CS3205 A4 REPORT

OSPF Routing Algorithm

Rudra Desai (CS18B012)

20-04-2021
3rd Year, BTech CSE

INDEX

AIM

This project aims to implement a simplified version of the Open Shortest Path First (OSPF) routing protocol.

INTRODUCTION

Open Shortest Path First (**OSPF**) is a routing protocol for Internet Protocol (IP) networks. It uses a link-state routing (LSR) algorithm and falls into the group of interior gateway protocols (IGPs), operating within a single autonomous system (AS). It implements **Dijkstra's algorithm**, also known as the shortest path first (SPF) algorithm.

The algorithm we implemented is divided into 3 parts, a) Exchange HELLO packets b) Exchange and forward LSA packets and c) Run Dijkstra's algorithm. The above specification is simplified for a better understanding of the fundamentals.

EXPERIMENTAL SETUP

- We created a script called ospf.py, which basically works as the router.
- Messages are exchanged between the routers using the UDP protocol.
- Each router acts as a server as well as a client while sending and receiving.
- The server's port number is fixed : (10000 + id), id = id of the router.
- Clients can send and receive from any port.

ENTITIES INVOLVED AND FUNCTIONS USED

1. **Router**
    - There are 4 threads running for each router
        i. Server thread: This thread receives all the messages sent by other routers ex: 'HELLOREPLY' and 'LSA', then, takes actions accordingly
        ii. Hello thread: This thread sends Hello packets after each 'a'(param) seconds
        iii. Lsa thread: This thread sends LSA packets after each 'b' (param) seconds.

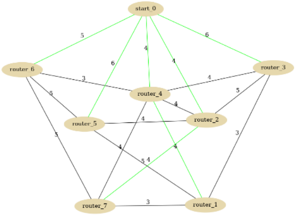
    iv. Spf thread: This thread runs Dijkstra’s algorithm on current topology information and prints the output in the output file

- After each LSA advertisement, the link-state is updated. This way the information about the network is spread.
- **Useful functions:**
    i. **parse_infile():** Parses the input file and updates the router information accordingly.
    ii. **send_msg(), receive_msg()**: Handles the UDP programming part of sending and receiving messages to and from other routers
    iii. **send_hello(), send_hello_reply(), accept_hello_reply():** Handles the HELLO packet updates.
    iv. **send_lsa(), receive_lsa():** Handles the LSA packet updates
    v. **minDistance(), getPath(), dijkstra():** Runs the Dijkstra algorithm on the current Link-state.
    vi. **debug()**: Prints various debug messages to stdout, based on the log level.

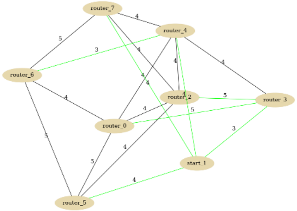
PLOTS

1. **Sample 1**
    - The input file is present in the submission.
    - Following plots were obtained after the completion of this experiment.
    - Here, the final graph including the shortest path is shown for each router.
    - Each green line represents the edge in the shortest path from the start

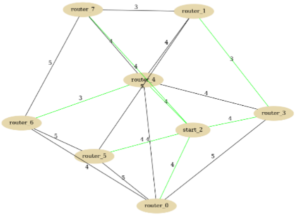
Router 2



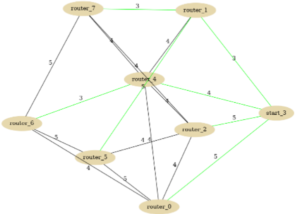
Router 3



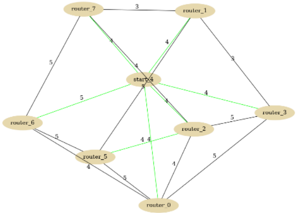
Router 4



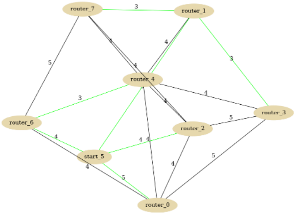
Router 5



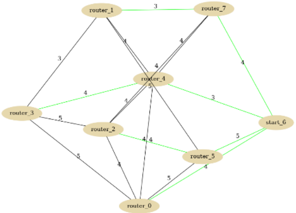
Router 6



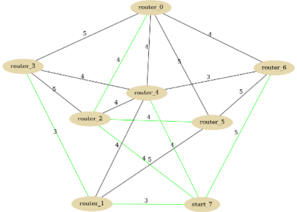
Router 7



Router 8



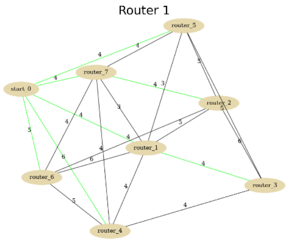
Router 9

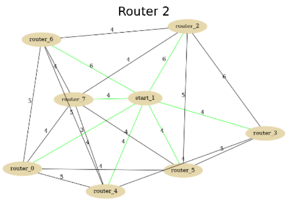


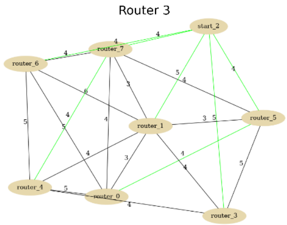
Note : This graph can be viewed properly on this link : link

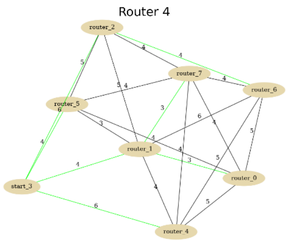
- **Green lines display the edges of the shortest path**
- **Black lines are the edges of the current Link-state for each router**
- **Here, edge weight between a - b may not be equal to edge weight between b - a, due to randomness in the input file and the delay in propagation.**

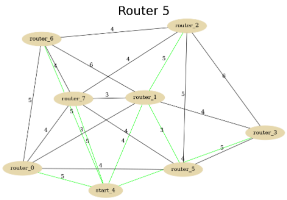
2. Sample 2

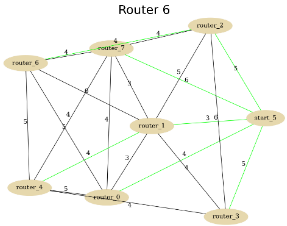


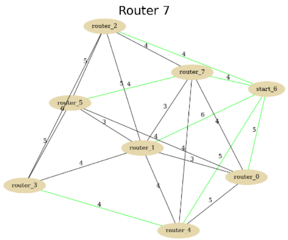


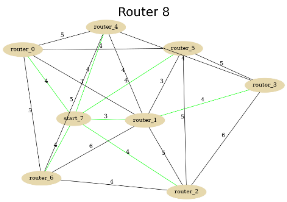












Note : This graph can be viewed properly on this link : link

- **Green lines display the edges of the shortest path**
- **Black lines are the edges of the current Link-state for each router**
- **Here, edge weight between a - b may not be equal to edge weight between b - a, due to randomness in the input file and the delay in propagation.**

RESULTS

1. The above plots displays the shortest path from source to each other routers.
2. The LSA information spread across the network is sufficient for each router to find the shortest path for the next packet it sends.
3. A thing to note in this simulation is that lesser the interval between sending HELLO packets, more is the randomness between graphs.
4. The randomness in the edge updates simulates the real-time delay between routers arising because of various reasons including congestion.

CONCLUSION

Even though the whole experiment was simplified for better understanding, we learned a lot. This experiment helped us understand one of the most important algorithms for routing - OSPF routing algorithm.

ADDITIONAL REMARKS

The simulation could have been made better by actually using multiple devices and then plotting the graphs

REFERENCES

1. https://en.wikipedia.org/wiki/Open_Shortest_Path_First
2. https://www.geeksforgeeks.org/open-shortest-path-first-ospf-protocol-states/