CS3205: Introduction to Computer Networks

Report: Assignment 3
OSPF routing protocol

Anirudh S
CS17B003

---

1 Problem Statement

The aim of this assignment is to implement a simplified version of the Open Shortest Path First (OSPF) routing algorithm. The simplification is in that the fact that the network topology is provided as an input file to the network. We will see more details about the problem in section 3

2 Introduction

2.1 OSPF

Open Shortest Path First (OSPF) protocol is an intra-domain routing protocol which is based on Dijkstra's algorithm. Routing refers to identifying a suitable path for data to flow from source to the destination. Usually this path is the shortest path, which is found out by Dijkstra's algorithm. To execute the algorithm, we need the links (edges) between all the routers (nodes) and their corresponding weights. This is achieved through the process called **flooding**. In flooding, every node sends their neighbouring nodes and the corresponding edge costs to all its neighbours. On reception of these packets, the neighbours in turn re-transmit to all their neighbours and so on. Finally, every node will have information about all he nodes and edges.

2.2 Dijkstra's algorithm

Dijkstra's algorithm is a single source shortest path (SSSP) algorithm, i.e, this algorithm finds the shortest paths from a single source node to every other node. Before seeing the algorithm, let us define the following terms.

- $c(i, j)$**:** Edge cost from node i to node j; if i and j are not neighbours, then the cost is infinity.
- $D(v)$**:** Current cost of the path from source to node v.
- $p(v)$**:** Immediate predecessor node to v, along path from source to v.
- S**:** Set of processed nodes; least cost paths to these nodes are definitively known

The Dijkstra's algorithm is given in figure 1.

```
1  Initialization:
2   S = {A};
3   for all nodes v
4     if v adjacent to A
5       then D(v) = c(A,v);
6       else D(v) = δ ;
7
8  Loop
9     find w not in S such that D(w) is a minimum;
10    add w to S;
11    update D(v) for all v adjacent to w and not in S:
12       D(v) = min( D(v), D(w) + c(w,v) );
           // new cost to v is either old cost to v or known
           // shortest path cost to w plus cost from w to v
13 until all nodes in S;
```

Figure 1: Dijkstra's Algorithm[3]

3 Problem description

3.1 Input file

The input file provided has the following structure.

- The first entry on the first line specifies the number of routers(N). The node indices go from 0 to $(N-1)$. The second entry on the first line specifies the number of links.
- Each subsequent row contains the tuple $(i, j, min(C_{ij}), max(C_{ij}))$ which denotes a bidirectional link between nodes i and j. $min(C_{ij})$ and $max(C_{ij})$ are link weight limits. Their use is described in section 3.2.

3.2 Router

Every router is given a unique id and the total network topology as input. Based on the latter, the router figures out its neighbours and their corresponding weight limits. A router node i is expected to do the following:

- Send a hello packet (**HELLO** i) to each of its neighbour every hello interval h_i.
- On reception of a hello packet from node j, generate the weight for link $i-j$ between the limits, say l_{ij}, if not already done. Then it must send a helloreply packet (**HELLOREPLY** i j l_{ij}) to j.
- On reception of a helloreply packet, update the appropriate link weight.
- Send a Link State Advertisement(LSA) packet (**LSA** i seq_number $num\_entries$ nbr_1 wt_1 nbr_2 wt_2 **...**) to all its neighbours every LSA interval lsa_i.
- On reception of a LSA packet with a sequence number greater than what it has already received, update the link weights accordingly and broadcast the received packet to all its neighbours.
- Run Dijkstra's algorithm every SPF interval spf_i and print the obtained routing table in the output file in the format: Destination node - Path - Cost.

- Communication between takes place using UDP datagrams. The router is assigned the port number 10000 + id.

4 Implementation details

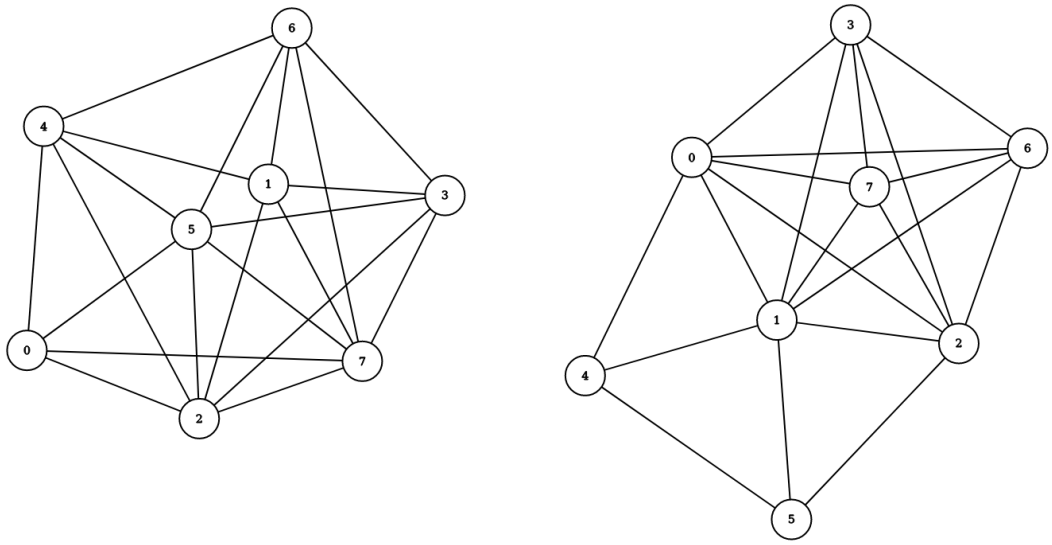
The choice of programming language for this assignment was C++. The implementation details are as follows:

1. **UDP:** The UDP implementation is based on the code given at [1]. The UDP interface is included as a header file `udp.h` into the main code.
2. **Threads:** To implement the router functionalities, four threads[2] were used.
   (a) `hello_send`: This thread sends the hello packet.
   (b) `lsai_send`: This thread sends the LSA packet.
   (c) `receive`: This thread handles all received packets.
      - HELLO: Generates edge weight if required and send helloreply packet.
      - LSA: Updates all the edge weights and transmits the received packet.
      - HELLOREPLY: If the weight was not set, it updates the received weight. If a weight was already set, then the lower of the old and new weights is retained.

   (d) `dijkstra_algo`: Runs the Dijkstra's algorithm and outputs into the file.
3. For periodic activities (such as hello packet send), the thread sleeps for the specified time before running again.

5 Experiments

The two input topologies used to test the code are shown in figure 2.



(a) Input 1 (b) Input 2

Figure 2: Input graphs

6 Results and Observations

6.1 Experiment 1

The outputs obtained for input 1 (figure 2a) is shown in figure 3.

```
Routing table for node 0 at time 60
Destination - Path - Cost
1 - 0,4,1 - 12
2 - 0,2 - 6
3 - 0,4,1,3 - 21
4 - 0,4 - 9
5 - 0,5 - 10
6 - 0,5,6 - 30
7 - 0,2,7 - 33
```

(a) Router 0

```
Routing table for node 1 at time 60
Destination - Path - Cost
0 - 1,4,0 - 12
2 - 1,4,2 - 10
3 - 1,3 - 9
4 - 1,4 - 3
5 - 1,4,0,5 - 22
6 - 1,3,7,6 - 29
7 - 1,3,7 - 23
```

(b) Router 1

```
Routing table for node 2 at time 60
Destination - Path - Cost
0 - 2,0 - 6
1 - 2,4,1 - 10
3 - 2,3 - 15
4 - 2,4 - 7
5 - 2,5 - 15
6 - 2,7,6 - 33
7 - 2,7 - 27
```

(c) Router 2

```
Routing table for node 3 at time 60
Destination - Path - Cost
0 - 3,2,0 - 21
1 - 3,1 - 9
2 - 3,2 - 15
4 - 3,1,4 - 12
5 - 3,2,5 - 30
6 - 3,7,6 - 20
7 - 3,7 - 14
```

(d) Router 3

```
Routing table for node 4 at time 60
Destination - Path - Cost
0 - 4,0 - 9
1 - 4,1 - 3
2 - 4,2 - 7
3 - 4,1,3 - 12
5 - 4,0,5 - 19
6 - 4,1,3,7,6 - 32
7 - 4,1,3,7 - 26
```

(e) Router 4

```
Routing table for node 5 at time 60
Destination - Path - Cost
0 - 5,0 - 10
1 - 5,0,4,1 - 22
2 - 5,2 - 15
3 - 5,2,3 - 30
4 - 5,0,4 - 19
6 - 5,6 - 20
7 - 5,6,7 - 26
```

(f) Router 5

```
Routing table for node 6 at time 60
Destination - Path - Cost
0 - 6,5,0 - 30
1 - 6,7,3,1 - 29
2 - 6,7,2 - 33
3 - 6,7,3 - 20
4 - 6,7,3,1,4 - 32
5 - 6,5 - 20
7 - 6,7 - 6
```

(g) Router 6

```
Routing table for node 7 at time 60
Destination - Path - Cost
0 - 7,2,0 - 33
1 - 7,3,1 - 23
2 - 7,2 - 27
3 - 7,3 - 14
4 - 7,3,1,4 - 26
5 - 7,6,5 - 26
6 - 7,6 - 6
```

(h) Router 7

Figure 3: Outputs obtained for input 1 (fig 2a)

6.2 Experiment 2

The outputs obtained for input 2 (figure 2b) is shown in figure 4.

```
Routing table for node 0 at time 60
Destination - Path - Cost
1 - 0,1 - 7
2 - 0,2 - 3
3 - 0,3 - 5
4 - 0,4 - 10
5 - 0,2,5 - 16
6 - 0,2,6 - 7
7 - 0,1,7 - 14
```

(a) Router 0

```
Routing table for node 1 at time 60
Destination - Path - Cost
0 - 1,0 - 7
2 - 1,0,2 - 10
3 - 1,0,3 - 12
4 - 1,0,4 - 17
5 - 1,0,2,5 - 23
6 - 1,0,2,6 - 14
7 - 1,7 - 7
```

(b) Router 1

```
Routing table for node 2 at time 60
Destination - Path - Cost
0 - 2,0 - 3
1 - 2,0,1 - 10
3 - 2,0,3 - 8
4 - 2,0,4 - 13
5 - 2,5 - 13
6 - 2,6 - 4
7 - 2,0,1,7 - 17
```

(c) Router 2

```
Routing table for node 3 at time 60
Destination - Path - Cost
0 - 3,0 - 5
1 - 3,0,1 - 12
2 - 3,0,2 - 8
4 - 3,0,4 - 15
5 - 3,0,2,5 - 21
6 - 3,0,2,6 - 12
7 - 3,0,1,7 - 19
```

(d) Router 3

```
Routing table for node 4 at time 60
Destination - Path - Cost
0 - 4,0 - 10
1 - 4,0,1 - 17
2 - 4,0,2 - 13
3 - 4,0,3 - 15
5 - 4,5 - 12
6 - 4,0,2,6 - 17
7 - 4,0,1,7 - 24
```

(e) Router 4

```
Routing table for node 5 at time 60
Destination - Path - Cost
0 - 5,2,0 - 16
1 - 5,2,0,1 - 23
2 - 5,2 - 13
3 - 5,2,0,3 - 21
4 - 5,4 - 12
6 - 5,2,6 - 17
7 - 5,2,0,1,7 - 30
```

(f) Router 5

```
Routing table for node 6 at time 60
Destination - Path - Cost
0 - 6,2,0 - 7
1 - 6,2,0,1 - 14
2 - 6,2 - 4
3 - 6,2,0,3 - 12
4 - 6,2,0,4 - 17
5 - 6,2,5 - 17
7 - 6,7 - 18
```

(g) Router 6

```
Routing table for node 7 at time 60
Destination - Path - Cost
0 - 7,1,0 - 14
1 - 7,1 - 7
2 - 7,1,0,2 - 17
3 - 7,1,0,3 - 19
4 - 7,1,0,4 - 24
5 - 7,1,0,2,5 - 30
6 - 7,6 - 18
```

(h) Router 7

Figure 4: Outputs obtained for input 2 (fig 2b)

7 Conclusion

Open Shortest Path First(OSPF) routing algorithm is a part of the backbone of the modern day networks. This assignment has been successful in exposing us to the implementation difficulties and challenges one might face in a real-life scenario. Through this assignment, we have gained in utilising UDP datagrams for communication as well as implementing a multi-threaded router which is close to an actual router. All in all, this assignment has proven to be a great exercise in providing us with theoretical clarity as well as technical expertise.

References

[1] UDP Server-Client implementation in C
URL: https://www.geeksforgeeks.org/udp-server-client-implementation-c/

[2] Multithreading in C++ URL: https://www.geeksforgeeks.org/multithreading-in-cpp/

[3] Lecture 10: Intra-domain routing, COMP/ELEC 429 Introduction to Computer Networks, Slides from Edward W. Knightly, T. S. Eugene Ng, Ion Stoica, Hui Zhang