Franklin's leader election algorithm for bidirectional rings

Introduction

Franklin's algorithm discussed here performs leader election in bidirectional rings with asynchronous communication. Each process (node) has an unique identifier id known to itself. The pessimistic number of messages is at most $O(nlog_2 n)$.

The algorithm - description

At any time of the algorithm's execution, each process is either active, passive or a leader. There are either no leaders or exactly one leader at any given moment. Before an overview, let's define 'active neighbours'. By 'left/right active neighbour of node a' we mean the closest active node when traversing the ring in the direction specified by a's left/right link. Nodes may have different understanding of which side is 'left' and which 'right'; this is not an issue.

The algorithm is split into rounds. In each round an active process a sends two messages with its id to left and right active neighbours and receives two messages with $idLeft$ and $idRight$ from these nodes. If either of received identifiers is greater than a's id, then it becomes passive. If $idLeft == id$ or $idRight == id$, a becomes the leader. If neither of above conditions is met, a remains active.

Passive node a simply forwards messages in from either direction in the same direction. If a message is $final$, then a sets message's value as leader's identifier and terminates. It will become clear that an active node cannot receive a $final$ message. A leader node only sends a single $final$ message and terminates when it comes back (forwarded by other nodes, all passive).

The algorithm - pseudocode

Process a maintains:

- id_a — a unique identifier
- $leaderId$ — leader's id, must be set before termination
- $state_a \in \{active,\ passive,\ leader\}$

Message format:

- id — original sender's id
- $isFinal$ — true for the single message sent by leader and then forwarded; false otherwise

Procedures

Initialization: all processes are active and know their id's

Passive process: Upon receipt of a message msg , passes on the message. If $msg.isFinal$ is true, sets $leaderId$ to $msg.id$ and terminates.

Active process: In each phase a:

- sends message with id_a to its left and right neighbours,
- receives messages with id_{left} and id_{right} from its active neighbours;
- if $id_{left} > id_a$ or $id_{right} > id_a$, then a becomes passive,
- if $id_{left} == id_a$ or $id_{right} == id_a$, then a becomes a leader

Leader process: Leader process a sets $leaderId = i_a$ and sends message with id_a and $isFinal$ set to $true$ to its left or right neighbour. When it receives the message back, terminates.

Correctness

Correctness of Franklin's algorithm stems from the following observations:

- If there is a single *leader* node and other nodes are passive, all nodes will terminate with correct *leaderId* set. This is clear from the protocol.
- If there is a single *leader* node, all other nodes are passive. This is because in order to become a leader, node a has to receive a message with its id. Such message can only be sent by a (ad identifiers are unique). It can be received back only when it traversed entire ring, and this implies that all nodes other than a are passive.
- There can't be two leader nodes. This fact follows from the above point.
- If there are at least two active processes in a given round, at least one of them must become passive at the end of this round. If this was not true, each of the active nodes would have id bigger than its neighbour identifiers, which is impossible.
- The algorithm is deadlock-free. This is clear from the protocol.

Complexity analysis

Let n be the size of the ring. In each round, at least half of the active nodes become passive, as in order to remain active a node must have id greater than its active neighbours identifiers. Therefore, after at most $FLOOR(log_2 n)$ steps there is only one active process. When only one process a is active, it will send two messages which will be forwarded by the other nodes and come back to a - $2n$ message passes. Then a becomes leader and after the next n passes of the final message, the algorithm terminates. As there were exactly $2n$ message passes in any round with $k \geq 2$ active nodes, we conclude that the time complexity is:

$$2n * FLOOR(log_2 n) + 3n \in O\left(nlog_2 n\right)$$

It is stated in [1] that average number of messages passed was $O\left(n\right)$ in sample tests.

References

[1] Randolph Franklin. "On an improved algorithm for decentralized extrema finding in circular configurations of processors". In: *Communications of the ACM* 25.5 (1982), pp. 336–337.