Lecture Exercise

A leader Election Algorithm

Consider the following distributed algorithm. Assume a set of process connected in a ring. Let us call the processes $p[0], p[1], \ldots, p[N-1]$. The processes are connected by FIFO channels in one direction around the ring. The channel from $p[i]$ to $p[i+1]$ is called $chan[i]$. Each process $p[i]$ has a unique identity $id[i]$, which is a natural number. To solve certain problems in a distributed system, the processes must agree on a unique process to coordinate activity. The problem of electing exactly one process as a coordinator is referred to as the *leader election problem.* The following paragraph is an informal description of an algorithm, due to LeLann for electing one of the processes as a leader.

Initially all processes are idle, i.e., not interested in electing a leader. The algorithm starts when one or several of the processes becomes interested in electing a leader. A process $p[i]$ may then either

- first try to elect a leader (the process is then called an *initiator*), in which case it sends its identity in a message over $chan[i]$. Thereafter the process reads incoming identities from $chan[i-1]$. Each such identity is stored by $p[i]$ and forwarded over $chan[i]$. When the process receives its own identity from $chan[i-1]$ it does not forward the identity. Instead the process compares all identities received so far. The least of these identities is the identity of the elected leader.

- receive a message over $chan[i-1]$ before getting the idea of starting an election. In this case $p[i]$ does not become an initiator, and only forwards identities from $chan[i-1]$ to $chan[i]$ and does not bother about any other things.

After the algorithm has finished, the elected process could for instance broadcast information about himself to the other processes in the system.