COMP212 - 2020 - CA Assignment 1

Coordination and Leader Election Simulation and Evaluating Distributed Protocol in Java Report

Xuhui Gong

Department of Computer Science

February 2020

Declaration

I confirm that I have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with any other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).

Contents

1 Introduction

This report is an evaluation of the simulation of LCR and HS algorithm in Java. In this experiment, asynchronous loop is used to simulate the leader election between different processors. This report will describe in detail the implementation of LCR and HS algorithm in code, the verification of election correctness and the comparison of algorithm performance which are the number of rounds and the total number of messages transmitted until termination. Finally, the advantages and disadvantages of two different algorithms are pointed out, that is, LCR is superior to HS algorithm in most cases in the number of round and message times.

2 Function and Implementation of the Simulator

2.1 Function of the Simulator

The simulator can modify parameters in the main function, such as the number of simulation cycles, the increasing number of processors, the allocation of processor ID and a single scenario in a random situation time. And in each simulation, the simulator will verify the correctness of the two algorithms. When the verification fails, the program will have several errors to the output file. Finally, number of rounds and the total number of messages will output into the file. Then the data is processed by the special data processing tools to print diagram.

2.2 Implementation of the Simulator

First of all, this simulator uses asynchronous simulation synchronization to simulate the algorithm. In fact, multithreading is more conducive to simulate the real communication between different processors. However, since multithreading may cause unnecessary deadlock, this report does not consider multithreading. The flow of the two algorithms in Java implementation is shown in the figure 1.

For verification, after each algorithm, the simulator will carry out a maximum ID verification, and use bubble sorting to find the maximum ID in *NodeList* and compare it with the results of two kinds of election algorithms. In addition, the verification program also focuses on verifying the multiple uniqueness of leaders, if there is an error in the result, the program outputs the error to a file.

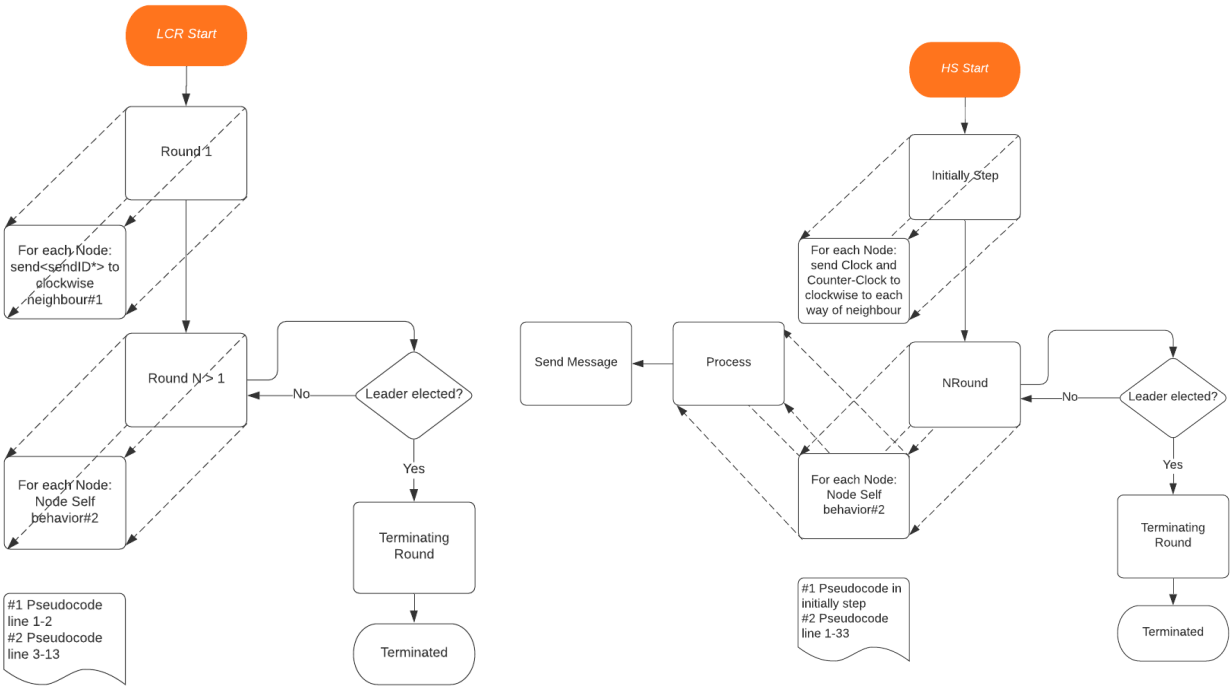


Figure 1: Two Algorithms in Java Implementation

At the beginning of the program, the user can choose the sorting method of nodes according to the user's requirements. In the implementation of the two algorithms, the program put the generated nodes into an *ArraryList*, and declare the previous and next nodes of the processor in the parameters of the node, because this is convenient for the call of the two algorithms. This part can be understood as binding processors together in the way of ring.

For the specific LCR algorithm, the simulation program provides a cache for each processor, which is used to store the values sent by neighboring processors. And after all processors receive the value (some processors do not send the value, in order to prevent null pointer, this program sets the ID size to 0, because 0 does not trigger any decision-making conditions to process itself, so 0 will not be recorded in the number of information transfers), the processor has a corresponding send buffer, and the value in this buffer will be transferred to the cache of adjacent processors after sending.

For HS algorithm, it is more complex. At this time, a processor defines two directions of cache to store the value received from two directions. Correspondingly, the buffer used to send is also changed from one to two. As shown in Figure 1, the HS algorithm first initializes, that is, each processor prepares the information to be sent in order, and then enters the loop. The exit condition of the loop is to find the leader. In the loop, each processor will make a series of judgments to determine whether it has passed the

received information, then each processor sends (the specific implementation of this algorithm still uses 0 to indicate that no information is sent to prevent null pointer).

Finally, each algorithm has the terminated step. However, it is different between the LCR and HS. For LCR, when the leader is elected, the leader will send a message in a clockwise direction. When the message returns from the beginning to the end, all processors will terminate. For HS, when the leader is elected, the leader will send his message from both directions at the same time, and when the processor receives the message, it will terminate.

3 Experiment

The experimental results are divided into three parts. The first part is when the processor ID is sorted in increasing clockwise order, the second part is in clockwise decreasing order, and the third part is random order. In the experiment, 100 processors are counted, and each experiment is repeated 100 times. When a single group of experiments is finished, the number of processors is increased by 10 each time until the number of processors reaches 9000 (due to the slow calculation speed due to the relationship between the calculation performance and single thread).

3.1 Increasing Clockwise Order

The increasing clockwise order message passing and round result is shown in the Figure 2 and Figure 3.

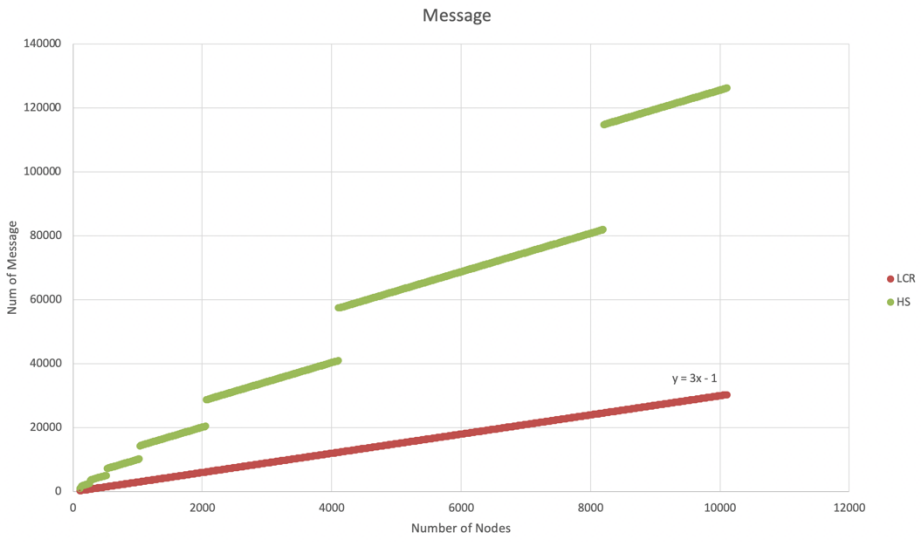


FIGURE 2. INCREASING CLOCKWISE ORDER MESSAGE

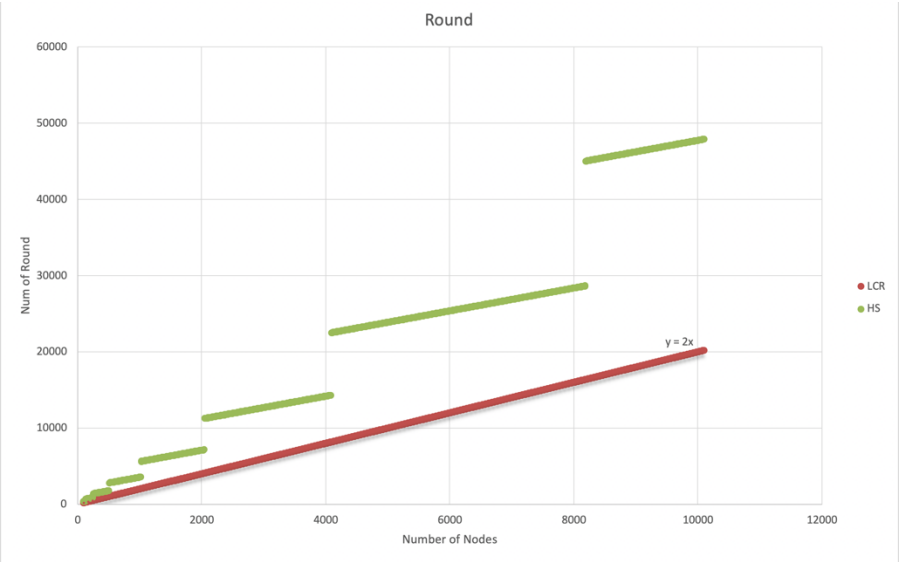


FIGURE 3. INCREASING CLOCKWISE ORDER ROUND

In the case of increasing the processor ID clockwise, the information transfer quantity of LCR algorithm is significantly lower than that of HS. This is mainly because in the ideal case of increasing the LCR clockwise, except for the maximum ID, the rest of the information will be eliminated in the second round. However, HS will send information from both directions at the same time, and the information from the counter-clockwise direction will be sent back, so it causes the waste of information transmission.

From the perspective of round, the number of LCR algorithms increasing in clockwise direction is less than that of HS. The reason is the same as message delivery, HS needs to consume more round to return the information sent by the counter-clockwise processor, which wastes round relatively.

3.2 Decreasing Clockwise Order

The decreasing clockwise order message passing and round result is shown in the Figure 4 and Figure 5. In this case, the advantages of HS algorithm are shown. When the number of ID decreases clockwise, the number of HS information transfer is less than LCR algorithm. Since HS can transmit data from two directions, in fact, the number of HS transmissions is consistent whether increasing or decreasing. However, for LCR algorithm, the number of information transmissions at the worst time is as follows under the condition that the program is not terminated:

$$1 + 2 + 3 + 4 + \ldots + (N - 1) + N$$

If N is a large number, the total message can be treated as: O(N^2) message in non-terminate version.

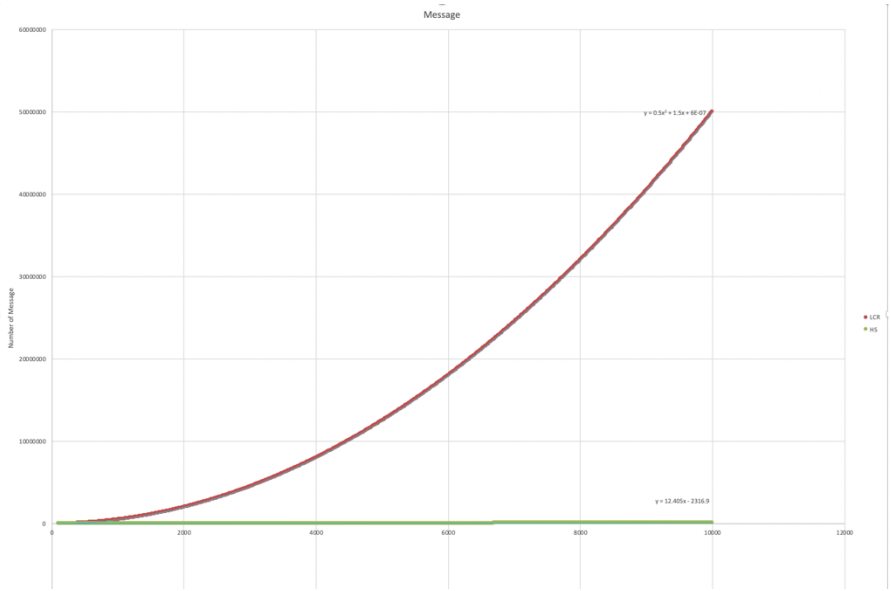


FIGURE 4. DECREASING CLOCKWISE ORDER MESSAGE

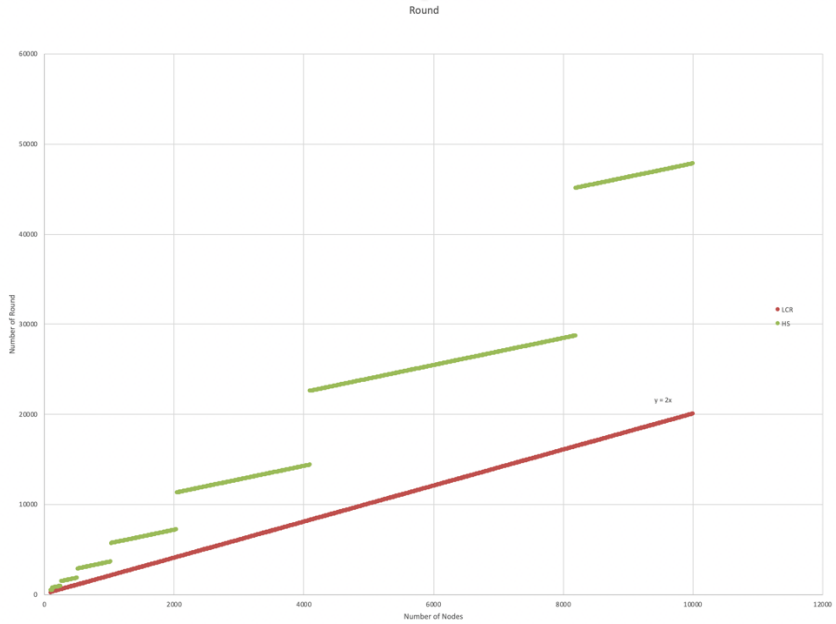
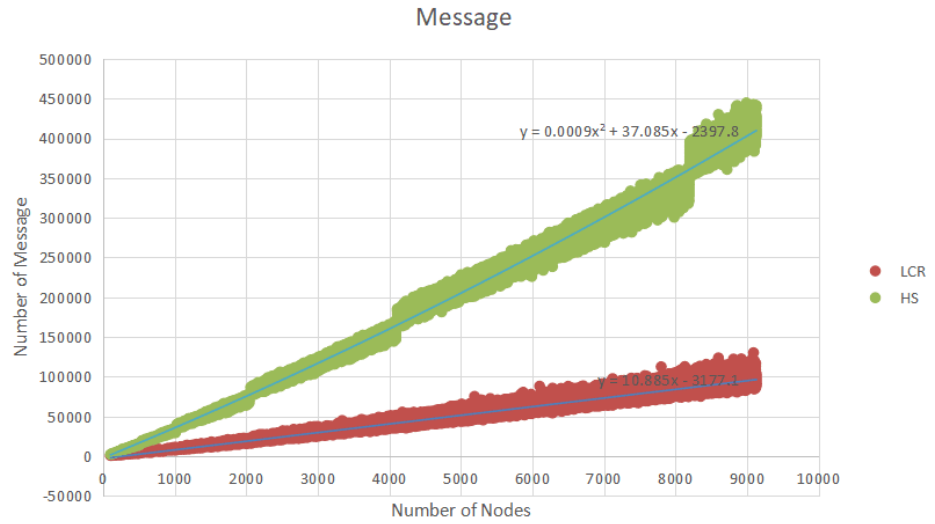


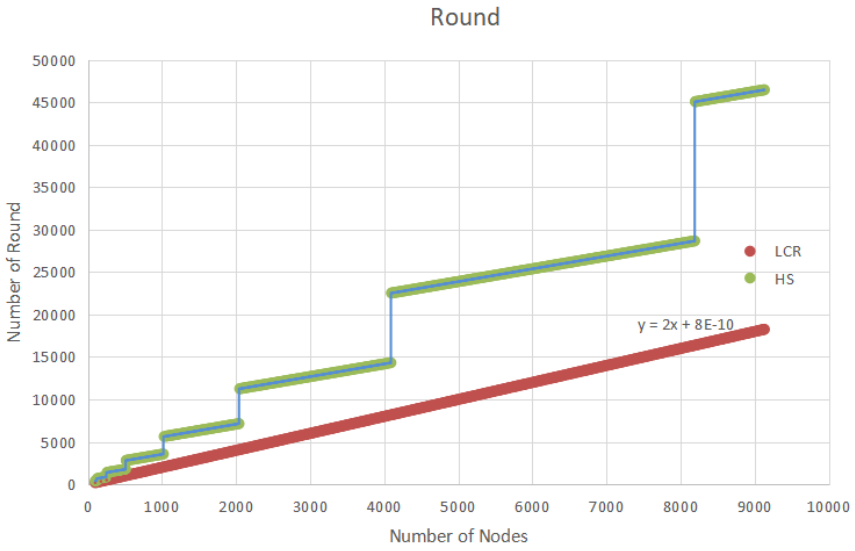
FIGURE 5. DECREASING CLOCKWISE ORDER ROUND

3.3 Random Order

The random order message passing and round result is shown in the Figure 6 and Figure 7. Although the data is fluctuating, it can be seen that the fluctuation of data is in a range, and since HS returns to the same sender twice, the change of phase in exponential level can also be

clearly shown in the Figure 6. For this kind of random situation, LCR is basically less than HS in both the number of information transfer and round.





4 Conclusion

In conclusion, the performance of LCR is basically better than HS algorithm, but in extreme conditions, that is, only when the ID decreases in a clockwise direction, the performance of LCR in the number of information transfer is far lower than that of HS algorithm. Therefore, from a certain point of view, LCR algorithm is unstable, but in general, its performance is higher than that of HS algorithm, but the number of information transfer and round times of HS algorithm are relatively stable.