

```

1  |----- MODULE PaxosStore -----|
   | Specification of the consensus protocol in PaxosStore.
   | See [PaxosStore@VLDB2017](https://www.vldb.org/pvldb/vol10/p1730-lin.pdf) by Tencent.
   | In this version:
   | - Client-restricted config (Ballot)
   |   - Message types: "Prepare", "Accept", "ACK"
13 | EXTENDS Integers, FiniteSets
14 |-----|
15 |  $Max(m, n) \triangleq \text{IF } m > n \text{ THEN } m \text{ ELSE } n$ 
16 |  $Injective(f) \triangleq \forall a, b \in \text{DOMAIN } f : (a \neq b) \Rightarrow (f[a] \neq f[b])$ 
17 |-----|
18 | CONSTANTS
19 |   Participant, the set of participants
20 |   Value the set of possible input values for Participant to propose
22 |  $None \triangleq \text{CHOOSE } b : b \notin \text{Value}$ 
23 |  $NP \triangleq \text{Cardinality}(\text{Participant})$  number of  $p \in \text{Participants}$ 
25 |  $Quorum \triangleq \{Q \in \text{SUBSET Participant} : \text{Cardinality}(Q) * 2 \geq NP + 1\}$ 
26 | ASSUME QuorumAssumption  $\triangleq$ 
27 |    $\wedge \forall Q \in Quorum : Q \subseteq Participant$ 
28 |    $\wedge \forall Q1, Q2 \in Quorum : Q1 \cap Q2 \neq \{\}$ 
30 |  $Ballot \triangleq Nat$ 
32 |  $PIndex \triangleq \text{CHOOSE } f \in [Participant \rightarrow 1 \dots NP] : Injective(f)$  TODO: (1) symmetry set? (2) model
33 |  $Bals(p) \triangleq \{b \in Ballot : b \% NP = PIndex[p] - 1\}$  allocate ballots for  $p \in Participant$ 
34 |-----|
35 |  $State \triangleq [maxBal : Ballot \cup \{-1\},$ 
36 |    $maxVBal : Ballot \cup \{-1\}, maxVVal : Value \cup \{None\}]$ 
38 |  $InitState \triangleq [maxBal \mapsto -1, maxVBal \mapsto -1, maxVVal \mapsto None]$ 
   | 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.
44 |  $Message \triangleq [type : \{\text{"Prepare"}, \text{"Accept"}, \text{"ACK"}\},$ 
45 |    $from : Participant, to : \text{SUBSET Participant},$  TODO: remove "to"
46 |    $state : [Participant \rightarrow State]]$ 
47 |-----|
48 | VARIABLES
49 |   state, state[p][q]: the state of  $q \in Participant$  from the view of  $p \in Participant$ 
50 |   msgs the set of messages that have been sent
52 |  $vars \triangleq \langle state, msgs \rangle$ 
54 |  $TypeOK \triangleq$ 
55 |    $\wedge state \in [Participant \rightarrow [Participant \rightarrow State]]$ 

```

```

56    $\wedge \text{msgs} \subseteq \text{Message}$ 
58    $\text{Send}(m) \triangleq \text{msgs}' = \text{msgs} \cup \{m\}$ 
59   

---


60    $\text{Init} \triangleq$ 
61      $\wedge \text{state} = [p \in \text{Participant} \mapsto [q \in \text{Participant} \mapsto \text{InitState}]]$ 
62      $\wedge \text{msgs} = \{\}$ 
63      $p \in \text{Participant}$  starts the prepare phase by issuing a ballot  $b \in \text{Ballot}$ .
64    $\text{Prepare}(p, b) \triangleq$ 
65      $\wedge \text{state}[p][p].\text{maxBal} < b$ 
66      $\wedge b \in \text{Bals}(p)$ 
67      $\wedge \text{state}' = [\text{state} \text{ EXCEPT } ![p][p].\text{maxBal} = b]$ 
68      $\wedge \text{Send}([type \mapsto \text{"Prepare"}, from \mapsto p, to \mapsto \text{Participant}, state \mapsto \text{state}'[p]])$ 
69      $q \in \text{Participant}$  updates its own state  $\text{state}[q]$  according to the actual state  $pp$  of  $p \in \text{Participant}$ 
70     extracted from a message  $m \in \text{Message}$  it receives. This is called by  $\text{OnMessage}(q)$ .
71   Note:  $pp$  is  $m.\text{state}[p]$ ; it may not be equal to  $\text{state}[p][p]$  at the time  $\text{UpdateState}$  is called.
72    $\text{UpdateState}(q, p, pp) \triangleq$ 
73      $\text{state}' = [\text{state} \text{ EXCEPT}$ 
74        $![q][p].\text{maxBal} = \text{Max}(@, pp.\text{maxBal}),$ 
75        $![q][p].\text{maxVbal} = \text{Max}(@, pp.\text{maxVbal}),$ 
76        $![q][p].\text{maxVval} = \text{IF } \text{state}[q][p].\text{maxVbal} < pp.\text{maxVbal}$ 
77         THEN  $pp.\text{maxVval}$  ELSE  $@,$ 
78        $![q][q].\text{maxBal} = \text{Max}(@, pp.\text{maxBal}),$ 
79        $![q][q].\text{maxVbal} = \text{IF } \text{state}[q][q].\text{maxBal} \leq pp.\text{maxVbal}$ 
80         THEN  $pp.\text{maxVbal}$  ELSE  $@,$  make promise
81        $![q][q].\text{maxVval} = \text{IF } \text{state}[q][q].\text{maxBal} \leq pp.\text{maxVbal}$  TODO: write-once?
82         THEN  $pp.\text{maxVval}$  ELSE  $@]$  accept
83    $q \in \text{Participant}$  receives and processes a message in  $\text{Message}$ .
84    $\text{OnMessage}(q) \triangleq$ 
85      $\exists m \in \text{msgs} :$ 
86        $\wedge m.type = \text{"ACK"} \Rightarrow m.to = \{q\}$ 
87        $\wedge \text{LET } p \triangleq m.from$ 
88       IN  $\text{UpdateState}(q, p, m.state[p])$ 
89        $\wedge \text{IF } \vee m.state[q].\text{maxBal} < \text{state}'[q][q].\text{maxBal}$  TODO: delete "if"?
90          $\vee m.state[q].\text{maxVbal} < \text{state}'[q][q].\text{maxVbal}$ 
91       THEN  $\text{Send}([type \mapsto \text{"ACK"}, from \mapsto q, to \mapsto \{m.from\}, state \mapsto \text{state}'[q]])$ 
92       ELSE UNCHANGED  $\text{msgs}$ 
93    $p \in \text{Participant}$  starts the accept phase by issuing the ballot  $b \in \text{Ballot}$  with value  $v \in \text{Value}$ .
94    $\text{Accept}(p, b, v) \triangleq$ 
95      $\wedge \neg \exists m \in \text{msgs} : \text{TODO: delete it? to allow repeating Phase2a?}$ 
96      $m.type = \text{"Accept"} \wedge m.state[p].\text{maxBal} = b$ 
97      $\wedge b \in \text{Bals}(p)$  TODO: delete it? to break "client-restricted config"?
98      $\wedge \exists Q \in \text{Quorum} : \forall q \in Q : \text{state}[p][q].\text{maxBal} = b$ 
99      $\wedge \forall q \in \text{Participant} : \text{state}[p][q].\text{maxVbal} = -1$  free to pick its own value

```

