Computer Science and Engineering Distributes Systems CTH HT11
Chalmers and Göteborg University TDA596
Göteborg GU-INN240
SAA

Election using Echo Algorithm

The Election Algorithm algorithm using Echo Algorithm should be implemented within the Netsim program.

General

When giving a `public void trigg()` to a node it should start a version of the algorithm.

When a node gets a request that is sent further it should also call `setWaken();` on the node interface. This will change the color of the node on the screen.

When a node has got all echoes and sends an echo back on its first link it should also call `setReady();` on the node interface.

The messages should have a tag with its starting sender identification and type of message (`Explorer`/`Echo`).

For each event a node should write to `public void writeLogg(String row);` on the node interface in order to trace the events. Then the real physical order of events can be viewed by selecting view -> system log on the menu bar.

N.B. The Echo Algorithm requires a network that is safe and FIFO but **not** that it will be fully connected.

The following holds.

- Each node has a unique identification.
- The identification can be ordered totally.
- each node has a variable *Superior* having the value of the largest known identity. From the beginning it has the node's own identity.
- The algorithm is a *multi-source* echo algorithm. When a node starts its version of the algorithm it puts its own identity in the *explorer messages*.
- One (or several) node starts the algorithm. When a node with a higher identity get its explorer messages it starts its own version of the algorithm and ignores the echo algorithms with lower identity. Only the node with the highest identity will see its echo algorithm terminate normally. Then it is elected!
- Any node can start the algorithm and then becomes Initiator Node, IN, in an echo algorithm which will be identified with the nodes *Name*.

Description of the algorithm

1. One node starts the algorithm by sending out *EM* to all its neighbors. These messages contains the node's identity. The node sets its *Superior* variable to its own identity.

2. If an inactive node gets an *EM* that has a value lower than the node's own identity, this message is ignored. Instead it starts a new echo algorithm as described in 1.

3. If an active node gets an *EM* that has a value lower than the node's *Superior* variable, this message is ignored.

4. If an active or an inactive node gets an *EM* that has a value higher than the node's *Superior* variable, the node updates its *Superior* variable. The incoming link is marked as *FL* and any former existing *FL* is unmarked. Then the node sends out *EM* on all links except the *FL*. These messages contains the value of the *Superior* variable.

5. If an active node gets an *EM* that has a value equal to its *Superior* variable it sends an *ECHO* message back on the corresponding link. This message contains the value of the *Superior* variable.

6. If a node gets an *ECHO* message that has a value lower than the node's *Superior* variable, this message is ignored.

7. If a node gets an *ECHO* message that has a value same as the node's *Superior* variable, it is book-kept.
   When a node has got a *ECHO* message with the same value as the *Superior* variable on each link except the *FL* it sends a corresponding *ECHO* message on its *FL*.
   If the node's identity is the same as the *Superior* variable the algorithm terminates and the node is elected.

8. If a node gets an *ECHO* message that has a value higher than the node's *Superior* variable, indicates that there is a programming error!

Please note that a leaf node (that only has one neighbor) does not send *EM* messages but instead will send an *ECHO* reply at once!