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 $rAbortResps$

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

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

A counter used to limit the state space for model checking

VARIABLE $transitions$

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

This section provides helpers for the spec.

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 $Replicas : IsQuorum(r)\}$

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

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

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'

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

$\land$ UNCHANGED $\langle cCommits \rangle$
$\land$ $Discard(m)$
$\land$ UNCHANGED $\langle globalVars,\ replicaVars,\ cTime,\ cSeqNum \rangle$

This section models the replica protocol.

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

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

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

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

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

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

```
                                                    ![c] = Append(rLog[r][c], e)]]
    IN
        ∨ ∧ isSequential
          ∧ isLinear
          ∧ rLog' = append(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,
                      value     ↦ m.value,
                      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    ↦ rViewID[r],
                          seqNum    ↦ m.seqNum,
                          succeeded ↦ FALSE])
              ∧ UNCHANGED ⟨rStatus, rAbortPoint, rAbortResps⟩
          ∧ UNCHANGED ⟨rLog, rSeqNum, rTimestamp⟩
    ∨ ∧ m.viewID < rViewID[r]
      ∧ Reply(m, [src       ↦ r,
                  dest      ↦ c,
                  type      ↦ MClientResponse,
                  viewID    ↦ rViewID[r],
                  seqNum    ↦ m.seqNum,
                  succeeded ↦ FALSE])
      ∧ UNCHANGED ⟨rStatus, rLog, rSeqNum, rTimestamp, rAbortPoint, rAbortResps⟩
    ∧ UNCHANGED ⟨globalVars, clientVars, rViewID, rLastViewID, rViewChanges⟩

Replica 'r' handles replica 's' repair request 'm'
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, rAbortPoint, rAbortResps⟩
          ∨ ∧ index = Len(rLog[r][m.client]) + 1
            ∧ Abort(r, m.client, m)
    ∧ UNCHANGED ⟨globalVars, clientVars⟩

 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]) + 1 − (rSeqNum[r][m.client] − m.seqNum)
        entry ≜ [type ↦ TNoOp, timestamp ↦ m.timestamp]
        replace(i, e) ≜ [j ∈ 1 .. Max({Len(rLog[r][m.client]), i}) ↦
                          IF j = i THEN e ELSE rLog[r][m.client][j]]
      IN
        ∧ offset ≤ Len(rLog[r][m.client]) + 1
        ∧ rLog′ = replace(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,
```

$rAbortResps,\ rViewID,\ rLastViewID,\ rViewChanges\rangle$

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

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

Replica 'r' requests a view change

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

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

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

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

```
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)
          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]
```

$\land\ rSeqNum' = [rSeqNum \text{ EXCEPT } ![r] = [c \in Clients \mapsto 0]]$
$\land\ rTimestamp' = [rTimestamp \text{ EXCEPT } ![r] = m.timestamp]$
$\land\ rStatus' = [rStatus \text{ EXCEPT } ![r] = SNormal]$
$\land\ rViewID' = [rViewID \text{ EXCEPT } ![r] = m.viewID]$
$\land\ rLastViewID' = [rLastViewID \text{ EXCEPT } ![r] = m.viewID]$
$\land\ Discard(m)$
$\land\ \text{UNCHANGED } \langle globalVars, clientVars, rAbortPoint, rAbortResps, rViewChanges\rangle$

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

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

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

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

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

$TypeOK \triangleq$
$\forall\, c1, c2 \in Clients :$
$\forall\, e1 \in cCommits[c1] :$
$\neg\exists\, e2 \in cCommits[c2] :$

$\land\ e1.index = e2.index$
$\land\ e1.value \neq e2.value$

$Transition \triangleq transitions' = transitions + 1$

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

$$
\begin{array}{l}
\quad \lor \exists\, m \in messages : \\
\qquad \land\, m.type = MStartViewRequest \\
\qquad \land\, HandleStartViewRequest(m.dest,\, m.src,\, m) \\
\qquad \land\, Transition \\
\quad \lor \exists\, m \in messages : Discard(m)
\end{array}
$$

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

\ * Modification History
\ * Last modified *Tue Sep* 22 12:57:49 *PDT* 2020 by *jordanhalterman*
\ * Created *Fri Sep* 18 22:45:21 *PDT* 2020 by *jordanhalterman*