```
{
` client-states,fields and functions
VOTING:1;
JOINED:2;
UNKNOWN:3;
WAITING:4;
_ref:1;
_state:2;
_name:3;
_lasttimer:4;
_continue:5;

clients:[];
mapover(lst,L)::if (!(lst~empty),
        {L(car(lst));
        mapover(cdr(lst),L)});
map(lambda(client))::mapover(clients,lambda);
mapoverstate(state,lambda(client))::
        {pred(client)::if(client[_state]=state,lambda(client));
        mapover(clients,pred)};
newclient(ref)::[ref,UNKNOWN,"",0,void];
getclient(ref)::
        {
        result:void;
        map(if (client[_ref]~ref,result:=client));
        if (is_void(result),clients:=[result:=newclient(ref),clients]);
        result
        };

`mogelijke messages en state-guarding
guard:void;
timerid:1;
join(ref,name)::guard.__join(getclient(ref),name);
mesg(ref,msg)::guard.__mesg(getclient(ref),msg);
okay(ref,id)::guard.__okay(getclient(ref),id);
contwait(ref,cw)::guard.__contwait(getclient(ref),cw);
```

```
mesg_timeout(id)::guard.__mesg_timeout(id);
vote_timeout(id)::guard.__vote_timeout(id);

`logging
serverlog(txt)::display(txt+eoln);
clientlog(client,txt)::serverlog(text(client[_ref])+" / "+client[_name]+":
"+txt);
ignore(client,txt)::clientlog(client,"ignoring "+txt);

`REPEATING-mode
repeat_queuelast:repeat_queue:[];
repeating_mode()::
    {
    enter()::
        {
        callmsg(cell)::__mesg(cell[1],cell[2]);
        guard:=clone();
        serverlog("---- Repeating Mode ----");
        mapover(repeat_queue,callmsg);
        repeat_queuelast:=repeat_queue:=[]
        };
    __vote_timeout(id)::void;
    __contwait(client,cw)::void;
    __mesg(client,msg)::if (client[_state]=JOINED,
        {
        timerid:=timerid+1;
        mapoverstate(JOINED,
            {clientref:client[_ref];
            clientref.mesg(msg,timerid)});
        starttimer(10,mesg_timeout,timerid);
        clientlog(client,"repeating msg "+text(msg))
        }, ignore(client,"mesg "+text(msg)));
    __join(client,name)::if (client[_state]=UNKNOWN,
        {
        client[_state]:=JOINED;
        client[_name]:=name;
```

```
                clientlog(client,"has joined")
                });
        __okay(client,id)::if (client[_state]=JOINED,
                {
                clientlog(client,"said okay");
                client[_lasttimer]:=id
                });
        __mesg_timeout(id)::
                {
                timeouts:false;
                mapoverstate(JOINED,if (client[_lasttimer]<id,
                        {
                        clientref:client[_ref];
                        clientref.disconnect();
                        client[_state]:=WAITING;
                        clientlog(client,"disconnected");
                        timeouts:=true
                        }));
                if (timeouts,voting_mode())
                };
        enter()
        };

`VOTING-mode
` before timeout -- we wait until everybody says continue
` after timeout -- we continue unless somebody said wait
voting_mode()::
        {
        enter()::
                {
                mapoverstate(WAITING,
                        mapoverstate(JOINED,
                                {voter_ref:client[_ref];
                                voter_ref.vote()
                                }));
                mapoverstate(JOINED,
            {client[_state]:=VOTING;
            client[_continue]:=void});
        starttimer(10,vote_timeout,timerid:=timerid+1);
        guard:=clone();
        serverlog("---- Voting Mode ----")
        };
    continuecheck()::
        {
        ` check continue without waiting votes
        continuewithoutwait:true;
        mapoverstate(VOTING,if (is_void(client[_continue]),
            continuewithoutwait:=false,
            continuewithoutwait:=continuewithoutwait &
client[_continue]));
        ` check whether there are waiting others
        waitingothers:false;
        mapoverstate(WAITING,waitingothers:=true);
        serverlog("continue voted ? ["+text(continuewithoutwait)+"]
waiting others ? ["+text(waitingothers)+"]");
        if ((continuewithoutwait | !waitingothers),
            {
            ` remove voting states and waiting states and back to
repeating mode
            mapoverstate(WAITING,client[_state]:=UNKNOWN);
            mapoverstate(VOTING,client[_state]:=JOINED);
            repeating_mode()
            })
        };
    __okay(client,id)::if (client[_state]=JOINED,
        {
        clientlog(client,"said okay");
        client[_lasttimer]:=id
        });
    __contwait(client,cont)::
        {
        if (client[_state]=VOTING,client[_continue]:=cont);
        continuecheck()
        };
    __mesg(client,msg)::if ((client[_state]=JOINED) |
(client[_state]=VOTING),
        {
        if (repeat_queuelast~empty,
repeat_queue:=repeat_queuelast:=[[client,msg],[]],
            {
            repeat_queuelast[2]:=[[client,msg],[]];
            repeat_queuelast:=repeat_queuelast[2]
            });
        clientlog(client,"queueing message message")
        }, clientlog(client,"not joined or connected, ignoring message"));
    __join(client,name)::
        if (client[_state]=WAITING,
            {
            client[_state]:=JOINED;
            client[_name]:=name;
            clientlog(client,"has reconnected");
            continuecheck()
            },
        if (client[_state]=UNKNOWN,
            {
            client[_state]:=JOINED;
            client[_name]:=name;
            clientlog(client,"has joined")
            }));
    __mesg_timeout(id):serverlog("ignoring old message timout");
    __vote_timeout(id)::if (id=timerid,
        {
        map(if (is_void(client[_continue]),client[_continue]:=true));
        continuecheck()
        });
    enter()
    };
```

```
repeating_mode()
}
```