MODULE *HPaxosProof*

EXTENDS *Integers*, *FiniteSets*

$Ballot \triangleq Nat$

CONSTANT *Value*
$None \triangleq$ CHOOSE $v : v \notin Value$

CONSTANTS *Acceptor*,
SafeAcceptor,
FakeAcceptor,
ByzQuorum,
Learner

ASSUME $SafeAccAssumption \triangleq$
$\land SafeAcceptor \cap FakeAcceptor = \{\}$
$\land SafeAcceptor \cup FakeAcceptor = Acceptor$

ASSUME $BQAssumption \triangleq$
$\land \forall Q \in ByzQuorum : Q \subseteq Acceptor$

ASSUME $BallotAssumption \triangleq$
$\land (Ballot \cup \{-1\}) \cap Acceptor = \{\}$
$\land (Ballot \cup \{-1\}) \cap ByzQuorum = \{\}$
$\land (Ballot \cup \{-1\}) \cap Learner = \{\}$

Learner graph

CONSTANT *TrustLive*
ASSUME $TrustLive \in$ SUBSET $[lr : Learner,\ q : ByzQuorum]$

CONSTANT *TrustSafe*
ASSUME $TrustSafe \in$ SUBSET $[from : Learner,\ to : Learner,\ q : ByzQuorum]$

ASSUME $LearnerGraphAssumption \triangleq$
symmetry
$\land \forall E \in TrustSafe :$
$[from \mapsto E.to,\ to \mapsto E.from,\ q \mapsto E.q] \in TrustSafe$
transitivity
$\land \forall E1,\ E2 \in TrustSafe :$
$E1.q = E2.q \Rightarrow$
$[from \mapsto E1.from,\ to \mapsto E2.to,\ q \mapsto E1.q] \in TrustSafe$
closure
$\land \forall E \in TrustSafe : \forall Q \in ByzQuorum :$
$E.q \subseteq Q \Rightarrow$
$[from \mapsto E.from,\ to \mapsto E.to,\ q \mapsto Q] \in TrustSafe$

validity
$\land \forall\, E \in TrustSafe : \forall\, Q1,\ Q2 \in ByzQuorum :$
$([lr \mapsto E.from,\ q \mapsto Q1] \in TrustLive \land [lr \mapsto E.to,\ q \mapsto Q2] \in TrustLive) \Rightarrow$
$\exists\, N \in Acceptor :$
$N \in Q1 \land N \in Q2 \land N \in SafeAcceptor$

CONSTANT Ent
ASSUME $EntanglementAssumption \triangleq$
$\land\ Ent \in \text{SUBSET}\ (Learner \times Learner)$
$\land \forall\, L1,\ L2 \in Learner :$
$\langle L1,\ L2 \rangle \in Ent \equiv$
$[from \mapsto L1,\ to \mapsto L2,\ q \mapsto SafeAcceptor] \in TrustSafe$

$1aMessage \triangleq [type : \{\text{“1a"}\},\ lr : Learner,\ bal : Ballot]$

$1bMessage \triangleq$
[
$type : \{\text{“1b"}\},$
$lr \ : Learner,$
$acc \ : Acceptor,$
$bal \ : Ballot,$
$votes : \text{SUBSET}\ [lr : Learner,\ bal : Ballot,\ val : Value],$
$proposals : \text{SUBSET}\ [bal : Ballot,\ val : Value]$
]

$1cMessage \triangleq$
$[type : \{\text{“1c"}\},\ lr : Learner,\ bal : Ballot,\ val : Value]$

$2avMessage \triangleq$
$[type : \{\text{“2av"}\},\ lr : Learner,\ acc : Acceptor,\ bal : Ballot,\ val : Value]$

$2bMessage \triangleq$
$[type : \{\text{“2b"}\},\ lr : Learner,\ acc : Acceptor,\ bal : Ballot,\ val : Value]$

$BMessage \triangleq$
$1aMessage \cup 1bMessage \cup 1cMessage \cup 2avMessage \cup 2bMessage$

****

--algorithm *HPaxos* **{**

variables
$maxBal \ = [l \in Learner,\ a \in Acceptor \mapsto -1],$
$votesSent = [a \in Acceptor \mapsto \{\}],$
$2avSent = [l \in Learner,\ a \in Acceptor \mapsto \{\}],$
$received = [l \in Learner,\ a \in Acceptor \mapsto \{\}],$
$connected = [a \in Acceptor \mapsto \{\}],$
$receivedByLearner = [l \in Learner \mapsto \{\}],$

$decision = [l \in Learner,\ b \in Ballot \mapsto \{\}]$,
$msgs = \{\}$

define {

$sentMsgs(type,\ lr,\ bal) \triangleq$
$\{m \in msgs : m.type = type \land m.lr = lr \land m.bal = bal\}$

$sentMsgsAnywhere(type,\ bal) \triangleq$
$\{m \in msgs : m.type = type \land m.bal = bal\}$

$initializedBallot(lr,\ bal) \triangleq sentMsgs(\text{“1a”},\ lr,\ bal) \neq \{\}$
$announcements(lr,\ bal) \triangleq sentMsgs(\text{“1c”},\ lr,\ bal)$
$announcedValues(lr,\ bal) \triangleq \{m.val : m \in sentMsgs(\text{“1c”},\ lr,\ bal)\}$

$KnowsSafeAt(l,\ ac,\ b,\ v) \triangleq$
LET $S \triangleq \{m \in received[ac] : m.type = \text{“1b”} \land m.lr = l \land m.bal = b\}$
IN $\lor \exists\, BQ \in ByzQuorum :$
$\land [lr \mapsto l,\ q \mapsto BQ] \in TrustLive$
$\land \forall\, a \in BQ :$
$\exists\, m \in S :$
$\land\ m.acc = a$
$\land \forall\, p \in \{pp \in m.votes : \langle pp.lr,\ l\rangle \in connected[ac]\} :$
$b \leq p.bal$
$\lor \exists\, c \in 0\,..\,(b-1) :$
$\land \exists\, BQ \in ByzQuorum :$
$\land [lr \mapsto l,\ q \mapsto BQ] \in TrustLive$
$\land \forall\, a \in BQ :$
$\exists\, m \in S :$
$\land\ m.acc = a$
$\land \forall\, p \in \{pp \in m.votes : \langle pp.lr,\ l\rangle \in connected[ac]\} :$
$\land\ p.bal \leq c$
$\land\ (p.bal = c) \Rightarrow (p.val = v)$
$\land \exists\, WQ \in ByzQuorum :$
$\land [lr \mapsto l,\ q \mapsto WQ] \in TrustLive$
$\land \forall\, a \in WQ :$
$\exists\, m \in S\ :$
$\land\ m.acc = a$
$\land \exists\, p \in m.proposals :$
$\land\ p.bal = c\ \ TODO$
$\land\ p.val = v$

}

macro $SendMessage(m)\{msgs := msgs \cup \{m\}\}$

macro $Phase1a(l)\{SendMessage([type \mapsto \text{“1a”},\ lr \mapsto l,\ bal \mapsto self])\}$

macro $Phase1b(l,\ b)$

```
{
    when ∧ maxBal[l, self] ≤ b
         ∧ initializedBallot(l, b) ;
    maxBal[l, self] := b ;
    SendMessage(
        [
            type ↦ "1b",
            lr ↦ l,
            acc ↦ self,
            bal ↦ b,
            votes ↦ {p ∈ votesSent[self] : p.bal < b},
            proposals ↦ {p ∈ 2avSent[l, self] : p.bal < b}
        ]
    )
}

macro Phase1c(l)
{
    with (m ∈ [type : {"1c"}, lr : {l}, bal : {self}, val : Value])
    {
        SendMessage(m)
    }
}

macro Phase2av(l, b)
{
    when ∧ maxBal[l, self] ≤ b
         ∧ initializedBallot(l, b) ;
    with (
        v ∈ {va ∈ announcedValues(l, b) :
            ∧ ∀ L ∈ Learner :
              ∀ P ∈ {p ∈ 2avSent[L, self] : p.bal = b} :
                  P.val = va
            ∧ KnowsSafeAt(l, self, b, va)}
    )
    {
        SendMessage(
            [type ↦ "2av", lr ↦ l, acc ↦ self, bal ↦ b, val ↦ v]
        ) ;
        2avSent[l, self] :=
            2avSent[l, self] ∪ {[bal ↦ b, val ↦ v]}
    }
}

macro Phase2b(l, b)
{
```

```
    when ∀ L ∈ Learner : maxBal[L, self] ≤ b ;
    with (
        v ∈ {vv ∈ Value :
                ∃ Q ∈ ByzQuorum :
                    ∧ [lr ↦ l, q ↦ Q] ∈ TrustLive
                    ∧ ∀ aa ∈ Q :
                        ∃ m ∈ {mm ∈ received[l, self] :
                                    ∧ mm.type = "2av"
                                    ∧ mm.bal = b} :
                            ∧ m.val = vv
                            ∧ m.acc = aa}
    )
    {
        SendMessage(
            [type ↦ "2b", lr ↦ l, acc ↦ self, bal ↦ b, val ↦ v]
        ) ;
        votesSent[self] := votesSent[self] ∪ {[lr ↦ l, bal ↦ b, val ↦ v]}
    }
}

macro Receive(l, b)
{
    with (m ∈ sentMsgs("1b", l, b) ∪ sentMsgs("2av", l, b))
    {
        received[l, self] := received[l, self] ∪ {m}
    }
}

macro LearnerReceive(b)
{
    with (m ∈ sentMsgs("2b", self, b))
    {
        receivedByLearner[self] := receivedByLearner[self] ∪ {m}
    }
}

macro FakingAcceptor()
{
    with (m ∈ {mm ∈ 1bMessage ∪ 2avMessage ∪ 2bMessage : mm.acc = self})
    {
        SendMessage(m)
    }
}

macro Decide(b)
{
```

```
    with (
        v ∈ {vv ∈ Value :
                ∃ Q ∈ ByzQuorum :
                    ∧ [lr ↦ self, q ↦ Q] ∈ TrustLive
                    ∧ ∀ aa ∈ Q :
                        ∃ m ∈ {mm ∈ receivedByLearner[self] : mm.bal = b} :
                            ∧ m.val = vv
                            ∧ m.acc = aa}
    )
    {
        decision[self, b] := decision[self, b] ∪ {v}
    }
}

macro LearnDisconnected()
{
    with (P ∈ SUBSET {LL ∈ Learner × Learner : LL ∉ Ent})
    {
        connected[self] := connected[self] \ P
    }
}

process (leader ∈ Ballot)
{
    ldr: while (TRUE)
    {
        with (l ∈ Learner){either Phase1a(l)or Phase1c(l)}
    }
}

process (acceptor ∈ SafeAcceptor)
{
    acc: while (TRUE)
    {
        with (b ∈ Ballot, l ∈ Learner)
        {
            either Phase1b(l, b)
              or Phase2av(l, b)
              or Phase2b(l, b)
              or Receive(l, b)
              or LearnDisconnected()
        }
    }
}

process (facceptor ∈ FakeAcceptor)
```

```
    {
        facc : while (TRUE){FakingAcceptor()}
    }

    process (learner ∈ Learner)
    {
        lrn : while (TRUE){with (b ∈ Ballot){Decide(b)}}
    }
}
```

****

BEGIN TRANSLATION ($chksum(pcal) =$ "$557b65dc$" $\land$ $chksum(tla) =$ "$6efe7a8b$")

VARIABLES $maxBal$, $votesSent$, $2avSent$, $received$, $connected$, $receivedByLearner$, $decision$, $msgs$

define statement

$$sentMsgs(type,\ lr,\ bal) \triangleq \{m \in msgs : m.type = type \land m.lr = lr \land m.bal = bal\}$$

$$sentMsgsAnywhere(type,\ bal) \triangleq \{m \in msgs : m.type = type \land m.bal = bal\}$$

$$initializedBallot(lr,\ bal) \triangleq sentMsgs(\text{“1a”},\ lr,\ bal) \neq \{\}$$

$$announcedValues(lr,\ bal) \triangleq \{m.val : m \in sentMsgs(\text{“1c”},\ lr,\ bal)\}$$

$$KnowsSafeAt(l,\ ac,\ b,\ v) \triangleq$$

```
    LET S ≜ {m ∈ received[ac] : m.type = "1b" ∧ m.lr = l ∧ m.bal = b}
    IN  ∨ ∃ BQ ∈ ByzQuorum :
            ∧ [lr ↦ l, q ↦ BQ] ∈ TrustLive
            ∧ ∀ a ∈ BQ :
                ∃ m ∈ S :
                    ∧ m.acc = a
                    ∧ ∀ p ∈ {pp ∈ m.votes : ⟨pp.lr, l⟩ ∈ connected[ac]} :
                          b ≤ p.bal
        ∨ ∃ c ∈ 0 .. (b − 1) :
            ∧ ∃ BQ ∈ ByzQuorum :
                ∧ [lr ↦ l, q ↦ BQ] ∈ TrustLive
                ∧ ∀ a ∈ BQ :
                    ∃ m ∈ S :
                        ∧ m.acc = a
                        ∧ ∀ p ∈ {pp ∈ m.votes : ⟨pp.lr, l⟩ ∈ connected[ac]} :
                            ∧ p.bal ≤ c
                            ∧ (p.bal = c) ⇒ (p.val = v)
            ∧ ∃ WQ ∈ ByzQuorum :
                ∧ [lr ↦ l, q ↦ WQ] ∈ TrustLive
                ∧ ∀ a ∈ WQ :
```

$$
\begin{array}{l}
\exists\, m \in S \ : \\
\quad \land\ m.acc = a \\
\quad \land\ \exists\, p \in m.proposals : \\
\qquad \land\ p.bal = c \\
\qquad \land\ p.val = v
\end{array}
$$

$vars \triangleq \langle maxBal,\ votesSent,\ 2avSent,\ received,\ connected,\ receivedByLearner,\ decision,\ msgs\rangle$

$ProcSet \triangleq (Ballot) \cup (SafeAcceptor) \cup (FakeAcceptor) \cup (Learner)$

$Init \triangleq$ Global variables

$$
\begin{array}{l}
\land\ maxBal = [l \in Learner,\ a \in Acceptor \mapsto\ -1] \\
\land\ votesSent = [a \in Acceptor \mapsto \{\}] \\
\land\ 2avSent = [l \in Learner,\ a \in Acceptor \mapsto \{\}] \\
\land\ received = [l \in Learner,\ a \in Acceptor \mapsto \{\}] \\
\land\ connected = [a \in Acceptor \mapsto \{\}] \\
\land\ receivedByLearner = [l \in Learner \mapsto \{\}] \\
\land\ decision = [l \in Learner,\ b \in Ballot \mapsto \{\}] \\
\land\ msgs = \{\}
\end{array}
$$

$$
\begin{array}{l}
leader(self) \triangleq\ \land \exists\, l \in Learner : \\
\qquad \lor\ \land\ msgs' = (msgs \cup \{([type \mapsto \text{“1a”},\ lr \mapsto l,\ bal \mapsto self])\}) \\
\qquad \lor\ \land \exists\, m \in [type : \{\text{“1c”}\},\ lr : \{l\},\ bal : \{self\},\ val : Value] : \\
\qquad\qquad msgs' = (msgs \cup \{m\}) \\
\qquad \land \text{UNCHANGED}\ \langle maxBal,\ votesSent,\ 2avSent,\ received, \\
\qquad\qquad\qquad connected,\ receivedByLearner,\ decision\rangle
\end{array}
$$

$$
\begin{array}{l}
acceptor(self) \triangleq\ \land \exists\, b \in Ballot : \\
\qquad \exists\, l \in Learner : \\
\qquad\quad \lor\ \land\ \land\ maxBal[l,\ self] \leq b \\
\qquad\qquad\quad \land\ initializedBallot(l,\ b) \\
\qquad\qquad \land\ maxBal' = [maxBal\ \text{EXCEPT}\ ![l,\ self] = b] \\
\qquad\qquad \land\ msgs' = (msgs \cup \{([\\
\qquad\qquad\qquad\qquad\qquad\qquad type \mapsto \text{“1b”}, \\
\qquad\qquad\qquad\qquad\qquad\qquad lr \mapsto l, \\
\qquad\qquad\qquad\qquad\qquad\qquad acc \mapsto self, \\
\qquad\qquad\qquad\qquad\qquad\qquad bal\ \mapsto b, \\
\qquad\qquad\qquad\qquad\qquad\qquad votes \mapsto \{p \in votesSent[self] : p.bal < b\}, \\
\qquad\qquad\qquad\qquad\qquad\qquad proposals \mapsto \{p \in 2avSent[l,\ self] : p.bal < b\} \\
\qquad\qquad\qquad\qquad\qquad])\}) \\
\qquad\qquad \land\ \text{UNCHANGED}\ \langle votesSent,\ 2avSent,\ received,\ connected\rangle \\
\qquad\quad \lor\ \land\ \land\ maxBal[l,\ self] \leq b \\
\qquad\qquad\quad \land\ initializedBallot(l,\ b) \\
\qquad\qquad \land \exists\, v \in\ \ \{va \in announcedValues(l,\ b) : \\
\qquad\qquad\qquad\qquad \land \forall\, L \in Learner :
\end{array}
$$

$$
\begin{array}{l}
\forall\, P \in \{p \in 2avSent[L,\, self] : p.bal = b\} : \\
\quad P.val = va \\
\quad \wedge\, KnowsSafeAt(l,\, self,\, b,\, va)\} : \\
\quad \wedge\, msgs' = (msgs \cup \{([type \mapsto \text{“2av”},\, lr \mapsto l,\, acc \mapsto self,\, bal \mapsto b,\, val \mapsto v]\} \\
\quad \wedge\, 2avSent' = [2avSent \text{ EXCEPT } ![l,\, self] = 2avSent[l,\, self] \cup \{[bal \mapsto b,\, va} \\
\wedge \text{ UNCHANGED } \langle maxBal,\, votesSent,\, received,\, connected\rangle \\
\vee\ \wedge \forall\, L \in Learner : maxBal[L,\, self] \leq b \\
\quad \wedge \exists\, v \in \{vv \in Value : \\
\qquad \exists\, Q \in ByzQuorum : \\
\qquad\quad \wedge [lr \mapsto l,\, q \mapsto Q] \in TrustLive \\
\qquad\quad \wedge \forall\, aa \in Q : \\
\qquad\qquad \exists\, m \in \{mm \in received[l,\, self] : \\
\qquad\qquad\qquad \wedge\, mm.type = \text{“2av”} \\
\qquad\qquad\qquad \wedge\, mm.bal = b\} : \\
\qquad\qquad\quad \wedge\, m.val = vv \\
\qquad\qquad\quad \wedge\, m.acc = aa\} : \\
\quad \wedge\, msgs' = (msgs \cup \{([type \mapsto \text{“2b”},\, lr \mapsto l,\, acc \mapsto self,\, bal \mapsto b,\, val \mapsto v]\}) \\
\quad \wedge\, votesSent' = [votesSent \text{ EXCEPT } ![self] = votesSent[self] \cup \{[lr \mapsto l,\, bal \\
\quad \wedge \text{ UNCHANGED } \langle maxBal,\, 2avSent,\, received,\, connected\rangle \\
\vee\ \wedge \exists\, m \in sentMsgs(\text{“1b”},\, l,\, b) \cup sentMsgs(\text{“2av”},\, l,\, b) : \\
\quad received' = [received \text{ EXCEPT } ![l,\, self] = received[l,\, self] \cup \{m\}] \\
\quad \wedge \text{ UNCHANGED } \langle maxBal,\, votesSent,\, 2avSent,\, connected,\, msgs\rangle \\
\vee\ \wedge \exists\, P \in \text{SUBSET } \{LL \in Learner \times Learner : LL \notin Ent\} : \\
\quad connected' = [connected \text{ EXCEPT } ![self] = connected[self] \setminus P] \\
\quad \wedge \text{ UNCHANGED } \langle maxBal,\, votesSent,\, 2avSent,\, received,\, msgs\rangle \\
\wedge \text{ UNCHANGED } \langle receivedByLearner,\, decision\rangle
\end{array}
$$

$$
\begin{array}{l}
facceptor(self) \triangleq\ \wedge \exists\, m \in \{mm \in 1bMessage \cup 2avMessage \cup 2bMessage : mm.acc = self\} : \\
\quad msgs' = (msgs \cup \{m\}) \\
\quad \wedge \text{ UNCHANGED } \langle maxBal,\, votesSent,\, 2avSent,\, received, \\
\qquad\qquad connected,\, receivedByLearner,\, decision\rangle
\end{array}
$$

$$
\begin{array}{l}
learner(self) \triangleq\ \wedge \exists\, b \in Ballot : \\
\quad \exists\, v \in \{vv \in Value : \\
\qquad \exists\, Q \in ByzQuorum : \\
\qquad\quad \wedge [lr \mapsto self,\, q \mapsto Q] \in TrustLive \\
\qquad\quad \wedge \forall\, aa \in Q : \\
\qquad\qquad \exists\, m \in \{mm \in receivedByLearner[self] : mm.bal = b\} : \\
\qquad\qquad\quad \wedge\, m.val = vv \\
\qquad\qquad\quad \wedge\, m.acc = aa\} : \\
\quad decision' = [decision \text{ EXCEPT } ![self,\, b] = decision[self,\, b] \cup \{v\}] \\
\wedge \text{ UNCHANGED } \langle maxBal,\, votesSent,\, 2avSent,\, received, \\
\qquad\qquad connected,\, receivedByLearner,\, msgs\rangle
\end{array}
$$

$$
\begin{array}{l}
Next \triangleq\ (\exists\, self \in Ballot : leader(self)) \\
\qquad \vee\, (\exists\, self \in SafeAcceptor : acceptor(self))
\end{array}
$$

$$\lor\ (\exists\, self \in FakeAcceptor : facceptor(self))$$
$$\lor\ (\exists\, self \in Learner : learner(self))$$

$Spec \triangleq Init \land \Box[Next]_{vars}$

END TRANSLATION

$Safety \triangleq$
$\forall\, \langle L1,\, L2\rangle \in Ent : \forall\, B1,\, B2 \in Ballot : \forall\, V1,\, V2 \in Value :$
$V1 \in decision[L1,\, B1] \land V2 \in decision[L2,\, B2] \Rightarrow V1 = V2$

THEOREM $Spec \Rightarrow \Box Safety$