─── MODULE *Balloting* ───

Specification of *SCP*'s balloting protocol following the IETF draft at:

https://datatracker.ietf.org/doc/html/draft-mazieres-dinrg-scp-05#section-3.5

This specification abstracts over some aspects of the protocol (*e.g.* increasing the ballot counter), but it does explicitly represent balloting messages. There are also some differences compared to the IETF draft, due to I suspect are omissions in the IETF draft.

Currently this specification covers only the PREPARE and COMMIT phases.

EXTENDS *DomainModel*

$Phase \triangleq \{ \text{"PREPARE"}, \text{"COMMIT"}, \text{"EXTERNALIZE"} \}$

$SCPPrepare \triangleq [$
    $type : \{ \text{"PREPARE"} \}$
, $ballot : Ballot$
, $prepared : BallotOrNull$
, $aCounter : BallotNumber$
, $hCounter : BallotNumber$
, $cCounter : BallotNumber]$

$SCPCommit \triangleq [$
    $type : \{ \text{"COMMIT"} \}$
, $ballot : Ballot$
, $preparedCounter : BallotNumber$
, $hCounter : BallotNumber$
, $cCounter : BallotNumber]$

$SCPExternalize \triangleq [$
    $type : \{ \text{"EXTERNALIZE"} \}$
, $commit : Ballot$
, $hCounter : BallotNumber]$

$Message \triangleq$
    $SCPPrepare \cup SCPCommit \cup SCPExternalize$

Some well-formedness conditions on messages:
$MessageInvariant(m) \triangleq$
    $\wedge\ m.type = \text{"PREPARE"} \Rightarrow$
        $\wedge\ m.ballot.counter > 0$
        $\wedge\ m.prepared.counter > -1 \Rightarrow$
            $\wedge\ m.prepared \preceq m.ballot$
            $\wedge\ m.aCounter \leq m.prepared.counter$
        $\wedge\ m.prepared.counter = -1 \Rightarrow m.aCounter = 0$
        $\wedge\ m.cCounter \leq m.hCounter$
    $\wedge\ m.type = \text{"COMMIT"} \Rightarrow$
        $\wedge\ m.cCounter > 0$

1

$\wedge$ $m.cCounter \le m.ballot.counter$

$\wedge$ $m.cCounter \le m.hCounter$

Meaning of the messages in terms of logical, federated-voting messages.

We will use this to show that this specification refines the *AbstractBalloting* specification.

$LogicalMessages(m) \triangleq$

    CASE $m.type =$ "PREPARE" $\rightarrow$ [

        $voteToAbort \mapsto \{b \in Ballot :$

          $LessThanAndIncompatible(b, m.ballot)\},$

        $acceptedAborted \mapsto \{b \in Ballot :$

          $\vee LessThanAndIncompatible(b, m.prepared)$

          $\vee b.counter < m.aCounter\},$

        $confirmedAborted \mapsto$

          IF $m.hCounter = 0$ THEN $\{\}$

          ELSE $\{b \in Ballot :$

            LET $h \triangleq [counter \mapsto m.hCounter, value \mapsto m.ballot.value]$

            IN $LessThanAndIncompatible(b, h)\},$

        $voteToCommit \mapsto$ IF $m.cCounter = 0$ THEN $\{\}$

          ELSE $\{b \in Ballot :$

            $\wedge m.cCounter \le b.counter \wedge b.counter \le m.hCounter$

            $\wedge b.value = m.ballot.value\},$

        $acceptedCommitted \mapsto \{\}]$

    $\square$ $m.type =$ "COMMIT" $\rightarrow$ [

        $voteToAbort \mapsto \{b \in Ballot : b.value \ne m.ballot.value\},$

        $acceptedAborted \mapsto$

          LET $maxPrepared \triangleq [counter \mapsto m.preparedCounter, value \mapsto m.ballot.value]$

          IN $\{b \in Ballot : LessThanAndIncompatible(b, maxPrepared)\},$

        $confirmedAborted \mapsto$

          LET $maxPrepared \triangleq [counter \mapsto m.hCounter, value \mapsto m.ballot.value]$

          IN $\{b \in Ballot : LessThanAndIncompatible(b, maxPrepared)\},$

        $voteToCommit \mapsto \{b \in Ballot :$

          $m.cCounter \le b.counter \wedge b.value = m.ballot.value\},$

        $acceptedCommitted \mapsto \{b \in Ballot :$

          $\wedge m.cCounter \le b.counter \wedge b.counter \le m.hCounter$

          $\wedge b.value = m.ballot.value\}]$

VARIABLES

    $ballot$ $ballot[n]$ is the current ballot being prepared or committed by node n

, $phase$ $phase[n]$ is the current phase of node n

, $prepared$ $prepared[n]$ is the highest accepted-prepared ballot by node n

, $aCounter$ $aCounter[n]$ is such that all lower ballots are accepted as aborted

    h and c track:

   in the *PREPARE* phase, the highest and lowest confirmed-prepared ballot
   in the *COMMIT* phase, the highest and lowest accepted committed ballot
   in the *EXTERNALIZE* phase, the highest and lowest confirmed committed ballot
   In phase *PREPARE*, *h.value* could be different from *ballot.value*

, h

, c

, $sent$ $sent[n]$ is the set of messages sent by node n

, byz the set of *Byzantine* nodes

$Init \triangleq$
  $\land\ ballot = [n \in N \mapsto NullBallot]$
  $\land\ phase = [n \in N \mapsto \text{"PREPARE"}]$
  $\land\ prepared\ \ = [n \in N \mapsto NullBallot]$
  $\land\ aCounter = [n \in N \mapsto 0]$
  $\land\ h = [n \in N \mapsto NullBallot]$
  $\land\ c = [n \in N \mapsto NullBallot]$
  $\land\ sent = [n \in N \mapsto \{\}]$
  $\land\ byz \in FailProneSet$

$TypeOK \triangleq$
  $\land\ \ \ ballot \in [N \to BallotOrNull]$
  $\land\ \ \ phase \in [N \to Phase]$
  $\land\ \ \ prepared\ \in [N \to BallotOrNull]$
  $\land\ \ \ aCounter \in [N \to BallotNumber]$
  $\land\ \ \ h \in [N \to BallotOrNull]$
  $\land\ \ \ c \in [N \to BallotOrNull]$
  $\land\ \ \ sent \in [N \to \text{SUBSET } Message]$
  $\land\ \ \ byz \in \text{SUBSET } N$

 faulty nodes can send any message they want

$ByzStep \triangleq \exists\, msgs \in [byz \to \text{SUBSET } Message] :$
  $\land\ \ \ sent' = [n\ \in N \mapsto \text{IF } n \notin byz \text{ THEN } sent[n] \text{ ELSE }\ \ msgs[n]]$
  $\land\ \ \ \text{UNCHANGED } \langle ballot,\ phase,\ prepared,\ aCounter,\ h,\ c,\ byz\rangle$

We start by specifying how a node updates its local state depending on the messages it receives.

At any point in time, we may increase the ballot counter and set the ballot value to the value of the highest confirmed prepared ballot, if any, or, if none, arbitrarily.

$IncreaseBallotCounter(n,\ b) \triangleq$
  $\land\ \ b > 0$
  $\land\ \ b > ballot[n].counter$
  $\land\ \ \text{IF } h[n].counter > 0 \text{ THEN}$
    $ballot' = [ballot \text{ EXCEPT } ![n] = [counter \mapsto b,\ value \mapsto h[n].value]]$
   ELSE
    $\exists\, v \in V : ballot' = [ballot \text{ EXCEPT } ![n] = [counter \mapsto b,\ value \mapsto v]]$
   TODO: optimization

3

$\wedge$ IF $b = 1$
$\quad$ THEN $c' = [c$ EXCEPT $![n] = ballot'[n]]$
$\quad$ ELSE UNCHANGED c
$\wedge$ UNCHANGED $\langle phase, prepared, aCounter, h, c, sent, byz \rangle$

$VotesToPrepare(b, m) \triangleq$
$\quad \vee \;\; \wedge \;\; m.type = $ "PREPARE"
$\quad\quad\quad \wedge \;\; \vee \;\; \wedge \;\; b.counter \leq m.ballot.counter$
$\quad\quad\quad\quad\quad\quad\quad \wedge \;\; b.value = m.ballot.value$
$\quad\quad\quad\quad\quad \vee \;\; \wedge \;\; b.counter \leq m.prepared.counter$
$\quad\quad\quad\quad\quad\quad\quad \wedge \;\; b.value = m.prepared.value$
$\quad\quad\quad\quad\quad \vee \;\; b.counter < m.aCounter$
$\quad \vee \;\; \wedge \;\; m.type = $ "COMMIT"
$\quad\quad\quad \wedge \;\; b.value = m.ballot.value$

$AcceptsPrepared(b, m) \triangleq$
$\quad \vee \;\; \wedge \;\; m.type = $ "PREPARE"
$\quad\quad\quad \wedge \;\; \vee \;\; \wedge \;\; b.counter \leq m.prepared.counter$
$\quad\quad\quad\quad\quad\quad\quad \wedge \;\; b.value = m.prepared.value$
$\quad\quad\quad\quad\quad \vee \;\; b.counter < m.aCounter$
$\quad \vee \;\; \wedge \;\; m.type = $ "COMMIT"
$\quad\quad\quad \wedge \;\; b.counter \leq m.preparedCounter$
$\quad\quad\quad \wedge \;\; b.value = m.ballot.value$

whether b is aborted given $aCounter$ and prepared:
$Aborted(b, a, p) \triangleq$
$\quad \vee \;\; b.counter < a$
$\quad \vee \;\; LessThanAndIncompatible(b, p)$

update prepared and $aCounter$ given a new accepted-prepared ballot
$UpdatePrepared(n, b) \triangleq$
$\quad$ $TODO$: what's commented out might be needed for liveness:
$\quad$ IF $prepared[n] \prec b$
$\quad$ THEN
$\quad\quad \wedge \;\; prepared' = [prepared$ EXCEPT $![n] = b]$
$\quad\quad \wedge \;\;$ IF $prepared[n].counter > -1 \wedge prepared[n].value \neq b.value$
$\quad\quad\quad\quad$ THEN $aCounter' = [aCounter$ EXCEPT $![n] =$
$\quad\quad\quad\quad\quad$ IF $prepared[n].value < b.value$
$\quad\quad\quad\quad\quad\quad$ THEN $prepared[n].counter$
$\quad\quad\quad\quad\quad\quad$ ELSE $prepared[n].counter + 1]$
$\quad\quad\quad\quad$ ELSE UNCHANGED $aCounter$
$\quad$ ELSE
$\quad\quad$ IF $b.value \neq prepared[n].value \wedge b.counter \geq aCounter[n]$
$\quad\quad$ THEN $aCounter' = [aCounter$ EXCEPT $![n] =$
$\quad\quad\quad$ IF $prepared[n].value < b.value$
$\quad\quad\quad\quad$ THEN $prepared[n].counter$

$$\text{ELSE} \quad prepared[n].counter + 1]$$
$$\text{ELSE}$$
$$\text{ELSE} \quad \text{UNCHANGED} \ aCounter$$

Update what is accepted as prepared:
$AcceptPrepared(n,\ b) \ \triangleq$
  $\wedge\ \ prepared[n] \prec b$
  $\wedge\ \ \vee\ \exists\, Q \in Quorum : \forall\, m \in Q : \exists\, msg \in sent[m] : VotesToPrepare(b,\ msg)$
    $\vee\ \exists\, B \in BlockingSet : \forall\, m \in B : \exists\, msg \in sent[m] : AcceptsPrepared(b,\ msg)$
  $\wedge\ \ UpdatePrepared(n,\ b)$
  Reset c to $NullBallot$ if it has been aborted:
  $\wedge\ \ \text{IF}\ \ c[n].counter > -1 \wedge Aborted(c[n],\ aCounter'[n],\ prepared'[n])$
    $\text{THEN}\ \ c' = [c\ \text{EXCEPT}\ ![n] = NullBallot]$
    $\text{ELSE}\ \ \text{UNCHANGED}\ c$
  $\wedge\ \ \text{UNCHANGED}\ \langle ballot,\ phase,\ h,\ sent,\ byz \rangle$

Update what is confirmed as prepared:
$ConfirmPrepared(n,\ b) \ \triangleq$
  $\wedge\ \ h[n] \prec b$
  $\wedge\ \ \exists\, Q \in Quorum : \forall\, m \in Q : \exists\, msg \in sent[m] : AcceptsPrepared(b,\ msg)$
  $\wedge\ \ h' = [h\ \text{EXCEPT}\ ![n] = b]$
  TODO what if we confirm prepared something that's lower and incompatible with prepared?
  Should we update $aCounter$? (see commented-out part of $UpdatePrepared$)
  $\wedge\ \ \text{IF}\ prepared[n] \prec b$ confirmed prepared implies accepted prepared
    $\text{THEN}\ \ UpdatePrepared(n,\ b)$
    $\text{ELSE}\ \ \text{UNCHANGED}\ \langle prepared,\ aCounter \rangle$
  Update c (either reset to $NullBallot$, if it has been aborted, or set it to b):
  $\wedge\ \ \text{IF}\ \ \wedge\ \ c[n].counter > -1$
     $\wedge\ \ \vee\ \ Aborted(c[n],\ aCounter'[n],\ prepared'[n])$
      $\vee\ \ LessThanAndIncompatible(c[n],\ b)$
    $\text{THEN}\ \ c' = [c\ \text{EXCEPT}\ ![n] = NullBallot]$
    ELSE
     $\text{IF}\ \ \wedge\ \ c[n].counter = -1$
      $\wedge\ \ b = ballot[n]$
      $\wedge\ \ \neg Aborted(b,\ aCounter'[n],\ prepared'[n])$
     $\text{THEN}\ \ c' = [c\ \text{EXCEPT}\ ![n] = b]$
     $\text{ELSE}\ \ \text{UNCHANGED}\ c$
  $\wedge\ \ \text{IF}\ b.counter > 0 \wedge ballot[n] \prec b$
    $\text{THEN}\ \ ballot' = [ballot\ \text{EXCEPT}\ ![n] = b]$ not strictly necessary, but might help curb the statespace
    $\text{ELSE}\ \ \text{UNCHANGED}\ ballot$
  $\wedge\ \ \text{UNCHANGED}\ \langle phase,\ sent,\ byz \rangle$

NOTE this should be consistent with $LogicalMessages$
$VotesToCommit(b,\ m) \ \triangleq$
  $\vee\ \ \wedge\ \ m.type = \text{"PREPARE"}$
    $\wedge\ \ m.cCounter > 0$

$\quad\quad\quad\wedge\ \ m.cCounter \leq b.counter$
$\quad\quad\quad\wedge\ \ b.counter \leq m.hCounter$
$\quad\quad\quad\wedge\ \ b.value = m.ballot.value$
$\quad\quad\vee\ \ \wedge\ \ m.type = \text{``COMMIT''}$
$\quad\quad\quad\wedge\ \ m.cCounter \leq b.counter$
$\quad\quad\quad\wedge\ \ b.value = m.ballot.value$

NOTE this should be consistent with *LogicalMessages*
$AcceptsCommitted(b,\ m)\ \triangleq$
$\quad\quad\wedge\ \ m.type = \text{``COMMIT''}$
$\quad\quad\wedge\ \ b.value = m.ballot.value$
$\quad\quad\wedge\ \ m.cCounter \leq b.counter$
$\quad\quad\wedge\ \ b.counter \leq m.hCounter$

$AcceptCommitted(n,\ b)\ \triangleq$
$\quad\quad\wedge\ \ b = ballot[n]$ *TODO* okay?
$\quad\quad\wedge\ \ \text{IF}\ phase[n]\ \ = \text{``PREPARE''}$
$\quad\quad\quad\text{THEN}\ phase' = [phase\ \text{EXCEPT}\ ![n] = \text{``COMMIT''}] \wedge c' = [c\ \text{EXCEPT}\ ![n] = b]$
$\quad\quad\quad\text{ELSE}\ \ \text{UNCHANGED}\ \langle phase,\ c\rangle$
$\quad\quad\wedge\ \ phase[n] = \text{``COMMIT''} \Rightarrow h[n] \prec b$
$\quad\quad\wedge\ \ \vee\ \exists\, Q \in Quorum : \forall\, m \in Q : \exists\, msg \in sent[m] : VotesToCommit(b,\ msg)$
$\quad\quad\quad\ \vee\ \exists\, B \in BlockingSet : \forall\, m \in B : \exists\, msg \in sent[m] : AcceptsCommitted(b,\ msg)$
$\quad\quad\wedge\ \ h' = [h\ \text{EXCEPT}\ ![n] = b]$
$\quad\quad\wedge\ \ \text{IF}\ prepared[n] \prec b$ accepted committed implies accepted prepared
$\quad\quad\quad\text{THEN}\ UpdatePrepared(n,\ b)$
$\quad\quad\quad\text{ELSE}\ \ \text{UNCHANGED}\ \langle prepared,\ aCounter\rangle$
$\quad\quad\wedge\ \ \text{UNCHANGED}\ \langle ballot,\ sent,\ byz\rangle$

Now we specify what messages can be sent by a node

Summarize what has been prepared, under the constraint that prepared is less than or equal to ballot:
$SummarizePrepared(n)\ \triangleq$
$\quad\ \text{IF}\ prepared[n] \preceq ballot[n]$
$\quad\quad\text{THEN}\ [prepared \mapsto prepared[n],\ aCounter \mapsto aCounter[n]]$
$\quad\quad\text{ELSE}$
$\quad\quad\quad\text{IF}\ ballot[n].value > prepared[n].value \vee aCounter[n] > ballot[n].counter$
$\quad\quad\quad\quad\text{THEN}\ [$
$\quad\quad\quad\quad\quad prepared\ \ \mapsto [counter \mapsto ballot[n].counter,\ value \mapsto prepared[n].value],$
$\quad\quad\quad\quad\quad aCounter \mapsto Min(aCounter[n],\ ballot[n].counter)]$
$\quad\quad\quad\quad\text{ELSE}\ \ [$
$\quad\quad\quad\quad\quad prepared\ \ \mapsto [counter \mapsto ballot[n].counter - 1,\ value \mapsto prepared[n].value],$
$\quad\quad\quad\quad\quad aCounter \mapsto Min(aCounter[n],\ ballot[n].counter - 1)]$

$SendPrepare(n)\ \triangleq$
$\quad\quad\wedge\ \ ballot[n].counter > 0$
$\quad\quad\wedge\ \ phase[n] = \text{``PREPARE''}$

6

$\wedge$ LET $msg \triangleq [$
$\qquad type \mapsto$ "PREPARE"
$\qquad, \quad ballot \mapsto ballot[n]$
$\qquad, \quad prepared \mapsto SummarizePrepared(n).prepared$
$\qquad, \quad aCounter \mapsto SummarizePrepared(n).aCounter$
$\qquad, \quad hCounter \mapsto$
$\qquad\qquad$ IF $h[n].counter > -1 \wedge h[n].value = ballot[n].value$
$\qquad\qquad$ THEN $h[n].counter$
$\qquad\qquad$ ELSE 0
$\qquad, \quad cCounter \mapsto Max(c[n].counter, 0)]$
$\quad$ IN
$\qquad sent' = [sent$ EXCEPT $![n] = sent[n] \cup \{msg\}]$
$\wedge$ UNCHANGED $\langle ballot, phase, prepared, aCounter, h, c, byz \rangle$

$SendCommit(n) \triangleq$
$\quad \wedge phase[n] =$ "COMMIT"
$\quad \wedge$ LET $msg \triangleq [$
$\qquad type \mapsto$ "COMMIT"
$\qquad, \quad ballot \mapsto ballot[n]$
$\qquad, \quad preparedCounter \mapsto prepared[n].counter$
$\qquad, \quad hCounter \mapsto h[n].counter$
$\qquad, \quad cCounter \mapsto c[n].counter]$
$\quad$ IN
$\qquad sent' = [sent$ EXCEPT $![n] = sent[n] \cup \{msg\}]$
$\quad \wedge$ UNCHANGED $\langle ballot, phase, prepared, aCounter, h, c, byz \rangle$

$SendExternalize(n) \triangleq$
$\quad \wedge phase[n] =$ "EXTERNALIZE"
$\quad \wedge$ LET $msg \triangleq [$
$\qquad type \mapsto$ "EXTERNALIZE"
$\qquad, \quad commit \mapsto ballot[n]$
$\qquad, \quad hCounter \mapsto h[n].counter]$
$\quad$ IN
$\qquad sent' = [sent$ EXCEPT $![n] = sent[n] \cup \{msg\}]$
$\quad \wedge$ UNCHANGED $\langle ballot, phase, prepared, aCounter, h, c, byz \rangle$

We can now give the full specification

$Next \triangleq$
$\quad \vee ByzStep$
$\quad \vee \exists\, n \in N \setminus byz :$
$\qquad \vee \exists\, cnt \in BallotNumber : IncreaseBallotCounter(n, cnt)$
$\qquad \vee \exists\, b \in Ballot :$
$\qquad\quad \vee \; AcceptPrepared(n, b)$
$\qquad\quad \vee \; ConfirmPrepared(n, b)$
$\qquad\quad \vee \; AcceptCommitted(n, b)$

$$\lor\ SendPrepare(n)$$
$$\lor\ SendCommit(n)$$
$$\lor\ SendExternalize(n)$$

$vars\ \triangleq\ \langle ballot,\ phase,\ prepared,\ aCounter,\ h,\ c,\ sent,\ byz\rangle$

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

$Invariant\ \triangleq$
$\quad \land\ \ TypeOK$
$\quad \land\ \ \forall\, n \in N \setminus byz :$
$\qquad \land\ \ \forall\, m \in sent[n] : MessageInvariant(m)$
$\qquad \land\ \ ballot[n].counter = -1 \lor ballot[n].counter > 0$
$\qquad \land\ \ prepared[n].counter > -1 \Rightarrow aCounter[n] \leq prepared[n].counter$
$\qquad \land\ \ prepared[n].counter = -1 \Rightarrow aCounter[n] = 0$
$\qquad \land\ \ h[n] \preceq prepared[n]$
$\qquad \land\ \ c[n].counter = -1 \lor c[n].counter > 0$
$\qquad \land\ \ c[n].counter \leq h[n].counter$
$\qquad \land\ \ c[n].counter \leq ballot[n].counter$
$\qquad \land\ \ c[n].counter > 0 \Rightarrow$
$\qquad\qquad \land\ \ c[n].value = h[n].value$
$\qquad\qquad \land\ \ c[n].value = prepared[n].value$
$\qquad\qquad \land\ \ c[n].value = ballot[n].value$

Next we instantiate the *AbstractBalloting* specification

$voteToAbort\ \triangleq\ [n \in N \mapsto \text{UNION}\ \{LogicalMessages(m).voteToAbort : m \in sent[n]\}]$
$acceptedAborted\ \triangleq\ [n \in N \mapsto \text{UNION}\ \{LogicalMessages(m).acceptedAborted : m \in sent[n]\}]$
$confirmedAborted\ \triangleq\ [n \in N \mapsto \text{UNION}\ \{LogicalMessages(m).confirmedAborted : m \in sent[n]\}]$
$voteToCommit\ \triangleq\ [n \in N \mapsto \text{UNION}\ \{LogicalMessages(m).voteToCommit : m \in sent[n]\}]$
$acceptedCommitted\ \triangleq\ [n \in N \mapsto \text{UNION}\ \{LogicalMessages(m).acceptedCommitted : m \in sent[n]\}]$
$externalized\ \triangleq\ [n \in N \mapsto \{\}]$

$AB\ \triangleq\ \text{INSTANCE}\ AbstractBalloting$

We have a correct refinement:
THEOREM $Spec \Rightarrow AB!Spec$

To check the refinement with *TLC*:
$InitRefinement\ \triangleq$
$\quad Init \Rightarrow AB!Init$
$NextRefinement\ \triangleq$
$\quad \Box[Next \Rightarrow AB!Next]_{vars}$

Debugging canaries:

$Canary1 \triangleq \neg($
$\quad \exists\, n \in N \setminus byz : phase[n] = \text{"COMMIT"}$
$)$
$Canary2 \triangleq \neg($
$\quad \exists\, n \in N \setminus byz : \exists\, msg \in sent[n] :$
$\quad\quad \wedge\ msg.type = \text{"PREPARE"}$
$\quad\quad \wedge\ msg.cCounter = 1$
$)$
$Canary3 \triangleq \neg($
$\quad \exists\, Q \in Quorum : \forall\, n \in Q \setminus byz : c[n].counter = 1$
$)$