MODULE *Pactus*

The specification of the *Pactus* consensus algorithm based on Practical *Byzantine* Fault Tolerant.
For more information check here: https://pactus.org/learn/consensus/protocol/

EXTENDS *Integers*, *Sequences*, *FiniteSets*, *TLC*

CONSTANT

The total number of faulty nodes

$NumFaulty$,

The maximum number of round per height.
this is to restrict the allowed behaviours that *TLC* scans through.

$MaxRound$

ASSUME

$\land\ NumFaulty \geq 1$

VARIABLES

log,

$states$

Total number of replicas that is $3f + 1$ where f is number of faulty nodes.

$Replicas \triangleq (3 * NumFaulty) + 1$

2/3 of total replicas that is $2f + 1$

$QuorumCnt \triangleq (2 * NumFaulty) + 1$

1/3 of total replicas that is $f + 1$

$OneThird \triangleq NumFaulty + 1$

A tuple with all variables in the spec (for ease of use in temporal conditions)

$vars \triangleq \langle states, log \rangle$

Helper functions

Fetch a subset of messages in the network based on the *params* filter.

$SubsetOfMsgs(params) \triangleq$

$\quad \{msg \in log : \forall field \in \text{DOMAIN } params : msg[field] = params[field]\}$

IsProposer checks if the replica is the proposer for this round

$IsProposer(index) \triangleq$

$\quad (states[index].round + states[index].proposerIndex) \% Replicas = index$

HasPrepareQuorum checks if there is a quorum of the *PREPARE* votes in each round.

$HasPrepareQuorum(index) \triangleq$

$\quad Cardinality(SubsetOfMsgs([$

$\qquad type \mapsto \text{“PREPARE”},$

$\qquad height \mapsto states[index].height,$

$\qquad round \mapsto states[index].round])) \geq QuorumCnt$

HasPrecommitQuorum checks if there is a quorum of the *PRECOMMIT* votes in each round.

```
$HasPrecommitQuorum(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ “PRECOMMIT”,
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round])) \geq QuorumCnt$
```

HasChangeProposerQuorum checks if there is a quorum of the CHANGE-PROPOSER votes in each round.

```
$HasChangeProposerQuorum(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ “CHANGE-PROPOSER”,
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round])) \geq QuorumCnt$

$HasOneThirdOfChangeProposer(index) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto$ “CHANGE-PROPOSER”,
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round])) \geq OneThird$

$GetProposal(height, round) \triangleq$
    $SubsetOfMsgs([type \mapsto$ “PROPOSAL”, $height \mapsto height, round \mapsto round])$

$HasProposal(height, round) \triangleq$
    $Cardinality(GetProposal(height, round)) > 0$

$IsCommitted(height) \triangleq$
    $Cardinality(SubsetOfMsgs([type \mapsto$ “BLOCK-ANNOUNCE”, $height \mapsto height])) > 0$

$HasVoted(index, type) \triangleq$
    $Cardinality(SubsetOfMsgs([$
        $type \mapsto type,$
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $index \mapsto index])) > 0$
```

Network functions

SendMsg broadcasts the message iff the current height is not committed yet.

```
$SendMsg(msg) \triangleq$
    IF $\neg IsCommitted(msg.height)$
     THEN $log' = log \cup \{msg\}$
     ELSE $log' = log$
```

SendProposal is used to broadcast the *PROPOSAL* into the network.

```
$SendProposal(index) \triangleq$
    $SendMsg([$
        $type \mapsto$ "PROPOSAL",
        $height \mapsto states[index].height$,
        $round \mapsto states[index].round$,
        $index \mapsto index$])
```

SendPrepareVote is used to broadcast *PREPARE* votes into the network.

```
$SendPrepareVote(index) \triangleq$
    $SendMsg$([
        $type \mapsto$ "PREPARE",
        $height \mapsto states[index].height$,
        $round \mapsto states[index].round$,
        $index \mapsto index$])
```

SendPrecommitVote is used to broadcast *PRECOMMIT* votes into the network.

```
$SendPrecommitVote(index) \triangleq$
    $SendMsg$([
        $type \mapsto$ "PRECOMMIT",
        $height \mapsto states[index].height$,
        $round \mapsto states[index].round$,
        $index \mapsto index$])
```

SendChangeProposerRequest is used to broadcast CHANGE-PROPOSER votes into the network.

```
$SendChangeProposerRequest(index) \triangleq$
    $SendMsg$([
        $type \mapsto$ "CHANGE-PROPOSER",
        $height \mapsto states[index].height$,
        $round \mapsto states[index].round$,
        $index \mapsto index$])
```

AnnounceBlock announces the block for the current height and clears the logs.

```
$AnnounceBlock(index) \triangleq$
    $log' = \{msg \in log : (msg.type =$ "BLOCK-ANNOUNCE"$) \lor msg.height > states[index].height\} \cup \{[$
        $type \mapsto$ "BLOCK-ANNOUNCE",
        $height \mapsto states[index].height$,
        $round \mapsto states[index].round$,
        $index \mapsto states[index].proposerIndex]\}$
```

States functions

NewHeight state

```
$NewHeight(index) \triangleq$
     $\land states[index].name =$ "new-height"
     $\land states' = [states$ EXCEPT
        $![index].name =$ "propose",
        $![index].height = states[index].height + 1$,
        $![index].round = 0]$
```

```
    $\land$ UNCHANGED $\langle log \rangle$

Propose state
$Propose(index) \triangleq$
    $\land$  $states[index].name =$ "propose"
    $\land$  IF $IsProposer(index)$
          THEN $SendProposal(index)$
          ELSE $log' = log$
    $\land$  $states' = [states$ EXCEPT $![index].name =$ "prepare"$]$

Prepare state
$Prepare(index) \triangleq$
    $\land$  $states[index].name =$ "prepare"
    $\land$  IF $\land HasProposal(states[index].height, states[index].round)$
             $\land \neg HasOneThirdOfChangeProposer(index)$
             $\lor states[index].round \geq MaxRound$
          THEN $\land SendPrepareVote(index)$
               $\land$ IF $HasPrepareQuorum(index)$
                    THEN $states' = [states$ EXCEPT $![index].name =$ "precommit"$]$
                    ELSE $states' = states$
          ELSE $\land SendChangeProposerRequest(index)$
               $\land states' = [states$ EXCEPT $![index].name =$ "change-proposer"$]$

Precommit state
$Precommit(index) \triangleq$
    $\land states[index].name =$ "precommit"
    $\land SendPrecommitVote(index)$
    $\land$ IF  $\land HasPrecommitQuorum(index)$
            $\land \neg HasOneThirdOfChangeProposer(index)$
            $\land HasVoted(index,$ "PRECOMMIT"$)$
        THEN $states' = [states$ EXCEPT $![index].name =$ "commit"$]$
        ELSE $states' = states$

Commit state
$Commit(index) \triangleq$
    $\land states[index].name =$ "commit"
    $\land AnnounceBlock(index)$
    $\land states' = [states$ EXCEPT
        $![index].name =$ "new-height",
        $![index].proposerIndex = (states[index].round + 1) \% Replicas]$

ChangeProposer state
$ChangeProposer(index) \triangleq$
    $\land states[index].name =$ "change-proposer"
    ∧ IF  $HasChangeProposerQuorum(index)$
        THEN  $states' = [states$ EXCEPT
                $![index].name =$ "propose",
                $![index].round = states[index].round + 1]$
        ELSE  $states' = states$
    ∧ UNCHANGED  $\langle log \rangle$
```

Sync checks the *log* for the committed blocks at the current height.
If such a block exists, it commits and moves to the next height.

```
$Sync(index) \triangleq$
    LET
        $blocks \triangleq SubsetOfMsgs([type \mapsto$ "BLOCK-ANNOUNCE", $height \mapsto states[index].height])$
    IN
        $\land Cardinality(blocks) > 0$
        $\land states' = [states$ EXCEPT
            $![index].name =$ "propose",
            $![index].height = states[index].height + 1,$
            $![index].round = 0,$
            $![index].proposerIndex = (($CHOOSE $b \in blocks :$ TRUE$).round + 1)\%Replicas]$
        $\land log' = log$
```

```
$Init \triangleq$
    $\land log = \{\}$
    $\land states = [index \in 0 \mathrel{..} Replicas - 1 \mapsto [$
        $name$             $\mapsto$ "new-height",
        $height$           $\mapsto 0,$
        $round$            $\mapsto 0,$
        $proposerIndex$    $\mapsto 0]]$

$Next \triangleq$
    $\exists\, index \in 0 \mathrel{..} Replicas - 1 :$
        $\lor Sync(index)$
        $\lor NewHeight(index)$
        $\lor Propose(index)$
        $\lor Prepare(index)$
        $\lor Precommit(index)$
        $\lor Commit(index)$
        $\lor ChangeProposer(index)$

$Spec \triangleq$
    $Init \land \Box[Next]_{vars}$
```

TypeOK is the type-correctness invariant.

```
$TypeOK \triangleq$
$\land$   $\forall\, index \in 0 \,..\, Replicas - 1 :$
      $\land\, states[index].name \in \{$"new-height", "propose", "prepare",
         "precommit", "commit", "change-proposer"$\}$
      $\land\, \neg IsCommitted(states[index].height) \Rightarrow$
         $\land\, states[index].name =$ "new-height" $\land\, states[index].height > 1 \Rightarrow$
            $IsCommitted(states[index].height - 1)$
         $\land\, states[index].name =$ "propose" $\Rightarrow$
            $Cardinality(SubsetOfMsgs([index \mapsto index,\ height \mapsto states[index].height,\ round \mapsto states[ind$
         $\land\, states[index].name =$ "precommit" $\Rightarrow$
            $HasPrepareQuorum(index)$
         $\land\, states[index].name =$ "commit" $\Rightarrow$
            $HasPrecommitQuorum(index)$
         $\land\, \forall\, round \in 0 \,..\, states[index].round :$
            $\land\, Cardinality(\,GetProposal(states[index].height,\ round)) \leq 1$ not more than two proposals per rou
            $\land\, round > 0 \Rightarrow Cardinality(SubsetOfMsgs([type \mapsto$ "CHANGE-PROPOSER"$,\ round \mapsto round$
      $\land\, IsCommitted(states[index].height) \Rightarrow$
         Check all blocks are same
         $\land$
            $\exists\, x \in SubsetOfMsgs([type \mapsto$ "BLOCK-ANNOUNCE"$,\ height \mapsto states[index].height]) :$
               $\forall\, y \in SubsetOfMsgs([type \mapsto$ "BLOCK-ANNOUNCE"$,\ height \mapsto states[index].height]) :$
                  $x = y$
```