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

$\quad \land \mathit{NumFaulty} \geq 1$

VARIABLES

  log,
  states

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

$\mathit{Replicas} \triangleq (3 * \mathit{NumFaulty}) + 1$

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

$\mathit{QuorumCnt} \triangleq (2 * \mathit{NumFaulty}) + 1$

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

$\mathit{OneThird} \triangleq \mathit{NumFaulty} + 1$

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

$\mathit{vars} \triangleq \langle \mathit{states}, \mathit{log} \rangle$

Helper functions

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

$\mathit{SubsetOfMsgs}(\mathit{params}) \triangleq$

$\quad \{\mathit{msg} \in \mathit{log} : \forall \mathit{field} \in \textsc{domain}\ \mathit{params} : \mathit{msg}[\mathit{field}] = \mathit{params}[\mathit{field}]\}$

IsProposer checks if the replica is the proposer for this round
To simplify, we assume the proposer always starts with the first replica
and moves to the next by the change-proposer phase.

$\mathit{IsProposer}(\mathit{index}) \triangleq$

$\quad \mathit{states}[\mathit{index}].\mathit{round} \% \mathit{Replicas} = \mathit{index}$

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

$\mathit{HasPrepareQuorum}(\mathit{index}) \triangleq$

$\quad \mathit{Cardinality}(\mathit{SubsetOfMsgs}([$

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

$\qquad \mathit{height} \mapsto \mathit{states}[\mathit{index}].\mathit{height},$

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

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",
        $height \mapsto states[index].height,$
        $round \mapsto states[index].round,$
        $index \mapsto -1]\}$

$IsFaulty(index) \triangleq index \geq 3 * NumFaulty$
```

States functions

NewHeight state

```
$NewHeight(index) \triangleq$
    $\land states[index].name =$ "new-height"
    $\land \neg IsFaulty(index)$
    $\land states' = [states$ 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"
    $\land$ $\neg IsFaulty(index)$
    $\land$ 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"
    $\land$ $\neg IsFaulty(index)$
    $\land$ $\neg HasOneThirdOfChangeProposer(index)$   This check is optional
    $\land$ $HasProposal(states[index].height, states[index].round)$
    $\land$ $SendPrepareVote(index)$
    $\land$ IF $\land HasPrepareQuorum(index)$
          THEN $states' = [states$ EXCEPT $![index].name = $ "precommit"$]$
          ELSE $states' = states$
```

Precommit state

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

```
$Timeout(index) \triangleq$
    $\land$
        $\lor states[index].name = $ "prepare"
        $\lor states[index].name = $ "precommit"
    $\land \neg IsFaulty(index)$
    $\land (states[index].round < MaxRound)$
    $\land\ SendChangeProposerRequest(index)$
    $\land\ states' = [states$ EXCEPT
       $![index].name =$ "change-proposer"$]$

$Byzantine(index) \triangleq$
    $\land\ IsFaulty(index)$
    $\land\ SendChangeProposerRequest(index)$
    $\land\ states' = [states$ EXCEPT
       $![index].name =$ "change-proposer"$]$

 Commit state
$Commit(index) \triangleq$
    $\land\ states[index].name =$ "commit"
    $\land\ \neg IsFaulty(index)$
    $\land\ AnnounceBlock(index)$  this clear the logs
    $\land\ states' = [states$ EXCEPT
       $![index].name =$ "new-height"$]$

 ChangeProposer state
$ChangeProposer(index) \triangleq$
    $\land\ states[index].name =$ "change-proposer"
    $\land\ \neg IsFaulty(index)$
    $\land$ IF $HasChangeProposerQuorum(index)$
       THEN $states' = [states$ EXCEPT
              $![index].name =$ "propose",
              $![index].round = states[index].round + 1]$
       ELSE $states' = states$
    $\land$ 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",
            $![index].height = states[index].height + 1,$
            $![index].round = 0]$
         $\land\ log' = log$
```

$Init \triangleq$

```
    $\land\ log = \{\}$
    $\land\ states = [index \in 0 \mathrel{..} Replicas - 1 \mapsto [$
        $name \quad\quad\quad \mapsto \text{“new-height”},$
        $height \quad\quad\ \mapsto 0,$
        $round \quad\quad\ \ \mapsto 0]]$

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

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

TypeOK is the type-correctness invariant.

```
$TypeOK \triangleq$
    $\land \quad \forall\, index \in 0 \mathrel{..} Replicas - 1 :$
          $\land\ states[index].name \in \{\text{“new-height”}, \text{“propose”}, \text{“prepare”},$
              $\text{“precommit”}, \text{“commit”}, \text{“change-proposer”}\}$
          $\land\ \neg IsCommitted(states[index].height) \Rightarrow$
              $\land\ states[index].name = \text{“new-height”} \land states[index].height > 1 \Rightarrow$
                  $IsCommitted(states[index].height - 1)$
              $\land\ states[index].name = \text{“propose”} \land states[index].round > 0 \Rightarrow$
                  $\land\ Cardinality(SubsetOfMsgs([$
                          $type \quad \mapsto \text{“CHANGE-PROPOSER”},$
                          $height \mapsto states[index].height,$
                          $round \mapsto states[index].round - 1])) \geq QuorumCnt$
              $\land\ states[index].name = \text{“precommit”} \Rightarrow$
                  $\land\ HasPrepareQuorum(index)$
                  $\land\ HasProposal(states[index].height, states[index].round)$
              $\land\ states[index].name = \text{“commit”} \Rightarrow$
                  $\land\ HasPrepareQuorum(index)$
                  $\land\ HasPrecommitQuorum(index)$
                  $\land\ HasProposal(states[index].height, states[index].round)$
              $\land\ \forall\, round \in 0 \mathrel{..} states[index].round :$
                  Not more than one proposal per round
                  $\land\ Cardinality(GetProposal(states[index].height, round)) \leq 1$
```