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 also do not model *Byzantine* behavior explicitly. Instead, whenever a node checks that a set (a quorum or a blocking set) voted or accepted a statement, it only checks that the non-Byzantine members of the set did so. This soundly models what could happen under *Byzantine* behavior because *Byzantine* nodes, being unconstrained, could have voted or accepted whatever is needed to make the check pass.

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

TypeOK ≜
      ∧  ballot ∈ [N → BallotOrNull]
      ∧  h ∈ [N → BallotOrNull]
      ∧  voteToPrepare ∈ [N → SUBSET Ballot]
      ∧  acceptedPrepared ∈ [N → SUBSET Ballot]
      ∧  voteToCommit ∈ [N → SUBSET Ballot]
      ∧  acceptedCommitted ∈ [N → SUBSET Ballot]
      ∧  externalized ∈ [N → SUBSET Ballot]
      ∧  byz ∈ SUBSET N

Init ≜
      ∧ ballot = [n ∈ N ↦ nullBallot]  current ballot of each node
      ∧ h =  [n ∈ N ↦ nullBallot]  current highest confirmed-prepared ballot of each node
      ∧ voteToPrepare = [n ∈ N ↦ {}]
      ∧ acceptedPrepared = [n ∈ N ↦ {}]
      ∧ voteToCommit = [n ∈ N ↦ {}]
      ∧ acceptedCommitted  = [n ∈ N ↦ {}]
      ∧ externalized = [n    ∈ N ↦ {}]
      ∧ byz ∈ FailProneSet  byz is initially set to an arbitrary fail-prone set
```

Node n enters a new ballot and votes to prepare it. Note that n votes to prepare its new ballot $\mathit{ballot}'[n]$ regardless of whether it has previously voted 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, 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 $h[n].\mathit{value} \neq b.\mathit{value}$, 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.

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

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

Note that we could check that nothing less-and-incompatible is accepted committed. That would simplify the agreement proof but complicate the liveness proof. In any case, it is an invariant that nothing less-and-incompatible is accepted committed at this point (see *AcceptNeverContradictory*).

```
$\mathit{AcceptPrepared}(n,\ b) \triangleq$
    $\land\ \lor\ \exists\, Q \in \mathit{Quorum} : \forall\, n2 \in Q \setminus \mathit{byz} : b \in \mathit{voteToPrepare}[n2] \cup \mathit{acceptedPrepared}[n2]$
        $\lor\ \exists\, \mathit{Bl} \in \mathit{BlockingSet} : \forall\, n2 \in \mathit{Bl} \setminus \mathit{byz} : b \in \mathit{acceptedPrepared}[n2]$
    $\land\ \mathit{acceptedPrepared}' = [\mathit{acceptedPrepared}\ \text{EXCEPT}\ ![n] = @ \cup \{b\}]$
    $\land$ UNCHANGED $\langle \mathit{ballot},\ h,\ \mathit{voteToPrepare},\ \mathit{voteToCommit},\ \mathit{acceptedCommitted},\ \mathit{externalized},\ \mathit{byz}\rangle$

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

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.

```
$\mathit{VoteToCommit}(n,\ b) \triangleq$
    $\land\ b.\mathit{counter} > 0$
    $\land\ b = \mathit{ballot}[n]$
    $\land\ \forall\, b2 \in \mathit{Ballot} : \mathit{LessThanAndIncompatible}(b,\ b2) \Rightarrow$
            $b2 \notin \mathit{voteToPrepare}[n] \cup \mathit{acceptedPrepared}[n]$
    $\land\ b \prec h[n] \Rightarrow b.\mathit{value} = h[n].\mathit{value}$
    $\land\ \exists\, Q \in \mathit{Quorum} : \forall\, n2 \in Q \setminus \mathit{byz} : b \in \mathit{acceptedPrepared}[n2]$
    $\land\ \mathit{voteToCommit}' = [\mathit{voteToCommit}\ \textsc{except}\ ![n] = @ \cup \{b\}]$
    $\land\ \textsc{if}\ h[n] \preceq b$
         $\textsc{then}\ h' = [h\ \textsc{except}\ ![n] = b]$
         $\textsc{else}\ \ \textsc{unchanged}\ h$
    $\land\ \textsc{unchanged}\ \langle \mathit{ballot}, \mathit{voteToPrepare}, \mathit{acceptedPrepared}, \mathit{acceptedCommitted}, \mathit{externalized}, \mathit{byz} \rangle$
```

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

```
$\mathit{AcceptCommitted}(n, b) \triangleq$
    $\land\ b = \mathit{ballot}[n]$
    $\land\ \lor\ \exists\, Q \in \mathit{Quorum} : \forall\, n2 \in Q \setminus \mathit{byz} : b \in \mathit{voteToCommit}[n2]$
       $\lor\ \exists\, \mathit{Bl} \in \mathit{BlockingSet} : \forall\, n2 \in \mathit{Bl} \setminus \mathit{byz} : b \in \mathit{acceptedCommitted}[n2]$
    $\land\ \mathit{acceptedCommitted}' = [\mathit{acceptedCommitted}\ \textsc{except}\ ![n] = @ \cup \{b\}]$
    $\land\ \textsc{unchanged}\ \langle \mathit{ballot}, h, \mathit{voteToPrepare}, \mathit{acceptedPrepared}, \mathit{voteToCommit}, \mathit{externalized}, \mathit{byz} \rangle$

$\mathit{Externalize}(n, b) \triangleq$
    $\land\ b = \mathit{ballot}[n]$
    $\land\ \exists\, Q \in \mathit{Quorum} : \forall\, n2 \in Q \setminus \mathit{byz} : b \in \mathit{acceptedCommitted}[n2]$
    $\land\ \mathit{externalized}' = [\mathit{externalized}\ \textsc{except}\ ![n] = @ \cup \{b\}]$
    $\land\ \textsc{unchanged}\ \langle \mathit{ballot}, h, \mathit{voteToPrepare}, \mathit{acceptedPrepared}, \mathit{voteToCommit}, \mathit{acceptedCommitted}, \mathit{byz} \rangle$
```

Finally we put everything together:

```
$\mathit{Next} \triangleq$
    $\lor\ \exists\, n \in N \setminus \mathit{byz}, c \in \mathit{BallotNumber}, v \in V :$
       $\textsc{let}\ b \triangleq \mathit{bal}(c, v)\textsc{in}$
            $\lor\ \mathit{IncreaseBallotCounter}(n, c)$
            $\lor\ \mathit{AcceptPrepared}(n, b)$
            $\lor\ \mathit{ConfirmPrepared}(n, b)$
            $\lor\ \mathit{VoteToCommit}(n, b)$
            $\lor\ \mathit{AcceptCommitted}(n, b)$
            $\lor\ \mathit{Externalize}(n, b)$

$\mathit{vars} \triangleq \langle \mathit{ballot}, h, \mathit{voteToPrepare}, \mathit{acceptedPrepared}, \mathit{voteToCommit}, \mathit{acceptedCommitted}, \mathit{externalized}, \mathit{byz} \rangle$

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

$\mathit{Agreement} \triangleq$
   $\forall\, n1, n2 \in N \setminus \mathit{byz} : \forall\, b1, b2 \in \mathit{Ballot} :$
      $b1 \in \mathit{externalized}[n1] \land b2 \in \mathit{externalized}[n2] \Rightarrow b1.\mathit{value} = b2.\mathit{value}$
```

Here is an inductive invariant that implies agreement:

```
$\mathit{InductiveInvariant} \triangleq$
```

First, the boring stuff:

```
    $\land\ \mathit{TypeOK}$
```

```
$\land\ byz \in \mathit{FailProneSet}$
$\land\ \forall n \in N \setminus byz,\ c1,\ c2 \in \mathit{BallotNumber},\ v1,\ v2 \in V :$
   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
    Next, the crux of the matter:
    (in short, a node only overrides “commit $v$” only 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:
$\land\ Agreement$

An additional property implies by the inductive invariant:
$AcceptNeverContradictory \triangleq \forall\, b1,\ b2 \in Ballot,\ n1,\ n2 \in N \setminus byz :$
   $\land\ b1 \in acceptedCommitted[n1]$
   $\land\ b2 \in acceptedPrepared[n2]$
   $\land\ b1 \prec b2$
   $\Rightarrow b1.value = b2.value$
```