MODULE *ScyllaPaxos*

EXTENDS *Integers*

$Maximum(S) \triangleq$

If S is a set of numbers, then this define $Maximum(S)$ to be the maximum of those numbers, or -1 if S is empty.

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

$Same(S) \triangleq$

If S is not empty, then this define $Same(S)$ to be the if or not the element of S is same. It is designed for determine whether the results of Read are the same

$\land\ S \neq \{\}$
$\land\ \exists\, n \in S : \forall\, m \in S : n = m$

CONSTANTS *Value*, *Acceptor*, *Quorum*, *Operator*

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

$Ballot \triangleq Nat$
$Version \triangleq Nat$
$None \triangleq$ CHOOSE $v : v \notin Value$

$MeetCondition(ev,\ val) \triangleq \lor\ val = None$
$\lor$ 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

$Message \triangleq$
$[type : \{\text{“Prepare”}\},\ bal : Ballot]$
$\cup\ [type : \{\text{“Promise”}\},\ acc : Acceptor,\ bal : Ballot,$
$maxAccBal : Ballot \cup \{-1\},\ maxAccVal : Value \cup \{None\},$
$maxComBal : Ballot \cup \{-1\},\ maxComVal : Value \cup \{None\},$
$value : Value,\ version : Version]$
$\cup\ [type : \{\text{“Repair”}\},\ value : Value \cup \{None\},\ version : Version]$
$\cup\ [type : \{\text{“Propose”}\},\ bal : Ballot,\ val : Value]$
$\cup\ [type : \{\text{“Accept”}\},\ acc : Acceptor,\ bal : Ballot,\ val : Value]$
$\cup\ [type : \{\text{“Learn”}\},\ bal : Ballot,\ val : Value]$
$\cup\ [type : \{\text{“Ack”}\},\ acc : Acceptor,\ bal : Ballot,\ val : Value]$
$\cup\ [type : \{\text{“Terminate”}\},\ bal : Ballot]$

VARIABLES *maxBal*, *maxAccBal*, *maxAccVal*, *maxComBal*,
maxComVal, *msgs*, *dataResult*, *balValue*

$$vars \triangleq \langle maxBal,\ maxAccBal,\ maxAccVal,\ maxComBal,\ maxComVal,\ msgs,\ dataResult,\ balValue \rangle$$

$$\begin{aligned}
TypeOK \triangleq\ & \land maxBal \in [Acceptor \to Ballot \cup \{-1\}] \\
& \land maxAccBal \in [Acceptor \to Ballot \cup \{-1\}] \\
& \land maxAccVal \in [Acceptor \to Value \cup \{None\}] \\
& \land maxComBal \in [Acceptor \to Ballot \cup \{-1\}] \\
& \land maxComVal \in [Acceptor \to Value \cup \{None\}] \\
& \land dataResult \in [Acceptor \to [value : Value \cup \{None\}, \\
& \qquad\qquad version : Version]] \\
& \land msgs \subseteq Message \\
& \land balValue \in [Ballot \to [expVal : Value \cup \{None\}, \\
& \qquad\qquad setVal : Value \cup \{None\}]]
\end{aligned}$$

$$\begin{aligned}
Init \triangleq\ & \land maxBal = [a \in Acceptor \mapsto -1] \\
& \land maxAccBal = [a \in Acceptor \mapsto -1] \\
& \land maxAccVal = [a \in Acceptor \mapsto None] \\
& \land maxComBal = [a \in Acceptor \mapsto -1] \\
& \land maxComVal = [a \in Acceptor \mapsto None] \\
& \land dataResult = [a \in Acceptor \mapsto [value \mapsto None,\ version \mapsto 0]] \\
& \land msgs = \{\} \\
& \land balValue = [b \in Ballot \mapsto [expVal \mapsto None,\ setVal \mapsto None]]
\end{aligned}$$

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

$$\begin{aligned}
CAS(ev,\ sv,\ b) \triangleq\ & \land \neg\exists\, m \in msgs : m.type = \text{“Prepare”} \land m.bal = b \\
& \land Send([type \mapsto \text{“Prepare”},\ bal \mapsto b]) \\
& \land balValue' = [balValue \text{ EXCEPT } ![b] = \\
& \qquad\qquad [expVal \mapsto ev,\ setVal \mapsto sv]] \\
& \land \text{UNCHANGED } \langle maxBal,\ maxAccBal,\ maxAccVal,\ maxComBal, \\
& \qquad maxComVal,\ dataResult \rangle
\end{aligned}$$

Promise *Message* add *ReadResult*(*value*, *version*)

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

```
Propose(b) ≜  ∧ ¬∃ m ∈ msgs : m.type = "Propose" ∧ m.bal = b
              ∧ ∃ Q ∈ Quorum :
                 LET Qmset ≜ {m ∈ msgs : ∧ m.type = "Promise"
                                         ∧ m.acc ∈ Q
                                         ∧ m.bal = b}
                     maxAccbal ≜ Maximum({m.maxAccBal : m ∈ Qmset})
                     maxCombal ≜ Maximum({m.maxComBal : m ∈ Qmset})
                     preValue ≜ (CHOOSE m ∈ Qmset : m.maxAccBal = maxAccbal).maxAccVal
                     QResult ≜ {m.value : m ∈ Qmset}
                     maxVersion ≜ Maximum({m.version : m ∈ Qmset})
                     maxValue ≜ (CHOOSE m ∈ Qmset : m.version = maxVersion).value
                 IN  ∧ ∀ a ∈ Q : ∃ m ∈ Qmset : m.acc = a
                     ∧ IF maxAccbal > maxCombal
                       THEN Send([type ↦ "Propose", bal ↦ b, val ↦ preValue])
                       ELSE IF MeetCondition(balValue[b].expVal, maxValue)
                            THEN Send([type ↦ "Propose", bal ↦ b,
                                       val ↦ balValue[b].setVal])
                            ELSE Send([type ↦ "Terminate", bal ↦ b])
                     ∧ IF ¬Same(QResult) THEN Send([type ↦ "Repair",
                                                    value ↦ maxValue,
                                                    version ↦ maxVersion])
              ∧ UNCHANGED ⟨maxBal, maxAccBal, maxAccVal, maxComBal, maxComVal,
                           dataResult, balValue⟩

**************************************************************************************
Repair(a) ≜  ∧ ∃ m ∈ msgs: ∧ m.type = "Repair"
                           ∧ dataResult' = [dataResult EXCEPT ![a] = [value ↦ m.value, version ↦
                                            m.version]]
             ∧ UNCHANGED ⟨maxBal, maxAccBal, maxAccVal, maxComBal, maxComVal,
                          msgs, balValue⟩
**************************************************************************************

Accept(a) ≜  ∧ ∃ m ∈ msgs : ∧ m.type = "Propose"
                            ∧ 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⟩

Learn(b) ≜  ∧ ¬∃ m ∈ msgs : m.type = "Learn" ∧ 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 ↦ "Learn", bal ↦ b, val ↦ balValue[b].setVal])
          ∧ UNCHANGED ⟨maxBal, maxAccBal, maxAccVal, maxComBal,
                       maxComVal, dataResult, balValue⟩

Ack(a) ≜  ∧ ∃ m ∈ msgs : ∧ m.type = "Learn"
                         ∧ maxBal[a] ≤ m.bal
                         ∧ maxBal' = [maxBal EXCEPT ![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⟩

Next ≜  ∨ ∃ ev, sv ∈ Value, b ∈ Ballot : CAS(ev, sv, b)
        ∨ ∃ b ∈ Ballot : ∨ Propose(b)
                         ∨ Learn(b)
        ∨ ∃ a ∈ Acceptor : ∨ Promise(a)
                           ∨ Accept(a)
                           ∨ Ack(a)
                           ∨ Repair(a)

Spec ≜ Init ∧ □[Next]_vars

                                                                              ┘
\* Modification History
\* Last modified Sun Feb 27 11:17:43 CST 2022 by LENOVO
\* Created Sun Feb 27 08:57:09 CST 2022 by LENOVO
```