MODULE *JustInTimePaxos*

EXTENDS *Naturals*, *Sequences*, *FiniteSets*, *TLC*

The set of *Paxos* replicas

CONSTANT *Replicas*

The set of *Paxos* clients

CONSTANT *Clients*

An empty value

CONSTANT *Nil*

Client request/response types+

CONSTANTS
- *MWriteRequest*,
- *MWriteResponse*,
- *MReadRequest*,
- *MReadResponse*

Server request/response types

CONSTANTS
- *MRepairRequest*,
- *MRepairResponse*,
- *MAbortRequest*,
- *MAbortResponse*,
- *MViewChangeRequest*,
- *MViewChangeResponse*,
- *MStartViewRequest*

Replica roles

CONSTANTS
- *SNormal*,
- *SAborting*,
- *SViewChange*

VARIABLE *replicas*

$globalVars \triangleq \langle replicas \rangle$

VARIABLE *messages*

$messageVars \triangleq \langle messages \rangle$

VARIABLE *cTime*

VARIABLE *cViewID*

VARIABLE $cSeqNum$

VARIABLE $cResps$

VARIABLE $cWrites$

VARIABLE $cReads$

$clientVars \triangleq \langle cTime, cViewID, cSeqNum, cResps, cWrites, cReads \rangle$

VARIABLE $rStatus$

VARIABLE $rLog$

VARIABLE $rViewID$

VARIABLE $rSeqNum$

VARIABLE $rLastView$

VARIABLE $rViewChanges$

VARIABLE $rAbortSeqNum$

VARIABLE $rAbortResps$

$replicaVars \triangleq \langle rStatus, rLog, rViewID, rSeqNum, rLastView, rViewChanges, rAbortSeqNum, rAbortResps \rangle$

VARIABLE $transitions$

$vars \triangleq \langle globalVars, messageVars, clientVars, replicaVars, transitions \rangle$

Helpers

```
RECURSIVE $SeqFromSet(\_)$
$SeqFromSet(S) \triangleq$
  IF $S = \{\}$ THEN $\langle\rangle$
   ELSE LET $x \triangleq$ CHOOSE $x \in S$ : TRUE
        IN  $\langle x \rangle \circ SeqFromSet(S \setminus \{x\})$
```

$Max(s) \triangleq$ CHOOSE $x \in s : \forall y \in s : x \geq y$

$IsQuorum(s) \triangleq Cardinality(s) * 2 \geq Cardinality(Replicas)$

$Quorums \triangleq \{r \in$ SUBSET $Replicas : IsQuorum(r)\}$

$Primary(v) \triangleq replicas[(v \% Len(replicas)) + ($IF $v \geq Len(replicas)$ THEN 1 ELSE $0)]$

$IsPrimary(r) \triangleq Primary(rViewID[r]) = r$

$Replace(l, i, x) \triangleq [j \in 1 \mathrel{..} Max(\{Len(l), i\}) \mapsto$ IF $j = i$ THEN x ELSE $l[j]]$

Messaging helpers

$Sends(ms) \triangleq messages' = messages \cup ms$

$Send(m) \triangleq Sends(\{m\})$

$Replies(req, resps) \triangleq messages' = (messages \cup resps) \setminus \{req\}$

$Reply(req, resp) \triangleq Replies(req, \{resp\})$

$Discard(m) \triangleq messages' = messages \setminus \{m\}$

```
$Write(c) \triangleq$
    $\land cTime' = cTime + 1$
    $\land cSeqNum' = [cSeqNum \text{ EXCEPT } ![c] = cSeqNum[c] + 1]$
    $\land Sends(\{[src \mapsto c,$
                 $dest \mapsto r,$
                 $type \mapsto MWriteRequest,$
                 $viewID \mapsto cViewID[c],$
                 $seqNum \mapsto cSeqNum'[c],$
                 $timestamp \mapsto cTime'] : r \in Replicas\})$
    $\land \text{UNCHANGED } \langle globalVars, replicaVars, cViewID, cResps, cWrites, cReads \rangle$

$Read(c) \triangleq$
      $\land Sends(\{[src \mapsto c,$
                   $dest \mapsto r,$
                   $type \mapsto MReadRequest,$
                   $viewID \mapsto cViewID[c]] : r \in Replicas\})$
      $\land \text{UNCHANGED } \langle globalVars, replicaVars, cTime, cSeqNum, cResps, cWrites, cReads \rangle$

$ChecksumsMatch(c1, c2) \triangleq$
    $\land Len(c1) = Len(c2)$
    $\land \neg\exists i \in \text{DOMAIN } c1 : c1[i] \neq c2[i]$

$IsCommitted(acks) \triangleq$
    $\exists msgs \in \text{SUBSET } acks :$
      $\land \{m.src : m \in msgs\} \in Quorums$
      $\land \exists m1 \in msgs : \forall m2 \in msgs : m1.viewID = m2.viewID \land ChecksumsMatch(m1.checksum, m2.checksu$
      $\land \exists m \in msgs : m.primary$

$HandleWriteResponse(c, r, m) \triangleq$
    $\land \neg\exists w \in cWrites[c] : w.seqNum = m.seqNum$
    $\land \lor \land m.seqNum \notin \text{DOMAIN } cResps[c][r]$
            $\land cResps' = [cResps \text{ EXCEPT } ![c] = [cResps[c] \text{ EXCEPT } ![r] = cResps[c][r] @@ (m.seqNum :> m)]]$
            $\land \text{UNCHANGED } \langle cWrites \rangle$
```

```
        $\lor \land m.seqNum \in \textsc{domain}\ cResps[c][r]$
            Do not overwrite a response from a newer view
            $\land cResps[c][r][m.seqNum].viewID \leq m.viewID$
            $\land cResps' = [cResps\ \textsc{except}\ ![c] = [cResps[c]\ \textsc{except}\ ![r] = [cResps[c][r]\ \textsc{except}\ ![m.seqNum] =$
            $\land \textsc{let}\ committed \triangleq IsCommitted(\{cResps'[c][x][m.seqNum] : x \in \{x \in Replicas : m.seqNum \in \textsc{dom}$
              $\textsc{in}$
                $\lor \land committed$
                  $\land cWrites' = [cWrites\ \textsc{except}\ ![c] = cWrites[c] \cup \{m\}]$
                $\lor \land \neg committed$
                  $\land \textsc{unchanged}\ \langle cWrites \rangle$
    $\land Discard(m)$
    $\land \textsc{unchanged}\ \langle globalVars, replicaVars, cTime, cSeqNum, cReads \rangle$

$HandleReadResponse(c, r, m) \triangleq$
    $\land \lor \land m.primary$
            $\land m \notin cReads[c]$
            $\land cReads' = [cReads\ \textsc{except}\ ![c] = cReads[c] \cup \{m\}]$
        $\lor \land \neg m.primary$
            $\land \textsc{unchanged}\ \langle cReads \rangle$
    $\land Discard(m)$
    $\land \textsc{unchanged}\ \langle globalVars, replicaVars, cTime, cSeqNum, cResps, cWrites \rangle$
```

```
Server request/response handling

$Repair(r, c, m) \triangleq$
    $\land Replies(m, \{[src \mapsto r,$
                       $dest \mapsto d,$
                       $type \mapsto MRepairRequest,$
                       $viewID \mapsto rViewID[r],$
                       $client \mapsto c,$
                       $seqNum \mapsto rSeqNum[r][c] + 1] : d \in Replicas\})$

$Abort(r, c, m) \triangleq$
    $\land IsPrimary(r)$
    $\land rStatus[r] = SNormal$
    $\land rStatus' = [rStatus\ \textsc{except}\ ![r] = SAborting]$
    $\land rAbortResps' = [rAbortResps\ \textsc{except}\ ![r] = [rAbortResps[r]\ \textsc{except}\ ![c] = \{\}]]$
    $\land rAbortSeqNum' = [rAbortSeqNum\ \textsc{except}\ ![r] = [rAbortSeqNum[r]\ \textsc{except}\ ![c] = m.seqNum]]$
    $\land Replies(m, \{[src \mapsto r,$
                       $dest \mapsto d,$
                       $type \mapsto MAbortRequest,$
                       $viewID \mapsto rViewID[r],$
                       $client \mapsto c,$
                       $seqNum \mapsto m.seqNum] : d \in Replicas\})$
```

```
HandleWriteRequest(r, c, m) ≜
    ∧ rStatus[r] = SNormal
    ∧ ∨ ∧ m.viewID  = rViewID[r]
        ∧ m.seqNum = rSeqNum[r][c] + 1
        ∧ rLog′ = [rLog EXCEPT ![r] = Append(rLog[r], m)]
        ∧ rSeqNum′ = [rSeqNum EXCEPT ![r] = m.seqNum]
        ∧ Reply(m, [src       ↦ r,
                    dest      ↦ c,
                    type      ↦ MWriteResponse,
                    seqNum    ↦ m.seqNum,
                    viewID    ↦ rViewID[r],
                    succeeded ↦ TRUE])
      ∨ ∧ m.viewID  = rViewID[r]
        ∧ m.seqNum > rSeqNum[r][c] + 1
        ∧ ∨ ∧ IsPrimary(r)
            ∧ Abort(r, c, m)
          ∨ ∧ ¬IsPrimary(r)
            ∧ Repair(r, c, m)
        ∧ UNCHANGED ⟨rLog⟩
      ∨ ∧ m.viewID < rViewID[r]
        ∧ Reply(m, [src       ↦ r,
                    dest      ↦ c,
                    type      ↦ MWriteResponse,
                    seqNum    ↦ m.seqNum,
                    viewID    ↦ rViewID[r],
                    succeeded ↦ FALSE])
        ∧ UNCHANGED ⟨rLog⟩
    ∧ UNCHANGED ⟨globalVars, clientVars, rStatus, rViewID, rLastView, rViewChanges⟩

HandleReadRequest(r, c, m) ≜
    ∧ rStatus[r] = SNormal
    ∧ Len(rLog[r]) > 0
    ∧ Reply(m, [src       ↦ r,
                dest      ↦ c,
                type      ↦ MReadResponse,
                viewID    ↦ rViewID[r],
                primary   ↦ IsPrimary(r),
                index     ↦ Len(rLog[r]),
                checksum  ↦ rLog[r][Len(rLog[r])].checksum,
                succeeded ↦ TRUE])
    ∧ UNCHANGED ⟨globalVars, clientVars, rStatus, rLog, rViewID, rLastView, rViewChanges⟩

HandleRepairRequest(r, s, m) ≜
    ∧ m.viewID = rViewID[r]
    ∧ IsPrimary(r)
    $\land\ rStatus[r] = SNormal$
    $\land\ \lor\ \land\ m.seqNum \leq Len(rLog[r][m.client])$
            $\land\ Reply(m, [src \mapsto r,$
                              $dest \mapsto s,$
                              $type \mapsto MRepairResponse,$
                              $viewID \mapsto rViewID[r],$
                              $client \mapsto m.client,$
                              $seqNum \mapsto m.seqNum])$
            $\land\ \text{UNCHANGED}\ \langle rStatus, rAbortResps, rAbortSeqNum \rangle$
        $\lor\ \land\ m.seqNum = Len(rLog[r][m.client]) + 1$
            $\land\ Abort(r, m.client, m)$
    $\land\ \text{UNCHANGED}\ \langle globalVars, clientVars \rangle$

$HandleRepairResponse(r, s, m) \triangleq$
    $\land\ HandleWriteRequest(r, m.client, [m\ \text{EXCEPT}\ !.src = m.client])$

$HandleAbortRequest(r, s, m) \triangleq$
    $\land\ m.viewID = rViewID[r]$
    $\land\ m.seqNum \leq Len(rLog[r][m.client]) + 1$
    $\land\ rStatus[r] \in \{SNormal, SAborting\}$
    $\land\ rLog' = [rLog\ \text{EXCEPT}\ ![r] = [rLog[r]\ \text{EXCEPT}\ ![m.client] = Replace(rLog[r][m.client], m.seqNum, Nil)$
    $\land\ \lor\ \land\ m.seqNum > rSeqNum[r][m.client]$
            $\land\ rSeqNum' = [rSeqNum\ \text{EXCEPT}\ ![r] = [rSeqNum[r]\ \text{EXCEPT}\ ![m.client] = m.seqNum]]$
        $\lor\ \land\ m.seqNum \leq rSeqNum[r][m.client]$
            $\land\ \text{UNCHANGED}\ \langle rSeqNum \rangle$
    $\land\ Replies(m, \{[src \mapsto r,$
                          $dest \mapsto Primary(rViewID[r]),$
                          $type \mapsto MAbortResponse,$
                          $viewID \mapsto rViewID[r],$
                          $seqNum \mapsto m.seqNum],$
                         $[src \mapsto r,$
                          $dest \mapsto Primary(rViewID[r]),$
                          $type \mapsto MWriteResponse,$
                          $viewID \mapsto rViewID[r],$
                          $seqNum \mapsto m.seqNum,$
                          $succeeded \mapsto \text{FALSE}]\})$
    $\land\ \text{UNCHANGED}\ \langle globalVars, clientVars, rStatus, rViewID, rLastView, rViewChanges \rangle$

$HandleAbortResponse(r, s, m) \triangleq$
    $\land\ rStatus[r] = SAborting$
    $\land\ m.viewID = rViewID[r]$
    $\land\ IsPrimary(r)$
    $\land\ m.seqNum = rAbortSeqNum[r][m.client]$
    $\land\ rAbortResps' = [rAbortResps\ \text{EXCEPT}\ ![r] = [rAbortResps[r]\ \text{EXCEPT}\ ![m.client] = rAbortResps[r][m.cl$
    $\land\ \text{LET}\ resps \triangleq \{res.src : res \in \{resp \in rAbortResps'[r][m.client] :$
                                             $\land\ resp.viewID = rViewID[r]$
                              $\land$ $resp.seqNum = rAbortSeqNum[r][m.client]\}\}$
           $isQuorum \triangleq r \in resps \land resps \in Quorums$
       IN
         $\lor$ $\land$ $isQuorum$
            $\land$ $rStatus' = [rStatus$ EXCEPT $![r] = [rStatus[r]$ EXCEPT $![m.client] = SNormal]]$
         $\lor$ $\land$ $\neg isQuorum$
            $\land$ UNCHANGED $\langle rStatus \rangle$
   $\land$ UNCHANGED $\langle globalVars, clientVars \rangle$

$ChangeView(r) \triangleq$
   $\land$ $Sends(\{[src \mapsto r,$
                 $dest \mapsto d,$
                 $type \mapsto MViewChangeRequest,$
                 $viewID \mapsto rViewID[r] + 1] : d \in Replicas\})$
   $\land$ UNCHANGED $\langle globalVars, clientVars, replicaVars \rangle$

$HandleViewChangeRequest(r, s, m) \triangleq$
   $\land$ $rViewID[r] < m.viewID$
   $\land$ $rViewID'$      $= [rViewID$ EXCEPT $![r] = m.viewID]$
   $\land$ $rStatus'$      $= [rStatus$ EXCEPT $![r] = SViewChange]$
   $\land$ $rViewChanges' = [rViewChanges$ EXCEPT $![r] = \{\}]$
   $\land$ $Reply(m, [src \mapsto r,$
                  $dest \mapsto Primary(m.viewID),$
                  $type \mapsto MViewChangeResponse,$
                  $viewID \mapsto m.viewID,$
                  $lastNormal \mapsto rLastView[r],$
                  $log \mapsto rLog[r]])$
   $\land$ UNCHANGED $\langle globalVars, clientVars, rLog, rLastView \rangle$

$HandleViewChangeResponse(r, s, m) \triangleq$
   $\land$ $IsPrimary(r)$
   $\land$ $rViewID[r]$    $= m.viewID$
   $\land$ $rStatus[r]$    $= SViewChange$
   $\land$ $rViewChanges' = [rViewChanges$ EXCEPT $![r] = rViewChanges[r] \cup \{m\}]$
   $\land$ LET
         $isViewQuorum(vs) \triangleq IsQuorum(vs) \land \exists v \in vs : v.src = r$
         $newViewChanges$  $\triangleq \{v \in rViewChanges'[r] : v.viewID = rViewID[r]\}$
         $normalViews$     $\triangleq \{v.lastNormal : v \in newViewChanges\}$
         $lastNormal$      $\triangleq$ CHOOSE $v \in normalViews : \forall v2 \in normalViews : v2 \leq v$
         $goodLogs$        $\triangleq \{n.log : n \in \{v \in newViewChanges : v.lastNormal = lastNormal\}\}$
         $combineLogs(ls)$ $\triangleq$
           LET
             $indexLogs(i)$          $\triangleq \{l \in ls : Len(l) \geq i\}$
             $indexEntries(i)$       $\triangleq \{l[i] : l \in indexLogs(i)\}$
             $quorumLogs(i)$         $\triangleq \{L \in$ SUBSET $indexLogs(i) : IsQuorum(L)\}$
             $isCommittedEntry(i, e)$ $\triangleq \forall L \in quorumLogs(i) :$
```

```
                                    ∃ l ∈ L :
                                      ChecksumsMatch(e.checksum, l[i].checksum)
              isCommittedIndex(i)   ≜ ∃ e ∈ indexEntries(i) : isCommittedEntry(i, e)
              commit(i)             ≜ CHOOSE e ∈ indexEntries(i) : isCommittedEntry(i, e)
              maxIndex              ≜ Max({Len(l) : l ∈ ls})
              committedIndexes      ≜ {i ∈ 1 . . maxIndex : isCommittedIndex(i)}
              maxCommit             ≜ IF Cardinality(committedIndexes) > 0 THEN Max(committedIndexe
            IN
              [i ∈ 1 . . maxCommit ↦ commit(i)]
        IN
          ∨ ∧ isViewQuorum(newViewChanges)
            ∧ Replies(m, {[src    ↦ r,
                           dest   ↦ d,
                           type   ↦ MStartViewRequest,
                           viewID ↦ rViewID[r],
                           log    ↦ combineLogs(goodLogs)] : d ∈ Replicas})
          ∨ ∧ ¬isViewQuorum(newViewChanges)
            ∧ Discard(m)
    ∧ UNCHANGED ⟨globalVars, clientVars, rStatus, rViewID, rLog, rLastView⟩

HandleStartViewRequest(r, s, m) ≜
    ∧ ∨ rViewID[r] < m.viewID
      ∨ ∧ rViewID[r] = m.viewID
        ∧ rStatus[r]  = SViewChange
    ∧ rLog′          = [rLog EXCEPT ![r] = m.log]
    ∧ rStatus′       = [rStatus EXCEPT ![r]  = SNormal]
    ∧ rViewID′       = [rViewID EXCEPT ![r] = m.viewID]
    ∧ rLastView′ = [rLastView EXCEPT ![r] = m.viewID]
    ∧ Discard(m)
    ∧ UNCHANGED ⟨globalVars, clientVars, rViewChanges⟩
```

```
InitMessageVars ≜
    ∧ messages = {}

InitClientVars ≜
    ∧ cTime      = 0
    ∧ cViewID    = [c ∈ Clients ↦ 1]
    ∧ cSeqNum    = [c ∈ Clients ↦ 0]
    ∧ cResps = [c ∈ Clients ↦ [r ∈ Replicas ↦ [s ∈ {} ↦ [index ↦ 0, checksum ↦ Nil]]]]
    ∧ cWrites    = [c ∈ Clients ↦ {}]
    ∧ cReads     = [c ∈ Clients ↦ {}]

InitReplicaVars ≜
    ∧ replicas        = SeqFromSet(Replicas)
```

```
$\land\ rStatus \qquad\qquad = [r \in Replicas \mapsto SNormal]$
$\land\ rLog \qquad\qquad\quad = [r \in Replicas \mapsto [c \in Clients \mapsto \langle\rangle]]$
$\land\ rSeqNum \qquad\quad = [r \in Replicas \mapsto [c \in Clients \mapsto 0]]$
$\land\ rAbortSeqNum = [r \in Replicas \mapsto [c \in Clients \mapsto 0]]$
$\land\ rAbortResps \quad = [r \in Replicas \mapsto [c \in Clients \mapsto \{\}]]$
$\land\ rViewID \qquad\quad = [r \in Replicas \mapsto 1]$
$\land\ rLastView \qquad = [r \in Replicas \mapsto 1]$
$\land\ rViewChanges = [r \in Replicas \mapsto \{\}]$

$Init \triangleq$
    $\land\ InitMessageVars$
    $\land\ InitClientVars$
    $\land\ InitReplicaVars$
    $\land\ transitions = 0$
```

The type invariant checks that no read ever reads a different value than a previous write

```
$Inv \triangleq$
   $\land\ \forall\, c1,\ c2 \in Clients :$
        $\neg\exists\, r \in cReads[c1] :$
            $\exists\, w \in cWrites[c2] :$
                $\land\ r.index = w.index$
                $\land\ \neg ChecksumsMatch(r.checksum,\ w.checksum)$
   $\land\ \forall\, c1,\ c2 \in Clients :$
        $\neg\exists\, r1 \in cReads[c1] :$
            $\exists\, r2 \in cReads[c2] :$
                $\land\ r1.index = r2.index$
                $\land\ \neg ChecksumsMatch(r1.checksum,\ r2.checksum)$

$Transition \triangleq transitions' = transitions + 1$

$Next \triangleq$
    $\lor\ \exists\, c \in Clients :$
        $\land\ Write(c)$
        $\land\ Transition$
    $\lor\ \exists\, c \in Clients :$
        $\land\ Read(c)$
        $\land\ Transition$
    $\lor\ \exists\, r \in Replicas :$
        $\land\ ChangeView(r)$
        $\land\ Transition$
    $\lor\ \exists\, m \in messages :$
        $\land\ m.type = MWriteRequest$
        $\land\ HandleWriteRequest(m.dest,\ m.src,\ m)$
        $\land\ Transition$
```

```
$\lor \exists m \in messages :$
    $\land m.type = MWriteResponse$
    $\land HandleWriteResponse(m.dest, m.src, m)$
    $\land Transition$
$\lor \exists m \in messages :$
    $\land m.type = MReadRequest$
    $\land HandleReadRequest(m.dest, m.src, m)$
    $\land Transition$
$\lor \exists m \in messages :$
    $\land m.type = MReadResponse$
    $\land HandleReadResponse(m.dest, m.src, m)$
    $\land Transition$
$\lor \exists m \in messages :$
    $\land m.type = MRepairRequest$
    $\land HandleRepairRequest(m.dest, m.src, m)$
    $\land Transition$
$\lor \exists m \in messages :$
    $\land m.type = MRepairResponse$
    $\land HandleRepairResponse(m.dest, m.src, m)$
    $\land Transition$
$\lor \exists m \in messages :$
    $\land m.type = MAbortRequest$
    $\land HandleAbortRequest(m.dest, m.src, m)$
    $\land Transition$
$\lor \exists m \in messages :$
    $\land m.type = MAbortResponse$
    $\land HandleAbortResponse(m.dest, m.src, m)$
    $\land Transition$
$\lor \exists m \in messages :$
    $\land m.type = MViewChangeRequest$
    $\land HandleViewChangeRequest(m.dest, m.src, m)$
    $\land Transition$
$\lor \exists m \in messages :$
    $\land m.type = MViewChangeResponse$
    $\land HandleViewChangeResponse(m.dest, m.src, m)$
    $\land Transition$
$\lor \exists m \in messages :$
    $\land m.type = MStartViewRequest$
    $\land HandleStartViewRequest(m.dest, m.src, m)$
    $\land Transition$
```

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

\ ∗ Modification History

\ ∗ Last modified *Tue Sep* 22 03:02:49 *PDT* 2020 by *jordanhalterman*

$\backslash *$ Created *Fri Sep* 18 22:45:21 *PDT* 2020 by *jordanhalterman*