Theoretical Computer Science 584 (2015) 115–130

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Analysis of fully distributed splitting and naming probabilistic procedures and applications

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

*Université de Bordeaux, Bordeaux INP, LaBRI, UMR CNRS 5800, 351 cours de la Libération, 33405 Talence, France*

ARTICLE INFO

*Article history:*
Received 12 November 2013
Received in revised form 2 February 2015
Accepted 10 February 2015
Available online 17 February 2015

*Keywords:*
Monte Carlo algorithm
Spanning tree computation
Counting
Election algorithm
Probabilistic analysis
Splitting and naming

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 $O\left((\log n)(\log^* n)^2\right)$ with probability $1-o(n^{-c})$ for any $c \geq 1$; their expected size is $O\left((\log n)(\log^* n)\right)$.

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 $\epsilon_v$; the error probability $\epsilon_v$ 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 $1-o(n^{-1})$.



## 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.

* Corresponding author.
*E-mail addresses:* metivier@labri.fr (Y. Métivier), robson@labri.fr (J.M. Robson), zemmari@labri.fr (A. Zemmari).

http://dx.doi.org/10.1016/j.tcs.2015.02.016
0304-3975/

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 [7]. 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 [1].

### 1.2. The model

Our model is the usual asynchronous message passing model [5,16]. 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 [16] (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 [16], 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 $\log^* n = min\{i | \log^{(i)} n \leq 2\}$, 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 $\{0, \ldots, 2^{t_v + 3\log(t_v)} - 1\}$; finally, $v$ is labelled with the couple $(t_v, id_v)$.

Let $T$ be the maximal value in the set $\{t_v | v \in V(G)\}$. 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: $\{0, \ldots, 2^{t_v \log^* t_v} - 1\}$ 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 $O((\log n)(\log^* n))$.