ANALYSIS OF A DISTRIBUTED ALGORITHM FOR EXTREMA FINDING IN A RING

by

D. Rotem(+), E. Korach(++) and N. Santoro(+++)

SCS-TR-61

August 1984

(+) University of Waterloo, Department of Computer Science, Waterloo, Ontario. N2L 3G1, Canada. Currently on sabbatical leave at Lawrence Berkeley Labs, Computer Science Research, Berkeley CA 94720.

(++) IBM Research Center, Technion, Haifa, Israel

(+++) School of Computer Science, Carleton University, Ottawa, Canada.

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada and in part by Applied Mathematics Sciences Research Subprogram of the Office of Energy Research, U.S., Department of Energy, under contract number DEAC03-763F00098.

ANALYSIS OF A DISTRIBUTED ALGORITHM FOR EXTREMA FINDING IN A RING

by

D.Rotem , E.Korach and N.Santoro

ABSTRACT

A new and more detailed analysis of the algorithm of Chang and Roberts for distributed extrema finding on a ring is presented. This analysis shows that this simple algorithm which is known to be average case optimal , compares very favorably with all the other known algorithms as it requires $O(n\log n)$ messages with probability tending to one. A bidirectional version of this algorithm is presented and shown to dominate the unidirectional one in its average message complexity. Finally , both the unidirectional and the bidirectional algorithms are generalized to perform k selection in the ring , i.e. , find the k largest labelled processors.

Categories and subject descriptors;C.2.4 [**Computer-Communication Networks**]: Distributed Systems- *Distributed applications;* D.4.1 [**Operating Systems**]:Process Management- *synchronisation;* F.2 [**Theory of Computation**]:Analysis of Algorithms and Problem Complexity

General Terms : Algorithms, Performance

Additional Key Words and phrases : Distributed Algorithms , Token rings , Average complexity , Probability , Chebychev's inequality

1. Introduction

In this paper we consider a system of n processors arranged in a circular configuration. Each processor is given a unique (integer) label (of which alone it is aware) and it can communicate only with its two neighbors on the ring. In such a configuration , finding distributively the processor with the maximum label , is an elemental problem whose solution is likely to be used in more complex algorithms. This problem was first discussed in [11] in connection with electing a new 'leader' in a token ring after the token is lost. The new leader (the processor with the maximum label) then initiates a new token. In general , whenever there is a change in the task of the processors as a result of some failure , an election algorithm may be used to designate a single processor as a temporary controller for the purpose of reorganization (see [5] for more information about distributed election algorithms). Another environment in which the algorithms described here may prove to be useful are parallel computer architectures in which the processors are connected in a ring. Such a system (also called a 'systolic loop') is currently under construction at the University of Waterloo (WATERLOOP/2) [15].

Different measures have been proposed in the literature to measure the efficiency of distributed algorithms. In this paper we consider two such measures. The first is the number of messages exchanged between the processors and the second is execution time. Since distributed systems may be asynchronous with unpredictable message delays along links , we cannot measure actual elapsed time. In order to get a relative measure , we count the number of time units required to complete the election process under the synchronicity assumption ,i.e. all processors start execution at time t_0

and a message sent at time t arrives at its destination (and gets processed) at time $t+1$, this is also called "ideal execution time " in [12]. This measure is realistic in distributed systems in which local clocks run at the same rate and are periodically synchronized with a bound on the communication delay along a link [10] or in a 'systolic loop' architecture. Recently, algorithms which gradually improved the number of messages sent in the worst case were presented in [1,2,4,6,9,14]. These algorithms can be classified as unidirectional or bidirectional according to whether messages can be sent only in one direction ,say clockwise, or both directions in the ring .

In this paper we are interested in the average number of messages sent by an election algorithm. Given a set of n distinct labels $L=(l_1,l_2,...,l_n)$, the average number of messages transmitted by algorithm A is defined as follows: for each circular permutation π of L , we form a ring of processors labelled by π and count the number of messages transmitted by algorithm A under the worst case communication delays. We average the message count over all circular permutations of L where each of the $(n-1)!$ permutations is considered equally likely. Thus the averaging is done over all possible labellings and not over communication delays. There does not seem to be any canonical distribution for the communication delays, and the message complexity of all the above mentioned algorithms is not sensitive to these delays. In a recent paper [13], the lower bound on the average message complexity of election algorithms on a ring was shown to be exactly nH_n ($H_n=\sum_{i=1}^{n}\frac{1}{i}$ is the n^{th} harmonic number which is very closely approximated for large n by $\log n$,the natural logarithm of n). This result proves that although the algorithm of Chang et al [1] ,which we call Algorithm-C, has a worst case message complexity of $O(n^2)$ messages ,it is average case

optimal among unidirectional algorithms since it meets the lower bound on the average number of messages . In section 5 , a stronger result is given which shows that Algorithm-C will use $O(n\log n)$ messages with probability tending to 1 . Furthermore , it is shown that the algorithm's performance can be improved by using bidirectionality in the communication links. To our knowledge this is the first case in which bidirectionality is shown to be beneficial in election algorithms on a ring . We propose and analyze a generalization of Algorithm-C which selects the k largest numbers in the ring in n time units. In addition to the theoretical interest in distributed selection , it may replace repeated applications of the election algorithm in cases where it is required to choose the next k leaders in advance. Again , a bidirectional version of the selection algorithm is shown to have an advantage over the unidirectional one in its average message complexity.

The paper is organized as follows. In section 2 , Algorithm-C is presented for reference purposes. In Section 3 , the algorithm is generalized to perform k selection and the generalized version, called Algorithm-C(k), is analyzed. In section 4 , the variance is used to bound the maximum deviation of the number of messages from the computed average.In Section 5 , a bidirectional version is presented and its average case message complexity is bounded and compared to Algorithm-C. Finally , some conclusions and open problems are mentioned in Section 6.

2. The Algorithm

For reference purposes we include here a description of Algorithm-C. In the description of this algorithm and in the rest of the paper , we simplify the notation by calling the processors and the messages that they originate by their respective rank among the n labels ,i.e. , by ' processor i' we mean

the processor which has the i^{th} smallest label.

Algorithm-C

Initialize: Each processor i,sends the message i to its clockwise neighbor.

The following rule is applied.

When processor i receives message j then;

a)if $j > i$ then i sends the message j to its clockwise neighbor,

b) if $j = i$ then i wins the election since it is the highest labelled processor

c) if $j < i$, the message j is discarded. ∎

The correctness of this algorithm follows immediately from the fact that only the highest message transmitted will return back to its originator. Under the synchronicity assumption , this will happen after n time units.

3. Finding the k largest numbers in the ring.

Algorithm-C can be generalized to perform an arbitrary selection in the ring, i.e. find all the k largest labels. The generalized algorithm which we call Algorithm-C(k) is implemented by adding to each message i a counter c_i which is set to zero by the message originator. Whenever the message $<i,c_i>$ arrives at a processor j where j is larger than i , the counter c_i is incremented by one and if $c_i=k$ the message is discarded. In all other cases the message is transmitted unchanged as in Algorithm-C. Clearly , a message will return back to its originator if and only if it contains one of the k largest labels as required.

Theorem 1

The expected number of messages, AC_k , transmitted under Algorithm-C(k) is

$$AC_k = kn(H_n - H_k + 1)$$

Proof

We consider the expected number of links traversed by message i before it is discarded. If $i>n-k$ it will always traverse the whole ring contributing n messages to the average , hence we consider only the messages i such that $i\leq n-k$. We can view the process of message i traversing the ring as if message i is drawing numbers randomly from a bag of $n-1$ numbers without replacement. There are $n-i$ numbers larger than i and $i-1$ numbers smaller than i in the bag. As soon as k numbers larger than i are drawn the process is stopped. The number of draws R_i required until k numbers larger than i are drawn (which is equal to the number of links traversed by message i before it is discarded) is a random variable which has a negative hypergeometric distribution . Its expected value is known to be

$$E(R_i) = \frac{kn}{n-i+1}$$

as given in [7 ,pg. 85]. Another way to view this result is to consider the set B_i of processors with labels greater or equal to i. There are $n-i+1$ processors in B_i which divide the set of n links in the ring into cells with an average of $\frac{n}{n+i-1}$ consecutive links per cell. The message i has to traverse k consecutive cells in the clockwise direction (starting from the location of processor i) before it is discarded . Thus the above expression for $E(R_i)$ is in fact the expected number of links in a randomly chosen sequence of k consecutive cells in a random ring. Summing $E(R_i)$ for all labels i smaller or equal to $n-k$ and adding kn to account for the distance travelled by the k largest messages gives the value of AC_k as stated in the theorem. ∎

4. The variance

The probability of having a large deviation of the number of message transmissions from the value of AC_k can be bounded by using Chebychev's inequality [3] . For a random variable R with expected value $E(R)$ and variance $V(R)$, this inequality states that for any constant t ,

$$prob(|R-E(R)| \geq t) \leq \frac{V(R)}{t^2} \quad .$$

We are interested in the random variable M where $M=\sum_{i=1}^{n} R_i$ which represents the total number of messages transmitted under Algorithm-C(k). In our case $E(M)=AC_k$, and since we are interested only in bounding the probability of M being larger than AC_k , we will use the following version of the inequality in which $t=\delta AC_k$ and the absolute value is omitted,

$$prob(M \geq (\delta+1)AC_k) \leq \frac{V(M)}{(\delta AC_k)^2} \quad .$$

In order to use this inequality we need to compute $V(M)$ which is the variance of a sum of random variables R_i. It is interesting to note that for an arbitrary pair i and j , the random variables R_i and R_j are in general not independent. For example in the case of $k=1$ and $n=6$ let us denote by A and B the events $R_4=4$ and $R_5=2$ respectively .The probability of the event $(A \cap B)$ is zero which is different from the product of the probabilities of A and B. To see this , we observe that the message 4 will travel a distance of 4 links only if processors 5 and 6 are adjacent to each other in the ring which implies that the message 5 may travel either a distance of 1 or 5. Hence the event $A \cap B$ is not realizable whereas each of the individual events A and B have nonzero probability. In general , it is not easy to compute the variance of a sum of non independent random variables as this involves computing covariances for each pair . Fortunately , as proved in the following two lem-

mas , the R_i's satisfy a weaker condition than independence as they are pairwise uncorrelated , i.e. , for each pair i and j , $E(R_iR_j)=E(R_i)E(R_j)$, and therefore all the covariances vanish and $V(M)$ is equal to the sum of the individual variances [3].

Lemma 1

Let us denote by I_x the event that i travels a distance of x links under Algorithm-C(k). For any pair i and j where $j>i$ and $1\leq x\leq i$,

$$E(R_j) = E(R_j / \ I_x \ ).$$

(The right hand term denotes the expected value of R_j conditioned on the event I_x.)

Proof

In words , this lemma states that the expected distance travelled by j averaged over rings in which i travels a distance of x, has the same value for any possible value of x (clearly x cannot exceed i). We will use the 'cell' argument that was used in Theorem 1. Let Π_x be the set of rings of n links in which the message i traverses x links under Algorithm-C(k). Given a ring $S\in\Pi_x$, the condition that i traverses x links in S , implies that the processor i and the $x-1$ processors immediately following it in the clockwise direction form an interval X which does not include any processor m where $m>i$. Hence the elements of B_j (the processors with labels larger or equal to j) partition S into $n-j+1$ cells , where the interval X is totally contained in one of the cells. The cell which contains X will be called a 'big cell', whereas each of the other $n-j$ cells will be called a 'small cell'. In a random ring of Π_x , the expected size of a 'small cell' is $s=\frac{n-x}{n-j+1}$ whereas the

expected size of a 'big cell' is $s+x$. The message j has to travel k consecutive cells , hence it will travel either k 'small cell's or $k-1$ 'small cell's and one 'big cell'. The probability p of the latter event satisfies $p=\frac{k}{n-j+1}$. Hence the expected distance travelled by message j averaged over all rings in Π_x is

$$(1-p)ks+p(ks+x)=ks+\frac{kx}{n-j+1}=\frac{kn}{n-j+1}.$$

Using the expression for $E(R_{j)}$ in Theorem 1 gives

$$E(R_j / I_x) = E(R_{j)}.$$

Lemma 2

For any pair i and j , $E(R_iR_j) = E(R_i)E(R_j)$.

Proof

Let us denote by $d(i,\pi)$ the distance travelled by message i in ring π under Algorithm-C(k). Then by definition

$$E(R_iR_j)=\frac{1}{(n-1)!}\sum_{\pi\in\Pi} d(i,\pi)d(j,\pi).$$

where Π is the set of all $(n-1)!$ rings. This can be written as a double sum

$$\frac{1}{(n-1)!}\sum_{x=1}^{i} x \sum_{\pi\in\Pi_x} d(j,\pi)$$

where Π_x denotes the set of rings in which i travels a distance of x. The internal sum above is equal by definition to

$$E(R_j / I_x)|\Pi_x|,$$

where $|\Pi_x|$ is the cardinality of the set Π_x. By using the previous lemma and noting that

$$prob(I_x)=\frac{|\Pi_x|}{(n-1)!}$$

we derive

$$E(R_i R_j)=E(R_j)\sum_{x=1}^{i} x\frac{|\Pi_x|}{(n-1)!}=E(R_j)E(R_i).$$

In the next theorem , it is shown that Algorithm-C(k) will require $O(nlogn)$ messages with very high probability which depends on n and k.

Theorem 2

For $n\to\infty$ and fixed k and δ , the total number of messages M transmitted under Algorithm-C(k) satisfies,

$$prob\,(M\leq(\delta+1)AC_k) \geq P= 1-O((logn)^{-2}).$$

Proof

As we mentioned before , for $1\leq i\leq n-k$ the R_i 's have a negative hypergeometric distribution with variance

$$V(R_i)=kn\frac{(i-1)(n-i+1-k)}{(n-i+1)^2(n-i+2)}$$

as given in [7,pg 85]. (For $i>n-k$,$R_i=n$ with probability 1 and $V(R_i)=0$).
From the above lemma it follows that

$$V(M)=\sum_{i=1}^{n} V(R_i)=kn\sum_{i=1}^{n-k}\frac{(i-1)(n-i+1-k)}{(n-i+1)^2(n-i+2)}$$

This can be bounded by

$$kn\sum_{i=1}^{n-k}\frac{(i-1)}{(n-(i-1))^2}=kn\sum_{k+1}^{n}\frac{(n-i)}{i^2}.$$

By using the fact that $\sum_{i=1}^{\infty}\frac{1}{i^2}=\frac{\pi^2}{6}$, we conclude that

$$V(M)\leq\frac{\pi^2}{6}kn^2.$$

Substituting the value of AC_k as given in Theorem 1 into Chebychev's inequality we find that ,

$$prob\,(M \geq (\delta+1)AC_k) \leq \frac{\frac{\pi^2}{6}}{k\,\delta^2(H_n - H_k + 1)^2}.$$

from which the theorem follows.

As an example of the significance of the above results , we note that the probability that Algorithm-C will outperform the best known algorithm [2] (which has a worst case of $1.356n\log_2 n = 1.956n\log n$ messages) is more than 0.98 in rings of 10 processors and 0.99 in rings of 500 processors .

5. A bidirectional algorithm.

Next , we present a bidirectional version of Algorithm-C which will be shown to dominate the average message complexity of Algorithm-C still performing in n time units . In this algorithm , no global sense of orientation is needed ,i.e. , clockwise can mean different directions to different processors [9].

Algorithm-P

Each processor i stores the largest number it has seen so far in the local variable MAX_i. The following algorithm is performed at each processor i.

Initialize: Set $i \rightarrow MAX_i$.

Choose randomly a direction $d(i) \in$ {clockwise,anticlockwise} where prob($d(i)$ = clockwise)=$\frac{1}{2}$, and then send the message i in the direction $d(i)$. Apply the following rules.

Rule 1: If processor i receives a message j from one of its neighbors then if $j > MAX_i$ the message j is sent to its other neighbor , if $j < MAX_i$ the message is discarded , and if $j = MAX_i$ the processor i wins the

election.

Rule 2: If processor i receives two messages, say k and j , from both its neighbors at the same time, the smaller message is ignored and Rule 1 is applied. ∎

The correctness and finiteness of this algorithm follow from the fact that once a direction $d(i)$ is chosen for message i, this message will continue to traverse the ring in this direction. A message will return back to its originator if and only if it is the largest number in the ring. Clearly , the worst and best case message complexity as well as running time is exactly the same as in Algorithm-C. However , it can be seen intuitively that Algorithm-P has an advantage over Algorithm-C since there is a positive probability that a message will be discarded earlier if it meets a larger message travelling in the opposite direction on its way. In order to quantify this observation we need some information about the meeting point of such messages. In the following analysis we will use the synchronicity assumption which implies that two messages i and j travelling in opposite directions meet approximately half way between processors i and j. It will be clear from the proof of the next theorem that Algorithm-P dominates Algorithm-C in its average message complexity even without this assumption, however , the amount of saving cannot be determined in that case.

The following definitions are useful in the analysis of the average message complexity of Algorithm-P. Let $\pi = <\sigma_1,\sigma_2, \ldots, \sigma_n>$ be a permutation of {1,2,...,n}. The ordered set of records of π, $REC(\pi)$,is defined as

$$REC(\pi) = <\sigma_i \in \pi |\ i>1 \text{ and } \textit{for all } j<i,\ \sigma_j<\sigma_i>$$

For example if π= <3,2,4,1,6,5,8,7> then $REC(\pi)$= <4,6,8>. The set $REC(\pi)\cup\sigma_1$ is also called *left to right maxima* in [8]. Let D be a permutation

of {1,2,...,n} written on a circle. We denote by $D(i)$ the permutation obtained by reading the elements of D in the clockwise direction starting with element i , $D^R(i)$ is defined similarly where the reading is performed anticlockwise (see Figure 1). For an element j in D we denote by $\Delta_D(i,j)$ the distance from i to j measured in the clockwise direction, i.e. , $\Delta_D(i,j)$ is one less than the position of j in $D(i)$. A processor r where $r>i$ is called a *c −eliminator* of i (*a −eliminator*) in D ,if there is a nonzero probability that i will be eliminated from the election process after it is compared with r where $d(i)$=clockwise (anticlockwise) . This can happen if i and r arrive at the same time at some processor l , or in the case that r is the current value of MAX_l when i arrives at l. The following lemma characterizes the set of *c −eliminators* of i in D.

Lemma 3

Let $REC(D(i))$ be the sequence of records of $D(i)$ with r_1 being the first record. A processor j is a *c −eliminator* of i if and only if:

$$(a)\, r \in REC(D(i)) \;\; and (b) \;\; \left\lceil \frac{\Delta_D(i,r)}{2} \right\rceil \leqslant \Delta_D(i,r_1).$$

Proof

In this case i is moving clockwise in the ring. If $i=n$ the set $REC(D(i))$ is empty and i is not eliminated so the lemma holds trivially. Also it is clear that r_1 is a *c −eliminator* of i since there is a nonzero probability that i will be eliminated at r_1 . For example this event will happen if all processors in $REC(D(i))$ choose their direction as clockwise . A processor r $(r>i)$ with $d(r)$ =anticlockwise may eliminate i if r arrives at some processor l where $\Delta_D(i,l) \leq \Delta_D(i,r_1)$ before or at the same time i arrives there. Two necessary

conditions must be satisfied for this event to happen (a) r must be in $REC(D(i))$ otherwise r itself will be eliminated before it reaches processor l and (b) by the synchronicity assumption , the distance from r to r_1 must not be greater than the distance from i to r_1 otherwise i will reach r_1 before r . These two conditions prove the lemma. In fact we can state the exact probability with which i is eliminated by each of its $c-eliminators$ as follows. Let $r_1, r_2, .., r_m$ be the sequence of $c-eliminators$ of i.

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

This follows since given that i moves clockwise, for $t>1$, r_t will eliminate i if and only if it moves anticlockwise while all the $c-eliminators$ $r_1, r_2, .., r_{t-1}$ move clockwise. The case $t=1$ is an exception since i may be eliminated by r_1 also in the case that all $c-eliminators$ (including r_1) move clockwise. This introduces an additional probability of $(\frac{1}{2})^m$ for this case . ∎

An analogous lemma for the case where i moves anticlockwise is stated below.

Lemma 4

Let $REC(D^R(i))$ be the sequence of records of $D^R(i)$ with s_1 being the first record . A processor s is an $a-eliminator$ of i if and only if:

$$(a)\ s \in REC(D^R(i))\ \text{and}\ (b)\ \left\lceil \frac{\Delta_{D^R}(i,s)}{2} \right\rceil \leqslant \Delta_{D^R}(i,s_1).$$ ∎

We are now in a position to prove the following theorem which provides a conservative estimate on the saving in average message complexity introduced by Algorithm-P.

Theorem 3

The average number of messages transmitted during Algorithm-P is bounded by

$$\frac{3}{4}n(H_n+\frac{2}{3}-\frac{1}{3n}) \text{ for } n>3.$$

Proof

For a given labelling of the processors on the ring, we can obtain two circular permutations D and its reverse D^R by reading the labels clockwise and anticlockwise respectively.For a given processor i , let $r_1,r_2,\ldots,r_m$ and $s_1,s_2,\ldots,s_k$ be the sequence of its *c−eliminators* and *a−eliminators* in D respectively. Note that *c−eliminators* in D are *a−eliminators* in D^R and vice versa. We divide the set of circular permutations on n numbers into $\frac{(n-1)!}{2}$ pairs such that each pair consists of a permutation and its reverse. For a given pair,D and its reverse D^R, the total distance travelled by message i $(i<n)$ under Algorithm-C is

$$\Delta_D(i,r_1)+\Delta_{D^R}(i,s_1).$$

Turning now to Algorithm-P , we use the probabilities given in the above lemma to compute , $E_i(D)$, the expected distance travelled by i in D. The factor half multiplies each term to represent the fact that i will choose the clockwise and anticlockwise directions with equal probabilities.

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

By symmetry arguments , $E_i(D)$ is equal to $E_i(D^R)$. We can get an upper bound on the sum of the expected distances by taking into account only the

eliminators r_1 and s_1 since all other eliminators will only further reduce the expected distance. The bound is

$$E_i(D) + E_i(D^r) \leq \frac{1}{2}\left[\Delta_D(i,r_1)+\Delta_{D^R}(i,s_1)+\left\lceil\frac{\Delta_D(i,r_1)}{2}\right\rceil+\left\lceil\frac{\Delta_{D^R}(i,s_1)}{2}\right\rceil\right]$$

This can be simplified to give

$$E_i(D) + E_i(D^R) \leq \frac{3}{4}\left[\Delta_D(i,r_1) + \Delta_{D^R}(i,s_1)\right] + \frac{1}{2}.$$

As this analysis shows , in a given pair D and D^R , the expected distance that message i will travel under Algorithm-P , is three quarters of the distance it will travel under Algorithm-C plus an additional constant of half (or one quarter per permutation) .This holds for every message except n (which travels a distance of n under both algorithms) and for every pair of permutations. The average message complexity of Algorithm-C when message n is ignored is $nH_n - n$, hence the average message complexity of Algorithm-P is bounded by

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

The second term adds one quarter for each of the $n-1$ messages smaller than n and the last term is the distance travelled by message n. Finally this simplifies to

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

We note that the only place where the synchronicity assumption was used is in part (b) of lemmas 3 and 4. Without this assumption , the expected distance travelled by any message i under Algorithm-P will be smaller or equal to its expected distance under Algorithm-C , but the amount of saving may not be determined.

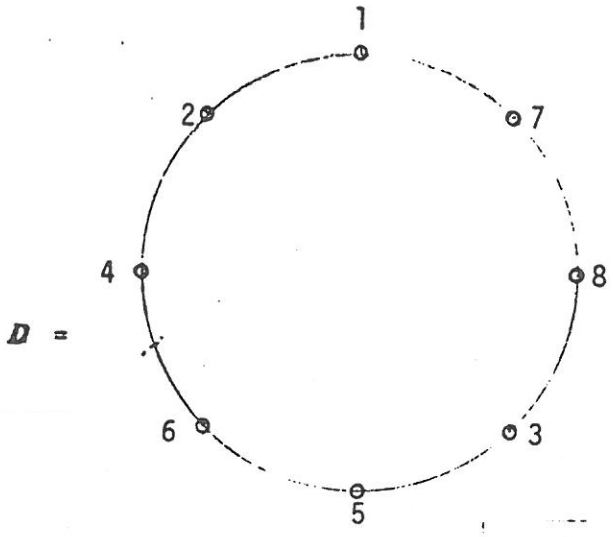
6. Conclusions

We have presented a more detailed analysis which shows that Chang's algorithm for distributed election on a ring requires $O(n\log n)$ messages with probability tending to 1. The constant implied by the 'O' notation is very small so that even for small rings the algorithm compares very favorably with the best known ones and should be considered in many cases to be the unidirectional algorithm of choice because of its simplicity and minimum tme requirements.

A bidirectional version of this algorithm was shown to have a better average message complexity still requiring the same amount of time. The bidirectional algorithm also does not require a global sense of orientation in the ring ,i.e. , clockwise does not have to be the same for all processors . Both algorithms were generalized to perform k selection in the ring for arbitrary k . The implementation of the unidirectional Algorithm-C(k) is simpler in this case and the advantage of the bidirectional algorithm is reduced with larger k. It is an interesting open question to find an exact expression for the average message complexity and the variance of Algorithm-P .Simulation results showed a good agreement with the bound given in Theorem 3 which indicate that most of the saving comes from the first *eliminators*. In general , the analysis of bidirectional algorithms seems harder than unidirectional ones and no lower bound on the average behaviour of these algorithms is known .

Acknowledgement

The authors thank Prof. N.Lynch for helpful suggestions on an earlier draft of this paper.



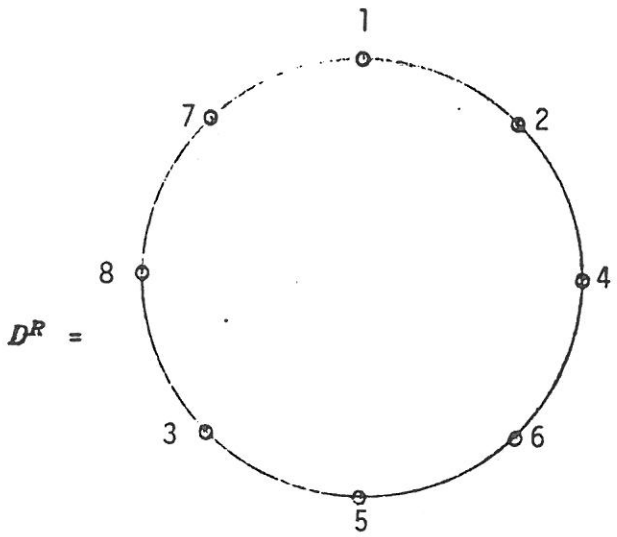


Figure 1

$D(1)$= <1,7,8,3,5,6,4,2>

$D^R(3)$ = <3,8,7,1,2,4,6,5>

$REC(D(3))$=<5,6,7,8> ;

The *c−eliminator*s of 2 in D are 7, 8 and 3 these are also the *a−eliminators* of 2 in D^R.

<7,8> ;

REFERENCES

[1] E.J. Chang and R. Roberts, "An improved algorithm for decentralized extrema-finding in circular configurations of processes", *Commun. Ass. Comput. Mach.*, vol. 22, pp 281-283(1979).

[2] D. Dolev, M. Klawe and M. Rodeh, "An $O(n \log n)$ unidirectional algorithm for extrema-finding in a circle", *J. Algorithms*, vol. 3, pp 245-260(1982).

[3] W.Feller, *Probability theory and its applications*, 2nd ed(1957) .,New York: John Wiley and Sons.

[4] W.R. Franklin, "On an improved agorithm for decentralized extrema finding in circular configurations of processors", *Commun. Ass. Comput. Mach.*, vol. 25, pp 336-337(1982).

[5] H. Garcia-Molina, "Elections in distributed computing systems", *IEEE Transactions on Computers*, Vol c-31,pp.48-59 (1982).

[6] D.S. Hirschberg and J.B. Sinclair, "Decentralized extrema finding in circular configurations of processes", *Commun. Ass. Comput. Mach.*, vol. 23, pp.627-628(1980).

[7] N.L. Johnson and S.Kotz , *Urn Models and their application*, 1977,John Wiley and Sons , New York.

[8] D.Knuth, *The art of computer programming*, Vol 3,Addison-Wesley,Reading,Mass 1973.

[9] E. Korach ,D. Rotem and N.Santoro., "Distributed election in a circle without a global sense of orientation" to appear in *Int. Journal of Computer Mathematics.*

[10] L.Lamport, "Using time instead of timeout for fault tolerant distributed systems", *ACM Transactions on programming Languages and Systems*,

Vol 6,No 2,April 1984,pp 254-280.

[11] G. LeLann, "Distributed systems - toward a formal approach", in *Proc. IFIP*, 1977,pp.155-160 , North Holland Amsterdam.

[12] T.A.Matsushita,"Distributed algorithms for selection",M.Sc Thesis, Univ. of Illinois,Urbana Champaign , July 1983.

[13] J. Pachl, E. Korach and D.Rotem, "Lower bounds for maximum-finding algorithms", to appear in JACM.

[14] G.L. Peterson, "An $O(n \log n)$ unidirectional algorithm for the circular extrema problem", *Trans. Progr. Lang. Syst.*, vol. 4, No 4,pp.758-762(1982).

[15] R.A. Whiteside, N.S. Ostlund and P.G.Hibbard, "A parallel Jacobi diagonalization algorithm for a loop multiple processor system" to appear in *IEEE Trans on Computers*.