CS3205 Computer Networks Lab
Assigment 3: OSPF Routing Algorithm

G Shyam Sundar CS18B015

April 20, 2021

1 Objective

To Implement the simplified version of Open Shortest Path First (OSPF) routing protocol.

2 Introduction

Internet topology is roughly organized as a two level hierarchy

First lower level: autonomous systems (AS's). AS is a region of network under a single administartive domain.
Each AS runs an intra-domain routing protocol.

- Distance Vector, e.g., Routing Information Protocol (RIP)
- Link State, e.g., Open Shortest Path First (OSPF)
- and other

Second lower level: inter-connected AS's. inter-domain routing protocols are executed between AS's. e.g., Border Gateway Routing (BGP)

Intra-domain Routing Protocol:
This Protocol is based on unreliable datagram delivery.
We will implement Link state Routing.

- Open Shortest Path First (OSPF), based on Dijkstra's algorithm
- Each router periodically floods immediate reachability information to other routers.
- Fast convergence, but high communication and computaion overhead.

Our Goal is to determine a good path which means the shortest path throgh the network from source to destination. But To compute the shortest path every Router Should know other Routers neighbouring information. That is why we use flooding.
Link State Routing (OSPF): Flooding

- Each node knows its connectivity and cost to a direct neighbor
- Every node tells every other node this local connectivity/cost information via flooding.
- In the end, every node learns the complete topology of the network

Flooding Details :

- Each node periodically generates Link State Packet (LSP) contains
  - ID of node created LSP
  - List of direct neighbours and costs
  - Sequence number (64 bit, assume to never wrap around)
  - Time to live
- Flood is reliable uses acknowledgement and retransmission
- Sequence number used to identify *newer* LSP so that older LSP can be discarded.
- Receiving node flood LSP to all its neighbors except the neighbor where the LSP came from
- LSP is also generated when a link's state changes (failed or restored)

Now we know the Net toplogy,links costs known to all nodes by flooding. We use **Dijkstra's algorithm** to compute least cost paths from one node ('source") to all other nodes and repeat for all sources.

Link State Routing Algorithm:
Notaions:

- c(i,j): link cost from node i to j; cost infinite if not direct neighbors
- D(v): current value of cost of path from source to node v
- p(v): predecessor node along path from source to v, that is next to v
- S: set of nodes whose least cost path definitively known

Algorithm 1: Dijkstra's algorithm

```
Result: Shortest Paths from A to all the nodes.
Initialization:
S = {A};
for all nodes v do
if v adjacent to A then
    D(v) = c(A,v);
else
    D(v) = inf;
end
endfor
while Some node is not in S do
    find w not in S such that D(w) is minimum;
    add w to S;
    for all v adjacent to w and not in S do
    D(v) = min(D(v),D(w)+c(w,v));
    endfor
end
```

3 Experimental Details

3.1 Simulation setup

Steps To follow:

- We will be given N routers.
- exchange HELLO packets with neighbours.
- create Link State Advertisement (LSA) packets based on neighboring nodes info.
- broadcast the LSA packets to all other routers in the network.(flooding)
- construct the network topology based on the LSA packets received from other routers.
- determining the routing table entries based on this topology, by using Dijkstra' algorithm (single source – all nodes shortest paths). If multiple equalcost paths exist, any one of them can be reported.
- The HELLO packets will be exchanged every x seconds; the LSA updates will be sent every y seconds; the routing table computation will be done every z seconds.

3.2 Entities involved and functions in each entity

- **Exchange of Hello Packets :**
  - Send a HELLO message to its neighbors, once every Hello Interval seconds. This value is specified in the command line; Default: 1 second.
  - Packet Format:

HELLO	srcid

  - When a router receives a Hello message on an interface, it will reply with the HelloReply message along with the cost. The cost reported for $link_ij$ by node j for a packet received from node i is a Random number between $MinC_{ij}$ and $MaxC_{ij}$. Node i, on receiving this message, will store this value as the cost for $link_{ij}$.
  - Packet Format:

HELLOREPLY	j	i	value for link ij

- **Exchange of LSA Packets :**
  - Send a Link State Advertisement (LSA) message to its neighbors, once every LSA INTERVAL seconds. This value is specified in the command line; Default: 5 seconds.
  - Packet Format:

LSA	srcid	Seq.Number	No. Entries	Neigh1	Cost1	Neigh2	Cost2	...

  - The sequence number is incremented by the sender for each LSA message that it sends.
  - When a node receives an LSA message from a neighbor, it will store the LSA information and forward the LSA to all interfaces other than the interface that the packet arrived on, if and only if the newly received LSA's sequence number is strictly greater than the last known sequence number from the sender.

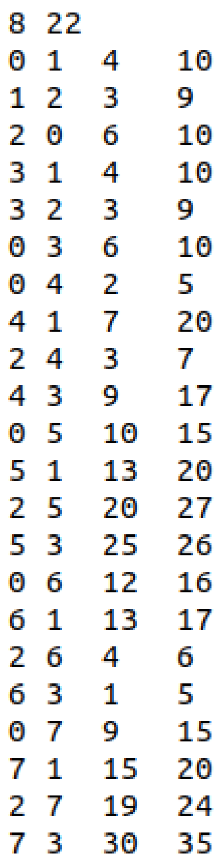
- **SPF Genration:**
  - Determine the topology using all recent Link State Advertisement (LSA) messages received from all other routers; and then run shortestcost path computation algorithm every SPF INTERVAL seconds. This value is specified in the command line; Default: 20 seconds.
  - The output will be stored in the routing table output file along with the time stamp. The output file name for Node i will be outfile–i.txt, where outfile is specified in the command line.
  - Ouput Format:
    Routing table for Node No.1 at Time 30

Destination	Path	Cost
2	1-3-2	5
3	1-3	2
4	1-3-2-4	10
...	...	...

3.3 Additional details

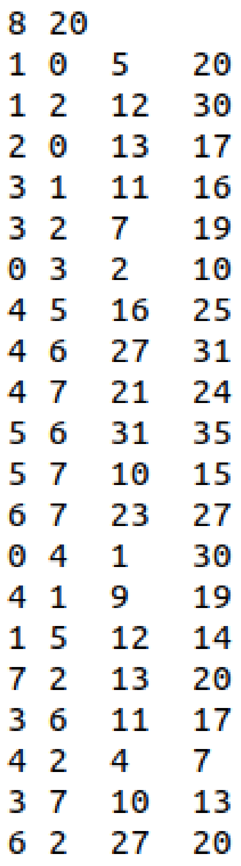
- Use a UDP socket on port number (10000 + i) for all OSPF communications.
- Read the input file and find out its neighboring node identifiers.

4 Results and Observations



```
8 22
0 1  4   10
1 2  3   9
2 0  6   10
3 1  4   10
3 2  3   9
0 3  6   10
0 4  2   5
4 1  7   20
2 4  3   7
4 3  9   17
0 5  10  15
5 1  13  20
2 5  20  27
5 3  25  26
0 6  12  16
6 1  13  17
2 6  4   6
6 3  1   5
0 7  9   15
7 1  15  20
2 7  19  24
7 3  30  35
```

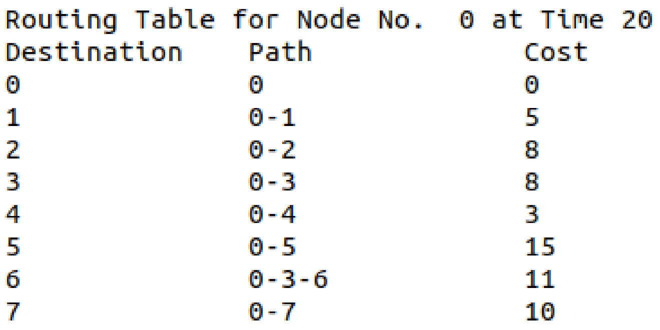
(a) **Input Topology-1**



```
8 20
1 0  5   20
1 2  12  30
2 0  13  17
3 1  11  16
3 2  7   19
0 3  2   10
4 5  16  25
4 6  27  31
4 7  21  24
5 6  31  35
5 7  10  15
6 7  23  27
0 4  1   30
4 1  9   19
1 5  12  14
7 2  13  20
3 6  11  17
4 2  4   7
3 7  10  13
6 2  27  20
```

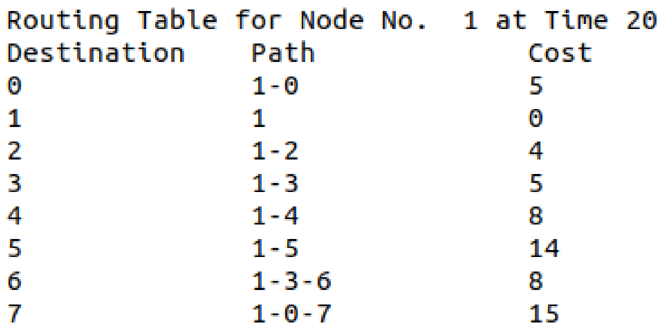
(b) **Input Topology-2**

Figure 1: Input Topologies Used



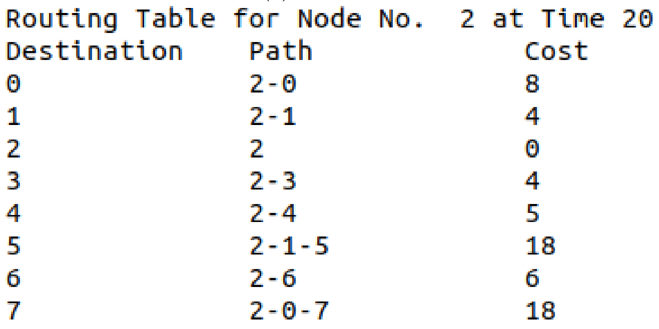
```
Routing Table for Node No.  0 at Time 20
Destination     Path          Cost
0               0             0
1               0-1           5
2               0-2           8
3               0-3           8
4               0-4           3
5               0-5           15
6               0-3-6         11
7               0-7           10
```

(a) **NODE 0**



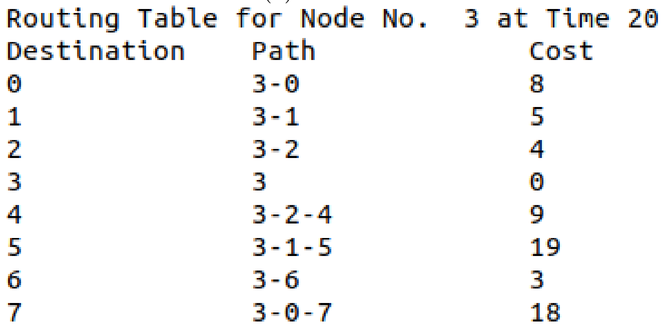
```
Routing Table for Node No.  1 at Time 20
Destination     Path          Cost
0               1-0           5
1               1             0
2               1-2           4
3               1-3           5
4               1-4           8
5               1-5           14
6               1-3-6         8
7               1-0-7         15
```

(b) **NODE 1**



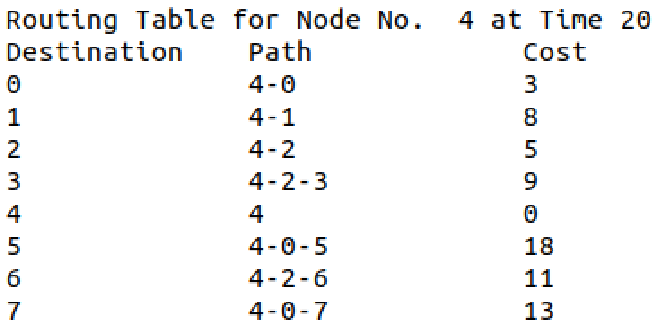
```
Routing Table for Node No.  2 at Time 20
Destination     Path          Cost
0               2-0           8
1               2-1           4
2               2             0
3               2-3           4
4               2-4           5
5               2-1-5         18
6               2-6           6
7               2-0-7         18
```

(c) **NODE 2**



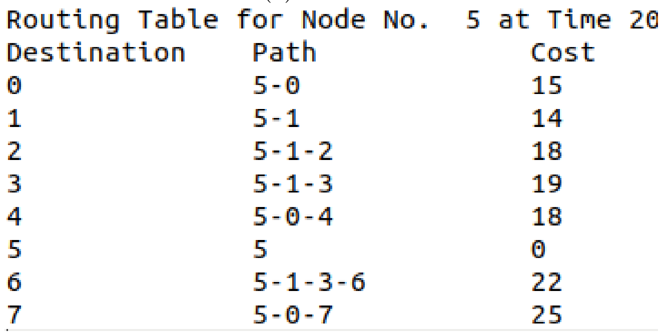
```
Routing Table for Node No.  3 at Time 20
Destination     Path          Cost
0               3-0           8
1               3-1           5
2               3-2           4
3               3             0
4               3-2-4         9
5               3-1-5         19
6               3-6           3
7               3-0-7         18
```

(d) **NODE 3**



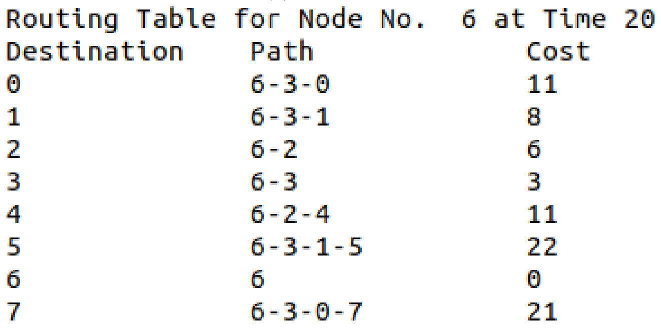
```
Routing Table for Node No.  4 at Time 20
Destination     Path          Cost
0               4-0           3
1               4-1           8
2               4-2           5
3               4-2-3         9
4               4             0
5               4-0-5         18
6               4-2-6         11
7               4-0-7         13
```

(e) **NODE 4**



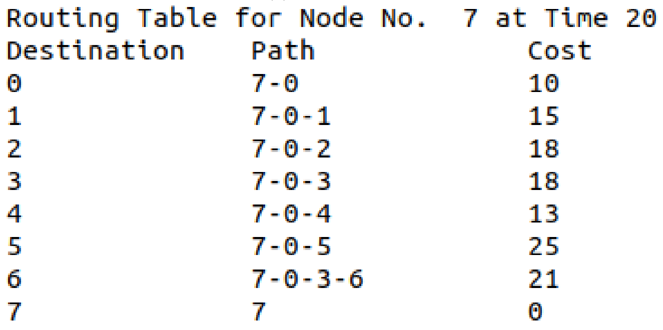
```
Routing Table for Node No.  5 at Time 20
Destination     Path          Cost
0               5-0           15
1               5-1           14
2               5-1-2         18
3               5-1-3         19
4               5-0-4         18
5               5             0
6               5-1-3-6       22
7               5-0-7         25
```

(f) **NODE 5**



```
Routing Table for Node No.  6 at Time 20
Destination     Path          Cost
0               6-3-0         11
1               6-3-1         8
2               6-2           6
3               6-3           3
4               6-2-4         11
5               6-3-1-5       22
6               6             0
7               6-3-0-7       21
```

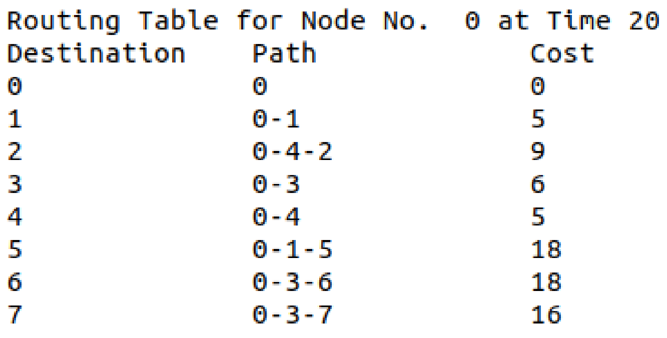
(g) **NODE 6**



```
Routing Table for Node No.  7 at Time 20
Destination     Path          Cost
0               7-0           10
1               7-0-1         15
2               7-0-2         18
3               7-0-3         18
4               7-0-4         13
5               7-0-5         25
6               7-0-3-6       21
7               7             0
```

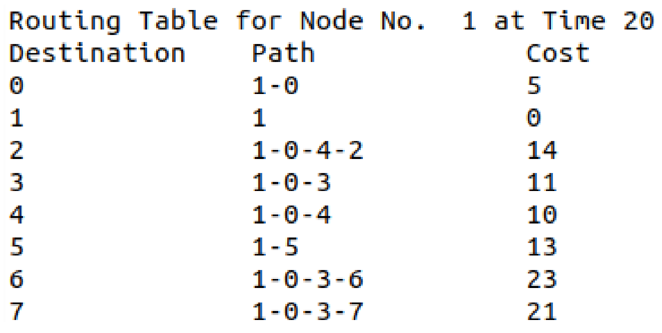
(h) **NODE 7**

Figure 2: Output for Input Topology-1



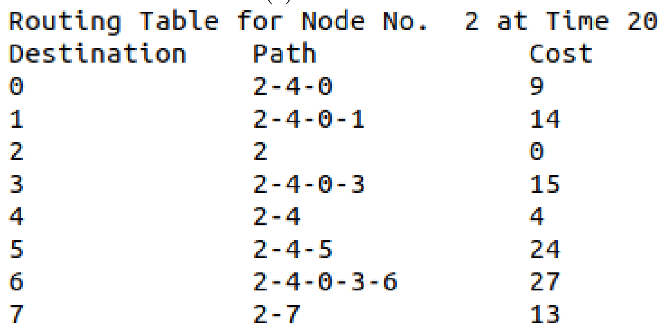
```
Routing Table for Node No.  0 at Time 20
Destination     Path            Cost
0               0               0
1               0-1             5
2               0-4-2           9
3               0-3             6
4               0-4             5
5               0-1-5           18
6               0-3-6           18
7               0-3-7           16
```

(a) **NODE 0**



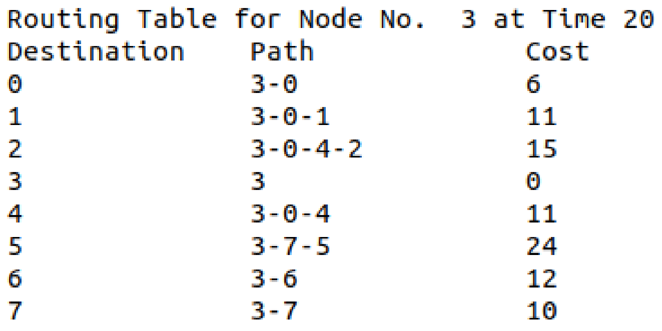
```
Routing Table for Node No.  1 at Time 20
Destination     Path            Cost
0               1-0             5
1               1               0
2               1-0-4-2         14
3               1-0-3           11
4               1-0-4           10
5               1-5             13
6               1-0-3-6         23
7               1-0-3-7         21
```

(b) **NODE 1**



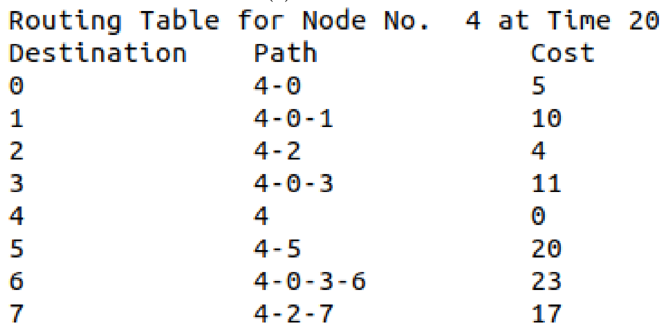
```
Routing Table for Node No.  2 at Time 20
Destination     Path            Cost
0               2-4-0           9
1               2-4-0-1         14
2               2               0
3               2-4-0-3         15
4               2-4             4
5               2-4-5           24
6               2-4-0-3-6       27
7               2-7             13
```

(c) **NODE 2**



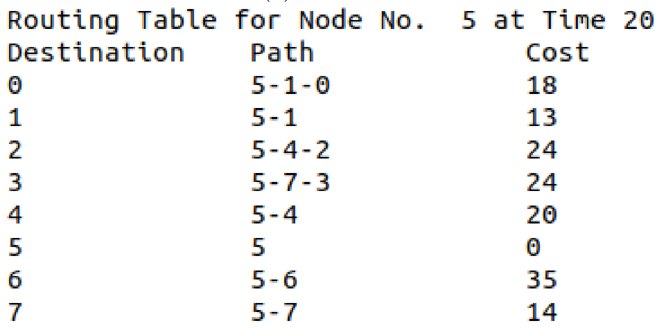
```
Routing Table for Node No.  3 at Time 20
Destination     Path            Cost
0               3-0             6
1               3-0-1           11
2               3-0-4-2         15
3               3               0
4               3-0-4           11
5               3-7-5           24
6               3-6             12
7               3-7             10
```

(d) **NODE 3**



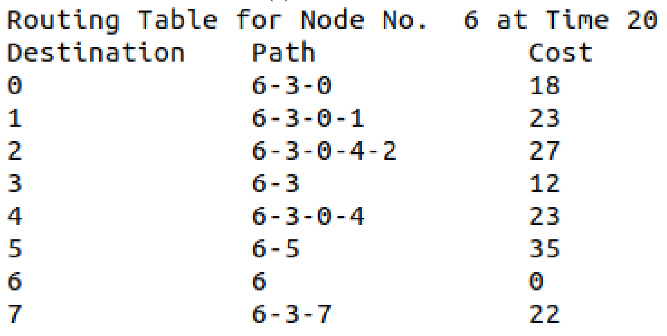
```
Routing Table for Node No.  4 at Time 20
Destination     Path            Cost
0               4-0             5
1               4-0-1           10
2               4-2             4
3               4-0-3           11
4               4               0
5               4-5             20
6               4-0-3-6         23
7               4-2-7           17
```

(e) **NODE 4**



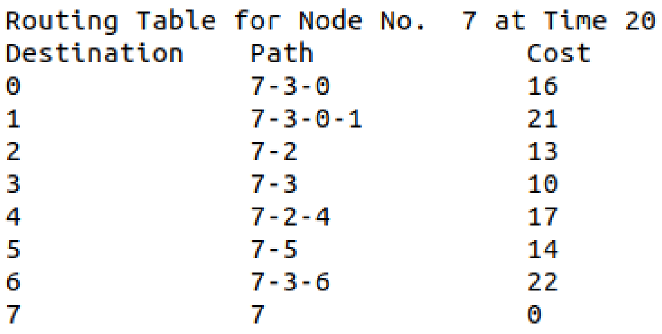
```
Routing Table for Node No.  5 at Time 20
Destination     Path            Cost
0               5-1-0           18
1               5-1             13
2               5-4-2           24
3               5-7-3           24
4               5-4             20
5               5               0
6               5-6             35
7               5-7             14
```

(f) **NODE 5**



```
Routing Table for Node No.  6 at Time 20
Destination     Path            Cost
0               6-3-0           18
1               6-3-0-1         23
2               6-3-0-4-2       27
3               6-3             12
4               6-3-0-4         23
5               6-5             35
6               6               0
7               6-3-7           22
```

(g) **NODE 6**



```
Routing Table for Node No.  7 at Time 20
Destination     Path            Cost
0               7-3-0           16
1               7-3-0-1         21
2               7-2             13
3               7-3             10
4               7-2-4           17
5               7-5             14
6               7-3-6           22
7               7               0
```

(h) **NODE 7**

Figure 3: Output for Input Topology-2

5 Learnings

- Learned how routers get shortest paths through communication.
- Learned how all the routers get know about other routers neighbouring information.
- Learned some Technical concepts like Using threads, Run programs paralell using bash script.

6 Conclusion

7 References

- https://www.geeksforgeeks.org/udp-server-client-implementation-c/
- https://stackoverflow.com/questions/21057676/need-to-call-a-function-at-periodic-time-intervals-in-c