Programming project - Implementation of Link State Routing Protocol

(30% of continuous assessment grade)

Group Size: 3 students

Deliverables: A project report (20%) **AND** a demo (20%)

Purpose

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

The LSR protocol primarily dictates the interactions between routers in terms of exchanging link state advertisements. The flooding technique adopted by LSR protocol ensures that up-to-date state information is rapidly propagated to all the routers within a routing area context. Importantly, these information are collated by individual routers and used as the primarily source of state information to compute the shortest path based on Dijkstra's algorithm. The combination of Link State Advertisement (LSA) packets provides adequate information for each router to take a snap shot of the network topology and the metric costs of using the links in the effort to compute the shortest path. The router computing the shortest path is known as the source router or the root node, in which a tree path representing the shortest path to all others nodes are computed based on the Dijkstra algorithm.

In this project, your group is required to write a standalone java program that processes LSA packets and to compute the shortest path from the source router to all other nodes. There are several basic requirements that are mandatory in this project, while others are fully optional. The next section describes the design requirements and specifications of the project.

Design Specifications

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

Process LSA packets from a file

The LSA packets to be processed by the router are assumed to be captured in an ASCII formatted text file, with the extension of lsa (e.g. routes.lsa). The LSA information contained in the file is able to fully describe the topology of the network and its cost metrics associated to the links. The format of the LSA packet fields of a network topology as shown in Figure 1 are 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

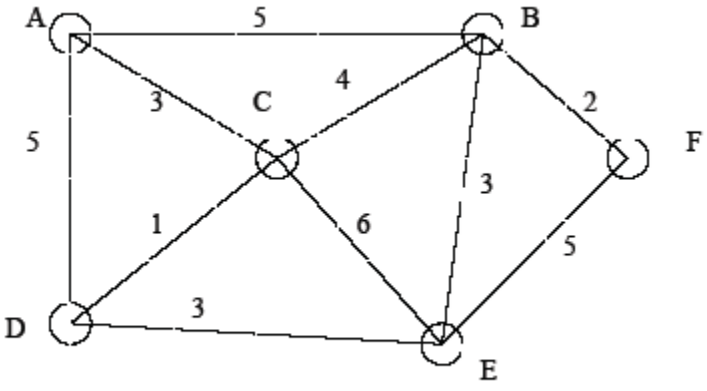


Figure 1. An example of network topology

Each field is separated by a space (or multiple spaces). The first column represents the node identifier. Every node in the network must have one and only one line representing state information of the node and its associated neighbor(s). To simplify computation, the link cost is applicable in both directions, i.e. if link cost from A to B is 5, then the link

cost from B to A is also 5. The links are described in notation to_node:weight, where to_node represents the destination node and weight is the cost of using that link to the node. The last line represents the last node that closes the topology and must be included as one of the destination nodes in the previous lines, else this is not a well-formed topology. You can write you program with GUI or just command prompt based (higher marks will be awarded for GUI). In the case of command prompt, user can type the following command:

java LSRCompute routes.lsa A

where routes.lsa represents the file and A to initiate computation with source anchored at A. In other words, the computation is from the perspective of router A.

Single Step versus Compute All

Your program should support both single step and compute all modes. In single step (SS) mode, your program will single step through the computation to allow user to trace the computation path. With each step, your program will search for the next node to visit and, to compute and display the node found. For example, a new node D may have been found during current iteration, the program will display a status line:

Found D: Path: A>C>D 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 mode, the computation is performed in one go and only the summary table shown above is displayed. In command prompt interface, the command to issue may look like this:

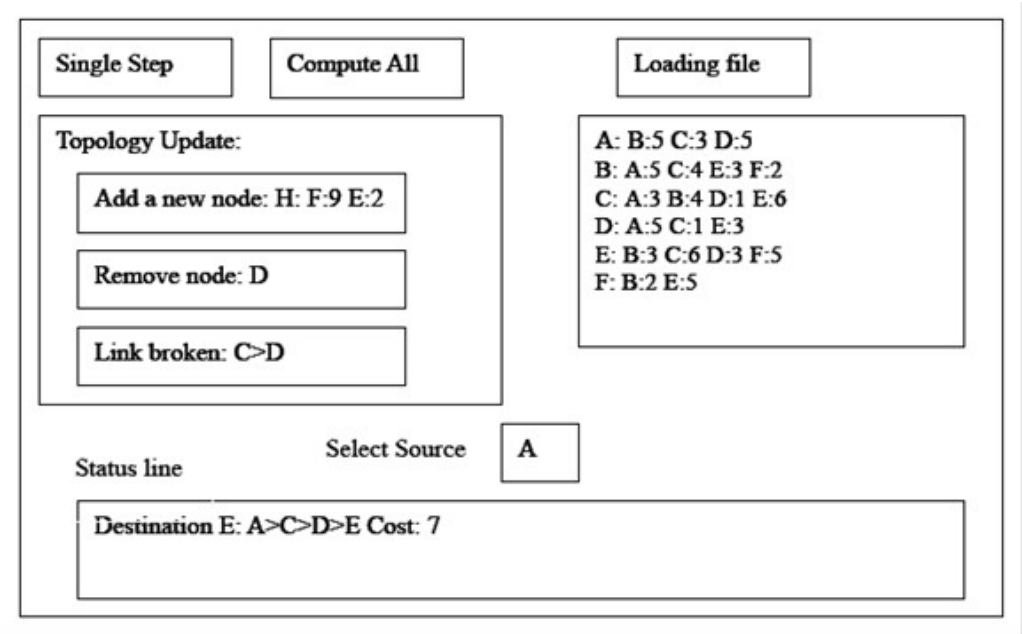
java LSRCompute routes.lsa A SS|CA

where SS represents single step or CA which represents compute all mode.

Optional Features:

This project is highly flexible in that you are completely free to exercise your creativity in implementing the program. Except the following options I suggested, you can create your own features, you will get higher score with more creative features.

(1) Use GUI interface that may look something like this:



You can choose any GUI as you like.

(2). Implement dynamic network topology. Initially, you have designed a network, and load 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 broken 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 the algorithm and apply it to your program if necessary. Give appropriate references and acknowledgments to those parts of your report that your group has made reference to.