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 statuses
CONSTANTS
SNormal,
SAborting,
SViewChange

Entry types
CONSTANTS
TValue,
TNoOp

A sequence of replicas used for deterministic primary election
VARIABLE *replicas*

$globalVars \triangleq \langle replicas \rangle$

The set of all messages on the network
VARIABLE *messages*

$messageVars \triangleq \langle messages \rangle$

Local client state

Strictly increasing representation of synchronized time
VARIABLE $cTime$

The highest known view *ID* for a client
VARIABLE $cViewID$

The current sequence number for a client
VARIABLE $cSeqNum$

A client response buffer
VARIABLE $cResps$

A set of all commits - used for model checking
VARIABLE $cCommits$

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

Local replica state

The current status of a replica
VARIABLE $rStatus$

A replica's commit *log*
VARIABLE $rLog$

The current view *ID* for a replica
VARIABLE $rViewID$

The current sequence number for each session
VARIABLE $rSeqNum$

The highest known timestamp for all sessions
VARIABLE $rTimestamp$

The last known normal view
VARIABLE $rLastViewID$

The set of received view change responses
VARIABLE $rViewChanges$

The point (*client* + sequence number) in the *log* currently being aborted
VARIABLE $rAbortPoint$

The set of abort responses received
VARIABLE $rAbortResps$

$replicaVars \triangleq \langle rStatus,\ rLog,\ rViewID,\ rSeqNum,\ rTimestamp,$

$rLastViewID,\ rViewChanges,\ rAbortPoint,\ rAbortResps\rangle$

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

This section provides helpers for the spec.

```
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
```

This section models the network.

Send a set of messages

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

Send a message

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

Reply to a message with a set of responses

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

Reply to a message

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

Discard a message

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

This section models client requests.

Client 'c' sends value 'v' to all replicas

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

Client 'c' handles a response 'm' from replica 'r'

$$
\begin{array}{l}
HandleClientResponse(c,\ r,\ m) \triangleq \\
\quad \land\ \lor\ \land\ m.viewID = cViewID[c] \\
\qquad\quad \land\ cResps' = [cResps \text{ EXCEPT } ![c] = cResps[c] \cup \{m\}] \\
\qquad\quad \land \text{ LET} \\
\qquad\qquad\quad seqNumResps \triangleq \{n \in cResps[c] : n.seqNum = m.seqNum\} \\
\qquad\qquad\quad goodResps \triangleq \{n \in seqNumResps : n.viewID = cViewID[c] \land n.succeeded\} \\
\qquad\qquad\quad isCommitted \triangleq \land \exists\, n \in goodResps : n.src = Primary(n.viewID) \\
\qquad\qquad\qquad\qquad\qquad\quad \land \{n.src : n \in goodResps\} \in Quorums \\
\qquad\qquad \text{IN} \\
\qquad\qquad\quad \land\ \lor\ \land\ isCommitted \\
\qquad\qquad\qquad\qquad \land\ cCommits' = [cCommits \text{ EXCEPT } ![c] = cCommits[c] \cup \\
\qquad\qquad\qquad\qquad\qquad\quad \{\text{CHOOSE } n \in goodResps : n.src = Primary(n.viewID)\}] \\
\qquad\qquad\qquad\quad\ \lor\ \land\ \neg isCommitted \\
\qquad\qquad\qquad\qquad \land \text{ UNCHANGED } \langle cCommits\rangle \\
\qquad\qquad\quad \land \text{ UNCHANGED } \langle cViewID,\ cSeqNum\rangle \\
\qquad\quad \lor\ \land\ m.viewID > cViewID[c] \\
\qquad\qquad \land\ cViewID' = [cViewID \text{ EXCEPT } ![c] = m.viewID] \\
\qquad\qquad \land\ cSeqNum' = [cSeqNum \text{ EXCEPT } ![c] = 0] \\
\qquad\qquad \land\ cResps' = [cResps \text{ EXCEPT } ![c] = \{\}] \\
\qquad\qquad \land \text{ UNCHANGED } \langle cCommits\rangle \\
\qquad\quad \lor\ \land\ m.viewID < cViewID[c] \\
\qquad\qquad \land \text{ UNCHANGED } \langle cCommits\rangle \\
\quad \land\ Discard(m) \\
\quad \land \text{ UNCHANGED } \langle globalVars,\ replicaVars,\ cTime,\ cSeqNum\rangle
\end{array}
$$

This section models the replica protocol.

Replica 'r' requests a repair of the client 'c' request 'm'

$$
\begin{array}{l}
Repair(r,\ c,\ m) \triangleq \\
\quad \land\ Replies(m,\ \{[src \mapsto r,\ dest \mapsto d,\ type \mapsto MRepairRequest,\ viewID \mapsto rViewID[r],\ client \mapsto c,\ seqNum \mapsto m.seqNum,\ timestamp \mapsto m.timestamp] : d \in Replicas\})
\end{array}
$$

Replica 'r' aborts the client 'c' request 'm'

$$
\begin{array}{l}
Abort(r,\ c,\ m) \triangleq \\
\quad \land\ IsPrimary(r) \\
\quad \land\ rStatus[r] = SNormal \\
\quad \land\ rStatus' = [rStatus \ \text{EXCEPT}\ ![r] = SAborting] \\
\quad \land\ rAbortResps' = [rAbortResps \ \text{EXCEPT}\ ![r] = \{\}] \\
\quad \land\ rAbortPoint' = [rAbortPoint \ \text{EXCEPT}\ ![r] = [client \mapsto c,\ seqNum \mapsto m.seqNum]] \\
\quad \land\ Replies(m,\ \{[src \mapsto r,\ dest \mapsto d,\ type \mapsto MAbortRequest,\ viewID \mapsto rViewID[r],\ client \mapsto c,\ seqNum \mapsto m.seqNum,\ timestamp \mapsto m.timestamp] : d \in Replicas\})
\end{array}
$$

Replica 'r' handles client 'c' request 'm'

$$
\begin{array}{l}
HandleClientRequest(r,\ c,\ m) \triangleq \\
\quad \land\ rStatus[r] = SNormal \\
\quad \land\ \lor\ \land\ m.viewID = rViewID[r] \\
\qquad \land\ \text{LET} \\
\qquad\quad lastIndex \triangleq Sum(\{Len(rLog[r][i]) : i \in Clients\}) \\
\qquad\quad index \triangleq lastIndex + 1 \\
\qquad\quad lastTimestamp \triangleq rTimestamp[r] \\
\qquad\quad isSequential \triangleq m.seqNum = rSeqNum[r][c] + 1 \\
\qquad\quad isLinear \triangleq m.timestamp > lastTimestamp \\
\qquad\quad entry \triangleq [type \mapsto TValue,\ index \mapsto index,\ value \mapsto m.value,\ timestamp \mapsto m.timestamp] \\
\qquad\quad append(e) \triangleq [rLog \ \text{EXCEPT}\ ![r] = [rLog[r] \ \text{EXCEPT}\ ![c] = Append(rLog[r][c],\ e)]] \\
\qquad \text{IN} \\
\qquad\quad \lor\ \land\ isSequential
\end{array}
$$

$\land$ *isLinear*
$\land$ $rLog'$ $= append(entry)$
$\land$ $rSeqNum'$ $= [rSeqNum$ EXCEPT $![r] = [rSeqNum[r]$ EXCEPT $![c] = m.seqNum]]$
$\land$ $rTimestamp' = [rTimestamp$ EXCEPT $![r] = m.timestamp]$
$\land$ $Reply(m, [src \mapsto r,$
$dest \mapsto c,$
$type \mapsto MClientResponse,$
$viewID \mapsto rViewID[r],$
$seqNum \mapsto m.seqNum,$
$index \mapsto index,$
$value \mapsto m.value,$
$succeeded \mapsto$ TRUE$])$
$\land$ UNCHANGED $\langle rStatus, rAbortPoint, rAbortResps\rangle$
$\lor \land \lor \land \neg isSequential$
$\land m.seqNum > rSeqNum[r][c] + 1$
$\lor \neg isLinear$
$\land \lor \land IsPrimary(r)$
$\land Abort(r, c, m)$
$\lor \land \neg IsPrimary(r)$
$\land Repair(r, c, m)$
$\land$ UNCHANGED $\langle rStatus, rAbortPoint, rAbortResps\rangle$
$\land$ UNCHANGED $\langle rLog, rSeqNum, rTimestamp\rangle$
$\lor \land m.viewID < rViewID[r]$
$\land Reply(m, [src \mapsto r,$
$dest \mapsto c,$
$type \mapsto MClientResponse,$
$viewID \mapsto rViewID[r],$
$seqNum \mapsto m.seqNum,$
$succeeded \mapsto$ FALSE$])$
$\land$ UNCHANGED $\langle rStatus, rLog, rSeqNum, rTimestamp, rAbortPoint, rAbortResps\rangle$
$\land$ UNCHANGED $\langle globalVars, clientVars, rViewID, rLastViewID, rViewChanges\rangle$

Replica 'r' handles replica 's' repair request 'm'

$HandleRepairRequest(r, s, m) \triangleq$
$\land m.viewID = rViewID[r]$
$\land IsPrimary(r)$
$\land rStatus[r] = SNormal$
$\land$ LET $offset \triangleq Len(rLog[r][m.client]) - (rSeqNum[r][m.client] - m.seqNum)$
IN
$\lor \land offset \leq Len(rLog[r][m.client])$
$\land Reply(m, [src \mapsto r,$
$dest \mapsto s,$
$type \mapsto MRepairResponse,$
$viewID \mapsto rViewID[r],$
$client \mapsto m.client,$

```
                                    seqNum    ↦ m.seqNum,
                                    value     ↦ rLog[r][m.client][offset].value,
                                    timestamp ↦ rLog[r][m.client][offset].timestamp])
                  ∧ UNCHANGED ⟨rStatus, rAbortPoint, rAbortResps⟩
               ∨ ∧ offset = Len(rLog[r][m.client]) + 1
                  ∧ Abort(r, m.client, m)
      ∧ UNCHANGED ⟨globalVars, clientVars, rLog, rSeqNum, rTimestamp, rViewID, rLastViewID, rViewChan

 Replica 'r' handles replica 's' repair response 'm'
HandleRepairResponse(r, s, m) ≜
      ∧ HandleClientRequest(r, m.client, [m EXCEPT !.src = m.client])

 Replica 'r' handles replica 's' abort request 'm'
HandleAbortRequest(r, s, m) ≜
      ∧ m.viewID = rViewID[r]
      ∧ rStatus[r] ∈ {SNormal, SAborting}
      ∧ LET
            offset ≜ Len(rLog[r][m.client]) − (rSeqNum[r][m.client] − m.seqNum)
            entry ≜ [type ↦ TNoOp, timestamp ↦ m.timestamp]
            replace(l, i, e) ≜ [j ∈ 1 .. Max({Len(l), i}) ↦ IF j = i THEN e ELSE l[j]]
        IN
            ∧ offset ≤ Len(rLog[r][m.client]) + 1
            ∧ rLog′ = [rLog EXCEPT ![r] = [rLog[r] EXCEPT
                                    ![m.client] = replace(rLog[r][m.client], offset, entry)]]
            ∧ rTimestamp′ = [rTimestamp EXCEPT ![r] = Max({rTimestamp[r], m.timestamp})]
            ∧ rSeqNum′ = [rSeqNum EXCEPT ![r] = [rSeqNum[r] EXCEPT
                                                ![m.client] = Max({rSeqNum[r][m.client], m.seqNum})]]
            ∧ 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, rViewChanges⟩

 Replica 'r' handles replica 's' repair response 'm'
HandleAbortResponse(r, s, m) ≜
      ∧ rStatus[r] = SAborting
      ∧ m.viewID = rViewID[r]
```

$\land\ IsPrimary(r)$
$\land\ m.seqNum = rAbortPoint[r].seqNum$
$\land\ rAbortResps' = [rAbortResps \text{ EXCEPT } ![r] = rAbortResps[r] \cup \{m\}]$
$\land$ LET $resps \triangleq \{res.src : res \in \{resp \in rAbortResps'[r] :$
$\land\ resp.viewID = rViewID[r]$
$\land\ resp.client = rAbortPoint[r].client$
$\land\ resp.seqNum = rAbortPoint[r].seqNum\}\}$
$isQuorum \triangleq r \in resps \land resps \in Quorums$
IN
$\lor\ \land\ isQuorum$
$\land\ rStatus' = [rStatus \text{ EXCEPT } ![r] = SNormal]$
$\lor\ \land\ \neg isQuorum$
$\land$ UNCHANGED $\langle rStatus \rangle$
$\land$ UNCHANGED $\langle globalVars, messageVars, clientVars, rLog, rSeqNum, rTimestamp,$
$rAbortPoint, rViewID, rViewChanges, rLastViewID \rangle$

Replica 'r' requests a view change

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

Replica 'r' handles replica 's' view change request 'm'

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

Replica 'r' handles replica 's' view change response 'm'

$HandleViewChangeResponse(r, s, m) \triangleq$
$\land\ IsPrimary(r)$
$\land\ rViewID[r] = m.viewID$
$\land\ rStatus[r] = SViewChange$
$\land\ rViewChanges' = [rViewChanges \text{ EXCEPT } ![r] = rViewChanges[r] \cup \{m\}]$
$\land$ LET $viewChanges \triangleq \{v \in rViewChanges'[r] : v.viewID = rViewID[r]\}$

```
        viewSources      ≜ {v.src : v ∈ viewChanges}
        isQuorum         ≜ r ∈ viewSources ∧ viewSources ∈ Quorums
        lastViewIDs      ≜ {v.lastViewID : v ∈ viewChanges}
        lastViewID       ≜ (CHOOSE v1 ∈ lastViewIDs : ∀ v2 ∈ lastViewIDs : v2 ≤ v1)
        lastViewChanges  ≜ {v2 ∈ viewChanges : v2.lastViewID = lastViewID}
        viewLogs         ≜ [c ∈ Clients ↦ {v1.logs[c] : v1 ∈ lastViewChanges}]
        mergeEnts(es)    ≜
            IF es = {} ∨ ∃ e ∈ es : e.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 ∈ Clients}})
                           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, rAbortResps, rLastViewID⟩
```

Replica 'r' handles replica 's' start view request 'm'

```
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)
```

$\land$ UNCHANGED $\langle globalVars,\ clientVars,\ rAbortPoint,\ rAbortResps,\ rViewChanges\rangle$

$InitMessageVars \triangleq$
$\land\ messages = \{\}$

$InitClientVars \triangleq$
$\land\ cTime = 0$
$\land\ cViewID = [c \in Clients \mapsto 1]$
$\land\ cSeqNum = [c \in Clients \mapsto 0]$
$\land\ cResps = [c \in Clients \mapsto \{\}]$
$\land\ cCommits = [c \in Clients \mapsto \{\}]$

$InitReplicaVars \triangleq$
$\land\ replicas = SeqFromSet(Replicas)$
$\land\ rStatus = [r \in Replicas \mapsto SNormal]$
$\land\ rLog = [r \in Replicas \mapsto [c \in Clients \mapsto \langle\rangle]]$
$\land\ rSeqNum = [r \in Replicas \mapsto [c \in Clients \mapsto 0]]$
$\land\ rTimestamp = [r \in Replicas \mapsto 0]$
$\land\ rAbortPoint = [r \in Replicas \mapsto [client \mapsto Nil,\ seqNum \mapsto 0]]$
$\land\ rAbortResps = [r \in Replicas \mapsto \{\}]$
$\land\ rViewID = [r \in Replicas \mapsto 1]$
$\land\ rLastViewID = [r \in Replicas \mapsto 1]$
$\land\ rViewChanges = [r \in Replicas \mapsto \{\}]$

$Init \triangleq$
$\land\ InitMessageVars$
$\land\ InitClientVars$
$\land\ InitReplicaVars$

The type invariant verifies that clients do not receive two commits at the same index with different values.

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

$Next \triangleq$
$\lor\ \exists\, c \in Clients :$
$\exists\, v \in Values :$
$\land\ ClientRequest(c,\ v)$
$\lor\ \exists\, r \in Replicas :$

$$
\begin{array}{l}
\quad \land \mathit{ChangeView}(r) \\
\lor \exists\, m \in \mathit{messages} : \\
\quad \land m.\mathit{type} = \mathit{MClientRequest} \\
\quad \land \mathit{HandleClientRequest}(m.\mathit{dest},\ m.\mathit{src},\ m) \\
\lor \exists\, m \in \mathit{messages} : \\
\quad \land m.\mathit{type} = \mathit{MClientResponse} \\
\quad \land \mathit{HandleClientResponse}(m.\mathit{dest},\ m.\mathit{src},\ m) \\
\lor \exists\, m \in \mathit{messages} : \\
\quad \land m.\mathit{type} = \mathit{MRepairRequest} \\
\quad \land \mathit{HandleRepairRequest}(m.\mathit{dest},\ m.\mathit{src},\ m) \\
\lor \exists\, m \in \mathit{messages} : \\
\quad \land m.\mathit{type} = \mathit{MRepairResponse} \\
\quad \land \mathit{HandleRepairResponse}(m.\mathit{dest},\ m.\mathit{src},\ m) \\
\lor \exists\, m \in \mathit{messages} : \\
\quad \land m.\mathit{type} = \mathit{MAbortRequest} \\
\quad \land \mathit{HandleAbortRequest}(m.\mathit{dest},\ m.\mathit{src},\ m) \\
\lor \exists\, m \in \mathit{messages} : \\
\quad \land m.\mathit{type} = \mathit{MAbortResponse} \\
\quad \land \mathit{HandleAbortResponse}(m.\mathit{dest},\ m.\mathit{src},\ m) \\
\lor \exists\, m \in \mathit{messages} : \\
\quad \land m.\mathit{type} = \mathit{MViewChangeRequest} \\
\quad \land \mathit{HandleViewChangeRequest}(m.\mathit{dest},\ m.\mathit{src},\ m) \\
\lor \exists\, m \in \mathit{messages} : \\
\quad \land m.\mathit{type} = \mathit{MViewChangeResponse} \\
\quad \land \mathit{HandleViewChangeResponse}(m.\mathit{dest},\ m.\mathit{src},\ m) \\
\lor \exists\, m \in \mathit{messages} : \\
\quad \land m.\mathit{type} = \mathit{MStartViewRequest} \\
\quad \land \mathit{HandleStartViewRequest}(m.\mathit{dest},\ m.\mathit{src},\ m) \\
\lor \exists\, m \in \mathit{messages} : \\
\quad \land \mathit{Discard}(m) \\
\quad \land \text{UNCHANGED } \langle \mathit{globalVars},\ \mathit{clientVars},\ \mathit{replicaVars} \rangle
\end{array}
$$

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

* Modification History
* Last modified *Tue Sep* 22 14:18:02 *PDT* 2020 by *jordanhalterman*
* Created *Fri Sep* 18 22:45:21 *PDT* 2020 by *jordanhalterman*