BLG 337E Principles of Computer Communications

Project 4

Naci Toygun Görmüş

150210719

1. Introduction

Project 4 focuses on the implementation of routing algorithms, specifically Link State and Distance Vector Routing. The program dynamically creates a network topology based on user-defined nodes and generates source-destination routes using both algorithms. The visual representation includes route paths, forwarding tables, packet transmission delay, total cost, runtime, and hop counts. This project offers an exploration of routing algorithms within a dynamically generated network environment.

2. Implementation

The project was implemented as a graphical user interface by utilizing the Tkinter toolkit in Python. The screen was divided into four main parts, similar to the previous project, where the upper left quarter of the screen displays the network topology that was randomly generated by using the user input, the upper right quarter displays the results of the computation of all the possible routes in the topology, the lower left quarter displays the forwarding tables for each node in the topology and the lower left quarter displays a input area for the user the input the desired number of nodes in the topology and a checkbox to select the algorithm to be used when computing shortest paths.

While generating the random network topology, the networkx library was utilized and the possibility of an edge to be generated between two nodes is set to be 70%. If an edge is generated, the cost of transmission between those two nodes is selected as a random integer between 1 and 10. Also, the delay of transmission between two nodes is assumed to be proportional to its cost. So the cost of 1 is equal to 1 ms delay. Considering this assumption, end to end delay on a route is calculated. Also, due to the difficulty encountered in fitting and displaying all forwarding tables and all possible route calculations in the GUI for larger number of nodes, log files are utilized when the number of nodes is larger than 5.

3. Guideline for the Simulation

Upon running the executable file, the user can enter a valid integer value representing the desired number of nodes on the graph topology. Then the user should select an algorithm for computing shortest routes on the graph and click the 'Run Simulation' button.

If the chosen algorithm is Link State algorithm, then the program will generate a random network topology and display it on the screen. If the number of nodes is not larger than 5, then the forwarding tables for each node is displayed in the lower left quarter. Also, all possible routes ,except the routes that include the node itself, will be displayed in the upper right corner along with the number of hops, the total cost of that route and the delay for that route. The run time of the Dijkstra algorithm will be printed in the console. If the number of nodes is greater than 5, then these information will be logged into two separate output files, one for the forwarding tables and one for the routing computation results.

If the chosen algorithm is Distance Vector algorithm, then upon generating the random topology, the next iteration button on the lower right quarter of the screen will be enabled. The program waits for the button to be clicked for the next iteration to be started (including the first iteration) to be able to efficiently display the forwarding tables for each node at the end of that iteration. The runtime of that iteration will be printed in the console. At the end of the algorithm, after the algorithm converges and the shortest routes have been calculated, the results will be displayed at the upper right quarter of the screen. The rule for number of nodes being larger than 5 also applies in this algorithm to be able to fit the results on the screen.

Small Note: For smaller number of nodes n, the Python time library can misinterpret the running time as 0 seconds. But works fine for larger number of nodes.

4. Outcomes and Observations

After finishing Project 4, I found out some interesting things about how the routing algorithms worked and how they performed in different situations. I saw that both Link State and Distance Vector Routing algorithms have their own ways of making decisions. I learned that each algorithm has its own strengths and weaknesses, and they work differently depending on the network setup. Seeing the routes on the network map helped me understand how the algorithms make decisions. Having clear visuals made it easier to figure out the best paths and spots where things might not work as well. The forwarding tables showing how each node makes decisions gave me a good look at how stable and fast the routing algorithms are. Looking at these tables helped me understand how well the algorithms adapt to changes in the network. After measuring different things like how long it takes for packets to travel, the total cost of the chosen path, how much time the algorithms took to run, and how many hops there were, these numbers helped me compare how well the algorithms performed in different situations. The program could make networks with different shapes based on what the user wanted. This flexibility allowed me to see how well the algorithms worked in various setups, helping us understand their real-world usefulness.

5. Conclusion

In summary, Project 4 was a useful learning experience. It let me apply what we learned about routing algorithms in a practical way. Actually doing things with the algorithms helped me understand how computer networks and routing work in the real world. The project taught me a lot about how Link State and Distance Vector Routing algorithms perform and adapt in different situations. The outcomes and observations help me understand these important concepts in computer networking.