MODULE *Zarb*

The specification of the *Zarb* consensus algorithm *Zarb* consensus algorithm based on Practical *Byzantine* Fault Tolerant . For more information check here: https://zarb.network/learn/consensus/consensus-mechanism.html

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.

$HasPrepareQuorum(index) \triangleq$
$\quad Cardinality(SubsetOfMsgs([$
$\qquad type \mapsto$ "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.

$$
\begin{array}{l}
HasPrecommitQuorum(index) \triangleq \\
\quad Cardinality(SubsetOfMsgs([\\
\qquad type \mapsto \text{“PRECOMMIT”}, \\
\qquad height \mapsto states[index].height, \\
\qquad round \mapsto states[index].round])) \geq QuorumCnt
\end{array}
$$

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

$$
\begin{array}{l}
HasChangeProposerQuorum(index) \triangleq \\
\quad Cardinality(SubsetOfMsgs([\\
\qquad type \mapsto \text{“CHANGE-PROPOSER”}, \\
\qquad height \mapsto states[index].height, \\
\qquad round \mapsto states[index].round])) \geq QuorumCnt
\end{array}
$$

$$
\begin{array}{l}
HasOneThirdOfChangeProposer(index) \triangleq \\
\quad Cardinality(SubsetOfMsgs([\\
\qquad type \mapsto \text{“CHANGE-PROPOSER”}, \\
\qquad height \mapsto states[index].height, \\
\qquad round \mapsto states[index].round])) \geq OneThird
\end{array}
$$

$$
\begin{array}{l}
GetProposal(height,\ round) \triangleq \\
\quad SubsetOfMsgs([type \mapsto \text{“PROPOSAL”},\ height \mapsto height,\ round \mapsto round])
\end{array}
$$

$$
\begin{array}{l}
HasProposal(height,\ round) \triangleq \\
\quad Cardinality(GetProposal(height,\ round)) > 0
\end{array}
$$

$$
\begin{array}{l}
IsCommitted(height) \triangleq \\
\quad Cardinality(SubsetOfMsgs([type \mapsto \text{“BLOCK-ANNOUNCE”},\ height \mapsto height])) > 0
\end{array}
$$

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

$$
\begin{array}{l}
SendProposal(index) \triangleq \\
\quad log' = log \cup \{[\\
\qquad type \mapsto \text{“PROPOSAL”}, \\
\qquad height \mapsto states[index].height, \\
\qquad round \mapsto states[index].round, \\
\qquad index \mapsto index \\
\qquad]\}
\end{array}
$$

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

$$
\begin{array}{l}
SendPrepareVote(index) \triangleq \\
\quad log' = log \cup \{[\\
\qquad type \mapsto \text{“PREPARE”}, \\
\qquad height \mapsto states[index].height, \\
\qquad round \mapsto states[index].round, \\
\qquad index \mapsto index
\end{array}
$$

]}

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

$SendPrecommitVote(index) \triangleq$
$log' = log \cup \{[$
$type \mapsto$ "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.

$SendChangeProposerRequest(index) \triangleq$
$log' = log \cup \{[$
$type \mapsto$ "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") $\vee\ msg.height > states[index].height\} \cup \{[$
$type \mapsto$ "BLOCK-ANNOUNCE",
$height \mapsto states[index].height,$
$round \mapsto states[index].round,$
$index \mapsto -1$
$]\}$

States functions

NewHeight state

$NewHeight(index) \triangleq$
$\wedge\ states[index].name =$ "new-height"
$\wedge\ states' = [states$ EXCEPT
$![index].name =$ "propose",
$![index].height = states[index].height + 1,$
$![index].round = 0]$
$\wedge$ UNCHANGED $\langle log \rangle$

Propose state

$Propose(index) \triangleq$
$\wedge\ \ states[index].name =$ "propose"

```
    ∧  IF IsProposer(index)
       THEN SendProposal(index)
       ELSE log' = log
    ∧  states' = [states EXCEPT ![index].name = "prepare"]

 Prepare state
Prepare(index) ≜
    ∧  states[index].name = "prepare"
    ∧  IF ∧ HasProposal(states[index].height, states[index].round)
          ∧ ¬HasOneThirdOfChangeProposer(index)
          ∨ states[index].round ≥ MaxRound
       THEN ∧ SendPrepareVote(index)
            ∧ IF HasPrepareQuorum(index)
              THEN states' = [states EXCEPT ![index].name = "precommit"]
              ELSE states' = states
       ELSE ∧ SendChangeProposerRequest(index)
            ∧ states' = [states EXCEPT ![index].name = "change-proposer"]

 Precommit state
Precommit(index) ≜
    ∧ states[index].name = "precommit"
    ∧ SendPrecommitVote(index)
    ∧ IF HasPrecommitQuorum(index) ∧ ¬HasOneThirdOfChangeProposer(index)
      THEN states' = [states EXCEPT ![index].name = "commit"]
      ELSE states' = states

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

 ChangeProposer state
ChangeProposer(index) ≜
    ∧ 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 ⟨log⟩
```

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

```
Init ≜
    ∧ log = {}
    ∧ states = [index ∈ 0 .. Replicas − 1 ↦ [
        name          ↦ "new-height",
        height        ↦ 0,
        round         ↦ 0,
        proposerIndex ↦ 0
      ]]

Next ≜
    ∃ index ∈ 0 .. Replicas − 1 :
        ∨ Sync(index)
        ∨ NewHeight(index)
        ∨ Propose(index)
        ∨ Prepare(index)
        ∨ Precommit(index)
        ∨ Commit(index)
        ∨ ChangeProposer(index)

Spec ≜
    Init ∧ □[Next]_vars
```

TypeOK is the type-correctness invariant.

```
TypeOK ≜
    ∧   ∀ index ∈ 0 .. Replicas − 1 :
          ∧ states[index].name ∈ {"new-height", "propose", "prepare",
              "precommit", "commit", "change-proposer"}
          ∧ ¬IsCommitted(states[index].height) ⇒
              ∧ states[index].name = "propose" ⇒
                  ∨ Cardinality(SubsetOfMsgs([index ↦ index, height ↦ states[index].height, round ↦ states[
              ∧ states[index].name = "precommit" ⇒
                  ∨ HasPrepareQuorum(index)
```

$\land\ states[index].name = \text{“commit”} \Rightarrow$
$\quad \lor\ HasPrecommitQuorum(index)$
$\land\ \forall\, round \in 0\,..\,states[index].round :$
$\quad \land\ Cardinality(GetProposal(states[index].height,\ round)) \leq 1$ not more than two proposals per rou
$\quad \land\ round > 0 \Rightarrow Cardinality(SubsetOfMsgs([type \mapsto \text{“CHANGE-PROPOSER”},\ round \mapsto round$