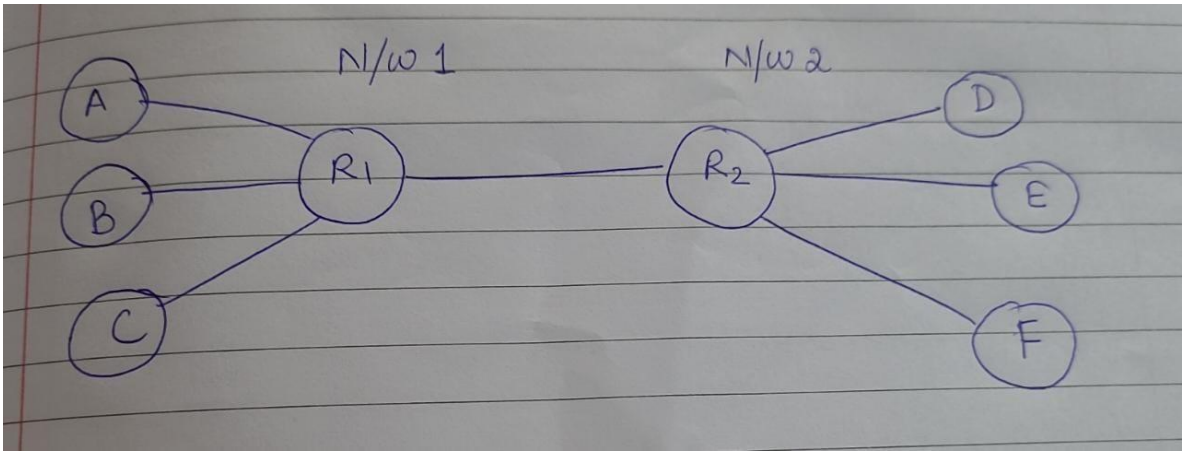
NAME: ANANYA PILLAI

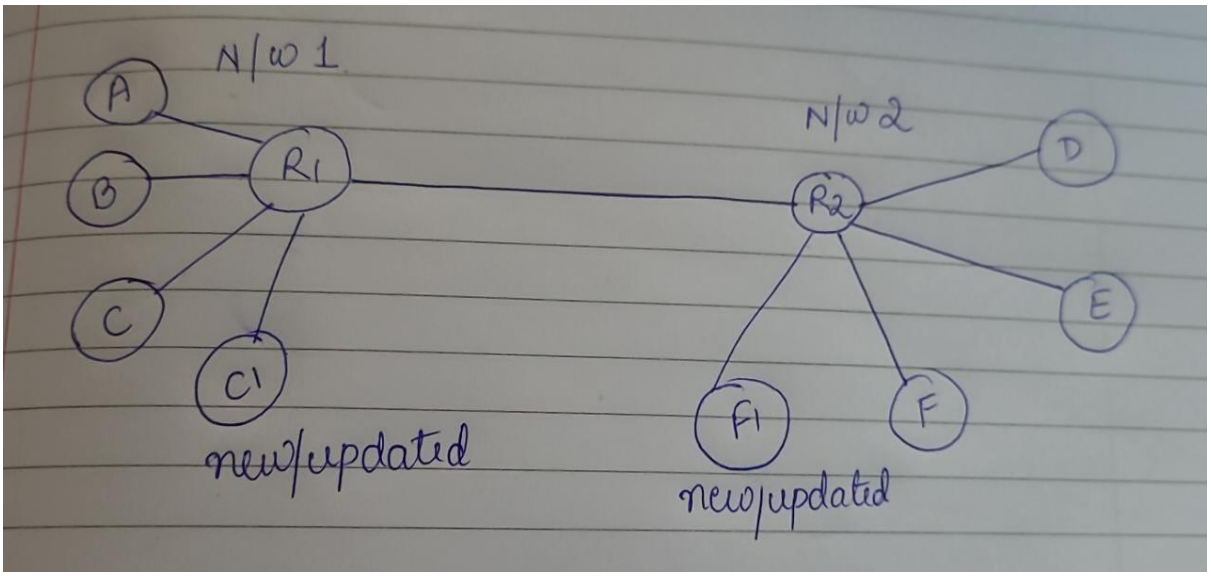
Q1) Using python, develop a simple UDP client/server application to simulate distance vector routing algorithm.

- Assume R1 and R2 are two edge routers, which connect two different networks.
- Assume minimum of three unique (directly connected) routers for each router (R1 & R2).
- Draw the network topology diagram (in lab note)
- Build initial routing tables at R1 & R2 using DVR algorithm.
- Print the initial routing tables at R1 & R2.
- Exchange the routing tables between R1 & R2 (using UDP socket programming)
- Assume R1 as UDP client & R2 as UDP server.
- Print the routing tables at R1, R2 after exchanging the routing tables.
- Add one extra router at R1 & R2. Print the updated routing tables at R1, R2.

NETWORK TOPOLOGY DIAGRAM:



NETWORK DIAGRAM AFTER ADDING NEW NODES (C1 AND F1):



SERVER CODE:

```
import pickle
import socket
sock_p = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock_p.bind(('localhost', 3000))
data = sock_p.recvfrom(1024)[0].decode()
print(data)
sock_p.close()
sock_n = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock_n.sendto('ANANYA PILLAI'.encode(), ('localhost', 3001))
sock_n.close()
class DVR:
    def __init_(self, router_name, neighbors, server_address=None, server_port=None):
        self.router_name = router_name
        self.routing_table = {}
        self.neighbors = neighbors
        self.server_address = server_address
        self.server_port = server_port

    def initialize_routing_table(self):
        self.routing_table = {}
        for neighbor in self.neighbors:
            self.routing_table[neighbor] = {
                'cost': float('inf'), 'next_hop': None}
        # Set cost to reach directly connected neighbors as 1
        for neighbor in self.neighbors:
            self.routing_table[neighbor]['cost'] = 1
            self.routing_table[neighbor]['next_hop'] = neighbor
```

```
    def send_routing_table(self):
        # Create UDP socket
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        # Serialize and send the routing table
        routing_table_data = pickle.dumps(self.routing_table)
        sock.sendto(routing_table_data, (self.server_address, self.server_port))
        # Close the socket
        sock.close()

    def receive_routing_table(self):
        # Create UDP socket
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        # Bind the socket to receive the routing table
        sock.bind(('localhost', self.server_port))

        # Receive and deserialize the routing table
        routing_table_data, _ = sock.recvfrom(4096)
        routing_table = pickle.loads(routing_table_data)

        # Close the socket
        sock.close()
        return routing_table

    def update_routing_table(self, neighbor_router, neighbor_routing_table):
        for destination, neighbor_info in neighbor_routing_table.items():
            # Add cost to reach neighbor
            neighbor_cost = neighbor_info['cost'] + 1
            if destination not in self.routing_table or neighbor_cost <
self.routing_table[destination]['cost']:
                if destination != self.router_name:
```

```
                self.routing_table[destination] = {
                    'cost': neighbor_cost, 'next_hop': neighbor_router}

    def print_routing_table(self):
        print("========================================")
        print(f"Routing table for {self.router_name}:")
        print("========================================")
        print("Destination\tCost\t\tNext Hop")
        for destination, info in self.routing_table.items():
            print(f"{destination}\t\t{info['cost']}\t\t{info['next_hop']}")

# Create R2 router with its neighbor routers and server details
R2 = DVR('R2', ['R1', 'A', 'B', 'C'], server_address='localhost', server_port=3000)
# Initialize routing table
R2.initialize_routing_table()
R2.print_routing_table()
# Receive and update routing table
R2.update_routing_table('R1', R2.receive_routing_table())
# Print final routing table
R2.print_routing_table()
R2.send_routing_table()
# Add new nodes to the neighbor list
R2.neighbors.append('C1')
# Reset routing table
R2.initialize_routing_table()
# Receive and update routing table again
R2.update_routing_table('R1', R2.receive_routing_table())
# Print final routing table
R2.print_routing_table()
R2.send_routing_table()
```

CLIENT CODE:

```
import socket
import pickle
sock_p = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock_p.sendto('21BIT0081'.encode(), ('localhost', 3000))
sock_p.close()
sock_n = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock_n.bind(('localhost', 3001))
data = sock_n.recvfrom(1024)[0].decode()
print(data)
sock_n.close()
class DVR:
    def __init_(self, router_name, neighbors, server_address=None, server_port=None):
        self.router_name = router_name
        self.routing_table = {}
        self.neighbors = neighbors
        self.server_address = server_address
        self.server_port = server_port

    def initialize_routing_table(self):
        self.routing_table = {}
        for neighbor in self.neighbors:
            self.routing_table[neighbor] = {
                'cost': float('inf'), 'next_hop': None}
        # Set cost to reach directly connected neighbors as 1
        for neighbor in self.neighbors:
            self.routing_table[neighbor]['cost'] = 1
            self.routing_table[neighbor]['next_hop'] = neighbor
```

```
    def send_routing_table(self):
        # Create UDP socket
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        # Serialize and send the routing table
        routing_table_data = pickle.dumps(self.routing_table)
        sock.sendto(routing_table_data, (self.server_address, self.server_port))
        # Close the socket
        sock.close()

    def receive_routing_table_data(self):
        # Create UDP socket
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

        # Bind the socket to receive the routing table
        sock.bind(('localhost', self.server_port))

        # Receive and deserialize the routing table
        routing_table_data, _ = sock.recvfrom(4096)
        routing_table = pickle.loads(routing_table_data)

        # Close the socket
        sock.close()
        return routing_table

    def update_routing_table(self, neighbor_router, neighbor_routing_table):
        for destination, neighbor_info in neighbor_routing_table.items():
            # Add cost to reach neighbor
            neighbor_cost = neighbor_info['cost'] + 1
            if destination not in self.routing_table or neighbor_cost <
self.routing_table[destination]['cost']:
```

```
            if destination != self.router_name:
                self.routing_table[destination] = {
                    'cost': neighbor_cost, 'next_hop': neighbor_router}

    def print_routing_table(self):
        print("========================================")
        print(f"Routing table for {self.router_name}:")
        print("========================================")
        print("Destination\tCost\t\tNext Hop")
        for destination, info in self.routing_table.items():
            print(f"{destination}\t\t{info['cost']}\t\t{info['next_hop']}")

# Create R1 router with its neighbor routers and server details
R1 = DVR('R1', ['R2', 'D', 'E','F'], 'localhost', 3000)
# Initialize routing table
R1.initialize_routing_table()
R1.print_routing_table()
# Send initial routing table
R1.send_routing_table()
# Receive and update routing table from R2
R1.update_routing_table('R2', R1.receive_routing_table_data())
# Print initial routing table
R1.print_routing_table()
# Add new nodes to the neighbor list
R1.neighbors.append('F1')
# Reset routing table
R1.initialize_routing_table()
# Send updated routing table after adding new nodes
R1.send_routing_table()
```

```
# Receive and update routing table from R2 after adding new nodes
R1.update_routing_table('R2', R1.receive_routing_table_data())
# Print final routing table after adding new nodes
R1.print_routing_table()
```

OUTPUT:

Output contains:

- Printing initial routing tables at R1 & R2
- Printing the routing tables at R1 & R2 after exchanging the routing information
- Printing the routing tables at R1 & R2 after adding extra routers

```
========== RESTART: C:/Users/hp/Desktop/server.py ==========
21BIT0081
========================================
Routing table for R2:
========================================
Destination     Cost            Next Hop
R1              1               R1
A               1               A
B               1               B
C               1               C
========================================
Routing table for R2:
========================================
Destination     Cost            Next Hop
R1              1               R1
A               1               A
B               1               B
C               1               C
D               2               R1
E               2               R1
F               2               R1
========================================
Routing table for R2:
========================================
Destination     Cost            Next Hop
R1              1               R1
A               1               A
B               1               B
C               1               C
C1              1               C1
D               2               R1
E               2               R1
F               2               R1
F1              2               R1
>>>
```

```
========== RESTART: C:/Users/hp/Desktop/client.py ==========
ANANYA PILLAI
=======================================
Routing table for R1:
=======================================
Destination     Cost            Next Hop
R2              1               R2
D               1               D
E               1               E
F               1               F
=======================================
Routing table for R1:
=======================================
Destination     Cost            Next Hop
R2              1               R2
D               1               D
E               1               E
F               1               F
A               2               R2
B               2               R2
C               2               R2
=======================================
Routing table for R1:
=======================================
Destination     Cost            Next Hop
R2              1               R2
D               1               D
E               1               E
F               1               F
F1              1               F1
A               2               R2
B               2               R2
C               2               R2
C1              2               R2
>>>
```

THANK YOU !