DS Assignment

Release date: 21 February 2020
Assignment deadline: **March 24, 2020, 4pm**
Feedback return: April 14, 2020

This assignment is worth **25%** of the final mark. The assignment is a programming exercise that will be marked out of 100.

Assignment Description

In this assignment, you have to create a simulation of a wireless network and implement the ring based Change and Roberts algorithm for leader election among the nodes of the network. You are given Java code for the Node and the Network classes, with basic methods implemented as a starting point. You are also given the input file which contains the network; i.e., the nodes with their neighbors.

System design

Your implementation should simulate a distributed system using multiple threads. Each node must run on its own thread. Additionally, a different thread will simulate the actions of the network delivering messages from one node to another. You can only use strings as messages.

Communication is assumed to be synchronous. Each synchronous round lasts for 20ms. Your program should simulate this behavior of the network receiving and delivering messages exactly once every round.

A node is allowed to send messages only to its neighbors as specified in the input file describing the network graph. It can send at most one message to each neighbor in one round. The network should enforce these constraints. You are also allowed to use a BROADCAST message that goes to all neighbors of a node.

Input file specification

There are three types of input files: graph.txt, elect.txt, fail.txt.

The input file graph.txt, contains the network graph. Each line describes a node; the first the item is the id of the node, and the following ones are its neighbors. The ordering of the rows gives an ordering of the nodes on the ring (the first follows the last).

The input file elect.txt contains a list of leader elections initiated by different sets of nodes. These lines start with ELECT followed by the round number, followed by nodes that start election at that round. For example:

ELECT 5 18 7

Means that in round 5, nodes 18 and 7 start leader election.

The input file fail.txt contains the lines describing nodes failing. The file begins with a single ELECT statement like the ones in elect.txt. Following this line, there are FAIL lines e.g.

FAIL 100 20

means that at round 100 node 20 fails. A sample set of input files is available on the assignment page.

Assignments

The assignment is divided into two parts. Your tasks for each part are as follows:

Part A (marks 70/100)

1. Read the network specification from the input file.
2. Construct the ring topology network based on the order in which the nodes appear in the input file.
3. Implement the Chang and Roberts algorithm for leader election among the nodes (details of the algorithm can be found in the course slides as well as in the suggested references); execute leader elections as

Your implementation should have a protocol to simulate the network. Communication must be synchronous. In your protocol, you should implement features that ensure proper communication between the nodes. (Hint: Nodes communicate by exchanging messages. At each round, the network can maintain a "list" of all the messages the nodes want to deliver and then perform the delivery.)

Part B (marks 30/100)

In the input file, we provide information on node failures. Your task for this part is to handle such failures in the network. The rounds in which the node fails is mentioned in the input file. Either the failing node is the current leader or a non-leader node, for as long as the network is connected, your system should initiate and elect a new leader after the failure. The file may have one or more elections before the failures. When the network becomes disconnected, your system can announce that and terminate.

There is no one way to recover from a failure, so the assignment is open ended where you have to design your own solutions. You will be marked on your approach, ideas and their implementation rather than perfect outputs.

Messages:

The types of messages the nodes can exchange are the following:
- election messages, when a node announces the initiation of an election process
format: **ELECT node_id**, where node_id is the id of the node initiating the election process

A node, upon receiving an election message can send two types of messages:
- forwarding messages – if its id is less than the current maximum id or if the node is a non-participant, as described for the algorithm in class
format: **FORWARD max{node_id, received_node_id}**, where node_id is the id of the node forwarding the message and received_node_id is the current maximum id
- leader messages – if its id is equal to the current maximum id
format: **LEADER node_id**, where node_is the id of the current node

Input

Create a shell script called run.sh. We will run your script with a graph file and an events file, e.g:

./run.sh graph.txt elect.txt

Or as:

./run.sh graph.txt fail.txt

With the graph file first and events/fail file as second argument. Note that we will try different input files, so file names can be different (but the order of the files will be the same) and your code should honor the sequence and handle general file names.

When run.sh is executed, it should compile your code and run it with the given inputs.

Outputs

Your simulator should produce a log file “log.txt” that contains the elected leader. For Part B, your output must contain the sequence of nodes elected as leaders after each failure. The last

line of the output must be “simulation completed”. For example, a log could be in the following format:

Part A
Leader Node 9

Part B
Leader Node 9
Leader Node 6
Leader Node 2
simulation completed

Print to screen: In addition, you should print to screen all important events as created and processed by your program. For example, initiation of elections, node failures, message delivery etc. Each must be on a separate line, and contain the round number, entities involved and contents.

Submission Instructions

You are expected to use **Java**. If you have a strong reason to use a different language, discuss it with us first. However, we strongly suggest that you use Java, for which we provide code to help you.

The code will be tested on DICE. **If it does not run directly on DICE, we cannot evaluate it**.

The submission should contain the following in a single folder:
1. Your source code. **NOT just the executable**. If the source code is not included in the submission folder, your submission cannot be evaluated. The source code must be well commented.
2. A shell script called run.sh. This must compile and run your program when executed, and should take the input file as a parameter e.g. ./run.sh input.txt
3. A readme file (max 1 page in 12 pt font, 1-inch margin) that contains a summary of any specific implementation decisions you have made and a description of the logic you followed for Part B. This is necessary in order to receive full marks (pdf, max 1 page in 12 pt font, 1-inch margin).

Please notice that if your implementation does not generate the **log.txt file** with the output written in it, you cannot be given full marks.

****IMPORTANT**** Output and submissions must conform to specifications given above. For example, log.txt must be generated and in the format above. All files must be in the top level

folder. Do not submit an IDE (like Eclipse) “project”, and make sure that the files work in flat source code format. If your submission does not conform to specifications, we will not evaluate it.

Always use the most recent description on the web page. That is, use the up to date one on the web page and not a previously downloaded version.

University regulations:

On good Scholarly Practice. Please remember the University requirement as regards all assessed work. Details about this can be found at:

http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

Remember, if you use ideas from elsewhere (including other students), cite them. It is easy to detect when code or descriptions are copied from other students or the web. So avoid doing that and getting into plagiarism charges.