

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 { "PREPARE", "COMMIT", "EXTERNALIZE" }

SCPPrep \triangleq [
 type : { "PREPARE" }
 , *ballot* : *Ballot*
 , *prepared* : *BallotOrNull*
 , *aCounter* : *BallotNumber*
 , *hCounter* : *BallotNumber*
 , *cCounter* : *BallotNumber*]

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

SCPEExternalize \triangleq [
 type : { "EXTERNALIZE" }
 , *commit* : *Ballot*
 , *hCounter* : *BallotNumber*]

Message \triangleq
 SCPPrep \cup *SCPCommit* \cup *SCPEExternalize*

Some well-formedness conditions on messages:

MessageInvariant(*m*) \triangleq
 \wedge *m.type* = "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* = "COMMIT" \Rightarrow
 \wedge *m.cCounter* > 0

$$\begin{aligned} &\wedge m.cCounter \leq m.ballot.counter \\ &\wedge m.cCounter \leq m.hCounter \end{aligned}$$

TODO: Page 13 mentions that we should have $m.hCounter \leq m.ballot.counter$ in a *PREPARE* message

This seems superfluous.

I guess the sender should have increased its ballot counter before sending the message, but it's not a safety problem.

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 $m.type = \text{"PREPARE"} \rightarrow [$
 $voteToAbort \mapsto \{b \in Ballot :$
 $LessThanAndIncompatible(b, m.ballot)\},$
 $acceptedAborted \mapsto \{b \in Ballot :$
 $\vee LessThanAndIncompatible(b, m.prepared)$
 $\vee b.counter < m.aCounter\},$
 $confirmedAborted \mapsto$
 $IF m.hCounter = 0 THEN \{\}$
 $ELSE \{b \in Ballot :$
 $LET h \triangleq [counter \mapsto m.hCounter, value \mapsto m.ballot.value]$
 $IN LessThanAndIncompatible(b, h)\},$
 $voteToCommit \mapsto IF m.cCounter = 0 THEN \{\}$
 $ELSE \{b \in Ballot :$
 $\wedge m.cCounter \leq b.counter \wedge b.counter \leq m.hCounter$
 $\wedge b.value = m.ballot.value\},$
 $acceptedCommitted \mapsto \{\}]$
 $\square m.type = \text{"COMMIT"} \rightarrow [$
 $voteToAbort \mapsto \{b \in Ballot : b.value \neq m.ballot.value\},$
 $acceptedAborted \mapsto$
 $LET maxPrepared \triangleq [counter \mapsto m.preparedCounter, value \mapsto m.ballot.value]$
 $IN \{b \in Ballot : LessThanAndIncompatible(b, maxPrepared)\},$
 $confirmedAborted \mapsto$
 $LET maxPrepared \triangleq [counter \mapsto m.hCounter, value \mapsto m.ballot.value]$
 $IN \{b \in Ballot : LessThanAndIncompatible(b, maxPrepared)\},$
 $voteToCommit \mapsto \{b \in Ballot :$
 $m.cCounter \leq b.counter \wedge b.value = m.ballot.value\},$
 $acceptedCommitted \mapsto \{b \in Ballot :$
 $\wedge m.cCounter \leq b.counter \wedge b.counter \leq m.hCounter$
 $\wedge b.value = m.ballot.value\}]$

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

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

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

faulty nodes can send any message they want
 $ByzStep \triangleq \exists msgs \in [byz \rightarrow \text{SUBSET } Message] :$
 $\wedge sent' = [n \in N \mapsto \text{IF } n \notin byz \text{ THEN } sent[n] \text{ ELSE } msgs[n]]$
 $\wedge \text{UNCHANGED } \langle ballot, phase, prepared, aCounter, h, c, byz \rangle$

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.

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

```

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

VotesToPrepare( $b, m$ )  $\triangleq$ 
  ∨ ∧  $m.type = \text{"PREPARE"}$ 
    ∧ ∨ ∧  $b.counter \leq m.ballot.counter$ 
      ∧  $b.value = m.ballot.value$ 
    ∨ ∧  $b.counter \leq m.prepared.counter$ 
      ∧  $b.value = m.prepared.value$ 
    ∨  $b.counter < m.aCounter$ 
  ∨ ∧  $m.type = \text{"COMMIT"}$ 
    ∧  $b.value = m.ballot.value$ 

AcceptsPrepared( $b, m$ )  $\triangleq$ 
  ∨ ∧  $m.type = \text{"PREPARE"}$ 
    ∧ ∨ ∧  $b.counter \leq m.prepared.counter$ 
      ∧  $b.value = m.prepared.value$ 
    ∨  $b.counter < m.aCounter$ 
  ∨ ∧  $m.type = \text{"COMMIT"}$ 
    ∧  $b.counter \leq m.preparedCounter$ 
    ∧  $b.value = m.ballot.value$ 

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

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

```

```

ELSE prepared[n].counter + 1]
ELSE
ELSE UNCHANGED aCounter

```

Update what is accepted as prepared:

```

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

```

Update what is confirmed as prepared:

```

ConfirmPrepared(n, b)  $\triangleq$ 
   $\wedge$  h[n]  $\prec$  b
   $\wedge$   $\exists Q \in \text{Quorum} : \forall m \in Q : \exists \text{msg} \in \text{sent}[m] : \text{AcceptsPrepared}(b, \text{msg})$ 
   $\wedge$  h' = [h 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)
  IF prepared[n]  $\prec$  b confirmed prepared implies accepted prepared
    THEN UpdatePrepared(n, b)
    ELSE UNCHANGED  $\langle \text{prepared}, aCounter \rangle$ 
  Update c (either reset to nullBallot, if it has been aborted, or set it to b):
  IF  $\wedge$  c[n].counter > -1
     $\wedge$   $\forall$  Aborted(c[n], aCounter'[n], prepared'[n])
       $\vee$  LessThanAndIncompatible(c[n], b)
    THEN c' = [c EXCEPT ![n] = nullBallot]
    ELSE
      IF  $\wedge$  c[n].counter = -1
         $\wedge$  b = ballot[n]
         $\wedge$   $\neg$ Aborted(b, aCounter'[n], prepared'[n])
      THEN c' = [c EXCEPT ![n] = b]
      ELSE UNCHANGED c
  IF b.counter > 0  $\wedge$  ballot[n].counter < b.counter
    THEN ballot' = [ballot EXCEPT ![n] = b] not strictly necessary, but might help curb the statespace
    ELSE UNCHANGED ballot
   $\wedge$  UNCHANGED  $\langle \text{phase}, \text{sent}, \text{byz} \rangle$ 

```

NOTE this should be consistent with LogicalMessages

```

VotesToCommit(b, m)  $\triangleq$ 
   $\vee$   $\wedge$  m.type = "PREPARE"
   $\wedge$  m.cCounter > 0

```

$$\begin{aligned}
& \wedge m.cCounter \leq b.counter \\
& \wedge b.counter \leq m.hCounter \\
& \wedge b.value = m.ballot.value \\
\vee & \wedge m.type = \text{"COMMIT"} \\
& \wedge m.cCounter \leq b.counter \\
& \wedge b.value = m.ballot.value
\end{aligned}$$

NOTE this should be consistent with *LogicalMessages*

$AcceptsCommitted(b, m) \triangleq$

$$\begin{aligned}
& \wedge m.type = \text{"COMMIT"} \\
& \wedge b.value = m.ballot.value \\
& \wedge m.cCounter \leq b.counter \\
& \wedge b.counter \leq m.hCounter
\end{aligned}$$

$AcceptCommitted(n, b) \triangleq$

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

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:

$SummarizePrepared(n) \triangleq$

$$\begin{aligned}
& \text{IF } prepared[n] \preceq ballot[n] \\
& \quad \text{THEN } [prepared \mapsto prepared[n], aCounter \mapsto aCounter[n]] \\
& \quad \text{ELSE} \\
& \quad \text{IF } ballot[n].value > prepared[n].value \vee aCounter[n] > ballot[n].counter \\
& \quad \quad \text{THEN } [\\
& \quad \quad \quad prepared \mapsto [counter \mapsto ballot[n].counter, value \mapsto prepared[n].value], \\
& \quad \quad \quad aCounter \mapsto \text{Min}(aCounter[n], ballot[n].counter)] \\
& \quad \quad \text{ELSE} \\
& \quad \quad \text{IF } aCounter[n] = ballot[n].counter \\
& \quad \quad \quad \text{TODO okay?} \\
& \quad \quad \quad \text{THEN } [\\
& \quad \quad \quad \quad prepared \mapsto [counter \mapsto ballot[n].counter, value \mapsto ballot[n].value], \\
& \quad \quad \quad \quad aCounter \mapsto aCounter[n]] \\
& \quad \quad \quad \text{ELSE } [
\end{aligned}$$

$$\begin{aligned} prepared &\mapsto [counter \mapsto ballot[n].counter - 1, value \mapsto prepared[n].value], \\ aCounter &\mapsto Min(aCounter[n], ballot[n].counter - 1)] \end{aligned}$$

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

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

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

We can now give the full specification

$$\begin{aligned} Next &\triangleq \\ &\vee ByzStep \end{aligned}$$

$$\begin{aligned}
& \vee \exists n \in N \setminus \text{byz} : \\
& \quad \vee \exists \text{cnt} \in \text{BallotNumber} : \text{IncreaseBallotCounter}(n, \text{cnt}) \\
& \quad \vee \exists b \in \text{Ballot} : \\
& \quad \quad \vee \text{AcceptPrepared}(n, b) \\
& \quad \quad \vee \text{ConfirmPrepared}(n, b) \\
& \quad \quad \vee \text{AcceptCommitted}(n, b) \\
& \quad \vee \text{SendPrepare}(n) \\
& \quad \vee \text{SendCommit}(n) \\
& \quad \vee \text{SendExternalize}(n)
\end{aligned}$$

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

$$\begin{aligned}
\text{Spec} & \triangleq \\
& \text{Init} \wedge \Box[\text{Next}]_{\text{vars}}
\end{aligned}$$

We now turn to correctness properties

$$\begin{aligned}
\text{Invariant} & \triangleq \\
& \wedge \text{TypeOK} \\
& \wedge \forall n \in N \setminus \text{byz} : \\
& \quad \wedge \forall m \in \text{sent}[n] : \text{MessageInvariant}(m) \\
& \quad \wedge \text{ballot}[n].\text{counter} = -1 \vee \text{ballot}[n].\text{counter} > 0 \\
& \quad \wedge \text{prepared}[n].\text{counter} > -1 \Rightarrow \text{aCounter}[n] \leq \text{prepared}[n].\text{counter} \\
& \quad \wedge \text{prepared}[n].\text{counter} = -1 \Rightarrow \text{aCounter}[n] = 0 \\
& \quad \wedge h[n] \preceq \text{prepared}[n] \\
& \quad \wedge c[n].\text{counter} = -1 \vee c[n].\text{counter} > 0 \\
& \quad \wedge c[n].\text{counter} \leq h[n].\text{counter} \\
& \quad \wedge c[n].\text{counter} \leq \text{ballot}[n].\text{counter} \\
& \quad \wedge c[n].\text{counter} > 0 \Rightarrow \\
& \quad \quad \wedge c[n].\text{value} = h[n].\text{value} \\
& \quad \quad \wedge c[n].\text{value} = \text{prepared}[n].\text{value} \\
& \quad \quad \wedge c[n].\text{value} = \text{ballot}[n].\text{value}
\end{aligned}$$

Next we instantiate the *AbstractBalloting* specification

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

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

We have a correct refinement:

THEOREM $\text{Spec} \Rightarrow \text{AB!Spec}$

To check the refinement with *TLC*:

$$InitRefinement \triangleq$$

$$AB!Init$$

$$NextRefinement \triangleq$$

$$\Box[AB!Next]_{vars}$$