A lower bound for probabilistic distributed algorithms

Jan K. Pachl

Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

Research Report CS-85-25

August 1985

A lower bound for probabilistic distributed algorithms

Jan K. Pachl

Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

ABSTRACT

The main contribution of the paper is a negative result about probabilistic algorithms: The probabilistic distributed algorithms to find the maximum label in every asynchronous unidirectional ring configuration are not more efficient than deterministic (non-probabilistic) algorithms.

1. Introduction.

In [3] we proved an exact lower bound for a certain class of distributed algorithms in unidirectional rings of processes. The purpose of the present paper is to prove the same bound for *probabilistic* distributed algorithms.

A *unidirectional ring* consists of processes connected by communication channels to form a circular configuration in which each process can send messages to its immediate neighbor in one direction; the direction is the same for all processes. The processes can communicate only by sending and receiving messages. Every message transmitted to a channel is eventually delivered, but the transmission delays are variable and not a priori bounded; thus the system is *asynchronous*.

Every process in the ring has a unique integer label, of which only the process itself is initially aware; the processes know neither the range of all labels in the ring nor the ring size. This paper deals with the distributed algorithms that find the maximum label. The algorithms are *probabilistic*. This means that the processes can test values of random variables (of known distribution). To prevent implicit communication between processes, it is assumed that the random variables used by different processes are independent.

The performance measure used to evaluate algorithms is the average expected number of messages transmitted when all processes begin execution simultaneously.

Research supported by the Natural Sciences and Engineering Research Council of Canada.

The main result of the paper is Theorem 5.3, in which a lower bound from [3] is extended to probabilistic algorithms. Theorem 6.3, which contains a matching upper bound, shows that the lower bound in Theorem 5.3 is exact.

2. Technical assumptions.

As in [3], it is assumed that (i) the communication channels are first-in first-out (FIFO) queues, (ii) the distributed algorithms under consideration are message-driven, and (iii) the execution in each process is sequential and is fully determined by the received messages and by the values of the random variables tested by the process. These assumptions are important in the *proof* of Lemma 2.1 below. However, the main *results* of this paper do not depend on the assumptions (i), (ii) and (iii).

The assumption that messages in each channel are delivered in the same order as sent (i.e. the channels are FIFO queues) does not change the number of messages necessary or sufficient to find the maximum label. Indeed, one can include a sequence number in each message and simulate the in-order delivery within the receiving process.

Processes send and receive messages. The execution of a process is message-driven if the process concludes all its activity arising from a received message before receiving the next message. In more concrete terms, a distributed algorithm is *message-driven* if processes receive messages by means of the *blocking-receive* operation ([4], p. 481), and have no other way of examining their incoming channels. Blocking receive means "if the next message has not arrived yet, suspend execution and wait until a message arrives". A process executing a message-driven algorithm can examine its incoming channel only by executing the blocking-receive operation (there is no "Is there a message waiting?" operation); the process can check whether the next message has arrived only if it commits itself to wait until the message arrives. It will be proved in section 4 that this restriction (assumption (ii)) does not limit the generality of Theorem 5.3.

If the processes in a (unidirectional) ring are labeled $s_1, s_2, \ldots, s_n$, with the communication channels leading from s_n to s_1, from s_1 to s_2, ..., and from s_{n-1} to s_n, then the ring is said to be *labeled by the sequence* $(s_1\ s_2\ \ldots\ s_n)$; the ring is also said to be *labeled by the set* $\{s_1, s_2, \ldots, s_n\}$.

The *trace* of a message is defined as in [3]: If the message sender has the label s_1 and has previously received no message then the trace of the message is the sequence (s_1). If the last message previously received by the sender has trace $(s_1\ s_2\ \ldots\ s_{k-1})$, $k \geq 2$, and the sender has the label s_k, then the trace is $(s_1\ s_2\ \ldots\ s_k)$. The trace is an upper bound on the information about process labels carried by the message: The message whose trace is s contains no information about the labels outside of the segment labeled by s.

2.1. Lemma. For a given algorithm executing in a given unidirectional ring, if all processes start their execution simultaneously then the probability of sending a given message does not depend on transmission delays. Moreover, if t is a prefix of s and if the same algorithm is executed in the ring labeled by t and in the ring labeled by s, in both cases with all processes starting their execution simultaneously, then the probability of sending a message with trace t in one ring is the same as in the other.

Proof depends on the assumptions formulated at the beginning of this section. Let all processes begin execution simultaneously. Every process receives messages from a single source and, by the assumption (i), the order of received messages does not depend on transmission delays. Moreover, in view of the assumption (ii), the absolute values of transmission delays cannot be observed by the processes, and hence have no effect on the execution sequence in each process and on the content of the transmitted messages. By the assumption (iii), the only non-determinism is introduced to the execution by the random variables tested by the processes. Hence (for fixed values of the random variables tested by the processes that correspond to the labels in t) the execution that leads to the transmission of the message with trace t in one ring takes place in the other ring as well.

□

3. The performance measure.

Let I be a set of n labels, and consider a (probabilistic) distributed algorithm A. The average expected number of messages transmitted by A in the rings labeled by I, denoted $\overline{ave}_A(I)$, is defined as follows. For every permutation s of I, let $\nu_A(s)$ be the number of messages transmitted when all processes in the ring labeled by s begin execution simultaneously and execute according to the algorithm A. Since the algorithm A is probabilistic, $\nu_A(s)$ is a random quantity; let $\overline{\nu}_A(s)$ be its expected value. By Lemma 2.1, $\overline{\nu}_A(s)$ does not depend on transmission delays. Define $\overline{ave}_A(I)$ to be the average of $\overline{\nu}_A(s)$ over all permutations s of I.

Let p be a real number in the closed interval [0,1]. A *p-algorithm* is a probabilistic distributed algorithm that finds, with probability at least p, the maximum label in every unidirectional ring. More precisely, in any unidirectional ring of any size, with probability at least p at least one process in the ring eventually knows what the value of the maximum label is.

For the results in this paper, the questions of effective computability are irrelevant. The term *algorithm* (or *p-algorithm*) does not imply any effective procedure. A probabilistic algorithm executed by a process is simply a strategy to make the decision, based on the received messages and on the values of the random variables tested by the process, whether a message is to be transmitted at a given time and what its content is to be. The decision need not be effectively computable. This usage does not limit the validity of the forthcoming results; in fact, it can only make the lower

bounds more general.

4. The assumption that algorithms are message-driven.

The purpose of this section is to show that, although the *proof* of the lower bound in section 5 assumes that algorithms are message-driven, the bound itself holds true without the assumption. The rest of the paper may be read independently of this section.

A process is said to be in an *ambiguous state* when it can both receive a message if one has arrived and continue computation if no message has arrived. For example, a process is in an ambiguous state when it executes the conditional receive operation (which means "receive the next message if there is one, otherwise continue execution"). According to the definition in section 2, a distributed algorithm is message-driven if and only if none of its processes is ever in an ambiguous state.

The execution of a distributed algorithm that is not message-driven may depend on transmission delays. In that case the definitions in section 3 are extended as follows: A distributed algorithm A is a p-algorithm if it works correctly (i.e. with probability at least p at least one process in the ring eventually knows the value of the maximum label) *for any choice of delays.* Since there is no canonical probability distribution on the delays, the quantity $\nu_A(s)$ is taken to mean the number of messages for the *worst choice* of delays. More precisely, $\nu_A(s)$ is the least upper bound, over all possible combinations of transmission delays, of the number of messages transmitted when all processes begin execution simultaneously. The quantities $\overline{\nu}_A(s)$ and $\overline{ave}_A(I)$ are defined as in section 3, in terms of $\nu_A(s)$.

Consider a fixed process executing a p-algorithm. In the next lemma and in the proof of 4.2, $pr\_active(t)$ is the probability that if the process receives no message at or after the time t, it will eventually (at a time $t' \geq t$) either compute the value of the maximum label (and stop) or send a message. Denote $exp\_msg(t)$ the expected number of messages that the process will send at or after the time t if the process receives no message at or after t.

4.1. Lemma. Let t_0 be a time in the process execution such that $exp\_msg(t_0) < \infty$. For every $\beta > 0$ there exists $t_1 > t_0$ such that if the process receives no message in the time interval $[t_0, t_1]$ then $pr\_active(t_1) < \beta$.

Proof. For $t_0 \leq a < b$, denote $p(a,b)$ the probability that the process computes the value of the maximum label (and stops) or sends a message at or after the time a but before the time b, under the assumption that the process receives no message at or after the time t_0. The process computes the maximum label at most once, and therefore

$$\sum_{i=0}^{\infty} p(t_0+i,t_0+i+1) \leq 1+exp\_msg(t_0) < \infty \ .$$

Hence there is an integer $k>0$ such that

$$pr\_active(t_0+k) \leq \sum_{i=k}^{\infty} p(t_0+i,t_0+i+1) < \beta \ .$$

□

4.2. Proposition. For every p-algorithm A (not necessarily message-driven) and for every $\alpha>0$, $\alpha<p$, there exists a message-driven $(p-\alpha)$-algorithm B such that

$$\overline{ave}_A(I) \ \geq \ \overline{ave}_B(I)$$

for every finite set I of integers.

Proof. It is sufficient to prove that there is a message-driven $(p-\alpha)$-algorithm B such that $\overline{\nu}_A(s) \geq \overline{\nu}_B(s)$ for every $s \in D$.

The algorithm B will be constructed to mimic A, except for ambiguous states in the execution of A. Whenever a process executing A is in an ambiguous state, the corresponding process executing B either executes the blocking-receive operation (and defers any computation until after it has received the next message, if any) or continues computation (and defers receiving any message), but not both. The choice is made as follows.

Assume that a process whose label is s_i executes the algorithm A. Let z be the absolute value of s_i. Let the process be in an ambiguous state at a time t, and let y be the number of messages received by the process before the time t. If $exp\_msg(t)=\infty$ then the process executing B stops simulating A and initiates the execution of any maximum-finding algorithm for which the average expected number of messages is finite (for example the algorithm of Chang and Roberts [1], which is a 1-algorithm in the terminology of the present paper). Since $exp\_msg(t)=\infty$ for the algorithm A, transmission delays in the execution of A can be chosen so that $\overline{\nu}_A(s) \geq \overline{\nu}_B(s)$. Hence in the rest of the proof it will be assumed that $exp\_msg(t)<\infty$.

If $pr\_active(t) \geq \alpha/2^{z+y+3}$, then the algorithm B specifies that the process should continue its computation and not attempt to receive any messages. If $pr\_active(t)<\alpha/2^{z+y+3}$, then the process executes the blocking-receive operation and defers any remaining computation until after the next message has been received.

From Lemma 4.1 it follows that transmission delays can be arranged so that a process executing A receives a message (in an ambiguous state) only if the corresponding process executing B receives a message (using the blocking-receive operation). In the rest of this proof it is assumed that the transmission delays are chosen in that manner.

In a ring of processes labeled by a sequence s, let **E** be this random event: An ambiguous state in a process executing the algorithm A has been replaced by a blocking-receive operation (in the process executing B) and the process executing A sends a message or computes the value of the maximum label before the next message arrives. From the definition of the algorithm B it follows that the probability of **E** is smaller than

$$\sum_{s_i,y} \frac{\alpha}{2^{z+y+3}} < 2\alpha \sum_{n=0}^{\infty} \sum_{y=0}^{\infty} \frac{1}{2^{n+y+3}} = \alpha$$

where, in the first sum, z is the absolute value of the label s_i.

If **E** does not occur then A and B send exactly the same messages, and a process in the execution of A establishes the maximum label if and only if the same process in the execution of B does. Hence B is a $(p-\alpha)$-algorithm. Moreover, if **E** does occur then obviously B does not send more messages than A. It follows that $\nu_A(s) \geq \nu_B(s)$ and therefore $\overline{\nu}_A(s) \geq \overline{\nu}_B(s)$.

□

4.3. Corollary. Let I be a finite set of integers. If $b(p)$ is a continuous function of p, $0 \leq p \leq 1$, such that

$$b(p) \leq \overline{ave}_B(I)$$

for every $p \in [0,1]$ and for every message-driven p-algorithm B, then

$$b(p) \leq \overline{ave}_A(I)$$

for every $p \in [0,1]$ and for every (not necessarily message-driven) p-algorithm A.

□

5. From algorithms to functions on sequences: A lower bound.

Let Z be the set of integers. Define

$$D = \{ (s_1\ s_2\ \ldots\ s_k) \mid k \geq 1,\ s_i \in Z \text{ for } 1 \leq i \leq k, \text{ and } s_i \neq s_j \text{ for } i \neq j \}.$$

When $s = (s_1\ s_2\ \ldots\ s_k)$ is a sequence of integers, denote by $\mathrm{len}(s)$ the length of s, and by $C(s)$ the set of cyclic permutations of s.

Let p be a real number, $0 \leq p \leq 1$. Consider the following two properties of functions $e : D \longrightarrow [0,1]$.

P1. *Prefix property.* If t is a nonempty prefix of s then $e(t) \geq e(s)$.

P2(p). *Cyclic sum property.* If $s \in D$ then

$$\sum \{ e(t) \mid t \in C(s) \} \geq p\ .$$

Note that if a function e assumes only the values 0 and p then e has the properties **P1** and **P2**(p) if and only if the set $\{ s \in D \mid e(s)=p \}$ is exhaustive in the sense of [3].

For $s \in D$ and a function $e:D \longrightarrow [0,1]$, define

$$N(s,e) = \sum \{ e(t) \mid t \text{ is a prefix of some } r \in C(s) \} .$$

In the sequel, H_n is the nth harmonic number ([2], p. 73).

5.1. Proposition. If $e:D \longrightarrow [0,1]$ has the properties **P1** and **P2**(p) and I is a set of n integers then

$$\sum_s N(s,e) \geq n!\, n\, p\, H_n ,$$

where the sum is over all permutations s of I.

Proof is a generalization of the proof of 3.2 in [3]. Write

$$\sum_s N(s,e) = \sum_{k=1}^{n} \sum_s \sum \{ e(t) \mid t \text{ is a prefix of some } r \in C(s), \text{ and } \mathrm{len}(t)=k \} .$$

For a fixed k, $1 \leq k \leq n$, and a fixed permutation s of I, there are n prefixes t of cyclic permutations of s such that $\mathrm{len}(t)=k$. Hence for all permutations s there are $n!n$ instances of such prefixes t. In view of the cyclic sum property, one can form $n!n/k$ groups of these prefixes so that the sum of $e(t)$ over t in each group is at least p. It follows that

$$\sum_s N(s,e) \geq \sum_{k=1}^{n} \frac{n!\, n}{k}\, p = n!\, n\, p\, H_n .$$

□

Let A be a p-algorithm. Say that "with probability q the algorithm A sends a message with trace s" if the statement is true when all processes start execution simultaneously in the ring labeled by s. By virtue of Lemma 2.1, the statement is then also true (when all processes start execution simultaneously) in any ring labeled by a sequence of which s is a prefix.

5.2. Proposition. Let A be a p-algorithm, $0 \leq p \leq 1$, and for each sequence s of labels define $e_A(s)$ to be the probability that A sends a message with trace s. Then the function $e_A:D \longrightarrow [0,1]$ has the properties **P1** and **P2**(p).

Proof. Let t and s be two nonempty sequences of integers such that t is a prefix of s. From the definition of trace it follows that if A sends a message with trace s then A also sends a message with trace t. Hence $e_A(t) \geq e_A(s)$.

Let $s=(s_1\ s_2\ \ldots\ s_k)$ be any sequence in D. Consider executions of A in the ring labeled by s. No process can establish the maximum label in the ring without receiving a message whose trace has length at least k; since A is a p-algorithm, the probability of sending at least one message whose trace has length at least k is at least p. However, if a message whose trace has length at least k is sent then also a message whose trace has length *exactly* k is sent (the latter is the prefix of length k of the former). Consequently

$$\sum_{t\in C(s)} e_A(t) \geq p\ .$$

□

5.3. Theorem. If A is a p-algorithm and I is a finite set of n integers then

$$\overline{ave}_A(I) \geq n\,p\,H_n\ .$$

Proof. From the definition of $\overline{\nu}_A(s)$, $N(s,e)$ and e_A it follows that, for any permutation s of I,

$$\overline{\nu}_A(s) \geq N(s,e_A)\ .$$

Proposition 5.1 yields

$$\overline{ave}_A(I) = \frac{1}{n!}\sum_s \overline{\nu}_A(s) \geq \frac{1}{n!}\sum_s N(s,e_A) \geq n\,p\,H_n\ .$$

□

6. From functions on sequences to algorithms: An upper bound.

Consider the following property of functions $e:D\longrightarrow[0,1]$.

P3(p). *Cyclic product property.* If $s\in D$ then

$$\prod\{\,1-e(t)\mid t\in C(s)\,\} \leq 1-p\ .$$

Again, if e assumes only the values 0 and p then e has the properties **P1** and **P3(p)** if and only if the set $\{\,s\in D \mid e(s)=p\,\}$ is exhaustive in the sense of [3]. However, the function $e:D\longrightarrow[0,1]$ defined by $e(s)=1/\mathrm{len}(s)$ satisfies **P2(p)** but not **P3(p)** for $p=1$.

6.1. Theorem. If $e:D\longrightarrow[0,1]$ has the properties **P1** and **P3(p)** then there is a p-algorithm A such that for every finite set I of n integers we have

$$\overline{ave}_A(I) = \frac{1}{n!} \sum_s N(s,e) ,$$

where the sum is over all permutations s of I.

Proof. Let every process execute the following program:

```
constant integer   local_label
variable sequence  old_msg, msg

begin
   msg := (local_label)
   Send(msg) with probability e(msg)

   repeat
      begin
         Receive(old_msg)
         if local_label is in old_msg then goto DONE

         msg := Concatenate(old_msg, (local_label))
         Send(msg) with probability e(msg) / e(old_msg)
      end

   DONE: claim Max(msg) is the maximum label
end
```

In this program, the random transmissions ("Send(msg) with probability q") are arranged to be independent for different messages. That is, whenever a random transmission statement is to be executed, a new independent random variable is tested to determine whether the message should be sent.

In this algorithm, the content of each message is its trace. By induction in len(s) one can easily prove that, for each $s \in D$, the probability that this algorithm A sends a message with trace s is $e(s)$. In addition, if $F \subseteq D$ and no sequence in F is a prefix of another sequence in F then the events of sending a message with trace s are independent for $s \in F$. Therefore the probability that no process in the ring labeled by s reaches the statement label DONE is $\prod \{1-e(t) \mid t \in C(s)\}$, and by **P3**($p$) the probability that at least one process reaches DONE is at least p. Thus the algorithm is a p-algorithm.

Moreover, no message is sent whose trace does not belong to D, and for each $s \in D$ at most one message with the trace s is sent. Hence $\overline{\nu}_A(s) = N(s,e)$, and the theorem follows.

□

6.2. Theorem. For every p, $0 \le p \le 1$, there is a function $e_p : D \longrightarrow [0,1]$ that has the properties **P1** and **P3**(p) and such that, for every finite set I of n integers,

$$\sum_s N(s,e_p) = n!\, n\, p\, H_n \ ,$$

where the sum is over all permutations s of I.

Proof (cf. [1] and Example 2.3 in [3]). For $s=(s_1\ s_2\ \dots\ s_n)$ define

$e_p(s) = p$ if $s_1 = \max\limits_{1\le i\le n} s_i$;

$e_p(s) = 0$ otherwise .

It is easy to verify that e_p has the properties **P1** and **P3**(p).

Let card X denote the cardinality of the set X. For a fixed k, evaluate the sum

$$\sum_s \text{card}\ \{\, t=(t_1\ \dots\ t_k) \mid t_1 = \max_{1\le i\le k} t_i \ \text{ and } t \text{ is a prefix of some } r \in C(s) \,\}$$

(where the sum is over all permutations s of I): There are $n!/[k!(n-k)!]$ subsets J of I such that card$J=k$. For each such J there are $(k-1)!$ permutations $t=(t_1\ \dots\ t_k)$ of J such that $t_1 = \max\limits_{1\le i\le k} t_i$. For each such t there are $(n-k)!$ n permutations s of I such that t is a prefix of some $r \in C(s)$. Therefore the sum is equal to

$$\frac{n!}{k!\,(n-k)!}\,(k-1)!\,(n-k)!\,n \ = \ n!\,n\,\frac{1}{k}$$

and

$$\sum_s N(s,e_p) = \sum_s \sum \{\, e_p(t) \mid t \text{ is a prefix of some } r \in C(s) \,\}$$

$$= p \sum_{k=1}^{n} \sum_s \text{card}\ \{\, t=(t_1\ \dots\ t_k) \mid t_1 = \max_{1\le i\le k} t_i \text{ and } t \text{ is a prefix of some } r \in C(s) \,\}$$

$$= n!\, n\, p\, H_n \ .$$

□

6.3. Theorem. For every p, $0 \le p \le 1$, there is a p-algorithm A such that

$$\overline{ave}_A(I) = n\, p\, H_n$$

for every finite set I of n integers.

Proof. Combine 6.1 and 6.2.

□

6.4. Example. The following is a program description of an algorithm whose existence is proved in Theorem 6.3. (For $p=1$, the algorithm becomes that of Chang and Roberts [1].)

```
constant integer  local_label
variable integer  msg

begin
   msg := local_label
   Send(msg)  with probability  p

   repeat
      begin
         Receive(msg)
         if  msg = local_label  then goto DONE
         if  msg > local_label then Send(msg)
      end

   DONE:  claim  local_label  is the maximum label
end
```

7. Conclusion.

In this paper, probabilistic distributed algorithms for finding the maximum label in every asynchronous unidirectional ring configuration are described by functions from the set D to the interval [0,1]. The description is used to prove (in Theorem 5.3) that the algorithm in Example 6.4 leads to the smallest possible average expected number of messages.

The algorithm in Example 6.4, which depends on the parameter p, finds the maximum label with probability p. For $p=1$ the algorithm is deterministic. Thus, in the construction of algorithms to find the maximum label with probability 1, randomization does not decrease the average expected number of messages.

References

[1] E. Chang and R. Roberts, An improved algorithm for decentralized extrema-finding in circular configurations of processes. *Commun. ACM 22*, 5 (1979), 281-283.

[2] D. Knuth, *The Art of Computer Programming.* Vol. 1, *Fundamental Algorithms.* 2nd ed. Addison-Wesley, Reading, Mass., 1973.

[3] J. Pachl, E. Korach and D. Rotem, Lower bounds for distributed maximum-finding algorithms. *J. ACM 31*, 4 (1984), 905-918.

[4] A. S. Tanenbaum, *Computer Networks.* Prentice-Hall, New York, 1981.