Distributed and Randomized Enumeration

Piotr Dembiñski[1]
Institute of Computer Science
Polish Academy of Sciences
ul. Ordona 21, 01-237 Warsaw, Poland

Abstract

The paper describes a randomized, distributed enumeration algorithm which (in contrast to deterministic solutions) works for all network topologies and with fully asynchronous communication. The algorithm's correctness and efficiency are argued. The expectation value of its execution time is of the order of the time needed for a broadcast.

Key words: *distributed algorithms; analysis of algorithms; interconnection networks*.

1. Introduction

Distributed algorithms (protocols) aim at solving a global network problem by local means, that is, by computations of network nodes that depend solely on the nodes' local states and exchange of messages between nodes. When the algorithm terminates the global result is known to all participating nodes.

Leader election or *nodes enumeration* are good examples of global problems for which known distributed solutions exhibit most of the typical design difficulties. Both problems require that no component is privileged and, therefore, each node performs exactly the same algorithm. Such distributed algorithms (protocols) are called sometimes *symmetric*. Let us formulate more precisely the enumeration problem which is the subject of interest in this paper.

Assume we have a *network of processes* $P_1, P_2, \ldots, P_N$ the graph of which is finite, connected and without self-loops. Nodes of the graph correspond to the network processes and the graph edges represent the bi-directional communication links between them (unidirectional case can be also considered, but additional graph assumptions are needed). Each process (node) can perform some local computations (changing its local state) and it can exchange messages with *neighbor processes* (nodes), i.e., those directly linked to it by a communication channel (graph edge). Local computations are completely independent and inter-process communication is asynchronous.

The *enumeration problem* for a network of N nodes consists in finding a distributed algorithm (i.e., an algorithm that is performed independently by each node in the network) that:

- terminates with all nodes of the network (graph) *properly enumerated*, i.e., each node gets a unique label from the set {1, 2, ... , N},
- the final distribution of labels is known to all participating nodes,
- none of the nodes is privileged by the algorithm, i.e., each of them has equal chance to get anyone of the labels.

It is important to stress that the only global information, i.e., known to all nodes at the beginning, is the network size (number N).

One can immediately see that a solution to the enumeration problem is also a solution to the leader election problem. It is enough to agree that the node labeled by a selected label (e.g., 1 or N) is the leader. It is also true that a leader election algorithm may serve as a basis for solving the enumeration problem [10].

Considering only networks with asynchronous communication and deterministic protocols, it is known that the existence of solutions to both problems depends on the network topology represented by its graph. It is known that for some graphs - so called *ambiguous graphs* - there are no deterministic solutions. A recent and elegant characteristics of such graphs and impossibility of enumeration can be found in [8]. Other impossibility results are described, e.g., in [2] and [7]. Roughly

[1] Partially supported by the Esprit Project 20288 CRIT-2 and INCO-Copernicus Project 977124 IDECOMP

speaking, in an ambiguous graph it is impossible to distinguish, by local means, some of its subgraphs or, in other words, the graph can be "folded" to a smaller graph by a mapping that preserves the graph relation (i.e., a morphism) and is bijective if restricted to the neighborhood of any of the graph nodes. An example of a simple ambiguous graph structure is a ring (with at least six nodes) - the structure very often chosen for a network organization.

The situation changes when randomized protocols are considered, i.e., protocols in which each process can draw a random number to determine its next step. This fact has been recognized earlier, especially in works on fault-tolerant computations where randomized solutions to the Byzantine Agreement Problem (introduced first in [9]) have been studied as, e.g., in [1].

This short paper presents a simple randomized enumeration protocol that works efficiently for any network graph. A comparable result was independently proposed in [10] where iterated leader election (randomized) algorithm enumerates network nodes. Some randomized elements were also introduced in a version of the otherwise deterministic solution studied in [5].

We start this paper with some simplifying assumptions that are formulated in Section 2. Section 3 describes the protocol in detail. A sketch of the protocol correctness and an estimation of its performance are given in Section 4. Details on the estimation can be found in Appendix. (It is worth to point out that both the correctness and efficiency analysis are much easier to treat in randomized case if compared to a deterministic solution.) Concluding remarks (Section 5) end the presentation.

The algorithm has been implemented [4] in Estelle [6] and extensively simulated by the EDT simulator [3]. The simulation fully confirmed the obtained performance results.

2. Broadcasting assumptions

Consider a network of N nodes x_1, x_2, ... , x_N. Assume that each node x_k in the network receiving a message from a neighbor x_j, performs the following:

if the message has been already received,
i.e., it is already stored in the x_k memory
then the message is discarded
else the message is stored by x_k and it is forwarded to all x_k's neighbors but x_j.

Assume now that a message m is created by the node x_i, stored in the node local memory, and then sent to all the node neighbors. It seems obvious, in view of the above, that:

Fact 2.1. The procedure of forwarding the message m in the network eventually terminates and m is known to each node of the network i.e., it is received and stored by each node.

Due to the Fact 2.1 we can assume that a message sent by a node is received after certain delay by each other node in the network as if all nodes were directly connected. For example, if a network forms a unidirectional ring, then the maximal delay is proportional to the ring size. Therefore, further on we are not considering the problem of how the messages are broadcast. On the other hand, this problem is explicitly treated in the algorithm implementation in Estelle as specified in [4].

3. Randomized enumeration algorithm

As in previous sections, we consider a N-node network. The algorithm is performed in a (finite) number of rounds (DrawingRound procedure given in four steps below) and it can be described very shortly as follows (*active nodes* in a round are those not enumerated in previous rounds):

> In the current (first) drawing round, each active node independently draws a random number and broadcasts it (Step 1). Then it collects those numbers from all other active nodes (Step 2). Those active nodes whose random numbers are not repeated in other active nodes are separated and sorted (Step 3). The separated nodes are then enumerated by the order of their random values (Step 4). The remaining active nodes (if any), and only these nodes, are going to compete in the next round for labels greater than those assigned in the current (first) attempt. The whole algorithm terminates if there are no more nodes to be enumerated in the next round.

Let us specify the above idea in details. A message m is a triple m = (x, r, R), where x is a node (name) in the set of node (names) X, r denotes a round number and R is a (random) number, from a given interval, selected by the node x. Each node uses the following internal variables:

r	integer variable storing the current round number of the algorithm; initially r=0;
K	set of messages collected during the current drawing round; initially K=∅;
K1	subset of messages in K that identify nodes properly enumerated in the

	current round; initially K1=∅;
E	properly enumerated nodes until the current round, i.e., a set of pairs: a node and its assigned label (number) by the enumeration algorithm; initially E=∅;
s	size of the enumeration problem in the current drawing round; initially s=N;

The algorithm, for each node in the network, is defined by the following repeat-until construct:

repeat
DrawingRound
until (|E| = N).

(|E| denotes the cardinality of the set E).

With the above definition of E, the DrawingRound procedure is performed until all N nodes in X are properly enumerated.

DrawingRound procedure, for each node x in X:

Step 1: r := r+1;
if x is not enumerated yet, i.e.,
(x, j) is not in E, for some j,
then create the message m = (x, r, R),
with R a randomly selected number;
add m into the set K;
broadcast m;
fi;

Step 2 : collect and add into K all arriving messages that have different nodes and the round number that equals r - there is s or s-1 such messages depending on whether x hasn't or has been enumerated in previous rounds (i.e., whether the "if" condition from Step 1 was or was not satisfied, respectively);

Step 3 : K1 := {(y, r, P) ∈ K: there is no message (y', r, P') ∈ K with y≠y' and P=P' };
if K1 ≠ ∅
then sort K1 with respect to the third (random) value of its messages
fi;

Step 4 : **for** all messages (y, r, P) in K1 **do**
if (y, r, P) is k-th in the order created by sorting K1 in Step 3
then insert the pair (y, N-s+k) into the set E
fi;
s := |K-K1|; K := ∅; K1:= ∅;

It is worth to note that, with the assumption that a message broadcast (Step 1) by a node eventually reaches all other nodes in the network (Step 2), the DrawingRound procedure always terminates.

4. Analysis of the algorithm

In this section we formulate some simple facts about the randomized enumeration algorithm defined in the previous section. These facts prove that the algorithm always properly terminates (i.e., it terminates achieving a proper enumeration). An estimation of the algorithm efficiency is also discussed.

Let $E|_1$ ($E|_2$) denotes the set of first (second) elements of pairs in the set E, i.e., $E|_1$ is the set of nodes already enumerated and $E|_2$ is the set of their enumeration labels. Assume also in the formulation of (a)-(e) below, that if E and s denote the values of these variables when entering the DrawingRound procedure, then E' and s' will denote their values after the DrawingRound termination.

Directly from the definition of the DrawingRound steps, we have the following simple properties:

(a) $E \subseteq E'$;
(one can only add elements to the set)
(b) $E|_1 \subseteq E'|_1 \subseteq X$;
(labels are only assigned to nodes of the considered network)
(c) $E|_2 = \{1, 2,..., N-s\}$;
(labels are properly chosen)
(d) $E'|_2 = E|_2 \cup \{N-s+1, ..., N-s+(s-s')\} =$
$= \{1, 2,..., N-s'\}$;
(labels cannot be repeated in two different rounds)
(e) $N-|E| = s \geq s' = N-|E'|$.

From the properties (a)-(d) above, we have a general invariant property

(f) $|E| = |E|_1| = |E|_2|$

which says that for every two pairs (x, l) and (x', l') in E, both $x \neq x'$ and $l \neq l'$.

Let us recall, that a finite set of nodes X is properly enumerated if to each element $x \in X$ a unique label $l(x) \in \{1, 2,..., |X|\}$ is assigned. From (a)-(d) and (f) above, we can conclude

Fact 4.1. The set E defines a proper enumeration of all nodes in $E|_1$.

Fact 4.2. Each time, entering the DrawingRound procedure, the value of the variable E is the same for all nodes of the enumerated network (each node has the same "enumeration knowledge").

The termination condition of the enumeration algorithm (repeat loop condition) together with Facts 4.1 an 4.2 result in

Fact 4.3. If the algorithm terminates, then the final value of E in each node defines the same proper enumeration of the network (it means also that computations in all nodes terminate by the same round, even though the nodes may be enumerated in different rounds).

Let [1 : M] be an integer interval from which random numbers are selected and M >> N. With this assumption and the property (e) above one can prove the following (details can be found in the Appendix):

Fact 4.4. The probability that the value of s is the same in the next round as it was in the previous one is M^{-s+1}.

The probability that the first (k-th) computation round is the last one is non-zero and grows while the value s decreases (see property (e) above and Appendix). Therefore, the probability that the whole enumeration algorithm never terminates, i.e., that it may have infinite number of rounds, equals 0.

Fact 4.5. The enumeration algorithm always terminates.

Fact 4.6. The expected number of rounds to terminate the enumeration algorithm is bounded by M/(M-N(N-1)).

The last result says that, with M >> N, it is almost certain that the algorithm terminates just in one round. For instance, if N=100 and M = 100000, then the above bound equals 10/9. All simulation experiments confirmed this "one round" result. In practice, it means that the time that is needed to reach the enumeration agreement is proportional to time necessary to spread a message in the network. In simulation, the random numbers were generated by a simple linear congruential method. The same method was used in all nodes. Each node generator had a different starting value assigned while initialized.

At last, we need not to add that, due to the probabilistic arguments, each node has equal chance to get anyone of the labels, i.e., none of the nodes is privileged by the algorithm.

5. Conclusions

A simple randomized enumeration algorithm has been presented. The power of the randomized approach (at least in this case) is in that information created (random number) locally by each node of a network needs only to be broadcast to all other nodes. No synchronization is required even if reduced to neighbor nodes as it is the case in [8]. The result is that the algorithm is truly distributed and works for all network configurations (graphs) and not only for a selected class of them (unambiguous graphs). It seems that the randomized approach leads usually to algorithms that are much simpler, easier to verify and, last but not least, much more efficient. Finally, it is worth to stress, that enumeration problem is a basis for many other distributed algorithm (one can say the same about the leader election algorithm) therefore its efficiency and feasibility is of great importance.

References

[1] M. Ben-Or, Randomized Agreement Protocols, *Lecture Notes in Com. Sci.* 448, 72-83 (1989)

[2] L. Bougé, On the Existance of Symmetric Algorithms to Find Leaders in Networks of Communicating Sequential Processes, *Acta Informatica* 25, 179-201 (1988)

[3] S. Budkowski, Estelle Development Toolset, *J. of Computer Networks and ISDN Systems*, vol. 25, No.1 (1992)

[4] P. Dembiński, Randomized Enumeration, Report 848, Institute of Computer Science of PAS (1998)

[5] P. Dembiński, Enumeration protocol in Estelle: an exercise in stepwise development, in S. Budkowski, A. Cavalli, E. Najm eds., Formal Description Techniques and Protocol Specification, Testing and Verification, Kluwer Academic Publishers, 147-162 (1998)

[6] ISO 9074, Information processing systems - Open Systems Interconnection - Estelle - A formal description technique based on an extended state transition model (1989)

[7] I.Litovsky, Y. Métivier, W. Zielonka, The power and limitations of local computations on graphs and networks, *Lecture Notes in Com. Sci.* 657, 333-345 (1993)

[8] A. Mazurkiewicz, Distributed Enumeration, *Information Processing Letters* 61, 233-239 (1997)

[9] M. Pease, R.Shostak, L. Lamport, Reaching Agreement in the Presence of Faults, *JACM 27*, 228-234 (1980)

[10] F. Vernadat, P. Azema, K. Drida, Distributed Coin Tossing, Proc. of the 14th Intern. Conference on Distributed Systems, IEEE Computer Society Press, 244-249 (1994)

Appendix

Drawing N numbers with replacement, one after another, from the set of M numbers, where M >> N, is called an *experiment*. A *successful experiment* is the one which results in all different N numbers. Because there are:

- M^N of N-element sequences from the set of M numbers, and
- $M(M-1) \ldots (M-N+1)$ of N-element sequences in which every number is different,

then

$$pse = M(M-1) \ldots (M-N+1)\, M^{-N} = M!/((M-N)!M^N) \quad (1)$$

is the probability of performing a successful experiment.

Consider now a sequence of single experiments in the above sense. The task is to estimate the expected number of experiments needed to achieve a successful one. It is easy to show that this expected number equals 1/pse (in fact, it is the expected value of the geometric distribution). We are going to show that

$1/pse < M/(M-N(N-1))$ or, equivalently,

$$pse > 1 - N(N-1)/M. \quad (2)$$

Assuming $M > N(N-1)$ we directly have

$$M(M-1) \ldots (M-N+1) > (M-(N-1))^N \quad (3)$$

Let us prove, by induction with respect to k, the following, general inequality:

$$(m-a)^k > m^k - akm^{k-1} \quad (4)$$

for all k=2,3,..., with m>a (by (3) and substituting m=M, a=N-1 and k=N, we obtain (2)). We can write (4) in yet another, equivalent form

$$(m-a)^k = m^k - akm^{k-1} + b_{a,m,k}, \quad (5)$$

$k=2,3,\ldots,b_{a,m,k} > 0$

For k=2, (5) obviously holds. Assume now, that (5) holds for some k>2. We have:

$$(m-a)^{k+1} = (m-a)(m^k - akm^{k-1} + b_{a,m,k})$$

$$= m^{k+1} - akm^k - am^k + a^2km^{k-1} + (m-a)b_{a,m,k}$$

$$= m^{k+1} - am^k(k+1) + b_{a,m,k+1},$$

where $b_{a,m,k+1} = a^2km^{k-1} + (m-a)b_{a,m,k} > 0$.

Hence, by inductive argument, (5) holds. Therefore, by the above reasoning, we obtained (2).