Leader election in an directed ring

Given n processes in a directed ring with synchronous communication, communicating only with message passing to its neighbors, the circular extrema-finding (or election) problem is to select a maximum process. Each process has a unique value in a set with a total order. These values may be transmitted and compared. All processes are identical (except for their value) i.e run the same algorithm and n, the number of processes, is not initially known.

Here we describe an algorithm presented by Peterson [1982], which achieves the above and uses at most $2n\lfloor \log n \rfloor + n$ messages in the worst case.

The algorithm

Definitions and assumptions

The algorithm will consist of electoral rounds.

We will distinguish 4 states a node can be in:

active — node is taking part in this electoral round

relay — node does not participate in election anymore and only passes messages

non-leader — node is not the leader

leader — node is the leader

Messages sent will consist of pairs of "value" and "status". Values will be from the set of unique identifiers of the nodes participating, while status can be either *active* or *leader*.

Description

Assume all messages are passed to the right. All processes are initially *active*. *Active* processes operate in phases. *Relay* processes in one phase just pass two at most messages they receive. The primary goal is to have the number of active processes be cut at least in haft during each phase. Each process starts with its own value as its temporary identifier (tid). During a phase, each *active* process receives the *tid* of its nearest *active* neighbor to the left and that neighbor's nearest left *active* neighbor's *tid*. If the first process sees that its left *active* neighbor has the largest of the three tids, then it sets its *tid* to that value and starts the next phase. Otherwise, it becomes a *relay* process.

Procedures

Each round starts with each *active* sending message (*tid*, *active*) to right neighbour. Afterwards, each *active* awaits to receive a message from left.

Let's denote message received from left as $(m1.ntid, m1.status)$. We need to *check* whether $m1.status$ is *leader* or $m1.ntid$ is equal to node's unique identifier. In first case we need set our state to *non-leader* and pass the message to right. In second case we need to declare this node as leader: set state to *leader* and send message with $(i, leader)$ to right. If *check* is failed then we await for one more message from left as $(m2.nntid, m2.status)$. Do the same *check* as previously, and if it's failed then there are two cases. In first case $m1.ntid \geq tid \wedge m1.ntid \geq m2.nntid$ we assign $m1.ntid$ to node's *tid*, otherwise we set the node state to *relay*.

Nodes that become *non-leader* are terminated. *Leader* finishes when it receives message with status *leader*.

Relay await for at most two messages from left processed one by one, for each message do *check* and if it's failed just pass the message to right.

Correctness

The correctness of the algorithm follows from two facts: during any phase there is exactly one process whose *tid* is the maximum value, and no process except the maximum will ever see its own initial value. The first fact comes by noting that the process whose *tid* is the maximum value is by definition larger than its two *active* neighbors, so the *active* process to its right will have its *tid* equal to the maximum

on the next phase. The second fact follows by first noting that, for any process, its value is either the *tid* value of the nearest *active* process to its right, or its value no longer is being propagated. It therefore follows that in order for some nonmaximum value of a process to cycle around the ring, the largest value must already have passed the process (lest the *active* process with *tid* equal to the maximum prevents passage of the lower value), but then the smaller value is no longer being propagated since the *tid* value to the right of the process is now the maximum.

Complexity analysis

It is very simple to see that at most $2n \log n + n$ messages are sent. First one notes that there can be at most $\lfloor \log n \rfloor$ phases to reach one *active* process. This follows from the fact that any *active* process survives a phase only if the *active* process to its left does not survive that phase. Hence of m *active* processes during a phase, at most $\lfloor \frac{m}{2} \rfloor$ will survive for $m > 1$. During a phase every process sends (and receives) two messages, and on the last phase the lone *active* process sends one message which is relayed around to the maximum for a total of $2n \log n + n$.

References

Gary L. Peterson. An $o(n \log n)$ unidirectional algorithm for the circular extrema problem. *ACM Trans. Program. Lang. Syst.*, 4(4):758–762, oct 1982. ISSN 0164-0925. doi: 10.1145/69622.357194. URL https://doi.org/10.1145/69622.357194.