Efficient leader election among numbered agents

R. Chaturvedi, P. Dymond and M. Jenkin
Computer Science and Engineering, York University, Canada
(rahul, dymond, jenkin)@cse.yorku.ca

***Abstract*—When large groups of agents are tasked to a particular problem it can be useful to be able to elect a leader from among the agents. This leader can then be used to coordinate the agents to complete the task. But how can we elect a leader for a distributed collection of agents in an efficient manner? Here we present an algorithm for leader election for a collection of numbered (but unbounded) set of agents. We demonstrate a distributed probabilistic algorithm that the agents can use to elect a leader and to decide that the election is complete. Simulations demonstrate the efficacy of the approach.**

***Keywords*-leader election; robotics, infection; epidemics**

I. Introduction

Suppose that a group of distributed agents are required to complete some specific task. One way of ensuring that the work is performed correctly and distributed equitably is to establish some leader to distribute the work. This leader would then distribute the work among the agents and allocates resources and tasks to them. One way to go about choosing a leader is to preassign a certain agent in the population as the leader. But this approach may lead to fragile systems where separation or death of the designated 'leader' agent can lead to total system breakdown. And of course, a central authority must already exist to establish the leader. Given the disadvantages of the 'pre-established' leader approach, methods are required that enable a collection of agents to elect their own leader(s). The process of electing a single leader from a population of agents is called 'leader election'.

Leader election is a well-studied process, especially in collections of anonymous agents where the agents lack any uniquely identifying characteristic such as a serial or identification number. Population protocols (see[1]) were developed originally to model sensor networks in which agents had very limited resources and no control over their movements. Key features of the basic population protocol model introduced in [1] are

- Agents: The system consists of a large population of identical finite-state agents.
- Pairwise interactions: Agents do not send messages or share memory. Instead, an interaction between two agents updates both of their states according to a joint transition table. Interactions are atomic and independent of other activity within the collection of agents.
- Unpredictable interaction patterns: Individual agents have no control over their movement or interactions.
- Distributed inputs and outputs: The input to a population protocol is distributed across the initial state of the entire population. Similarly, the output is distributed to all agents.
- Convergence rather than termination: Agents converge to the correct value at the end of any computation, but may themselves be unaware that the computation is finished.

More formally a population protocol $\mathcal{A}(B, X, Y, I, O, \delta)$ consists of $B = \{b_1, b_2, ..., b_k\}$ a finite set of agent states, $X = \{x_1, x_2, ...x_m\}$ a finite set of input symbols, $Y = \{y_1, y_2, ..., y_p\}$ a finite set of output symbols, $I : X \rightarrow B$ an input function mapping inputs to initial states, $O : B \rightarrow Y$ an output function mapping states to outputs, and a transition relations δ, $B \times B \xrightarrow{\delta} B \times B$.

A population protocol operates on a population of finite size n. A population $\mathcal{P}$ consists of a set of n agents $A = \{a_1, a_2, ..., a_n\}$ together with an irreflexive relation $E \subseteq A \times A$ that we interpret as the directed edges of an interaction graph. E describes which agents may interact during the computation.

A population configuration is a mapping $C : A \rightarrow B$ specifying the state of each agent in the population. A computation is a sequence of configurations $C_0, C_1, ...$ such that every configuration C_i goes to a configuration C_{i+1} by a single interaction between a pair of agents in the population. An infinite computation is *fair* if for any two configurations C_i and C_{i+1}, where C_{i+1} is reachable via a single interaction from C_i, if C_i occurs infinitely often in the computation then C_{i+1} also occurs infinitely often in the computation. Here we are specifically interested in random interactions where each pair of agents is equally likely to meet in each interaction. We call this the uniform interaction model, and note that this constraint implies the "fairness" property above.

Angluin et al. developed a leader election algorithm for anonymous agents that takes $\Theta(n^2)$[1] interactions, assuming uniform interactions. The population protocol described in [1] assumes that the agents are anonymous. For many real world applications the agents are not anonymous but rather

[1]A function $f(n)$ is said to be $\Theta(n^k)$ if there exist positive constants c_1 and c_2 such that for sufficiently large n $c_1 n^k \leq f(n) \leq c_2 n^k$.

are equipped with unique serial or identification numbers. Can more efficient algorithms be developed for leader election in such applications?

II. Background

The concept of population protocols was introduced in [1] where the authors present a leader election algorithm with $\Theta(n^2)$ interactions. This algorithm is based on the idea that, assuming uniform interactions, eventually all agents will interact with each other and consensus will be reached regarding the identity of the leader. [2] proposed a protocol based on the use of epidemics to propagate information quickly through the population. In epidemic algorithms (e.g [3], [4]) an agent that wishes to disseminate a new piece of information sends the information to a set of peer agents, chosen at random. In turn, these peer agents forward the information to randomly selected agents, and so forth.

The community protocol model was introduced in [5]. Community protocols allow agents to be initially assigned unique identifiers from a large set, and to store a constant number of identifiers in their memory. These additional capabilities preserve much of the simple nature of the agents as in population protocols, but allow them to be uniquely identifiable. The transition rules cannot be dependent on the values of the identifiers: the identifiers can only be tested for equality with one another.

III. Leader election for numbered agents

We begin by observing that if the agents have unique identifiers and the agents know exactly the bounds of the identifiers then agent election can be performed in $O(1)$ steps. Suppose that the agents are numbered from 1 to n. Each agent i maintains copies of $leader_i$, the label(number) of the agent that agent i currently believes to be the leader; boolean $isLeader_i$, indicating that agent i currently believes itself to be the leader; and aid_i, agent i's unique number. Then in one step the entire set of agents can determine their leader following the trivial algorithm given in Algorithm 1.

```
Initially: For all agents $i$:
    $leader_i \leftarrow n$
    $isLeader_i \leftarrow false$
Agent $i$ executes the following
    $isLeader_i \leftarrow (aid_i = leader_i)$
```

Algorithm 1: Leader election for agents numbered $1 \dots n$.

Note that the election process is efficient as each agent knows the identification number of the eventual leader. Now suppose that this is not the case. For ease of exposition let us assume that the identifiers are drawn from the set of Natural numbers, and that the leader to be chosen is the agent with the highest aid_i value in this set.

Initially every agent is a candidate for leadership. Without additional information an agent cannot excuse itself from eventual leadership so initially $(leader_i, isLeader_i) = (aid_i, true)$. That is, each agent sees itself as the potential leader. When two agents interact the agent with the lower value of the leader variable replaces its leader value with the higher value of the leader variable. Using this method of infection or epidemics, the value of the highest identifier in the population is propagated among the agents. Assuming uniformly fair interactions among agents, the population elects a unique leader using this method. This process is described in Algorithm 2.

```
Initially: For all agents $i$:
    $leader_i \leftarrow aid_i$
    $isLeader_i \leftarrow true$
Interaction: Between agents $i$ and $j$
    if $leader_i > leader_j$ then
        $leader_j \leftarrow leader_i$
        $isLeader_j \leftarrow false$
    else
        $leader_i \leftarrow leader_j$
        $isLeader_i \leftarrow false$
    end
```

Algorithm 2: Electing a leader from a population of agents with identifiers drawn from the set of Natural numbers.

Assuming fair interactions, after a sufficient number of interactions only one agent will have a true value for $isLeader$. Unfortunately there is no deterministic way for the agents to know when the election is complete. One approach to determining probabilistically that the election process is complete is to use timer agents (e.g., [1]), but this will slow down the election process. In the following we analyze the complexity of the leader election algorithm, and explore properties of epidemics that help the agents detect the completion of the leader election process more efficiently.

Using the epidemic method from Algorithm 2, all agents can be expected to have the same $leader_i$ value in O($n \ln n$) interactions. This identifies the new leader of the population. But is it known that the election has completed, since individual agents do not have access to the value of n?

Call the agent with the highest identifier the eventual leader. Every agent with a $leader$ value equal to that of the eventual leader (although they do not know this) in the population is called a follower. (Using this definition, the eventual leader is also a follower.) Agents that are not followers are called free agents. The process of changing a free agent into a follower is called a conversion. Given that the agents are unaware of n, a technique is required that allows the eventual leader to estimate that the conversion of all agents is complete. We begin by proving certain properties of the epidemic:

Theorem 1. Using Algorithm 2 all agents are expected to have the same $leader_i$ value after a total of O($n \ln n$)

interactions.

Proof For simplicity assume that the n agents are numbered from 1 to n and that self-interaction of agents is allowed. Note that this last assumption can only increase the expected number of interactions in the population. Consider an interaction in the agent population when i agents are followers. Then the probability that this round will increase the number of followers by one is given by

$$\begin{aligned} P_{bi} &= p(\text{follower meets non-follower})+ \\ &\quad\ p(\text{non-follower meets follower}) \\ &= \tfrac{i}{n} * \tfrac{n-i}{n} + \tfrac{n-i}{n} * \tfrac{i}{n} \\ &= 2\tfrac{i}{n}\tfrac{n-i}{n}. \end{aligned}$$

Since the probability above is a geometric distribution, the expected number of rounds required to increase the number of followers by 1 is given by

$$E_{bi} = \frac{1}{P_{bi}} = \frac{n^2}{2i(n-i)}.$$

By linearity of expectation, summing over i provides the expected number of interactions required to convert all of the agents to followers, so

$$E_b = \sum_{i=1}^{n-1} E_{bi} = \sum_{i=1}^{n-1} \frac{n^2}{2i(n-i)}.$$

Thus,

$$E_b = \frac{n^2}{2}\sum_{i=1}^{n-1}\frac{1}{i(n-i)}$$

Simplifying (see [6] Ex. 7.35, attributed therein to T. Feder)

$$E_b = \frac{n^2}{2} * \frac{2}{n} H(n-1) < nH(n) = \Theta(n \ln n)$$

where $H(n-1)$ and $H(n)$ are the $n-1^{\text{th}}$ and n^{th} Harmonic numbers respectively.

Theorem 2. Using Algorithm 2 the eventual leader is involved in an expected $\Theta(\ln n)$ conversions.

Proof When the election is complete every agent will have become aware of the identification number of the eventual leader. Call the process of an agent becoming aware of the identification number of the eventual leader a conversion. Assuming fair interactions then the probability of the eventual leader being involved in the first conversion is 1, in the second conversion $\frac{1}{2}$, in the third conversion $\frac{1}{3}$ and so on. Therefore, the expected number of conversions that the eventual leader is involved in can be expressed as E_{lca}

$$E_{lca} = \sum_{i=1}^{n-1}\frac{1}{i} < H(n) < \Theta(\ln n).$$

Theorem 3. The eventual leader is expected to be involved in $\Theta(\ln n)$ non-conversion interactions with followers before all the free agents are converted.

Proof Consider an interaction in which there are i followers. The probability that the number of followers increases in this round is P_{bi}. The expected number of rounds required to increase the number of followers from i to $i+1$ is E_{bi} This is the expected number of non-conversion rounds that occur between the $(i-1)^{th}$ and i^{th} conversions.

The probability of a non-conversion round in which the eventual leader interacts with an already converted follower is P_{b2} is given by

$$\begin{aligned} P_{b2} &= p(\text{eventual leader meets a follower})+ \\ &\quad\ p(\text{follower meets the eventual leader}) \\ &= \tfrac{1}{n}\tfrac{i}{n} + \tfrac{i}{n}\tfrac{1}{n} = \tfrac{2i}{n^2} \end{aligned}$$

We want to find the number of times that the eventual leader is expected to encounter a follower between the $(i-1)^{th}$ and i^{th} conversions. This is $P_{b2} * E_{bi} = \frac{2i}{n^2}\frac{n^2}{2i(n-i)} = \frac{1}{n-i}$. Summing this over all i gives E_{lmf}, the expected number of times a leader sees one of its followers before all free agents are converted.

$$E_{lmf} = \sum_{i=1}^{n-1}\frac{1}{n-i} = H(n-1) < H(n) = \Theta(lnn).$$

Theorems 2 and 3 show that the expected number of conversions carried out by the eventual leader is approximately equal to the number of interactions between the eventual leader and followers. This property can be used by a potential leader to estimate if the election process is complete with a high degree of confidence. When a potential leader is involved in the same number of conversion and non-conversion interactions it can conclude that the election is probably complete.

In practice, rather than testing for equality it is desirable to ensure that the number of followers met exceeds the number of conversions by some margin. This buffer factor takes into account the large number assumptions in the order estimates (the expected size is $\Theta(\ln n)$ which ignores lower order terms. Algorithm 3 summarizes the basic process of leader election for numbered leaders with numbers drawn form $\mathcal{N}_0$. Algorithm 3 assumes large numbers of agents and sets the termination criteria based on a multiplicative buffer set by m. For smaller sized populations the termination test should also require that a minimum number of interactions has occurred.

IV. Experimental evaluation

Figure 1 shows results of experimental evaluation of the leader election and termination algorithm. Populations of various sizes from 1000 to 10000 were simulated and the leader election algorithm was run. The value used for m was 4. Because the algorithm declares that the election is complete probabilistically, this declaration may not always be correct. In the figure, blue points indicate the point at which the election was actually complete. Red points indicate the point at which the eventual leader agent actually determined that the election was complete. A solid line fits

```
Initially: For all agents $i$:
    $leader_i \leftarrow aid_i$
    $isLeader_i \leftarrow false$
    $electionComplete_i \leftarrow false$
    $conversions_i \leftarrow 0$
    $metFollowers_i \leftarrow 0$
Interaction: Between agents $i$ and $j$
    if $electionComplete_i$ or $electionComplete_j$ then
        $electionComplete_i \leftarrow true$
        $electionComplete_j \leftarrow true$
    else
        if $leader_i > leader_j$ then
            $leader_j \leftarrow leader_i$
            if $leader_i = aid_i$ then
                $conversions_i \leftarrow conversions_i + 1$
                if $conversions_i < m \times metFollowers_i$
                then
                    $isLeader_i \leftarrow true$
                    $electionComplete_i \leftarrow true$
                end
            end
        else if $leader_i < leader_j$ then
            $leader_i \leftarrow leader_j$
            if $leader_j = aid_j$ then
                $conversions_j \leftarrow conversions_j + 1$
                if $conversions_j < m \times metFollowers_j$
                then
                    $isLeader_j \leftarrow true$
                    $electionComplete_j \leftarrow true$
                end
            end
        else
            if $leader_i = aid_i$ then
                $metFollowers_i \leftarrow metFollowers_i + 1$
            end
            if $leader_j = aid_j$ then
                $metFollowers_j \leftarrow metFollowers_j + 1$
            end
        end
    end
```

Algorithm 3: Election with election termination.

the blue responses and a dashed line fits the red ones. For smaller values of n (under 3335) the algorithm terminated correctly in over 99.2 per cent of cases, improving to 99.8 per cent for the largest third of the populations simulated.

V. Conclusion

Leader election is a critical problem in a range of tasks from network controllers to autonomous agents. Elections have long been studied in collections of agents lacking unique identification numbers; however this model of agents may be limiting for many real world tasks in which simple agents are assigned unique numbers, such as MAC addresses or serial numbers. For such environments, we exhibit a probabilistic algorithm with very simple local computation for deciding that the election process has completed. The algorithm operates probabilistically in O($n \ln n$) interactions to both establish the leader of the collection of agents and decide that the election is complete.

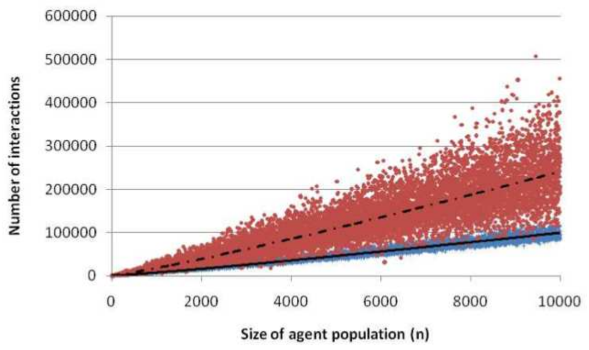

Figure 1. Election algorithm performance. For experiments with uniform random interactions we show the number of interactions required to elect a single leader is plotted for different population sizes. Blue points indicate when the election actually completed, fitted by the solid line. Red points indicate when the eventual leader agent declared the election complete, fitted by the dashed line.

Acknowledgments

The financial support of NSERC is gratefully acknowledged. The authors thank Eric Ruppert for helpful discussions.

References

[1] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta, "Computation in networks of passively mobile finite-state sensors," in *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of Distributed Computing*. New York: ACM, 2004, pp. 290–299.

[2] D. Angluin, J. Aspnes, and D. Eisenstat, "Fast computation by population protocols with a leader," in *DISC*, ser. Lecture Notes in Computer Science, S. Dolev and S. Dolev, Eds., vol. 4167. Berlin, Heidelberg: Springer, 2006, pp. 61–75.

[3] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal multicast," *ACM Trans. Comput. Syst.*, vol. 17, no. 2, pp. 41–88, May 1999.

[4] P. Eugster, R. Guerraoui, A. M. Kermarrec, and L. Massoulie, "From epidemics to distributed computing," *IEEE Computer*, vol. 37, no. 5, pp. 60–67, May 2004.

[5] R. Guerraoui and E. Ruppert, "Even small birds are unique: Population protocols with identifiers," Dept of Computer Science and Engineering, York University, Tech. Rep., September 2007. [Online]. Available: http://infoscience.epfl.ch/record/111565

[6] R. Graham, D. Knuth, and O. Patashnik, *Concrete mathematics*, 2nd ed. Reading, MA: Addison-Wesley, 1994.