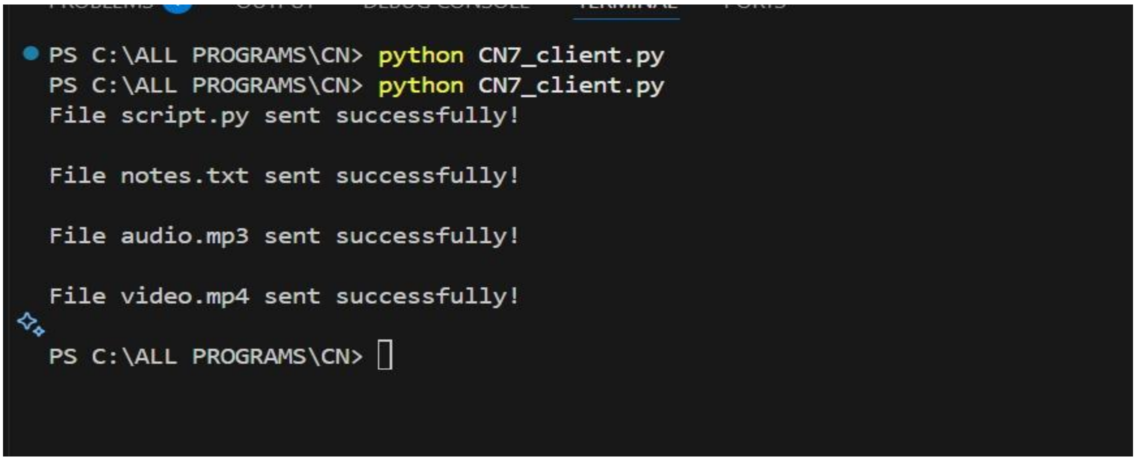
```
import socket
import os

# Server settings
SERVER_IP = "0.0.0.0"
SERVER_PORT = 5001
BUFFER_SIZE = 4096
```

```
SAVE_DIR = "received_files"

os.makedirs(SAVE_DIR, exist_ok=True)

# Create UDP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind((SERVER_IP, SERVER_PORT))

print(f"UDP Server listening on {SERVER_IP}:{SERVER_PORT}...")

while True:
    # Receive file name
    filename, client_addr = server_socket.recvfrom(BUFFER_SIZE)
    filename = filename.decode()
    print(f"Receiving file: {filename} from {client_addr}")

    # Open file for writing
    with open(os.path.join(SAVE_DIR, filename), "wb") as f:
        while True:
            data, addr = server_socket.recvfrom(BUFFER_SIZE)
            if data == b"EOF":
                break
            f.write(data)

    print(f"File {filename} received successfully!\n")
```

OUTPUT



```
PS C:\ALL PROGRAMS\CN> python -u "c:\ALL PROGRAMS\CN\CN7_server.py"
UDP Server listening on 0.0.0.0:5001...
UDP Server listening on 0.0.0.0:5001...
Receiving file: script.py from ('127.0.0.1', 55964)
File script.py received successfully!

Receiving file: notes.txt from ('127.0.0.1', 50257)
File notes.txt received successfully!

Receiving file: audio.mp3 from ('127.0.0.1', 50257)
File audio.mp3 received successfully!

Receiving file: video.mp4 from ('127.0.0.1', 50257)
File video.mp4 received successfully!
```



```
PS C:\ALL PROGRAMS\CN> python CN7_client.py
PS C:\ALL PROGRAMS\CN> python CN7_client.py
File script.py sent successfully!

File notes.txt sent successfully!

File audio.mp3 sent successfully!

File video.mp4 sent successfully!

PS C:\ALL PROGRAMS\CN> 
```