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⟩
```

A counter used to limit the state space for model checking

```
VARIABLE transitions

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

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) ≜ messages′ = messages ∪ ms
```

Send a message

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

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

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

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

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

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

```
$HandleClientRequest(r,\ c,\ m) \triangleq$
    $\land$ $rStatus[r] = SNormal$
    $\land$ $\lor$ $\land$ $m.viewID = rViewID[r]$
           $\land$ LET
                $lastIndex \triangleq Sum(\{Len(rLog[r][i]) : i \in Clients\})$
                $index \triangleq lastIndex + 1$
                $lastTimestamp \triangleq rTimestamp[r]$
                $isSequential \triangleq m.seqNum = rSeqNum[r][c] + 1$
                $isLinear \triangleq m.timestamp > lastTimestamp$
                $entry \triangleq [type \mapsto TValue,$
                              $index \mapsto index,$
                              $value \mapsto m.value,$
                              $timestamp \mapsto m.timestamp]$
```

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

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

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

```
$HandleAbortResponse(r, s, m) \triangleq$
    $\land rStatus[r] = SAborting$
    $\land m.viewID = rViewID[r]$
    $\land IsPrimary(r)$
    $\land m.seqNum = rAbortPoint[r].seqNum$
    $\land rAbortResps' = [rAbortResps \text{ EXCEPT } ![r] = rAbortResps[r] \cup \{m\}]$
    $\land$ 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

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

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

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

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

```
$HandleViewChangeResponse(r, s, m) \triangleq$
    $\land IsPrimary(r)$
```

```
$\land rViewID[r] \quad = m.viewID$
$\land rStatus[r] \quad = SViewChange$
$\land rViewChanges' = [rViewChanges \text{ EXCEPT } ![r] = rViewChanges[r] \cup \{m\}]$
$\land \text{LET } viewChanges \triangleq \{v \in rViewChanges'[r] : v.viewID = rViewID[r]\}$
        $viewSources \triangleq \{v.src : v \in viewChanges\}$
        $isQuorum \triangleq r \in viewSources \land viewSources \in Quorums$
        $lastViewIDs \triangleq \{v.lastViewID : v \in viewChanges\}$
        $lastViewID \triangleq (\text{CHOOSE } v1 \in lastViewIDs : \forall v2 \in lastViewIDs : v2 \leq v1)$
        $lastViewChanges \triangleq \{v2 \in viewChanges : v2.lastViewID = lastViewID\}$
        $viewLogs \triangleq [c \in Clients \mapsto \{v1.logs[c] : v1 \in lastViewChanges\}]$
        $mergeEnts(es) \triangleq$
            $\text{IF } es = \{\} \lor \exists e \in es : e.type = TNoOp \text{ THEN}$
                $[type \mapsto TNoOp]$
             $\text{ELSE}$
                $\text{CHOOSE } e \in es : e.type \neq TNoOp$
        $range(ls) \triangleq Max(\{Len(l) : l \in ls\})$
        $entries(ls, i) \triangleq \{l[i] : l \in \{k \in ls : i \leq Len(k)\}\}$
        $mergeLogs(ls) \triangleq [i \in 1 \,..\, range(ls) \mapsto mergeEnts(entries(ls, i))]$
        $viewLog \triangleq [c \in Clients \mapsto mergeLogs(viewLogs[c])]$
        $viewRange \triangleq Max(\{Len(viewLog[c]) : c \in Clients\})$
        $viewTimestamp \triangleq \text{IF } viewRange > 0 \text{ THEN}$
                              $Max(\text{UNION } \{\{l[i].timestamp : i \in \text{DOMAIN } l\} :$
                                          $l \in \{viewLog[c] : c \in Clients\}\})$
                          $\text{ELSE } 0$
  $\text{IN}$
      $\lor \land isQuorum$
         $\land Replies(m, \{[src \mapsto r,$
                           $dest \mapsto d,$
                           $type \mapsto MStartViewRequest,$
                           $viewID \mapsto rViewID[r],$
                           $timestamp \mapsto viewTimestamp,$
                           $log \mapsto viewLog] : d \in Replicas\})$
      $\lor \land \neg isQuorum$
         $\land Discard(m)$
$\land \text{UNCHANGED } \langle globalVars, clientVars, rStatus, rViewID, rLog, rSeqNum,$
                   $rTimestamp, rAbortPoint, rAbortResps, rLastViewID\rangle$
```

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

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

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

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

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

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

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

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

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

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

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

\ * Modification History
\ * Last modified *Tue Sep* 22 13:34:27 *PDT* 2020 by *jordanhalterman*
\ * Created *Fri Sep* 18 22:45:21 *PDT* 2020 by *jordanhalterman*