MODULE *AbstractBallotingWithPrepare*

This is a specification of *SCP*'s balloting protocol. We work at a high level of abstraction where we do not explicitely model messges. Instead, we track what statements are voted/accepted prepared and committed by each node. What we do model explicitely is how each node n votes and accepts statements based on its current ballot $ballot[n]$ and its highest confirmed-prepared ballot $h[n]$.

We provide an inductive invariant that implies the agreement property, and we check its inductiveness exhaustively for small instances of the domain model.

An informal specification of *SCP* can be found at: https://datatracker.ietf.org/doc/html/draft-mazieres-dinrg-scp-05#section-3.5

EXTENDS *DomainModel*

VARIABLES
ballot
, *h*
, *voteToPrepare*
, *acceptedPrepared*
, *voteToCommit*
, *acceptedCommitted*
, *externalized*
, *byz* the set of malicious nodes

$$
\begin{aligned}
&TypeOK \triangleq \\
&\quad \land\ ballot \in [N \to BallotOrNull] \\
&\quad \land\ h \in [N \to BallotOrNull] \\
&\quad \land\ voteToPrepare \in [N \to \text{SUBSET } Ballot] \\
&\quad \land\ acceptedPrepared \in [N \to \text{SUBSET } Ballot] \\
&\quad \land\ voteToCommit \in [N \to \text{SUBSET } Ballot] \\
&\quad \land\ acceptedCommitted \in [N \to \text{SUBSET } Ballot] \\
&\quad \land\ externalized \in [N \to \text{SUBSET } Ballot] \\
&\quad \land\ byz \in \text{SUBSET } N
\end{aligned}
$$

$$
\begin{aligned}
&Init \triangleq \\
&\quad \land\ ballot = [n \in N \mapsto nullBallot] \text{ current ballot of each node} \\
&\quad \land\ h = [n \in N \mapsto nullBallot] \text{ current highest confirmed-prepared ballot of each node} \\
&\quad \land\ voteToPrepare = [n \in N \mapsto \{\}] \\
&\quad \land\ acceptedPrepared = [n \in N \mapsto \{\}] \\
&\quad \land\ voteToCommit = [n \in N \mapsto \{\}] \\
&\quad \land\ acceptedCommitted = [n \in N \mapsto \{\}] \\
&\quad \land\ externalized = [n \in N \mapsto \{\}] \\
&\quad \land\ byz \in FailProneSet \text{ } byz \text{ is initially set to an arbitrary fail-prone set}
\end{aligned}
$$

Node n enters a new ballot and votes to prepare it. Note that n votes to prepare its new ballot $ballot'[n]$ regardless of whether it has previously vote to commit an incompatible ballot b. The main subtlety of the protocol is that this is okay because:

1) We must have $b \prec h[n]$ because we, when n votes to commit b (see *VoteToCommit*), it sets $h[n] = b$ if $h[n] \prec b$, and subsequently $h[n]$ can only grow, and

2) therefore, if $h[n].value \neq b.value$, then n confirmed $h[n]$ as prepared (by definition of how $h[n]$ is updated) and thus we know that, even though n voted to commit b, b can never gather a quorum of votes to commit.

Note how this reasoning appears in the inductive invariant below.

$IncreaseBallotCounter(n,\ c) \triangleq$

$\land\ c > 0$
$\land\ c > ballot[n].counter$
$\land\ h[n].counter \leq c$
$\land$ IF $h[n] \neq nullBallot$
THEN $ballot' = [ballot$ EXCEPT $![n] = bal(c,\ h[n].value)]$
ELSE $\exists\, v \in V : ballot' = [ballot$ EXCEPT $![n] = bal(c,\ v)]$
$\land\ voteToPrepare' = [voteToPrepare$ EXCEPT $![n] = @ \cup \{ballot[n]'\}]$
$\land$ UNCHANGED $\langle h,\ acceptedPrepared,\ voteToCommit,\ acceptedCommitted,\ externalized,\ byz\rangle$

Next we describe when a node accepts and confirms ballots prepared. Nothing surprising here.

$AcceptPrepared(n,\ b) \triangleq$

$\land\ \lor \exists\, Q \in Quorum : \forall\, n2 \in Q \setminus byz : b \in voteToPrepare[n2] \cup acceptedPrepared[n2]$
$\lor \exists\, Bl \in BlockingSet : \forall\, n2 \in Bl \setminus byz : b \in acceptedPrepared[n2]$

NOTE: here we could check that nothing less-and-incompatible is accepted committed. That would simplify the agreement

$\land\ acceptedPrepared' = [acceptedPrepared$ EXCEPT $![n] = @ \cup \{b\}]$
$\land$ UNCHANGED $\langle ballot,\ h,\ voteToPrepare,\ voteToCommit,\ acceptedCommitted,\ externalized,\ byz\rangle$

$ConfirmPrepared(n,\ b) \triangleq$

$\land\ b.counter > -1$
$\land\ h[n] \prec b$
$\land\ \exists\, Q \in Quorum : \forall\, n2 \in Q \setminus byz : b \in acceptedPrepared[n2]$
$\land\ h' = [h$ EXCEPT $![n] = b]$
$\land$ UNCHANGED $\langle ballot,\ voteToPrepare,\ acceptedPrepared,\ voteToCommit,\ acceptedCommitted,\ externalize$

When a node votes to commit a ballot, it must check that it has not already voted or accepted to abort it. This is crucial to avoid externalizing two different values in two different ballots. We also update $h[n]$ if needed to reflect the new highest-confirmed prepared ballot.

$VoteToCommit(n,\ b) \triangleq$

$\land\ b.counter > 0$
$\land\ b = ballot[n]$
$\land\ \forall\, b2 \in Ballot : LessThanAndIncompatible(b,\ b2) \Rightarrow$
$b2 \notin voteToPrepare[n] \cup acceptedPrepared[n]$
$\land\ b \prec h[n] \Rightarrow b.value = h[n].value$
$\land\ \exists\, Q \in Quorum : \forall\, n2 \in Q \setminus byz : b \in acceptedPrepared[n2]$
$\land\ voteToCommit' = [voteToCommit$ EXCEPT $![n] = @ \cup \{b\}]$
$\land$ IF $h[n] \preceq b$
THEN $h' = [h$ EXCEPT $![n] = b]$

```
        ELSE  UNCHANGED h
    ∧  UNCHANGED ⟨ballot, voteToPrepare, acceptedPrepared, acceptedCommitted, externalized, byz⟩
```

Next we describe when a node accepts and confirms ballots committed. Nothing surprising here.

```
AcceptCommitted(n, b) ≜
    ∧  b = ballot[n]
    ∧  ∨ ∃ Q ∈ Quorum : ∀ n2 ∈ Q \ byz : b ∈ voteToCommit[n2]
       ∨ ∃ Bl ∈ BlockingSet : ∀ n2 ∈ Bl \ byz : b ∈ acceptedCommitted[n2]
    ∧  acceptedCommitted' = [acceptedCommitted EXCEPT ![n] = @ ∪ {b}]
    ∧  UNCHANGED ⟨ballot, h, voteToPrepare, acceptedPrepared, voteToCommit, externalized, byz⟩

Externalize(n, b) ≜
    ∧  b = ballot[n]
    ∧  ∃ Q ∈ Quorum : ∀ n2 ∈ Q \ byz : b ∈ acceptedCommitted[n2]
    ∧  externalized' = [externalized EXCEPT ![n] = @ ∪ {b}]
    ∧  UNCHANGED ⟨ballot, h, voteToPrepare, acceptedPrepared, voteToCommit, acceptedCommitted, byz⟩

ByzantineHavoc ≜
  ∧ ∃ x ∈ [byz → SUBSET Ballot] :
      voteToPrepare' = [n ∈ N ↦ IF n ∈ byz THEN x[n] ELSE voteToPrepare[n]]
  ∧ ∃ x ∈ [byz → SUBSET Ballot] :
      acceptedPrepared' = [n ∈ N ↦ IF n ∈ byz THEN x[n] ELSE acceptedPrepared[n]]
  ∧ ∃ x ∈ [byz → SUBSET Ballot] :
      voteToCommit' = [n ∈ N ↦ IF n ∈ byz THEN x[n] ELSE voteToCommit[n]]
  ∧ ∃ x ∈ [byz → SUBSET Ballot] :
      acceptedCommitted' = [n ∈ N ↦ IF n ∈ byz THEN x[n] ELSE acceptedCommitted[n]]
  ∧ UNCHANGED ⟨h, externalized, byz⟩
```

Finally we put everything together:

```
Next ≜
    ∨ ∃ n ∈ N \ byz, c ∈ BallotNumber, v ∈ V :
        LET b ≜ bal(c, v) IN
            ∨ IncreaseBallotCounter(n, c)
            ∨ AcceptPrepared(n, b)
            ∨ ConfirmPrepared(n, b)
            ∨ VoteToCommit(n, b)
            ∨ AcceptCommitted(n, b)
            ∨ Externalize(n, b)
    ∨ ByzantineHavoc

vars ≜ ⟨ballot, h, voteToPrepare, acceptedPrepared, voteToCommit, acceptedCommitted, externalized, byz⟩

Spec ≜ Init ∧ □[Next]_vars

Agreement ≜
    ∀ n1, n2 ∈ N \ byz : ∀ b1, b2 ∈ Ballot :
```

$$b1 \in externalized[n1] \land b2 \in externalized[n2] \Rightarrow b1.value = b2.value$$

Here is an inductive invariant that implies agreement:

$InductiveInvariant \triangleq$

First, the boring stuff:

$\land\ TypeOK$

$\land\ byz \in FailProneSet$

$\land\ \forall\, n \in N \setminus byz,\ c1,\ c2 \in BallotNumber,\ v1,\ v2 \in V :$

LET $b1 \triangleq bal(c1,\ v1) b2 \triangleq bal(c2,\ v2)$ IN

$\land\ ballot[n].counter > -1 \Rightarrow ballot[n].counter > 0$

$\land\ b1 \in voteToPrepare[n] \lor b1 \in voteToCommit[n] \Rightarrow b1.counter > 0 \land b1.counter \leq ballot[n].counte$

$\land\ b1 \in acceptedPrepared[n] \Rightarrow \exists\, Q \in Quorum : \forall\, n2 \in Q \setminus byz : b1 \in voteToPrepare[n2]$

$\land\ b1 \in acceptedCommitted[n] \Rightarrow \exists\, Q \in Quorum : \forall\, n2 \in Q \setminus byz : b1 \in voteToCommit[n2]$

$\land\ h[n].counter > 0 \Rightarrow \exists\, Q \in Quorum : \forall\, n2 \in Q \setminus byz : h[n] \in acceptedPrepared[n2]$

$\land\ b1 \in externalized[n] \Rightarrow \exists\, Q \in Quorum : \forall\, n2 \in Q \setminus byz : b1 \in acceptedCommitted[n2]$

$\land\ b1 \in voteToPrepare[n] \lor b1 \in voteToCommit[n] \Rightarrow$

$\land\ b1.counter \leq ballot[n].counter$

$\land\ b1.counter = ballot[n].counter \Rightarrow b1.value = ballot[n].value$

$\land\ bal(c1,\ v1) \in voteToPrepare[n] \land bal(c1,\ v2) \in voteToPrepare[n] \Rightarrow v1 = v2$

$\land\ bal(c1,\ v1) \in voteToCommit[n] \land bal(c1,\ v2) \in voteToCommit[n] \Rightarrow v1 = v2$

$\land\ b1 \in voteToCommit[n] \Rightarrow$

$\land\ \exists\, Q \in Quorum : \forall\, n2 \in Q \setminus byz : b1 \in acceptedPrepared[n2]$

$\land\ b1 \preceq h[n]$ note this is important as it implies that, if we later vote to abort $b1$, we must have confirmed

Next, the crux of the matter:

(in short, a node only overrides "commit v" if it is sure that "commit v" cannot reach quorum threshold)

$\land\ \land\ b1 \in voteToCommit[n]$

$\land\ LessThanAndIncompatible(b1,\ b2)$

$\land\ b2 \in voteToPrepare[n]$

$\Rightarrow \forall\, Q \in Quorum : \exists\, n2 \in Q \setminus byz :$

$b1 \notin voteToCommit[n2] \land ballot[n2].counter > b1.counter$

Finally, our goal:

$\land\ Agreement$

$AcceptNeverContradictory \triangleq \forall\, b1,\ b2 \in Ballot,\ n1,\ n2 \in N \setminus byz :$

$\land\ b1 \in acceptedCommitted[n1]$

$\land\ b2 \in acceptedPrepared[n2]$

$\land\ b1 \prec b2$

$\Rightarrow b1.value = b2.value$