

Leader Election in Rings with Homonyms

Carole Delporte-Gallet^(✉), Hugues Fauconnier, and Hung Tran-The

LIAFA- Université Paris-Diderot, Paris, France
{cd,hf,Hung.Tran-The}@liafa.univ-paris-diderot.fr

Abstract. Considering the case of homonyms processes (some processes may share the same identifier) on a ring, we give here a necessary and sufficient condition on the number of identifiers to enable leader election. We prove that if l is the number of identifiers then message-terminating election is possible if and only if l is greater than the greatest proper divisor of the ring size even if the processes do not know the ring size. If the ring size is known, we propose a process-terminating algorithm exchanging $O(n \log(n))$ messages that is optimal.

1 Introduction

The goal of the model of homonym processes [10] is to be an extension of the classical model in which every process has its own unique identity and the model in which the processes are fully anonymous. With homonyms, processes have an identifier coming from a set of identifiers \mathcal{L} and as we always assume that each identifier is the identifier of at least one process, if $|\mathcal{L}|$ is equal to the number of processes n each process has its own identity and $|\mathcal{L}|$ is 1, the processes are fully anonymous. But the cases where $0 < |\mathcal{L}| < n$ are rather natural and useful. For example in communication signatures [9], signatures are identifiers and process groups share the same signature to maintain some kind of privacy (it could be impossible to distinguish between processes with the same identifier). Moreover it is generally very costly to assign unique identities to processes and some mechanisms for this (e.g. name servers, hash function) do not avoid to have sites with the same identifier.

Until now homonyms have been mainly studied in the context of process failures (Byzantine, omission, crash) for the Consensus problem [4, 10–12] and it has been proved that in many cases unique identifiers are not needed.

A classical and fundamental problem in distributed algorithms is the leader election problem. The main purpose of our paper is to study the leader election problem in the framework of homonyms processes and determine in which way identifiers are needed. In this paper we restrict ourselves to ring topologies (of known or unknown size). We give necessary and sufficient conditions on $|\mathcal{L}|$ the number of identifiers enabling to solve the leader election problem.

From years, many results have been proved concerning the leader election problem depending on the network topology (e.g. [2, 5, 7, 22, 23]). Clearly some

This work is supported by the ANR DISPLEXITY.

of our results, in particular impossibility results, in the framework of homonyms may be deduced from those results. Nevertheless the results and algorithms given in this paper are rather simple and neat. Moreover they depend only on the number of identifiers and not on their layout.

More than thirty years ago it has been proved that with anonymous processes there is, in general, no deterministic solution for the leader election problem [2].

To avoid impossibility results, randomization is a well known technique to break symmetry and randomized election algorithms have been considered on a variety of network topology (e.g. ring, tree, complete graph) in, for example, [1, 17, 21], but in this paper we do consider only deterministic algorithms.

The main purpose of our work is to study how to break symmetry with homonymous and not the properties of communication graphs. Hence we restrict ourselves to leader election in (bidirectional) rings with homonyms.

Concerning anonymous processes in rings, Dijkstra [13] observed that, with a particular communication model called central demon, a deterministic leader election algorithm cannot exist if the ring size is a composite number. Several papers [6, 16] present leader election algorithms for anonymous rings of prime size with some kinds of demons. In this paper we consider asynchronous message passing and with this model of communication it is easy to verify that no deterministic election algorithm is possible with anonymous processes. On the other hand, if each process has its own identity the election on a ring is easy. Between these extremes, weakening anonymous condition with homonyms processes, it is interesting to determine how many identifiers are sufficient and necessary to have a deterministic election algorithm.

More precisely we directly prove that there are deterministic solutions for election on a ring if and only if the number of identifiers is strictly greater than the greatest proper divisor of the ring size and we explicitly give a solution. For example, if the ring size is a prime number, then there is a deterministic election algorithm if and only there are at least two identifiers.

An interesting point is the fact that this necessary and sufficient condition on the number of identifiers holds even if the ring size is not known by processes when we consider deterministic message terminating algorithms (i.e. processes do not terminate but only a finite number of messages are exchanged). Moreover in the last section when the number of identifiers is known, we propose a deterministic process terminating election algorithm exchanging $O(n \log(n))$ messages. This message complexity is clearly optimal [19].

2 Model and Definition

We consider a distributed message-passing system of n processes ($n \geq 2$). There are neither process failures nor link failures. In the following, we establish some results assuming that process knows n and some results without this knowledge. Each process gets an identifier from a set of identifiers $\mathcal{L} = \{1, 2, \dots, l\}$. We assume that each identifier is assigned to at least one process, that knows it, but some processes may share the same identifier. Processes with the same identifier

have the same deterministic algorithm. In our paper, we sometimes refer to individual processes using names like p , but these names cannot be used by the processes themselves in their algorithm. The identifier of a process p is denoted $Id(p)$.

Message-passing communication between processes is asynchronous. Processes are organized in a bidirectional ring, so that a process can send messages to its two neighbors. The communication channels are reliable and Fifo. The distance from p to q , say $d(p, q)$, is 1 plus the number of processes between p and q in the clockwise ring. Note that if p_0, \dots, p_{k-1} are any k different processes in the ring $\sum_{i=0}^{k-1} d(p_i, p_{i+1 \bmod k}) = n$

A sequence of n identifiers represents a ring of size n . For example $\langle 1, 2, 3, 3, 1, 2, 3, 3 \rangle$ is the ring of Fig. 1. All rotations or reflections of this sequence represent the same ring (i.e. $\langle 2, 3, 3, 1, 2, 3, 3, 1 \rangle$ denotes also the ring of Fig. 1).

We consider the *leader election* problem. Each process has a boolean variable *Leader* and a process is *elected* if its variable *Leader* is equal to true.

Definition 1. *A process p is a leader if eventually it is elected forever: there is a time τ after which the variable *Leader* of p is forever equal to true.*

An algorithm solves the leader election problem if there is a unique leader. We consider two variants of this problem *process-terminating* and *message-terminating*. In the case of process-terminating, eventually all processes terminate, in a case of message-terminating, eventually there is no message exchanged.

Definition 2. *An algorithm solves process-terminating leader election problem for n processes if for any ring of size n : (1)(leader) there is a unique leader, and (2)(process-terminating) eventually all processes terminate.*

Note that process-terminating leader election problem is equivalent to the problem where each process takes an irrevocable decision to be leader or not.

Definition 3. *An algorithm solves message-terminating leader election problem for n processes if for any ring of size n : (1)(leader) there is a unique leader, and (2)(message-terminating) eventually there is no exchanged message.*

Definition 4. *If x is an integer, the greatest proper divisor of x , $gpd(x)$, is the greatest divisor of x excluding x itself.*

Note that if x is prime, $gpd(x) = 1$. From this definition we get:

Proposition 1. *If $x \geq 2$, then $x/gpd(x) \geq 2$.*

3 Impossibility Results

To strengthen our results, we prove them when the communication is synchronous. So we can assume that we have a round model in which in each round a process (1) may send a message to its two neighbors, (2) waits δ (during this time it receives the messages sent by its neighbors if any), and (3) computes its new state according to its current state and the received messages.

3.1 Size n Known by Processes

The following result can be deduced from [5, 8, 22], but in our case it is easier to show it directly.

Theorem 1. *Assuming that n is known by all processes and $|\mathcal{L}| \leq \text{gpd}(n)$, process-terminating leader election for n processes is unsolvable.*

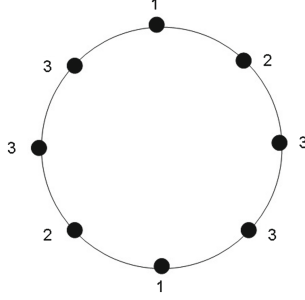


Fig. 1. An example: $n = 8$, $|\mathcal{L}| = 3$.

Proof. We show this theorem by contradiction. Assume that there exists an algorithm \mathcal{Alg} that solves process-terminating leader election problem for n processes. We construct a ring of size n in which process terminating leader election problem is unsolvable. If $|\mathcal{L}| = 1$, all processes have the same identifier and we are in an anonymous system. From the classical result [2], process terminating leader election is unsolvable in an anonymous ring and in synchronous case. Thus the proposition holds. Now, consider that $|\mathcal{L}| \geq 2$.

Let k be $\text{gpd}(n)$. We call a k -segment a sequence of k processes positioned in the clockwise direction as follows: the process with identifier 1 is positioned at the first position of the segment, the process with identifier 2 at the second position of the segment, ..., process having identifier $(|\mathcal{L}| - 1)$ at the $(|\mathcal{L}| - 1)$ th position of the segment. $k - |\mathcal{L}| + 1$ remaining processes having identifier $|\mathcal{L}|$ are positioned from the $|\mathcal{L}|$ th to the k th position. We can construct a ring of size n by n/k continuous segments. They are symmetrically distributed in the ring. For example, $n = 8$, $|\mathcal{L}| = 3$ (see Fig. 1).

It is straightforward to verify, by induction on the number of rounds, that in every round, the n/k processes with the identical positions on the segments are in identical states. (All processes with identical positions on the segments have the same identifier, they start in the same state and execute the same code).

Consider the round τ , at which all processes have terminated, they end up in the same state.

If one process in some segment is leader then every process with the identical position of another segment is also leader. Then n/k processes are leader. By Proposition 1, n/k is greater than 2, contradicting the fact that \mathcal{Alg} solves the leader election problem and we must have a unique leader.

Theorem 2. *Assuming that n is known by all processes and $|\mathcal{L}| \leq \text{gpd}(n)$, message-terminating leader election for n processes is unsolvable.*

Proof. We show this theorem by contradiction. Assume that there exists an algorithm that solves message terminating leader election problem for n processes. The proof is the same that the proof of Theorem 1 but instead of considering the round τ , at which all processes have terminated, we consider the round τ after which no message are exchanged.

3.2 Size n Unknown by Processes

If n is unknown, that means that the algorithm for a process with some identifier is the same for any values of n .

Theorem 3. *Even if $|\mathcal{L}| > \text{gpd}(n)$, there exists no algorithm for process-terminating leader election if the processes don't know the ring size.*

Proof. Assume there is an algorithm \mathcal{Alg} that solves process-terminating leader election for n processes.

Let $S = \langle i_1, i_2, \dots, i_n \rangle$ be a ring of size n , consider an execution e of \mathcal{Alg} on this ring. By hypothesis, there exists r such that after r rounds all processes have terminated and exactly one process in S is *leader*.

We consider the ring $S' = \langle j_1, \dots, j_m \rangle$ of size $2n$ that is composed by $\langle i_1, i_2, \dots, i_n \rangle$ 2 times. s_1, s_2 denotes these 2 segments. Consider now an execution of \mathcal{Alg} on S' . After r rounds, the k th process of segment s_1 and the k th process of segment s_2 is at each end of round in the same state that the k th process of S in the execution e . Then after r rounds, all processes in s_1 and s_2 terminate. As in execution e there is exactly one leader, in the execution of \mathcal{Alg} on S' there are at least 2 leaders contradicting the fact that \mathcal{Alg} solves the leader election problem.

4 Deterministic Leader Election with Homonyms

4.1 The Simple Protocol

The model we consider in this subsection is essentially a model of population protocol [3], the only difference is that here the state of an agent is an integer and hence an agent is not a finite state automaton. We present a very simple election protocol in this model.

In the *simple protocol* we consider a set, \mathcal{A} , of fully anonymous agents, each agent a is described by an automaton. The state of an agent is an integer initialized at some value $v_a > 0$ for agent a . As in population protocol, two agents may meet together, in this case if agent a meets agent b then they both change their states following the rule R (s_x denote the state of agent x before the meeting and s'_x the state for agent x after the meeting):

$$\text{if } s_a > s_b \text{ then } s'_a = s_b + s_a \text{ and } s'_b = 0$$

If $s_a = s_b$ the states of the agents are not modified.

A global state, S of the system is the states of each agent: for global state S and agent a , $S(a)$ is the value of s_a in this state.

Let $V = \sum_{a \in \mathcal{A}} v_a$. An easy induction proves that the sum of the states of each agent is invariant, hence for each global state S of the system, we have $\sum_{a \in \mathcal{A}} S(a) = V$.

On the other hand, assuming that each agent meets infinitely often all the other agents, it is easy to verify that eventually a global state will be reached such that all agent states are either equal to 0 or to some common value M . Moreover after such a global state T , any meeting between agents will not change T . Hence, the protocol converges to a global state T such that there is a M for all $a \in \mathcal{A}$ we have $T(a) = M$ or $T(a) = 0$.

Remark that $M \geq \max\{v_a | a \in \mathcal{A}\}$. If $Lead$ is the set of agents with state equal to M in global state T , then from the invariant ($\sum_{a \in \mathcal{A}} S(a) = V$) we deduce $M \times |Lead| = V$ and hence M and $|Lead|$ divide V .

To summarize:

Lemma 1. *The simple protocol converges to a global state T such that there is an integer M and a set $Lead$ of agents such that for all $a \in Lead$ we have $T(a) = M$ and for all $a \in \mathcal{A} \setminus Lead$ we have $T(a) = 0$.*

The set $Lead$ may be considered as the set of the leaders and:

Lemma 2. *The simple protocol is a leader election if and only if $\max\{v_a \in \mathcal{A}\} > \text{gpd}(V)$. Moreover the state of the leader is equal to V .*

In particular, if V is prime and there are at least two agents a and b such that $v_a \neq v_b$, then $Lead$ is a singleton and hence the simple protocol is a leader election.

4.2 Message-Terminating Leader Election in a Ring with Homonyms

Assuming that the number of identifiers is strictly larger than the greatest proper divisor of the ring size, we propose a message-terminating leader election in a ring of unknown size with homonyms. The algorithm is described in Fig. 2.

Consider a ring of n processes with homonymous processes and let \mathcal{L} be the set of identifiers. Assuming that the ring is oriented, in the following, next (resp. previous) will denote the next process (resp. previous process) for the clockwise direction.

We first give some intuitions about the algorithm. Roughly speaking, the algorithm realizes in parallel leader elections for each set of processes having the same identifier. These elections are based on the same principles as in the simple protocol and when the election converges the chosen processes have an estimate est of the ring size. For each identifier i , at least one process (but perhaps several) is eventually elected, and eventually each leader for identifier l has the same estimate est of the ring size. Moreover, these estimates of the ring

```

var:  $Id, state = active, est = 0, Leader = false, \forall i \in \mathcal{L} \ L[i] = 0;$ 
1  procedure Compute()
2  begin
3    /* compute distance to the next active process with identifier  $Id^*$  */
4    send(right,  $Id$ ) to next
5    receive(ALeft,  $Id, v$ ) from next
6     $est = v$ 
7    /* compute distance to the previous active process with identifier  $Id^*$  */
8    send(left,  $Id$ ) to previous
9    receive(ARight,  $Id, v$ ) from previous
10    $lefttest := v$ 
11   if  $est < lefttest$  then
12      $state = passive$ 
13     send(restart) to next
14     send(restart) to previous
15   else if  $est = lefttest$  then send(publish,  $Id, est$ ) to next
16 end of procedure

initially
17 Compute()

messages
18 on receive(left,  $i$ ) from previous
19   if  $state = active \wedge i = Id$  then send(ARight,  $i, 1$ ) to previous
20   else send (left,  $i$ ) to next
21 on receive(right,  $i$ ) from next
22   if  $state = active \wedge i = Id$  then send(ALeft,  $i, 1$ ) to next
23   else send (right,  $i, v$ ) to next
24 on receive(ARight,  $i, v$ ) from previous  $\wedge (state = passive \vee i \neq Id)$ 
25   send(ARight,  $i, v + 1$ ) to next
26 on receive(ALeft,  $i, v$ ) from next  $\wedge (state = passive \vee i \neq Id)$ 
27   send(ALeft,  $i, v + 1$ ) to previous
28 on receive(restart)
29   Compute()
30 on receive(publish,  $i, v$ )
31 if  $L[i] < v$  then
32    $L[i] = v$ 
33   send(publish,  $i, v$ ) to next
34    $(*, j) = \max\{(L[j], j) | j \in \mathcal{L}\}$ 
35   if  $(j = Id) \wedge state = active$ 
36     then  $Leader = true$ 
37     else  $Leader = false$ 

```

Fig. 2. Message-terminating leader election algorithm for identifier Id .

size are strictly lower than n if more than one leader is elected, and is equal to n when there is only one leader for identifier i .

We now describe more precisely the leader election for processes with the same identifier i . A process is initially *active* and may become definitely *passive*.

Only an active process may be a leader. Each active process having i as identifier computes the distance to the next (Lines 3–6) and the previous (Lines 8–10) active process with the same identifier. This computation is easy: a process p with identifier i sends a message *right* and a message *left* to respectively its successor and its predecessor on the ring. These messages are retransmitted (in the same direction) by all passive processes or processes with identifier different from i . When the first active process with identifier i receives one of these messages it sends back a message (*Aleft*, respectively *Arigh*t) with a counter to the initiator of the message. The counter is incremented in each node that retransmits the message until the message reaches p . When received by p , the counter associated with a message *Aleft* gives the distance between p and the next active process with identifier i . This distance is recorded by p in its variable *est*. In the same way, p will get the distance to its previous active process and stores it in variable *leftest*.

If the estimate *est* is strictly lower than *leftest* then p becomes passive (Line 11) and sends message *restart* to next and previous active processes with identifier i (Lines 13 and 14) these messages lead to a computation of the distances for these processes (Lines 28–29).

The point is here that the behaviour of the nodes emulates the simple protocol. For this, considering the ring with the processes having the same identifier i , and let the state s_p of p for the simple protocol be the value of *est* for p . We obtain an emulation of the simple protocol. Indeed, as *est* is the distance to the next process with identifier i , the sum of all the s_p is equal to n . When a process becomes passive we may consider that it sets s_p to 0 and that the previous active process q updates s_q to $s_q + s_p$. Hence the moves of the algorithm emulate the rule R of the simple algorithm.

From Lemma 1 the election for identifier i stabilizes to a set *Lead* of processes that we call leaders for identifier i , of agents having the same estimate *est*. Moreover, if the cardinality of \mathcal{L} is greater than the greatest proper divisor of n , for at least one identifier, say l , from Lemma 2 the leader election for identifier i eventually gives only one leader for i . It remains to choose one identifier among the identifiers leading to one leader. This is done by choosing the identifier whose the estimate is the biggest (using order of identifiers if more than one identifier is the biggest). Then a process leader with this identifier considers itself as the global leader (variable *Leader* Lines 30–37).

An active process with identifier i may be a leader for this identifier only if the values *est* and *leftest* are the same (else the election is not “stabilized” for this identifier). To collect information about leaders for each identifier, when *est* equals to *leftest*, an active process with identifier i sends a message *publish* with its estimation of the ring size (Line 15) that will circulate on the ring (in such a way that at most n messages are generated). All active processes in the ring maintain an array L containing for each identifier the best estimation of the ring size. Then each active process chooses (s, i) as the max among all the $(T[j], j)$ (ordered by $(a, b) < (a', b')$ if and only if $a < a'$ or $a = a'$ and $b < b'$) and considers itself as the (global leader) if its identifier is i .

Proposition 2. *If $|\mathcal{L}| > \text{gpd}(n)$, algorithm of Fig. 2 is a message-terminating election algorithm with homonyms.*

Proof. We only give the main steps of the proof.

Adapting the proof of Lemma 1 we get the main Lemma:

Lemma 3. *For each identifier i , there is a non empty set L_i of processes with identifier i and there is $v > 0$ such that (1) there is a time after which L_i is the set of active processes with identifier i , (2) v is greater or equal the maximum of distance on the ring between processes with identifier i , and (3) $n = v \times |L_i|$.*

Under the condition of this previous lemma, we say that the election on identifier i converges to L_i with estimate v for the size of the ring.

Lemma 4. *If the election for identifier i converges to estimate e then for any estimate est of processes with identifier i , $e \leq est$.*

If the election for identifier i converges to set L_i and $p \in L_i$, p eventually sends a *publish* message. As *publish* messages reach all processes with Lemma 4 we get:

Lemma 5. *If the election for identifier i converges to estimate e then eventually at each process $L[i] = e$.*

We say that process p chooses identifier i as leader if after Line 30 for some x $(e, j) = (x, i)$. Call a process with variable *Leader* equal to true a winner, and a process with variable *Leader* equal to false a loser.

Lemma 6. *Eventually all processes choose the same identifier i as leader. Then eventually all processes with identifiers different from i are forever loser and only active processes with identifiers equal to i are forever winner.*

Lemma 7. *If $|\mathcal{L}| > \text{gpd}(n)$, then for at least one identifier i , there are p and q with identifier i such that the distance from p to q is greater than $\text{gpd}(n)$.*

Proof. Indeed let $G(i)$ be the set of processes with identifier i and $d_i(p)$ be the distance on the ring to the next process with identifier i (if the next such process is p itself $d_i(p) = n$) clearly $\sum_{p \in G(i)} d_i(p) = n$.

Let $M = \{i \in \mathcal{L} \mid \max\{d_i(p) \mid p \in G(i)\} > \text{gpd}(n)\}$. By contradiction assume that $M = \emptyset$, then for all $i \in \mathcal{L}$: $\sum_{p \in G(i)} d_i(p) = n \leq \text{gpd}(n) \cdot |G(i)|$. Thus

$$\text{gpd}(n) \cdot n = \text{gpd}(n) \cdot \left(\sum_{i \in \mathcal{L}} |G(i)| \right) = \sum_{i \in \mathcal{L}} \text{gpd}(n) \cdot |G(i)| \geq |\mathcal{L}|n$$

Hence $\text{gpd}(n) \geq |\mathcal{L}|$ a contradiction.

Then from Lemma 3:

Lemma 8. *If $|\mathcal{L}| > \text{gpd}(n)$ then for at least one $i \in \mathcal{L}$ the election converges to a singleton with estimate n .*

From Lemmas 8 and 6, we deduce that eventually only one process is winner and all other processes are looser. A simple verification enables to prove that only a finite number of messages are exchanged, finishing the proof of the proposition.

If n , the ring size, is known algorithm of Fig. 2 may easily be changed to get a process-terminating leader election algorithm. Essentially for each identifier active processes verifies that there is only one leader for the identifier by sending a message *Probe*. Details are given in Fig. 3.

```

/* replace Line 15 of Algorithm of Figure 2 */
if ( $est = leftist$ ) then
  if  $est = n$  then send( $publish, Id, n$ )
  else send( $Probe, Id, est, true, 1$ ) to next

/* replace Line 30 to 37 of Algorithm of Figure 2 */
on receive( $publish, i, v$ )
   $L[i] = v$ 
  if  $\forall k \in \mathcal{L} \ L[k] \neq 0$  then
    if ( $Id = \max\{j \in \mathcal{L} | L[j] = n\} \wedge (state = active)$ ) then
       $Leader = true$ 
      send( $Terminate$ )
      terminate
    send( $publish, i, v$ ) to next

/* code for messages Terminate */
on receive( $Terminate$ )
   $Leader = False$ 
  send( $Terminate$ )
  terminate

/* code for messages Probe */
on receive( $Probe, i, v, b, c$ )
  if ( $c = n \wedge (b = true)$ ) then send( $publish, i, v$ )
  else if ( $c < n$ ) then
    if ( $(Id \neq i) \vee (state = passive)$ ) then send( $Probe, i, v, b, c + 1$ ) to next
  else
    if  $est = v$  then send ( $Probe, l, v, b, c + 1$ )
    else send ( $Probe, i, v, false, c + 1$ )

```

Fig. 3. From message-terminating to process-terminating.

Proposition 3. *If $|\mathcal{L}| > gpd(n)$ and n is known by processes, algorithm of Fig. 3 is a process-terminating election algorithm with homonyms.*

Remarks. It is only for simplify the code that we refer in the algorithm to \mathcal{L} . \mathcal{L} is used only in algorithm Fig. 2 for array L indexed by identifiers in Lines 30–34. Instead of array L it is possible to deal with an ordered list without a priori knowledge of the set of identifiers. Hence we get a deterministic message-terminating election algorithm with neither the knowledge of the ring size neither the knowledge of the number of identifiers.

In the same way, if $|\mathcal{L}| > \text{gpd}(n)$, if n is not a prime number, n is lower or equal to $|\mathcal{L}|^2$. With the knowledge of $|\mathcal{L}|$ we get a bound on n , it is easy to adapt the algorithm of Fig. 3 in such a way we obtain process-terminating election algorithm with only the knowledge of $|\mathcal{L}|$.

5 A Process Terminating Leader Election Algorithm with $O(n \times \log(n))$ Messages

5.1 Definitions

We encode in a multi-sequence, a number of processes and a set of identifiers. A *multi-sequence* is a sequence $((a_1, i_1), (a_2, i_2), \dots, (a_u, i_u))$, where for each $1 \leq h \leq u$, i_h is an identifier, a_h is the multiplicity of i_h , and $i_1 < i_2 < \dots < i_u$. For a multi-sequence $\mathcal{I} = ((a_1, i_1), (a_2, i_2), \dots, (a_u, i_u))$, $\text{Id}(\mathcal{I}) = \{i_1, i_2, \dots, i_u\}$ and $\text{nb}(\mathcal{I}) = a_1 + a_2 + \dots + a_u$. We define a total order on such multi-sequences. Given two sequences $\mathcal{I} = ((a_1, i_1), (a_2, i_2), \dots, (a_u, i_u))$ and $\mathcal{J} = ((b_1, j_1), (b_2, j_2), \dots, (b_v, j_v))$, we say that $\mathcal{I} < \mathcal{J}$ if

- it exists $k \leq \min\{u, v\}$ such that
 - for every $0 \leq h < k$: $i_{u-h} = j_{v-h}$ and $a_{u-h} = b_{v-h}$
 - $i_{u-k} < j_{v-k}$ or ($i_{u-k} = j_{v-k}$ and $a_{u-k} < b_{v-k}$)
- or
- $u < v$ and for every $0 \leq h < u$: $i_{u-h} = j_{v-h}$ and $a_{u-h} = b_{v-h}$

It can be easily verified that we have defined a total order between multi-sequence. In the following $\langle \rangle$ denotes the empty multi-sequence and \oplus is the operator adding an element to a multi-sequence.

5.2 Algorithm

Our leader election algorithm uses a similar approach to the ones in [15, 20], where processes have distinct identifiers. Recall that in their algorithms, each process is initially active. Processes that are passive, simply forward the message they receive. At each step, an active process sends a message with its identifier to its both neighbors and receives messages from its both nearest active neighbors.

If one of the received messages contains an identifier greater than its own identifier then it becomes passive. Otherwise, it starts a new step. If a process receives its own message (i.e. a message with its own identifier) then the process becomes leader. The algorithm terminates after $O(\log(n))$ steps, and each step requires $2n$ messages.

In our algorithm, we keep the idea that each process is initially active and, at each step, each remaining active process p will exchange messages with its two nearest active neighbors and it will determine if it becomes passive (as for the previous algorithm).

But to determine if a process p becomes passive we use not only the identifiers of its two nearest active neighbors q and s but also the distances $d(p, q)$ and $d(p, s)$ and the identifiers of the processes between p and q and p and s .

We encode these two last information in multi-sequence and we use the total order on these multi-sequences.

The algorithm ensures that p and q are active and neighbors then the information they get each other are the same: if p receives from its left $(Id(q), \mathcal{D}_{\mathcal{L}})$ then q receives from its right $(Id(p), \mathcal{D}_{\mathcal{R}})$ with $\mathcal{D}_{\mathcal{R}} = \mathcal{D}_{\mathcal{L}}$.

After the exchange of messages, p compares $(Id(q), \mathcal{D}_{\mathcal{L}})$ and $(Id(s), \mathcal{D}_{\mathcal{R}})$ that it received from its two nearest active neighbors. If the number of processes between itself and its neighbors is the size of the ring, this process is the only active process. Then it is the leader. It sends a message STOP in order that all passive processes terminate and it terminates. If the number of processes between itself and its neighbors is different from the ring size, the process becomes passive if (1) $Id(p) < Id(q)$ or $Id(p) < Id(r)$ or (2) if $Id(p) = Id(q) = Id(s)$ and $(\mathcal{D}_{\mathcal{R}} < \mathcal{D}_{\mathcal{L}}$ or $\mathcal{D}_{\mathcal{R}} = \mathcal{D}_{\mathcal{L}}$) (condition (2) is an adaptation for homonyms). Otherwise, it remains active and starts a new step.

If we consider three consecutive active processes by the previous rules at least one of them become passive. In each step, at least a third of active processes are hence eliminated. By repetitive application of these rules eventually it remains only at most one active process (the ring structure ensures that if two processes are active at least one will be eliminated). Thus after $O(\log(n))$ steps, it remains at most one process.

But we have to avoid that all processes become passive. The fact that $|\mathcal{L}|$ is greater than the greatest proper divisor of n ensures that it is impossible that there is a subset of processes with the same identifier such that the distance between them is the same and the set of identifiers of processes between them are the same (see Lemma 10).

Thus, the algorithm will finish with only one active process. The algorithm is described in more details in Fig. 4.

We give below the main steps of the proof of this theorem and we sketch their proofs. By abuse of language we say that a process is active (resp. passive) if its *status* variable is *active* (resp. *passive*). As channels are FIFO, a computation of an active process can be decomposed in (asynchronous) rounds. As long as a process is active and has not terminate, it executes a round: it waits to receive messages from the left and the right and computes its new state after receiving them. We name s_p^r the state of p after r rounds. Let s_p^0 denotes the initial state. If a process is terminated or passive in s_p^r then for all r' greater than $s_p^{r'} = s_p^r$.

Directly from the algorithm we have:

Lemma 9. *If process q is active at s_q^r and p is its the nearest active right neighbor (i.e. the nearest right neighbor p such that p is active in s_p^r) then if q receives $\mathcal{D}_{\mathcal{R}}^q$ from its right and p receives $\mathcal{D}_{\mathcal{L}}^p$ from its left then $\mathcal{D}_{\mathcal{R}}^q = \mathcal{D}_{\mathcal{L}}^p$.*

Lemma 10. *Let G be a subset of processes with the same identifier, there do not exist a set of identifiers \mathcal{I} and an integer x such that for any two processes p and q of G , there are x processes between p and q and the set of identifiers of processes between p and q is \mathcal{I} .*

Code for a process with identifier i

```

1  status = "active"
2  Leader = false
3  send( $i, <>, 1$ ) in both directions

4  for ever
5    if status = "passive" then
6      wait until received a message:
7      On receiving ( $j, \mathcal{D}$ ):
8        send( $j, \mathcal{D} \oplus < i >$ ) in the same direction

9      On receiving (STOP):
10       send(STOP) in the same direction; halt

11  if status = "active" then
12    wait until received a message from the left and the right:
13    assume that ( $j_R, \mathcal{D}_R$ ) came from right and ( $j_L, \mathcal{D}_L$ ) came from left
14    if  $nb(\mathcal{D}_R) = n$  then Leader = true; send(STOP) to left; halt
15    else if ( $i < j_R$  or  $i < j_L$ ) or ( $i = j_R = j_L$  and ( $\mathcal{D}_R < \mathcal{D}_L$  or  $D_R = D_L$ ))
16      then status = "passive"
17    else send( $i, <>, 1$ ) in both directions

```

Fig. 4. $O(n \log(n))$ algorithm for process terminating leader election

Proof. Assume that there exists a subset G of processes with the same identifier, a set of identifier \mathcal{I} and an integer x such that for any two processes p and q of G , there are x processes between p and q and the set of identifiers of processes between p and q is \mathcal{I} . Note that $x \geq |\mathcal{I}|$. We have $n = |G| * x$, then (1) x is a divisor of n . As for each identifier, the ring contains at least one process with this identifier, we get $\mathcal{I} = \mathcal{L}$.

Then $x \geq gpd(n)$, with (1) this implies that G contains only one process.

Lemma 11. *For all r , there is at least one process p active in s_p^r .*

Proof. We show this property by induction, it holds trivially for $r = 0$. Assume it is true until $r - 1$. Assume for contradiction that all processes become passive in r . Let G be the set of processes that changes its state from active to passive in round r .

If G contains at least two processes with a different identifier, we consider the subset H of G that contain the processes with the greatest identifier. As there is a process in G with another identifier, there is at least one process in H that has from right active neighbor a process in $H \setminus G$. From condition line 15, this process remains active.

If all processes in G have the same identifier, by Lemma 10, for at least one process we have $\mathcal{D}_L > \mathcal{D}_R$. From condition Line 15, this process remains active.

Lemma 12. *For $r > 0$, let $x = |\{p|p \text{ is active in } s_p^{r-1}\}|$, we have $|\{p|p \text{ is active in } s_p^r\}| \leq (2/3) * x + x \bmod 3$.*

Proof. Consider the set $\{p|p \text{ is active in } s_p^{r-1}\}$ and $x = |\{p|p \text{ is active in } s_p^{r-1}\}|$. We split this set in sets of three processes that are neighbors in the ring (it remains $x \bmod 3$ processes). Consider some sets of three processes $\{p, q, v\}$ and assume that p is the left neighbor of q and x its right neighbor.

If p and q or q and v have not the same identifier then the process with the smallest identifiers become passive. Let \mathcal{D}_R^q and \mathcal{D}_L^q the values that it receives from its nearest active neighbors. If they have the same identifier, q remains active only if $\mathcal{D}_R^q > \mathcal{D}_L^q$ but in this case by Lemma 9, p becomes passif.

Lemma 13. *For $r > 0$, let $|\{p|p \text{ is active in } s_p^{r-1}\}| = 2$, we have $|\{p|p \text{ is active in } s_p^r\}| = 1$.*

Proof. It remains two active processes, let p and q be these processes.

If p and q have not the same identifier then the process with the smallest identifiers become passive. Let \mathcal{D}_R^q and \mathcal{D}_L^q the values that it receives from its nearest active neighbors. If p and q have the same identifier, q remains active only if $\mathcal{D}_R^q > \mathcal{D}_L^q$ but in this case by Lemma 9, p becomes passive.

From Lemmas 11, 12 and 13, we deduce that eventually it remains only one active process. Directly from the algorithm this process becomes leader and propagates the termination by sending a STOP messages.

Theorem 4. *If n is known by processes and $|\mathcal{L}| > \text{gpd}(n)$ then there exists an algorithm that solves process terminating leader election with $O(n \log(n))$ messages.*

6 Concluding Remarks

Given a ring of size n , $\text{gpd}(n)$ identifiers are needed to have a deterministic election algorithm. If n is a prime number then two identifiers are sufficient, but in the worst case when n is even $n/2$ (hence $O(n)$) identifiers are needed. Moreover, if there is a deterministic solution for leader election, then we do not have any penalty: there is an election with the same order of number of exchanged messages as when all processes have their own identity. Then limiting the number of identifiers without avoiding the existence of deterministic election algorithm is mainly interesting if some properties of the size ring are known (e.g. this size is a prime number).

The leader election problem has been studied in framework similar to homonyms: [5, 7, 8, 22, 23] consider “partially anonymous”, [14] “nonunique label” and [18] “partially eponymous”. Some of the results in these papers are similar to the ones we present here but we present our results, restricted to the ring topology, in term of number of process identifiers rather than properties of the general communication graph. Even if [14] concerns rings with asynchronous communication, our algorithm, with our conditions on $|\mathcal{L}|$ gets the optimality in term of messages that is not achieved in a more general framework by [14].

References

1. Abrahamson, K., Adler, A., Gelbart, R., Higham, L., Kirkpatrick, D.: The bit complexity of randomized leader election on a ring. *SIAM J. Comput.* **18**(1), 12–29 (1989)
2. Angluin, D.: Local and global properties in networks of processors (extended abstract). In: *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing, STOC '80*, pp. 82–93. ACM, New York (1980)
3. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. *Distrib. Comput.* **18**(4), 235–253 (2006)
4. Arévalo, S., Anta, A.F., Imbs, D., Jiménez, E., Raynal, M.: Failure detectors in homonymous distributed systems (with an application to consensus). In: *ICDCS*, pp. 275–284 (2012)
5. Boldi, P., Shammah, S., Vigna, S., Codenotti, B., Gemmell, P., Simon, J.: Symmetry breaking in anonymous networks: characterizations. In: *ISTCS*, pp. 16–26 (1996)
6. Burns, J.E., Pachl, J.K.: Uniform self-stabilizing rings. *ACM Trans. Program. Lang. Syst.* **11**(2), 330–344 (1989)
7. Chalopin, J., Godard, E., Métivier, Y.: Election in partially anonymous networks with arbitrary knowledge in message passing systems. *Distrib. Comput.* **25**(4), 297–311 (2012)
8. Chalopin, J., Métivier, Y., Morsellino, T.: Enumeration and leader election in partially anonymous and multi-hop broadcast networks. *Fundam. Inform.* **120**(1), 1–27 (2012)
9. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) *EUROCRYPT 1991*. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
10. Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Kermarrec, A.-M., Ruppert, E., Tran-The, H.: Byzantine agreement with homonyms. In: *PODC*, pp. 21–30. ACM (2011)
11. Delporte-Gallet, C., Fauconnier, H., Tran-The, H.: Byzantine agreement with homonyms in synchronous systems. In: Bononi, L., Datta, A.K., Devismes, S., Misra, A. (eds.) *ICDCN 2012*. LNCS, vol. 7129, pp. 76–90. Springer, Heidelberg (2012)
12. Delporte-Gallet, C., Fauconnier, H., Tran-The, H.: Homonyms with forgeable identifiers. In: Even, G., Halldórsson, M.M. (eds.) *SIROCCO 2012*. LNCS, vol. 7355, pp. 171–182. Springer, Heidelberg (2012)
13. Dijkstra, E.W.: Self-stabilizing systems in spite of distributed control. *Commun. ACM* **17**(11), 643–644 (1974)
14. Dobrev, S., Pelc, A.: Leader election in rings with nonunique labels. *Fundam. Inform.* **59**(4), 333–347 (2004)
15. Franklin, R.: On an improved algorithm for decentralized extrema finding in circular configurations of processors. *Commun. ACM* **25**(5), 336–337 (1982)
16. Huang, S.-T.: Leader election in uniform rings. *ACM Trans. Program. Lang. Syst.* **15**(3), 563–573 (1993)
17. Kuten, S., Pandurangan, G., Peleg, D., Robinson, P., Trehan, A.: Sublinear bounds for randomized leader election. In: Frey, D., Raynal, M., Sarkar, S., Shyamasundar, R.K., Sinha, P. (eds.) *ICDCN 2013*. LNCS, vol. 7730, pp. 348–362. Springer, Heidelberg (2013)

18. Mavronicolas, M., Michael, L., Spirakis, P.G.: Computing on a partially eponymous ring. *Theor. Comput. Sci.* **410**(6–7), 595–613 (2009)
19. Pachl, J.K., Korach, E., Rotem, D.: Lower bounds for distributed maximum-finding algorithms. *J. ACM* **31**(4), 905–918 (1984)
20. Peterson, G.L.: An $o(n \log n)$ unidirectional algorithm for the circular extrema problem. *ACM Trans. Program. Lang. Syst.* **4**(4), 758–762 (1982)
21. Xu, Z., Srimani, P.K.: Self-stabilizing anonymous leader election in a tree. In: *IPDPS*. IEEE Computer Society (2005)
22. Yamashita, M., Kameda, T.: Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Trans. Parallel Distrib. Syst.* **10**(9), 878–887 (1999)
23. Yamashita, M., Kameda, T.: Modeling k -coterie by well-covered graphs. *Networks* **34**(3), 221–228 (1999)

Networked Systems

Second International Conference, NETYS 2014,
Marrakech, Morocco, May 15-17, 2014. Revised
Selected Papers

Noubir, G.; Raynal, M. (Eds.)

2014, XV, 348 p. 110 illus., Softcover

ISBN: 978-3-319-09580-6