MODULE *NominationPlusCal*

This is a high-level specification of *SCP* focusing on the nomination protocol.

Currently, as implemented, before voting for a txset hash, nodes wait to obtain its preimage. Delaying the point at which we wait for the pre-image would leave more room for disseminating the txset in parallel to nomination. However this has to be done carefully to maintain the main property of nomination: assuming that there is a nomination round with a good leader and during which the network is fast engough, at least a Tier-1 quorum must eventually enter *balloting*.

In the version specified in this document, we do not wait on the pre-image to vote for a txset hash, but we do wait for the pre-image before accepting it.

In the previous version of this document, we even accepted without a pre-image. There is a problem with this: it could create a situation in which not enough nodes can start *balloting* (*i.e.* not a full quorum) and the whole system is stuck.

The problem stems from the fact that, in the nomination protocol, nodes that confirm a candidate then stop voting for new values (otherwise nomination is not guaranteed to converge). So if a blocking set B confirms a candidate but somehow other nodes cannot get the pre-images they need to do so, more nomination rounds will not help because the members of B have stopped voting, which blocks the progress of any new candidate. Depending on how pre-images are disseminated, this can potentially be exploited by an attacker to halt the system.

So accepting without a pre-image is only workable if there is some way to guarantee that, once a Tier-1 blocking set has a pre-image, then everybody in Tier-1 eventually gets it.

Another problem is that we want it to be likely that a quorum starts *balloting* already in agreement and roughly at the same time. If we delay checking pre-images to the confirm stage, an attacker could first send the pre-image to a set A of nodes, which then enter *balloting* at time T_A, but not send the pre-image to another set B of nodes, which then enter *balloting* at time $T\_B \rangle T\_A$ because they need to get the pre-image from A before starting *balloting*. For example, if it takes $500ms$ for members of B to get the pre-image from members of A, then $T\_B = T\_A + 500ms$. This can cause the first ballot to end without a decision. Members of B could also start a new nomination round before T_B and then enter *balloting* not only late but also with a different value than members of A.

EXTENDS *Naturals*, *FiniteSets*

CONSTANTS

```
    $V$,  validators
    $TxSet$,  blocks
    $Bot$,  default value
    $Quorum(\_)$,  $Quorum(v)$ is the set of quorums of validator $v$
    $Blocking(\_)$,  $Blocking(v)$ is the set of blocking sets of validator $v$
    $Combine(\_)$,  the functions that combines candidates to produce a txset for balloting
    $H$,  domain of hashes
    $Hash(\_)$  hash function
```

```
  --algorithm $SCP${
   variables  global variables (e.g. representing messages or cross-component variables like ballotingTxSet):
       $ballotingTxSet = [v \in V \mapsto Bot]$;  for each validator, the nominated txset for balloting
       $decision = [v \in V \mapsto Bot]$;  for each validator, the balloting decision
       $voted = [v \in V \mapsto \{\}]$;  X in the whitepaper (nomination Section)
```

```
        $accepted = [v \in V \mapsto \{\}]$ ;   Y in the whitepaper (nomination Section)
    process ( $nomination \in V$ )
        variables   local variables:
            $round = 0$ ;   nothing happens in round 0; the protocol start at round 1
            $candidates = \{\}$ ;   Z in the whitepaper (nomination Section)
            $preImage = [h \in H \mapsto Bot]$ ;   the pre-images the validator knows about
            $leader = Bot$ ;   leader for the current round
    {
$ln1$:  while ( TRUE )
        either {   timeout and go to next round (this also starts round 1)
            $round := round + 1$ ;
            with ( $l \in V$ ) {   pick a leader
                $leader := l$ ;
                if ( $l = self$ )   if the leader is the current node, pick a txset and vote for it
                with ( $txs \in TxSet$ ) {
                    $preImage[Hash(txs)] := txs$ ;
                    $voted[self] := voted[self] \cup \{Hash(txs)\}$
                 }
             }
         }
        or if ( $candidates = \{\}$ ) {   vote for what the leader voted for, unless we have a candidate already
            when $leader \neq Bot$ ;
            with ( $hs = voted[leader]$ ) {
                await $hs \neq \{\}$ ;   wait to hear from the leader
                $voted[self] := voted[self] \cup hs$   vote for what the leader has voted for
                 in the whitepaper version, we would only vote for the hashes for which we have a pre-image:
                 with ($hsWithPreimage = \{h \in hs : preImage[h] \neq Bot\}$)
                 $voted[self] := voted[self] \cup hsWithPreimage$
             }
         }
        or with ( $Q \in Quorum(self), h \in H$ ) {   accept when voted or accepted by a quorum and we have the pre-im
            when $preImage[h] \neq Bot$ ;   we must have received the block
            when $\forall w \in Q : h \in voted[w] \vee h \in accepted[w]$ ;   a quorum has voted or accepted $h$:
            $accepted[self] := accepted[self] \cup \{h\}$ ;   accept $h$
         }
        or with ( $Bl \in Blocking(self), h \in H$ ) {   accept when accepted by a blocking set and we have the pre-image
            when $preImage[h] \neq Bot$ ;   we must have received the block
            when $\forall w \in Bl : h \in accepted[w]$ ;
            $accepted[self] := accepted[self] \cup \{h\}$ ;   accept $h$
         }
        or with ( $txs \in TxSet$ ) {   receive a txset
            $preImage[Hash(txs)] := txs$ ;
         }
        or with ( $Q \in Quorum(self), h \in H$ ) {   confirm $b$ as candidate
            when $preImage[h] \neq Bot$ ;   we must have received the block
            when $\forall w \in Q : h \in accepted[w]$ ;   a quorum has accepted $h$:
            $candidates := candidates \cup \{preImage[h]\}$ ;   add $h$ to the confirmed candidates
             update the block used in *balloting*:
            $ballotingTxSet[self] := Combine(candidates)$ ;   this starts the *balloting* protocol (see below)
        }
    }
    as a first approximation, *balloting* just decides on one of the *balloting* blocks:
    note we cannot reuse the process ID identifiers used in nomination, so we add the "balloting" tag
    process ( $balloting \in \{\langle v, \text{"balloting"}\rangle : v \in V\}$ ) {
lb1:   await $ballotingTxSet[self[1]] \neq Bot$ ;   wait for a confirmed candidate from nomination
lb2:   with ( $b \in \{ballotingTxSet[v] : v \in V\} \setminus \{Bot\}$ ) {
           when $\forall w \in V : decision[w] \neq Bot \Rightarrow b = decision[w]$ ;
           $decision[self[1]] := b$ ;
       }
    }
}
```

BEGIN TRANSLATION ($chksum(pcal) =$ "$b24885fb$" $\wedge$ $chksum(tla) =$ "$95d64d8b$")

VARIABLES $ballotingTxSet$, $decision$, $voted$, $accepted$, pc, $round$, $candidates$, $preImage$, $leader$

$vars \triangleq \langle ballotingTxSet, decision, voted, accepted, pc, round, candidates, preImage, leader\rangle$

$ProcSet \triangleq (V) \cup (\{\langle v, \text{"balloting"}\rangle : v \in V\})$

```
Init ≜  Global variables
        $\wedge\ ballotingTxSet = [v \in V \mapsto Bot]$
        $\wedge\ decision = [v \in V \mapsto Bot]$
        $\wedge\ voted = [v \in V \mapsto \{\}]$
        $\wedge\ accepted = [v \in V \mapsto \{\}]$
        Process nomination
        $\wedge\ round = [self \in V \mapsto 0]$
        $\wedge\ candidates = [self \in V \mapsto \{\}]$
        $\wedge\ preImage = [self \in V \mapsto [h \in H \mapsto Bot]]$
        $\wedge\ leader = [self \in V \mapsto Bot]$
        $\wedge\ pc = [self \in ProcSet \mapsto$ CASE $self \in V \rightarrow$ "ln1"
                                  $\Box$  $self \in \{\langle v, \text{"balloting"}\rangle : v \in V\} \rightarrow$ "lb1"]

ln1(self) ≜  $\wedge\ pc[self] =$ "ln1"
             $\wedge\ \vee\ \wedge\ round' = [round$ EXCEPT $![self] = round[self] + 1]$
                    $\wedge\ \exists\, l \in V$ :
                        $\wedge\ leader' = [leader$ EXCEPT $![self] = l]$
                        $\wedge$ IF $l = self$
                             THEN $\wedge\ \exists\, txs \in TxSet$ :
                                      $\wedge\ preImage' = [preImage$ EXCEPT $![self][Hash(txs)] = txs]$
                                      $\wedge\ voted' = [voted$ EXCEPT $![self] = voted[self] \cup \{Hash(txs)\}]$
                    ELSE  $\land$ TRUE
                          $\land$ UNCHANGED $\langle voted, preImage \rangle$
            $\land$ UNCHANGED $\langle ballotingTxSet, accepted, candidates \rangle$
         $\lor$ $\land$ IF $candidates[self] = \{\}$
                  THEN $\land$ $leader[self] \neq Bot$
                       $\land$ LET $hs \triangleq voted[leader[self]]$ IN
                            $\land$ $hs \neq \{\}$
                            $\land$ $voted' = [voted$ EXCEPT $![self] = voted[self] \cup hs]$
                  ELSE $\land$ TRUE
                       $\land$ $voted' = voted$
            $\land$ UNCHANGED $\langle ballotingTxSet, accepted, round, candidates, preImage, leader \rangle$
         $\lor$ $\land$ $\exists\, Q \in Quorum(self)$ :
                 $\exists\, h \in H$ :
                    $\land$ $preImage[self][h] \neq Bot$
                    $\land$ $\forall\, w \in Q : h \in voted[w] \lor h \in accepted[w]$
                    $\land$ $accepted' = [accepted$ EXCEPT $![self] = accepted[self] \cup \{h\}]$
            $\land$ UNCHANGED $\langle ballotingTxSet, voted, round, candidates, preImage, leader \rangle$
         $\lor$ $\land$ $\exists\, Bl \in Blocking(self)$ :
                 $\exists\, h \in H$ :
                    $\land$ $preImage[self][h] \neq Bot$
                    $\land$ $\forall\, w \in Bl : h \in accepted[w]$
                    $\land$ $accepted' = [accepted$ EXCEPT $![self] = accepted[self] \cup \{h\}]$
            $\land$ UNCHANGED $\langle ballotingTxSet, voted, round, candidates, preImage, leader \rangle$
         $\lor$ $\land$ $\exists\, txs \in TxSet$ :
                 $preImage' = [preImage$ EXCEPT $![self][Hash(txs)] = txs]$
            $\land$ UNCHANGED $\langle ballotingTxSet, voted, accepted, round, candidates, leader \rangle$
         $\lor$ $\land$ $\exists\, Q \in Quorum(self)$ :
                 $\exists\, h \in H$ :
                    $\land$ $preImage[self][h] \neq Bot$
                    $\land$ $\forall\, w \in Q : h \in accepted[w]$
                    $\land$ $candidates' = [candidates$ EXCEPT $![self] = candidates[self] \cup \{preImage[self][h]\}]$
                    $\land$ $ballotingTxSet' = [ballotingTxSet$ EXCEPT $![self] = Combine(candidates'[self])]$
            $\land$ UNCHANGED $\langle voted, accepted, round, preImage, leader \rangle$
       $\land$ $pc' = [pc$ EXCEPT $![self] =$ "ln1"$]$
       $\land$ UNCHANGED $decision$

$nomination(self) \triangleq ln1(self)$

$lb1(self) \triangleq$ $\land$ $pc[self] =$ "lb1"
                 $\land$ $ballotingTxSet[self[1]] \neq Bot$
                 $\land$ $pc' = [pc$ EXCEPT $![self] =$ "lb2"$]$
                 $\land$ UNCHANGED $\langle ballotingTxSet, decision, voted, accepted, round,$
                                  $candidates, preImage, leader \rangle$

$lb2(self) \triangleq$ $\land$ $pc[self] =$ "lb2"
                 $\land$ $\exists\, b \in \{ballotingTxSet[v] : v \in V\} \setminus \{Bot\}$ :
```

$$\begin{aligned}
&\qquad\qquad \land \forall w \in V : \mathit{decision}[w] \neq \mathit{Bot} \Rightarrow b = \mathit{decision}[w] \\
&\qquad\qquad \land \mathit{decision}' = [\mathit{decision} \text{ EXCEPT } ![\mathit{self}[1]] = b] \\
&\qquad \land \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\mathit{self}] = \text{“Done”}] \\
&\qquad \land \text{UNCHANGED } \langle \mathit{ballotingTxSet}, \mathit{voted}, \mathit{accepted}, \mathit{round}, \\
&\qquad\qquad\qquad\qquad \mathit{candidates}, \mathit{preImage}, \mathit{leader} \rangle
\end{aligned}$$

$\mathit{balloting}(\mathit{self}) \triangleq \mathit{lb}1(\mathit{self}) \lor \mathit{lb}2(\mathit{self})$

$$\begin{aligned}
\mathit{Next} \triangleq\ & (\exists\, \mathit{self} \in V : \mathit{nomination}(\mathit{self})) \\
& \lor (\exists\, \mathit{self} \in \{\langle v, \text{“balloting”}\rangle : v \in V\} : \mathit{balloting}(\mathit{self}))
\end{aligned}$$

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

END TRANSLATION

The type-safety invariant:

$$\begin{aligned}
&\mathit{TypeOkay} \triangleq \\
&\quad \land \mathit{ballotingTxSet} \in [V \to \mathit{TxSet} \cup \{\mathit{Bot}\}] \\
&\quad \land \mathit{decision} \in [V \to \mathit{TxSet} \cup \{\mathit{Bot}\}] \\
&\quad \land \mathit{voted} \in [V \to \text{SUBSET } H] \\
&\quad \land \mathit{accepted} \in [V \to \text{SUBSET } H] \\
&\quad \land \mathit{round} \in [V \to \mathit{Nat}] \\
&\quad \land \mathit{candidates} \in [V \to \text{SUBSET } \mathit{TxSet}] \\
&\quad \land \mathit{preImage} \in [V \to [H \to \mathit{TxSet} \cup \{\mathit{Bot}\}]] \\
&\quad \land \mathit{leader} \in [V \to V \cup \{\mathit{Bot}\}]
\end{aligned}$$

Next we specify a liveness property that we can easily check with the *TLC* model-checker.

This property is that, if a validator *v* enters *balloting*, then eventually all validators enter *balloting*. This will hold in simple configurations where the whole network is top tier.

For the property to hold, we also need to add fairness assumptions (*e.g.* if a node can vote for a value, it will eventually do so). Unfortunately the TLA+ code generated from the *PlusCal* specification is in a form that makes stating fairness assumptions hard. It seems that we would need to rewrite this in pure TLA+ to tackle liveness.

$$\begin{aligned}
&\mathit{NominationLiveness} \triangleq \\
&\quad \forall v, w \in V : \Box(\mathit{ballotingTxSet}[v] \neq \mathit{Bot} \Rightarrow \Diamond(\mathit{ballotingTxSet}[w] \neq \mathit{Bot}))
\end{aligned}$$

Definition for model-checking:

Concrete hashing for the model-checker:

$\mathit{TestH} \triangleq 1 \mathrel{..} \mathit{Cardinality}(\mathit{TxSet})$

$$\begin{aligned}
&\mathit{TestHash}(b) \triangleq \\
&\quad \text{LET } f \triangleq \text{CHOOSE } f \in [\mathit{TxSet} \to H] : \forall\, \mathit{txs}1, \mathit{txs}2 \in \mathit{TxSet} : \mathit{txs}1 \neq \mathit{txs}2 \Rightarrow f[\mathit{txs}1] \neq f[\mathit{txs}2] \\
&\quad \text{IN } \ f[b]
\end{aligned}$$

Debugging canaries:

$\mathit{Canary}1 \triangleq \forall v \in V : \mathit{decision}[v] = \mathit{Bot}$

$Canary2 \triangleq \forall v \in V : Cardinality(candidates[v]) \leq 1$
$Canary3 \triangleq \forall v \in V : ballotingTxSet[v] = Bot$

$TestQuorums \triangleq \{Q \in \text{SUBSET } V : 2 * Cardinality(Q) > Cardinality(V)\}$
$TestBlocking \triangleq \{Bl \in \text{SUBSET } V : Cardinality(Bl) > 1\}$

\\ * Modification History
\\ * Last modified *Thu Mar* 30 20:16:04 *PDT* 2023 by *nano*
\\ * Created *Thu Mar* 30 20:15:37 *PDT* 2023 by *nano*