1 Leader election in asynchronous undirected ring

The goal is to elect a leader in an undirected ring with n nodes that communicate in asynchronous rounds. We assume that processes do not carry any identity and that the size of the ring is known. We focus on communication with reliable channels but no message order preservation. Each node has a list of its neighbours in an arbitrary order.

We present a probabilistic leader election algorithm [1] based on an algorithm from Franklin [2], augmented with random identity selection, hop counters, and election round numbers modulo 2.

2 The algorithm

Consider a ring of processes $p_0, \ldots, p_{n-1}$ for $n > 1$. Each process is either active or passive. The idea is that we send a message from active processes to their neighbours. When an active process receives messages from both of its neighbours, it decides to stay active or to change to the passive state. In the end, there is only one active process remaining, hence it becomes the leader.

In our algorithm every process p_i maintains the following parameters:

- $id_i \in \{1, \ldots, k\}$ - represents the identity, not necessarily unique;
- $state_i \in \{active, passive, leader, nonleader\}$;
- $bit_i \in \{0, 1\}$ - represents the number of the current election round mod 2.

All messages are of the form (id, hop, bit), where

- id is the identity of the process that created the message;
- $hop \in \{1, \ldots, n\}$ is a counter which is increased by 1 every time the message is passed;
- bit represents the round of the process mod 2 (at the moment when the message was created).

Initially, for all processes p_i: $state_i = active$, $bit_i = 0$. The algorithm goes as follows:

1. At the start of each election round, an active process randomly selects an identity $id \in \{1, \ldots, k\}$ and sends the message $(id, 1, bit)$ in both directions.
2. Upon receipt of a message (id, hop, bit), a process p_i executes the following steps:
   (a) if $state_i$ is passive, then p_i passes the message $(id, hop+1, bit)$ in the same direction;
   (b) if $state_i$ is active and $bit_i \neq bit$, then p_i stores the message to process it in the next round;
   (c) if $state_i$ is active and $bit_i = bit$, then:
      i. if $hop = n$, then p_i becomes the leader;
      ii. if $hop < n$, then p_i waits for a message with the same value of bit from the other neighbour.
3. Upon receipt of messages with a proper value of bit from both directions, p_i checks if any message carries an id larger than id_i. If this is the case, then p_i becomes passive; otherwise, p_i starts a new election round by inverting bit_i, getting a new value of id_i and sending the message $(id_i, 1, bit_i)$ once again.
4. When a process becomes the leader, it sends this information as a special message in one direction. If a passive process receives a message from the leader, it passes it in the same direction, changes its own state to $nonleader$ and terminates.

3 Correctness and complexity

Theorem 1. *If channels are FIFO, the algorithm elects exactly one leader, even if processes and messages do not keep track of round number.*

Proof. We observe that if channels are FIFO, it is guaranteed that in each election round, a process receives messages from the current round since messages are created in the correct order and cannot change that order during communication via channels.

In each round, active processes with the largest id_i do not become passive. An active process can become a leader only if all other processes are passive. From this it follows that the algorithm elects exactly one leader. □

In our case channels are not FIFO, but we can still enforce FIFO behaviour of channels.

Theorem 2. *Between each pair of active processes p_1, p_2, there are exactly two messages m_1, m_2:*

- *if messages travel in opposite directions, p_1, p_2, m_1, m_2 carry the same bit;*
- *if messages travel in the same directions, p_1, p_2 carry opposite bits, as well as m_1 and m_2.*

Proof. First, we observe that there are exactly two messages between each pair of active nodes. When an active node receives two messages from neighbours, it can either stay active and send another two messages, or become passive without sending any other messages; thus, the invariant holds.

The rest of the theorem can be proven by analysing all possible scenarios of messages flow between three adjacent active processes. We will analyse one of the scenarios as all of them are similar to each other.

Assume we have three adjacent active processes p_1, p_2, p_3, where $b_i = 0$ for all three processes. Every process sends a message with $bit = 0$ to its neighbours. Based of the id of the messages, p_2 can stay active or become passive. If it becomes passive, there are two messages with $bit = 0$ between p_1 and p_3, where $bit_1 = bit_3 = 0$ as well. If p_2 stays active, it sends a message with $bit = 1$ to p_1 and p_3. In that case there are two messages between p_1 and p_2 with opposite bit traveling in the same direction, and $bit_1 = \neg bit_2$. The same holds for p_2 and p_3. □

Theorem 3. *In the probabilistic Franklin algorithm with no FIFO channels, exactly one leader has been elected.*

Proof. From Theorem 2 it follows that if there are two messages travelling in the same direction, they carry opposite bits. It means that an active process knows which message was created in the current round, so it can store the other message for next round. With this approach we can simulate FIFO channels, hence the theorem follows from Theorem 1. □

Theorem 4. *The probabilistic Franklin algorithm terminates with probability $p = 1$.*

Proof. With $x > 1$ active processes, they all remain active only if they choose the same identity all the time. The probability of selecting the same identity in one round is $\left(\frac{1}{k}\right)^{x-1} < 1$, so it tends towards 0 as the number of rounds gets larger. It means that there will be one active node remaining with probability $p = 1$. □

Theorem 5. *On average, the probabilistic Franklin algorithm takes $O(n \log n)$ messages.*

Proof. On average, in each round about half of the active nodes become passive, so there is $O(\log n)$ rounds. Each round takes $O(n)$ messages, hence the algorithm takes $O(n \log n)$ messages. □

References

[1] R. Bakhshi, W. Fokkink, J. Pang, J. van de Pol in Fifth Ifip International Conference On Theoretical Computer Science – Tcs 2008, (Eds.: G. Ausiello, J. Karhumäki, G. Mauri, L. Ong), Springer US, Boston, MA, **2008**, pp. 57–72.

[2] R. Franklin, "On an Improved Algorithm for Decentralized Extrema Finding in Circular Configurations of Processors", *Commun. ACM* **1982**, *25*, 336–337, `https://doi.org/10.1145/358506.358517`.