```
MODULE bully
EXTENDS TLC, Integers, FiniteSets, Randomization

CONSTANT PeersAmount

ASSUME PeersAmount ∈ Nat \ {0, 1}

IDS ≜ 1 .. PeersAmount

  --algorithm bully
variables
    failed_leader = PeersAmount,

    initiator ∈ IDS \ {failed_leader},

     Some coordinated peers also can fail
    n ∈ 0 .. Cardinality(IDS \ {failed_leader, initiator}),
    others_who_failed = RandomSubset(n, IDS \ {failed_leader, initiator}),

     Channel buffers between each pair of peers
    channels = [sender ∈ IDS ↦ [receiver ∈ IDS \ {sender} ↦ ""]],

     Current leader for each peer
    leader = [id ∈ IDS ↦ failed_leader] ;

define
    IDSBiggerThan  ≜ [id_1 ∈ IDS ↦ {id_2 ∈ IDS : id_2 > id_1}]

    IDSSmallerThan ≜ [id_1 ∈ IDS ↦ {id_2 ∈ IDS : id_2 < id_1}]

    IDSBiggerThanExceptFailedLeader ≜
        [id ∈ IDS ↦ IDSBiggerThan[id] \ {failed_leader}]

     We don't need to wait for a timeout because we already know
     which peers can't answer our requests.
    DoesNotReceiveAnyResponse(id) ≜
        IDSBiggerThanExceptFailedLeader[id] \ others_who_failed = {}

     Peers, that declared themselves as leaders to the receiver
    NewLeaders(receiver) ≜
        {sender ∈ IDS \ {receiver} : channels[sender][receiver] = "Leader"}

     We don't need to wait for a timeout because we already know
     which peers can't answer our requests.
    DoesNotReceiveOKResponseFromNewLeaders(receiver) ≜
        LET old_leader ≜ leader[receiver] IN
        ∃ new_leader  ∈ IDSBiggerThanExceptFailedLeader[receiver] :
         ∧ new_leader ∉ others_who_failed
         ∧ IF old_leader = failed_leader THEN new_leader > old_leader ELSE TRUE
     Peers, that have sent "Election" requests to the receiver
    $ElectionInitiators(receiver) \triangleq$
        $\{sender \in IDS \setminus \{receiver\} : channels[sender][receiver] = \text{"Election"}\}$

     Peers, that have sent any messages to the receiver
    $MessageSenders(receiver) \triangleq$
        $\{sender \in IDS \setminus \{receiver\} : channels[sender][receiver] \neq \text{""}\}$

    $FailedIDS \triangleq others\_who\_failed \cup \{failed\_leader\}$

    $WorkingIDS \triangleq IDS \setminus FailedIDS$

    $IDThatShouldBecomeNewLeader \triangleq$
        $\text{CHOOSE } new\_leader \in WorkingIDS :$
        $\forall id \in WorkingIDS \setminus \{new\_leader\} : new\_leader > id$

    $AllWorkingIDSAreCoordinatedByNewLeader \triangleq$
        $\forall id \in WorkingIDS : leader[id] = IDThatShouldBecomeNewLeader$

    $EventuallySolved \triangleq \Box\Diamond AllWorkingIDSAreCoordinatedByNewLeader$

**end define** ;

**fair process** $Peer \in IDS$
**begin**
$Initialize$:
    **if** $self \in FailedIDS$ **then**
        **goto** $Failed$ ;
    **elsif** $self = initiator$ **then**
        **goto** $BecomeLeaderOrStartElection$ ;
    **else**
        **goto** $NormalExecution$ ;
    **end if** ;

   If there are no peers with ID bigger than this peer has, then he himself becomes
   the new leader and sends "Leader" requests to all the other peers.
   Otherwise, the peer sends "Election" messages to peers that have bigger IDs.
$BecomeLeaderOrStartElection$:
    **if** $IDSBiggerThanExceptFailedLeader[self] = \{\}$ **then**
        $leader[self] := self$ ||
        $channels[self] :=$
        $[receiver \in \text{DOMAIN } channels[self] \mapsto$
            $\text{IF } receiver \in IDSSmallerThan[self] \text{ THEN } \text{"Leader" ELSE ""}]$ ;
        **goto** $NormalExecution$ ;
    **else**
        $channels[self] :=$
        $[receiver \in \text{DOMAIN } channels[self] \mapsto$
            $\text{IF } receiver \in IDSBiggerThan[self] \text{ THEN } \text{"Election" ELSE ""}]$ ;
```

```
    end if ;

  If nobody responses to "Election" requests (timeout), then this peer becomes
  the new leader and sends "Leader" requests to all the other peers.
CheckElectionTimeout:
    if DoesNotReceiveAnyResponse(self) then
        leader[self] := self ||
        channels[self] :=
        [receiver ∈ DOMAIN channels[self] ↦
            IF receiver ∈ IDSSmallerThan[self] THEN "Leader" ELSE ""] ;
        goto NormalExecution ;
    end if ;

  Receives "OK" responses from proclaimed leaders until it reachers timeout.
CheckOkTimeout:
    if DoesNotReceiveOKResponseFromNewLeaders(self) then
        goto NormalExecution ;
    end if ;
AcceptNewLeader:
    with new_leader ∈ NewLeaders(self) do
        leader[self] := new_leader ||
        channels[new_leader][self] := "" ;
        goto CheckOkTimeout ;
    end with ;

  If the peer receives "Election" request, he sends "OK" response and then acts like initiator
NormalExecution:
    with sender ∈ MessageSenders(self) do
        if channels[sender][self] = "Election" then
            channels[self][sender] := "OK" ||
            channels[sender][self] := "" ;
            goto BecomeLeaderOrStartElection ;
        elsif channels[sender][self] = "Leader" then
            leader[self] := sender ||
            channels[sender][self] := "" ;
            goto NormalExecution ;
        else
            channels[sender][self] := "" ;
        end if ;
    end with ;

Failed:
    skip ;
end process ;

end algorithm
```

```
BEGIN TRANSLATION ($chksum(pcal)$ = "499ab1c7" $\land$ $chksum(tla)$ = "70ed46e0")
VARIABLES $failed\_leader$, $initiator$, $n$, $others\_who\_failed$, $channels$, $leader$,
          $pc$

define statement
$IDSBiggerThan \triangleq [id\_1 \in IDS \mapsto \{id\_2 \in IDS : id\_2 > id\_1\}]$

$IDSSmallerThan \triangleq [id\_1 \in IDS \mapsto \{id\_2 \in IDS : id\_2 < id\_1\}]$

$IDSBiggerThanExceptFailedLeader \triangleq$
    $[id \in IDS \mapsto IDSBiggerThan[id] \setminus \{failed\_leader\}]$


$DoesNotReceiveAnyResponse(id) \triangleq$
    $IDSBiggerThanExceptFailedLeader[id] \setminus others\_who\_failed = \{\}$


$NewLeaders(receiver) \triangleq$
    $\{sender \in IDS \setminus \{receiver\} : channels[sender][receiver] =$ "Leader"$\}$


$DoesNotReceiveOKResponseFromNewLeaders(receiver) \triangleq$
    LET $old\_leader \triangleq leader[receiver]$ IN
    $\exists\, new\_leader \in IDSBiggerThanExceptFailedLeader[receiver] :$
     $\land\ new\_leader \notin others\_who\_failed$
     $\land$ IF $old\_leader = failed\_leader$ THEN $new\_leader > old\_leader$ ELSE TRUE


$ElectionInitiators(receiver) \triangleq$
    $\{sender \in IDS \setminus \{receiver\} : channels[sender][receiver] =$ "Election"$\}$


$MessageSenders(receiver) \triangleq$
    $\{sender \in IDS \setminus \{receiver\} : channels[sender][receiver] \neq$ ""$\}$

$FailedIDS \triangleq others\_who\_failed \cup \{failed\_leader\}$

$WorkingIDS \triangleq IDS \setminus FailedIDS$

$IDThatShouldBecomeNewLeader \triangleq$
    CHOOSE $new\_leader \in WorkingIDS :$
    $\forall\, id \in WorkingIDS \setminus \{new\_leader\} : new\_leader > id$

$AllWorkingIDSAreCoordinatedByNewLeader \triangleq$
    $\forall\, id \in WorkingIDS : leader[id] = IDThatShouldBecomeNewLeader$

$EventuallySolved \triangleq \Box\Diamond AllWorkingIDSAreCoordinatedByNewLeader$


$vars \triangleq \langle failed\_leader, initiator, n, others\_who\_failed, channels, leader,$
```

```
        $pc \rangle$

$ProcSet \triangleq (IDS)$

$Init \triangleq$  Global variables
        $\land failed\_leader = PeersAmount$
        $\land initiator \in IDS \setminus \{failed\_leader\}$
        $\land n \in 0 \,..\, Cardinality(IDS \setminus \{failed\_leader, initiator\})$
        $\land others\_who\_failed = RandomSubset(n, IDS \setminus \{failed\_leader, initiator\})$
        $\land channels = [sender \in IDS \mapsto [receiver \in IDS \setminus \{sender\} \mapsto \text{“”}]]$
        $\land leader = [id \in IDS \mapsto failed\_leader]$
        $\land pc = [self \in ProcSet \mapsto \text{“Initialize”}]$

$Initialize(self) \triangleq \land pc[self] = \text{“Initialize”}$
                      $\land$ IF $self \in FailedIDS$
                           THEN $\land pc' = [pc$ EXCEPT $![self] = \text{“Failed”}]$
                           ELSE $\land$ IF $self = initiator$
                                     THEN $\land pc' = [pc$ EXCEPT $![self] = \text{“BecomeLeaderOrStartElection”}]$
                                     ELSE $\land pc' = [pc$ EXCEPT $![self] = \text{“NormalExecution”}]$
                      $\land$ UNCHANGED $\langle failed\_leader, initiator, n,$
                                      $others\_who\_failed, channels, leader \rangle$

$BecomeLeaderOrStartElection(self) \triangleq \land pc[self] = \text{“BecomeLeaderOrStartElection”}$
                                      $\land$ IF $IDSBiggerThanExceptFailedLeader[self] = \{\}$
                                           THEN $\land \land channels' = [channels$ EXCEPT $![self] = [receiver$
                                                                                                    IF $re$
                                                  $\land leader' = [leader$ EXCEPT $![self] = self]$
                                                $\land pc' = [pc$ EXCEPT $![self] = \text{“NormalExecution”}]$
                                           ELSE $\land channels' = [channels$ EXCEPT $![self] = [receiver \in$
                                                                                                    IF $recei$
                                                $\land pc' = [pc$ EXCEPT $![self] = \text{“CheckElectionTimeout”}$
                                                $\land$ UNCHANGED $leader$
                                      $\land$ UNCHANGED $\langle failed\_leader, initiator,$
                                                      $n, others\_who\_failed \rangle$

$CheckElectionTimeout(self) \triangleq \land pc[self] = \text{“CheckElectionTimeout”}$
                                $\land$ IF $DoesNotReceiveAnyResponse(self)$
                                     THEN $\land \land channels' = [channels$ EXCEPT $![self] = [receiver \in$ DOM
                                                                                              IF $receiver \in$
                                            $\land leader' = [leader$ EXCEPT $![self] = self]$
                                          $\land pc' = [pc$ EXCEPT $![self] = \text{“NormalExecution”}]$
                                     ELSE $\land pc' = [pc$ EXCEPT $![self] = \text{“CheckOkTimeout”}]$
                                          $\land$ UNCHANGED $\langle channels, leader \rangle$
                                $\land$ UNCHANGED $\langle failed\_leader, initiator, n,$
                                                $others\_who\_failed \rangle$

$CheckOkTimeout(self) \triangleq \land pc[self] = \text{“CheckOkTimeout”}$
```

```
                                   $\land$ IF $DoesNotReceiveOKResponseFromNewLeaders(self)$
                                          THEN $\land\ pc' = [pc$ EXCEPT $![self] =$ "NormalExecution"$]$
                                          ELSE $\land\ pc' = [pc$ EXCEPT $![self] =$ "AcceptNewLeader"$]$
                                   $\land$ UNCHANGED $\langle failed\_leader, initiator, n,$
                                                     $others\_who\_failed, channels, leader\rangle$

$AcceptNewLeader(self) \triangleq$ $\land\ pc[self] =$ "AcceptNewLeader"
                                   $\land\ \exists\ new\_leader \in NewLeaders(self):$
                                        $\land\ \land\ channels' = [channels$ EXCEPT $![new\_leader][self] =$ ""$]$
                                           $\land\ leader' = [leader$ EXCEPT $![self] = new\_leader]$
                                        $\land\ pc' = [pc$ EXCEPT $![self] =$ "CheckOkTimeout"$]$
                                   $\land$ UNCHANGED $\langle failed\_leader, initiator, n,$
                                                     $others\_who\_failed\rangle$

$NormalExecution(self) \triangleq$ $\land\ pc[self] =$ "NormalExecution"
                                   $\land\ \exists\ sender \in MessageSenders(self):$
                                        IF $channels[sender][self] =$ "Election"
                                           THEN $\land\ channels' = [channels$ EXCEPT $![self][sender] =$ "OK",
                                                                                  $![sender][self] =$ ""$]$
                                                $\land\ pc' = [pc$ EXCEPT $![self] =$ "BecomeLeaderOrStartElection"$]$
                                                $\land$ UNCHANGED $leader$
                                           ELSE $\land$ IF $channels[sender][self] =$ "Leader"
                                                        THEN $\land\ \land\ channels' = [channels$ EXCEPT $![sender][self]$
                                                                $\land\ leader' = [leader$ EXCEPT $![self] = sender]$
                                                             $\land\ pc' = [pc$ EXCEPT $![self] =$ "NormalExecution"$]$
                                                        ELSE $\land\ channels' = [channels$ EXCEPT $![sender][self] =$
                                                             $\land\ pc' = [pc$ EXCEPT $![self] =$ "Failed"$]$
                                                             $\land$ UNCHANGED $leader$
                                   $\land$ UNCHANGED $\langle failed\_leader, initiator, n,$
                                                     $others\_who\_failed\rangle$

$Failed(self) \triangleq$ $\land\ pc[self] =$ "Failed"
                          $\land$ TRUE
                          $\land\ pc' = [pc$ EXCEPT $![self] =$ "Done"$]$
                          $\land$ UNCHANGED $\langle failed\_leader, initiator, n, others\_who\_failed,$
                                            $channels, leader\rangle$

$Peer(self) \triangleq$ $Initialize(self) \lor BecomeLeaderOrStartElection(self)$
                          $\lor\ CheckElectionTimeout(self) \lor CheckOkTimeout(self)$
                          $\lor\ AcceptNewLeader(self) \lor NormalExecution(self)$
                          $\lor\ Failed(self)$

 Allow infinite stuttering to prevent deadlock on termination.
$Terminating \triangleq$ $\land\ \forall\ self \in ProcSet : pc[self] =$ "Done"
                          $\land$ UNCHANGED $vars$

$Next \triangleq (\exists\ self \in IDS : Peer(self))$
```

$\lor$ *Terminating*

$Spec \triangleq \land Init \land \Box[Next]_{vars}$
$\qquad\qquad \land \forall self \in IDS : \mathrm{WF}_{vars}(Peer(self))$

$Termination \triangleq \Diamond(\forall self \in ProcSet : pc[self] = \text{"Done"})$

END TRANSLATION