Spanning Tree Construction
for Nameless Networks

Ivan LAVALLÉE       Christian LAVAULT

INRIA, Domaine de Voluceau
Rocquencourt, B.P. 105
78153 Le Chesnay Cedex. France
E.mail: lavallie@seti.inria.fr

Abstract

Two types of distributed fully asynchronous probabilistic algorithms are given in the present paper which elect a leader and find a spanning tree in arbitrary anonymous networks of processes. Our algorithms are simpler than in [11] and slightly improve on those in [9,11] with respect to communication complexity. So far, the present algorithms are very likely to be the first fully and precisely specified distributed communication protocols for nameless networks. They are basically patterned upon the spanning tree algorithm designed in [7,8], and motivated by the previous works proposed in [9,11].

For the case where no bound is known on the network size, we give a message terminating algorithm with error probability $\epsilon$ which requires $O(m\log\log(nr) + n\log n)$ messages on the average, each of size $O(\log r + \log \log n)$, where $n$ and $m$ are the number of nodes and links in the network, and $r = 1/\epsilon$. In the case where some bounds are known on $n$ ($N < n \leq KN$, with $K \geq 1$), we give a process terminating algorithm, with error probability $\epsilon$, with $O(m + n\log n)$ messages of size $O(\log n)$ in the worst case. In either case, the (virtual) time complexity is $O(D \times \log \log(nr))$. In the particular case where the exact value of $n$ is known, a variant of the preceding algorithm process terminates and always succeeds in $O(m + n\log n)$ messages of size $O(\log n)$.

1 Introduction

In a distributed algorithm, a network of processes collaborate to solve a given problem. In this framework, each site or process acquires, via local interaction with its neighbours, some global information about the system: e.g. the size of the network, its location in a (minimum-weight) spanning tree, the distances to all other processes, etc. Typically, one assumes that the processes have distinct identification labels or identities, which means that some global coordination between the processes has taken place beforehand. What happens if this assumption is dropped, so that the processes are indistinguishable? Consider for example regular nameless networks of some fixed degree. The executions of a deterministic algorithm may end with all processes in the same state, irrespectively of the network size or structure; a deterministic algorithm cannot distinguish between processes of a regular distributed system, nor can it distinguish between distinct regular networks of the same degree (see [4,5]).

The situation is different if processes are assumed to make independent probabilistic choices; probabilistic choices can be used to break symmetry in anonymous networks of indistinguishable, nameless processes. When designing election algorithms for the leader election problem (LEP) and spanning tree construction problem (STP) in anonymous networks, one has to consider the following issues: relative information on the network size and termination detection [1,2,3,5,6,8,9,11].

*Supported in part by C* (COPARADIS Group)
2 Preliminaries and results

2.1 Definitions

We consider here the standard model of static asynchronous network. This is a point-to-point communication network, described by an undirected communication graph \( G = (V, E) \) where the set of nodes \( V \) represents processes of the network and the set of edges \( E \) represents bidirectional non-interfering communication channels (links) operating between neighbouring nodes; \( |V| = n \) and \( |E| = m \). No common memory is shared by the processes. We confine ourselves only to message-driven algorithms, which do not have central controller and do not use time-outs, i.e. processes cannot access a global clock in order to decide what to do. In a transition, a process receives a message on one of its links, and changes state; a transition may be probabilistic. We also assume throughout the processes and the communication subsystem to be error-free, and that the links operate in a FIFO-manner.

The course of any execution of an algorithm is determined by a scheduler, that chooses at each step the next message to be received, as a function of the current network state. An algorithm process terminates if in every execution all processes reach a special halting state; this corresponds to an algorithm with termination detection. An algorithm message terminates if in every execution the network reaches a quiescent state where there are no pending messages on the links [11]. In message termination, the processes may ignore that the computation is halted; this corresponds to an algorithm without termination detection. An algorithm has error probability \( \varepsilon \) if, for any scheduler and any input, the probability that the algorithm terminates with the right answer is at least \( 1 - \varepsilon \).

2.2 Results

We address the problem of computing a function whose value depends on all the network processes, such as counting the number of nodes in the graph \( G \), or solving LEP and STP for \( G \). In [6], Itai and Rodeh showed that these problems can be solved on a ring by an algorithm which processor terminates (or distributively terminates) and always succeeds if and only if the ring size \( n \) is known up to a factor of two (see also [1,2,5] for improvements). It was also shown in [6] that it is possible to solve LEP in an anonymous network, with termination detection and with error probability \( \varepsilon \) only if an upper bound on the network size is known.

In [8,9,11] and in the present paper, all of these results were extended to arbitrary networks, while improving some of the bounds. In [11], Schieber and Snir presented efficient schemes of algorithms for LEP and STP; in [9], Matias and Afek proposed more detailed schemes of three types of simple algorithms which efficiently solve LEP.

Given some \( 0 < \varepsilon < 1 \), let \( r = 1/\varepsilon \). On the assumption that the exact value of \( n \) is a priori unknown to any process of the network — and thus without termination detection —, the probabilistic solutions given in [11] require \( O(m \log \log (nr) + n \log n) \) messages of size \( O(\log n + \log r) \), for fixed error probability \( \varepsilon \). The probabilistic solutions given in [9] require \( O(m \log n \times r \log r) \) messages of size \( O(\log r + \log \log n) \) on the same assumptions. In the case when \( n \) is known up to a factor of \( K \geq 1 \), that is \( N < n < KN \), [9] achieves \( O(m \times r \log (Kr) \log r) \) messages of size \( O(\log n) \), in the worst case, with \( N < n < 2N \). The time complexity in [9,11] is \( O(D) \) and \( O(n) \), respectively, where \( D \) is the diameter of the network.

Our algorithms are simpler than in [11] and slightly improve on those in [9,11] with respect to communication complexity. Compared to [11], we give overall solutions with improved bit complexity and less bit information per node; the message complexity in [9] is also higher than ours. Moreover, to the best of the authors' knowledge, the present algorithms are very likely to be the first fully and

\(^1\)Throughout the paper, \( \log \) denotes the base two logarithm and \( \ln \) the natural logarithm.