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

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 $cWrites$

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 $Replicas : IsQuorum(r)\}$

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

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

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

Messaging helpers

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

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

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

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

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

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

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

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

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

```
HandleWriteResponse(c, r, m) ≜
    ∧ ¬∃ w ∈ cWrites[c] : w.seqNum = m.seqNum
    ∧ ∨ ∧ m.seqNum ∉ DOMAIN cResps[c][r]
        ∧ cResps' = [cResps EXCEPT ![c] = [cResps[c] EXCEPT ![r] = cResps[c][r] @@ (m.seqNum :> m)]]
        ∧ UNCHANGED ⟨cWrites⟩
      ∨ ∧ m.seqNum ∈ DOMAIN cResps[c][r]
        Do not overwrite a response from a newer view
        ∧ cResps[c][r][m.seqNum].viewID ≤ m.viewID
        ∧ cResps' = [cResps EXCEPT ![c] = [cResps[c] EXCEPT ![r] = [cResps[c][r] EXCEPT ![m.seqNum] =
        ∧ LET committed ≜ IsCommitted({cResps'[c][x][m.seqNum] : x ∈ {x ∈ Replicas : m.seqNum ∈ DOM
          IN
            ∨ ∧ committed
              ∧ cWrites' = [cWrites EXCEPT ![c] = cWrites[c] ∪ {m}]
            ∨ ∧ ¬committed
              ∧ UNCHANGED ⟨cWrites⟩
    ∧ Discard(m)
    ∧ UNCHANGED ⟨globalVars, replicaVars, cTime, cSeqNum, cReads⟩

HandleReadResponse(c, r, m) ≜
    ∧ ∨ ∧ m.primary
        ∧ m ∉ cReads[c]
        ∧ cReads' = [cReads EXCEPT ![c] = cReads[c] ∪ {m}]
      ∨ ∧ ¬m.primary
        ∧ UNCHANGED ⟨cReads⟩
    ∧ Discard(m)
    ∧ UNCHANGED ⟨globalVars, replicaVars, cTime, cSeqNum, cResps, cWrites⟩
```

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,
```

$$
\begin{array}{ll}
dest & \mapsto d, \\
type & \mapsto MAbortRequest, \\
viewID & \mapsto rViewID[r], \\
client & \mapsto c, \\
seqNum & \mapsto m.seqNum] : d \in Replicas\})
\end{array}
$$

$$
\begin{array}{l}
HandleWriteRequest(r,\ c,\ m) \triangleq \\
\quad \land\ rStatus[r] = SNormal \\
\quad \land\ \lor\ \land\ m.viewID\ = rViewID[r] \\
\qquad\ \land\ m.seqNum = rSeqNum[r][c] + 1 \\
\qquad\ \land \text{LET}\ entry \triangleq [type \mapsto TValue,\ value \mapsto m.value,\ timestamp \mapsto m.timestamp] \\
\qquad\quad \text{IN}\quad rLog' = [rLog\ \text{EXCEPT}\ ![r] = Append(rLog[r],\ entry)] \\
\qquad\ \land\ rSeqNum' = [rSeqNum\ \text{EXCEPT}\ ![r] = m.seqNum] \\
\qquad\ \land\ Reply(m,\ [src \mapsto r, \\
\qquad\qquad\qquad dest \mapsto c, \\
\qquad\qquad\qquad type \mapsto MWriteResponse, \\
\qquad\qquad\qquad seqNum \mapsto m.seqNum, \\
\qquad\qquad\qquad viewID \mapsto rViewID[r], \\
\qquad\qquad\qquad succeeded \mapsto \text{TRUE}]) \\
\quad\ \lor\ \land\ m.viewID\ = rViewID[r] \\
\qquad\ \land\ m.seqNum > rSeqNum[r][c] + 1 \\
\qquad\ \land\ \lor\ \land\ IsPrimary(r) \\
\qquad\qquad\qquad\ \land\ Abort(r,\ c,\ m) \\
\qquad\qquad\quad \lor\ \land\ \neg IsPrimary(r) \\
\qquad\qquad\qquad\ \land\ Repair(r,\ c,\ m) \\
\qquad\ \land \text{UNCHANGED}\ \langle rLog \rangle \\
\quad\ \lor\ \land\ m.viewID < rViewID[r] \\
\qquad\ \land\ Reply(m,\ [src \mapsto r, \\
\qquad\qquad\qquad dest \mapsto c, \\
\qquad\qquad\qquad type \mapsto MWriteResponse, \\
\qquad\qquad\qquad seqNum \mapsto m.seqNum, \\
\qquad\qquad\qquad viewID \mapsto rViewID[r], \\
\qquad\qquad\qquad succeeded \mapsto \text{FALSE}]) \\
\qquad\ \land \text{UNCHANGED}\ \langle rLog \rangle \\
\quad \land \text{UNCHANGED}\ \langle globalVars,\ clientVars,\ rStatus,\ rViewID,\ rLastView,\ rViewChanges \rangle
\end{array}
$$

$$
\begin{array}{l}
HandleReadRequest(r,\ c,\ m) \triangleq \\
\quad \land\ rStatus[r] = SNormal \\
\quad \land\ Len(rLog[r]) > 0 \\
\quad \land\ Reply(m,\ [src \mapsto r, \\
\qquad\qquad\ dest \mapsto c, \\
\qquad\qquad\ type \mapsto MReadResponse, \\
\qquad\qquad\ viewID \mapsto rViewID[r], \\
\qquad\qquad\ primary \mapsto IsPrimary(r), \\
\qquad\qquad\ index \mapsto Len(rLog[r]),
\end{array}
$$

```
                    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)
    ∧ rStatus[r] = SNormal
    ∧ ∨ ∧ m.seqNum ≤ 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⟩
      ∨ ∧ m.seqNum = Len(rLog[r][m.client]) + 1
        ∧ Abort(r, m.client, m)
    ∧ UNCHANGED ⟨globalVars, clientVars⟩

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

HandleAbortRequest(r, s, m) ≜
    ∧ m.viewID  = rViewID[r]
    ∧ m.seqNum ≤ Len(rLog[r][m.client]) + 1
    ∧ rStatus[r] ∈ {SNormal, SAborting}
    ∧ LET entry ≜ [type ↦ TNoOp]
      IN   rLog' = [rLog EXCEPT ![r] = [rLog[r] EXCEPT ![m.client] = Replace(rLog[r][m.client], m.seqNum,
    ∧ ∨ ∧ 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      ↦ MWriteResponse,
                   viewID    ↦ rViewID[r],
                   seqNum    ↦ m.seqNum,
                   succeeded ↦ FALSE]})
    ∧ UNCHANGED ⟨globalVars, clientVars, rStatus, rViewID, rLastView, rViewChanges⟩
```

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

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

$$
\begin{array}{l}
HandleViewChangeRequest(r,\ s,\ m)\ \triangleq \\
\quad \land\ rViewID[r] < m.viewID \\
\quad \land\ rViewID' = [rViewID \text{ EXCEPT } ![r] = m.viewID] \\
\quad \land\ rStatus' = [rStatus \text{ EXCEPT } ![r] = SViewChange] \\
\quad \land\ rViewChanges' = [rViewChanges \text{ EXCEPT } ![r] = \{\}] \\
\quad \land\ Reply(m, [src \mapsto r, \\
\qquad\qquad dest \mapsto Primary(m.viewID), \\
\qquad\qquad type \mapsto MViewChangeResponse, \\
\qquad\qquad viewID \mapsto m.viewID, \\
\qquad\qquad lastViewID \mapsto rLastView[r], \\
\qquad\qquad logs \mapsto rLog[r]]) \\
\quad \land \text{ UNCHANGED } \langle globalVars,\ clientVars,\ rLog,\ rSeqNum,\ rAbortSeqNum,\ rAbortResps,\ rLastView \rangle
\end{array}
$$

$$
\begin{array}{l}
HandleViewChangeResponse(r,\ s,\ m)\ \triangleq \\
\quad \land\ IsPrimary(r) \\
\quad \land\ rViewID[r] = m.viewID \\
\quad \land\ rStatus[r] = SViewChange \\
\quad \land\ rViewChanges' = [rViewChanges \text{ EXCEPT } ![r] = rViewChanges[r] \cup \{m\}] \\
\quad \land \text{ LET } viewChanges \triangleq \{v \in rViewChanges'[r][m.client] : \land\ v.viewID = rViewID[r]\} \\
\qquad viewSources \triangleq \{v.src : v \in viewChanges\} \\
\qquad isQuorum \triangleq r \in viewSources \land viewSources \in Quorums \\
\qquad lastViews \triangleq \{v.lastViewID : v \in viewChanges\}
\end{array}
$$

```
        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 ≜
    ∧ 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 ≜
```

$$
\begin{aligned}
&\land\ replicas &&= SeqFromSet(Replicas)\\
&\land\ rStatus &&= [r \in Replicas \mapsto SNormal]\\
&\land\ rLog &&= [r \in Replicas \mapsto [c \in Clients \mapsto \langle\rangle]]\\
&\land\ rSeqNum &&= [r \in Replicas \mapsto [c \in Clients \mapsto 0]]\\
&\land\ rAbortSeqNum &&= [r \in Replicas \mapsto [c \in Clients \mapsto 0]]\\
&\land\ rAbortResps &&= [r \in Replicas \mapsto [c \in Clients \mapsto \{\}]]\\
&\land\ rViewID &&= [r \in Replicas \mapsto 1]\\
&\land\ rLastView &&= [r \in Replicas \mapsto 1]\\
&\land\ rViewChanges &&= [r \in Replicas \mapsto \{\}]
\end{aligned}
$$

$$
\begin{aligned}
&Init \triangleq\\
&\quad \land\ InitMessageVars\\
&\quad \land\ InitClientVars\\
&\quad \land\ InitReplicaVars\\
&\quad \land\ transitions = 0
\end{aligned}
$$

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

$$
\begin{aligned}
&Inv \triangleq\\
&\quad \land \forall\, c1,\, c2 \in Clients :\\
&\qquad \neg\exists\, r \in cReads[c1] :\\
&\qquad\quad \exists\, w \in cWrites[c2] :\\
&\qquad\qquad \land\ r.index = w.index\\
&\qquad\qquad \land \neg ChecksumsMatch(r.checksum,\, w.checksum)\\
&\quad \land \forall\, c1,\, c2 \in Clients :\\
&\qquad \neg\exists\, r1 \in cReads[c1] :\\
&\qquad\quad \exists\, r2 \in cReads[c2] :\\
&\qquad\qquad \land\ r1.index = r2.index\\
&\qquad\qquad \land \neg ChecksumsMatch(r1.checksum,\, r2.checksum)
\end{aligned}
$$

$$Transition \triangleq transitions' = transitions + 1$$

$$
\begin{aligned}
&Next \triangleq\\
&\quad \lor \exists\, c \in Clients :\\
&\qquad \land\ Write(c)\\
&\qquad \land\ Transition\\
&\quad \lor \exists\, c \in Clients :\\
&\qquad \land\ Read(c)\\
&\qquad \land\ Transition\\
&\quad \lor \exists\, r \in Replicas :\\
&\qquad \land\ ChangeView(r)\\
&\qquad \land\ Transition\\
&\quad \lor \exists\, m \in messages :\\
&\qquad \land\ m.type = MWriteRequest\\
&\qquad \land\ HandleWriteRequest(m.dest,\, m.src,\, m)
\end{aligned}
$$

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

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

\ * Modification History
\ * Last modified *Tue Sep* 22 03:38:33 *PDT* 2020 by *jordanhalterman*
\ * Created *Fri Sep* 18 22:45:21 *PDT* 2020 by *jordanhalterman*