MODULE *PODCommit*

This specification is the very basic version of POD (Proof of Devotion) from *Nebulas*. In this specification, we have the following assumptions to simplity the basic idea.

- No dumber node.
- No dynasty change.
- No node change or abdication.
- Assume one node only propose one value.
- Assume there is no failure node, and eventually all nodes should be consistent.
- We don't consider the liveness problem.
- We don't consider normal nodes besides validators.

CONSTANT *Validator*, The set of validators
Majority $1+ n * 2/3$ validators

VARIABLES *vrState*, *vrState*[*r*] is the state of validator
vrPrepared, *vrPrepared*[*r*] is the set of validators from which *r* has received "Prepared" messages for *v*'s proposal
vrCommitted, *vrCommitted*[*r*] is the set of validators from which *r* has received "vote" messages for *v*'s proposal
vrFinal, *vrFinal*[*r*] is the final value, which the proposer.
msgs

In the protocol, processes communicate with one another by sending messages. For simplicity, we represent message passing with the variable *msgs* whose value is the set of all messages that have been sent. A message is sent by adding it to the set *msgs*. An action that, in an implementation, would be enabled by the receipt of a certain message is here enabled by the presence of that message in *msgs*. For simplicity, messages are never removed from *msgs*. This allows a single message to be received by multiple receivers. Receipt of the same message twice is therefore allowed; but in this particular protocol, that shouldn't be a problem.

ASSUME
$\land\ Majority \subseteq \text{SUBSET}\ Validator$
$\land\ \forall\ MS1,\ MS2,\ MS3 \in Majority : MS1 \cap MS2 \cap MS3 \neq \{\}$

All we assume about the set *Majority* of majorities is that any three majorities have non-empty intersection, which makes sure *Majority* is at least 2/3 validators.

$Messages \triangleq$

The set of all possible messages. The *ins* field indicates the sender. For "propose" message, the "*ins*" field means she propose a block. Since we do not mind the proposed value, we do not record the proposed value here. The *acc* field indicates the sender of a message.

$[type : \{\text{"propose"}\},\ ins : Validator,\ acc : Validator]$
$\cup$
$[type : \{\text{"prepare"}\},\ ins : Validator,\ acc : Validator]$
$\cup$
$[type : \{\text{"vote"}\},\ ins : Validator,\ acc : Validator]$

$PODTypeOK \triangleq$
$\land\ vrState \in [Validator \to \{\text{"working"},\ \text{"prepared"},\ \text{"committed"},\ \text{"finality"}\}]$
$\land\ vrPrepared \in [Validator \to \{Validator\}]$
$\land\ vrCommitted \in [Validator \to \{Validator\}]$
$\land\ vrFinal \in [Validator \to Validator \cup \{\text{"none"}\}]$

$\land\ msgs \subseteq Messages$

$PODInit \triangleq$ The initial predicate

$\land\ vrState = [v \in Validator \mapsto \text{“working”}]$
$\land\ vrPrepared = [v \in Validator \mapsto \{\}]$
$\land\ vrCommitted = [v \in Validator \mapsto \{\}]$
$\land\ vrFinal = [v \in Validator \mapsto \text{“none”}]$
$\land\ msgs = \{\}$

THE ACTIONS

$Send(m) \triangleq msgs' = msgs \cup \{m\}$

An action expression that describes the sending of message m.

$PreparedSet(set,\ r) \triangleq \{m \in set : m.acc = r\}$
$CommittedSet(set,\ r) \triangleq \{m \in set : m.acc = r\}$

Validator ACTIONS

$ValidatorPropose(r) \triangleq$

Validator try to propose a block

$\land\ vrState[r] = \text{“working”}$
$\land\ vrState' = [vrState \text{ EXCEPT } ![r] = \text{“prepared”}]$
$\land\ vrPrepared' = [vrPrepared \text{ EXCEPT } ![r] = \{[type \mapsto \text{“prepare”},\ ins \mapsto r,\ acc \mapsto r]\}]$
$\land\ Send([type \mapsto \text{“propose”},\ ins \mapsto r,\ acc \mapsto r])$
$\land\ Send([type \mapsto \text{“prepare”},\ ins \mapsto r,\ acc \mapsto r])$
$\land\ \text{UNCHANGED } \langle vrCommitted,\ vrFinal\rangle$

$ValidatorChooseToCommit \triangleq$

Validator try to vote a block

$\land\ \text{LET } ChooseToCommit(r,\ v) \triangleq$
$\quad \land\ \text{LET } Prepared \triangleq \{m.ins : m \in PreparedSet(vrPrepared[r],\ v)\}$
$\quad\quad \text{IN } Prepared \in Majority$
$\quad \land\ vrState[r] = \text{“prepared”}$
$\quad \land\ vrState' = [vrState \text{ EXCEPT } ![r] = \text{“committed”}]$
$\quad \land\ vrCommitted' = [vrCommitted \text{ EXCEPT } ![r] = vrCommitted[r] \cup \{[type \mapsto \text{“vote”},\ ins \mapsto r,\ acc$
$\quad \land\ Send([type \mapsto \text{“vote”},\ ins \mapsto r,\ acc \mapsto v])$
IN
$\quad \forall\, r \in Validator,\ v \in Validator : ChooseToCommit(r,\ v)$
$\land\ \text{UNCHANGED } \langle vrPrepared,\ vrFinal\rangle$

$ValidatorChooseToFinal \triangleq$

Validator try to final a block.

$\land\ \text{LET } ChooseToFinal(r,\ v) \triangleq$
$\quad \land\ \text{LET } Committed \triangleq \{m.ins : m \in CommittedSet(vrCommitted[r],\ v)\}$

IN $Committed \in Majority$
$\land\ vrState[r] = \text{“committed”}$
$\land\ vrState' = [vrState \text{ EXCEPT } ![r] = \text{“finality”}]$
$\land\ vrFinal' = [vrFinal \text{ EXCEPT } ![r] = v]$

IN
$\forall\, r \in Validator,\ v \in Validator : ChooseToFinal(r,\ v)$
$\land$ UNCHANGED $\langle vrPrepared,\ vrCommitted \rangle$

RECV messages

$RecvPropose(r,\ v) \triangleq$

The action when recv a prepare message.

$\land\ vrState[r] = \text{“working”}$
$\land \exists\, m \in msgs :$
$\quad\land\ m.type = \text{“propose”}$
$\quad\land\ m.ins = v$
$\land\ vrState' = [vrState \text{ EXCEPT } ![r] = \text{“prepared”}]$
$\land\ Send([type \mapsto \text{“prepare”},\ ins \mapsto r,\ acc \mapsto v])$
$\land\ vrPrepared' = [vrPrepared \text{ EXCEPT } ![r] = \{[type \mapsto \text{“prepare”},\ ins \mapsto r,\ acc \mapsto v]\}]$
$\land$ UNCHANGED $\langle vrCommitted,\ vrFinal \rangle$

$RecvPrepare(r,\ from,\ v) \triangleq$

The action when recv a prepare message.

$\land\ vrState[r] = \text{“prepared”}$
$\land \exists\, m \in msgs :$
$\quad\land\ m.type = \text{“prepare”}$
$\quad\land\ m.acc = v$
$\quad\land\ m.ins = from$
$\land\ vrPrepared' = [vrPrepared \text{ EXCEPT } ![r] = vrPrepared[r] \cup \{[type \mapsto \text{“prepare”},\ ins \mapsto r,\ acc \mapsto v]\}]$
$\land$ UNCHANGED $\langle vrCommitted,\ vrState,\ vrFinal \rangle$

$RecvVote(r,\ from,\ v) \triangleq$

The action when recv a vote message.

$\land\ vrState[r] = \text{“prepared”}$
$\land \exists\, m \in msgs :$
$\quad\land\ m.type = \text{“vote”}$
$\quad\land\ m.acc = v$
$\quad\land\ m.ins = from$
$\land\ vrCommitted' = [vrCommitted \text{ EXCEPT } ![r] = vrCommitted[r] \cup \{[type \mapsto \text{“prepare”},\ ins \mapsto r,\ acc \mapsto v]$
$\land$ UNCHANGED $\langle vrPrepared,\ vrState \rangle$

$PODNext \triangleq$
$\lor \exists\, r \in Validator : ValidatorPropose(r)$
$\lor\ ValidatorChooseToCommit$

$\lor$ *ValidatorChooseToFinal*
$\lor \exists\, r,\, v \in \mathit{Validator} : \mathit{RecvPropose}(r,\, v)$
$\lor \exists\, r,\, \mathit{from},\, v \in \mathit{Validator} :\ \lor \mathit{RecvPrepare}(r,\, \mathit{from},\, v)$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \lor \mathit{RecvVote}(r,\, \mathit{from},\, v)$

PODConsistent $\triangleq$

A state predicate asserting that two *Validators* have not arrived at conflicting decisions. It is an invariant of the specification. Actually, *PoD* don't need this, so no consistency requirement.

$\land \forall\, r1,\, r2 \in \mathit{Validator} : \neg \land \mathit{vrState}[r1] = \text{“aborted”}$
$\qquad\qquad\qquad\qquad\quad \land \mathit{vrState}[r2] = \text{“finality”}$
$\land$ LET $\mathit{FinalValidators} \triangleq \{r \in \mathit{Validator} :$
$\qquad\qquad\qquad\qquad\qquad\qquad \mathit{vrState}[r] = \text{“finality”}\}$
IN $\quad \forall\, r1,\, r2 \in \mathit{FinalValidators}:$
$\qquad\qquad\qquad\quad \mathit{vrFinal}[r1] = \mathit{vrFinal}[r2]$

$\mathit{PODSpec} \triangleq \mathit{PODInit} \land \Box[\mathit{PODNext}]_{\langle \mathit{vrState},\, \mathit{vrPrepared},\, \mathit{vrCommitted},\, \mathit{vrFinal}\rangle}$

THEOREM $\mathit{PODSpec} \Rightarrow \Box\ (\mathit{PODTypeOK} \land \mathit{PODConsistent})$

\ ∗ Modification History
\ ∗ Last modified Sat *Jan* 06 20:12:31 *CST* 2018 by *xuepeng*
\ ∗ Created *Wed Jan* 03 23:52:11 *CST* 2018 by *xuepeng*