MODULE *VDFConsensus*

EXTENDS *FiniteSets*, *Integers*, *TLC*

CONSTANTS
P the set of processes
, B the set of malicious processes
, $tAdv$ the time it takes for a malicious process to produce a message
, tWB the time it takes for a well-behaved process to produce a message

ASSUME $B \subseteq P$ malicious processes are a subset of all processes
$W \triangleq P \setminus B$ the set of well-behaved processes

$Tick \triangleq Nat$ a tick is a real-time clock tick
$Round \triangleq Nat$ a round is just a tag on a message

Processes build a DAG of messages. The message-production rate of well-behaved processes is of 1 message per tWB ticks, and that of malicious processes is of 1 message per $tAdv$ ticks. We require that, collectively, well-behaved processes produce messages at a rate strictly higher than that of malicious processes.

ASSUME $Cardinality(W) * tAdv > Cardinality(B) * tWB$

$MessageID \triangleq Nat$

A message consists of a unique ID, a round number, and a coffer containing the IDs of a set of predecessor messages:
$Message \triangleq [sender : P, id : MessageID, round : Round, coffer : \text{SUBSET } MessageID]$

We will need the intersection of a set of sets:
RECURSIVE $Intersection(\_)$
$Intersection(Ss) \triangleq$
CASE
$Ss = \{\} \rightarrow \{\}$
$\Box\ \exists S \in Ss : Ss = \{S\} \rightarrow \text{CHOOSE } S \in Ss : Ss = \{S\}$
$\Box$ OTHER $\rightarrow$
LET $S \triangleq (\text{CHOOSE } S \in Ss : \text{TRUE})$
IN $S \cap Intersection(Ss \setminus \{S\})$

A set of messages is consistent when the intersection of the coffers of each message is a strict majority of the coffer of each message.

$ConsistentSet(M) \triangleq$
LET $I \triangleq Intersection(\{m.coffer : m \in M\})$
IN $\forall m \in M : 2 * Cardinality(I) > Cardinality(m.coffer)$

A consistent chain is a subset of the messages in the DAG that potentially has some dangling pointers (*i.e.* messages that have predecessors not in the chain) and that satisfies the following recursive predicate:

* Any set of messages which all have a round of 0 is a consistent chain.

* A set of messages C with some non-zero rounds and maximal round r is a consistent chain when, with *Tip* being the set of messages in the chain that have round r and *Pred* being the set of messages in the chain with round $r - 1$, *Pred* is a strict majority of the set of predecessors of each message in *Tip* and $C \setminus Tip$ is a consistent chain. (Note that this implies that *Tip* is a consistent set)

$$
\begin{array}{l}
Max(X,\ Leq(\_,\ \_)) \triangleq \\
\quad \text{CHOOSE } m \in X : \forall\, x \in X : Leq(x,\ m)
\end{array}
$$

$$
\begin{array}{l}
\text{RECURSIVE } ConsistentChain(\_) \\
ConsistentChain(M) \triangleq \\
\quad \text{IF } M = \{\} \\
\quad \text{THEN FALSE} \\
\quad \text{ELSE LET } r \triangleq Max(\{m.round : m \in M\},\ \leq) \text{IN} \\
\qquad \lor\ r = 0 \\
\qquad \lor\ \text{LET } Tip \triangleq \{m \in M : m.round = r\} \\
\qquad\qquad\quad Pred \triangleq \{m \in M : m.round = r - 1\} \\
\qquad\quad \text{IN} \quad \land\ Tip \neq \{\} \\
\qquad\qquad\quad \land\ \forall\, m \in Tip : \\
\qquad\qquad\qquad \land\, \forall\, m2 \in Pred : m2.id \in m.coffer \\
\qquad\qquad\qquad \land\ 2 * Cardinality(Pred) > Cardinality(m.coffer) \\
\qquad\qquad\quad \land\ ConsistentChain(M \setminus Tip)
\end{array}
$$

Given a message DAG, the heaviest consistent chain is a consistent chain in the DAG that has a maximal number of messages.

$$
\begin{array}{l}
HeaviestConsistentChain(M) \triangleq \\
\quad \text{LET } r \triangleq Max(\{m.round : m \in M\},\ \leq) \\
\qquad Cs \triangleq \{C \in \text{SUBSET } M : (\exists\, m \in C : m.round = r) \land ConsistentChain(C)\} \\
\quad \text{IN} \\
\qquad \text{IF } Cs = \{\} \text{ THEN } \{\} \\
\qquad \text{ELSE } Max(Cs,\ \text{LAMBDA } C1,\ C2 : Cardinality(C1) \leq Cardinality(C2))
\end{array}
$$

Now we specify the algorithm

--algorithm *Algo***{**

variables

$messages = \{\}$ **;**

$tick = 0$ **;**

$phase =$ "start" **;** each tick has two phases: "start" and "end"

$donePhase = [p \in P \mapsto \text{"end"}]$ **;**

$pendingMessage = [p \in P \mapsto \langle\rangle]$ **;**

$messageCount = 0$ **;** used to generate unique message *IDs*

define {

$currentRound \triangleq tick \div tWB$ round of well-behaved processes

$wellBehavedMessages \triangleq \{m \in messages : m.sender \in P \setminus B\}$

possible sets of messages received by a well-behaved process:

$receivedMsgsSets \triangleq$

```
            ignore messages from future rounds:
            LET msgs ≜ {m ∈ messages : m.round < currentRound} IN
            {wellBehavedMessages ∪ byzMsgs :
                byzMsgs ∈ SUBSET (msgs \ wellBehavedMessages)}
  }
  macro sendMessage( m ) {
      messages := messages ∪ {m}
  }
  process ( clock ∈ {"clock"} ) {
tick:  while ( TRUE ) {
          await ∀ p ∈ P : donePhase[p] = phase ;
          if ( phase = "start" )
              phase := "end"
          else {
              phase := "start" ;
              tick := tick + 1
          }
      }
  }
  process ( proc ∈ P \ B )   a well-behaved process
  {
l1:    while ( TRUE ) {
          await phase = "start" ;
          if ( tick%tWB = 0 ) {
               Start the VDF computation for the next message:
              with ( msgs ∈ receivedMsgsSets )
              with ( hCC = HeaviestConsistentChain(msgs) )
              with ( predMsgs = {m ∈ hCC : m.round = currentRound − 1} ) {
                  pendingMessage[self] := [
                      sender ↦ self,
                      id ↦ messageCount + 1,
                      round ↦ currentRound,
                      coffer ↦ {m.id : m ∈ predMsgs}] ;
                  messageCount := messageCount + 1 ;
              }
          } ;
          donePhase[self] := "start" ;
l2:       await phase = "end" ;
          if ( tick%tWB = tWB − 1 )
               it's the end of the tWB period, the VDF has been computed
              sendMessage(pendingMessage[self]) ;
          donePhase[self] := "end" ;
      }
  }
  process ( byz ∈ B )   a malicious process
```

$\{$

$lb1$: **while** (TRUE) $\{$

await $phase = $ "start" ;

if ($tick \% tAdv = 0$) $\{$

Start the VDF computation for the next message:

with ($maxRound = Max(\{m.round : m \in messages\} \cup \{0\}, \ \leq)$)

with ($rnd \in \{maxRound, \ maxRound + 1\}$)

with ($predMsgs \in \text{SUBSET } \{m \in messages : m.round = rnd - 1\}$) $\{$

when $rnd > 0 \Rightarrow predMsgs \neq \{\}$;

$pendingMessage[self] := [$

$sender \mapsto self,$

$id \mapsto messageCount + 1,$

$round \mapsto rnd,$

$coffer \mapsto \{m.id : m \in predMsgs\}]$;

$messageCount := messageCount + 1$;

$\}$

$\}$;

$donePhase[self] :=$ "start" ;

$lb2$: **await** $phase = $ "end" ;

if ($tick \% tAdv = tAdv - 1$)

$sendMessage(pendingMessage[self])$;

$donePhase[self] :=$ "end" ;

$\}$;

$\}$

$\}$

$TypeOK \triangleq$

$\land \ messages \in \text{SUBSET } Message$

$\land \ pendingMessage \in [P \rightarrow Message \cup \{\langle\rangle\}]$

$\land \ tick \in Tick$

$\land \ phase \in \{\text{"start"}, \text{"end"}\}$

$\land \ donePhase \in [P \rightarrow \{\text{"start"}, \text{"end"}\}]$

$\land \ messageCount \in Nat$

$messageWithID(id) \triangleq \text{CHOOSE } m \in messages : m.id = id$

The main property we want to establish is that, each round, for each message m of a well-behaved process, the messages of well-behaved processes from the previous round are all in m's coffer and consist of a strict majority of m's coffer.

$Safety \triangleq \forall m \in messages : m.round > 0 \land m.sender \notin B \Rightarrow$

$\land \ \forall m2 \in wellBehavedMessages : m2.round = m.round - 1 \Rightarrow m2.id \in m.coffer$

$\land \ \text{LET } M \triangleq \{m2 \in wellBehavedMessages : m2.round = m.round - 1\}$

$\quad \text{IN} \quad 2 * Cardinality(M) > Cardinality(m.coffer)$

A basic well-formedness property:

$Inv1 \triangleq \forall m \in messages : \forall id \in m.coffer :$

$\land \ \exists m2 \in messages : m2.id = id$

$\land \quad messageWithID(id).round = m.round - 1$