Franklin's leader election in synchronized undirected ring

Introduction

Our goal is to elect a leader in synchronized circular configuration of processors, where every node has unique id and is able to send and receive messages to and from two adjacent nodes. In presented algorithm we will choose node with highest id.

The algorithm

Process stores:

- Id_i – unique process identifier
- $State_i \in$ {unknown, nonleader, leader}

Message contains:

- $Type \in$ {normal, ending}
- $Value$ – field of the same type as process id, filled with significant value only in normal message

Initialization

Process sets its state as unknown and sends messages (normal, Id_i) to both of its neighbours.

Nonleader state process i

Receives messages from both neighbours.
If any of them was of type ending, process sends it to the other neighbour and terminates.
Else it sends message received from left neighbour to right neighbour and vice versa.

Unknown state process i

Receives messages from both neighbours.
If any of them contains value greater than Id_i it sets $State_i$ as nonleader.
Else if any of them contains value equal to Id_i process sets $State_i$ as leader, sends messages (ending) to both of its neighbours and terminates.
Else process sends messages (normal, Id_i) to both of its neighbours.

Synchronized ring implementation details

In synchronized environment process must send messages in each round so in case where process changes $State_i$ from unknown to nonleader it additionally sends empty messages to both of its neighbours. Additionally unknown state processes send empty messages instead of (normal, Id_i) messages when they received an empty message from specific neighbour. Empty messages are ignored when unknown state processes compare message value to Id_i.

Correctness

It is clear to see that only one process can become a leader and when it happens all other processes are already marked as nonleaders. It is because process i sets $State_i$ as leader only when it received a message with Id_i and for it to happen this massage must have passed through all other nodes, which means that they were already nonleaders. On the other hand processes will gradually become nonleaders. It is because, until leader is elected, in each round there are normal messages being relayed between every two nearest unknown state processes. One of those processes have smaller id than the other. Because of that when it receives normal message it will become a nonleader.

Complexity analysis

We will measure complexity of algorithm as number of nonempty messages passed between processes. Let's say that nonempty messages are "created" by unknown state processes (nonleader processes only relay messages). We can see that when sum of message passes of created messages reaches 2N (where N is number of processes), so when every node created or relayed two nonempty messages, then every processor received nonempty messages send by nearest unknown state processes. Process in order to remain in unknown state and not change into nonleader must have higher id than received value in both of those messages. Because of that, number of unknown state processes will reduce by at least half after 2N message passes. Number of such reductions is at most log N, so when we consider that additional message passes at the end of algorithm are O(N), then complexity of the whole algorithm is O(N log N).

References

- R. Franklin On an Improved Algorithm for Decentralized Extrema Finding in Circular Configurations of Processors. Comm. ACM, 25, 5 (May 1979), 336-337