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,
MClientReply,
MRepairRequest,
MRepairReply,
MAbortRequest,
MAbortReply,
MViewChangeRequest,
MViewChangeReply,
MStartViewRequest

Replica statuses
CONSTANTS
SNormal,
SAborting,
SViewChange

Entry types
CONSTANTS
TValue,
TNoOp

Message schemas

$ViewIDs \triangleq [viewID \mapsto n \in (1 \,..\,)]$

ClientRequest
$[src \mapsto c \in Clients,$
$dest \mapsto r \in Replicas,$
$type \mapsto MClientRequest,$
$viewID \mapsto i \in ViewIDs,$
$value \mapsto v \in Values,$
$seqNum \mapsto s \in (1 \,..\,),$

$timestamp \mapsto t \in (1 \mathinner{..})]$

ClientReply
$[src \mapsto r \in Replicas,$
$dest \mapsto c \in Clients,$
$type \mapsto MClientReply,$
$viewID \mapsto i \in ViewIDs,$
$value \mapsto v \in Values,$
$seqNum \mapsto s \in (1 \mathinner{..}),$
$index \mapsto i \in (1 \mathinner{..}),$
$succeeded \mapsto \text{TRUE} \lor \text{FALSE}]$

RepairRequest
$[src \mapsto r \in Replicas,$
$dest \mapsto r \in Replicas,$
$type \mapsto MRepairRequest,$
$viewID \mapsto i \in ViewIDs,$
$client \mapsto c \in Clients,$
$seqNum \mapsto s \in (1 \mathinner{..}),$
$timestamp \mapsto t \in (1 \mathinner{..})]$

RepairReply
$[src \mapsto r \in Replicas,$
$dest \mapsto r \in Replicas,$
$type \mapsto MRepairReply,$
$viewID \mapsto i \in ViewIDs,$
$client \mapsto c \in Clients,$
$seqNum \mapsto s \in (1 \mathinner{..}),$
$value \mapsto v \in Values,$
$timestamp \mapsto t \in (1 \mathinner{..})]$

AbortRequest
$[src \mapsto r \in Replicas,$
$dest \mapsto r \in Replicas,$
$type \mapsto MAbortRequest,$
$viewID \mapsto i \in ViewIDs,$
$client \mapsto c \in Clients,$
$seqNum \mapsto s \in (1 \mathinner{..}),$
$timestamp \mapsto t \in (1 \mathinner{..})]$

AbortReply
$[src \mapsto r \in Replicas,$
$dest \mapsto r \in Replicas,$
$type \mapsto MAbortReply,$
$viewID \mapsto i \in ViewIDs,$
$client \mapsto c \in Clients,$
$seqNum \mapsto s \in (1 \mathinner{..})]$

ViewChangeRequest
$[src \mapsto r \in Replicas,$
$dest \mapsto r \in Replicas,$
$type \mapsto MViewChangeRequest,$
$viewID \mapsto i \in ViewIDs]$

$$
\begin{array}{l}
\textit{ViewChangeReply} \\
\quad [\textit{src} \quad \mapsto r \in \textit{Replicas}, \\
\quad\ \textit{dest} \quad \mapsto r \in \textit{Replicas}, \\
\quad\ \textit{type} \quad \mapsto \textit{MViewChangeReply}, \\
\quad\ \textit{viewID} \quad \mapsto i \in \textit{ViewIDs}, \\
\quad\ \textit{lastViewID} \mapsto i \in (1\,..\,), \\
\quad\ \textit{log} \quad \mapsto [c \in \textit{Clients} \mapsto [\, i \in 1\,..\,(1\,..\,) \mapsto [\\
\qquad\qquad\qquad \textit{value} \mapsto v \in \textit{Values}, \textit{timestamp} \mapsto (1\,..\,)]]]] \\
\\
\textit{StartViewRequest} \\
\quad [\textit{src} \quad \mapsto r \in \textit{Replicas}, \\
\quad\ \textit{dest} \quad \mapsto r \in \textit{Replicas}, \\
\quad\ \textit{type} \quad \mapsto \textit{MStartViewRequest}, \\
\quad\ \textit{viewID} \mapsto i \in \textit{ViewIDs}, \\
\quad\ \textit{timestamp} \mapsto t \in (1\,..\,), \textit{log} \quad \mapsto [c \in \textit{Clients} \mapsto [\\
\qquad\qquad i \in 1\,..\,(1\,..\,) \mapsto [\, \textit{value} \mapsto v \in \textit{Values}, \textit{timestamp} \mapsto (1\,..\,)]]]]
\end{array}
$$

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 $messages$

The total number of messages sent

VARIABLE $messageCount$

$messageVars \triangleq \langle messages, messageCount \rangle$

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 $cReps$

A set of all commits - used for model checking

VARIABLE $cCommits$

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

Local replica state

The current status of a replica
VARIABLE $rStatus$

The current view ID for a replica
VARIABLE $rViewID$

A replica's commit log
VARIABLE $rLog$

The current log index for a replica
VARIABLE $rIndex$

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 $rAbortReps$

$$replicaVars \triangleq \langle rStatus,\ rViewID,\ rLog,\ rIndex,\ rSeqNum,\ rTimestamp,\ rLastViewID,\ rViewChanges,\ rAbortPoint,\ rAbortReps\rangle$$

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

This section provides helpers for the spec.

Creates a sequence from set 'S'

$$\begin{array}{l}\text{RECURSIVE } SeqFromSet(\_)\\ SeqFromSet(S) \triangleq\\ \quad \text{IF } S = \{\} \text{ THEN}\\ \qquad \langle\rangle\\ \quad \text{ELSE LET } x \triangleq \text{CHOOSE } x \in S : \text{TRUE}\\ \qquad \text{IN } \langle x\rangle \circ SeqFromSet(S \setminus \{x\})\end{array}$$

Selects an element of set 'S'

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

```
RECURSIVE SetReduce(_, _, _)
SetReduce(Op(_, _), S, value) ≜
    IF S = {} THEN
        value
     ELSE
        LET s ≜ Pick(S)
        IN    SetReduce(Op, S \ {s}, Op(s, value))

 Computes the greatest vlue in set 'S'
Max(S) ≜ CHOOSE x ∈ S : ∀ y ∈ S : x ≥ y

 Computes the sum of numbers in set 'S'
Sum(S) ≜ LET _op(a, b) ≜ a + b
            IN    SetReduce(_op, S, 0)

 A boolean indicating whether the given set is a quorum
IsQuorum(s) ≜ Cardinality(s) * 2 ≥ Cardinality(Replicas)

 The set of all quorums
Quorums ≜ {r ∈ SUBSET Replicas : IsQuorum(r)}

 The primary for view 'v'
Primary(v) ≜ replicas[(v%Len(replicas)) + (IF v ≥ Len(replicas) THEN 1 ELSE 0)]

 A boolean indicating whether replica 'r' is the primary for the current view
IsPrimary(r) ≜ Primary(rViewID[r]) = r
```

```
This section models the network.

 Send a set of messages
Sends(ms) ≜
    ∧ messages'      = messages ∪ ms
    ∧ messageCount' = messageCount + Cardinality(ms)

 Send a message
Send(m) ≜ Sends({m})

 Reply to a message with a set of responses
Replies(req, reps) ≜
    ∧  messages'      = (messages ∪ reps) \ {req}
    ∧  messageCount' = messageCount + Cardinality(reps)

 Reply to a message
Reply(req, resp) ≜ Replies(req, {resp})

 Discard a message
Discard(m) ≜
    ∧   messages'      = messages \ {m}
```

$\land \quad messageCount' = messageCount + 1$

This section models client requests.

Client 'c' sends value 'v' to all replicas
Client requests are ordered globally using physical timestamps and locally (within the client) using client sequence numbers. Sequence numbers are sequential and unique within each view.
When the client sends a request is generates a new timestamp. Physical timestamps are modeled here as a strictly increasing global clock simulating synchronized system clocks. The sequence number for the client is also incremented and sent with the request.

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

Client 'c' handles a response 'm' from replica 'r'
When a response is received by the client, if the client is still in the request view it can process the response. The client is responsible for determining commitment by counting responses for each sequence number. Once a response is received from a majority of the replicas including the primary replica, the response is committed. Committed responses are stored in a history variable for checking against invariants.

$HandleClientReply(c,\ r,\ m) \triangleq$
$\land\ \lor\ \land\ m.viewID = cViewID[c]$
$\land\ cReps' = [cReps \text{ EXCEPT } ![c] = cReps[c] \cup \{m\}]$
$\land$ LET
$okReps \triangleq \{n \in cReps'[c] : n.seqNum = m.seqNum \land n.succeeded\}$
$hasPrimary \triangleq \exists\, n \in okReps : Primary(n.viewID) = n.src$
IN
IF $hasPrimary$ THEN
LET
$primaryRep \triangleq$ CHOOSE $n \in okReps : n.src = Primary(n.viewID)$
$matchReps \triangleq \{n \in okReps : n.index = primaryRep.index\}$
$isCommitted \triangleq \{n.src : n \in matchReps\} \in Quorums$
IN
IF $isCommitted$ THEN
$cCommits' = [cCommits \text{ EXCEPT } ![c] = cCommits[c] \cup \{primaryRep\}]$

```
                    ELSE
                        UNCHANGED ⟨cCommits⟩
                ELSE
                    UNCHANGED ⟨cCommits⟩
            ∧ UNCHANGED ⟨cViewID, cSeqNum⟩
        ∨ ∧ m.viewID > cViewID[c]
            ∧ cViewID' = [cViewID EXCEPT ![c] = m.viewID]
            ∧ cSeqNum' = [cSeqNum EXCEPT ![c] = 0]
            ∧ cReps'   = [cReps   EXCEPT ![c] = {}]
            ∧ UNCHANGED ⟨cCommits⟩
        ∨ ∧ m.viewID < cViewID[c]
            ∧ UNCHANGED ⟨cViewID, cSeqNum, cReps, cCommits⟩
    ∧ Discard(m)
    ∧ UNCHANGED ⟨globalVars, replicaVars, cTime⟩
```

This section models the replica protocol.

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

```
Repair(r, c, m) ≜
    ∧ Replies(m, {[src       ↦ r,
                   dest      ↦ d,
                   type      ↦ MRepairRequest,
                   viewID    ↦ rViewID[r],
                   client    ↦ c,
                   seqNum    ↦ m.seqNum,
                   timestamp ↦ m.timestamp] : d ∈ Replicas})
```

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

```
Abort(r, c, m) ≜
    ∧ IsPrimary(r)
    ∧ rStatus[r]     = SNormal
    ∧ rStatus'       = [rStatus      EXCEPT ![r] = SAborting]
    ∧ rAbortReps'    = [rAbortReps   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})
```

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

Client requests with a view *ID* not matching the replica's view are rejected.

Clients reset their sequence number of

For requests in the correct view, the request must be sequential and linear
to be appended to the *log*. That is, the request must have a 'seqNum' that is
$1+$ the prior 'seqNum' for the client, and the 'timestamp' must be greater
than all prior timestamps in the *log*. This is necessary to ensure the primary
log does not change when requests are reordered. The client can retry requests
that are reordered with a new sequence number and timestamp.
To maintain consistency within the *log*, a separate sequence is maintained for
each session (client), and each sequence number is assigned to a unique
position in the session *log*. Session logs are logically merged into a totally
ordered *log* using the request timestamps.
When a sequence number is skipped, the primary must commit a *TNoOp* entry to
the *log*. It does so by running the *AbortRequest* protocol.
When a sequence number is skipped on a non-primary replica, the replica attempts
to recover the request using the *RepairRequest* protocol.

$$
\begin{array}{l}
HandleClientRequest(r,\ c,\ m) \triangleq \\
\quad \land\ rStatus[r] = SNormal \\
\quad \land\ \lor\ \land\ m.viewID = rViewID[r] \\
\qquad \land\ \text{LET} \\
\qquad\quad index \triangleq rIndex[r] + 1 \\
\qquad\quad lastTimestamp \triangleq rTimestamp[r] \\
\qquad\quad isSequential \triangleq m.seqNum = rSeqNum[r][c] + 1 \\
\qquad\quad isLinear \triangleq m.timestamp > lastTimestamp \\
\qquad\quad entry \triangleq [type \mapsto TValue, \\
\qquad\qquad\qquad\qquad index \mapsto index, \\
\qquad\qquad\qquad\qquad value \mapsto m.value, \\
\qquad\qquad\qquad\qquad timestamp \mapsto m.timestamp] \\
\qquad\quad append(e) \triangleq [rLog\ \text{EXCEPT}\ ![r] = [rLog[r]\ \text{EXCEPT} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![c] = Append(rLog[r][c],\ e)]] \\
\qquad \text{IN} \\
\qquad\quad \lor\ \land\ isSequential \\
\qquad\qquad\ \land\ isLinear \\
\qquad\qquad\ \land\ rLog' = append(entry) \\
\qquad\qquad\ \land\ rIndex' = [rIndex\ \text{EXCEPT}\ ![r] = index] \\
\qquad\qquad\ \land\ rSeqNum' = [rSeqNum\ \text{EXCEPT}\ ![r] = \\
\qquad\qquad\qquad\qquad\qquad\qquad [rSeqNum[r]\ \text{EXCEPT}\ ![c] = m.seqNum]] \\
\qquad\qquad\ \land\ rTimestamp' = [rTimestamp\ \text{EXCEPT}\ ![r] = m.timestamp] \\
\qquad\qquad\ \land\ Reply(m, [src \mapsto r, \\
\qquad\qquad\qquad\qquad\qquad dest \mapsto c, \\
\qquad\qquad\qquad\qquad\qquad type \mapsto MClientReply, \\
\qquad\qquad\qquad\qquad\qquad viewID \mapsto rViewID[r], \\
\qquad\qquad\qquad\qquad\qquad seqNum \mapsto m.seqNum, \\
\qquad\qquad\qquad\qquad\qquad index \mapsto index, \\
\qquad\qquad\qquad\qquad\qquad timestamp \mapsto m.timestamp, \\
\qquad\qquad\qquad\qquad\qquad value \mapsto m.value, \\
\qquad\qquad\qquad\qquad\qquad succeeded \mapsto \text{TRUE}])
\end{array}
$$

```
                    ∧ UNCHANGED ⟨rStatus, rAbortPoint, rAbortReps⟩
                ∨ ∧ ∨ ∧ ¬isSequential
                      ∧ m.seqNum > rSeqNum[r][c] + 1
                  ∨ ¬isLinear
                  ∧ ∨ ∧ IsPrimary(r)
                      ∧ Abort(r, c, m)
                    ∨ ∧ ¬IsPrimary(r)
                      ∧ Repair(r, c, m)
                      ∧ UNCHANGED ⟨rStatus, rAbortPoint, rAbortReps⟩
                  ∧ UNCHANGED ⟨rLog, rIndex, rSeqNum, rTimestamp⟩
        ∨ ∧ m.viewID < rViewID[r]
          ∧ Reply(m, [src       ↦ r,
                      dest      ↦ c,
                      type      ↦ MClientReply,
                      viewID    ↦ rViewID[r],
                      seqNum    ↦ m.seqNum,
                      succeeded ↦ FALSE])
          ∧ UNCHANGED ⟨rStatus, rLog, rIndex, rSeqNum, rTimestamp, rAbortPoint, rAbortReps⟩
    ∧ UNCHANGED ⟨globalVars, clientVars, rViewID, rLastViewID, rViewChanges⟩

Replica 'r' handles replica 's' repair request 'm'
When a repair request is received, if the requested sequence number is in the session
log, the entry is returned. Otherwise, the primary aborts the request.
HandleRepairRequest(r, s, m) ≜
    ∧ m.viewID = rViewID[r]
    ∧ IsPrimary(r)
    ∧ rStatus[r] = SNormal
    ∧ LET offset ≜ Len(rLog[r][m.client]) − (rSeqNum[r][m.client] − m.seqNum)
      IN
        ∨ ∧ offset ≤ Len(rLog[r][m.client])
          ∧ Reply(m, [src       ↦ r,
                      dest      ↦ s,
                      type      ↦ MRepairReply,
                      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, rAbortReps⟩
        ∨ ∧ offset = Len(rLog[r][m.client]) + 1
          ∧ Abort(r, m.client, m)
    ∧ UNCHANGED ⟨globalVars, clientVars, rLog, rIndex, rSeqNum, rTimestamp, rViewID, rLastViewID, rV

Replica 'r' handles replica 's' repair response 'm'
Repair responses are handled like client requests.
```

$HandleRepairReply(r,\ s,\ m) \triangleq$
$\quad HandleClientRequest(r,\ m.client,\ [m \text{ EXCEPT } !.src = m.client])$

Replica 'r' handles replica 's' abort request 'm'
If the aborted sequence number is in the session *log*, the entry is replaced with
a no-op entry. If the sequence number can be appebded to the *log*, it is.

$HandleAbortRequest(r,\ s,\ m) \triangleq$
$\quad \land\ m.viewID = rViewID[r]$
$\quad \land\ rStatus[r] \in \{SNormal,\ SAborting\}$
$\quad \land$ LET
$\qquad offset \triangleq Len(rLog[r][m.client]) - (rSeqNum[r][m.client] - m.seqNum)$
$\qquad entry \triangleq [type \mapsto TNoOp,\ value \mapsto Nil,\ timestamp \mapsto 0]$
$\quad$ IN
$\qquad \land\ \lor\ \land\ offset \leq Len(rLog[r][m.client])$
$\qquad\qquad \land\ rLog' = [rLog \text{ EXCEPT } ![r] = [$
$\qquad\qquad\qquad rLog[r] \text{ EXCEPT } ![m.client] = [$
$\qquad\qquad\qquad rLog[r][m.client] \text{ EXCEPT } ![offset] = [$
$\qquad\qquad\qquad rLog[r][m.client][offset] \text{ EXCEPT } !.type = TNoOp]]]]$
$\qquad\qquad \land$ UNCHANGED $\langle rTimestamp,\ rSeqNum\rangle$
$\qquad\qquad \lor\ \land\ offset = Len(rLog[r][m.client]) + 1$
$\qquad\qquad\qquad \land\ rLog' = [rLog \text{ EXCEPT } ![r] = [$
$\qquad\qquad\qquad\qquad rLog[r] \text{ EXCEPT } ![m.client] =$
$\qquad\qquad\qquad\qquad\quad Append(rLog[r][m.client],\ entry)]]$
$\qquad\qquad\qquad \land\ rTimestamp' = [rTimestamp \text{ EXCEPT } ![r] = Max(\{rTimestamp[r],\ m.timestamp\})]$
$\qquad\qquad\qquad \land\ rSeqNum' = [rSeqNum \text{ EXCEPT } ![r] = [$
$\qquad\qquad\qquad\qquad rSeqNum[r] \text{ EXCEPT } ![m.client] = m.seqNum]]$
$\qquad \land\ Replies(m,\ \{[src \mapsto r,$
$\qquad\qquad dest \mapsto Primary(rViewID[r]),$
$\qquad\qquad type \mapsto MAbortReply,$
$\qquad\qquad viewID \mapsto rViewID[r],$
$\qquad\qquad client \mapsto m.client,$
$\qquad\qquad seqNum \mapsto m.seqNum],$
$\qquad\qquad [src \mapsto r,$
$\qquad\qquad dest \mapsto m.client,$
$\qquad\qquad type \mapsto MClientReply,$
$\qquad\qquad viewID \mapsto rViewID[r],$
$\qquad\qquad seqNum \mapsto m.seqNum,$
$\qquad\qquad succeeded \mapsto \text{FALSE}]\})$
$\quad \land$ UNCHANGED $\langle globalVars,\ clientVars,\ rStatus,\ rAbortPoint,$
$\qquad\qquad rAbortReps,\ rViewID,\ rLastViewID,\ rViewChanges\rangle$

Replica 'r' handles replica 's' repair response 'm'
$HandleAbortReply(r,\ s,\ m) \triangleq$
$\quad \land\ rStatus[r] = SAborting$
$\quad \land\ m.viewID = rViewID[r]$

```
    ∧ IsPrimary(r)
    ∧ m.seqNum = rAbortPoint[r].seqNum
    ∧ rAbortReps' = [rAbortReps EXCEPT ![r] = rAbortReps[r] ∪ {m}]
    ∧ LET reps ≜ {res.src : res ∈ {resp ∈ rAbortReps'[r] :
                        ∧ resp.viewID  = rViewID[r]
                        ∧ resp.client  = rAbortPoint[r].client
                        ∧ resp.seqNum = rAbortPoint[r].seqNum}}
          isQuorum ≜ r ∈ reps ∧ reps ∈ Quorums
      IN
        ∨ ∧ isQuorum
          ∧ rStatus' = [rStatus EXCEPT ![r] = SNormal]
        ∨ ∧ ¬isQuorum
          ∧ UNCHANGED ⟨rStatus⟩
    ∧ UNCHANGED ⟨globalVars, messageVars, clientVars, rLog, rIndex, rSeqNum,
                 rTimestamp, rAbortPoint, rViewID, rViewChanges, rLastViewID⟩
```

Replica 'r' requests a view change
The view change is requested by sending a *ViewChangeRequest* to each replica.

```
ChangeView(r) ≜
    ∧ Sends({[src    ↦ r,
              dest   ↦ d,
              type   ↦ MViewChangeRequest,
              viewID ↦ rViewID[r] + 1] : d ∈ Replicas})
    ∧ UNCHANGED ⟨globalVars, clientVars, replicaVars⟩
```

Replica 'r' handles replica 's' view change request 'm'
Replicas respond to *ViewChangeRequests* with the contents of their logs
for reconciliation. When a new view change is requested, the replica updates
its *ViewID* and transitions to the *ViewChange* status to block writes during
the transition.

```
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       ↦ MViewChangeReply,
                viewID     ↦ m.viewID,
                lastViewID ↦ rLastViewID[r],
                log        ↦ rLog[r]])
    ∧ UNCHANGED ⟨globalVars, clientVars, rLog, rIndex, rSeqNum,
                 rTimestamp, rAbortPoint, rAbortReps, rLastViewID⟩
```

Replica 'r' handles replica 's' view change response 'm'
ViewChangeReplys are handled by the primary for the new view. Once

responses are received from a majority of the replicas including the new primary, the logs received from each replica are merged together to form the *log* for the new view. For each known session, the logs from each replica are merged by comparing each entry and keeping all non-empty sequential entries in the quorum. An updated timestamp is calculated from the reconciled *log*, and a *StartViewRequest* containing the new logs is sent to each replica.

$$
\begin{array}{l}
HandleViewChangeReply(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] : v.viewID = rViewID[r]\} \\
\qquad\quad viewSources \triangleq \{v.src : v \in viewChanges\} \\
\qquad\quad isQuorum \triangleq r \in viewSources \land viewSources \in Quorums \\
\qquad\quad lastViewIDs \triangleq \{v.lastViewID : v \in viewChanges\} \\
\qquad\quad lastViewID \triangleq (\text{CHOOSE } v1 \in lastViewIDs : \forall\, v2 \in lastViewIDs : v2 \leq v1) \\
\qquad\quad lastViewChanges \triangleq \{v2 \in viewChanges : v2.lastViewID = lastViewID\} \\
\qquad\quad viewLogs \triangleq [c \in Clients \mapsto \{v1.log[c] : v1 \in lastViewChanges\}] \\
\qquad\quad mergeEnts(es) \triangleq \\
\qquad\qquad \text{IF } es = \{\} \lor \exists\, e \in es : e.type = TNoOp \text{ THEN} \\
\qquad\qquad\quad \text{CHOOSE } e \in es : e.type = TNoOp \\
\qquad\qquad \text{ELSE} \\
\qquad\qquad\quad \text{CHOOSE } e \in es : e.type \neq TNoOp \land e.index = Max(\{f.index : f \in es\}) \\
\qquad\quad range(ls) \triangleq Max(\{Len(l) : l \in ls\}) \\
\qquad\quad entries(ls,\ i) \triangleq \{l[i] : l \in \{k \in ls : i \leq Len(k)\}\} \\
\qquad\quad mergeLogs(ls) \triangleq [i \in 1\,..\,range(ls) \mapsto mergeEnts(entries(ls,\ i))] \\
\qquad\quad viewLog \triangleq [c \in Clients \mapsto mergeLogs(viewLogs[c])] \\
\qquad\quad viewRange \triangleq Max(\{Len(viewLog[c]) : c \in Clients\}) \\
\qquad\quad liveEntries(l) \triangleq \{l[i] : i \in \{i \in \text{DOMAIN } l : l[i].type \neq TNoOp\}\} \\
\qquad\quad viewIndex \triangleq Sum(\{Cardinality(liveEntries(viewLog[c])) : c \in Clients\}) \\
\qquad\quad viewTimestamp \triangleq \text{IF } viewRange > 0 \text{ THEN} \\
\qquad\qquad\qquad\qquad Max(\text{UNION } \{\{l[i].timestamp : i \in \text{DOMAIN } l\} : \\
\qquad\qquad\qquad\qquad\qquad\qquad l \in \{viewLog[c] : c \in Clients\}\}) \\
\qquad\qquad\qquad\quad \text{ELSE } 0 \\
\text{IN} \\
\qquad \lor\ \land\ isQuorum \\
\qquad\quad \land\ Replies(m, \{[src \mapsto r, \\
\qquad\qquad\qquad\qquad dest \mapsto d, \\
\qquad\qquad\qquad\qquad type \mapsto MStartViewRequest, \\
\qquad\qquad\qquad\qquad viewID \mapsto rViewID[r], \\
\qquad\qquad\qquad\qquad index \mapsto viewIndex, \\
\qquad\qquad\qquad\qquad timestamp \mapsto viewTimestamp, \\
\qquad\qquad\qquad\qquad log \mapsto viewLog] : d \in Replicas\}) \\
\qquad \lor\ \land\ \neg isQuorum \\
\qquad\quad \land\ Discard(m)
\end{array}
$$

$\land$ UNCHANGED $\langle globalVars,\ clientVars,\ rStatus,\ rViewID,\ rLog,\ rIndex,$
$rSeqNum,\ rTimestamp,\ rAbortPoint,\ rAbortReps,\ rLastViewID\rangle$

Replica 'r' handles replica 's' start view request 'm'
If the view is new, the replica updates its logs and session state from the request.

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

$InitMessageVars \triangleq$
$\land\ messages = \{\}$
$\land\ messageCount = 0$

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

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

$Init \triangleq$

$\land$ *InitMessageVars*
$\land$ *InitClientVars*
$\land$ *InitReplicaVars*

This section specifies the invariants for the protocol.

Merge client logs together

$$
\begin{array}{l}
\mathit{MergeLogs}(l) \triangleq \\
\quad \text{LET } \mathit{entries} \triangleq \text{UNION } \{\{l[i][j] : j \in \text{DOMAIN } l[i]\} : i \in \text{DOMAIN } l\} \\
\quad \text{IN } \quad [i \in 1 \,..\, \mathit{Cardinality}(\mathit{entries}) \mapsto \text{CHOOSE } e \in \mathit{entries} : e.\mathit{index} = i]
\end{array}
$$

The type invariant asserts that the leader's *log* will never contain a different value at the same index as a client commit.

$$
\begin{array}{l}
\mathit{TypeOK} \triangleq \\
\quad \forall\, c \in \mathit{Clients} : \\
\quad\quad \forall\, e \in \mathit{cCommits}[c] : \\
\quad\quad\quad \neg\exists\, r \in \mathit{Replicas} : \\
\quad\quad\quad\quad \land\ \mathit{Primary}(\mathit{rViewID}[r]) = r \\
\quad\quad\quad\quad \land\ \mathit{rStatus}[r] = \mathit{SNormal} \\
\quad\quad\quad\quad \land\ \text{LET } \mathit{logs} \triangleq \mathit{MergeLogs}(\mathit{rLog}[r]) \\
\quad\quad\quad\quad\quad \text{IN} \\
\quad\quad\quad\quad\quad\quad \land\ \mathit{Len}(\mathit{logs}) \geq e.\mathit{index} \\
\quad\quad\quad\quad\quad\quad \land\ \mathit{logs}[e.\mathit{index}].\mathit{value} \neq e.\mathit{value}
\end{array}
$$

$$
\begin{array}{l}
\mathit{NextClientRequest} \triangleq \\
\quad \exists\, c \in \mathit{Clients} : \\
\quad\quad \exists\, v \in \mathit{Values} : \\
\quad\quad\quad \mathit{ClientRequest}(c,\, v)
\end{array}
$$

$$
\begin{array}{l}
\mathit{NextChangeView} \triangleq \\
\quad \exists\, r \in \mathit{Replicas} : \\
\quad\quad \mathit{ChangeView}(r)
\end{array}
$$

$$
\begin{array}{l}
\mathit{NextHandleClientRequest} \triangleq \\
\quad \exists\, m \in \mathit{messages} : \\
\quad\quad \land\ m.\mathit{type} = \mathit{MClientRequest} \\
\quad\quad \land\ \mathit{HandleClientRequest}(m.\mathit{dest},\, m.\mathit{src},\, m)
\end{array}
$$

$$
\begin{array}{l}
\mathit{NextHandleClientReply} \triangleq \\
\quad \exists\, m \in \mathit{messages} : \\
\quad\quad \land\ m.\mathit{type} = \mathit{MClientReply} \\
\quad\quad \land\ \mathit{HandleClientReply}(m.\mathit{dest},\, m.\mathit{src},\, m)
\end{array}
$$

$$
\begin{array}{l}
\mathit{NextHandleRepairRequest} \triangleq \\
\quad \exists\, m \in \mathit{messages} :
\end{array}
$$

$\land\ m.type = MRepairRequest$
$\land\ HandleRepairRequest(m.dest,\ m.src,\ m)$

$NextHandleRepairReply\ \triangleq$
$\exists\, m \in messages :$
$\land\ m.type = MRepairReply$
$\land\ HandleRepairReply(m.dest,\ m.src,\ m)$

$NextHandleAbortRequest\ \triangleq$
$\exists\, m \in messages :$
$\land\ m.type = MAbortRequest$
$\land\ HandleAbortRequest(m.dest,\ m.src,\ m)$

$NextHandleAbortReply\ \triangleq$
$\exists\, m \in messages :$
$\land\ m.type = MAbortReply$
$\land\ HandleAbortReply(m.dest,\ m.src,\ m)$

$NextHandleViewChangeRequest\ \triangleq$
$\exists\, m \in messages :$
$\land\ m.type = MViewChangeRequest$
$\land\ HandleViewChangeRequest(m.dest,\ m.src,\ m)$

$NextHandleViewChangeReply\ \triangleq$
$\exists\, m \in messages :$
$\land\ m.type = MViewChangeReply$
$\land\ HandleViewChangeReply(m.dest,\ m.src,\ m)$

$NextHandleStartViewRequest\ \triangleq$
$\exists\, m \in messages :$
$\land\ m.type = MStartViewRequest$
$\land\ HandleStartViewRequest(m.dest,\ m.src,\ m)$

$NextDropMessage\ \triangleq$
$\exists\, m \in messages :$
$\land\ Discard(m)$
$\land\ \text{UNCHANGED}\ \langle globalVars,\ clientVars,\ replicaVars\rangle$

$Next\ \triangleq$
$\lor\ NextClientRequest$
$\lor\ NextChangeView$
$\lor\ NextHandleClientRequest$
$\lor\ NextHandleClientReply$
$\lor\ NextHandleRepairRequest$
$\lor\ NextHandleRepairReply$
$\lor\ NextHandleAbortRequest$
$\lor\ NextHandleAbortReply$

$\lor$ *NextHandleViewChangeRequest*
$\lor$ *NextHandleViewChangeReply*
$\lor$ *NextHandleStartViewRequest*
$\lor$ *NextDropMessage*

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

\ * Modification History
\ * Last modified *Mon Sep* 28 17:46:34 *PDT* 2020 by *jordanhalterman*
\ * Created *Fri Sep* 18 22:45:21 *PDT* 2020 by *jordanhalterman*