```
-module(gms2).
-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, []).

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} ->
            Master ! joined,
            Ref = erlang:monitor(process, Leader), ✓
            slave(Name, Master, Leader, Slaves, Ref)
    after ?timeout -> ✓
            Master ✓ ! {error, "no reply from leader"}
    end.

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

% bcast procedure for MS4
bcast(_, Msg, Nodes) ->
    lists:foreach(fun(Node) -> Node ! Msg end, Nodes).

% bcast procedure for MS5
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) ->
    receive
        {mcast, Msg} ->
            Leader ! {mcast, Msg}, ✓
            slave(Name, Master, Leader, Slaves, Ref ✓ );
        {join, Peer} ->
            Leader ! {join, Peer}, ✓
            slave(Name, Master, Leader, Slaves, Ref ✓ );
        {msg, Msg} ->
            Master ! {deliver, Msg}, ✓
            slave(Name, Master, Leader, Slaves, Ref ✓ );
        {view, Leader, NewSlaves} ->
            slave(Name, Master, Leader, NewSlaves ✓ , Ref ✓ );
        {view, NewLeader, NewSlaves} ->
            erlang:demonitor(Ref, [flush]),
            NewRef = erlang:monitor(process, NewLeader), ✓
            slave(Name, Master, NewLeader ✓ , NewSlaves ✓ , NewRef ✓ );
        {'DOWN', _Ref, process, Leader, _Reason} ->
            election( Name ✓ , Master ✓ , Slaves ✓ );
        stop ->
            ok;
        Error ->
            io:format("slave ~s: strange message ~w~n", [Name, Error])
    end.

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