```
┌──────────────── 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 ≜ ⟨replicas⟩

VARIABLE messages

messageVars ≜ ⟨messages⟩
```

VARIABLE $cTime$

VARIABLE $cViewID$

VARIABLE $cSeqNum$

VARIABLE $cResps$

VARIABLE $cCommits$

$clientVars \triangleq \langle cTime, cViewID, cSeqNum, cResps, cCommits \rangle$

VARIABLE $rStatus$

VARIABLE $rLog$

VARIABLE $rViewID$

VARIABLE $rSeqNum$

VARIABLE $rTimestamp$

VARIABLE $rLastViewID$

VARIABLE $rViewChanges$

VARIABLE $rAbortPoint$

VARIABLE $rAbortResps$

$replicaVars \triangleq \langle rStatus, rLog, rViewID, rSeqNum, rTimestamp, rLastViewID, rViewChanges, rAbortPoint, r$

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    $\langle x \rangle \circ SeqFromSet(S \setminus \{x\})$

$Pick(S) \triangleq$ CHOOSE $s \in S$ : TRUE

RECURSIVE $SetReduce(\_, \_, \_)$
$SetReduce(Op(\_, \_), S, value) \triangleq$
    IF $S = \{\}$ THEN
```

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

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

```
$Sum(S) \triangleq$ LET $\_op(a, b) \triangleq a + b$
                   IN  $SetReduce(\_op, S, 0)$
```

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

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

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

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

Messaging helpers

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

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

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

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

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

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

$HandleClientResponse(c, r, m) \triangleq$
    $\land \lor \land m.viewID = cViewID[c]$
            $\land$ IF $m.seqNum \notin$ DOMAIN $cResps[c][r]$ THEN
                  $cResps' = [cResps$ EXCEPT $![c] = [cResps[c]$ EXCEPT $![r] = cResps[c][r] @@ (m.seqNum :> m)]]$
```

```
           ELSE
             cResps′ = [cResps EXCEPT ![c] = [cResps[c] EXCEPT ![r] = [cResps[c][r] EXCEPT ![m.seqNum]
        ∧ LET
             allResps        ≜ {cResps[c][r1][m.seqNum] : r1 ∈ {r2 ∈ Replicas : m.seqNum ∈ DOMAIN cR
             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]]

AppendEntry(l, r, c, e) ≜ [l EXCEPT ![r] = [l[r] EXCEPT ![c] = Append(l[r][c], e)]]
```

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] = {}]
    ∧ rAbortPoint′ = [rAbortPoint EXCEPT ![r] = [client ↦ c, seqNum ↦ m.seqNum]]
    ∧ Replies(m, {[src       ↦ r,
                   dest      ↦ d,
                   type      ↦ MAbortRequest,
                   viewID    ↦ rViewID[r],
                   client    ↦ c,
                   seqNum    ↦ m.seqNum,
                   timestamp ↦ m.timestamp] : d ∈ Replicas})

HandleClientRequest(r, c, m) ≜
    ∧ rStatus[r] = SNormal
    ∧ ∨ ∧ m.viewID = rViewID[r]
        ∧ LET
              lastIndex     ≜ Sum({Len(rLog[r][i]) : i ∈ Clients})
              index         ≜ lastIndex + 1
              lastTimestamp ≜ rTimestamp[r]
              isSequential  ≜ m.seqNum = rSeqNum[r][c] + 1
              isLinear      ≜ m.timestamp > lastTimestamp
              entry         ≜ [type      ↦ TValue,
                               index     ↦ index,
                               value     ↦ m.value,
                               timestamp ↦ m.timestamp]
          IN
              ∨ ∧ isSequential
                ∧ isLinear
                ∧ rLog′ = AppendEntry(rLog, r, c, entry)
                ∧ rSeqNum′ = [rSeqNum EXCEPT ![r] = [rSeqNum[r] EXCEPT ![c] = m.seqNum]]
                ∧ rTimestamp′ = [rTimestamp EXCEPT ![r] = m.timestamp]
                ∧ Reply(m, [src       ↦ r,
                            dest      ↦ c,
                            type      ↦ MClientResponse,
                            viewID    ↦ rViewID[r],
                            seqNum    ↦ m.seqNum,
                            index     ↦ index,
                            succeeded ↦ TRUE])
                ∧ UNCHANGED ⟨rStatus, rAbortPoint, rAbortResps⟩
              ∨ ∧ ∨ ¬isSequential
                  ∨ ¬isLinear
                ∧ ∨ ∧ IsPrimary(r)
                    ∧ Abort(r, c, m)
                  ∨ ∧ ¬IsPrimary(r)
                    ∧ Reply(m, [src       ↦ r,
                                dest      ↦ c,
                                type      ↦ MClientResponse,
                                          $viewID \mapsto rViewID[r],$
                                          $seqNum \mapsto m.seqNum,$
                                          $succeeded \mapsto \text{FALSE}])$
                                  $\land$ UNCHANGED $\langle rStatus, rAbortPoint, rAbortResps \rangle$
                              $\land$ UNCHANGED $\langle rLog, rSeqNum, rTimestamp \rangle$
        $\lor$ $\land$ $m.viewID < rViewID[r]$
           $\land$ $Reply(m, [src \mapsto r,$
                          $dest \mapsto c,$
                          $type \mapsto MClientResponse,$
                          $viewID \mapsto rViewID[r],$
                          $seqNum \mapsto m.seqNum,$
                          $succeeded \mapsto \text{FALSE}])$
           $\land$ UNCHANGED $\langle rStatus, rLog, rSeqNum, rTimestamp, rAbortPoint, rAbortResps \rangle$
    $\land$ UNCHANGED $\langle globalVars, clientVars, rViewID, rLastViewID, rViewChanges \rangle$

$HandleRepairRequest(r, s, m) \triangleq$
    $\land$ $m.viewID = rViewID[r]$
    $\land$ $IsPrimary(r)$
    $\land$ $rStatus[r] = SNormal$
    $\land$ LET $index \triangleq Len(rLog[r][m.client]) + 1 - (rSeqNum[r] - m.seqNum)$
      IN
         $\land$ $\lor$ $\land$ $index \leq Len(rLog[r][m.client])$
                $\land$ $Reply(m, [src \mapsto r,$
                               $dest \mapsto s,$
                               $type \mapsto MRepairResponse,$
                               $viewID \mapsto rViewID[r],$
                               $client \mapsto m.client,$
                               $seqNum \mapsto m.seqNum])$
                $\land$ UNCHANGED $\langle rStatus, rAbortPoint, rAbortResps \rangle$
            $\lor$ $\land$ $index = Len(rLog[r][m.client]) + 1$
                $\land$ $Abort(r, m.client, m)$
    $\land$ UNCHANGED $\langle globalVars, clientVars \rangle$

$HandleRepairResponse(r, s, m) \triangleq$
    $\land$ $HandleClientRequest(r, m.client, [m$ EXCEPT $!.src = m.client])$

$HandleAbortRequest(r, s, m) \triangleq$
    $\land$ $m.viewID = rViewID[r]$
    $\land$ $rStatus[r] \in \{SNormal, SAborting\}$
    $\land$ LET
          $offset \triangleq Len(rLog[r][m.client]) + 1 - (rSeqNum[r][m.client] - m.seqNum)$
          $entry \triangleq [type \mapsto TNoOp, timestamp \mapsto m.timestamp]$
      IN
         $\land$ $offset \leq Len(rLog[r][m.client]) + 1$
         $\land$ $rLog' = AppendEntry(rLog, r, m.client, entry)$
         $\land$ $rTimestamp' = [rTimestamp$ EXCEPT $![r] = Max(\{rTimestamp[r], m.timestamp\})]$
```

```
          ∧ rSeqNum′ = [rSeqNum EXCEPT ![r] = [rSeqNum[r] EXCEPT ![m.client] = Max({rSeqNum[r][m.cl
          ∧ Replies(m, {[src        ↦ r,
                        dest       ↦ Primary(rViewID[r]),
                        type       ↦ MAbortResponse,
                        viewID     ↦ rViewID[r],
                        client     ↦ m.client,
                        seqNum     ↦ m.seqNum],
                       [src        ↦ r,
                        dest       ↦ m.client,
                        type       ↦ MClientResponse,
                        viewID     ↦ rViewID[r],
                        seqNum     ↦ m.seqNum,
                        succeeded  ↦ FALSE]})
      ∧ UNCHANGED ⟨globalVars, clientVars, rStatus, rAbortPoint, rAbortResps, rViewID, rLastViewID, rView

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

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    ↦ rLastViewID[r],
                    logs          ↦ rLog[r]])
      ∧ UNCHANGED ⟨globalVars, clientVars, rLog, rSeqNum, rTimestamp, rAbortPoint, rAbortResps, rLastVie

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] : v.viewID = rViewID[r]}
            viewSources      ≜ {v.src : v ∈ viewChanges}
            isQuorum         ≜ r ∈ viewSources ∧ viewSources ∈ Quorums
            lastViewIDs      ≜ {v.lastViewID : v ∈ viewChanges}
            lastViewID       ≜ (CHOOSE v1 ∈ lastViewIDs : ∀ v2 ∈ lastViewIDs : v2 ≤ v1)
            viewLogs         ≜ [c ∈ Clients ↦ {v1.logs[c] : v1 ∈ {v2 ∈ viewChanges : v2.lastViewID = lastVie
            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))]
            viewLog          ≜ [c ∈ Clients ↦ mergeLogs(viewLogs[c])]
            viewRange        ≜ Max({Len(viewLog[c]) : c ∈ Clients})
            viewTimestamp    ≜ IF viewRange > 0 THEN
                                     Max(UNION {{l[i].timestamp : i ∈ DOMAIN l} : l ∈ {viewLog[c] : c ∈ Client
                                  ELSE 0
        IN
            ∨ ∧ isQuorum
              ∧ Replies(m, {[src          ↦ r,
                             dest         ↦ d,
                             type         ↦ MStartViewRequest,
                             viewID       ↦ rViewID[r],
                             timestamp    ↦ viewTimestamp,
                             log          ↦ viewLog] : d ∈ Replicas})
            ∨ ∧ ¬isQuorum
              ∧ Discard(m)
      ∧ UNCHANGED ⟨globalVars, clientVars, rStatus, rViewID, rLog, rSeqNum, rTimestamp, rAbortPoint, rAb

HandleStartViewRequest(r, s, m) ≜
      ∧ ∨ rViewID[r] < m.viewID
```

```
      ∨ ∧ rViewID[r] = m.viewID
        ∧ rStatus[r]  = SViewChange
    ∧ rLog'         = [rLog         EXCEPT ![r]  = m.log]
    ∧ rSeqNum'      = [rSeqNum      EXCEPT ![r] = [c ∈ Clients ↦ 0]]
    ∧ rTimestamp'   = [rTimestamp   EXCEPT ![r] = m.timestamp]
    ∧ rStatus'      = [rStatus      EXCEPT ![r]  = SNormal]
    ∧ rViewID'      = [rViewID      EXCEPT ![r]  = m.viewID]
    ∧ rLastViewID'  = [rLastViewID  EXCEPT ![r]  = m.viewID]
    ∧ Discard(m)
    ∧ UNCHANGED ⟨globalVars, clientVars, rAbortPoint, rAbortResps, rViewChanges⟩
```

```
InitMessageVars ≜
    ∧ messages = {}

InitClientVars ≜
    ∧ cTime     = 0
    ∧ cViewID   = [c ∈ Clients ↦ 1]
    ∧ cSeqNum   = [c ∈ Clients ↦ 0]
    ∧ cResps    = [c ∈ Clients ↦ [r ∈ Replicas ↦ [s ∈ {} ↦ [index ↦ 0]]]]
    ∧ cCommits  = [c ∈ Clients ↦ {}]

InitReplicaVars ≜
    ∧ replicas      = SeqFromSet(Replicas)
    ∧ rStatus       = [r ∈ Replicas ↦ SNormal]
    ∧ rLog          = [r ∈ Replicas ↦ [c ∈ Clients ↦ ⟨⟩]]
    ∧ rSeqNum       = [r ∈ Replicas ↦ [c ∈ Clients ↦ 0]]
    ∧ rTimestamp    = [r ∈ Replicas ↦ 0]
    ∧ rAbortPoint   = [r ∈ Replicas ↦ [client ↦ Nil, seqNum ↦ 0]]
    ∧ rAbortResps   = [r ∈ Replicas ↦ {}]
    ∧ rViewID       = [r ∈ Replicas ↦ 1]
    ∧ rLastViewID   = [r ∈ Replicas ↦ 1]
    ∧ rViewChanges  = [r ∈ Replicas ↦ {}]

Init ≜
    ∧ InitMessageVars
    ∧ InitClientVars
    ∧ InitReplicaVars
    ∧ transitions = 0
```

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

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

$Transition \triangleq transitions' = transitions + 1$

$Next \triangleq$
    $\lor\ \exists c \in Clients :$
          $\exists v \in Values :$
            $\land\ Write(c, v)$
            $\land\ Transition$
    $\lor\ \exists r \in Replicas :$
          $\land\ ChangeView(r)$
          $\land\ Transition$
    $\lor\ \exists m \in messages :$
          $\land\ m.type = MClientRequest$
          $\land\ HandleClientRequest(m.dest, m.src, m)$
          $\land\ Transition$
    $\lor\ \exists m \in messages :$
          $\land\ m.type = MClientResponse$
          $\land\ HandleClientResponse(m.dest, m.src, m)$
          $\land\ Transition$
    $\lor\ \exists m \in messages :$
          $\land\ m.type = MRepairRequest$
          $\land\ HandleRepairRequest(m.dest, m.src, m)$
          $\land\ Transition$
    $\lor\ \exists m \in messages :$
          $\land\ m.type = MRepairResponse$
          $\land\ HandleRepairResponse(m.dest, m.src, m)$
          $\land\ Transition$
    $\lor\ \exists m \in messages :$
          $\land\ m.type = MAbortRequest$
          $\land\ HandleAbortRequest(m.dest, m.src, m)$
          $\land\ Transition$
    $\lor\ \exists m \in messages :$
          $\land\ m.type = MAbortResponse$
          $\land\ HandleAbortResponse(m.dest, m.src, m)$
          $\land\ Transition$
    $\lor\ \exists m \in messages :$
          $\land\ m.type = MViewChangeRequest$
          $\land\ HandleViewChangeRequest(m.dest, m.src, m)$
          $\land\ Transition$
    $\lor\ \exists m \in messages :$
          $\land\ m.type = MViewChangeResponse$
```

```
            $\land\ HandleViewChangeResponse(m.dest,\ m.src,\ m)$
            $\land\ Transition$
        $\lor\ \exists\, m \in messages :$
            $\land\ m.type = MStartViewRequest$
            $\land\ HandleStartViewRequest(m.dest,\ m.src,\ m)$
            $\land\ Transition$
```

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

\ * Modification History
\ * Last modified *Tue Sep* 22 12:13:15 *PDT* 2020 by *jordanhalterman*
\ * Created *Fri Sep* 18 22:45:21 *PDT* 2020 by *jordanhalterman*