MODULE *CassandraPaxos*

This is a high-level specification of the *CassandraPaxos* algorithm used by *Cassandra* to implement *LWT* (lightweight transactions).

This spec is adapted from that of *Paxos* consensus algorithm by *Leslie Lamport*, which can be found at https://*github.com*/tlaplus/Examples/blob/master/specifications/*PaxosHowToWinATuringAward*/*Paxos.tla*.

EXTENDS *Integers*, *Sequences*, *FiniteSets*

$Max(S) \triangleq$
  IF $S = \{\}$ THEN -1
  ELSE CHOOSE $n \in S : \forall\, m \in S : n \geq m$

CONSTANTS
  Value, the set of values to be proposed and chosen from
  Acceptor, the set acceptors
  Quorum, the quorum system on acceptors
  Operator *TODO*

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

ASSUME $\land \forall\, Q \in Quorum : Q \subseteq Acceptor$
  $\land \forall\, Q1,\, Q2 \in Quorum : Q1 \cap Q2 \neq \{\}$

$Ballot \triangleq Nat$
$Version \triangleq Nat$ *TODO*

$\langle + \rangle$ The set of all possible *CAS* operations. The *CAS* operations with $cmpVal = None$ are initialization operations. We assume that the new values (*i.e.*, $swapVal$) are not *None*.

$CASOperation \triangleq [cmpVal : Value \cup \{None\},\, swapVal : Value]$

$MeetCondition(ev,\, val) \triangleq \lor\, val = None$
  $\lor$ CASE $Operator = \text{“>”} \to val > ev$
    $\square$ $Operator = \text{“<”} \to val < ev$
    $\square$ $Operator = \text{“=”} \to val = ev$
    $\square$ $Operator = \text{“>=”} \to val \geq ev$
    $\square$ $Operator = \text{“<=”} \to val \leq ev$
    $\square$ $Operator = \text{“/=”} \to val \neq ev$
    $\square$ OTHER $\to$ FALSE

$Message \triangleq$
    $[type : \{\text{“Prepare”}\},\, bal : Ballot]$
  $\cup$ $[type : \{\text{“Promise”}\},\, acc : Acceptor,\, bal : Ballot,$
    $maxAccBal : Ballot \cup \{-1\},\, maxAccVal : Value \cup \{None\},$
    $maxComBal : Ballot \cup \{-1\},\, maxComVal : Value \cup \{None\}]$
  $\cup$ $[type : \{\text{“Read”}\},\, bal : Ballot]$
  $\cup$ $[type : \{\text{“Result”}\},\, acc : Acceptor,\, bal : Ballot,$
    $value : Value \cup \{None\},\, version : Version]$

```
    ∪  [type : {"Repair"}, value : Value ∪ {None}, version : Version]
    ∪  [type : {"Propose"}, bal : Ballot, val : Value]
    ∪  [type : {"Accept"}, acc : Acceptor, bal : Ballot, val : Value]
    ∪  [type : {"Commit"}, bal : Ballot, val : Value]
    ∪  [type : {"Ack"}, acc : Acceptor, bal : Ballot, val : Value]
    ∪  [type : {"Terminate"}, bal : Ballot]  ⟨+⟩ the messages sent to the user
```

```
VARIABLES
    maxBal,
    maxAccBal,
    maxAccVal,
    maxComBal,
    maxComVal,
    msgs,
    dataResult,
    balValue,     TODO: remove it?
    ops           ⟨+⟩ops[b]: the CAS operation to be proposed at ballot b ∈ Ballot

vars ≜ ⟨maxBal, maxAccBal, maxAccVal, maxComBal,
        maxComVal, msgs, dataResult, balValue, ops⟩
```

```
TypeOK ≜
    ∧  maxBal ∈ [Acceptor → Ballot ∪ {−1}]
    ∧  maxAccBal ∈ [Acceptor → Ballot ∪ {−1}]
    ∧  maxAccVal ∈ [Acceptor → Value ∪ {None}]
    ∧  maxComBal ∈ [Acceptor → Ballot ∪ {−1}]
    ∧  maxComVal ∈ [Acceptor → Value ∪ {None}]
    ∧  msgs ⊆ Message
    ∧  dataResult ∈ [Acceptor → [value : Value ∪ {None}, version : Version]]
    ∧  balValue ∈ [Ballot → [cmpVal : Value ∪ {None}, swapVal : Value ∪ {None}]]
    ∧  ops ∈ [Ballot → CASOperation]  ⟨+⟩
```

```
Init ≜  ∧ maxBal = [a ∈ Acceptor ↦ −1]
        ∧ maxAccBal = [a ∈ Acceptor ↦ −1]
        ∧ maxAccVal = [a ∈ Acceptor ↦ None]
        ∧ maxComBal = [a ∈ Acceptor ↦ −1]
        ∧ maxComVal = [a ∈ Acceptor ↦ None]
        ∧ msgs = {}
        ∧ dataResult = [a ∈ Acceptor ↦ [value ↦ None, version ↦ 0]]
        ∧ balValue = [b ∈ Ballot ↦ [cmpVal ↦ None, swapVal ↦ None]]
         ⟨+⟩ops remains unchanged; we utilize TLC to explore all possible CAS operations.
        ∧ ops ∈ [Ballot → CASOperation]
```

```
Send(m) ≜ msgs′ = msgs ∪ {m}
```

The leader of ballot $b \in Ballot$ sends a "Prepare" message.

$$
\begin{array}{l}
Prepare(b) \triangleq \\
\quad \land \quad Send([type \mapsto \text{"Prepare"},\ bal \mapsto b]) \\
\quad \land \quad \text{UNCHANGED } \langle maxBal,\ maxAccBal,\ maxAccVal,\ maxComBal,\ maxComVal,\ balValue,\ dataResult,\ ops\rangle
\end{array}
$$

The acceptor $a \in Acceptor$ receives a "Prepare" message and sends back a "Promise" message.

$$
\begin{array}{l}
Promise(a) \triangleq \\
\quad \land \exists\, m \in msgs : \\
\qquad \land\ m.type = \text{"Prepare"} \\
\qquad \land\ m.bal > maxBal[a] \\
\qquad \land\ maxBal' = [maxBal \text{ EXCEPT } ![a] = m.bal] \\
\qquad \land\ Send([type \mapsto \text{"Promise"},\ acc \mapsto a,\ bal \mapsto m.bal, \\
\qquad\qquad maxAccBal \mapsto maxAccBal[a],\ maxAccVal \mapsto maxAccVal[a], \\
\qquad\qquad maxComBal \mapsto maxComBal[a],\ maxComVal \mapsto maxComVal[a]]) \\
\quad \land \text{UNCHANGED } \langle maxAccBal,\ maxAccVal,\ maxComBal,\ maxComVal, \\
\qquad\qquad dataResult,\ balValue,\ ops\rangle
\end{array}
$$

$$
\begin{array}{l}
Propose(b) \triangleq \\
\quad \land \quad \neg\exists\, m \in msgs : m.type = \text{"Propose"} \land m.bal = b \\
\quad \land \quad \exists\, Q \in Quorum : \\
\qquad \text{LET } Qmset \triangleq \{m \in msgs :\ \land\ m.type = \text{"Promise"} \\
\qquad\qquad\qquad\qquad\qquad\qquad \land\ m.acc \in Q \\
\qquad\qquad\qquad\qquad\qquad\qquad \land\ m.bal = b\} \\
\qquad\qquad maxAccbal \triangleq Max(\{m.maxAccBal : m \in Qmset\}) \\
\qquad\qquad maxCombal \triangleq Max(\{m.maxComBal : m \in Qmset\}) \\
\qquad\qquad preValue \triangleq (\text{CHOOSE } m \in Qmset : m.maxAccBal = maxAccbal).maxAccVal \\
\qquad \text{IN} \quad \land\ \forall\, a \in Q : \exists\, m \in Qmset : m.acc = a \\
\qquad\qquad\quad \land \text{IF } maxAccbal > maxCombal \text{ THEN } Send([type \mapsto \text{"Propose"}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad bal \mapsto b,\ val \mapsto preValue]) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{ELSE } Send([type \mapsto \text{"Read"},\ bal \mapsto b]) \\
\quad \land \quad \text{UNCHANGED } \langle maxBal,\ maxAccBal,\ maxAccVal,\ maxComBal,\ maxComVal, \\
\qquad\qquad\qquad dataResult,\ balValue,\ ops\rangle
\end{array}
$$

$$
\begin{array}{l}
Read(a) \triangleq \ \land \exists\, m \in msgs :\ \land\ m.type = \text{"Read"} \\
\qquad\qquad\qquad\qquad\qquad\qquad \land\ Send([type \mapsto \text{"Result"},\ acc \mapsto a,\ bal \mapsto m.bal, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad value \mapsto dataResult[a].value, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad version \mapsto dataResult[a].version]) \\
\qquad\qquad\qquad \land \text{UNCHANGED } \langle maxBal,\ maxAccBal,\ maxAccVal,\ maxComBal,\ maxComVal, \\
\qquad\qquad\qquad\qquad\qquad dataResult,\ balValue\rangle
\end{array}
$$

$$
\begin{array}{l}
Result(b) \quad \triangleq \ \land \exists\, Q \in Quorum : \\
\qquad\qquad\qquad \text{LET } QRmset \triangleq \{m \in msgs :\ \land\ m.type = \text{"Result"} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \land\ m.acc \in Q \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \land\ m.bal = b\} \\
\qquad\qquad\qquad\qquad QResult \triangleq \{m.value : m \in QRmset\}
\end{array}
$$

```
                    maxVersion ≜ Max({m.version : m ∈ QRmset})
                    maxValue ≜ (CHOOSE m ∈ QRmset : m.version = maxVersion).value
              IN    ∧ ∀ a ∈ Q : ∃ m ∈ QRmset : m.acc = a
                    ∧ IF MeetCondition(balValue[b].cmpVal, maxValue)
                         THEN Send([type ↦ "Propose", bal ↦ b,
                                    val ↦ balValue[b].swapVal])
                         ELSE Send([type ↦ "Terminate", bal ↦ b])
                    ∧ UNCHANGED ⟨maxBal, maxAccBal, maxAccVal, maxComBal, maxComVal,
                                 dataResult, balValue⟩

Accept(a) ≜ ∧ ∃ m ∈ msgs : ∧ m.type = "Propose"
                           ∧ maxBal[a] = m.bal
                           ∧ maxBal[a] ≤ m.bal
                           ∧ maxBal' = [maxBal EXCEPT ![a] = m.bal]
                           ∧ maxAccBal' = [maxAccBal EXCEPT ![a] = m.bal]
                           ∧ maxAccVal' = [maxAccVal EXCEPT ![a] = m.val]
                           ∧ Send([type ↦ "Accept", bal ↦ m.bal,
                                   val ↦ m.val, acc ↦ a])
            ∧ UNCHANGED ⟨maxComBal, maxComVal, dataResult, balValue⟩
          ∧ UNCHANGED ⟨maxBal, maxComBal, maxComVal, dataResult, balValue⟩

Commit(b) ≜ ∧ ¬∃ m ∈ msgs : m.type = "Commit" ∧ m.bal = b
            ∧ ∃ Q ∈ Quorum :
                LET QAmset ≜ {m ∈ msgs : ∧ m.type = "Accept"
                                         ∧ m.acc ∈ Q
                                         ∧ m.bal = b}
                IN  ∧ ∀ a ∈ Q : ∃ m ∈ QAmset : m.acc = a
            ∧ Send([type ↦ "Commit", bal ↦ b,
                    val ↦ balValue[b].swapVal])
            ∧ UNCHANGED ⟨maxBal, maxAccBal, maxAccVal, maxComBal,
                         maxComVal, dataResult, balValue⟩

Ack(a) ≜ ∧ ∃ m ∈ msgs : ∧ m.type = "Commit"
                        ∧ maxBal[a] ≤ m.bal
                        ∧ maxBal' = [maxBal EXCEPT ![a] = m.bal]
                        ∧ maxBal[a] = m.bal
                        ∧ maxComBal' = [maxComBal EXCEPT ![a] = m.bal]
                        ∧ maxComVal' = [maxComVal EXCEPT ![a] = m.val]
                        ∧ dataResult' = [dataResult EXCEPT ![a] =
                            [value ↦ m.val, version ↦ (@.version + 1)]]
                        ∧ Send([type ↦ "Ack", bal ↦ m.bal,
                                val ↦ m.val, acc ↦ a])
         ∧ UNCHANGED ⟨maxAccBal, maxAccVal, balValue⟩
         ∧ UNCHANGED ⟨maxAccBal, maxAccVal, dataResult, balValue⟩
       ∧ UNCHANGED ⟨maxBal, maxAccBal, maxAccVal, dataResult, balValue⟩
```

$Next \triangleq$

$\quad \lor \exists\, ev, sv \in Value, b \in Ballot : Prepare(b)$

$\quad \lor \exists\, b \in Ballot : \lor Propose(b)$

$\qquad\qquad\qquad\qquad\quad \lor Result(b)$

$\qquad\qquad\qquad\qquad\quad \lor Commit(b)$

$\quad \lor \exists\, a \in Acceptor : \lor Promise(a)$

$\qquad\qquad\qquad\qquad\qquad \lor Accept(a)$

$\qquad\qquad\qquad\qquad\qquad \lor Ack(a)$

$\qquad\qquad\qquad\qquad\qquad \lor Read(a)$

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

* Modification History

* Last modified Sat *Aug* 20 13:41:52 *CST* 2022 by *hengxin*

* Last modified *Thu Mar* 03 17:37:19 *CST* 2022 by *LENOVO*

* Created *Thu Dec* 08 10:19:29 *CST* 2021 by *LENOVO*