Leader election in synchronised undirected ring

1 Introduction

We consider an undirected ring with n nodes. Each node is given a unique (integer) label (of which it alone is aware) wchich is the only factor differentiating it among other nodes and it can communicate only with its two neighbors on the ring in synchronous rounds. In such a configuration finding the node with minimal label is a fundamental problem, whose solution may be applied in more complex algorithms. The most straight-forward approach to solving this problem is making each node send its id to an arbitrary neighbor and instructing every node to pass on received messages to neighbor opposite to source of the message while maintaining a minimum among all ids which it has seen. Once node's id traveled the whole ring back to its origin the minimal id is known. This algorithm requires $O(n)$ rounds to terminate and sends $O(n^2)$ messages between nodes. Let's call this approach *All The Way*. Here we present a synchronous version of a probabilistic algorithm by Ephraim Korach, Doron Rotem, and Nicola Santoro [3] which on average costs approximately $.49n \log n$ message transfers, however the worst case is still $O(n^2)$ messages .

2 The algorithm

Let's improve the *All The Way* protocol by noticing that messages need not to further circle around the ring once they have encountered a node which has seen a bigger id. We can thus modify the *All The Way* protocol so that a node will only forward Election messages carrying an id smaller than the smallest seen so far by that entity. However, by doing so only the elected leader will be aware of algorithm's termination. We can easily fix this by making the leader send a *notify* message around the ring terminating all nodes, that strategy will incur an additional cost of $O(n)$ rounds and n messages. Let's call the algorithm with only this optimization *As Far As It Can*

To further improve performance of the algorithm, at initialization each node chooses with equal probability one of the two directions in which it will send its initial message, with messages going clockwise and counterclockwise on the ring, it is expected that many messages run into "smaller" nodes and hence are annihilated sooner, thus resulting in the smaller message complexity of the algorithm. The algorithm in every processor consists of three successive stages, as described below for a processor X_i.

Pseudo Code

1. Initialization

   $MAX_i := X_i$

   choose a direction d {left, right} with probability $\frac{1}{2}$

   send message $\langle X_i \rangle$ in direction d

2. Election

   Repeat following steps until the end of election is signaled by receipt of a ¡!¿ message. If two messages are encountered from left and right discard smaller message and proceed as if only the larger was received.

   Message $\langle X_j \rangle$ received

   if X_j>MAX_i then $MAX_i := X_j$ and pass $\langle X_i \rangle$ further on

   elif X_j==MAX_i then X_i won the election send $\langle ! \rangle$ message

3. Inauguration

   If a message $\langle ! \rangle$ is received by a leader then terminate otherwise pass on the message before terminating

3 Correctness

Processor X_i wins the election if and only if it has made a full round along the ring, which implies that X_i is the largest id (hence a sole winner) and all other processors must have their local X_j variables set to X_i. It follows that sending a $\langle ! \rangle$ message around the ring will allow all processors to terminate. We work under the assumption that all processes start simultaneously, otherwise the first message a processor receives wakes it up and it completes initialization step before processing the message.

4 Complexity analysis

Let's denote by $D(i)$ a sequence obtained by reading the elements of a permutation on $\{1 \dots n\}$ in the clockwise direction starting from i, and let $D^R(i)$ be defined similarly for anticlockwise reading. For an element j in D let $\triangle_D(i,j)$ be the distance from i to j measured in the clockwise direction. We call a processor r a c-eliminator of i (a-eliminator) in D, if there is a nonzero probability that message i will be discarded after a direct comparison with r or contact with a node which has its local MAX set to r where i is moving clockwise (anticlockwise).

for a permutation $\pi = \langle \sigma_1 \dots \sigma_n \rangle$ on $\{1 \dots n\}$ let's define

$$REC(\pi) := \{\sigma_i : 1 < i <= n \wedge \forall j < i, \quad \sigma_j < \sigma_i\}$$

Lemma 1. *Let $REC(D(i))$ be the sequence of records of $D(i)$ with r_1 as the first element. Element r is a c-eliminator of i if and only if*

$$r \in REC(D(i)) \quad \wedge \quad \left\lceil \frac{\triangle_D(i,r)}{2} \right\rceil < \triangle_D(i,r_1)$$

Proof. r obviously has to be bigger than all its predecessors to discard i, otherwise they will never meet. By synchronicity assumption distance from r to r_1 must not be greater than distance from i to r_1 otherwise i will be discarded by r_1 □

In fact we can easily derive probability that i will be discarded by a given c-eliminator. Given c-eliminators $r_1 \dots r_m$

$$P(i \text{ is eliminated by } r_t) = \begin{cases} \frac{1}{2} + (\frac{1}{2})^m & if \quad t = 1 \\ (\frac{1}{2})^t & if \quad t > 1 \end{cases}$$

the first case is the probability of r_1 going anticlockwise or all upper records going clockwise, the second case is the probability of r_t going anticlockwise and c-eliminators $r_1 \dots r_{t-1}$ moving clockwise.

Theorem 1. *Average number of messages is bounded by* $\frac{3}{4}n(H_n + \frac{2}{3} - \frac{1}{3n})$ *for* $n > 3$

Proof. For a given labeling of processors we can obtain two rings D and D^R. Let's consider processor i and its c-eliminators $r_1 \dots r_m$ and a-eliminators $s_1 \dots s_k$ in D. Note that c-eliminators in D are a-eliminators in D^R and vice-versa. For a given pair D and D^R, the total distance traveled by i under algorithm *As Far As It Can* is

$$\triangle_D(i, r1) + \triangle_{D^R}(i, s1)$$

Let's now use the earlier derived probabilities of c-eliminator discarding an element to analyze the probabilistic version of *As Far As It Can*. Let $E_i(D)$ be the expected distance traveled by i in ring D. The summands are multiplied by $\frac{1}{2}$ to count in the probability of choosing direction.

$$E_i(D) = \left(\sum_{j=1}^{m}(\frac{1}{2^{j+1}})\left\lceil\frac{\triangle_D(i,r_j)}{2}\right\rceil\right)+(\frac{1}{2^{m+1}})\triangle_D(i,r_1)+\left(\sum_{j=1}^{k}(\frac{1}{2^{j+1}})\left\lceil\frac{\triangle_{D^R}(i,s_j)}{2}\right\rceil\right)+(\frac{1}{2^{k+1}})\triangle_{D^R}(i,s_1)$$

By symmetry $E_i(D^R)$ is equal to $E_i(D)$. We can get the following upper bound on the sum of these two expected values by considering only r_1 and s_1

$$E_i(D) + E_i(D^R)) \leq \frac{1}{2}\left(\triangle_D(i,r_1) + \triangle_{D^R}(i,s_1) + \left\lceil\frac{\triangle_D(i,r_1)}{2}\right\rceil + \left\lceil\frac{\triangle_{D^R}(i,s_1)}{2}\right\rceil\right)$$

which can be simplified to

$$E_i(D) + E_i(D^R)) \leq \frac{3}{4}\left(\triangle_D(i,r_1) + \triangle_{D^R}(i,s_1)\right) + \frac{1}{2}$$

So the probabilistic algorithm for a given i travels three quarters of deterministic *As Far As It Can* and a constant $\frac{1}{2}$ (a quarter per permutation). This bound holds for all i except for n, which goes through n nodes in both algorithms. Average message complexity of *As Far As It Can* when ignoring n is $nH_n - n$ [4] hence the bound we calculated is

$$\frac{3}{4}(nH_n - n + \frac{n-1}{4}) + n$$

which simplifies to

$$\frac{3}{4}n(H_n + \frac{2}{3} - \frac{1}{3n})$$

□

The bound can be further tightened by considering in higher-order upper records. [2]

References

[1] H. L. Bodlaender, J. van Leeuwen in STACS 86, (Eds.: B. Monien, G. Vidal-Naquet), Springer Berlin Heidelberg, Berlin, Heidelberg, **1986**, pp. 119–129.

[2] C. Lavault, "Average number of messages for distributed leader-finding in rings of processors", *Information Processing Letters* **1989**, *30*, 167–176, `https://www.sciencedirect.com/science/article/pii/0020019089902081`.

[3] D. Rotem, E. Korach, N. Santoro, "Analysis of a distributed algorithm for extrema finding in a ring", *Journal of Parallel and Distributed Computing* **1987**, *4*, 575–591, https://www.sciencedirect.com/science/article/pii/0743731587900311.

[4] N. Santoro in Design and analysis of distributed algorithms, **2006**.