GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

PROBABILISTIC DISTRIBUTED ALGORITHM FOR THE SPANNING TREE CONSTRUCTION BASED ON A HANDSHAKE ALGORITHM

El Mehdi Stouti[*1], Jaafar El Bakkali[2] and Kamal Eddine El Kadiri[3]

[*1]LIROSA Laboratory Abdelmalek Essaâdi University, Morocco

[2]Abdelmalek Essaâdi University, Faculty of sciences, Tetouan, 93000, Morocco

[3]TIMS Research Team, LIROSA Laboratory Abdelmalek Essaâdi University, Morocco

ABSTRACT

In this paper, we introduce and analyze a probabilistic distributed algorithm for spanning tree construction. This algorithm is based on the handshake algorithm. The edges of the maximal matching s, produced by the hand-shack algorithm, are linked in a distributed manner and following some roles. We have proved that the residual graph is a cyclic and all vertices belong to it. Our algorithm is performed to compute a maximal matching and link all edges where hand-shacks are occurred in some way to obtain a single a cyclic graph.

Keywords- *Distributed Algorithms, Randomized Algorithm Analysis, Handshake, Maximal Matching, Spanning tree construction.*

I. INTRODUCTION

The spanning tree problem is a well-known combinatorial optimization problem concerned in linking all the vertices of an undirected and connected graph without creating any cycle. The methods for finding a spanning tree have played a central role in the design of computer algorithm [1].

The earliest known algorithm for finding a minimum spanning tree was given by Otakar Borüvka back in 1926 [2]. In every step of the algorithm, each vertex selects its smallest adjacent edge. These edges are added to the MST without creating any cycle.
Kruskal's algorithm was given by Joseph Kruskal in 1956 [3]. It creates a forest where each vertex in the graph is initially a separate tree. For each edge (u,v) in sorted order, this algorithm does the following: If the vertices u and v belong to two different trees, then add (u,v) to the forest, combining two trees into a single tree. It proceeds until all the edges have been processed.

Prim's algorithm was conceived by Robert Prim in 1957 [4]. It starts from an arbitrary vertex, and builds upon a single partial minimum spanning tree, at each step adding an edge connecting the vertex nearest to, but not already in the current partial minimum spanning tree. It grows until the tree spans all the vertices in the input graph.

For our study, we use graph theory to represent and analyze our network. So, a network is described by a undirected and connected graph. The vertices represent the nodes or network processes, whereas the edges represent the communication links between those processes.

In this paper, we present and analyze a probabilistic distributed algorithm to construct a spanning tree without a specific criterion. Our algorithm is based on the handshake algorithm presented in [5] which is a probabilistic distributed algorithm for finding a maximal matching.

We consider that the edges, where the hand-shacks are occurred, are in the spanning tree. Those edges constitute the base for our algorithm to construct a spanning tree.

In a distributed manner, we join those edges and all the isolated vertices in a connected graph avoiding cycles. The residual graph is the spanning tree for which we are looking.

The paper is organized as follows: Section 2 presents some notifications and definitions necessary for understanding the rest of the document. Section 3 exposes our model and assumptions. Section 4 is devoted to describe our algorithm, whereas Section 5 shows the analysis of the introduced algorithm. Finally, Section 6 concludes the paper and presents our further work.

II. DEFINITIONS AND NOTATIONS

In graph theory, given a graph G = (V,E), a matching M in G is a set of pairwise non-adjacent edges; that is, no two edges share a common vertex where V is the set of all the vertices of *G*, and E is the set of the edges linking those vertices. A vertex is matched (or saturated) if it is an endpoint of one of the edges in the matching. Otherwise the vertex is unmatched [6].

A maximal matching is a matching M of a graph G with the property that if any edge not in M is added to M, it is no longer a matching; that is, M is maximal if it is not a proper subset of any other matching in graph G. From one side, a matching M of a graph G is maximal if every edge in G has a non-empty intersection with at least one edge in M. The following figure shows an example of maximal matchings (edges in bold) in the graph.

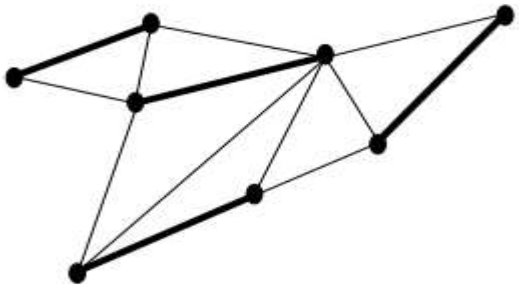

Fig. 1. Example of maximal matchings

In other side, a spanning tree T of a connected, undirected graph G is a tree composed of all the vertices and some (or perhaps all) of the edges of G. A spanning tree of G is a selection of edges of G that forms a tree spanning every vertex [7]. That is, every vertex lies in the tree, but no cycles (or loops) are formed. The following figure shows an example of the spanning tree obtained from the graph in Fig. 1.

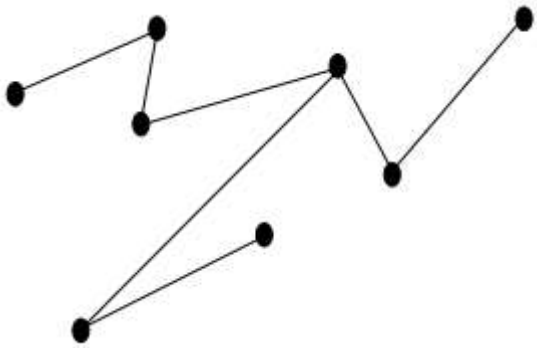

Fig. 2. Example of a spanning tree

III. MODEL AND ASSUMPTIONS

In this respect, we considered the standard model of communication networks point-to-point [8][9]. We assumed that each process has a local unshared memory with a bounded capacity and at least one processor. It can only communicate directly with its neighbors. Also, it is able to distinguish between its ports (i.e. a port from which a message is received or sent).

Many researches have treated the maximal matchings problem such as MSZ algorithm [10] and HS algorithm [11]. In MSZ algorithm the authors have proposed and analyzed a randomized algorithm to get hand-shacks between neighbors in an anonymous graph, and they also showed the efficiency of its algorithm in several types of graphs.

In HS algorithm [11], the authors have introduced and studied a handshake algorithm based on random delays which are generated uniformly in the real interval [0,1]. This algorithm can also be considered as a probabilistic distributed algorithm to find a maximal matching. The handshakes take place between neighbor processors if both processors are free at the generated time.

The expected number of hand-shacks found by HS algorithm, in which we based on, is substantially greater than that obtained in [10].

We consider the standard model of communication networks Point-to-point [8] [9]. A network is described by a undirected and connected graph G = (V,E) where V is the set of all the vertices of G, and E is the set of edges linking those vertices. The vertices represent the nodes or network processes, the edges represent the communication links between processes.
Every process has a local non-shared memory with a bounded capacity and at least one processor. It can only communicate directly with its neighbors. Also, it is able to distinguish between its doors. The identities (IP addresses on the Internet) of other processes, in particular those of its neighbors, are unknown (local orientation). It is reliable: it produces no failure or on the vertices or on the communication links.
The processes communicate in a network only by exchanges of messages. The exchange of messages is done without losing or alteration or modification. It consist the cost of execution dominating the algorithm on a network. How exchanges between processes are carried out determines the type of network. Processes or nodes are anonymous and each of them is able to distinguish its ports, and they know in which port a message is received or sent.

1. The global clock is common to all processes,
2. Every event that takes place at a certain time is of zero duration.

We assume that communications take no time, that is to say, that the period between the decision of a vertex (process) and notification to its neighbors is equal to 0.
So, for a given graph G, each vertex v 2 G generates uniformly a random variable for each neighbor. This random variables generated by a vertex for its neighbors belong to the real interval [0,1]. They represent waiting times to get a handshake with its neighbors.
The number of waiting times generated by all this processes is two-times the number of the edges (i.e., one waiting times for each half-edge). The vertex agenda is the set of waiting times proposed by this vertex to its neighbors.
When the clock reaches the smallest time of all the generated times, then a hand-shack occurs between the process that provides the time and the neighbor who is assigned to it. So, the two processes remove all other hand-shacks with their other neighbors. Indeed, their neighbors remove from their agendas the times that they offer to these two processes which have a handshake. The algorithm continues to run through the remaining process equipped with modified agendas until the lap time (unit time) expired.
The ends of a hand-shack choose the minimum value between the values associated with their half edges to add it to the edges of the spanning tree knowing that all handshakes are part of the spanning tree.

IV. ALGORITHM

In this section, we describe the two steps of our algorithm. The first one is to calculate the maximal matchings, while the second one is to compute a spanning tree. The two steps are executed in parallel.
The process q generates tq(r) = cv: a random variable (r.v.) uniformly chosen in the real interval [0,1] for all neighbor r, and that represent at the same time a color cv (we can ordered the colors).
Each vertex v communicates with its neighbors via sending and receiving messages.
Every vertex v contains the following parameters:

- $t_v(r)$ represents a random variable uniformly chosen by the vertex v in the real interval [0,1] for every neighbor r.
- P_v represents the existence of the vertex v in the residual graph (i.e. P_v equals to true if v belongs to the spanning tree).
- d_v a parameter that counts the number of 0 received by the vertex v (if d_v equals to the degree of v then v is s isolated).
- S_v the set of all values generated by v to each neighbor r of v, S_v contains $t_v(r)$, and let $S_{v,r} = \{S_v \cup S_r \setminus \{t_v(r),t_r(v)\}\}$.

Let V_v be a set of the neighbors of v such as 8b 2 V_v we have p_v = true (initially v can chose any neighbor) and $V_{v,r} = V_v \cup V_r$.

We suppose that each time we have a hand-shack between v and r, they can share directly the set $S_{v,r}$.
The algorithm bellow summarizes the two steps of our algorithm merged in one description.

Algorithm 1 Spanning tree construction

Data: The vertex v waits until one of the following events happened:

1: **for** $t < 1$ **do**
2: **if** v receives 0 from a neighbour r **then**
3: v increase d_v by 1 ;
4: **else if** v receives 2 from a neighbour r
5: **then** $V_q = V_q \setminus p_{q,r}$;
6: **else if** q receives 3 from a neighbour r
7: **then** $p_{q,b}$ becomes true such that $min(S_q)$
8: $= t_q(b) \in S_q(b\#r$ and $b \in V_q)$;
9: **else if** q receives 1, S_r and V_r from a
10: neighbour r and $V_{q,r} \# \emptyset$ **then**
11: **if** $min(S_{q,r}) = t_q(b) \in S_q$ **then** q sends
12: 2 to r and b and sends 0 for all
13: other neighbours ;
14: **else if** $min(S_{q,r}) = t_r(b) \in S_r(b\#q)$
15: **then** $p_{q,r}$ becomes true and q sends 3
16: to r and 0 to all other neighbours ;
17: **end if**
18: **else if** t = t_q(r) and q did not received
19: any message from a neighbour r
20: **then** q sends 1, S_q and V_q to r and 0
21: to all other neighbours ;
22: **else if** $t = 1$ **then**
23: **if** $\exists r \mid p_{q,r} = true$ **then**
24: q belongs to the spanning tree ;
25: **end if**
26: **else** the round ends without
27: participation of q to any
28: rendezvous or a potato so it is an
29: isolated vertex with $d_q = deg(q)$;
30: **end if**
31: **end for**

V. Analysis of the algorithm ANALYSIS OF THE ALGORITHM

The algorithm starts by generating uniformly 2|E| real independent random variables (r.v.) in the real interval [0,1]: a random variable for each half edge. We assume that (2|E|)! Permutations on the set of these real numbers have the same probability. This is the main assumption on which is based the result of our analysis.

In order to maintain this hypothesis, we simply assume that, for each edge e = {u,v} ∈ G, the algorithm will produce two continuous r.v. t_e(u) and t_e(v), we assume that these r.v. associated to the edges are all independent.

The first hand-shack takes place on an edge e = {u,v}, if at least one of the two is associated r.v. t_e(u) or t_e(v) is minimal in the whole graph. Thus, for the first handshake of G, every edge has the same chance 1 = |E| to be chosen.

The attribution of the first hand-shack to the vertices u and v on the edge fu; vg implies that these two vertices and their incident edges are removed from the graph. The algorithm continues to run on the residual graph (preserving the random generation of the remaining edges) until no edge remains in the set of edges of the graph.

The number of handshakes in a round is simply the total number of edges which the hand-shack are assigned. We denote by N(G) the number. It takes the value 0 with probability 1 if E = ∅, the value 1 with probability 1 if |E| = 1. Generally, it takes a value of the set of all maximal matching cardinalities in G, with a certain probability.

Proposition 5.1: Let G = (V,E) a graph with |V| = n and |E| = m, we have:

$$\Pr(N(G) = k) = \frac{1}{m} \sum \Pr(N(G_e) = k - 1), \forall k \geq 1.$$

We note that for m≥1, this algorithm has at least one handshake with probability equals to 1.

VI. CONCLUSION

In this paper, we have introduced and analyzed a probabilistic distributed algorithm for the spanning tree construction based on a handshake algorithm. According to our algorithm, and during each round (a unit of time), every vertex v executes a set of instructions and decides or there is a hand-shack with one neighbor or it will become an unmatched vertex.

As perspective, we attend to prove that the spanning tree generated by our algorithm is the minimal spanning tree. In addition, we will implement our algorithm in the VISIDIA simulator [12].

REFERENCES

1. *Y. Matsumoto, N. Kamiyama, and K. Imai, "An approximation algorithm dependent on edge-coloring number for minimum maximal matching problem", Information Processing Letters, vol. 111, no. 10, pp. 465- 468, 2011.*
2. *G. Berganti~nos and J. Vidal-Puga, "The folk solution and boruvka's algorithm in minimum cost spanning tree problems", Discrete Applied Mathematics, vol. 159, no. 12, pp. 1279-1283, 2011.*
3. *[3] Z. Ning and W. Longshu, "The complexity and algorithm for minimum expense spanning trees", Procedia Engineering, vol. 29, no. 0, pp. 118- 122, 2012, 2012 International Workshop on Information and Electronics Engineering.*
4. *C. Martel, "The expected complexity of prim's minimum spanning tree algorithm", Information Processing Letters, vol. 81, no. 4, pp. 197-201, 2002.*
5. *A. E. Hibaoui, J. M. Robson, N. Saheb-Djahromi, and A. Zemmari, "Uniform election in trees and polyominoids", Discrete Appl. Math., vol. 158, no. 9, pp. 981-987, May 2010.*
6. *R. Diestel, Graph Theory, second ed., electronic ed. Berlin: Springer, February 2000.*
7. *M. C. Golumbic, Algorithmic graph theory and perfect graphs, 2nd ed. Elsevier, 2004.*
8. *G. Tel, Introduction to distributed algorithms. Cambridge University Press, 2000.*
9. *C. Lavault, Evaluation des algorithmes distribués : analyse, complexité, méthodes, ser. Collection Informatique. Paris: Hermès, 1995 (53-Mayenne).*
10. *Y. Métivier, N. Saheb, and A. Zemmari, "Analysis of a randomized rendezvous algorithm", Inf. Comput, vol. 184, no. 1, pp. 109-128, 2003.*
11. *A. El Hibaoui, Y. Métivier, and N. Z. A. Robson, J.M.and Saheb, "Analysis of a randomized dynamic timetable handshake algorithm", 2009.*
12. *M. Mosbah and A. Sellami, "Visidia: A tool for the VIsualization and SImulation of Distributed Algorithms", http://www.labri.fr/visidia/.*