Distributed Computing: Spring 2017

Programming Assignment 2: Implementing Distributed Termination Detection Algorithms

Submission Date: 15th March 2017, 11:00 pm

Goal: The goal of this assignment is to implement two distributed termination algorithms: (1) Huang's weight-throwing algorithm and (2) Spanning Tree based token algorithm. Implement both these algorithms in C++. Then, you have to compare the message complexity of both the algorithms. and detection latency.

The Application. Implement a distributed system consisting of n nodes, numbered 0 to $n-1$, arranged in a certain topology (*e.g.*, ring, mesh etc.). At the time of the project demonstration, the TAs will run your program with a topology specified in a configuration file (like the previous assignment).

All channels in the system are bidirectional, reliable and satisfy the first-in-first-out (FIFO) property. You can use a reliable socket connection (TCP or SCTP) to implement a channel. *For each channel, the socket connection should be created at the beginning of the program and should stay intact until the end of the program.* All messages between neighboring nodes are exchanged over these connections.

All nodes execute the following protocol:

1. Initially, each node (also denoted as a process) in the system is either *active* or *passive*. This is specified in the input configuration file.

2. When a node N_i is *active*, it sends a certain number of messages randomly chosen between $minMsgLt$ and $maxMsgLt$ to its neighbors. Suppose N_i has k neighbors. Then it randomly chooses a neighbor out of k and then sends a message. N_i sends messages with a delay that is exponentially distributed with an average μ-$delay$. After sending M messages to its neighbors where M is randomly chosen to be such that $minMsgLt \le M \le maxMsgLt$, N_i becomes passive.

   An *active* node on receiving the message does not do anything.

3. Only an *active* node can send a message.

4. A *passive* node, on receiving a message, becomes *active* and follows the steps mentioned in Step 2. But to ensure that the computation eventually terminates, we add one extra rule: Once a node N_i has sent $maxSent$, it can't become active on the receipt of any new message.

Also ensure that all the messages sent by each node is logged onto a local buffer along with the time-stamp in the format: "Time: Message sent by N_i to N_j". We refer to the protocol described above as *Random Multicast* or *RM* protocol.

The Distributed Termination Detection Algorithm. As mentioned in the goals section, implement (1) Huang's weight-throwing algorithm and (2) Spanning Tree based token algorithm. Implement both these algorithms in C++/Java. Then, you have to compare the message-complexity of both the algorithms. and detection latency.

Notes:

- The book assumes for Huang's weight-throwing algorithm, every node is connected to every other node. This need not be true for the input topology.
- For detecting the termination, the book assumes the presence of an external node. This again need not be true. One of the pre-defined nodes in the system can take the responsibility of detecting the termination.

Input: The input to the program will be a file, named inp-params.txt, consisting of all the parameters described above and the graph topology. The first line of the input file will contain the parameters: $n, minMsgLt, maxMsgLt, \mu\text{-}delay, maxSent$.

In addition to inp-params.txt, you will have another file describing the topology of the input graph which is named as topology.txt. The format of the file is as follows:

The first line will be number of nodes in the system which is n. The next line will show the binding of nodes to IP addresses. For instance if n is 3, then the following is a binding:
1 - 192.168.1.1
2 - 192.168.1.2
3 - 192.168.1.3

After describing the address binding, The next few lines will contain the graph topology in the form of an adjacency list. For n as 3, a sample topology showing a complete graph is as follows:
1 2 3
2 1 3
3 1 2

Since you have to implement the spanning-tree based termination detection algorithm, you can specify a spanning tree in this topology file. Each line will show the parent and its children with the first line showing the children of the root. For instance a spanning tree with n being 7 as follows:
1 2 3
2 4 5
3 6 7

Clearly it can be seen that node 1 is the root.

Output: Your program must produce an output for each algorithm. The output is the combined log of all the events. It must show all the initial active processes, the timestamp when a node sends a message to another node, when a node becomes passive and active again etc. Finally, it must also show the announcement of detection of termination by the algorithms. For instance a sample output with n as 10 is as follows:

```
Initially nodes 2 5 7 8 are active.
10:00 Node 2 send a message to Node 4.
10:01 Node 5 send a message to Node 7.
10:02 Node 5 send a message to Node 9.
10:03 Node 4 receive a message from Node 2.
10:03 Node 4 becomes active.

    :
    :
    :
```

10:30 Node 2 announces termination

Report: You have to submit a report for this assignment. This report should contain a comparison of message-complexity of both the above mentioned algorithms: (1) Huang's weight-throwing algorithm and (2) Spanning Tree based token algorithm. You must run both these algorithms multiple times to compare the performances and display the result in form of a graph.

The graph in the report will be as follows: the x-axis will vary the number of nodes while the y-axis will show the average message complexity. Finally, you must also give an analysis of the results while explaining any anomalies observed.

Bonus: For bonus 20 points, you can also measure the detection latency for both the algorithms. In this case, you should clearly describe how you are measuring the detection latency. The graph in the report should also show the comparison of the detection latencies of both the algorithms.

Deliverables: You have to submit the following:

- The source file containing the actual program to execute. Name it as: ProgAssn2-<rollno>.cpp
- A readme.txt that explains how to execute the program
- The report as explained above

Zip all the three files and name it as ProgAssn2-<rollno>.zip. Then upload it on the google classroom page of this course. Submit it by **15th March 2017, 11:00 pm**.