$\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$ MODULE $UniversalPaxosStore$ $\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$

Specification of the consensus protocol in *PaxosStore*.

See [*PaxosStore*@*VLDB*2017]($https : //www.vldb.org/pvldb/vol10/p1730 - lin.pdf$) by *Tencent*.

In this version (adopted from "*PaxosStore.tla*"):

- Client-restricted config (Ballot)
$-$ *Message* types (*i.e.*, "Prepare", "Accept", "*ACK*") are deleted. No state flags (such as "Prepare", "Wait-Prepare", "Accept", "Wait-Accept" are needed.

14 EXTENDS *Integers*, *FiniteSets*

15 $\vdash$

16 $Max(m, n) \triangleq$ IF $m > n$ THEN m ELSE n
17 $Injective(f) \triangleq \forall a, b \in$ DOMAIN $f : (a \neq b) \Rightarrow (f[a] \neq f[b])$

18 $\vdash$

19 CONSTANTS
20    $Participant$,   the set of partipants
21    $Value$ the set of possible input values for *Participant* to propose

23 $None \triangleq$ CHOOSE $b : b \notin Value$
24 $NP \triangleq Cardinality(Participant)$ number of $p \in$ Participants

26 $Quorum \triangleq \{Q \in$ SUBSET $Participant : Cardinality(Q) * 2 = NP + 1\}$
27 ASSUME $QuorumAssumption \triangleq$
28    $\wedge \quad \forall Q \in Quorum : Q \subseteq Participant$
29    $\wedge \quad \forall Q1, Q2 \in Quorum : Q1 \cap Q2 \neq \{\}$

31 $Ballot \triangleq Nat$

33 $PIndex \triangleq$ CHOOSE $f \in [Participant \rightarrow 1 .. NP] : Injective(f)$
34 $Bals(p) \triangleq \{b \in Ballot : b\%NP = PIndex[p] - 1\}$ allocate ballots for each $p \in Participant$

35 $\vdash$

36 $State \triangleq [maxBal : Ballot \cup \{-1\},$
37        $maxVBal : Ballot \cup \{-1\}, maxVVal : Value \cup \{None\}]$

39 $InitState \triangleq [maxBal \mapsto -1, maxVBal \mapsto -1, maxVVal \mapsto None]$

For simplicity, in this specification, we choose to send the complete state of a participant each time. When receiving such a message, the participant processes only the "partial" state it needs.

45 $Message \triangleq [from : Participant, to :$ SUBSET $Participant, state : [Participant \rightarrow State]]$

46 $\vdash$

47 VARIABLES
48    $state$, $state[p][q]$: the state of $q \in Participant$ from the view of $p \in Participant$
49    $msgs$ the set of messages that have been sent

51 $vars \triangleq \langle state, msgs \rangle$

53 $TypeOK \triangleq$
54    $\wedge \quad state \in [Participant \rightarrow [Participant \rightarrow State]]$
55    $\wedge \quad msgs \subseteq Message$

$57 \quad Send(m) \;\triangleq\; msgs' = msgs \cup \{m\}$

$59 \quad Init \;\triangleq$
$60 \qquad \land state = [p \in Participant \mapsto [q \in Participant \mapsto InitState]]$
$61 \qquad \land msgs = \{\}$

$p \in Participant$ starts the prepare phase by issuing a ballot $b \in Ballot$.

$65 \quad Prepare(p,\, b) \;\triangleq$
$66 \qquad \land \quad state[p][p].maxBal < b$
$67 \qquad \land \quad b \in Bals(p)$
$68 \qquad \land \quad state' = [state \text{ EXCEPT } ![p][p].maxBal = b]$
$69 \qquad \land \quad Send([from \mapsto p,\, to \mapsto Participant,\, state \mapsto state'[p]])$

$q \in Participant$ updates its own state $state[q]$ according to the actual state pp of $p \in Participant$ extracted from a message $m \in Message$ it receives. This is called by $OnMessage(q)$.

Note: pp is $m.state[p]$; it may not be equal to $state[p][p]$ at the time $UpdateState$ is called.

$78 \quad UpdateState(q,\, p,\, pp) \;\triangleq$
$79 \qquad state' = [state \text{ EXCEPT}$
$80 \qquad\qquad\qquad ![q][p].maxBal = Max(@,\, pp.maxBal),$
$81 \qquad\qquad\qquad ![q][p].maxVBal = Max(@,\, pp.maxVBal),$
$82 \qquad\qquad\qquad ![q][p].maxVVal = \text{IF } state[q][p].maxVBal < pp.maxVBal$
$83 \qquad\qquad\qquad\qquad\qquad \text{THEN } pp.maxVVal \text{ ELSE } @,$
$84 \qquad\qquad\qquad ![q][q].maxBal = Max(@,\, pp.maxBal),$
$85 \qquad\qquad\qquad ![q][q].maxVBal = \text{IF } state[q][q].maxBal \le pp.maxVBal$
$86 \qquad\qquad\qquad\qquad\qquad \text{THEN } pp.maxVBal \text{ ELSE } @, \quad$ make promise
$87 \qquad\qquad\qquad ![q][q].maxVVal = \text{IF } state[q][q].maxBal \le pp.maxVBal$
$88 \qquad\qquad\qquad\qquad\qquad \text{THEN } pp.maxVVal \text{ ELSE } @] \quad$ accept

$q \in Participant$ receives and processes a message in $Message$.

$92 \quad OnMessage(q) \;\triangleq$
$93 \qquad \exists\, m \in msgs :$
$94 \qquad\quad \land q \in m.to$
$95 \qquad\quad \land \text{LET } p \;\triangleq\; m.from$
$96 \qquad\qquad \text{IN} \quad UpdateState(q,\, p,\, m.state[p])$
$97 \qquad\quad \land \text{IF } \lor m.state[q].maxBal < state'[q][q].maxBal$
$98 \qquad\qquad\qquad \lor m.state[q].maxVBal < state'[q][q].maxVBal$
$99 \qquad\qquad \text{THEN } Send([from \mapsto q,\, to \mapsto \{m.from\},\, state \mapsto state'[q]])$
$100 \qquad\qquad \text{ELSE } \text{UNCHANGED } msgs$

$p \in Participant$ starts the accept phase by issuing the ballot $b \in Ballot$ with value $v \in Value$.

$105 \quad Accept(p,\, b,\, v) \;\triangleq$
$106 \qquad \land b \in Bals(p)$
$107 \qquad \land \exists\, Q \in Quorum : \forall\, q \in Q : state[p][q].maxBal = b$
$108 \qquad \land \lor \forall\, q \in Participant : state[p][q].maxVBal = -1 \quad$ free to pick its own value
$109 \qquad\quad \lor \exists\, q \in Participant : \quad$ v is the value with the highest $maxVBal$
$110 \qquad\qquad \land state[p][q].maxVVal = v$
$111 \qquad\qquad \land \forall\, r \in Participant : state[p][q].maxVBal \ge state[p][r].maxVBal$

```
112          ∧ state′ = [state EXCEPT ![p][p].maxVBal = b,
113                                    ![p][p].maxVVal = v]
114          ∧ Send([from ↦ p, to ↦ Participant, state ↦ state′[p]])
```

```
116  Next  ≜  ∃ p ∈ Participant : ∨ OnMessage(p)
117                                ∨ ∃ b ∈ Ballot : ∨ Prepare(p, b)
118                                                 ∨ ∃ v ∈ Value : Accept(p, b, v)
119  Spec  ≜  Init ∧ □[Next]_vars
```

UniversalPaxosStore satisfies the Consistency property.

```
124  ChosenP(p)  ≜    the set of values chosen by p ∈ Participant
125      {v ∈ Value : ∃ b ∈ Ballot :
126                        ∃ Q ∈ Quorum : ∀ q ∈ Q : ∧ state[p][q].maxVBal = b
127                                                 ∧ state[p][q].maxVVal = v}
128  chosen  ≜  UNION {ChosenP(p) : p ∈ Participant}

130  Consistency  ≜  Cardinality(chosen) ≤ 1
131  THEOREM  Spec ⇒ □Consistency
```