Comp 4322 Internetworking Protocols, Software and Management

Term Project: Link State Routing

Due Time: 23:00pm, April 10th, 2022

Project objectives

This project aims to develop a Java program that emulates the process done by routers exercising the link state routing (LSR) protocol.

The LSR protocol dictates the interactions between routers in terms of exchanging link state advertisements (LSAs). The flooding technique adopted by the LSR protocol ensures that up-to-date state information is propagated to all the routers within a routing area. Importantly, these information are collected by individual routers and used as the primarily source of state information to compute the shortest path based on the Dijkstra's algorithm. The combination of LSA packets provides adequate information for each router to take a snap-shot of the network topology and compute the shortest paths to all other routers based on the Dijkstra's algorithm. The router computing the shortest path is known as the source router or the root node, from which a tree path represents the shortest paths to all other nodes.

In this project, students need to form groups with 3 members, and each group is required to write a Java program that processes LSA packets and computes the shortest paths from the source router to all other nodes. There are several basic requirements that are mandatory in this project, while others are optional. The next section describes the design requirements and specifications of the project.

Design requirements and specifications

Listed below are some mandatory design requirements and specifications of the project:

- ***Receive LSA packets from a file***

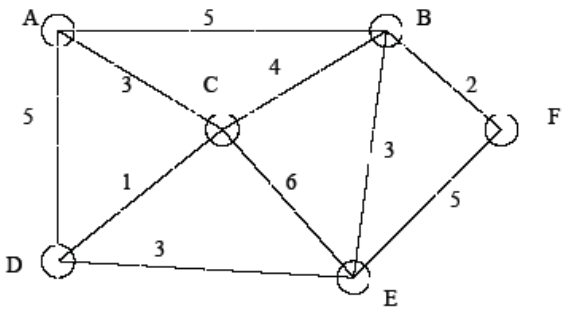


Figure 1. An example of network topology

The LSA packets to be processed by a router are assumed to be captured from an ASCII formatted text file, with the extension of lsa (e.g. routes.lsa). The LSA information contained in the file can fully describe the topology of the network that consists of both nodes and links, and the cost metrics associated with the links. For a network topology as shown in Figure 1, the format of the LSA packet is described below:

A: B:5 C:3 D:5
B: A:5 C:4 E:3 F:2
C: A:3 B:4 D:1 E:6
D: A:5 C:1 E:3
E: B:3 C:6 D:3 F:5
F: B:2 E:5

That is, in the LSA file, every node in the network must have one and only one line representing the state information of the node and its associated neighbor(s). For each line, the first column represents the node identifier; then a list of weighted links to its neighbor(s) is included. Each weighted link here is separated by a space (or multiple spaces). The weighted link is denoted as “to_node:weight”, where to_node represents a neighbor and weight is the cost of using the link to the neighbor. To simplify computation, the cost is applicable to the link in both directions, i.e. if the link cost from A to B is 5, then the link cost from B to A is also 5.

- ***Execution of the program***

You can write the program with a user-friendly GUI or just a command-line (higher marks will be awarded for GUI). In the case of using the command-line interface, a user can type the following command:

java LSRCompute routes.lsa A

where LSRCompute represents the program, routes.lsa represents the file, and A indicates the computation is from the perspective of node A.

- ***Execution mode***

Your program needs to support both single-step and compute-all modes. In the single-step (SS) mode, the program will run single step through the LSR computation to allow the user to trace the computation path. For each step, the program will search for the next node to visit and, compute and display the node found. For example, if a new node F has been found during current step, the program will display a status line:

Found F: Path: A>B>F Cost: 7 [press any key to continue]

The single step function will continue to operate until the last node in the network is visited, in which case the program will produce a summary table to show all the paths to all nodes from the source. The summary table may look like this:

Source A:
B: Path: A>B Cost: 5
C: Path: A>C Cost: 3
D: Path: A>C>D Cost: 4
E: Path: A>C>D>E Cost: 7
F: Path: A>B>F Cost: 7

In the compute-all (CA) mode, the computation is performed by one command and only the summary table shown above is displayed. In the command-line interface, the command may look like this:

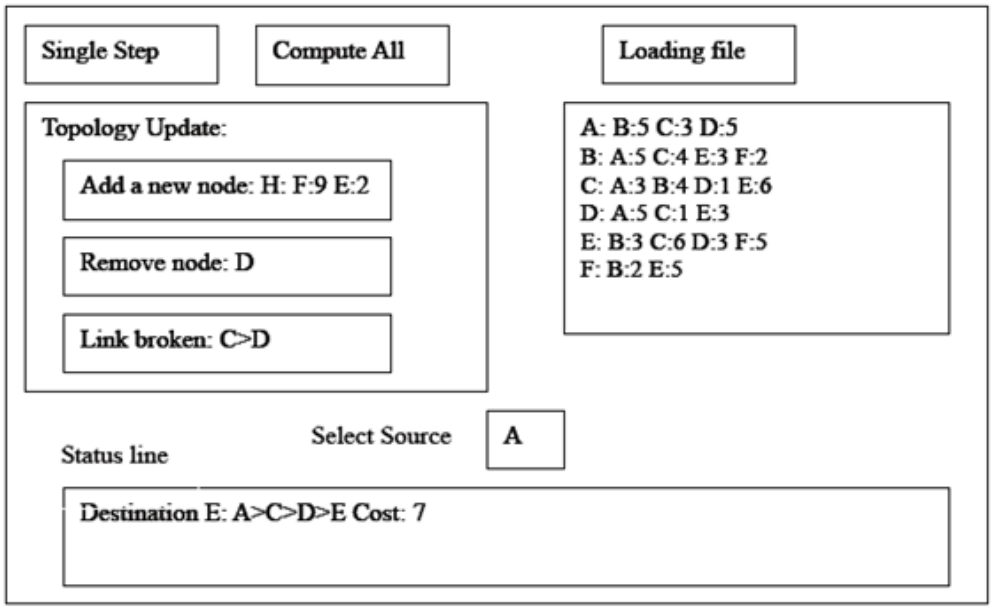
java LSRCompute routes.lsa A SS|CA

where SS represents the single-step mode and CA represents the compute-all mode, | means either the SS or CA mode can be selected.

Optional features

This project is flexible in that you are completely free to exercise your creativity in implementing the program. Besides the following options I list here, you can implement your own features (you will get higher marks with more creative features).

1. You can choose any GUI as you like. E.g., your program uses a GUI interface that may look something like this:



2. Your program can implement dynamic network topology. Initially, you have designed a network, and write all link information into the file routes.lsa. You can add new nodes into the network, or remove some existing nodes from the network, or just break down some links, then show the results. To add this feature, you need to modify the routes.lsa file through your program instead of manually changing routes.lsa file.

There are many references that can be found in the web. Feel free to study them as references but DO NOT COPY. Understand those references and apply them to your program if necessary. Give appropriate references and acknowledgments to those parts that your group has made references to in your report.

Submission requirements

Each group needs to prepare the following documents for the submission:

- Completed source code.
- A project report that contains:
    - Summary of the LSR algorithm
    - Summary of your design and implementation of the LSR protocol, including the architecture design, the functionality of each class, interfaces, UML diagrams, test results, and references. You are expected to give a high-level summary of your project (Do NOT include your source code here!).
    - A section describing the role each member played in the group. All members in the group will receive the same marks unless explicitly requested. In which case, justifications are required.
    - In the front page of the report, please state clearly the members in your group including their names and student IDs.
- A demonstration
    - Each group needs to give a 10-minute demonstration and explain the features (particularly the unique features) of the project.
    - The timeslot will be arranged for each group to present and demonstrate the project at the end of the semester.

Each group is required to submit the following documents before the due time:

- A softcopy of the project report and a package of source code through the Blackboard.
- A hardcopy of the report on the class.

Assessment rubrics

The following rubrics will be used to evaluate your project quality and to determine your grade (100%):

- Summary of the LSR algorithm (20%)
    - Correctness, easy to read, no grammatical errors or typos, high quality illustrations, full bibliography, etc.
- Summary of your design and implementation of the LSR protocol (30%)
    - Correctness, thoroughness, easy to read, no grammatical errors or typos, user-friendly interfaces, well-discussed test results, etc.
- Quality of your project's source code (35%)
    - Good naming and coding convention used in your source code
    - Correct implementation of the required functionalities of the LSR protocol
    - Implementation of optional features for the project
- Quality of your demonstration of your project (15%)
    - Present the project overview clearly
    - Compile the source code successfully
    - Execute the program without runtime errors
    - Successfully demonstrate the program's functionalities and ease of use
    - Manage the Q&A section smoothly

Comp 4322 Internetworking Protocols, Software and Management
Term Project Marking Sheet

<table>
<tr><td colspan="3">Group No:</td></tr>
<tr><td colspan="3">Members and contributions:</td></tr>
<tr><td>Name</td><td>Student ID</td><td>Contributions</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td colspan="3">Check List:</td></tr>
<tr><td colspan="2">• Summary of the LSR algorithm (20%)
- Correctness, easy to read, no grammatical errors or typos, high quality illustrations, full bibliography, etc.</td><td></td></tr>
<tr><td colspan="2">• Summary of your design and implementation of the LSR protocol (30%)
- Correctness, thoroughness, easy to read, no grammatical errors or typos, user-friendly interfaces, well-discussed test results, etc.</td><td></td></tr>
<tr><td colspan="2">• Quality of your project’s source code (35%)
- Good naming and coding convention used in your source code
- Correct implementation of the required functionalities of the LSR protocol
- Implementation of optional features for the project</td><td></td></tr>
<tr><td colspan="2">• Quality of your demonstration of your project (15%)
- Present the project overview clearly
- Compile the source code successfully
- Execute the program without runtime errors
- Successfully demonstrate the program’s functionalities and ease of use
- Manage the Q&A section smoothly</td><td></td></tr>
<tr><td colspan="3">Total</td></tr>
</table>