1 Introduction

Given a single-directional ring of n processors, it is required to program the processors such that they are able to find the size of the ring. When processors are distinguishable, it is sufficient to count distinct identifiers. Thus, we will consider indistinguishable processors. Itai and Rodeh (1990) [1] have proven that there exists no deterministic, always correct solution to this problem fit for any ring size - mostly due to it being impossible to recognise messages originating from the same processor. It is therefore necessary to consider probabilistic approach - an algorithm that always terminates with usually correct result. We shall consider processors with access to random bit generator. Itai and Rodeh (1990) have also proven that there exists no processor terminating (meaning all processors terminated) algorithm that solves this problem which is correct with probability $a > 0$. Thus, it is necessary to consider message terminating (meaning no processor will send a message without receiving one, and all messages have been received) algorithms. Itai and Rodeh (1990) also propose such an algorithm, which is correct with externally chosen probability $0 < a < 1$.

2 Algorithm

2.1 Overview

Throughout the algorithm, each processor maintains it's best estimation of the ring size (k_n). This estimation is initially 2 and is guaranteed to never exceed true ring size. In response to a received message, a processor may increase its estimation, or it may increase confidence in its estimation. The minimum confidence is the external parameter that decides probability of correctness. More specifically, let the parameter be r; in that case, we define confidence as $r \cdot k_n$ (so dependent on current ring size estimation). The algorithm terminates when all processors are confident in their value of k_n. At such a point, no processor initiates a message. It will be shown that when this happens, k_n is the same for all processors.

2.2 Description

A message m contains 3 fields: k_m, id_m, $count_m$. k_m is the ring size estimate of the processor originating the message. id_m is the randomly chosen identity, taken from random bit generator by the processor originating the message. $count_m$ is the number of processors the message has visited. When a message passes through processor n (without terminating) with $k_n < k_m$, the processor increases its $k_n = k_m$ and initiates testing for this value with confidence 0. When a message m terminates at processor n (meaning $k_m = count_m$ and processor has sent a message with k_m), there are several possibilities: if $k_m < k_n$, the message is forgotten - estimation in this message can not be correct; if $k_m > k_n$ the processor knows he was not author of the message, sets $k_n = k_m + 1$ and initiates testing for this value with confidence 0; if $k_m = k_n$ and processor has not sent a message or $id_m \neq id_n$ - processor also knows it wasn't author of the message, so proceeds as in previous case; finally, if $k_m = k_n$ and $id_m = id_n$ and processor has originated such a message - processor considers (possibly incorrectly) itself to be the author of the message, and thus increases confidence by 1. If the confidence is below the minimal confidence, it initiates another test. Finally, the test itself is defined as sending a message (k_n, random bit, 1).

2.3 Correctness and complexity

Lemma 1. *Throughout the algorithm*

1. $count_m \leq k_m$
2. $k_n \leq n$

Proof.
1. When a message reaches state in which $count_m = k_m$, it terminates, and is either forgotten, or initiates a test with new message that begins with $count_m = 1$.
2. $max_n(k_n)$ is initially 2, and possibly may be increased in two places: when $k_m > k_n$ or when $k_n = k_m = count_m$ but processor can't be the author. In both cases, there exist at least $k_m + 1$ processors, and that is the new k_n set by processor. Thus, $max_n(k_n) \leq n$.

□

The value of k_n is non-decreasing, and for each distinct k_n there are at most $r \cdot k_n$ messages initiated, and each message being sent at most k_n times. Thus, it follows that

Lemma 2. *The message complexity of the algorithm is* $O(r \cdot n^3)$.

Let f_n denote final value of k_n (held by processor n) - that is, the value held when algorithm terminates. We will show that the algorithm is consistent, that is it returns the same value for all processors

Lemma 3. *For all processors* u, w $f_u = f_w$.

Proof. [1] Suppose to the contrary, that there exist processors u and w for which $f_w < f_u$. The procedure originate, increases k_u to f_u. Thus, a message carrying f_u existed. This message could not be cancelled as a result of the arrival of another message. The message could disappear at a node v only if v has sent a message with the same value of f, exactly r times. Thus, some message carrying f_u succeeds passing through every node, in particular through w, increasing k_w. to f_u, thus $k_w \geq f_u$ □

Consider processors $v_0, v_1, ..., v_{n-1}$. Let f be the common value of f_i. Consider $f < n$ - that is, algorithm message terminated with incorrect result. v_0 sent r messages with value f and random bit id_0^j (where j means number of confirmation round. Those messages did not initiate a new increased k_n, as such k_n would be equal to $k_n = f + 1$. Let v_i be the processor who received these messages ($v_0 \neq v_i$). It follows that v_i also sent messages with the same id_i^j. Let R be a relation such that $v_a R v_b$ when there exists l such that $a = b + lf \ (mod\ n)$. Note that all classes generated by same l have the same number of elements. It follows from above that for all processors in each class of the relation, id^j is the same

Lemma 4. *For each class* C_i *of relation* R*, with* c *elements, for all confirmations rounds* j, $id_0^j = id_1^j = ... = id_{c-1}^j$.

Let g denote number of classes of relation R. Each class contains $h = n/g$ elements.

Lemma 5. *The probability of passing the test for* f *in each class is* $2^{-(h-1)r}$.

Proof. One processor chooses a specific bit, all the others must choose the same, each with probability of $1/2$. □

There are g classes in the relation. Because we are considering $f < n$, then $h \geq 2$. Since f was chosen, all of the classes must arrive at the erroneous result.

Lemma 6. *For any* n, r *the probability of error is not greater than* $2^{-nr/2}$

Proof. $2^{-(h-1)r^g} = 2^{-(gh-g)r} = 2^{-(n-g)r}$. Because $g = n/h$, the highest possible g is $g = n/2$. Thus, probability of error is not greater than $2^{-nr/2}$ □

Probability of returning an erroneous result depends on the arbitrarily chosen parameter r, which has no other implications on the algorithm (besides message complexity).

Finally, consider average complexity. Upper bound $O(r \cdot n^3)$ assumes each test that will fail, will fail at the last moment. In reality, expected value of retrials of test is a low constant

Lemma 7. *The expected number of tests* e_t *for each wrong value is* $e_t \leq 2$.

Proof. As was shown, each test has a chance of passing equal to $2^{-(n-g)}$. Thus, $e_t \leq \sum_{i=1}^{\infty} i \cdot (1 - 2^{-(n-g)}) 2^{-(n-g)(i-1)} \leq \sum_{i=1}^{\infty} i \cdot (1 - 2^{-2}) 2^{-2(i-1)} = 2$. □

For $k_n = n$, the test will take r rounds. We can conclude that

Lemma 8. *Expected message complexity of the algorithm is* $O(n^3 + r \cdot n^2)$.

References

[1] A. Itai, M. Rodeh, "Symmetry breaking in distributed networks", *Information and Computation* **1990**, *88*, 60–87, https://doi.org/10.1016/0890-5401(90)90004-2.