```
──────────────────── MODULE TwoStepOptimiticBroadcast ────────────────────
EXTENDS FiniteSets, Integers, TLC

CONSTANTS
    P  the set of parties
,   Faulty  the set of faulty parties
,   V  the set of value that may be broadcast

N ≜ Cardinality(P)
F ≜ Cardinality(Faulty)

ASSUME Faulty ⊆ P ∧  N > 3 ∗ F

CeilDiv(a, b) ≜ IF a%b = 0 THEN a ÷ b ELSE (a ÷ b) + 1

Message ≜   the set of possible messages in the network
    [src : P, dst : P, type : {"propose", "echo", "vote", "ready"}, val : V]

  --algorithm Broadcast{
    variables
        broadcaster ∈ {CHOOSE p ∈ Faulty : TRUE, CHOOSE p ∈ P \ Faulty : TRUE},   the distinguised broadca
        bcastValue = CHOOSE v ∈ V : TRUE,   the value to broadcast (faulty nodes will ignore)
        msgs = {} ;   the set of sent messages
    define {
        Msgs(self, v, type) ≜ {m ∈ msgs : m.type = type ∧ m.val = v ∧ m.dst = self}
        Echos(self, v) ≜ Msgs(self, v, "echo")
        Votes(self, v) ≜ Msgs(self, v, "vote")
        Readys(self, v) ≜ Msgs(self, v, "ready")
     }
    macro SendAll( type, value ) {
        msgs := msgs ∪ {[src ↦ self, dst ↦ d, type ↦ type, val ↦ value] : d ∈ P}
     }
    fair process ( correctParty ∈ P \ Faulty )
        variable delivered = ⟨⟩ ;   the delivered value
    {
l0:     while ( TRUE ) with ( v ∈ V ) {
            either {   send proposal
                when self = broadcaster ;
                when ∀ m ∈ msgs : ¬(m.src = self ∧ m.type = "propose") ;
                SendAll("propose", bcastValue)
             }
            or {   send echo
                when ∀ m ∈ msgs : ¬(m.src = self ∧ m.type = "echo") ;
                await [src ↦ broadcaster, dst ↦ self, type ↦ "propose", val ↦ v] ∈ msgs ;
                SendAll("echo", v)
             }
            or {  fast delivery
                await $Cardinality(Echos(self, v) \setminus \{broadcaster\}) \geq CeilDiv(N + 2 * F - 2, 2)$ ;
                $delivered := v$
             }
            or {  send vote
                when $\forall m \in msgs : \neg(m.src = self \land m.type = \text{"vote"})$ ;
                await $Cardinality(Echos(self, v) \setminus \{broadcaster\}) \geq CeilDiv(N, 2)$ ;
                $SendAll(\text{"vote"}, v)$
             }
            or {  send ready
                when $\forall m \in msgs : \neg(m.src = self \land m.type = \text{"ready"})$ ;
                await
                    $\lor\ Cardinality(Echos(self, v) \setminus \{broadcaster\}) \geq CeilDiv(N + F - 1, 2)$
                    $\lor\ Cardinality(Votes(self, v) \setminus \{broadcaster\}) \geq CeilDiv(N + F - 1, 2)$
                    $\lor\ Cardinality(Readys(self, v)) \geq F + 1$ ;
                $SendAll(\text{"ready"}, v)$
             }
            or {  slow delivery
                await $Cardinality(Readys(self, v)) \geq 2 * F + 1$ ;
                $delivered := v$
             }
          }
      }
      process ( $faultyParty \in Faulty$ ) {
$l1$:     with ( $v \in V,\ t \in \{\text{"propose"}, \text{"echo"}, \text{"vote"}, \text{"ready"}\},\ d \in P \setminus Faulty$ )
              $msgs := msgs \cup \{[src \mapsto self, dst \mapsto d, type \mapsto t, val \mapsto v]\}$
      }
 }
```

Correctness properties:

$$Agreement \triangleq \forall p1, p2 \in P \setminus Faulty : delivered[p1] \neq \langle\rangle \land delivered[p2] \neq \langle\rangle \Rightarrow delivered[p1] = delivered[p2]$$

$$Liveness \triangleq \land\ (broadcaster \notin Faulty \Rightarrow \forall p \in P \setminus Faulty : \Diamond(delivered[p] = bcastValue)) \quad \land\ \Box((\exists p \in P \setminus Faulty : delivered[p] \neq \langle\rangle) \Rightarrow \forall p \in P \setminus Faulty : \Diamond(delivered[p] \neq \langle\rangle))$$

$$Symm \triangleq Permutations(P \setminus (Faulty \cup \{\text{CHOOSE } p \in P \setminus Faulty : \text{TRUE}\}))$$