CS 571: Computer Networks **Spring 2021**

Programming Assignment I: The Extended LAN

Assigned: Feb. 11, 2021 *Due: Feb. 25, 2021*

1 Objective

In this programming assignment, you will write a **C**, **C++** or **Java** program to simulate the spanning tree algorithm for extended LANs. The objective of the assignment is to understand how the distributed version of the spanning tree algorithm works.

2 Input

The extended LAN will be given in a text file. We make several assumptions to simplify the implementation. An extended LAN with n bridges has $1, 2, 3, \cdots, n$ as the IDs of the bridges. If there are m LANs, they are represented by first m capital letters. The file contains n lines, one for each bridge. Each line begins with the ID of one bridge, followed by the list of the LANs it is connected to. The names of LANs are separated by exactly one space. For example, the extended LAN in Figure 3.10 on page 194 of the book can be represented as

```
1 D E F G H
2 C E
3 A C
4 H I J
5 A B D
6 G I
7 B F K
```

3 Spanning Tree Algorithm

3.1 States at Each Node

Node best configuration: Every bridge (node) records its own best configuration (which bridge is the root and the distance to the root) and how it got this best configuration (from which port and from which node). There are two cases: 1) it claimed itself to be the root; 2) it received a message from some node over one of its ports.

Best configuration for each port: In addition, every node records the best configuration for every port it has. It is the best message among the messages received over this port and the message the node itself may send over the port.

3.2 Initialization

At the beginning, each node considers itself to be the root. $< ID, 0, ID >$ should be the best configuration for the bridge and for each port, where ID is the ID of that bridge. Every port of the bridge should be open.

3.3 Sending and Processing Messages

We can randomly select a node to send its message. It only sends out to those ports, over which it has not received a better message. All nodes connected to that link will receive the message.

Upon receiving a message over a link, the node will determine whether the message is better than its configurations. There are three scenarios:

1. it can ignore the message, because the message is worse than its best configuration for that port [1];
2. it needs to change the best configuration at that port (and nothing else), because the message received is better for that port; and
3. in addition to changing the best configuration at a port as in the previous scenario, it needs to change its own node best configuration (and the port it got the node best configuration). Assume that the current best configuration of the node is $< R, d >$. This includes two cases: 1) $R = ID$; or 2) previously, one of its ports has the best configuration because it has received a message $< R, d-1, S >$ from node S. Now, it receives a message $< R', d', S' >$ that is better than its current best configuration at a port. If $< R', d'+1, S' >$ is better than $< R, d, S >$ [2], then the best configuration for the current node needs to be changed and information about from which port and node the current node gets the best configuration also needs to be changed. This can lead to changing the best configurations of its other ports. So the program needs to check each port.

For every node, it can determine that a port should be open, if 1) the sender ID in the best configuration of the port is its own; and 2) the port is the one from which it got its node best configuration.

All other ports should be closed.

With enough number of messages sent, we will be able to generate the spanning tree for the extended LAN.

4 Implementation and Test

You can implement a bridge as a class in C++ and Java.

In C, you can have an array of a bridge structure, each recording the status of one bridge. Then write a procedure for sending a message by the i-th bridge.

For the testing purpose, your program should accept a list of numbers indicating the order of nodes sending messages. For example, `myprogram LANfilename 1 3 4 6 2 3 1 5` means that nodes 1, 3, 4, 6, 2, 3, 1, and 5 will send messages in that order. The program should print out the best configuration for each bridge, and which ports are open and which ports are closed for each bridge after execution.

5 Submission

You need to write a README file to give a general description of your programs, and state any limitations of the implementation. You should also include instructions on how to compile/link and run your programs. A Makefile is encouraged, but not required. Your programs will be tested on a Linux environment similar to OpenStack VM. Comments are required. You should tar or zip all the files together and submit one tar/zip file.

[1] The exception is that the sender in the message is the same as the sender in the best configuration for that port. In this case, it may lead to changing the node best configuration. However, this does not need to be considered in this assignment.

[2] Actually, in case 1, $< R, d, ID >$ is $< ID, 0, ID >$; and in case 2, $< R, d >$ is the best configuration of the current node and S is the ID of the node from which the current node gets its best configuration.

Appendix

Example 1: If we run the program as

```
   myprogram LANfilename 1 3 4 6 2 3 1 5
```

the output will be:

```
Bridge 1: best configuration <1, 0>, from 1
    port D: <1, 0, 1>  open
    port E: <1, 0, 1>  open
    port F: <1, 0, 1>  open
    port G: <1, 0, 1>  open
    port H: <1, 0, 1>  open

Bridge 2: best configuration <1, 1>, from 1  via E
    port C: <1, 1, 2>  open
    port E: <1, 0, 1>  open

Bridge 3: best configuration <1, 2>, from 2  via C
    port A: <1, 1, 5>  closed
    port C: <1, 1, 2>  closed (open is also acceptable)

Bridge 4: best configuration <1, 1>, from 1  via H
    port H: <1, 0, 1>  open
    port I: <1, 1, 4>  open
    port J: <1, 1, 4>  open

Bridge 5: best configuration <1, 1>, from 1  via D
    port D: <1, 0, 1>  open
    port A: <1, 1, 5>  open
    port B: <1, 1, 5>  open

Bridge 6: best configuration <1, 1>, from 1  via G
    port G: <1, 0, 1>  closed  (open is also acceptable)
    port I: <1, 1, 4>  closed

Bridge 7: best configuration <1, 1>, from 1  via F
    port F: <1, 0, 1>  open
    port B: <1, 1, 5>  closed
    port K: <1, 1, 7>  open
```

Example 2: If we run the program as

```
   myprogram LANfilename 1 4
```

the output will be:

```
Bridge 1: best configuration <1, 0>, from 1  via none
    port D: <1, 0, 1>  open
    port E: <1, 0, 1>  open
    port F: <1, 0, 1>  open
    port G: <1, 0, 1>  open
    port H: <1, 0, 1>  open

Bridge 2: best configuration <1, 1>, from 1  via E
    port C: <1, 1, 2>  open
    port E: <1, 0, 1>  open

Bridge 3: best configuration <3, 0>, from 3  via none
    port A: <3, 0, 3>  open
    port C: <3, 0, 3>  open

Bridge 4: best configuration <1, 1>, from 1  via H
    port H: <1, 0, 1>  open
    port I: <1, 1, 4>  open
    port J: <1, 1, 4>  open

Bridge 5: best configuration <1, 1>, from 1  via D
    port D: <1, 0, 1>  open
    port A: <1, 1, 5>  open
    port B: <1, 1, 5>  open

Bridge 6: best configuration <1, 1>, from 1  via G
    port G: <1, 0, 1>  closed (open is also acceptable)
    port I: <1, 1, 4>  closed

Bridge 7: best configuration <1, 1>, from 1  via F
    port F: <1, 0, 1>  open
    port B: <1, 1, 7>  open
    port K: <1, 1, 7>  open
```