

router.py

```
import socket
import threading
import time
import sys
import json

def load_config(filename):
    with open(filename, "r") as file:
        lines = file.readlines()
        node_count = int(lines[0].strip())
        neighbors = []

        for line in lines[1:]:
            parts = line.strip().split()
            if len(parts) == 4:
                neighbors.append({
                    "label": parts[0],
                    "id": int(parts[1]),
                    "cost": int(parts[2]),
                    "port": int(parts[3]),
                })
            else:
                if line.strip(): # if line is empty or spaces
                    print(f"Ignoring line: {line.strip()}")

        return node_count, neighbors

def dijkstra(graph, source):
    distance = {node: float('inf') for node in range(len(graph))}
    previous = {node: None for node in range(len(graph))}
    distance[source] = 0
    unvisited = set(range(len(graph)))

    while unvisited:
        current_node = min(unvisited, key=lambda node: distance[node])
        unvisited.remove(current_node)

        for neighbor_info in graph[current_node]:
            neighbor, cost = neighbor_info["id"], neighbor_info["cost"]
            temp_value = distance[current_node] + cost
```

```

        if temp_value < distance[neighbor]:
            distance[neighbor] = temp_value
            previous[neighbor] = current_node

    return distance, previous

def send_udp(message, host, port):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        s.sendto(message.encode(), (host, port))

def receive_udp(port):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        s.bind(("", port))
        message, _ = s.recvfrom(1024)

    return message.decode()

def send_link_state(router_id, neighbors):
    while True:
        message = json.dumps({"id": router_id, "neighbors": neighbors})
        for neighbor in neighbors:
            send_udp(message, "localhost", neighbor["port"])

        time.sleep(1)

def receive_and_broadcast_link_state(port, neighbors, link_state):
    while True:
        message = receive_udp(port)
        data = json.loads(message)
        link_state[data["id"]] = data["neighbors"]
        for neighbor in neighbors:
            send_udp(message, "localhost", neighbor["port"])

def print_routing_table(router_id, distance, previous, node_count):
    router_label = chr(router_id + ord("A"))

    # Print Dijkstra
    print("-----")
    print("DestID Dist PrevID")

    for destination in range(node_count):
        if destination == router_id: #if self, distance = 0
            distance[destination] = 0
            prev_node_id = router_id

```

```

        else:
            prev_node_id = (previous[destination]
                            if previous[destination] is not None
                            else "-")

            print(f"{destination}          {distance[destination]}      {prev_node_id}")

# Forwarding table
print(f"\nThe forwarding table in {router_label} is printed as follows:")
print("DestID NextLabel")

for destination in range(node_count):
    if destination != router_id:
        next_hop = previous[destination]

        if next_hop == router_id: # if direct connection
            next_hop_label = chr(destination + ord("A"))

        else:
            while (previous[next_hop] is not None
                   and previous[next_hop] != router_id):

                next_hop = previous[next_hop]

            next_hop_label = (chr(next_hop + ord("A"))
                              if next_hop is not None else "None")

            print(f"{destination}      {next_hop_label}")

print("-----")

def main(router_id, router_port, config_file):
    node_count, neighbors = load_config(config_file)
    link_state = {router_id: neighbors} # Initialize with self neighbors

    # Threads for each component
    send_thread = threading.Thread(
        target=send_link_state, args=(router_id, neighbors))

    receive_thread = threading.Thread(
        target=receive_and_broadcast_link_state,
        args=(router_port, neighbors, link_state))

    send_thread.start()

```

```

receive_thread.start()

while True:
    if len(link_state) == node_count:
        distance, previous = dijkstra(link_state, router_id)
        print_routing_table(router_id, distance, previous, node_count)

    time.sleep(10)

if __name__ == "__main__":
    if len(sys.argv) != 4:
        print("Usage: python Router.py <router_id> <router_port> <config_file>")
        sys.exit(1)
    router_id = int(sys.argv[1])
    router_port = int(sys.argv[2])
    config_file = sys.argv[3]

    main(router_id, router_port, config_file)

```

Output:

The screenshot displays the output of the Router.py script for six routers (A-F). Each router window shows its forwarding table and a Dijkstra distance table. The forwarding tables for routers A, B, C, D, E, and F are identical, showing the next hop and distance to each destination node (0-5). The Dijkstra distance tables also show the distance from the router to each destination node.

Forwarding Table for Router A:

DestID	NextLabel
0	A
1	C
2	C
3	C
4	C
5	C

Dijkstra Dist Previd for Router A:

DestID	Dist	Previd
0	0	
1	2	0
2	1	0
3	3	0
4	2	0
5	4	0

Forwarding Table for Router B:

DestID	NextLabel
0	A
1	C
2	C
3	C
4	C
5	C

Dijkstra Dist Previd for Router B:

DestID	Dist	Previd
0	0	
1	2	0
2	1	0
3	3	0
4	2	0
5	4	0

Forwarding Table for Router C:

DestID	NextLabel
0	A
1	C
2	C
3	C
4	C
5	C

Dijkstra Dist Previd for Router C:

DestID	Dist	Previd
0	0	
1	2	0
2	1	0
3	3	0
4	2	0
5	4	0

Forwarding Table for Router D:

DestID	NextLabel
0	A
1	C
2	C
3	C
4	C
5	C

Dijkstra Dist Previd for Router D:

DestID	Dist	Previd
0	0	
1	2	0
2	1	0
3	3	0
4	2	0
5	4	0

Forwarding Table for Router E:

DestID	NextLabel
0	A
1	C
2	C
3	C
4	C
5	C

Dijkstra Dist Previd for Router E:

DestID	Dist	Previd
0	0	
1	2	0
2	1	0
3	3	0
4	2	0
5	4	0

Forwarding Table for Router F:

DestID	NextLabel
0	A
1	C
2	C
3	C
4	C
5	C

Dijkstra Dist Previd for Router F:

DestID	Dist	Previd
0	0	
1	2	0
2	1	0
3	3	0
4	2	0
5	4	0

## Better View of Screenshot:

```
Config A
4 2 2
5 4 4

The forwarding table in A is printed as follows:
DestID NextLabel
1 B
2 C
3 C
4 C
5 C
-----
DestID Dist PrevID
0 0 0
1 2 0
2 1 0
3 3 4
4 2 2
5 4 4

The forwarding table in A is printed as follows:
DestID NextLabel
1 B
2 C
3 C
4 C
5 C
-----
DestID Dist PrevID
0 0 0
1 2 0
2 1 0
3 3 4
4 2 2
5 4 4

The forwarding table in A is printed as follows:
DestID NextLabel
1 B
2 C
3 C
4 C
5 C
-----
DestID Dist PrevID
0 0 0
1 2 0
2 1 0
3 3 4
4 2 2
5 4 4

Config B
4 3 2
5 5 4

The forwarding table in B is printed as follows:
DestID NextLabel
0 A
2 C
3 D
4 C
5 C
-----
DestID Dist PrevID
0 2 1
1 0 1
2 2 1
3 3 1
4 3 2
5 5 4

The forwarding table in B is printed as follows:
DestID NextLabel
0 A
2 C
3 D
4 C
5 C
-----
DestID Dist PrevID
0 2 1
1 0 1
2 2 1
3 3 1
4 3 2
5 5 4

The forwarding table in B is printed as follows:
DestID NextLabel
0 A
2 C
3 D
4 C
5 C
-----
DestID Dist PrevID
0 2 1
1 0 1
2 2 1
3 3 1
4 3 2
5 5 4

Config C
4 1 2
5 3 4

The forwarding table in C is printed as follows:
DestID NextLabel
0 A
1 B
3 E
4 E
5 E
-----
DestID Dist PrevID
0 1 2
1 2 2
2 0 2
3 2 4
4 1 2
5 3 4

The forwarding table in C is printed as follows:
DestID NextLabel
0 A
1 B
3 E
4 E
5 E
-----
DestID Dist PrevID
0 1 2
1 2 2
2 0 2
3 2 4
4 1 2
5 3 4

The forwarding table in C is printed as follows:
DestID NextLabel
0 A
1 B
3 E
4 E
5 E
-----
DestID Dist PrevID
0 1 2
1 2 2
2 0 2
3 2 4
4 1 2
5 3 4

Config D
4 1 3
5 3 4

The forwarding table in D is printed as follows:
DestID NextLabel
0 E
1 B
2 E
4 E
5 E
-----
DestID Dist PrevID
0 3 2
1 3 3
2 2 4
3 0 3
4 1 3
5 3 4

The forwarding table in D is printed as follows:
DestID NextLabel
0 E
1 B
2 E
4 E
5 E
-----
DestID Dist PrevID
0 3 2
1 3 3
2 2 4
3 0 3
4 1 3
5 3 4

The forwarding table in D is printed as follows:
DestID NextLabel
0 E
1 B
2 E
4 E
5 E
-----
DestID Dist PrevID
0 3 2
1 3 3
2 2 4
3 0 3
4 1 3
5 3 4
```

Config E

DestID NextLabel

0 C

1 C

2 C

3 D

5 F

-----

DestID Dist PrevID

0 2 2

1 3 2

2 1 4

3 1 4

4 0 4

5 2 4

The forwarding table in E is printed as follows:

DestID NextLabel

0 C

1 C

2 C

3 D

5 F

-----

Config F

DestID NextLabel

0 E

1 E

2 E

3 E

4 E

-----

DestID Dist PrevID

0 4 2

1 5 2

2 3 4

3 3 4

4 2 5

5 0 5

The forwarding table in F is printed as follows:

DestID NextLabel

0 E

1 E

2 E

3 E

4 E

-----

OUTPUT

DEBUG CONSOLE

PORTS

TERMINAL

D:\Schoolwork\2023-4 Fall\ENSF 462-Labs-Lab 05>start "Config A" python Router.py 0 7000 config-A.txt

D:\Schoolwork\2023-4 Fall\ENSF 462-Labs-Lab 05>start "Config B" python Router.py 1 7001 config-B.txt

D:\Schoolwork\2023-4 Fall\ENSF 462-Labs-Lab 05>start "Config C" python Router.py 2 7002 config-C.txt

D:\Schoolwork\2023-4 Fall\ENSF 462-Labs-Lab 05>start "Config D" python Router.py 3 7003 config-D.txt

D:\Schoolwork\2023-4 Fall\ENSF 462-Labs-Lab 05>start "Config E" python Router.py 4 7004 config-E.txt

D:\Schoolwork\2023-4 Fall\ENSF 462-Labs-Lab 05>start "Config F" python Router.py 5 7005 config-F.txt

PS D:\Schoolwork\2023-4 Fall\ENSF 462-Labs-Lab 05> |

main

Live Share

Ln 10, Col 43   Spaces: 4   UTF-8   CRLF   Python   3.11.0.64-bit   Prettier   Baza