MODULE *TwoStepOptimiticBroadcastSafety*

This is a version of *TwoStepOptimiticBroadcast* that is optimized for model-checking safety properties of the protocol.

In order to reduce the state-space to explore, we do not model explicitly the echo, vote, and ready messages of *Byzantine* parties. Instead, we modify the threshold guards of honest party transitions such that the new guard is satisfied iff it is possible to come up with *Byzantine* messages that will make the original guard satisfied. For example, if an honest party originally waits for $2f+1$ ready messages, then the new guard is to wait for $f+1$ messages from honest parties.

EXTENDS *FiniteSets*, *Integers*, *TLC*

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

$N \triangleq Cardinality(P)$
$F \triangleq Cardinality(Faulty)$
$FNB \triangleq Cardinality(Faulty \setminus \{Broadcaster\})$

ASSUME $Faulty \subseteq P \land \; N > 3 * F$

Integer division, rounded up:
$CeilDiv(a, b) \triangleq \text{IF } a\%b = 0 \text{ THEN } a \div b \text{ ELSE } (a \div b) + 1$

The set of possible messages in the network:
$Message \triangleq$
  $[src : P, dst : P, type : \{\text{"proposal"}, \text{"echo"}, \text{"vote"}, \text{"ready"}\}, val : V]$

--algorithm *Broadcast***{**
  variables
    $msgs = \{\}$ **;** the set of sent messages
  define {
    $Msgs(self, v, type) \triangleq \{m \in msgs : m.type = type \land m.val = v \land m.dst = self\}$
    $Echos(self, v) \triangleq Msgs(self, v, \text{"echo"})$
    $Votes(self, v) \triangleq Msgs(self, v, \text{"vote"})$
    $Readys(self, v) \triangleq Msgs(self, v, \text{"ready"})$
  }
  macro *SendAll*(*type*, *value*) **{**
    $msgs := msgs \cup \{[src \mapsto self, dst \mapsto d, type \mapsto type, val \mapsto value] : d \in P\}$
  }
  fair process ($correctParty \in P \setminus Faulty$ **)**
    variable $delivered = \langle\rangle$ **;** the delivered value
  {
$l0$: **while (** TRUE **) with (** $v \in V$ **) {**

either { send proposal

when $self = Broadcaster$ **;**

when $\forall\, m \in msgs : \neg(m.src = self \land m.type =$ "proposal" $)$ **;**

with ($proposal \in V$ **)**

$SendAll($"proposal"$,\ proposal)$

}

or { send echo

when $\forall\, m \in msgs : \neg(m.src = self \land m.type =$ "echo" $)$ **;**

await $[src \mapsto Broadcaster,\ dst \mapsto self,\ type \mapsto$ "proposal"$,\ val \mapsto v] \in msgs$ **;**

$SendAll($"echo"$,\ v)$

}

or { fast delivery

await $Cardinality(\{m \in Echos(self,\ v) : m.src \neq Broadcaster\}) + FNB \geq CeilDiv(N + 2 *$

$delivered := v$

}

or { send vote

when $\forall\, m \in msgs : \neg(m.src = self \land m.type =$ "vote" $)$ **;**

await $Cardinality(\{m \in Echos(self,\ v) : m.src \neq Broadcaster\}) + FNB \geq CeilDiv(N,\ 2)$ **;**

$SendAll($"vote"$,\ v)$

}

or { send ready

when $\forall\, m \in msgs : \neg(m.src = self \land m.type =$ "ready" $)$ **;**

await

$\lor\ Cardinality(\{m \in Echos(self,\ v) : m.src \neq Broadcaster\}) + FNB \geq CeilDiv(N + F$

$\lor\ Cardinality(\{m \in Votes(self,\ v) : m.src \neq Broadcaster\}) + FNB \geq CeilDiv(N + F$

$\lor\ Cardinality(Readys(self,\ v)) + F \geq F + 1$ **;**

$SendAll($"ready"$,\ v)$

}

or { slow delivery

await $Cardinality(Readys(self,\ v)) + F \geq 2 * F + 1$ **;**

$delivered := v$

}

}

}

process ($faultyParty \in Faulty$ **) {**

a faulty broadcaster may equivocate on the proposal:

$l1$: **while (** TRUE **)**

with ($v \in V,\ d \in P \setminus Faulty$ **) {**

$msgs := msgs \cup \{[src \mapsto self,\ dst \mapsto d,\ type \mapsto$ "proposal"$,\ val \mapsto v]\}$

}

}

}

Correctness properties:

$$
\begin{aligned}
&\mathit{TypeOK} \triangleq \\
&\quad \land\ \forall m \in msgs : m \in Message \\
&\quad \land\ \forall p\ \in P \setminus Faulty : delivered[p] \in \{\langle\rangle\} \cup V
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{ReadySame} \triangleq \forall m1,\ m2 \in msgs : \\
&\quad \land\ m1.src \notin Faulty \land m2.src \notin Faulty \\
&\quad \land\ m1.type = \text{“ready”} \land m2.type = \text{“ready”} \\
&\quad \Rightarrow \\
&\qquad m1.val = m2.val
\end{aligned}
$$

to find an execution in which all correct parties deliver:

$$
\begin{aligned}
&\mathit{Falsy} \triangleq \neg(\\
&\quad \forall p \in P \setminus Faulty : delivered[p] \neq \langle\rangle \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{Agreement} \triangleq \forall p1,\ p2 \in P \setminus Faulty : \\
&\quad delivered[p1] \neq \langle\rangle \land delivered[p2] \neq \langle\rangle \Rightarrow delivered[p1] = delivered[p2]
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{Liveness} \triangleq \\
&\quad \land\ (Broadcaster \notin Faulty \Rightarrow \forall p \in P \setminus Faulty : \\
&\qquad\quad \Diamond(\exists v \in V : \\
&\qquad\qquad \land\ [src \mapsto Broadcaster,\ dst \mapsto p,\ type \mapsto \text{“proposal”},\ val \mapsto v] \in msgs \\
&\qquad\qquad \land\ delivered[p] = v)) \\
&\quad \land\ \Box((\exists p \in P \setminus Faulty : delivered[p] \neq \langle\rangle) \Rightarrow \forall p \in P \setminus Faulty : \Diamond(delivered[p] \neq \langle\rangle))
\end{aligned}
$$

Symmetry specification for the *TLC* model-checker:

$$
\mathit{Symm} \triangleq Permutations(P \setminus (Faulty \cup \{Broadcaster\})) \cup Permutations(V)
$$