```
1  ---------- MODULE PaxosStore ----------
2  EXTENDS Integers, FiniteSets, TLC
3  ----------
4  Max(m, n) ≜ IF m > n THEN m ELSE n
5  Injective(f) ≜ ∀ a, b ∈ DOMAIN f : (a ≠ b) ⇒ (f[a] ≠ f[b])
6  ----------
7  CONSTANTS
8      Value,        the set of values
9      Participant   the set of partipants

11 None ≜ CHOOSE b : b ∉ Value

13 NP ≜ Cardinality(Participant)   number of p ∈ Participants
14 Quorum ≜ {Q ∈ SUBSET Participant : Cardinality(Q) * 2 = NP + 1}

16 ASSUME QuorumAssumption ≜
17     ∧  ∀ Q ∈ Quorum : Q ⊆ Participant
18     ∧  ∀ Q1, Q2 ∈ Quorum : Q1 ∩ Q2 ≠ {}

20 Ballot ≜ Nat

22 PIndex ≜ CHOOSE f ∈ [Participant → 1 .. NP] : Injective(f)   TODO: (1) symmetry set? (2) model
23 Bals(p) ≜ {b ∈ Ballot : b%NP = PIndex[p] − 1}   allocate ballots for each p ∈ Participant
24 ----------
25 State ≜ [maxBal : Ballot ∪ {−1},
26            maxVBal : Ballot ∪ {−1}, maxVVal : Value ∪ {None}]

28 InitState ≜ [maxBal ↦ −1, maxVBal ↦ −1, maxVVal ↦ None]

30 Message ≜ [type : {"Prepare", "Accept", "ACK"},
31               from : Participant, to : SUBSET Participant,
32               state : [Participant → State]]
33 ----------
34 VARIABLES
35     state,    state[p][q]: the state of q ∈ Participant from the view of p ∈ Participant
36     msgs      set of messages
37 vars ≜ ⟨state, msgs⟩

39 Send(m) ≜ msgs′ = msgs ∪ {m}

41 TypeOK ≜
42     ∧  state ∈ [Participant → [Participant → State]]
43     ∧  msgs ⊆ Message
44 ----------
45 Init ≜
46     ∧ state = [p ∈ Participant ↦ [q ∈ Participant ↦ InitState]]
47     ∧ msgs = {}
```

```
49  Prepare(p, b) ≜
50      ∧  state[p][p].maxBal < b
51      ∧  b ∈ Bals(p)
52      ∧  state' = [state EXCEPT ![p][p].maxBal = b]
53      ∧  Send([type ↦ "Prepare", from ↦ p, to ↦ Participant, state ↦ state'[p]])

55  Accept(p, b, v) ≜
56      TODO: delete it? to allow duplication?
57      ∧ ¬∃ m ∈ msgs : m.type = "Accept" ∧ m.state[p].maxBal = b
58      ∧ b ∈ Bals(p)      TODO: delete it?
59      ∧ ∃ Q ∈ Quorum : ∀ q ∈ Q : state[p][q].maxBal = b  TODO: majority quorum (local ack?)
60      ∧ ∨ ∀ q ∈ Participant : state[p][q].maxVBal = −1  free to pick its own value
61        ∨ ∃ q ∈ Participant :  v is the value with the highest maxVBal
62              ∧ state[p][q].maxVVal = v
63              ∧ ∀ r ∈ Participant : state[p][q].maxVBal ≥ state[p][r].maxVBal
64      ∧ state' = [state EXCEPT ![p][p].maxVBal = b,
65                               ![p][p].maxVVal = v]
66      ∧ Send([type ↦ "Accept", from ↦ p, to ↦ Participant, state ↦ state'[p]])

    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.

76  UpdateState(q, p, pp) ≜
77      state' = [state EXCEPT
78                  ![q][p].maxBal = Max(@, pp.maxBal),
79                  ![q][p].maxVBal = Max(@, pp.maxVBal),
80                  ![q][p].maxVVal = IF state[q][p].maxVBal < pp.maxVBal
81                                    THEN pp.maxVVal ELSE @,
82                  ![q][q].maxBal = Max(@, pp.maxBal),
83                  ![q][q].maxVBal = IF state[q][q].maxBal ≤ pp.maxVBal
84                                    THEN pp.maxVBal ELSE @,    make promise
85                  ![q][q].maxVVal = IF state[q][q].maxBal ≤ pp.maxVBal
86                                    THEN pp.maxVVal ELSE @]   accept

88  OnMessage(q) ≜
89      ∃ m ∈ msgs :
90          ∧ m.type = "ACK" ⇒ m.to = {q}
91          ∧ LET p ≜ m.from
92            IN  UpdateState(q, p, m.state[p])
93          ∧ IF  ∨ m.state[q].maxBal < state'[q][q].maxBal   TODO: delete "if"?
94                ∨ m.state[q].maxVBal < state'[q][q].maxVBal
95            THEN Send([type ↦ "ACK", from ↦ q, to ↦ {m.from}, state ↦ state'[q]])
96            ELSE UNCHANGED msgs
97  ──────────────────────────────────────────────────────────────────────────
98  Next ≜ ∃ p ∈ Participant : ∨ OnMessage(p)
```

$$\lor \exists\, b \in Ballot : \lor Prepare(p,\, b)$$
$$\lor \exists\, v \in Value : Accept(p,\, b,\, v)$$

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

$ChosenP(p) \triangleq$

$\{v \in Value : \exists\, b \in Ballot :$
$\exists\, Q \in Quorum : \forall\, q \in Q : \land state[p][q].maxVBal = b$
$\land state[p][q].maxVVal = v\}$

$chosen \triangleq \text{UNION } \{ChosenP(p) : p \in Participant\}$

$Consistency \triangleq Cardinality(chosen) \leq 1$

THEOREM $Spec \Rightarrow \Box Consistency$

---

* Modification History
* Last modified *Mon Jul* 29 18:19:40 *CST* 2019 by *hengxin*
* Last modified *Mon Jul* 22 13:59:15 *CST* 2019 by *pure_*
* Last modified *Mon Jun* 03 21:26:09 *CST* 2019 by *stary*
* Last modified *Wed* May 09 21:39:31 *CST* 2018 by dell
* Created *Mon Apr* 23 15:47:52 *GMT* + 08:00 2018 by *pure_*