MODULE *Zarb*

The specification of the *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.

$HasPrepareQuorum(index) \triangleq$
$\quad Cardinality(SubsetOfMsgs([$
$\qquad type \mapsto \text{“PREPARE”},$
$\qquad height \mapsto states[index].height,$
$\qquad round \mapsto states[index].round])) \geq QuorumCnt$

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

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

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

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

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

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

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

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

Network functions

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

$$
\begin{array}{l}
SendMsg(msg) \triangleq \\
\quad \text{IF } \neg IsCommitted(msg.height) \\
\quad \text{THEN } log' = log \cup \{msg\} \\
\quad \text{ELSE } log' = log
\end{array}
$$

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

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

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

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
$Propose(index) \triangleq$
$\land \quad states[index].name =$ "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 checks the *log* for the committed blocks at the current height.
If such a block exists, it commits and moves to the next height.

$$
\begin{array}{l}
Sync(index) \triangleq \\
\quad \text{LET} \\
\quad\quad blocks \triangleq SubsetOfMsgs([type \mapsto \text{“BLOCK-ANNOUNCE”}, height \mapsto states[index].height]) \\
\quad \text{IN} \\
\quad\quad \land Cardinality(blocks) > 0 \\
\quad\quad \land states' = [states \text{ EXCEPT} \\
\quad\quad\quad ![index].name = \text{“propose”}, \\
\quad\quad\quad ![index].height = states[index].height + 1, \\
\quad\quad\quad ![index].round = 0, \\
\quad\quad\quad ![index].proposerIndex = ((\text{CHOOSE } b \in blocks : \text{TRUE}).round + 1)\%Replicas] \\
\quad\quad \land log' = log
\end{array}
$$

$$
\begin{array}{l}
Init \triangleq \\
\quad \land log = \{\} \\
\quad \land states = [index \in 0 \,..\, Replicas - 1 \mapsto [\\
\quad\quad name \quad\quad\quad\quad \mapsto \text{“new-height”}, \\
\quad\quad height \quad\quad\quad \mapsto 0, \\
\quad\quad round \quad\quad\quad\quad \mapsto 0, \\
\quad\quad proposerIndex \quad \mapsto 0]]
\end{array}
$$

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

$$
\begin{array}{l}
Spec \triangleq \\
\quad Init \land \Box[Next]_{vars}
\end{array}
$$

TypeOK is the type-correctness invariant.

$$
\begin{array}{l}
TypeOK \triangleq \\
\quad \land \quad \forall\, index \in 0 \,..\, Replicas - 1 : \\
\quad\quad\quad \land states[index].name \in \{\text{“new-height”}, \text{“propose”}, \text{“prepare”}, \\
\quad\quad\quad\quad \text{“precommit”}, \text{“commit”}, \text{“change-proposer”}\} \\
\quad\quad\quad \land \neg IsCommitted(states[index].height) \Rightarrow \\
\quad\quad\quad\quad \land states[index].name = \text{“new-height”} \land states[index].height > 1 \Rightarrow \\
\quad\quad\quad\quad\quad IsCommitted(states[index].height - 1) \\
\quad\quad\quad\quad \land states[index].name = \text{“propose”} \Rightarrow
\end{array}
$$

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