## MODULE *JustInTimePaxos*

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

The set of *Paxos* replicas

CONSTANT *Replicas*

The set of *Paxos* clients

CONSTANT *Clients*

The set of possible values

CONSTANT *Values*

An empty value

CONSTANT *Nil*

Request/response types+

CONSTANTS

    *MClientRequest*,
    *MClientResponse*,
    *MRepairRequest*,
    *MRepairResponse*,
    *MAbortRequest*,
    *MAbortResponse*,
    *MViewChangeRequest*,
    *MViewChangeResponse*,
    *MStartViewRequest*

Replica roles

CONSTANTS

    *SNormal*,
    *SAborting*,
    *SViewChange*

Entry types

CONSTANTS

    *TValue*,
    *TNoOp*

---

VARIABLE *replicas*

$globalVars \triangleq \langle replicas \rangle$

VARIABLE *messages*

$messageVars \triangleq \langle messages \rangle$

```
VARIABLE cTime

VARIABLE cViewID

VARIABLE cSeqNum

VARIABLE cResps

VARIABLE cCommits

clientVars ≜ ⟨cTime, cViewID, cSeqNum, cResps, cCommits⟩

VARIABLE rStatus

VARIABLE rLog

VARIABLE rViewID

VARIABLE rSeqNum

VARIABLE rTimestamp

VARIABLE rLastView

VARIABLE rViewChanges

VARIABLE rAbortSeqNum

VARIABLE rAbortResps

replicaVars ≜ ⟨rStatus, rLog, rViewID, rSeqNum, rTimestamp, rLastView, rViewChanges, rAbortSeqNum, r

VARIABLE transitions

vars ≜ ⟨globalVars, messageVars, clientVars, replicaVars, transitions⟩
```

---

Helpers

```
RECURSIVE SeqFromSet(_)
SeqFromSet(S) ≜
   IF S = {} THEN
      ⟨⟩
    ELSE LET x ≜ CHOOSE x ∈ S : TRUE
         IN   ⟨x⟩ ∘ SeqFromSet(S \ {x})

Pick(S) ≜ CHOOSE s ∈ S : TRUE

RECURSIVE SetReduce(_, _, _)
SetReduce(Op(_, _), S, value) ≜
   IF S = {} THEN
```

```
        value
    ELSE
        LET s ≜ Pick(S)
        IN  SetReduce(Op, S \ {s}, Op(s, value))

Max(s) ≜ CHOOSE x ∈ s : ∀ y ∈ s : x ≥ y

Sum(S) ≜ LET _op(a, b) ≜ a + b
         IN  SetReduce(_op, S, 0)

IsQuorum(s) ≜ Cardinality(s) * 2 ≥ Cardinality(Replicas)

Quorums ≜ {r ∈ SUBSET Replicas : IsQuorum(r)}

Primary(v) ≜ replicas[(v%Len(replicas)) + (IF v ≥ Len(replicas) THEN 1 ELSE 0)]

IsPrimary(r) ≜ Primary(rViewID[r]) = r
```

---

Messaging helpers

```
Sends(ms) ≜ messages′ = messages ∪ ms

Send(m) ≜ Sends({m})

Replies(req, resps) ≜ messages′ = (messages ∪ resps) \ {req}

Reply(req, resp) ≜ Replies(req, {resp})

Discard(m) ≜ messages′ = messages \ {m}
```

---

```
Write(c) ≜
    ∧ cTime′ = cTime + 1
    ∧ cSeqNum′ = [cSeqNum EXCEPT ![c] = cSeqNum[c] + 1]
    ∧ Sends({[src       ↦ c,
              dest      ↦ r,
              type      ↦ MClientRequest,
              viewID    ↦ cViewID[c],
              seqNum    ↦ cSeqNum′[c],
              timestamp ↦ cTime′] : r ∈ Replicas})
    ∧ UNCHANGED ⟨globalVars, replicaVars, cViewID, cResps⟩

HandleClientResponse(c, r, m) ≜
    ∧ ∨ ∧ m.viewID = cViewID[c]
        ∧ IF m.seqNum ∉ DOMAIN cResps[c][r] THEN
              cResps′ = [cResps EXCEPT ![c] = [cResps[c] EXCEPT ![r] = cResps[c][r] @@ (m.index :> m)]]
          ELSE
```

```
                 cResps′ = [cResps EXCEPT ![c] = [cResps[c] EXCEPT ![r] = [cResps[c][r] EXCEPT ![m.index] =
              ∧ LET
                   allResps      ≜ {cResps[c][r][r1] : r1 ∈ {r2 ∈ Replicas : r2 ∈ DOMAIN cResps[c][r]}}
                   succeededResps ≜ {resp ∈ allResps : resp.viewID = cViewID[c] ∧ resp.succeeded}
                   isCommitted  ≜  ∧ ∃ resp ∈ succeededResps : resp.src = Primary(resp.viewID)
                                   ∧ {resp.src : resp ∈ succeededResps} ∈ Quorums
                IN
                   ∧ ∨ ∧ isCommitted
                         ∧ cCommits′ = [cCommits EXCEPT ![c] = cCommits[c] ∪ {CHOOSE resp ∈ succeededResp
                       ∨ ∧ ¬isCommitted
                         ∧ UNCHANGED ⟨cCommits⟩
                   ∧ UNCHANGED ⟨cViewID, cSeqNum⟩
           ∨ ∧ m.viewID > cViewID[c]
              ∧ cViewID′ = [cViewID EXCEPT ![c] = m.viewID]
              ∧ cSeqNum′ = [cSeqNum EXCEPT ![c] = 0]
              ∧ cResps′   = [cResps  EXCEPT ![c]  = [i ∈ Replicas ↦ {}]]
              ∧ UNCHANGED ⟨cCommits⟩
           ∨ ∧ m.viewID < cViewID[c]
              ∧ UNCHANGED ⟨cCommits⟩
        ∧ Discard(m)
        ∧ UNCHANGED ⟨globalVars, replicaVars, cTime, cSeqNum⟩
```

---

Log helpers

```
ReplaceEntry(l, i, x) ≜ [j ∈ 1 . . Max({Len(l), i}) ↦ IF j = i THEN x ELSE l[j]]
```

---

Server request/response handling

```
Repair(r, c, m) ≜
    ∧ Replies(m, {[src     ↦ r,
                   dest    ↦ d,
                   type    ↦ MRepairRequest,
                   viewID  ↦ rViewID[r],
                   client  ↦ c,
                   seqNum  ↦ rSeqNum[r][c] + 1] : d ∈ Replicas})

Abort(r, c, m) ≜
    ∧ IsPrimary(r)
    ∧ rStatus[r]        = SNormal
    ∧ rStatus′          = [rStatus        EXCEPT ![r]  = SAborting]
    ∧ rAbortResps′      = [rAbortResps    EXCEPT ![r]  = [rAbortResps[r] EXCEPT ![c] = {}]]
    ∧ rAbortSeqNum′     = [rAbortSeqNum   EXCEPT ![r] = [rAbortSeqNum[r] EXCEPT ![c] = m.seqNum]]
    ∧ Replies(m, {[src       ↦ r,
                      $dest \mapsto d,$
                      $type \mapsto MAbortRequest,$
                      $viewID \mapsto rViewID[r],$
                      $client \mapsto c,$
                      $seqNum \mapsto m.seqNum] : d \in Replicas\})$

$HandleClientRequest(r, c, m) \triangleq$
    $\land rStatus[r] = SNormal$
    $\land \lor \land m.viewID = rViewID[r]$
          $\land$ LET
                 $lastIndex \triangleq Sum(\{Len(rLog[r][i]) : i \in Clients\})$
                 $index \triangleq lastIndex + 1$
                 $lastTimestamp \triangleq rTimestamp[r]$
                 $isSequential \triangleq m.seqNum = rSeqNum[r][c] + 1$
                 $isLinear \triangleq m.timestamp > lastTimestamp$
             IN
                $\lor \land isSequential$
                  $\land isLinear$
                  $\land rLog' = [rLog$ EXCEPT $![r] = [$
                            $rLog[r]$ EXCEPT $![c] =$
                                $Append(rLog[r][c], [type \mapsto TValue,$
                                                     $index \mapsto index,$
                                                     $value \mapsto m.value,$
                                                     $timestamp \mapsto m.timestamp])]]$
                  $\land rSeqNum' = [rSeqNum$ EXCEPT $![r] = [rSeqNum[r]$ EXCEPT $![c] = m.seqNum]]$
                  $\land rTimestamp' = [rTimestamp$ EXCEPT $![r] = m.timestamp]$
                  $\land Reply(m, [src \mapsto r,$
                                $dest \mapsto c,$
                                $type \mapsto MClientResponse,$
                                $index \mapsto index,$
                                $viewID \mapsto rViewID[r],$
                                $succeeded \mapsto$ TRUE$])$
                $\lor \land \lor \neg isSequential$
                     $\lor \neg isLinear$
                  $\land \lor \land IsPrimary(r)$
                        $\land Abort(r, c, m)$
                     $\lor \land \neg IsPrimary(r)$
                        $\land Reply(m, [src \mapsto r,$
                                      $dest \mapsto c,$
                                      $type \mapsto MClientResponse,$
                                      $index \mapsto index,$
                                      $viewID \mapsto rViewID[r],$
                                      $succeeded \mapsto$ FALSE$])$
                  $\land$ UNCHANGED $\langle rLog \rangle$
       $\lor \land m.viewID < rViewID[r]$
```

```
        ∧ Reply(m, [src        ↦ r,
                    dest       ↦ c,
                    type       ↦ MClientResponse,
                    viewID     ↦ rViewID[r],
                    succeeded  ↦ FALSE])
        ∧ UNCHANGED ⟨rLog⟩
    ∧ UNCHANGED ⟨globalVars, clientVars, rStatus, rViewID, rLastView, rViewChanges⟩

HandleRepairRequest(r, s, m) ≜
    ∧ m.viewID = rViewID[r]
    ∧ IsPrimary(r)
    ∧ rStatus[r] = SNormal
    ∧ LET index ≜ Len(rLog[r][m.client]) + 1 − (rSeqNum[r] − m.seqNum)
      IN
        ∧ ∨ ∧ index ≤ Len(rLog[r][m.client])
            ∧ Reply(m, [src      ↦ r,
                        dest     ↦ s,
                        type     ↦ MRepairResponse,
                        viewID   ↦ rViewID[r],
                        client   ↦ m.client,
                        seqNum   ↦ m.seqNum])
            ∧ UNCHANGED ⟨rStatus, rAbortResps, rAbortSeqNum⟩
          ∨ ∧ index = Len(rLog[r][m.client]) + 1
            ∧ Abort(r, m.client, m)
    ∧ UNCHANGED ⟨globalVars, clientVars⟩

HandleRepairResponse(r, s, m) ≜
    ∧ HandleClientRequest(r, m.client, [m EXCEPT !.src = m.client])

HandleAbortRequest(r, s, m) ≜
    ∧ m.viewID = rViewID[r]
    ∧ rStatus[r] ∈ {SNormal, SAborting}
    ∧ LET index ≜ Len(rLog[r][m.client]) + 1 − (rSeqNum[r] − m.seqNum)
      IN
        ∧ index ≤ Len(rLog[r][m.client]) + 1
        ∧ rLog′ = [rLog EXCEPT ![r] = [rLog[r] EXCEPT ![m.client] = ReplaceEntry(rLog[r][m.client], inde
        ∧ ∨ ∧ m.seqNum > rSeqNum[r][m.client]
            ∧ rSeqNum′ = [rSeqNum EXCEPT ![r] = [rSeqNum[r] EXCEPT ![m.client] = m.seqNum]]
          ∨ ∧ m.seqNum ≤ rSeqNum[r][m.client]
            ∧ UNCHANGED ⟨rSeqNum⟩
        ∧ Replies(m, {[src      ↦ r,
                       dest     ↦ Primary(rViewID[r]),
                       type     ↦ MAbortResponse,
                       viewID   ↦ rViewID[r],
                       seqNum   ↦ m.seqNum],
                      [src      ↦ r,
                             dest      ↦ Primary(rViewID[r]),
                             type      ↦ MClientResponse,
                             viewID    ↦ rViewID[r],
                             seqNum    ↦ m.seqNum,
                             succeeded ↦ FALSE]})
    ∧ UNCHANGED ⟨globalVars, clientVars, rStatus, rViewID, rLastView, rViewChanges⟩

HandleAbortResponse(r, s, m) ≜
    ∧ rStatus[r] = SAborting
    ∧ m.viewID = rViewID[r]
    ∧ IsPrimary(r)
    ∧ m.seqNum = rAbortSeqNum[r][m.client]
    ∧ rAbortResps′ = [rAbortResps EXCEPT ![r] = [rAbortResps[r] EXCEPT ![m.client] = rAbortResps[r][m.cl
    ∧ LET resps ≜ {res.src : res ∈ {resp ∈ rAbortResps′[r][m.client] :
                                   ∧ resp.viewID  = rViewID[r]
                                   ∧ resp.seqNum = rAbortSeqNum[r][m.client]}}
          isQuorum ≜ r ∈ resps ∧ resps ∈ Quorums
      IN
         ∨ ∧ isQuorum
           ∧ rStatus′ = [rStatus EXCEPT ![r] = [rStatus[r] EXCEPT ![m.client] = SNormal]]
         ∨ ∧ ¬isQuorum
           ∧ UNCHANGED ⟨rStatus⟩
    ∧ UNCHANGED ⟨globalVars, clientVars⟩

ChangeView(r) ≜
    ∧ Sends({[src    ↦ r,
              dest   ↦ d,
              type   ↦ MViewChangeRequest,
              viewID ↦ rViewID[r] + 1] : d ∈ Replicas})
    ∧ UNCHANGED ⟨globalVars, clientVars, replicaVars⟩

HandleViewChangeRequest(r, s, m) ≜
    ∧ rViewID[r] < m.viewID
    ∧ rViewID′      = [rViewID EXCEPT ![r] = m.viewID]
    ∧ rStatus′      = [rStatus EXCEPT ![r]  = SViewChange]
    ∧ rViewChanges′ = [rViewChanges EXCEPT ![r] = {}]
    ∧ Reply(m, [src        ↦ r,
                dest       ↦ Primary(m.viewID),
                type       ↦ MViewChangeResponse,
                viewID     ↦ m.viewID,
                lastViewID ↦ rLastView[r],
                logs       ↦ rLog[r]])
    ∧ UNCHANGED ⟨globalVars, clientVars, rLog, rSeqNum, rAbortSeqNum, rAbortResps, rLastView⟩

HandleViewChangeResponse(r, s, m) ≜
    ∧ IsPrimary(r)
    ∧ rViewID[r]       = m.viewID
    ∧ rStatus[r]       = SViewChange
    ∧ rViewChanges′ = [rViewChanges EXCEPT ![r] = rViewChanges[r] ∪ {m}]
    ∧ LET viewChanges      ≜ {v ∈ rViewChanges′[r][m.client] : ∧ v.viewID = rViewID[r]}
          viewSources      ≜ {v.src : v ∈ viewChanges}
          isQuorum         ≜ r ∈ viewSources ∧ viewSources ∈ Quorums
          lastViews        ≜ {v.lastViewID : v ∈ viewChanges}
          lastView         ≜ (CHOOSE v1 ∈ lastViews : ∀ v2 ∈ lastViews : v2 ≤ v1)
          viewLogs         ≜ [c ∈ Clients ↦ {v1.logs[c] : v1 ∈ {v2 ∈ viewChanges : v2.lastView = lastView}
          mergeEnts(es)    ≜
              IF es = {} ∨ ∃ e ∈ es : r.type = TNoOp THEN
                  [type ↦ TNoOp]
               ELSE
                  CHOOSE e ∈ es : e.type ≠ TNoOp
          range(ls)        ≜ Max({Len(l) : l ∈ ls})
          entries(ls, i)   ≜ {l[i] : l ∈ {k ∈ ls : i ≤ Len(k)}}
          mergeLogs(ls)    ≜ [i ∈ 1 . . range(ls) ↦ mergeEnts(entries(ls, i))]
      IN
          ∨ ∧ isQuorum
            ∧ Replies(m, {[src      ↦ r,
                           dest     ↦ d,
                           type     ↦ MStartViewRequest,
                           viewID   ↦ rViewID[r],
                           logs     ↦ [c ∈ Clients ↦ mergeLogs(viewLogs[c])]] : d ∈ Replicas})
          ∨ ∧ ¬isQuorum
            ∧ Discard(m)
    ∧ UNCHANGED ⟨globalVars, clientVars, rStatus, rViewID, rLog, rSeqNum, rAbortSeqNum, rAbortResps, r

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 ≜
```

```
    $\land\ cTime \quad\ \ = 0$
    $\land\ cViewID \ \ = [c \in Clients \mapsto 1]$
    $\land\ cSeqNum \ = [c \in Clients \mapsto 0]$
    $\land\ cResps \quad = [c \in Clients \mapsto [r \in Replicas \mapsto [s \in \{\} \mapsto [index \mapsto 0,\ checksum \mapsto Nil]]]]$
    $\land\ cCommits = [c \in Clients \mapsto \{\}]$

$InitReplicaVars \triangleq$
    $\land\ replicas \qquad\quad = SeqFromSet(Replicas)$
    $\land\ rStatus \qquad\ \ = [r \in Replicas \mapsto SNormal]$
    $\land\ rLog \qquad\qquad = [r \in Replicas \mapsto [c \in Clients \mapsto \langle\rangle]]$
    $\land\ rSeqNum \qquad = [r \in Replicas \mapsto [c \in Clients \mapsto 0]]$
    $\land\ rTimestamp \quad = [r \in Replicas \mapsto 0]$
    $\land\ rAbortSeqNum = [r \in Replicas \mapsto [c \in Clients \mapsto 0]]$
    $\land\ rAbortResps \quad = [r \in Replicas \mapsto [c \in Clients \mapsto \{\}]]$
    $\land\ rViewID \qquad\ = [r \in Replicas \mapsto 1]$
    $\land\ rLastView \quad\ \ = [r \in Replicas \mapsto 1]$
    $\land\ rViewChanges = [r \in Replicas \mapsto \{\}]$

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

---

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

```
$Inv \triangleq$
  $\forall\, c1,\ c2 \in Clients :$
    $\forall\, e1 \in cCommits[c1] :$
      $\neg\exists\, e2 \in cCommits[c2] :$
          $\land\ e1.index = e2.index$
          $\land\ e1.value \neq e2.value$

$Transition \triangleq transitions' = transitions + 1$

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

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

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

\ ∗ Modification History

\ ∗ Last modified *Tue Sep* 22 09:53:52 *PDT* 2020 by *jordanhalterman*

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