MODULE *Sailfish*1

Specification of the a variant of the *Sailfish* consensus algorithm at a high level of abstraction.

In this variant, a leader vertice is commited only when it has a quorum (*e.g.* $2f + 1$) of *DAG* parents in the next round.

EXTENDS *DomainModel*, *TLC*

CONSTANT
GST the first synchronous round (all later rounds are synchronous)

```
--algorithm Sailfish{
  variables
      vs = {},  the vertices of the DAG
      es = {},  the edges of the DAG
      no_vote = [n ∈ N ↦ {}];  no_vote messages sent by each node
  define {
      LeaderVertice(r) ≜ ⟨Leader(r), r⟩
      VerticeQuorums(r) ≜
          {VQ ∈ SUBSET vs :
                ∧ ∀ v ∈ VQ : Round(v) = r
                ∧ {Node(v) : v ∈ VQ} ∈ Quorum}
   }
  process ( correctNode ∈ N \ F )
      variables round = 0;  current round
  {
l0:   while ( TRUE )
      either with ( v = ⟨self, round⟩ ) {
           add a new vertex to the DAG and go to the next round
          vs := vs ∪ {v};
          if ( round > 0 )
          with ( vq ∈ VerticeQuorums(round − 1) ) {
               from GST onwards, each node receives all correct vertices of the previous round:
              when round ≥ GST ⇒ (N \ F) ⊆ {Node(v2) : v2 ∈ vq};
              es := es ∪ {⟨v, pv⟩ : pv ∈ vq};  add the edges
              if ( LeaderVertice(round − 1) ∉ vq )  send no_vote if previous leader vertice not included
                  no_vote[self] := no_vote[self] ∪ {LeaderVertice(round − 1)}
           } ;
          round := round + 1
       }
      or with ( r ∈ {r ∈ R : r > round} ) {
           go to a higher round
          when round < GST;  from GST onwards, correct nodes do not skip rounds
          round := r
       }
   }
```

Next comes our model of *Byzantine* nodes. Because the real protocol disseminates *DAG* vertices using reliable broadcast, *Byzantine* nodes cannot equivocate and cannot deviate much from the protocol (lest their messages be ignored).

$$
\begin{array}{ll}
 & \quad \textbf{process} \ (\ \mathit{byzantineNode} \in F \ ) \\
 & \qquad \textbf{variables} \ \mathit{round\_} = 0 \, \textbf{;} \\
 & \quad \textbf{\{} \\
l0: & \qquad \textbf{while} \ (\ \text{TRUE} \ ) \ \textbf{\{} \\
 & \qquad\qquad \text{maybe add a vertices to the } \mathit{DAG}\text{:} \\
 & \qquad\qquad \textbf{either with} \ (\ v = \langle \mathit{self}, \ \mathit{round\_} \rangle \ ) \ \textbf{\{} \\
 & \qquad\qquad\quad \mathit{vs} := \mathit{vs} \cup \{v\} \, \textbf{;} \\
 & \qquad\qquad\quad \textbf{if} \ (\ \mathit{round\_} > 0 \ ) \\
 & \qquad\qquad\qquad \textbf{with} \ (\ \mathit{vq} \in \mathit{VerticeQuorums}(\mathit{round\_} - 1) \ ) \\
 & \qquad\qquad\qquad\quad \mathit{es} := \mathit{es} \cup \{\langle v, \ \mathit{pv} \rangle : \mathit{pv} \in \mathit{vq}\} \\
 & \qquad\qquad \textbf{\} or skip ;} \\
 & \qquad\qquad \text{maybe send a } \mathit{no\_vote} \text{ messages:} \\
 & \qquad\qquad \textbf{if} \ (\ \mathit{round\_} > 0 \ ) \\
 & \qquad\qquad \textbf{either} \\
 & \qquad\qquad\quad \mathit{no\_vote}[\mathit{self}] := \mathit{no\_vote}[\mathit{self}] \cup \{\mathit{LeaderVertice}(\mathit{round\_} - 1)\} \\
 & \qquad\qquad \textbf{or skip ;} \\
 & \qquad\qquad \text{go to the next round:} \\
 & \qquad\qquad \mathit{round\_} := \mathit{round\_} + 1 \\
 & \qquad \textbf{\}} \\
 & \quad \textbf{\}} \\
\textbf{\}} &
\end{array}
$$

Next we define the safety and liveness properties

$$
\begin{array}{l}
\mathit{Committed}(v) \triangleq \\
\quad \land \ v \in \mathit{vs} \\
\quad \land \ \mathit{Node}(v) = \mathit{Leader}(\mathit{Round}(v)) \\
\quad \land \ \{\mathit{Node}(\mathit{pv}) : \mathit{pv} \in \mathit{Parents}(v, \ \mathit{es})\} \in \mathit{Quorum} \\
\quad \land \ \lor \ \mathit{Round}(v) = 0 \\
\quad \quad\;\; \lor \ \mathit{LeaderVertice}(\mathit{Round}(v) - 1) \in \mathit{Children}(v, \ \mathit{es}) \\
\quad \quad\;\; \lor \ \exists \, Q \in \mathit{Quorum} : \forall \, n \in Q : \\
\quad \quad\quad\quad\quad \mathit{LeaderVertice}(\mathit{Round}(v) - 1) \in \mathit{no\_vote}[n] \\
\mathit{Safety} \triangleq \forall \, v1, \ v2 \in \mathit{vs} : \\
\quad \land \ \mathit{Committed}(v1) \\
\quad \land \ \mathit{Committed}(v2) \\
\quad \land \ \mathit{Round}(v1) \leq \mathit{Round}(v2) \\
\quad \Rightarrow \mathit{Reachable}(v2, \ v1, \ \mathit{es})
\end{array}
$$

$$
\begin{array}{l}
\mathit{Liveness} \triangleq \forall \, r \in R : \\
\quad \land \ r \geq \mathit{GST} \\
\quad \land \ \mathit{Leader}(r) \notin F \\
\quad \land \ \forall \, n \in N \setminus F : \mathit{round}[n] > r + 1 \\
\quad \Rightarrow \mathit{Committed}(\mathit{LeaderVertice}(r))
\end{array}
$$

Finally we make a few auxiliary definitions used for model-checking with TLC

The round of a node, whether *Byzantine* or not

$$Round\_(n) \triangleq \text{IF } n \in F \text{ THEN } round\_[n] \text{ ELSE } round[n]$$

Basic typing invariant:

$$
\begin{aligned}
&TypeOK \triangleq \\
&\quad \land\ \forall v \in vs : Node(v) \in N \land Round(v) \in Nat \\
&\quad \land\ \forall e \in es : \\
&\qquad\quad \land\ e = \langle e[1],\ e[2] \rangle \\
&\qquad\quad \land\ \{e[1],\ e[2]\} \subseteq vs \\
&\qquad\quad \land\ Round(e[1]) > Round(e[2]) \\
&\quad \land\ \forall n \in N : \\
&\qquad \land\ Round\_(n) \in Nat \\
&\qquad \land\ no\_vote[n] \subseteq \{\langle Leader(r),\ r \rangle : r \in R\}
\end{aligned}
$$

Sequentialization constraints, which enforce a particular ordering of the actions. Because of how actions commute, the set of reachable states remains unchanged. This speeds up model-checking a lot.

$$SeqConstraints(n) \triangleq$$

wait for all nodes to finish previous rounds:

$$\land (Round\_(n) > 0 \Rightarrow \forall n2 \in N : Round\_(n2) \geq Round\_(n))$$

wait for all nodes with lower index to leave the round:

$$\land \forall n2 \in N : NodeIndex(n2) < NodeIndex(n) \Rightarrow Round\_(n2) > Round\_(n)$$

$$
\begin{aligned}
SeqNext &\triangleq (\exists self \in N \setminus F : SeqConstraints(self) \land correctNode(self)) \\
&\qquad \lor (\exists self \in F : SeqConstraints(self) \land byzantineNode(self)) \\
SeqSpec &\triangleq Init \land \Box[SeqNext]_{vars}
\end{aligned}
$$

Example assignment of leaders to rounds:

$$ModLeader(r) \triangleq NodeSeq[(r\%Cardinality(N)) + 1]$$

Constraint to stop the model checker:

$$
\begin{aligned}
&StateConstraint \triangleq \\
&\quad \text{LET } Max(S) \triangleq \text{CHOOSE } x \in S : \forall y \in S : y \leq x \text{IN} \\
&\qquad \forall n \in N : Round\_(n) \in 0 \,..\, (Max(R) + 1)
\end{aligned}
$$

Some properties we expect to be violated:

$$
\begin{aligned}
&Falsy1 \triangleq \neg(\\
&\quad \land Committed(\langle Leader(1),\ 1 \rangle) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&Falsy2 \triangleq \neg(\\
&\quad \land Committed(\langle Leader(0),\ 0 \rangle) \\
&\quad \land \neg Committed(\langle Leader(1),\ 1 \rangle) \\
&\quad \land \neg Committed(\langle Leader(2),\ 2 \rangle)
\end{aligned}
$$

$\quad \land\ \textit{Committed}(\langle \textit{Leader}(3),\ 3\rangle)$

$)$