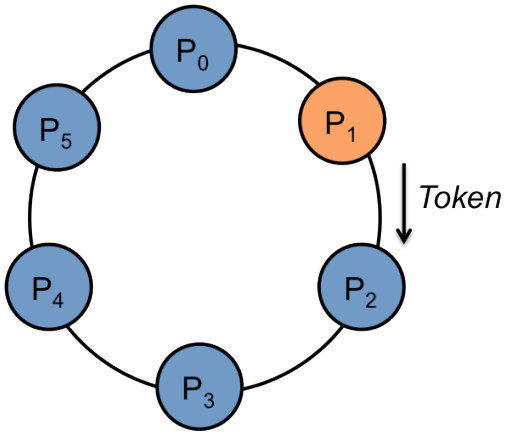
Ilea Cristian
Multiple Agent System
July 14, 2018

Distributed Ring Leader Election

1. Introduction

Ring leader election represents a problem where we can consider p processes that are connected to each other in a configuration where process p_i can communicate only with p_{i-1} and p_{i+1}. This in fact represents a graph that is shaped like a ring. The goal of the system is to decide who is the leader among all the processes. The main limitation is the ability of a process p to only communicate with it's neighboring processes.

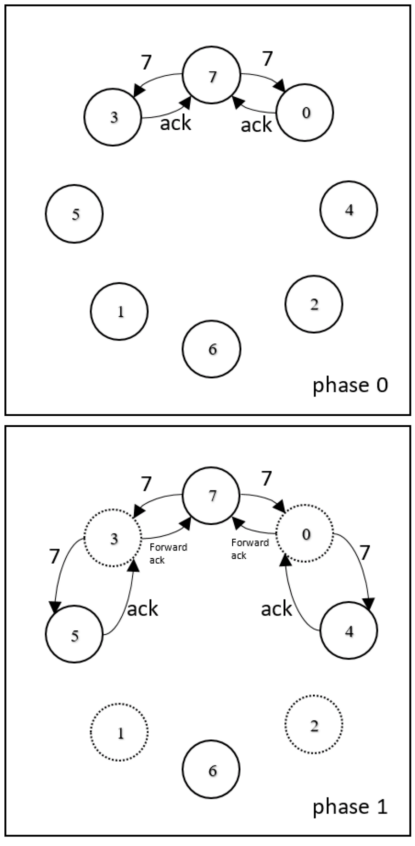
The distributed ring leader election problems comes from distributed systems where there are multiple peer-2-peer computers, machines, processes that need to communicate to decide upon a leader. The complexity comes from the fact that there is no main server coordinating all the actors involved. Thus, we can model the problem as a multiple agent system where each process is an agent that decides what to do by communicating with the neighboring agents.



2. How does it work

Let's consider the problem of finding the maximum number associated with each node among p process nodes. The naive approach would be to start with the first node and propagate its number as further as possible. The first process will send one message. The second one will pass two messages (his own number and the number of the first node). The first one will pass three messages and so on. We can see that this is inefficient.

A smarter approach is to let each process ask its left and right neighbor. If the left and right neighbor are both lower in value, then the two neighbor nodes become "inactive" — they will only pass messages. We now have a local maximum. After the first phase, we will already eliminate almost half the nodes. In the second phase we continue with the local maximum nodes and ask them to ask their neighbors at a 2-hop distance. The algorithm repeats until there is one single node that won the election.



3. Complexity and performance

The complexity of the naive approach will give us a very poor performance of $O(n^2)$ messages sent. The smarter approach presented in the previous section will give us a performance of $O(n \cdot log_2(n))$.

The beauty of the algorithm is that it can be easily parallelized. A distributed version of the ring leader election algorithm can run exactly the same way. The difference is that certain local maxima nodes will probably be at different phases in the election (they already asked neighbors more "hops" away) compared to other slower nodes (threads/processes). This is not an issue since it doesn't matter how slower or faster is a node. When two different nodes meet, one will win over the other and in the end there will remain a single winner.

We will now show some results of my Python implementation. The test was run on a ring graph with 5000 nodes with random numbers between 0 and 10000. Given our graph size the $O(n^2)$ version would have taken 25,000,000 messages. The $O(n \cdot log_2(n))$ version would take approximately ≈ 61438 steps.

We can see in the following results that making the algorithm distributed we roughly doubled the performance of the algorithm. Note: there were no improvements done for optimizing the number of threads to reduce overhead.

```
max node is [9998 -> 3540]
max is actually: 9998
synchronous took 13.740337133407593

max node is [9997 -> 1816]
max is actually: 9997
distributed took 6.68111777305603
```

4. Applications

In radio network protocols, leader election is often used as a first step to approach more advanced communication primitives, such as message gathering or broadcasts. The very nature of wireless networks induces collisions when adjacent nodes transmit at the same time; electing a leader allows to better coordinate this process.

Another area where leader election is useful is in peer-2-peer and torrent systems where different processes/machines need to agree on who has the most resources, bandwidth, the most data, etc.

It is part of a set of other tricky problems to tackle in the field of distributed system, and nowadays of more and more importance given that there is such a high increase in interconnected devices, computers, etc. IOT is also a domain where problems like this can be applied. Any system where concepts can be modeled as agents that interact is well suited for distributed solutions.

5. Bibliography

- Leader election https://en.wikipedia.org/wiki/Leader_election
- A modular technique for the design of efficient distributed leader finding algorithms https://dl.acm.org/citation.cfm?doid=77606.77610
- Lecture notes Washington University https://homes.cs.washington.edu/~arvind/cs425/lectureNotes/leader-6.pdf
- Lecture notes ETH Zurich https://disco.ethz.ch/courses/podc_allstars/references/stefan/class02mp-leader.pdf