```
-module(gms3).
-export([start/1, start/2]).
-define(timeout, 1000).
-define(arghh, 100).

start(Name) ->
    Self = self(),
    spawn_link(fun()-> init(Name, Self) end).

init(Name, Master) ->
    leader(Name, Master, [], 1 ✔ ).


start(Name, Grp) ->
    Self = self(),
    spawn_link(fun()-> init(Name, Grp, Self) end).

init(Name, Grp, Master) ->
    Self = self(),
    Grp ! {join, Self},
    receive
        {view, Leader, Slaves, N} = NewLast ->
            Master ! joined,
            Ref = erlang:monitor(process, Leader), ✔
            slave(Name, Master, Leader, Slaves, Ref, N+1 ✔ , NewLast ✔ )
    after ?timeout -> ✔
            Master ✔ ! {error, "no reply from leader"}
    end.

leader(Name, Master, Slaves, N) ->
    receive
        {mcast, Msg} ->
            bcast(Name, {msg,Msg,N} ✔ , Slaves ✔ ),
            Master ! {deliver,Msg}, ✔
            leader(Name, Master, Slaves, N+1 ✔ );
        {join, Peer} ->
            NewSlaves = lists:append(Slaves, [Peer]),
            bcast(Name, {view,self(),NewSlaves,N} ✔ , NewSlaves ✔ ),
            leader(Name, Master, NewSlaves ✔ , N+1 ✔ );
        stop ->
            ok;
        Error ->
            io:format("leader ~s: strange message ~w~n", [Name, Error])
    end.

bcast(Name, Msg, Nodes) ->
    lists:foreach(fun(Node) -> Node ! Msg, crash(Name, Msg) end, Nodes).

crash(Name, Msg) ->
    case rand:uniform(?arghh) of
        ?arghh ->
            io:format("leader ~s CRASHED: msg ~w~n", [Name, Msg]),
            exit(no_luck);
        _ ->
            ok
    end.

slave(Name, Master, Leader, Slaves, Ref, N, Last) ->
    receive
        {mcast, Msg} ->
            Leader ! {mcast, Msg}, ✔
            slave(Name, Master, Leader, Slaves, Ref ✔ , N ✔ , Last ✔ );
        {join, Peer} ->
```

```
            Leader ! {join, Peer},
            slave(Name, Master, Leader, Slaves, Ref , N , Last );
        {msg, _, I} = NewLast when I < N ->
            slave(Name, Master, Leader, Slaves, Ref , N , Last );
        {msg, Msg, N} = NewLast ->
            Master ! {deliver, Msg},
            slave(Name, Master, Leader, Slaves, Ref , N+1 , NewLast );
        {view, _, _, I} = NewLast when I < N ->
            slave(Name, Master, Leader, Slaves , Ref , N , Last );
        {view, Leader, NewSlaves, N} = NewLast ->
            slave(Name, Master, Leader, NewSlaves , Ref , N+1 ,
NewLast );
        {view, NewLeader, NewSlaves, N} = NewLast ->
            erlang:demonitor(Ref, [flush]),
            NewRef = erlang:monitor(process, NewLeader),
            slave(Name, Master, NewLeader , NewSlaves , NewRef , N+1 ,
NewLast );
        {'DOWN', _Ref, process, Leader, _Reason} ->
            election( Name , Master , Slaves , N , Last );
        stop ->
            ok;
        Error ->
            io:format("slave ~s: strange message ~w~n", [Name, Error])
    end.

election(Name, Master, Slaves, N, Last) ->
    Self = self(),
    case Slaves of
        [Self|Rest] ->
            bcast(Name,Last,Rest),
            bcast(Name,{view,Self,Rest,N},Rest),
            leader( Name , Master , Rest , N+1 );
        [NewLeader|Rest] ->
            NewRef = erlang:monitor(process,NewLeader),
            slave( Name , Master , NewLeader , Rest , NewRef , N ,
Last )
    end.
```