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

TODO: *I* think we're going to need $Cardinality(W) * tAdv > 2 * Cardinality(B) * tWB$

A message consists of a unique *ID*, a round number, and a coffer containing the *IDs* of a set of predecessor messages:

$MessageID \triangleq P \times Nat$

$Message \triangleq [id : MessageID, round : Round, coffer : \text{SUBSET } MessageID]$

$sender(m) \triangleq m.id[1]$

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)

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

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

$$
\begin{array}{l}
MaximalElements(X,\ Leq(\_,\ \_)) \triangleq \\
\quad \{m \in X : \forall\, x \in X : \neg(Leq(m,\ x) \land \neg Leq(x,\ m))\}
\end{array}
$$

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

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

$$
\begin{array}{l}
ConsistentChains(M) \triangleq \\
\quad \text{LET } r \triangleq Max(\{m.round : m \in M\},\ \leq) \\
\quad \text{IN} \quad \{C \in \text{SUBSET } M : (\exists\, m \in C : m.round = r) \land ConsistentChain(C)\}
\end{array}
$$

$$
StronglyConsistentChains(M) \triangleq
$$

$$
\begin{array}{l}
\quad \text{LET } r \triangleq Max(\{m.round : m \in M\}, \leq) \\
\quad \text{IN} \quad \{C \in \text{SUBSET } M : (\exists\, m \in C : m.round = r) \land StronglyConsistentChain(C)\}
\end{array}
$$

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

$$
\begin{array}{l}
HeaviestConsistentChain(M) \triangleq \\
\quad \text{LET } CCs \triangleq ConsistentChains(M) \\
\quad \text{IN} \\
\qquad \text{IF } CCs = \{\} \text{ THEN } \{\} \\
\qquad \ \text{ELSE } Max(CCs, \text{LAMBDA } C1, C2 : Cardinality(C1) \leq Cardinality(C2))
\end{array}
$$

$$
\begin{array}{l}
HeaviestConsistentChains(M) \triangleq \\
\quad \text{LET } CCs \triangleq ConsistentChains(M) \\
\quad \text{IN} \quad MaximalElements(CCs, \text{LAMBDA } C1, C2 : Cardinality(C1) \leq Cardinality(C2))
\end{array}
$$

Two chains are disjoint when there is a round in which they have no messages in common:

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

$\text{RECURSIVE } ComponentOf(\_, \_)$

The connected component of chain C amongs chains Cs

$$
\begin{array}{l}
ComponentOf(C, Cs) \triangleq \\
\quad \text{IF } \exists\, C2 \in Cs : \neg DisjointChains(C, C2) \\
\quad\ \text{THEN} \\
\qquad \text{LET } C2 \triangleq \text{CHOOSE } C2 \in Cs : \neg DisjointChains(C, C2) \\
\qquad \text{IN} \quad ComponentOf(C \cup C2, Cs \setminus \{C2\}) \\
\quad\ \text{ELSE } C
\end{array}
$$

$\text{RECURSIVE } Components(\_)$

All the components in Cs:

$$
\begin{array}{l}
Components(Cs) \triangleq \\
\quad \text{IF } Cs = \{\} \text{ THEN } \{\} \\
\quad\ \text{ELSE} \\
\qquad \text{LET } C \triangleq \text{CHOOSE } C \in Cs : \text{TRUE} \\
\qquad\qquad Comp \triangleq ComponentOf(C, Cs) \\
\qquad\ \text{IN} \quad \{Comp\} \cup Components(\{C2 \in Cs : DisjointChains(C2, Comp)\})
\end{array}
$$

$$
\begin{array}{l}
HeaviestComponent(M) \triangleq \\
\quad \text{LET } Comps \triangleq Components(StronglyConsistentChains(M)) \\
\quad \text{IN} \quad m \\
\qquad \text{IF } Comps = \{\} \text{ THEN } \{\} \\
\qquad \ \text{ELSE } Max(Comps, \text{LAMBDA } C1, C2 : Cardinality(C1) \leq Cardinality(C2))
\end{array}
$$

Now we specify the algorithm

```
--algorithm Algo{
  variables
     messages = {} ;
     tick = 0 ;
     phase = "start" ;   each tick has two phases: "start" and "end"
     donePhase = [p ∈ P ↦ "end"] ;
     pendingMessage = [p ∈ P ↦ ⟨⟩] ;
     messageCount = [p ∈ P ↦ 0] ;
  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
        {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 ( C = HeaviestComponent(msgs) )
           with ( predMsgs = {m ∈ C : m.round = currentRound − 1} ) {
              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" ;
        } ;
    }
}
```

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

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

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.

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

Basic well-formedness properties:

$$
\begin{array}{l}
\mathit{Inv1} \triangleq \forall\, m \in \mathit{messages} : \\
\quad \land\ \forall\, m2 \in \mathit{messages} : m \neq m2 \Rightarrow m.\mathit{id} \neq m2.\mathit{id} \\
\quad \land\ \forall\, \mathit{id} \in m.\mathit{coffer} : \\
\quad\quad \land\ \exists\, m2 \in \mathit{messages} : m2.\mathit{id} = \mathit{id} \\
\quad\quad \land\ \mathit{messageWithID}(\mathit{id}).\mathit{round} = m.\mathit{round} - 1
\end{array}
$$