```
──────────── MODULE OneVaulePerBallotPaxosStore ────────────
Specification of the consensus protocol in PaxosStore.

See [PaxosStore@VLDB2017](https://www.vldb.org/pvldb/vol10/p1730-lin.pdf) by Tencent.

In this version (adapted from "UniversalPaxosStore.tla"):

- Adding
EXTENDS Integers, FiniteSets
────────────────────────────────────────────────────────────
Max(m, n) ≜ IF m > n THEN m ELSE n
────────────────────────────────────────────────────────────
CONSTANTS
    Participant,   the set of partipants
    Value          the set of possible input values for Participant to propose

None ≜ CHOOSE b : b ∉ Value

Quorum ≜ {Q ∈ SUBSET Participant : Cardinality(Q) * 2 = Cardinality(Participant) + 1}
ASSUME QuorumAssumption ≜
    ∧ ∀ Q ∈ Quorum : Q ⊆ Participant
    ∧ ∀ Q1, Q2 ∈ Quorum : Q1 ∩ Q2 ≠ {}

Ballot ≜ Nat
────────────────────────────────────────────────────────────
State ≜ [maxBal  : Ballot ∪ {-1},
         maxVBal : Ballot ∪ {-1}, maxVVal : Value ∪ {None}]

InitState ≜ [maxBal ↦ -1, maxVBal ↦ -1, maxVVal ↦ 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.
Message ≜ [from : Participant, to : SUBSET Participant, state : [Participant → State]]
────────────────────────────────────────────────────────────
VARIABLES
    state,   state[p][q]: the state of q ∈ Participant from the view of p ∈ Participant
    bals,    the subset of Ballot consisting of b ∈ Ballot such that Accept(_, b, _)
    msgs     the set of messages that have been sent

vars ≜ ⟨state, bals, msgs⟩

TypeOK ≜
    ∧ state ∈ [Participant → [Participant → State]]
    ∧ bals  ⊆ Ballot
    ∧ msgs ⊆ Message

Send(m) ≜ msgs' = msgs ∪ {m}
────────────────────────────────────────────────────────────
Init ≜
    ∧ state = [p ∈ Participant ↦ [q ∈ Participant ↦ InitState]]
    ∧ bals  = {}
    ∧ msgs = {}
```

p ∈ *Participant* starts the prepare phase by issuing a ballot *b* ∈ *Ballot*.

```
Prepare(p, b) ≜
    ∧  state[p][p].maxBal < b
    ∧  state′ = [state EXCEPT ![p][p].maxBal = b]
    ∧  Send([from ↦ p, to ↦ Participant, state ↦ state′[p]])
    ∧  UNCHANGED bals
```

q ∈ *Participant* updates its own state *state*[*q*] according to the actual state *pp* of *p* ∈ *Participant* extracted from a message *m* ∈ *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.

```
UpdateState(q, p, pp) ≜
    state′ = [state EXCEPT
                ![q][p].maxBal = Max(@, pp.maxBal),
                ![q][p].maxVBal = Max(@, pp.maxVBal),
                ![q][p].maxVVal = IF state[q][p].maxVBal < pp.maxVBal
                                      THEN pp.maxVVal ELSE @,
                ![q][q].maxBal = Max(@, pp.maxBal),
                ![q][q].maxVBal = IF state[q][q].maxBal ≤ pp.maxVBal
                                      THEN pp.maxVBal ELSE @,    make promise
                ![q][q].maxVVal = IF state[q][q].maxBal ≤ pp.maxVBal
                                      THEN pp.maxVVal ELSE @]   accept
```

q ∈ *Participant* receives and processes a message in *Message*.

```
OnMessage(q) ≜
    ∧ ∃ m ∈ msgs :
        ∧ q ∈ m.to
        ∧ LET p ≜ m.from
            IN   UpdateState(q, p, m.state[p])
        ∧ IF  ∨ m.state[q].maxBal < state′[q][q].maxBal
              ∨ m.state[q].maxVBal < state′[q][q].maxVBal
          THEN Send([from ↦ q, to ↦ {m.from}, state ↦ state′[q]])
          ELSE UNCHANGED msgs
    ∧ UNCHANGED bals
```

p ∈ *Participant* starts the accept phase by issuing the ballot *b* ∈ *Ballot* with value *v* ∈ *Value*.

```
Accept(p, b, v) ≜
    ∧ b ∉ bals
    ∧ ∃ Q ∈ Quorum : ∀ q ∈ Q : state[p][q].maxBal = b
    ∧ ∨ ∀ q ∈ Participant : state[p][q].maxVBal = −1  free to pick its own value
      ∨ ∃ q ∈ Participant :  v is the value with the highest maxVBal
            ∧ state[p][q].maxVVal = v
            ∧ ∀ r ∈ Participant : state[p][q].maxVBal ≥ state[p][r].maxVBal
    ∧ state′ = [state EXCEPT ![p][p].maxVBal = b,
                             ![p][p].maxVVal = v]
    ∧ Send([from ↦ p, to ↦ Participant, state ↦ state′[p]])
```

```
      $\land bals' = bals \cup \{b\}$
────────────────────────────────────────────────
$Next \triangleq \exists p \in Participant : \lor OnMessage(p)$
                                   $\lor \exists b \in Ballot : \lor Prepare(p, b)$
                                                  $\lor \exists v \in Value : Accept(p, b, v)$
$Spec \triangleq Init \land \Box[Next]_{vars}$
────────────────────────────────────────────────
$ChosenP(p) \triangleq$   the set of values chosen by $p \in Participant$
   $\{v \in Value : \exists b \in Ballot :$
                  $\exists Q \in Quorum : \forall q \in Q : \land state[p][q].maxVBal = b$
                                              $\land state[p][q].maxVVal = v\}$
$chosen \triangleq$ UNION $\{ChosenP(p) : p \in Participant\}$

$Consistency \triangleq Cardinality(chosen) \leq 1$
THEOREM $Spec \Rightarrow \Box Consistency$
────────────────────────────────────────────────
```

\ ∗ Modification History

\ ∗ Last modified *Mon Aug* 05 15:11:26 *CST* 2019 by *hengxin*

\ ∗ Created *Mon Aug* 05 15:01:11 *CST* 2019 by *hengxin*