

Analysis of Fully Distributed Splitting and Naming Probabilistic Procedures and Applications

Y. Métivier, J.M. Robson, and A. Zemmari

*Université de Bordeaux, Bordeaux INP
LaBRI, UMR CNRS 5800
351 cours de la Libération, 33405 Talence, France
{metivier, robson, zemmari}@labri.fr*

Abstract

This paper proposes and analyses two fully distributed probabilistic splitting and naming procedures which assign a label to each vertex of a given anonymous graph G without any initial knowledge. We prove, in particular, that with probability $1 - o(n^{-1})$ (resp. with probability $1 - o(n^{-c})$ for any $c \geq 1$) there is a unique vertex with the maximal label in the graph G having n vertices. In the first case, the size of labels is $O(\log n)$ with probability $1 - o(n^{-1})$ and the expected value of the size of labels is also $O(\log n)$. In the second case, the size of labels is $O((\log n)(\log^* n)^2)$ with probability $1 - o(n^{-c})$ for any $c \geq 1$; their expected size is $O((\log n)(\log^* n))$.

We analyse a basic simple maximum broadcasting algorithm and prove that if vertices of a graph G use the same probabilistic distribution to choose a label then, for broadcasting the maximal label over the labelled graph, each vertex sends $O(\log n)$ messages with probability $1 - o(n^{-1})$.

From these probabilistic procedures we deduce Monte Carlo algorithms for electing or computing a spanning tree in anonymous graphs without any initial knowledge and for counting vertices of an anonymous ring; these algorithms are correct with probability $1 - o(n^{-1})$ or with probability $1 - o(n^{-c})$ for any $c \geq 1$. The size of messages has the same value as the size of labels. The number of messages is $O(m \log n)$ for electing and computing a spanning tree; it is $O(n \log n)$ for counting the vertices of a ring. These algorithms can be easily extended to also ensure for each vertex v an error probability bounded by ϵ_v ; the error probability ϵ_v is decided by v in a totally decentralised way.

We illustrate the power of the splitting procedure by giving a probabilistic election algorithm for rings having n vertices with identities which is correct and always terminates; its message complexity is equal to $O(n \log n)$ with probability $1 - o(n^{-1})$.

Keywords: Monte Carlo algorithm, spanning tree computation, counting, election algorithm, probabilistic analysis, splitting and naming.

1. Introduction

1.1. The problem

We consider anonymous, and, more generally, partially anonymous networks: unique identities are not available to distinguish the processes (or we cannot guarantee that they are distinct). We do not assume any global knowledge of the network, not even its size or an upper bound on its size. The processes have no knowledge on position or distance.

In this context, solutions for classical distributed problems, such as the construction of spanning trees, counting or election, must use probabilistic algorithms. This paper presents and studies splitting and naming procedures which provide solutions to these problems.

The question of anonymity is often considered when processes must not divulge their identities during execution, due to privacy concerns or security policy issues [GR05]. In addition, each process may be built in large scale quantities from which it is quite infeasible to ensure uniqueness. Therefore, each process must execute the same finite algorithm in the same way, regardless of its identity, as explained in [AAER07].

1.2. The Model

Our model is the usual asynchronous message passing model [AW04, Tel00]. A network is represented by a simple connected graph $G = (V(G), E(G)) = (V, E)$ where vertices correspond to processes and edges to direct communication links.

Each process can distinguish different incident edges, i.e., for each $u \in V$ there exists a bijection between the neighbours of u in G and $[1, \deg_G(u)]$ (where $\deg_G(u)$ is the number of neighbours of u in G). The numbers associated by each vertex to its neighbours are called *port-numbers*.

Each process v in the network represents an entity that is capable of performing computation steps, sending messages via some ports and receiving any message via some port that was sent by the corresponding neighbour. We consider asynchronous systems, i.e., each computation may take an unpredictable (but finite) amount of time. Note that we consider only reliable systems: no fault can occur on processes or communication links.

In this model, a distributed algorithm is given by a local algorithm that all processes should execute; thus all processes having the same degree have the same algorithm. A local algorithm consists of a sequence of computation steps interspersed with instructions to send and to receive messages.

As Tel [Tel00] (p. 71), we define the time complexity by supposing that internal events need zero time units and that the transmission time (i.e., the time between sending and receiving a message) is at most one time unit. This corresponds to the number of rounds needed by a synchronous execution of the algorithm.

A probabilistic algorithm is an algorithm which makes some random choices based on some given probability distributions; non-probabilistic algorithms are called deterministic.

A distributed probabilistic algorithm is a collection of local probabilistic algorithms. Since our networks are anonymous, if two processes have the same degree their local probabilistic algorithms are identical and have the same probability distribution.

A Las Vegas algorithm is a probabilistic algorithm which terminates with a positive probability (in general 1) and always produces a correct result.

A Monte Carlo algorithm is a probabilistic algorithm which always terminates; nevertheless the result may be incorrect with a certain probability.

Let \mathcal{A} be a distributed algorithm. Let G be a network. A configuration is defined by the states of all processes and the states of all communication links. A terminal configuration is a configuration in which no further steps of \mathcal{A} are applicable (see [Tel00], Chapter 8).

Distributed algorithms presented in this paper are message terminating: algorithms reach a terminal configuration and processes are not aware that the computation has terminated. We speak of process termination if, when algorithms reach a terminal configuration, processes are in a terminal state (a state in which there is no event of the process applicable).

Some results on graphs having n vertices are expressed with high probability, meaning with probability $1 - o(n^{-1})$ (w.h.p. for short) or with very high probability, meaning with probability $1 - o(n^{-c})$ for any $c \geq 1$ (w.v.h.p. for short).

We recall that $\log^* n = \min\{i \mid \log^{(i)} n \leq 2\}$, where $\log^{(1)} n = \log n$ and $\log^{(i+1)} n = \log(\log^{(i)} n)$.

Unless otherwise noted, all logarithms through the paper are to base 2. As usual, the natural logarithm is denoted \ln

1.3. Our Contribution

Let $G = (V, E)$ be an anonymous connected graph having n vertices. We assume no knowledge on G .

In the first part of this paper, we propose and analyse the following procedure by which each vertex builds its label. Each vertex v of G draws a bit b_v uniformly at random. Let t_v be the number of random draws of b_v on the vertex v until $b_v = 1$; it is called the lifetime of the vertex v . Each vertex v uses its lifetime to draw at random a number id_v in the set $\{0, \dots, 2^{t_v+3\log(t_v)} - 1\}$; finally, v is labelled with the couple (t_v, id_v) .

Let T be the maximal value in the set $\{t_v \mid v \in V(G)\}$. We prove that w.h.p.:

$$\log n - \log(2 \ln n) < T < 2 \log n + \log^* n.$$

We prove that, w.h.p., there exists exactly one vertex v such that $t_v = T$ and $id_v > id_w$ for any vertex w different from v such that $t_w = T$. The size of labels is $O(\log n)$ w.h.p. and the expected value of the size of labels is also $O(\log n)$.

We also prove that w.v.h.p.: $\frac{1}{2} \log n < T < (\log^* n) \log n$. If each vertex v draws id_v uniformly at random in the set: $\{0, \dots, 2^{t_v \log^* t_v} - 1\}$ then, w.v.h.p., there exists exactly one vertex v such that $t_v = T$ and $id_v > id_w$ for any

vertex w different from v such that $t_w = T$. In this case the size of labels is $O((\log n)(\log^* n)^2)$ w.v.h.p.; their expected size is $O((\log n)(\log^* n))$.

We analyse a very simple maximum broadcasting algorithm and prove that if vertices of a graph G use the same distribution to choose their label then, for broadcasting the maximal label over G , each vertex sends $O(\log n)$ messages w.h.p.

In the second part of this paper, we apply these procedures to classical problems, in anonymous graphs without any knowledge, such as spanning tree construction, counting the number of vertices of a ring or electing. In this way, we obtain:

- Monte Carlo spanning tree algorithms correct w.h.p. (resp. w.v.h.p.),
- Monte Carlo counting ring size algorithms correct w.h.p. (resp. w.v.h.p.),
- Monte Carlo election algorithms correct w.h.p. (resp. w.v.h.p.).

The size of messages used by these algorithms is the same as the size of labels generated by splitting and naming procedures. We prove also that the message complexities (the number of messages through the graph) is $O(m \log n)$ for the spanning tree computation and election; it is $O(n \log n)$ for counting the vertices of a ring.

We illustrate the power of the splitting procedure by giving a probabilistic election algorithm for ring graphs with identities which is correct and always terminates; its message complexity is equal to $O(n \log n)$ w.h.p. (where n is the number of vertices).

To finish, we explain how to obtain Monte Carlo algorithms which solve the problems discussed above w.h.p. (or w.v.h.p.) and which ensure for each vertex v an error probability bounded by ϵ_v where ϵ_v is determined by v in a fully decentralised way. To be more precise, these algorithms ensure an error probability bounded by ϵ where ϵ is the smallest value among the set of error probabilities determined independently by each vertex.

1.4. Related Work: Comparisons and Comments

Chapter 9 of [Tel00] and [Lav95] give a survey of what can be done and of impossibility results in anonymous networks. In particular, no deterministic algorithm can elect (see Angluin [Ang80], Attiya et al. [ASW88] and Yamashita and Kameda [YK88]); furthermore, with no knowledge on the network, there exists no Las Vegas election algorithm [IR90]. In fact, [IR90] proves that, in this context, the best we can achieve for electing, counting or spanning tree computing are message terminating Monte Carlo algorithms; there are no such algorithms which are process terminating.

Message terminating Monte Carlo election algorithms for anonymous graphs without knowledge are presented in [IR90, AM94, SS94].

The idea that each vertex draws at random a bit until it gets 1 (or 0) is used in two different contexts in [Pro93, FMS96, KMW11] and in [AM94].

In [Pro93, FMS96, KMW11], typically, as explained in [FMS96], processors have identification numbers and know the number of processors. A group of n processors play a game to identify a winner by tossing fair coins; the winner is the elected vertex. All processors that throw heads are eliminated; those that throw tails remain candidate and flip their coins again. The process is repeated among candidate winners until a single winner is identified. If at any stage all remaining candidate winners throw heads then all remaining players participate again as candidate winners in the next round of coin tossing. The main parameters studied are the number of rounds till termination and the total number of coin flips. Finally, the more general idea for breaking symmetry using a "logarithmic" trick appears also in radio network protocols as for example for the wakeup problem in [JS02] or for the MIS problem in [MW05]. They each use probabilities of broadcasting that vary with time so that, in a "reasonable" time, they reach values which are "good" for the actual n .

In the case where no knowledge on the network is available, Monte Carlo election algorithms presented in [AM94, SS94] are correct with probability $1 - \epsilon$, where ϵ is fixed and known to all vertices. In [AM94] executing the algorithm, presented in Section 2 (Networks with unknown size), each vertex tosses a fair coin until it gets a head for the first time. The number of these tosses is used only to have a small number of vertices which compete for the election. To obtain its number, each vertex v selects at random an element id_v of $[1, \dots, d]$ where $d = 36 \log 4r$, $r = 1/\epsilon$ and ϵ is fixed, given a priori (see [AM94], p. 315 Fig. 2.). Let $M = \max\{id_v | v \in V\}$, then there is a unique vertex u such that t_u is maximal and $id_u = M$ with probability greater than or equal to $1 - \epsilon$. The expected size of each message is $O(\log \log n + \log \epsilon^{-1})$. (The idea of eliminating some vertices before the election appears also in [RFJ⁺07]; the goal is to reduce the number of messages for the election.)

The algorithm presented in [SS94], maintains a rooted spanning forest of G . Each vertex belongs to a tree (initially it is alone). In the course of the algorithm trees expand by merging with adjacent trees. The level of a tree T , denoted $Level(T)$, is the integer part of the (base two) logarithm of the estimated number of nodes of T . The label of T (and thus of its root) is redrawn from the domain $d2^{level(T)}$, where $d = \lceil 2/\epsilon \rceil$ and ϵ is fixed ([SS94] p. 90). Upon termination, the forest consists of one tree with probability $1 - \epsilon$; the root of this tree is taken to be the elected vertex. The size of messages is $O(\log(n/\epsilon))$.

A message terminating Monte Carlo counting algorithm for rings is presented in [IR90]. Each vertex generates messages formed by: an estimation (initially 2) of the size of the ring and a random bit. Next the vertex sends the message along the ring. If a vertex receives a message which indicates that the estimation is not correct then it increments its estimation and generates another message. Finally, for each ϵ , [IR90] provides a counting algorithm correct with probability $1 - \epsilon$.

Tables 1 and 2 summarise the comparison of these various algorithms and those that are presented in this paper.

	Correct with probability	Message Complexity	Message size (number of bits)	Time
Afek-Matias	$1 - \epsilon$	Expected-value: $O(m(\log n)\epsilon^{-1} \log \epsilon^{-1})$ and $O(m(\log n)\alpha\epsilon^{-1} \log \epsilon^{-1})$ with probability $1 - n^{-\alpha} \forall \alpha$	Expected-value: $O(\log \epsilon^{-1} + \log(\log n))$, and $O(\log \epsilon^{-1} + \log(\log n) + \log \alpha)$ with probability $1 - n^{-\alpha} \forall \alpha$	$O(D)$
Schieber-Snir	$1 - \epsilon$	$O(m \log n)$	$O(\log n)$	$O(n)$
Elect-w.h.p. Section 7	w.h.p.	$O(m \log n)$ w.h.p.	$O(\log n)$ w.h.p., it is also the expected value	$O(D)$
Elect-w.v.h.p. Section 7	w.v.h.p.	$O(m \log n)$ w.h.p.	$O((\log n)(\log^* n)^2)$ w.v.h.p. and the expected value is $O((\log n)(\log^* n))$	$O(D)$

Table 1: Comparison of various Monte Carlo election algorithms for anonymous graphs without any initial knowledge (ϵ is a given constant known by each vertex, n is the number of vertices, m is the number of edges, D is the diameter of the graph).

	Correct with probability	Message Complexity	Message size (number of bits)	Time
Itai-Rodeh	$1 - \epsilon$	$O(n^3)$	$O(\log n)$	$O(n)$
Counting-ring-whp Section 6	w.h.p.	$O(n \log n)$ w.h.p.	$O(\log n)$ w.h.p., it is also the expected value	$O(n)$
Counting-ring-wvhp Section 6	w.v.h.p.	$O(n \log n)$ w.h.p.	$O((\log n)(\log^* n)^2)$ w.v.h.p. and the expected value is $O((\log n)(\log^* n))$	$O(n)$

Table 2: Probabilistic computation of the ring size (ϵ is a given constant known by each vertex and n is the number of vertices).

1.5. Summary

This paper is organised as follows. Sections 2 and 3 present and analyse two fully distributed splitting and naming probabilistic procedures which generate labels on vertices of graphs. Section 4 analyses a classical broadcasting algorithm. Sections 5, 6 and 7 present applications to spanning tree construction, counting and electing for anonymous graphs without any knowledge; more precisely they present Monte Carlo algorithms which solve these problems and which are correct w.h.p. or w.v.h.p. Section 8 applies the splitting procedure to obtain an efficient deterministic election algorithm for named rings. Section 9 explains how to modify previous procedures to ensure to each vertex v an upper bound of the error probability.

2. Analysis of a Splitting and Naming Probabilistic Procedure

This section presents and analyses a fully distributed probabilistic procedure, denoted Splitting-Naming-whp (see Algorithm 1), which assigns to each vertex v of a graph G a label (t_v, id_v) defined as follows.

A vertex v draws uniformly at random (u.a.r. for short) a bit b_v until $b_v = 1$. We denote by t_v the number of bits generated by the vertex v ; it is called the lifetime of v .

The number id_v is obtained by generating a number chosen u.a.r. in the set $\{0, \dots, 2^{t_v+3\log(t_v)} - 1\}$.

Algorithm 1: Procedure Splitting-Naming-whp(v).

```
1:  $t_v := 0$ 
2: repeat
3:   draw uniformly at random a bit  $b(v)$ ;
4:    $t_v := t_v + 1$ 
5: until  $b(v) = 1$ 
6: choose uniformly at random a number  $id_v$  in the set  $\{0, \dots, 2^{t_v+3\log(t_v)} - 1\}$ ;
```

We define the order, denoted $<$, on couples by: $(t_v, id_v) < (t'_v, id'_v)$ if:

- either $t_v < t'_v$
- or $t_v = t'_v$ and $id_v < id'_v$.

From now on, labels are couples that are ordered by $<$.

The sequel of this section analyses Procedure Splitting-Naming-whp(v), and more precisely:

1. the lifetime of each vertex,
2. the number of vertices that share the maximal lifetime,
3. the number of vertices that share the maximal lifetime and the same maximal number,
4. the size of the labels.

2.1. Analysis of the Maximal Number of Bits Drawn by each Vertex

We first analyse the value of t_v for any vertex v and the maximal value of t_v among vertices of the graph.

Any vertex has probability $1/2$ to draw the bit 1. Then t_v , for any vertex v , is a geometric random variable (r.v. for short) with parameter $1/2$. The maximal lifetime is simply $\max_{v \in V} t_v$, the maximum of n independent identically distributed (i.i.d. for short) geometric r.v. and hence:

Proposition 1. *Let $G = (V, E)$ be a graph having $n > 0$ vertices. We consider a run of Procedure Splitting-Naming-w.h.p. on vertices of G . Let T denote the maximal value of the set $\{t_v | v \in V\}$; T satisfies the following inequalities:*

1. $T < 2 \log n + \log^* n$ w.h.p.
2. $T > \log n - \log(2 \ln n)$ w.h.p.

PROOF. Initially the number of vertices is equal to n . Therefore after t random bit flips the expected number of vertices still alive is $\frac{n}{2^t}$. In particular after $2 \log n + \log^* n$ random bit flips it is $\frac{1}{n 2^{\log^* n}}$ so that the probability that any vertex remains is less than $\frac{1}{n 2^{\log^* n}}$. This proves the first claim.

On the other hand, for any $t > 0$, we have the following equality:

$$\begin{aligned} \Pr(T > t) &= \Pr(\exists v \in V \text{ s.t. } t_v > t) \\ &= 1 - \left(1 - \frac{1}{2^t}\right)^n. \end{aligned} \quad (1)$$

We also have the exponential approximation $(1 - a)^n \sim e^{-an}$ as $n \rightarrow \infty$, thus,

$$\Pr(T > t) \sim 1 - e^{-\frac{n}{2^t}}, \text{ as } n \rightarrow \infty. \quad (2)$$

Taking $t = \log n - \log(2 \ln n)$ in (2), we obtain:

$$\Pr(T > t) \geq 1 - o\left(\frac{1}{n}\right), \quad (3)$$

this proves the second claim which ends the proof. \square

Remark 1. Note that, in Proposition 1, the term $\log^* n$ can be replaced by any slowly-growing non bounded function $g(n)$.

From Proposition 1, we can deduce the following:

Corollary 1. *The expected value of T is equal to $\Theta(\log n)$.*

PROOF. At the end of each round, half of the vertices still active become inactive. Thus $\mathbb{E}(T)$, the expected value of T , is $O(\log n)$.

On the other hand, using the Markov inequality, we have:

$$\mathbb{E}(T) \geq (\log n - \log(2 \ln n)) \times \Pr(T > \log n - \log(2 \ln n)). \quad (4)$$

Then, using the second claim of Proposition 1, this proves that $\mathbb{E}(T) = \Omega(\log n)$. Which ends the proof. \square

2.2. *Analysis of the Number of Vertices that Share the Same Maximum Lifetime*

For any $t \geq 0$, we denote by X_t the number of vertices still alive at time t . For any $i > 0$, we have:

$$\Pr(X_{t+1} = 0 \mid X_t = i) = \frac{1}{2^i}. \quad (5)$$

This yields the following claim:

Claim 1.

$$\Pr(X_{t+1} = 0 \mid X_t \geq 2 \log n) \leq \frac{1}{n^2}. \quad (6)$$

Then, we obtain the following proposition:

Proposition 2. *The number of vertices which have the same maximum lifetime is, with high probability, at most $2 \log n$.*

PROOF. We consider the probability C_t that at time t , there are still more than $2 \log n$ vertices alive and that they all vanish at the next round. The sum of all C_t is the probability that there are more than $2 \log n$ vertices sharing the same maximum lifetime. We have:

$$\begin{aligned} C_t &= \Pr(\text{the number of alive vertices at time } t > 2 \log n) \\ &\quad \times \Pr(\text{all alive vertices finish} \mid \text{there are } > 2 \log n) \\ &\leq \Pr(\text{all alive vertices finish} \mid \text{there are } > 2 \log n). \end{aligned}$$

Then, using (6), we obtain:

$$C_t \leq 2^{-2 \log n} = n^{-2}. \quad (7)$$

On the other hand, we have:

$$\Pr(X_T \geq 2 \log n) \leq \Pr(T > 2 \log n) + \sum_{t \leq 2 \log n} C_t = o\left(\frac{1}{n}\right). \quad (8)$$

Which ends the proof. \square

2.3. *Analysis of the Number of Vertices that Have the same Maximum Lifetime and the same Maximum Number*

At the end of the initialisation phase, each vertex v obtains an integer t_v . Then it chooses u.a.r. a number

$$id_v \in \{0, \dots, 2^{t_v+3 \log(t_v)} - 1\}.$$

We have the following proposition :

Proposition 3. *With high probability, there exists a unique vertex v with label (t_v, id_v) such that for any $w \in V \setminus \{v\}$: $(t_v, id_v) > (t_w, id_w)$.*

PROOF. Let $S = \{v_1, \dots, v_k\}$ be the set of vertices that share the maximum lifetime. By Claim 2 of Proposition 1, w.h.p., any vertex v in S is such that $t_v > \log n - \log(2 \log n)$. Thus, each vertex v in S will choose u.a.r. a number id_v from a set which contains w.h.p. the set:

$$\left\{ 0, \dots, \frac{n}{2 \ln n} \times \left(\log \left(\frac{n}{2 \ln n} \right) \right)^3 - 1 \right\}.$$

If we denote $f(n) = \frac{n}{2 \ln n} \times \left(\log \left(\frac{n}{2 \ln n} \right) \right)^3$, then, using enumeration arguments, we show that the probability for v to be the unique vertex with the highest number is given by:

$$\begin{aligned} \Pr(id_v > id_w, \forall w \in S \setminus \{v\}) &\geq \sum_{i=1}^{f(n)-1} \frac{1}{f(n)} \times \left(\frac{i-1}{f(n)} \right)^{k-1} \\ &\sim \frac{1}{k} \left(1 - \frac{1}{f(n)} \right)^k \text{ as } n \rightarrow \infty. \end{aligned} \quad (9)$$

Thus, the probability that a unique vertex in S has the highest number is given by:

$$\Pr(\exists v \text{ s.t. } id_v > id_w, \forall w \in S \setminus \{v\}) \sim \left(1 - \frac{1}{f(n)} \right)^k \text{ as } n \rightarrow \infty. \quad (10)$$

Now, by Proposition 2, we have $k < 2 \log n$, w.h.p., thus :

$$\begin{aligned} \Pr(\exists v \text{ s.t. } id_v > id_w, \forall w \in S \setminus \{v\}) &\geq \left(1 - \frac{1}{f(n)} \right)^{2 \log n} \\ &\sim e^{-2 \frac{\log n}{f(n)}} \\ &= 1 - o\left(\frac{1}{n}\right), \end{aligned}$$

which ends the proof. □

2.4. Analysis of the Size of the Random Numbers

A vertex v chooses u.a.r. a number id_v from the set:

$$\left\{ 0, \dots, 2^{t_v+3 \log(t_v)} - 1 \right\}.$$

This implies that this random number has a size of at most $2 \log n + O(\log \log n)$ bits w.h.p. Furthermore, from Corollary 1 we deduce directly that the expected value of the size of id_v is $O(\log n)$.

3. Analysis of a Variant of the Splitting and Naming Probabilistic Procedure

This section presents and analyses a variant of Procedure Splitting-Naming-whp, it is denoted Splitting-Naming-wvhp (see Algorithm 2).

Now, the label of the vertex v is the couple (t_v, id_v) where t_v is still the lifetime of v . The difference is in the drawing of id_v : the number id_v is obtained by generating a number chosen u.a.r. in the set $\{0, \dots, 2^{t_v \log^* t_v} - 1\}$.

Algorithm 2: Procedure Splitting-Naming-wvhp(v).

- 1: $t_v := 0$
 - 2: **repeat**
 - 3: draw uniformly at random a bit $b(v)$;
 - 4: $t_v := t_v + 1$
 - 5: **until** $b(v) = 1$
 - 6: choose uniformly at random a number id_v in the set $\{0, \dots, 2^{t_v \log^* t_v} - 1\}$;
-

The remainder of the section is devoted to the analysis of Procedure Splitting-Naming-wvhp.

3.1. Analysis of the Maximal Number of Bits Drawn by each Vertex

Proposition 4. *Let $G = (V, E)$ be a graph with $n > 0$ vertices. We consider a run of Procedure Splitting-Naming-whp. Let T denote the maximal value of the set $\{t_v | v \in V\}$; T satisfies the following inequalities:*

1. $T < (\log n) \log^* n$. *w.v.h.p.*
2. $T > \frac{1}{2} \log n$ *w.v.h.p.*

PROOF. Taking $t = \frac{1}{2} \log n$ in (2), we obtain:

$$\begin{aligned} \mathbb{Pr} \left(T > \frac{1}{2} \log n \right) &\sim 1 - e^{-\sqrt{n}}, \text{ as } n \rightarrow \infty \\ &\sim 1 - o\left(\frac{1}{n^c}\right) \text{ for any } c \geq 1. \end{aligned} \quad (11)$$

On the other hand, after $(\log^* n) \log n$ rounds, the expected number of vertices still alive is $\frac{n}{n^{\log^* n}}$ so that the probability that any vertex remains is less than $\frac{1}{n^{\log^* n - 1}}$ which is $o\left(\frac{1}{n^c}\right)$ for any $c \geq 1$.

Remark 2. Note that, in Proposition 4, as in Proposition 1, the term $\log^* n$ can be replaced by any slowly-growing non bounded function $g(n)$.

3.2. *Analysis of the Number of Vertices that Have the same Maximum LifeTime and the same Maximum Number*

Proposition 5. *With very high probability, there exists a unique vertex v with the label (t_v, id_v) such that for any $w \in V \setminus \{v\}$: $(t_v, id_v) > (t_w, id_w)$.*

PROOF. Recall that id_v is obtained by generating a number chosen u.a.r. in the set $\{0, \dots, 2^{t_v \log^* t_v} - 1\}$.

Recall that $S = \{v_1, \dots, v_k\}$ is the set of vertices that share the maximum lifetime.

Thus, each vertex v in S chooses u.a.r. a number in a set which contains w.v.h.p. the set:

$$\left\{0, \dots, 2^{\left(\frac{\log n}{2}\right)(\log^* \left(\frac{\log n}{2}\right))} - 1\right\}.$$

The probability that at least two vertices, with the same maximal value $t_v = T$, obtain the same number satisfies:

$$\begin{aligned} \Pr(\exists u, v \in S \text{ s.t. } u \neq v \text{ and } id_u = id_v) &< \frac{n^2}{2^{\left(\frac{\log n}{2}\right)(\log^* \left(\frac{\log n}{2}\right))}} \\ &= o\left(\frac{1}{n^c}\right) \text{ for any } c \geq 1, \text{ as } n \rightarrow \infty. \end{aligned}$$

This ends the proof. □

3.3. *Analysis of the Size of the Random Numbers*

Proposition 6. *The size of numbers id_v is w.v.h.p. $O((\log n)(\log^* n)^2)$. Its expected value is $O((\log n)(\log^* n))$.*

PROOF. The first part is a direct consequence of Proposition 4 and of the choice of id_v in the set:

$$\left\{0, \dots, 2^{T \log^* T} - 1\right\}.$$

For the second part, let $S(id_v)$ denote the size of id_v , and let $S_{max} = \max_{v \in V} S(id_v) = T \log^* T$.

Then:

$$\begin{aligned} \mathbb{E}(S_{max}) &= \sum_{x \geq 1} \Pr(S_{max} \geq x) \\ &= \sum_{x=1}^{(\ln n)(\log^* n)} \Pr(S_{max} \geq x) + \sum_{x > (\ln n)(\log^* n)} \Pr(S_{max} \geq x), \end{aligned}$$

yielding:

$$\mathbb{E}(S_{max}) \leq (\ln n)(\log^* n) + \sum_{x \geq (\ln n)(\log^* (\ln n))} \Pr(S_{max} \geq x). \quad (12)$$

Then the second part of the sum in expression (12) satisfies:

$$\sum_{x \geq (\ln n)(\log^*(\ln n))} \Pr(S_{max} \geq x) = \sum_{t \geq \ln n} \sum_{y=t \log^* t}^{(t+1) \log^*(t+1)-1} \Pr(S_{max} \geq y) \quad (13)$$

On the other hand, for any y in the interval $[t \log^* t, (t+1) \log^*(t+1) - 1]$:

$$\begin{aligned} \Pr(S_{max} \geq y) &= \Pr(S_{max} \geq t \log^* t) \\ &= \Pr(T \geq t) \leq \frac{n}{2^t}, \end{aligned} \quad (14)$$

yielding:

$$\sum_{x \geq (\ln n)(\log^*(\ln n))} \Pr(S_{max} \geq x) \leq \sum_{t \geq \ln n} \frac{n((t+1) \log^*(t+1) - t \log^* t)}{2^t}. \quad (15)$$

Since $\log^*(t+1) \leq \log^* t + 1$, this gives:

$$\begin{aligned} \sum_{x \geq (\ln n)(\log^*(\ln n))} \Pr(S_{max} \geq x) &\leq \sum_{t \geq \ln n} \frac{n(t + \log^* t + 1)}{2^t} \\ &= O(\log n). \end{aligned} \quad (16)$$

Hence:

$$\begin{aligned} \mathbb{E}(S_{max}) &\leq (\ln n)(\log^* n) + O(\log n) \\ &= O((\log n)(\log^* n)), \end{aligned} \quad (17)$$

which ends the proof. \square

4. Analysis of the Message Complexity of a Maximum Broadcasting Algorithm

This section analyses the message complexity of a maximum broadcasting algorithm. More precisely, it analyses the number of messages exchanged in a labelled graph for broadcasting the maximal label. This analysis will be useful in next sections.

Let G be a graph and L a set of labels totally ordered by a relation $>$. Each vertex v of G chooses a label from L (all the vertices use the same distribution to choose a label), memorises its value in max_v and sends it to neighbours. If vertex v receives a label l greater than max_v then it memorises it in max_v and sends it to neighbours. The precise description of the algorithm is given in Algorithm 3 (initially, for each vertex v , $label_v$ is not defined).

We have the following proposition:

Proposition 7. *Let G be a graph and let L be a totally ordered set of labels. Each vertex v of G chooses a label from L ; all the vertices use the same distribution to choose a label. For each run of Algorithm Broadcasting-Max the number of messages sent by each vertex of G is w.h.p. $O(\log n)$.*

Algorithm 3: Broadcasting-Max.

```

I : {If  $label_v$  is not defined}
begin
  | choose at random  $label_v$  from  $L$ ;
  |  $max_v := label_v$ ;
  | send  $\langle label_v \rangle$  to all neighbours
end
B : {A Message ( $label$ ) has arrived at  $v$  through port  $p$ }
begin
  | if  $label_v$  is not defined then
  | | choose at random  $label_v$  from  $L$ ;
  | |  $max_v := label_v$ ;
  | | send  $\langle label_v \rangle$  to all neighbours
  | if  $label > max_v$  then
  | |  $max_v := label$ ;
  | | send  $\langle label \rangle$  to all neighbours except through port  $p$ 
end

```

PROOF. Let G be a graph having n vertices and v be a vertex of G . Let $max_v^{(i)}$ be the value of max_v at the i^{th} round of Algorithm Broadcasting-Max. By convention, $max_v^{(0)} = label_v$.

Let $L_i = \{l \in L | \exists w \in V, label_w = l \text{ and } l > max_v^{(i)}\}$ and let X_i denote the size of L_i . Clearly $L_0 \subseteq L$ and $X_0 = |L_0| \leq n$.

On the other hand, all the vertices in the graph have the same distribution for generating labels, then for any pair of vertices (u, w) , we have:

$$\begin{aligned}
 \Pr(max_u > max_w) &= \Pr(max_w > max_u) \\
 &= \Pr(max_u \neq max_w) / 2 \\
 &\leq 1/2.
 \end{aligned} \tag{18}$$

For any $i \geq 0$, let Y_i be the r.v. such that $Y_i = 1$ if v sends a message at round i and 0 otherwise.

We also define the sequence $(k_j)_{j \geq 0}$ as $k_0 = 1$ and for any $j \geq 0$, $k_{j+1} = i - i'$ where $i' = \sum_{x=0}^j k_x$, $Y_i = Y_{i'} = 1$ and for any $i' < l < i$, $Y_l = 0$.

For any $j > 0$, using (18), we have:

$$\mathbb{E}(X_{k_0 + \dots + k_j + k_{j+1}} | L_{k_0 + \dots + k_j}) \leq X_{k_0 + \dots + k_j} / 2. \tag{19}$$

Now, defining the r.v. $(Z_j)_{j \geq 0}$ by $Z_j = 2^j X_{k_0 + \dots + k_j}$ for any $j \geq 0$, we have:

$$\mathbb{E}(Z_{j+1} | L_{k_0 + \dots + k_j}) = 2^{j+1} \mathbb{E}(X_{k_0 + \dots + k_j + k_{j+1}} | L_{k_0 + \dots + k_j}). \tag{20}$$

Then, using (19) we obtain:

$$\mathbb{E}(Z_{j+1} | L_{k_0 + \dots + k_j}) \leq 2^j X_{k_0 + \dots + k_j} = Z_j. \tag{21}$$

Thus, the r.v. Z_j is a super-martingale (see [Wil93]), and then:

$$\mathbb{E}(Z_{j+1}) = \mathbb{E}(\mathbb{E}(Z_{j+1} | L_{k_0 + \dots + k_j})) \leq \mathbb{E}(Z_j). \tag{22}$$

A direct application of properties on supermartingales (see [Wil93], Chapter 10, p. 99) yields $\mathbb{E}(Z_i) \leq Z_0 \leq n$. Thus:

$$\mathbb{E}(X_{k_0+\dots+k_j}) = \frac{1}{2^j} \mathbb{E}(Z_j) \leq \frac{n}{2^j}. \quad (23)$$

Then, taking $j > j_0 = (3 + \varepsilon) \log n$ for any $\varepsilon > 0$, we obtain:

$$\mathbb{E}(X_{k_0+\dots+k_j}) \leq \frac{1}{n^{2+\varepsilon}}. \quad (24)$$

Thus, using the Markov inequality, this yields:

$$\mathbb{P}r(X_{k_0+\dots+k_j} > 1) \leq \frac{1}{n^{2+\varepsilon}}. \quad (25)$$

Hence, the probability that some vertex in the graph sends more than $(3+\varepsilon) \log n$ messages is $\frac{1}{n^{1+\varepsilon}}$ which is $o(\frac{1}{n})$. This ends the proof. \square

Corollary 2. *Let G be a graph of bounded degree having n vertices and let L be a totally ordered set of labels. Each vertex v of G chooses a label from L ; all the vertices use the same distribution to choose a label. Procedure Broadcasting-Max has a message complexity equal to $O(n \log n)$.*

5. A Monte Carlo Spanning Tree Algorithm Correct w.h.p. (resp. w.v.h.p.)

This section presents message terminating Monte Carlo algorithms, Algorithm 4 and a variant, for computing a spanning tree of an anonymous graph without any initial knowledge and with no distinguished vertex; they are correct w.h.p. (resp. w.v.h.p.).

Each vertex v is initially labelled with the label (t_v, id_v) generated by the probabilistic procedure Splitting-naming-whp (resp. Splitting-naming-wvhp). Each vertex attempts to build a tree considering that it is the root. If two vertices are competing to capture a third vertex w then w will join the tree whose root has the higher label; this label is indicated in max_v . The father of vertex v is indicated by the port number $father_v$ (by convention, if $root_v = true$ then $father_v = 0$). The children of v correspond to the set of port numbers $children_v$, neighbours of v which are neither children nor the father of v are indicated in the set of port numbers $other_v$.

Claim 2. *Let G be a graph. For each run of Spanning-Tree-whp, eventually, the network reaches a terminal configuration. The time complexity is $O(D)$, where D is the diameter of the graph.*

Let (t_{max}, id_{max}) be the maximal label of the vertices of G . When the network reaches a terminal configuration the set of $father_v$ encodes a spanning forest, the root of each tree is a vertex labelled (t_{max}, id_{max}) . If there is a unique vertex v such that $(t_v, id_v) = (t_{max}, id_{max})$ then the spanning forest is a spanning tree whose root is v .

Algorithm 4: Spanning-Tree-whp.

```
I : {If  $(t_v, id_v)$  is not defined}
begin
  call Splitting-Naming-whp(v);
   $max_v := (t_v, id_v)$ ;
   $root_v := true$ ;
   $father_v := 0$ ;  $other_v := \emptyset$ ;  $children_v := \emptyset$ ;
  send  $\langle (t_v, id_v) \rangle$  to all neighbours
end
D : {A Message  $(t, id)$  has arrived at  $v$  through port  $l$ }
begin
  if  $(t_v, id_v)$  is not defined then
    call Splitting-Naming-whp(v);
     $max_v := (t_v, id_v)$ ;
     $root_v := true$ ;
     $father_v := 0$ ;  $other_v := \emptyset$ ;  $children_v := \emptyset$ ;
    send  $\langle (t_v, id_v) \rangle$  to all neighbours
  if  $(t, id) > max_v$  then
     $max_v := (t, id)$ ;
     $root_v := false$ ;
     $father_v := l$ ;  $children_v := \emptyset$ ;  $other_v := \emptyset$ ;
    send  $\langle (parent, max_v) \rangle$  through  $l$ ;
    send  $\langle max_v \rangle$  to all neighbours except through  $l$ 
  else
    if  $(t, id) = max_v$  then
      send  $\langle (already, max_v) \rangle$  through  $l$ 
  end
end
P : {A Message  $\langle (parent, m) \rangle$  has arrived at  $v$  through port  $l$ }
begin
  if  $m = max_v$  then
    add  $l$  to  $children_v$ 
  end
end
A : {A Message  $\langle (already, m) \rangle$  has arrived at  $v$  through port  $l$ }
begin
  if  $m = max_v$  then
    add  $l$  to  $other_v$ 
  end
end
```

Remark 3. Algorithm Spanning-Tree-wvhp is obtained by substituting Splitting-Naming-wvhp to Splitting-Naming-whp in Algorithm Spanning-Tree-whp. Claim 2 is still valid for Algorithm Spanning-Tree-wvhp.

Concerning the analysis of these algorithms, results of previous sections imply:

Proposition 8. *Let G be a graph having n vertices and m edges. Algorithm Spanning-Tree-whp (resp. Algorithm Spanning-Tree-wvhp) computes a spanning tree w.h.p. (resp. w.v.h.p.). The size of messages of Algorithm Spanning-Tree-whp (resp. Spanning-Tree-wvhp) is $O(\log n)$ w.h.p., it is also the expected value (resp. $O((\log n)(\log^* n)^2)$ w.v.h.p. and the expected value is $O((\log n)(\log^* n))$). The message complexity of these two algorithms is w.h.p. $O(m \log n)$.*

6. A Monte Carlo Counting Algorithm for Rings Correct w.h.p. (resp. w.v.h.p.)

This section presents a message terminating Monte Carlo algorithm, Algorithm 5, for computing the size of an anonymous ring without any initial knowledge correct w.h.p. It also presents a variant correct w.v.h.p.

The main idea is very simple: each vertex v generates a label $label_v$ with the probabilistic procedure Splitting-naming-whp. Then v sends over the ring a message containing this label and a counter equal to 1. Each vertex v memorises in max_v the largest value it has received. A message is rejected if it is received by a vertex u such that max_u is greater than it. If not, the counter is incremented until the message is received by a vertex w having the same label as v . In this case w considers that the message is its own message and the value of the counter is the size of the ring; therefore it sends over the ring a new message to indicate this fact to each vertex which memorises this size and the associated label.

Algorithm 5: Counting-Ring-whp.

```

I : {If  $label_v$  is not defined}
begin
  | call Splitting-Naming-whp( $v$ );
  |  $label_v := (t_v, id_v)$ ;
  |  $max_v := label_v$ ;
  | send  $\langle (label_v, 1) \rangle$  to  $Next_v$ 
end
D : {A Message  $((t, id), s)$  has arrived at  $v$ }
begin
  | if  $label_v$  is not defined then
  | | call Splitting-Naming-whp( $v$ );
  | |  $label_v := (t_v, id_v)$ ;
  | |  $max_v := label_v$ ;
  | | send  $\langle (label_v, 1) \rangle$  to  $Next_v$ 
  | if  $(t, id) > max_v$  then
  | |  $max_v := (t, id)$ ;
  | |  $s := s + 1$ ;
  | | send  $\langle ((t, id), s) \rangle$  to  $Next_v$ 
  | else
  | | if  $(t, id) = max_v = (t_v, id_v)$  then
  | | |  $size_v := s$ ;
  | | | send  $\langle (RES, max_v, s) \rangle$  to  $Next_v$ 
  | end
end
R : {A Message  $(RES, m, s)$  has arrived at  $v$ }
begin
  | if  $label_v$  is not defined then
  | | call Splitting-Naming-whp( $v$ );
  | |  $label_v := (t_v, id_v)$ ;
  | |  $max_v := label_v$ 
  | | send  $\langle (label_v, 1) \rangle$  to  $Next_v$ 
  | if  $m > max_v$  then
  | |  $size_v := s$ ;
  | |  $max_v := m$ ;
  | | send  $\langle (RES, m, s) \rangle$  to  $Next_v$ 
end

```

Claim 3. *Let G be a ring. For each run of Counting-Ring-whp, eventually, the network reaches a terminal configuration. The time complexity is $O(n)$, where n is the number of vertices of the ring.*

Let (t_{max}, id_{max}) be the maximal label of the vertices of G . If there is a unique vertex v such that $(t_v, id_v) = (t_{max}, id_{max})$ then for each vertex v of G , $size_v$ is equal to the number of vertices of G and $max_v = (t_{max}, id_{max})$.

Remark 4. Algorithm Counting-Ring-wvhp is obtained by substituting Splitting-Naming-wvhp to Splitting-Naming-whp in Algorithm Counting-Ring-whp. Claim 3 is still valid for Algorithm Counting-Ring-wvhp.

As for the spanning tree computation, properties concerning Algorithm Counting-Ring-whp (resp. Algorithm Counting-Ring-wvhp) are deduced immediately from previous sections and summarised by:

Proposition 9. *Let G be a ring graph having n vertices. Algorithm Counting-Ring-whp (resp. Algorithm Counting-Ring-wvhp)*

is correct w.h.p. (resp. w.v.h.p.). The size of messages of Algorithm Counting-Ring-whp (resp. Counting-Ring-wvhp) is $O(\log n)$ w.h.p., it is also the expected value (resp. $O((\log n)(\log^ n)^2)$ w.v.h.p. and the expected value is $O((\log n)(\log^* n))$). The message complexity of these two algorithms is w.h.p. $O(n \log n)$.*

7. A Monte Carlo Election Algorithm Correct w.h.p. (resp. w.v.h.p.)

This section presents, after some preliminaries, message terminating Monte Carlo election algorithms for anonymous graphs without any initial knowledge correct w.h.p. (resp. w.v.h.p.). They differ from Algorithm ELECT in [AM94] p. 315 only by the choice of labels of vertices.

7.1. Preliminaries

We assume that initially every node is marked *undetermined*. An election algorithm is an algorithm such that in a final configuration exactly one process is marked as *elected* and all the other processes are labelled *non-elected*. Moreover, it is supposed that once a process becomes *elected* or *non-elected* then it remains in such a state until the end of the algorithm. In fact for anonymous ring graphs without any initial knowledge we have:

Theorem 1. *There is no election algorithm for ring graphs that is correct with probability $\alpha > 0$.*

This theorem is a direct consequence of Theorem 4.2 in [IR90] given below.

Theorem 4.2[IR90] *There is no process terminating algorithm for computing the ring size that is correct with probability $\alpha > 0$.*

Thus in the sequel of this section we use the sentence “election algorithm” as Afek and Matias in [AM94] considering that the state *elected* may be not terminal: a process marked as *elected* can be marked *non-elected* later.

7.2. An Election Algorithm

The aim of Algorithm *Elect-whp* (Algorithm 6) is to choose as elected the unique vertex v (if there exists a unique), such that:

$$\forall w \neq v \quad (t_w, id_w) < (t_v, id_v).$$

Algorithm 6: Elect-whp.

```

I : {If  $(t_v, id_v)$  is not defined}
begin
  call Splitting-Naming-whp(v);
   $max_v := (t_v, id_v)$ ;
   $leader_v := true$ ;
  send  $\langle (t_v, id_v) \rangle$  to all neighbours
end
D : {A Message  $(t, id)$  has arrived at  $v$  through port  $l$ }
begin
  if  $(t_v, id_v)$  is not defined then
    call Splitting-Naming-whp(v);
     $max_v := (t_v, id_v)$ ;
     $leader_v := true$ ;
    send  $\langle (t_v, id_v) \rangle$  to all neighbours
  if  $(t, id) > max_v$  then
     $max_v := (t, id)$ ;
     $leader_v := false$ ;
    send  $\langle (t, id) \rangle$  to all neighbours except through  $l$ 
end

```

Remark 5. Initially, each vertex is labelled elected and each vertex modifies its label at most once.

Claim 4. Let G be a graph. For each run of *Elect-whp*, eventually, the network reaches a terminal configuration. The time complexity is $O(D)$, where D is the diameter of G .

Let (t_{max}, id_{max}) be the maximal label of the vertices of G . If there is a unique vertex v such that $(t_v, id_v) = (t_{max}, id_{max})$ then there is a unique vertex such that $leader$ is true, the others have $leader$ equal to false.

Remark 6. Algorithm *Elect-wvhp* is obtained by substituting *Splitting-Naming-wvhp* to *Splitting-Naming-whp* in Algorithm *Elect-whp*. Claim 4 is still valid for Algorithm *Elect-wvhp*.

Concerning the analysis of these algorithms, results of previous sections imply:

Proposition 10. Let G be a graph having n vertices and m edges. For each run of Algorithm *Elect-whp* (resp. Algorithm *Elect-wvhp*), G reaches a stable configuration and there is exactly one vertex with the label elected w.h.p. (resp. w.v.h.p.). The size of messages of Algorithm *Elect-whp* is $O(\log n)$ w.h.p., it is also the expected value. The size of messages of Algorithm *Elect-wvhp* is $O((\log n)(\log^* n)^2)$ w.v.h.p. and the expected value is $O((\log n)(\log^* n))$. The message complexity of these two algorithms is w.h.p. $O(m \log n)$.

8. Ring with Identities + Splitting Procedure = $O(n \log n)$ Message Complexity w.h.p. Election Algorithm

This section gives an illustration of the power of the splitting procedure studied in Section 2 thanks to Proposition 7.

Let R be a ring having n vertices. We assume that vertices of R are aware of n , and each vertex v has a unique identity denoted $ident_v$.

We consider the order defined in Section 2.

Algorithm 7: Procedure Splitting(v).

```

1:  $t_v := 0$ 
2: repeat
3:   draw uniformly at random a bit  $b(v)$ ;
4:    $t_v := t_v + 1$ 
5: until  $b(v) = 1$ 

```

For electing a vertex of R , first each vertex v of R applies Procedure Splitting obtaining t_v , and then it broadcasts over the ring the triple $(t_v, ident_v, hop)$, where $ident_v$ is the initial identity of v and hop is an integer (its initial value is 1) which is incremented as it moves over the ring. Each vertex memorises the maximal value that it sees. A message is stopped as soon as it reaches a vertex w having a maximal value greater than or equal to the couple of the message.

Algorithm 8: Elect-ring.

```

I : {If  $t_v$  is not defined}
begin
  | call Splitting( $v$ );
  |  $max_v := (t_v, ident_v)$ ;
  | send  $\langle (t_v, ident_v, 1) \rangle$  to Next
end
D : {A Message  $(t, ident, h)$  has arrived at  $v$ }
begin
  | if  $t_v$  is not defined then
  |   | call Splitting( $v$ );
  |   |  $max_v := (t_v, ident_v)$ ;
  |   | send  $\langle (t_v, ident_v, 1) \rangle$  to Next
  | if  $(t, ident) > max_v$  then
  |   |  $max_v := (t, ident)$ ;
  |   | send  $\langle (t, ident, h + 1) \rangle$  to Next
  | else
  |   | if  $(t, id) = (t_v, ident_v)$  and  $h = n$  then
  |     |  $Leader_v := true$ 
  | end
end

```

The elected vertex is the vertex u which receives a message $(t, ident, h)$ with $t = t_u$, $ident = ident_u$ and $h = n$. Initially, for each vertex v $leader_v$ is undefined.

Proposition 11. *Let R be a ring having n vertices such that each vertex has a unique identity and knows n . Algorithm *Elect-ring* terminates and elects a vertex with messages of size $O(\log n)$ w.h.p. and the number of messages through the ring is w.h.p. $O(n \log n)$.*

PROOF. Any two vertices v and w of R choose at random with the same distribution the first elements t_v and t_w of their couples. Proposition 7 and the order on couples imply that each vertex sends w.h.p. $O(\log n)$ messages with respect to the first element of couples. The proposition is then a direct consequence of Proposition 2.

Remark 7. In a certain sense our result is optimal since any decentralised wave algorithm for ring networks exchanges $\Omega(n \log n)$ messages, on the average as well as the worst case (see [Tel00] Corollary 7.14).

9. From Asymptotic Error Probabilities to Decentralised Instantaneous Error Probabilities

9.1. Presentation of the Problem

One may wonder whether it is possible to obtain Monte Carlo algorithms which solve problems discussed above and which ensure for each vertex v an error probability bounded by a constant ϵ_v determined by v in a fully decentralised way.

This section gives a positive answer to this question illustrated through the election problem.

9.2. The Case of the Election Problem

We prove the following theorem:

Theorem 2. *Let $G = (V, E)$ be a graph having n vertices. For each $v \in V$, let $\epsilon_v > 0$ be an error probability. There exists a message terminating algorithm to perform election in G w.h.p. with error probability $\epsilon = \min\{\epsilon_v \mid v \in V\}$. Its time complexity is $O(D)$ (where D is the diameter of the graph), its message complexity is $O(m \log n)$ w.h.p. and the size of each message is $O(\log(n + \epsilon^{-1}))$ with probability at least $1 - \epsilon$.*

PROOF (BEGIN OF THE PROOF OF THEOREM 2). Let G be a graph, v be a vertex of G and let ϵ_v be a constant ($\epsilon_v < 1$) known by v . The vertex v wants to ensure that the election performed by the Monte Carlo algorithm is correct with probability $1 - \epsilon_v$. Instead of drawing a bit uniformly at random until it gets 1, the vertex v does this operation $r_v = \lceil \epsilon_v^{-1} \rceil$ times and it memorises the maximal lifetime, denoted $t_v^{(r_v)}$, among the r_v lifetimes it obtains. The sequel of the algorithm is the same as *Elect-whp*: the vertex v draws at random a number id_v in the set $\{0, \dots, 2^{t_v^{(r_v)} + 3 \log(t_v^{(r_v)})} - 1\}$ etc.

Now we prove that, in this way, we obtain a Monte Carlo algorithm which elects in a graph w.h.p. and ensures an error probability bounded by a constant

ϵ where ϵ is the smallest value among the set of error probabilities fixed independently by each vertex of the graph G (this algorithm is denoted *Elect-whp- ϵ*).

Thus Algorithm *Elect-whp- ϵ* resembles the standard election in a virtual graph of $N = \sum_v r_v$ vertices with the difference that only one of the virtual vertices corresponding to v and having the same maximum lifetime t_v continues the drawing process. We would intuitively expect this pruning of vertices to reduce the probability of failure but our argument does not depend on a proof of this intuition.

As for Algorithm *Elect-whp*, the time complexity of *Elect-whp- ϵ* is $O(D)$ (where D is the diameter of the graph) and the message complexity is w.h.p. $O(m \log n)$.

We number the vertices of G from 1 to n with the first vertex v_1 one of those with the minimum ϵ and abbreviate r_{v_1} as r_1 etc.

For any t_0 , we have $\mathbb{P}r(\text{failure}) \leq \mathbb{P}r(T < t_0) + \mathbb{P}r(\text{failure}|T \geq t_0)$.

Before continuing the proof of the theorem we prove a lemma concerning the second term in this sum.

Lemma 3. $\mathbb{P}r(\text{failure}|T \geq t_0) \leq (N - r_1) \times 2^{-(2t_0+3 \log t_0)}$.

PROOF. Since $T \geq t_0$, there is at least one vertex v which has drawn 00...0 ($t_0 - 1$ times) in at least one of its r_v drawings. Let v_i be the first of these vertices. We consider, for $i \leq j \leq n$, the number e_j of excess vertices, that is those in $v_i \dots v_j$ which have the maximum couple (t, id) not counting the first with this maximum. Initially $j = i$ and e_i is zero; passing from j to $j + 1$, either v_{j+1} has the same couple as the maximum so far (in $i..j$) and $e_{j+1} = e_j + 1$, or its couple is greater than this maximum and $e_{j+1} = 0$, or else it is less and $e_{j+1} = e_j$. Let $t-1$ be the number of 0-bits at the start of the current maximum drawing; $t \geq t_0$. The probability that v_{j+1} has the same couple is at most the product of three probabilities:

- some drawing started with $t - 1$ 0-bits: probability at most $r_{j+1}2^{1-t}$,
- all drawings that started with $t - 1$ 0-bits had a 1-bit in position t : probability at most $1/2$,
- the id_{v_j} chosen (in $0 \dots 2^{t+3 \log t} - 1$) was identical to the maximum so far: probability $2^{-(t+3 \log t)}$.

The product of these three probabilities is at most $r_{j+1}2^{-(2t_0+3 \log t_0)}$. Hence we deduce by induction that:

$$\mathbb{E}(e_j) \leq \sum_{m=i+1}^j r_m 2^{-(2t_0+3 \log t_0)}$$

and, in particular,:

$$\mathbb{E}(e_n) \leq \sum_{m=i+1}^n r_m 2^{-(2t_0+3 \log t_0)}.$$

Thus we have that $\mathbb{E}(e_n)$ is the weighted sum of values (for the possible values of i) each $\leq (N - r_1) \times 2^{-(2t_0 + 3\log t_0)}$, and so is itself upper bounded by this value.

Finally we have $\mathbb{Pr}(\text{failure}|T \geq t_0) = \mathbb{Pr}(e_n > 0|T \geq t_0) \leq \mathbb{E}(e_n|T \geq t_0)$ completing the proof of the lemma. \square

PROOF (CONTINUATION OF THE PROOF OF THEOREM 2). We return to the upper bound $\mathbb{Pr}(T < t_0) + \mathbb{Pr}(\text{failure}|T \geq t_0)$.

The first term is the probability that no virtual vertex draws $t_0 - 1$ or more 0-bits before the first 1. We will use the upper bound $e^{-N/2^{t_0-1}}$ except when $t_0 = 2$ for which we have the exact value 2^{-N} .

The bound on the probability is given, for each N , by choosing a suitable value for t_0 ; this value is $t_0 = \lceil \log N - \log \log N \rceil$ except for $3 \leq N \leq 4$ for which $t_0 = 2$.

We consider two cases:

- $3 \leq N \leq 4$: $t_0 = 2$ giving failure probability $\leq 2^{-N} + (N - 1)/2^7 < 1/N$.
- $N > 4$: We will show that each of the two terms in the sum is bounded by half the allowed error probability of $1/r_1$. For the first term we have $t_0 < \log N - \log \log N + 1$ giving $2^{t_0-1} < N/\log N$ so that our upper bound on $\mathbb{Pr}(T < t_0)$ is less than $e^{-\log N} = N^{-\log e} < 1/2N$.

For the second term we use the lemma that it is less than $(N - r_1) \times 2^{-(2t_0 + 3\lceil \log t_0 \rceil)}$ and claim that this is at most $1/2r_1$. This follows because $2r_1(N - r_1) \leq N^2/2$ and $t_0 \geq \log N - \log \log N$ so that $2^{(2t_0 + 3\log t_0)} \geq N^2 t_0^3 / \log^2 N \geq N^2/2$. the critical case being $N = 16$, $t_0 = 2$ for which equality holds.

Then, we have proved that the probability of failure is upper bounded by $1/N$ which is less than ϵ .

We proved that if each vertex v acts as a set of $r_v = \lceil \epsilon^{-1} \rceil$, then we obtain an election algorithm correct w.h.p. and with probability at least $1 - \epsilon$, where $\epsilon = \min\{\epsilon_v \mid v \in V\}$.

To achieve the proof of the theorem we analyse the size of the messages.

Exchanged messages are of size at most $T + 3 \log T$ where T is the maximum lifetime over all the N vertices. We have:

$$\begin{aligned} \mathbb{Pr}\left(T > 3 \log\left(n + \frac{1}{\epsilon}\right)\right) &= \mathbb{Pr}\left(X_{3 \log(n + \frac{1}{\epsilon})} \geq 1\right) \\ &\leq \mathbb{E}\left(X_{3 \log_3(n + \frac{1}{\epsilon})}\right) \\ &= \frac{n/\epsilon}{(n + \frac{1}{\epsilon})^3} \leq \epsilon. \end{aligned} \tag{26}$$

Thus, with probability at least $1 - \epsilon$, messages are of size $O\left(\log\left(n + \frac{1}{\epsilon}\right)\right)$.

Remark 8. The same constructions and results can be obtained for spanning tree construction and other problems which are easily solved once a leader is elected.

Remark 9. A similar method based on modifying Algorithm *Elect-whp* gives an algorithm which is correct both w.v.h.p. and with the probability $1 - \epsilon_v$ required by each vertex v . The range in which id_v is chosen must be the maximum of those used by algorithms 1 and 2 for this to be correct for small N .

10. Conclusion

We have analysed splitting and naming procedures and proved, in particular, that w.h.p. or w.v.h.p. they provide exactly one maximal labelled vertex which can be used later for computing a spanning tree, electing or even counting the number of vertices of a ring. The splitting procedure may be used also to ensure that the total number of messages broadcast through a network for deterministically solving some problems, e.g. election or spanning tree computation, is low w.h.p. Thanks to these procedures we can also obtain Monte Carlo algorithms correct with w.h.p. (resp. w.v.h.p.) and correct with a probability $1 - \epsilon_v$ required by a vertex v .

It seems interesting to investigate relations between fundamental techniques and principles of protocols in radio networks presented in [JS02, MW05] (see also references therein) and the ones studied in this paper.

References

- [AAER07] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.
- [AM94] Y. Afek and Y. Matias. Elections in anonymous networks. *Inf. Comput.*, 113(2):312–330, 1994.
- [Ang80] D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Symposium on Theory of Computing*, pages 82–93, 1980.
- [ASW88] H. Attiya, M. Snir, and M. K. Warmuth. Computing on an anonymous ring. *J. ACM*, 35(4):845–875, 1988.
- [AW04] H. Attiya and J. Welch. *Distributed computing: fundamentals, simulations, and advanced topics*. John Wiley & Sons, 2004.
- [FMS96] J. A. Fill, H. M. Mahmoud, and W. Szpankowski. On the distribution for the duration of a randomized leader election algorithm. *Ann. Appl. Probab.*, 6:1260–1283, 1996.
- [GR05] R. Guerraoui and E. Ruppert. What can be implemented anonymously? In *DISC*, pages 244–259, 2005.
- [IR90] A. Itai and M. Rodeh. Symmetry breaking in distributed networks. *Inf. Comput.*, 88(1):60–87, 1990.

- [JS02] T. Jurdzinski and G. Stachowiak. Probabilistic algorithms for the wakeup problem in single-hop radio networks. In *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, pages 535–549, 2002.
- [KMW11] R. Kalpathy, H. M. Mahmoud, and M. D. Ward. Asymptotic properties of a leader election algorithm. *Journal of applied probability*, 48(2):569–575, 2011.
- [Lav95] C. Lavault. *Evaluation des algorithmes distribués*. Hermès, Paris, 1995.
- [MW05] T. Moscibroda and R. Wattenhofer. Maximal independent set in radio networks. In *Proceedings of the 25 Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 148–157. ACM Press, 2005.
- [Pro93] H. Prodinger. How to select a loser. *Discrete Mathematics*, 120(1-3):149–159, 1993.
- [RFJ+07] M. K. Ramanathan, R. A. Ferreira, S. Jagannathan, A. Grama, and W. Szpankowski. Randomized leader election. *Distributed Computing*, 19(5-6):403–418, 2007.
- [SS94] B. Schieber and M. Snir. Calling names on nameless networks. *Inf. Comput.*, 113(1):80–101, 1994.
- [Tel00] G. Tel. *Introduction to distributed algorithms*. Cambridge University Press, 2000.
- [Wil93] D. Williams. *Probability with Martingales*. Cambridge University Press, 1993.
- [YK88] M. Yamashita and T. Kameda. Computing on an anonymous network. In *PODC*, pages 117–130, 1988.