MODULE *ReliableBroadcast*

```
EXTENDS LearnerGraph, FiniteSets

CONSTANTS
    LG,   the learner graph
    B,    the set of malicious acceptors
    W,    the set of well-behaved acceptors, i.e. honest and available
    V     the set of values that can be broadcast

ASSUME $B \cap W = \{\}$

ASSUME $IsValidLearnerGraph(LG)$

$Learner \triangleq LG.learners$
$Acceptor \triangleq LG.acceptors$

$HonestAcceptor \triangleq Acceptor \setminus B$
  Note that HonestAcceptor is not necessary equal to W


  --algorithm ReliableBroadcast{
    variables
        $bcast \in (\text{SUBSET } V) \setminus \{\{\}\}$ ;   the value(s) broadcast; multiple values model a malicious sender
        $echo = [a \in Acceptor \mapsto \{\}]$ ;
        $ready = [a \in Acceptor \mapsto [l \in Learner \mapsto \{\}]]$ ;
    define {
        $ProvenMalicious(a) \triangleq \exists v1, v2 \in V :$
            $v1 \neq v2 \land \{v1, v2\} \subseteq echo[a]$
        $NotEntangled(l1, l2) \triangleq$
            $\land\ l1 \neq l2$  a learner is always entangled with itself
            $\land\ \forall S \in LG.safeSets[\langle l1, l2 \rangle] :$
                $\exists a \in S : ProvenMalicious(a)$
    }
    fair process ( $learner \in Learner$ )
        variables
            $output = \langle\rangle$ ;
    {
l0:     with ( $v \in V$ ) {
            when $\exists Q \in LG.quorums[self] :$
                $\forall a \in Q : v \in ready[a][self]$ ;
            $output := v$ ;
        }
    }
    process ( $acceptor \in HonestAcceptor$ ) {
$l0$:    while ( TRUE )
        either
            with ( $v \in V$ ) {
                when $v \in bcast \land echo[self] = \{\}$ ;
                $echo[self] := echo[self] \cup \{v\}$ ;
            }
        or
            with ( $v \in V$ )
            with ( $l \in Learner$ )
            with ( $Q \in LG.quorums[l]$ ) {
                when $ready[self][l] = \{\}$ ;
                when $\forall a \in Q : v \in echo[a]$ ;
                  check for conflicts:
                when $\forall l2 \in Learner \setminus \{l\} : \forall v2 \in V \setminus \{v\}$ :
                    $v2 \in ready[self][l2] \Rightarrow NotEntangled(l, l2)$ ;
                $ready[self][l] := ready[self][l] \cup \{v\}$ ;
            }
        or
            with ( $v \in V$ )
            with ( $l1 \in Learner, l2 \in Learner$ ) {
                when $\forall Q \in LG.quorums[l1] : \exists a2 \in Q : v \in ready[a2][l2]$ ;
                  check for conflicts:
                when $\forall l3 \in Learner : \forall v2 \in V \setminus \{v\}$ :
                    $v2 \in ready[self][l3] \Rightarrow NotEntangled(l1, l3)$ ;
                $ready[self][l1] := ready[self][l1] \cup \{v\}$ ;
            }
        }
    process ( $byzAcceptor \in B$ ) {
$l0$:    while ( TRUE ) {
            either
            with ( $v \in V$ )
                $echo[self] := echo[self] \cup \{v\}$
            or
            with ( $l \in Learner$ ) {
                with ( $v \in V$ )
                    $ready[self][l] := ready[self][l] \cup \{v\}$ ;
            }
        }
    }
}
```

```
$TypeOK \triangleq$
    $\land\ bcast \in (\text{SUBSET } V) \setminus \{\{\}\}$
    $\land\ echo \in [Acceptor \rightarrow (\text{SUBSET } V)]$
    $\land\ ready \in [Acceptor \rightarrow [Learner \rightarrow (\text{SUBSET } V)]]$
    $\land\ output \in [Learner \rightarrow V \cup \{\langle\rangle\}]$
```

Two learners must agree if one of their safe sets is fully well-behaved:

```
$Entangled(l1, l2) \triangleq \exists S \in LG.safeSets[\langle l1, l2\rangle] :$
    $S \cap B = \{\}$

$LiveLearner \triangleq \{l \in Learner :$
    $\exists Q \in LG.quorums[l] : Q \subseteq W\}$

$Safety \triangleq$
    $\land\ \forall l \in Learner :$
        $\land\ pc[l] = \text{“Done”}$
        $\land\ \exists Q \in LG.quorums[l] : Q \cap B = \{\}$
        $\Rightarrow output[l] \in bcast$
    $\land\ \forall l1, l2 \in Learner :$
        $\land\ Entangled(l1, l2)$
        $\land\ pc[l1] = \text{“Done”}$
        $\land\ pc[l2] = \text{“Done”}$
        $\Rightarrow output[l1] = output[l2]$

$Liveness \triangleq$
    $\land\ Cardinality(bcast) = 1 \Rightarrow$
            $\forall l \in LiveLearner : \Diamond(pc[l] = \text{“Done”} \land bcast = \{output[l]\})$
     This one is interesting (I think this is the best we can guarantee):
    $\land\ \forall l1 \in Learner : \forall l2 \in LiveLearner : Entangled(l1, l2) \Rightarrow$
            $\Box(pc[l1] = \text{“Done”} \Rightarrow \Diamond(pc[l2] = \text{“Done”}))$

$FairSpec \triangleq$
    $\land\ Spec$
    $\land\ \forall a \in W : \text{WF}_{vars}(acceptor(a))$
```