COMP212 CA 1 Report
Coordination and Leader Election Simulating and Evaluating Distributed Protocols in Java

Robert Szafarczyk – 201307211

1. LCR Algorithm

Termination stage

Once a node elects itself the leader, it generates an *LCRLeaderMsg*, in the form of *<leaderID>* and transmits it to its clockwise neighbour.

```
1:   upon receiving an LCRLeaderMsg <leaderID> from counter clockwise
     neighbour
2:   if leaderID = myID
3:     terminate
4:     end algorithm
5:   else
6:     store leaderID
7:     sendToClockwise := LCRLeaderMsg
8:     terminate
9:   endif
```

Correctness

The above approach ensures that all nodes terminate, know the id of the leader and the leader id is the highest in the ring. After each run of the algorithm, a method checks if the correct leader was selected, and keeps a counter for the error rate. This is the pseudo code.

```
1:   leaderId := anyNode.leaderID
2:   for n in nodes
3:     if n.leaderID != leaderID or n.getID > leaderID
4:       return false
5:     endif
6:   endfor
```

In my testing, LCR never elected a leader incorrectly. The testing results for applying this method can be found under *resource A*.

Performance

Time complexity

The number of rounds until termination will always be *2n*, no matter the order of ids. This is because the greatest ID of the leader has to travel through all nodes, arriving at the leader node in the *n-th* round and then the termination stage will take another *n* rounds. Thus, the time complexity will be *2n,* and is bounded by *O(n).* See *resource B.*

Communication complexity

The worst-case communication complexity for LCR is in a ring with strictly counter clockwise id assignment. This is because through *n-1* rounds there are *n-1* messages transmitted every round. Thus, the worst-case communication complexity of LCR is $O(n^2)$. See *resource B.2*. The best-case scenario is in a setting with clockwise ordered ids. In this scenario, only the leader token will be transmitted from round 2 to rounds *2n*. Therefore, the best-case communication complexity for LCR is *O(n).* See *resource B.1.* The average case is *O(n log(n))*. This can be seen in the case of random id assignment. See *resource B.3*.

2. HS Algorithm

Termination stage

Once a node elects itself the leader, it generates an HSLeaderMsg, in the form of *<leaderID, hopCount>* for both its clockwise and counter clockwise neighbours. The *hopCount* will be equal to *numberOfNodes / 2* rounded down. Thus, each node in the ring will receive only one *HSLeaderMsg* and terminate. This approach ensures that all nodes will eventually terminate and know the id of the elected leader.
A crucial observation for this approach is that the leader node can obtain the number of nodes by calculating:

$$numberOfNodes = \ 2^{phase} - hopCountFromRcvdMsg + 1$$

```
 1:   after electing itself the leader
 2:   sendClock, sendCounter := <myID, numberOfNodes/2>
 3:   terminate
 4:
 5:   upon receiving an HSLeaderMsg<leaderID, hopCount> from clockwise
      neighbour
 6:   store leaderID
 7:   if hopCount = 1
 8:     terminate
 9:   elif hopCount > 1
10:     sendCounter := <leaderID, hopCount - 1>
11:     terminate
12:   endif
13:
```

```
14:   upon receiving an HSLeaderMsg<leaderID, hopCount> from counter
      clockwise neighbour
15:   store leaderID
16:   if hopCount = 1
17:     terminate
18:   elif hopCount > 1
19:     sendClock := <leaderID, hopCount - 1>
20:     terminate
21:   endif
```

Correctness

To check if the HS algorithm has elected the correct leader, I've used the same validation method as for LCR. As for LCR, the HS algorithm never elected a leader incorrectly. Both algorithms were simulated thousands of times for different ID assignments and for different ring sizes and had a 0-error rate. See *resource A.*

Performance

Time complexity

The algorithm works in phases, each phase *k* requires $2 * 2^k$ rounds to complete. For a token traveling outwards, it takes $log_2(n)$ phases to reach its origin. This is because the hop count will be equal to $2^{log2(n)} = n$. After this, we only have to account for the termination stage, which in my implementation will take $\frac{n}{2}$ rounds. Thus, the total number of rounds is:

$$2 + 4 + 8 + \cdots + n + \frac{n}{2} = O(n)$$

HS's time complexity isn't affected by different id assignments. See all time complexity plots for HS in *resource B*. It's worth mentioning that whenever the ring size exceeds a power of 2, the HS complexity function (both time and communication) rises sharply, and then stabilizes until the next power of 2.
This means that LCR's performance (even for counter clockwise id order) is better for ring sizes up to 20.

Communication complexity

In phase 0 all nodes are transmitting messages. So, *4n* messages are transmitted. In phases > 0, call it phase *k*, not all nodes transmit messages. A node will only send a message if it's received 2 tokens inbound in phase *k – 1*. It's crucial to see that only one node in a sequence of $2^{k-1} + 1$ will generate a message in phase *k.* This will be the node with the highest id. So, in the whole ring at phase *k* there will be $\frac{n}{2^k+1}$ nodes generating a message. The total number of messages in phase *k* will be $4n * \frac{2}{2^k+1}$.
Since we know that the total number of phases is *O(log n),* the message complexity of HS is

$$4n * \frac{2}{2^k + 1} * \log(n) = O(n \log n)$$

The message complexity is also not affected by different id assignments. See all communication complexity plots for HS in *resource B*.

3. Resources

A. LCR and HS error rates for random, clockwise and counter clockwise id assignment.

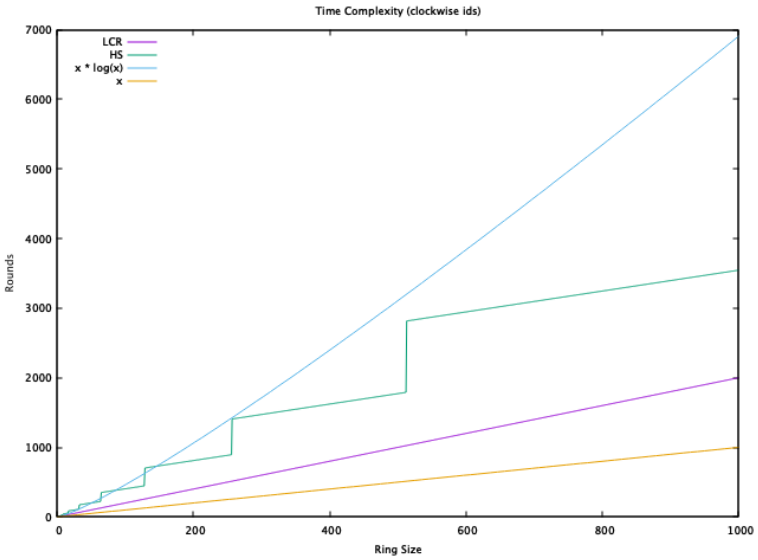
```
Roberts-MacBook-Pro:src robert$ java -cp JavaPlot.jar:. Simulation 1000 4 -r
 Progress 1000/1000  ||  LCR error count: 0  ||  HS error count: 0
Roberts-MacBook-Pro:src robert$
Roberts-MacBook-Pro:src robert$ java -cp JavaPlot.jar:. Simulation 1000 4 -a
 Progress 1000/1000  ||  LCR error count: 0  ||  HS error count: 0
Roberts-MacBook-Pro:src robert$
Roberts-MacBook-Pro:src robert$ java -cp JavaPlot.jar:. Simulation 1000 4 -d
 Progress 1000/1000  ||  LCR error count: 0  ||  HS error count: 0
 Roberts-MacBook-Pro:src robert$
```

B. Time and communication complexity plots

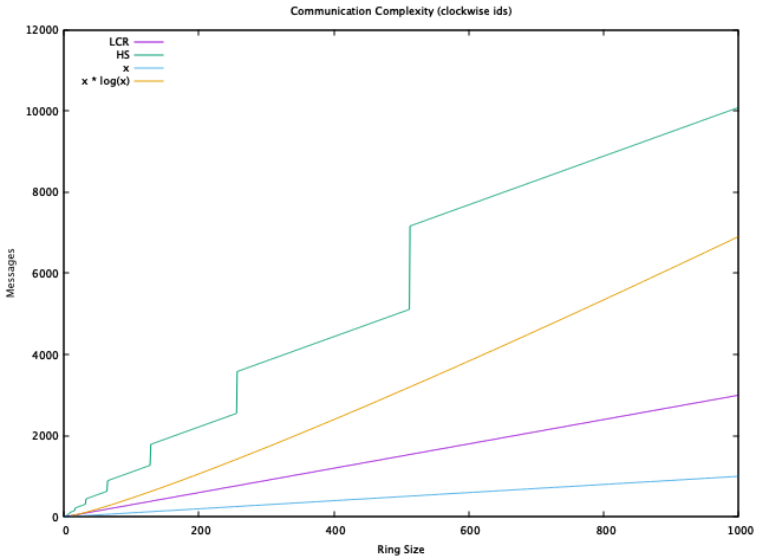
The HS complexity functions rise when the ring size reaches a power of 2. The time complexity functions are not affected by different id assignments. Only the communication complexity function for LCR varies depending on id assignment.

Time Complexity **Communication Complexity**

1. Clockwise id order



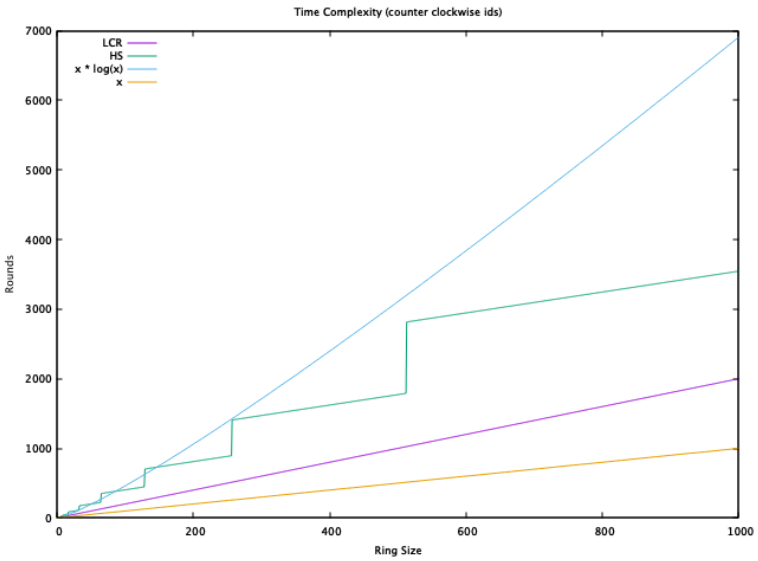
Both HS and LCR are *O(n).*



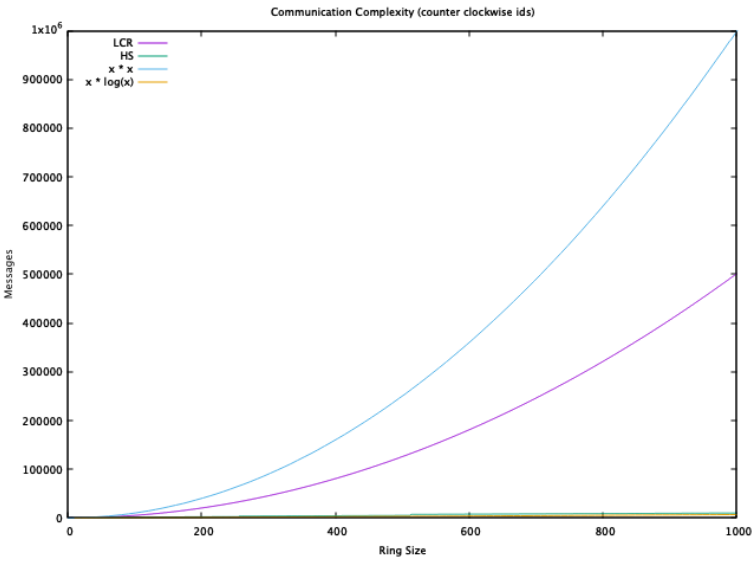
HS is *O(n logn).* LCR is *O(n)*

Time Complexity **Communication Complexity**

2. Counter clockwise id order

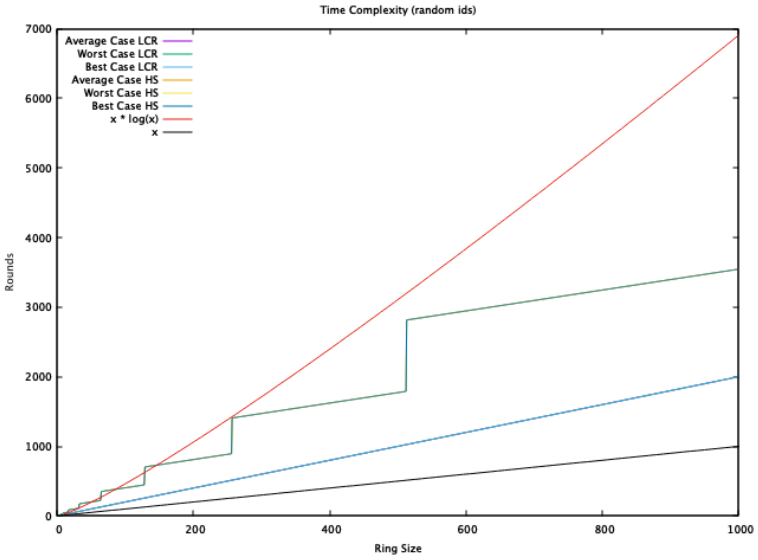


Both HS and LCR are *O(n).*

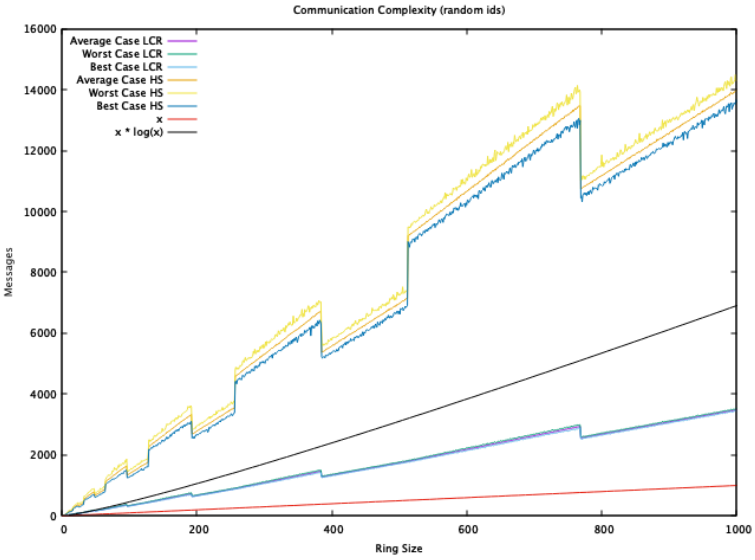


LCR is $O(n^2)$. HS is *O(n logn).*

3. Random id order



Both HS and LCR are *O(n).*



Both HS and LCR are *O(n logn)* in all best, worst and average cases.

All the above plots can be found in the /res directory.