NEW PROTOCOLS FOR THE ELECTION OF A LEADER IN A RING *

A. MARCHETTI-SPACCAMELA
Dipartimento di Informatica e Sistemistica, Università di Roma, 00185 Roma, Italy.

Abstract. In this paper we investigate the impact of time for the election of a leader in a distributed environment. We propose a new protocol schema that can be specialized to obtain several protocols with different communication-time characteristics when the network is ring-shaped and the communications between processors are synchronous.

1. Introduction

In recent years great attention has been devoted to the computational resources required to solve problems in a distributed environment. In this paper we consider the problem of electing a leader in a synchronous ring-shaped network. There are n processors, but this is not known to the processors. The processors have only local information of the network and are identical except that each one has its own identifier. At various points in time one or more processors 'wake up' and initiate their participation in an election to decide on a unique leader among the participating processors. We assume that the ring is unidirectional (i.e., each processor receives messages from one of its neighbours and sends messages to the other one). The interesting resources are the total number of messages used and the time.

This problem is important not only because it occurs in practical situations (i.e., crash recoveries), but also because the communication costs (both upper and lower bounds) required in order to achieve any agreement in a decentralized network seem to be of the same order as the costs for the election of a leader. The problem has received attention by a number of researchers [1-8]. In the case of a ring-shaped asynchronous network $O(n \log n)$ messages and $O(n)$ time are sufficient [3, 7]; furthermore, it has been proved that $\Omega(n \log n)$ messages are necessary, even if all processors know the size of the ring [2]. Since the lower bound proof does not apply to the case of synchronous networks, it is interesting to study the impact of synchronization on the number of messages used.

A first important step in this direction has been performed by Frederickson and Lynch [4] and, independently, by Vitanyi [8]. They found a protocol that uses $O(n)$ messages and exponential time. Namely, the protocol assumes that the identities of the processors are numbers and requires $O(n2^{i_{\min}})$ time units, where $i_{\min}$ is the

* Work supported by project MPI Analisi e Progetto di Algoritmi.

identity of the leader. Furthermore, it has been shown that both the use of time and of the identity number is essential if we want to use less than $O(n \log n)$ messages. In fact, Frederickson and Lynch [4] proved that in the synchronous case

(i) if we consider algorithms that use only comparisons, then $\Omega(n \log n)$ messages are necessary;

(ii) there is a (fast increasing) function $f(n, t)$ such that if we allow identities to be chosen between 1 and $f(n, t)$, then any protocol that finds a leader in less than t time units needs $O(n \log n)$ messages.

As a consequence of these results it is quite natural to investigate the trade-off between time and number of messages required for the election of a leader in a synchronous ring. In this paper we address the question whether it is possible to obtain protocols that use less than exponential time and between $O(n)$ and $O(n \log n)$ messages. We positively answer this question by showing a protocol scheme based on a new technique that can be specialized in order to obtain several protocols with different communication-time characteristics.

In Section 2 we will present the main idea of the protocol scheme that will be presented in detail in Section 3. In Section 4 we will analyse the resources required by the scheme and in Section 5 we will obtain a number of different protocols. If $i_{\min}$ is the number representing the identity of the leader we can obtain protocols with the following characteristics:

(P1) $O(n \log_c n)$ messages and $O(c\, n\, i_{\min})$ time, $c \geq 2$;

(P2) $O(n \log_a \log_c n)$ messages and $O(n^a\, i_{\min})$ time, $c \geq 2$, $a > 1$;

(P3) $O(n \log_2 \log_2 \log_2 n)$ messages and $O(n^{\log_2 n}\, i_{\min})$ time;

(P4) $O(n \log^* n)$ messages and $O(2^{n^\varepsilon}\, i_{\min})$ time, $\varepsilon > 0$.

For the rest of the paper we assume, without loss of generality, that the identities of the processors are positive numbers and that the leader will be the participating processor with the smallest identity.

2. A (too much) simplified protocol

In this section we present a simple and efficient protocol based on relaxing some of the assumptions of the problem. The simple protocol we obtain is a good starting point in order to present how time is used in the protocol scheme presented in Section 3. We make the following assumptions:

(a) each processor knows n, the size of the ring;

(b) all processors start the election at the same time $t = 0$.

If assumptions (a) and (b) hold and processor 1 takes part in the election, then it is the winner; hence, the only thing it has to do is to broadcast to all processors the message '1 elected'. If processor 1 does not participate in the election, then there is no message in the ring for the first n time units. This implies that if processor 2 participates at the election and does not receive the message '1 elected', it knows at time $n - 1$ that it is the winner and broadcasts at time n the message '2 elected'.

Analogously if processor i is participating at the election, it waits for $n(i-1)-1$ time units. If it has not received any message, it is the winner and sends at time $n(i-1)$ the message 'i elected'.

Clearly, the protocol is correct and requires $n+1$ messages and time $O(i_{\min} n)$ where $i_{\min}$ is the identity of the winner.

If we know an upper bound N on n, the protocol can be easily modified and its time complexity is $O(i_{\min} N)$. It is possible to obtain a modified protocol when assumption (b) does not hold, but in order to eliminate assumption (a) we need major modifications because the knowledge on n (or an upper bound on n) is essential in this protocol.

The main idea in order to circumvent the problem of knowing n is based on guessing its value. Roughly speaking the new protocol will use hypotheses $h_1, h_2, \ldots$ of increasing value on the length of the ring. When hypothesis h_m holds, the processors are in phase m and will behave according to the protocol introduced in this section as if the length of the ring is h_m. If n is greater than h_m, a new hypothesis h_{m+1} will be formulated. The protocol will proceed in this way until a hypothesis $h_t \geqslant n$ will be formulated; at the end of this phase the election is terminated. We will obtain different protocols depending on the way we pass from hypothesis h_m to hypothesis h_{m+1} as we will see in Section 5.

3. The protocol scheme

The first problem that we face in order to formalize the ideas sketched at the end of the preceding paragraph is that not all the processors wake up at the same time. In order to solve this problem we use an idea introduced in [4, 8]. Namely, we have a preliminary phase 0; when a processor decides to participate in the election, it spawns the message 'election started' to its neighbour. The message is transmitted along the ring until it meets a processor that has already sent this message. When a processor receives the message 'election started', it decides whether it wants to take part at the election; in any case it enters phase 1. A processor cannot decide to participate at the election after it has received the first message. Not all identities start phase 1 at the same time: let $t(i)$ be the time identity i starts phase 1.

In the following phases of the protocol we focus on the identities of the participating processors. At the beginning all participating identities are 'alive'; during the election they will travel along the ring, increasing their phase number, and will eventually 'die' as soon as they become aware that there is a 'smaller identity' participating in the election. At the end, the smallest identity will be the only identity alive and the corresponding processor will be the leader.

At each moment the identities will travel along the ring at different speeds depending on their value and on their phase number, and each processor will memorize the smallest participating identity it is aware of. When processor i receives identity j in phase m it decides on one of the following actions:

(a) if j is the identity of the processor itself, then the processor is the winner of the election and sends the message 'j elected';

(b) if it is aware of a smaller participating identity, it will kill j because j cannot be the leader any more;

(c) if j is the lowest identity the processor has seen, then if another identity j' is waiting at processor i, then j' will be killed because it cannot be the leader anymore. Then, if j has not completed phase m, processor i will send j to its neighbour; otherwise, if j has completed phase m, the processor will delay j for a time proportional to the value of j. If at the end of this period no better value has arrived, then processor i sends j to its neighbour.

In this way the number of alive messages will decrease as the election proceeds and the elected processor will be the only one whose identity will be back home (i.e., the only identity that will walk around the complete ring).

The stop and go schema is implemented as follows: at the beginning, each participating processor formulates an hypothesis h_1 on the length of the ring and will delay its identity i for $2ih_1$ time units. If, in the meantime, no better identity has arrived, the processor sends the message $\langle i, h_1, 1\rangle$ (identity, hypothesis, distance walked) to its clockwise neighbour that will either kill it or will send the message $\langle i, h_1, 2\rangle$ to its clockwise neighbour that will either kill it or will send the message $\langle i, h_1, 3\rangle$. The identity will walk in this way for h_1 processors. When a processor p receives the message $\langle i, h_1, h_1\rangle$, identity i has performed phase 1 at time $t(i)+h_1(2i+1)$.

Processor p will formulate for i a new hypothesis h_2, $h_2 > h_1$, on the length of the ring and will delay identity i for $2i(h_2-h_1)$ time units. If, at the end of this period, no better identity has arrived, identity i will start its second phase and the message $\langle i, h_2, h_1+1\rangle$ will be sent. During the phase identity i will walk for h_2-h_1 processors in the same way as before.

The phase will be completed when identity i has walked for h_2 processors or it has returned to its starting point. In the former case a new hypothesis will be formulated; in the latter case i is the winner. The identity i will proceed formulating new hypotheses on the length of the ring until it dies or it has returned to its starting point. Note that at each processor there is one waiting message at most.

Each processor i will use three local variables 'bestid', M' and 'count' with the following meaning:

- 'bestid' is an integer that memorizes the value of the best identity that i has seen in so far;
- M is a triple $\langle j, h_m, w\rangle$ that memorizes the message that is eventually waiting at i;
- 'count' is an integer that is used to count the delay of the waiting message.

The protocol scheme for processor i

```
/phase 0/
if awaken
```

```
then
begin
  send message ‘election started’ to clockwise neighbour;
  receive message;
  /the processor waits till it receives a message ‘election started’ from its anticlock-
  wise neighbour/
  /prepare message/ $M' := \langle i, h_1, 1\rangle$;
  count := $2ih_1$;
  /$M'$ will be delayed for count time units/
  bestid := $i$;
end
else
begin
  receive message;
  /the processor waits till it receives a message ‘election started’ from its anticlock-
  wise neighbour/
  send message ‘election started’ to clockwise neighbour;
  if willing to participate
  then begin
    /prepare message/ $M' := \langle i, h_1, 1\rangle$;
    count := $2ih_1$;
    /$M'$ will be delayed for count time units/
    bestid := $i$;
  end
  else bestid := $+\infty$
end;

/following phases/
repeat in each (local) time unit
  read incoming message $M$ from anticlockwise neighbour;
  /if no message is received in this time unit, then assume $M = \langle j, 1, 1\rangle$ with
  $j >$ bestid/
  if message $M$ is ‘$i$ elected’ then the election is finished;
  /everybody knows the winner is me, $i$/
  if the message is ‘$j$ elected’ and $j \neq i$
  then send message ‘$j$ elected’ to clockwise neighbour;
  if message $M$ is $\langle j, h_m, w\rangle$
  then begin
    if $j = i$ then/proclaim $j$ elected/
      send message ‘$j$ elected’ to clockwise neighbour;
    if $j <$ bestid
    then begin
      bestid := $j$;
```

```
        if there is a waiting message $M'$ then kill it;
        if $w < h_m$
        then /phase $m$ is not finished/
          send $\langle i, h_m, w+1\rangle$ to clockwise neighbour;
        else /phase $m$ is finished/
        begin
          /prepare message/ $M' := \langle j, h_{m+1}, w+1\rangle$
          count $:= 2j(h_{m+1} - h_m)$
          /$M'$ will be delayed for count time units/
        end
      end;
      if $j >$ bestid
      then begin count $:=$ count $- 1$;
        if count $= 0$ and $M'$ has not been killed
        then send $M'$ to clockwise neighbour
      end
    end
until the election is finished.
```

4. Analysis of the protocol schema

The correctness of the algorithm follows from the following observations:

(1) for each participating identity there is at most one message in the ring;
(2) at the end of phase m an alive identity either has walked for h_m steps or it is back at its starting point;
(3) for all m, h_{m+1} is greater than h_m;
(4) there is only one identity, the identity of the leader, that arrives at the processor it started from.

Theorem 4.1. *The protocol finishes in a finite amount of time and at the end a leader has been elected.*

Proof. The proof follows from observations (1) through (4). □

The rest of this section will be devoted to the analysis of the requirements of the protocol scheme.

The proof of the following theorem is trivial and is omitted.

Theorem 4.2. *Phase* 0 *requires n messages and there are at most n time units between the time the first processor wakes up and the time the last identity decides to participate at the election.*

The evaluation of the number of messages required in phase 1 is complicated by the fact that not all the identities start the phase at the same time. Furthermore, the analysis of the following phases is more difficult because the time at which identity i will start phase m, $m = 2, 3, \ldots$, depends on the time at which i starts phase 1 and on the value of the identity itself.

The following theorem allows to bound the number of messages sent for each phase.

Theorem 4.3. *For any set I of participating processors the total number of messages sent in phase m, $m > 0$ (i.e., the total number of messages of the form $\langle i, h_m, w\rangle$, $i \in I$, $h_{m-1} < w \leq h_m$), is less than $2n$.*

In order to prove the theorem we need some preliminary observations. First of all note that, during the execution of the algorithm, the relative order in the ring among alive identities does not change, because an identity that is passed by a faster one dies.

This allows to define a cyclic order on any set S of alive identities. We say that $S = (\text{id}(1), \text{id}(2), \ldots, \text{id}(n))$ is *ordered* if

(a) $\text{id}(i-1)$ is the immediate predecessor of $\text{id}(i)$ (in clockwise order), $i = 2, 3, \ldots$.

(b) $\text{id}(n)$ is the immediate predecessor of $\text{id}(1)$ (in clockwise order).

Given two identities i and j, let $\text{dist}(i, j)$ be the clockwise distance between processors i and j; observe that

$$\text{dist}(i,j) + \text{dist}(j,i) = n, \qquad t(j) \leq t(i) + \text{dist}(i,j).$$

The next lemma gives a bound on the maximum total delay that we can have for any ordered set of identities.

Lemma 4.4. *For any ordered set of identities $S = \text{id}(1)$, $i = 1, 2, \ldots, k$, we have*

$$\sum_{i=2}^{n} \max(t(\text{id}(i)) - t(\text{id}(i-1)), 0) + \max(t(\text{id}(1)) - t(\text{id}(n)), 0) \leq n.$$

Proof. It is sufficient to observe that a phase-0 message takes at most n time units to visit the ring. □

Proof of Theorem 4.3. We colour the set of alive identities at phase m as follows:

- identity i is *white* if there is some other identity j such that the segment walked through by i during phase m is completely contained in the segment walked through by j;
- identity i is *black* if there is no processor that, during phase m, sends i and, after i, another identity $j < i$;
- identity i is *red* if it is neither white nor black.

It is easy to see that the segments walked through by black identities are non-overlapping. Hence, the number of messages used for black identities is no more than n.

In order to bound the number of messages used for red and white identities we identify pairs of identities $\langle \mathrm{id}(1), k(1)\rangle$, $\langle \mathrm{id}(2), k(2)\rangle, \ldots, \langle \mathrm{id}(r), k(r)\rangle$ such that

(1) $\mathrm{id}(1), k(1), \mathrm{id}(2), \ldots$ are clockwise ordered;

(2) $k(i)$ is different from $\mathrm{id}(i)$, $i=1,2,\ldots,r$ (but it may happen that $k(i)=\mathrm{id}(i+1)$);

(3) $t(\mathrm{id}(i))$ is greater than $t(k(i))$, $i=1,2,\ldots,r$.

We first show that the total number of messages used for white and red identities is no more than

$$\sum_{i=1}^{r} t(\mathrm{id}(i))-t(k(i)).$$

Later on, Lemma 4.4 will be used to bound the above sum.

Let us consider a non-white identity i and let us define

$$\mathrm{id}(1)=i,$$

$$B(1)=\{j \mid \text{there is a processor that, during phase } m\text{, first sends } j\neq \mathrm{id}(1) \text{ and later } \mathrm{id}(1)\}.$$

Note that if j belongs to $B(1)$, then $j>\mathrm{id}(1)$ and j is not black (if either $j<\mathrm{id}(1)$ or j is black, then there is no processor that first sends j and later $\mathrm{id}(1)$). Furthermore, since identity j is greater than $\mathrm{id}(1)$, j enters phase m before identity $\mathrm{id}(1)$ finishes phase m (otherwise, there is no processor that, during phase m, sends both j and $\mathrm{id}(1)$).

This observation is formalized in the following lemma.

Lemma 4.5. *If j belongs to $B(1)$, then*

$$t(j)<t(\mathrm{id}(1))-h_m(j-\mathrm{id}(1)).$$

Proof. If the lemma does not hold, then j enters phase 1 at time

$$\begin{aligned} &t(j)+2h_m j+h_{m-1}\\ &\quad=t(j)+2h_m\,\mathrm{id}(1)+2h_m(j-\mathrm{id}(1))+h_{m-1}\\ &\quad>t(\mathrm{id}(1))+2h_m\,\mathrm{id}(1)+h_m.\end{aligned}$$

The last term denotes the time at which identity $\mathrm{id}(1)$ finishes phase m. This completes the proof of the lemma. □

Proof of Theorem 4.3 (*continued*). If $B(1)$ is not empty, then we distinguish two cases.

Case 1: there is no red identity in $B(1)$. Let us define $k(1)=\max(j, j\in B(1))$. Note that $k(1)\geq |B(1)|+\mathrm{id}(1)$. Applying Lemma 4.5 we have that the total number of messages sent with identities belonging to $B(1)$ is no more than

$$|B(1)|h_m\leq h_m(k(1)-\mathrm{id}(1))\leq t(\mathrm{id}(1))-t(k(1)).$$

Having bounded the number of messages belonging to $B(1)$ let id(2) be the first identity, in clockwise order, that follows id(1) and does not belong to $B(1)$. It is easy to see that id(2) is black. In fact, if id(2) is not black, then there is another identity s such that

(1) there exists a processor that during phase m first sends id(2) and after that s (since id(2) is not black);

(2) s belongs to $B(1)$ (since id(2) is the first identity that follows id(1) and does not belong to $B(1)$);

(3) s is white (since s belongs to $B(1)$ and we are in Case 1).

The fact that s is white implies that the segment walked through by s during phase m is completely contained in the segment walked through by id(1). This contradicts the fact that id(2) does not belong to $B(1)$.

Case 2: there is a red identity in $B(1)$. Now, let id(2) be the red identity that belongs to $B(1)$ and whose clockwise distance from id(1) at the beginning of phase m is largest. Let us define

$$\begin{aligned} B'(1) = \{j \mid j \in B(1) &\text{ and identity } j \text{ is at the beginning of phase } m \\ &\text{placed between id}(1) \text{ and id}(2)\} \\ &\cup \{\text{id}(2)\}, \\ k(1) = \max(j \in B'(1)). \end{aligned}$$

Note that $k(1) \geqslant \text{id}(1) + |B'(1)|$. Applying Lemma 4.5 we have that the total number of messages sent with identities belonging to $B'(1)$ is no more than

$$|B'(1)|h_m \leqslant h_m(k(1) - \text{id}(1)) \leqslant t(\text{id}(1)) - t(k(1)).$$

In both cases (a) and (b) we have defined two identities id(1) and id(2) and we have bounded the total number of messages sent with non-black identities placed in the ring between id(1) and id(2). Observe that if id(1) is red, the messages sent with identity id(1) have not been counted; they will be considered at the end. On the other hand, if id(2) is red, it belongs to $B'(1)$; this implies that the distance walked by it has been considered.

Having defined id(2) we continue in a similar way by defining

$$B(2) = \{j \mid \text{there is a processor that, during phase } m\text{, first sends } j \text{ and after id}(2)\}$$

and we distinguish two cases (a) and (b) depending on whether there is a red identity in $B(2)$ or not. In both cases we proceed as before by defining $k(2)$ as the largest identity belonging to $B(2)$ ($B'(2)$) and by bounding the total number of messages used for identities in $B(2)$ ($B'(2)$) by $t(\text{id}(2)) - t(k(2))$. We proceed in a similar way defining id(3), $k(3)$, id(4), ..., id(r), $k(r)$ until we have considered all alive non-black identities. Observe that if id(1) is red, then it will belong to $B'(r)$.

Note that each non-black identity has been considered. Suppose that there is a non-black identity s that has not been considered and suppose that it is placed between id(i) and id($i+1$). Since the messages sent with identity s have not been counted, we have that $B(i)$ does not contain a red identity and we are in Case 1 (otherwise all non-black identities between id(i) and id($i+1$) have been considered). Hence, s and id($i+1$) must coincide (since id($i+1$) has been chosen as the first clockwise identity that does not belong to $B(i)$). This contradicts the fact that s is not black.

As a conclusion the total number of messages used by red and white identities during phase m is not more than

$$\sum_{i=1}^{r} t(\mathrm{id}(i)) - t(k(i)).$$

The sequence of identities id(1), $k(1)$, id(2), ... is an ordered set of identities and applying Lemma 4.4 we have

$$\begin{aligned}
&\sum_{i=1}^{r} (t(\mathrm{id}(i)) - t(k(i))) \\
&\quad = \sum_{i=1}^{r-1} (t(\mathrm{id}(i+1)) - t(k(i))) + t(\mathrm{id}(1)) - t(k(r)) \\
&\quad \leqslant \sum_{i=1}^{r-1} (\max(t(\mathrm{id}(i+1)) - t(k(i))), 0) + \max(t(\mathrm{id}(1)) - t(k(r)), 0).
\end{aligned}$$

Applying Lemma 4.4 we obtain the thesis. This completes the proof of Theorem 4.3. □

5. Four different protocols for the ring

Theorem 4.3 is the key theorem in the analysis of the different protocols that we can obtain from the protocol scheme presented in Section 3. If a protocol uses $h_1, h_2, \ldots$ as hypotheses, it terminates when the winning identity has completed phase h_t, where h_t is such that $h_{t-1} < n \leqslant h_t$. Hence, the number of messages used is $O(t\,n)$ and the time required is $O(i_{\min}\, h_t)$. At this point the trade-off between time and total number of messages used is clear: in order to diminish the total number of messages sent, we want to have the number of phases as minimal as possible, but in this case we will have a rough approximation on the value of n that will affect the time required by the protocol.

Finally we show four different protocols based on four different ways of passing from one hypothesis to the following one.

(P1) $h_1 = c$, $c \geqslant 2$, $h_m = c\, h_{m-1}$;
(P2) $h_1 = c$, $c \geqslant 2$, $h_m = (h_{m-1})^a$, $a > 1$;
(P3) $h_1 = c$, $c > 2$, $h_m = (h_{m-1})^{\log_2 h_{m-1}}$;
(P4) $h_1 = c$, $c \geqslant 2$, $h_m = 2^{h_{m-1}^{\varepsilon}}$, $\varepsilon > 0$.

Theorem 5.1. *The number of messages and the time required by protocols* (P_i), $i = 1, 2, 3, 4$ *are as follows*:

(P1) *the time required is* $O(c\, n\, i_{\min})$ *and the number of messages is* $O(n \log_c n)$;

(P2) *the time required is* $O(n^a\, i_{\min})$ *and the number of messages is* $O(n \log_a \log_c n)$;

(P3) *the time required is* $O(n^{\log_2 n}\, i_{\min})$ *and the number of messages is* $O(n \log_2 \log_2 \log_2 n)$;

(P4) *the time required is* $O(2^{n^\varepsilon}\, i_{\min})$ *and the number of messages is* $O(n \log^* n)$.

Proof. (P1): We have

$$n \leqslant h_t = c^t < cn, \qquad t \leqslant \log_c n + 1;$$

hence, the time required is $O(c\, n\, i_{\min})$ and the number of messages is $O(n \log_c n)$.

(P2): We have

$$n \leqslant h_t = c^{a^t} < n^a, \qquad t \leqslant \log_a \log_c n + 1;$$

hence, the time required is $O(n^a\, i_{\min})$ and the number of messages is $O(n \log_a \log_c n)$.

(P3): We have

$$n \leqslant h_t = c^{\log_2 c)^{2^{t-1}}} < n^{\log_2 n},$$

$$t \leqslant a \log_2 \log_2 \log_2 n + 1 \quad \text{for some constant } a > 1;$$

hence, the time required is $O(n^{\log n}\, i_{\min})$ and the number of messages is $O(n \log_2 \log_2 \log_2 n)$.

(P4): We have

$$n \leqslant h_t < 2^{n^\varepsilon},$$

$$h_t = \left.2^{2^{2^{\cdot^{\cdot^{\cdot^{2^{c^\varepsilon}}}}}}}\right\} t-2 \text{ times},$$

$$t = O(\log^* n);$$

hence, the time required is $O(2^{n^\varepsilon}\, i_{\min})$ and the number of messages is $O(n \log^* n)$. □

References

[1] D. Angluin, Local and global properties in network of processors, in: *Proc. 12th Ann. ACM Symp. on Theory of Computing* (1980) 82-93.

[2] J.E. Burns, A formal model for message passing systems, Tech. Rept TR91, Indiana University, 1980.

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

[4] G. Frederickson, N. Lynch, The impact of synchronous communication on the problem of electing a leader in a ring, in: *Proc. 16th Ann. ACM Symp. on Theory of Computing* (1984) 493-503.

[5] E. Gafni, Improvements in the time complexity of two message-optimal election algorithms, in: *Proc. 4th ACM Symp. on Principles of Distributed Computing* (1985) 175-185.

[6] D.S. Hirschberg, J.B. Sinclair, Decentralized extrema finding in circular configurations of processes, *Comm. ACM* **23** (1980) 627-628.

[7] G.L. Peterson, An O(n log n) unidirectional algorithm for the circular extrema problem, *ACM Trans. Programming Languages Systems* **4**(4) (1982) 758-762.

[8] P. Vitanyi, Distributed elections in archimedean ring of processors, in: *Proc. 16th Ann. ACM Symp. on Theory of Computing* (1984) 542-547.