MODULE *VDFConsensus*

EXTENDS *FiniteSets*, *Integers*, *Utils*

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

A message consists of a unique *ID*, a round number, and a coffer containing the *IDs* of a set of predecessor messages:
$MessageID \triangleq P \times Nat$
$Message \triangleq [id : MessageID, round : Round, coffer : \text{SUBSET } MessageID]$
$sender(m) \triangleq m.id[1]$

A strongly 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 strongly consistent chain.

* A set of messages C with some non-zero rounds and maximal round r is a strongly 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.

```
RECURSIVE StronglyConsistentChain(_)
StronglyConsistentChain(M) ≜
  ∧ M ≠ {}
  ∧ LET r ≜ Max({m.round : m ∈ M}, ≤) IN
      ∨ r = 0
      ∨ LET Tip ≜ {m ∈ M : m.round = r}
            Pred ≜ {m ∈ M : m.round = r − 1}
        IN  ∧ Tip ≠ {}
            ∧ ∀ m ∈ Tip :
                ∧ ∀ m2 ∈ Pred : m2.id ∈ m.coffer
                ∧ 2 * Cardinality(Pred) > Cardinality(m.coffer)
            ∧ StronglyConsistentChain(M \ Tip)
```

A weaker version of the above:

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

The max round of a set of messages is the maximal round of its messages:

$$\mathit{MaxRound}(M) \triangleq \mathit{MaxInteger}(\{m.\mathit{round} : m \in M\})$$

$$\mathit{SubsetsWithMaxRound}(M,\ r) \triangleq \{M2 \in \text{SUBSET } M : \exists\, m \in M2 : m.\mathit{round} = r\}$$

The set of all consistent chains that can be found in M:

$$
\begin{array}{l}
\mathit{ConsistentChains}(M,\ r) \triangleq \\
\quad \{C \in \mathit{SubsetsWithMaxRound}(M,\ r) : \mathit{ConsistentChain}(C)\}
\end{array}
$$

The set of all strongly consistent chains that can be found in M:

$$
\begin{array}{l}
\mathit{StronglyConsistentChains}(M,\ r) \triangleq \\
\quad \{C \in \mathit{SubsetsWithMaxRound}(M,\ r) : \mathit{StronglyConsistentChain}(C)\}
\end{array}
$$

$$\mathit{HeaviestConsistentChains}(M,\ r) \triangleq \mathit{MaxCardinalitySets}(\mathit{ConsistentChains}(M,\ r))$$

$$\mathit{HeaviestStronglyConsistentChains}(M,\ r) \triangleq \mathit{MaxCardinalitySets}(\mathit{StronglyConsistentChains}(M,\ r))$$

Two chains are disjoint when there is a round smaller than their max round in which they have no messages in common. We assume $C1$ and $C2$ have the same max round.

$$
\begin{array}{l}
\mathit{DisjointChains}(C1,\ C2) \triangleq \\
\quad \text{LET } \mathit{rmax} \triangleq \mathit{Max}(\{m.\mathit{round} : m \in C1 \cup C2\},\ \leq) \\
\quad \text{IN} \quad \exists\, r \in 0\,..\,(\mathit{rmax} - 1) \quad : \\
\qquad\qquad \{m \in C1 : m.\mathit{round} = r\} \cap \{m \in C2 : m.\mathit{round} = r\} = \{\}
\end{array}
$$

Acceptance rule

$$
\begin{array}{l}
\mathit{AcceptedMessages}(M,\ r) \triangleq \{m \in M : \\
\quad \land\ m.\mathit{round} = r - 1 \\
\quad \land\ \text{LET } \mathit{HSCCs} \triangleq \mathit{HeaviestStronglyConsistentChains}(M,\ r - 1) \text{IN} \\
\qquad \land\ \exists\, C \in \mathit{HSCCs} : m \in C \\
\qquad \land\ \forall\, C1,\ C2 \in \mathit{HSCCs} :
\end{array}
$$

$\land\ m \in C1$
$\land\ m \notin C2$
$\land\ DisjointChains(C1,\ C2)$
$\Rightarrow Cardinality(C2) \leq Cardinality(C1)\}$

M does not having dangling edges:
$Closed(M) \triangleq \forall\, m \in M : \forall\, i \in m.coffer : \exists\, m2 \in M : m2.id = i$

Now we specify the algorithm

```
--algorithm Algo{
  variables
    messages = {} ;
    tick = 0 ;
    phase = "start" ;   each tick has two phases: "start" and "end"
    donePhase = [p ∈ P ↦ "end"] ;
    pendingMessage = [p ∈ P ↦ ⟨⟩] ;   message we're computing the VDF on
    messageCount = [p ∈ P ↦ 0] ;   used to generate unique message IDs
  define {
    currentRound ≜ tick ÷ tWB   round of well-behaved processes
    wellBehavedMessages ≜ {m ∈ messages : sender(m) ∈ P \ B}
     possible sets of messages received by a well-behaved process:
    receivedMsgsSets ≜
        ignore messages from future rounds:
      LET msgs ≜ {m ∈ messages : m.round < currentRound}IN
      {M ∈ SUBSET msgs :
          don't use a set of messages that has dangling edges (messages in coffers that are missing):
        ∧ Closed(M)
        ∧ wellBehavedMessages ⊆ M}
   }
  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 ( predMsgs = AcceptedMessages(msgs, currentRound) ) {
                  pendingMessage[self] := [
                      id ↦ ⟨self, messageCount[self] + 1⟩,
                      round ↦ currentRound,
                      coffer ↦ {m.id : m ∈ predMsgs}] ;
                  messageCount[self] := messageCount[self] + 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 ∈ messages} ∪ {0}, ≤ ) )
              with ( rnd ∈ {maxRound, maxRound + 1} )
              with ( predMsgs ∈ SUBSET {m ∈ messages : m.round = rnd − 1} ) {
                  when rnd > 0 ⇒ predMsgs ≠ {} ;
                  pendingMessage[self] := [
                      id ↦ ⟨self, messageCount[self] + 1⟩,
                      round ↦ rnd,
                      coffer ↦ {m.id : m ∈ predMsgs}] ;
                  messageCount[self] := messageCount[self] + 1
              }
           } ;
          donePhase[self] := "start" ;
lb2:      await phase = "end" ;
          if ( tick%tAdv = tAdv − 1 )
              sendMessage(pendingMessage[self]) ;
          donePhase[self] := "end" ;
       } ;
```

```
    }
}
```

Invariant describing the type of the variables:

$$
\begin{aligned}
&\mathit{TypeOK} \triangleq \\
&\quad \land\ \mathit{messages} \in \text{SUBSET } \mathit{Message} \\
&\quad \land\ \mathit{pendingMessage} \in [P \to \mathit{Message} \cup \{\langle\rangle\}] \\
&\quad \land\ \mathit{tick} \in \mathit{Tick} \\
&\quad \land\ \mathit{phase} \in \{\text{“start”}, \text{“end”}\} \\
&\quad \land\ \mathit{donePhase} \in [P \to \{\text{“start”}, \text{“end”}\}] \\
&\quad \land\ \mathit{messageCount} \in [P \to \mathit{Nat}]
\end{aligned}
$$

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.

$$
\begin{aligned}
&\mathit{Safety} \triangleq \forall\, p \in P \setminus B : \text{LET } m \triangleq \mathit{pendingMessage}[p] \text{IN} \\
&\quad \land\ m \neq \langle\rangle \\
&\quad \land\ m.\mathit{round} > 0 \\
&\quad \Rightarrow \\
&\quad \land\ \forall\, m2 \in \mathit{wellBehavedMessages} : m2.\mathit{round} = m.\mathit{round} - 1 \Rightarrow m2.\mathit{id} \in m.\mathit{coffer} \\
&\quad \land\ \text{LET } M \triangleq \{m2 \in \mathit{wellBehavedMessages} : m2.\mathit{round} = m.\mathit{round} - 1\} \\
&\qquad \text{IN} \quad 2 * \mathit{Cardinality}(M) > \mathit{Cardinality}(m.\mathit{coffer})
\end{aligned}
$$

helper definition:

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

Basic well-formedness properties:

$$
\begin{aligned}
&\mathit{Inv}1 \triangleq \forall\, m \in \mathit{messages} : \\
&\quad \land\ \forall\, m2 \in \mathit{messages} : m \neq m2 \Rightarrow m.\mathit{id} \neq m2.\mathit{id} \\
&\quad \land\ \forall\, \mathit{id} \in m.\mathit{coffer} : \\
&\qquad \land\ \exists\, m2 \in \mathit{messages} : m2.\mathit{id} = \mathit{id} \\
&\qquad \land\ \mathit{messageWithID}(\mathit{id}).\mathit{round} = m.\mathit{round} - 1
\end{aligned}
$$