MODULE *ParallelRaftCE*

EXTENDS *Integers*, *FiniteSets*, *Sequences*, *TLC*, *Naturals*

CONSTANTS *Server*, *Value*, *Nil*

CONSTANTS *Follower*, *Candidate*, *LeaderCandidate*, *Leader*

CONSTANTS *RequestVoteRequest*, *RequestVoteResponse*,
RequestCommitRequest, *RequestCommitResponse*,
RequestSyncRequest, *RequestSyncResponse*,
UpdateSyncRequest, *UpdateSyncResponse*

VARIABLE *messages*,
currentTerm,
currentState,
votedFor,
sync,
endPoint

$serverVars \triangleq \langle currentTerm, currentState, votedFor, sync, endPoint \rangle$

VARIABLE *log*
$logVars \triangleq log$

VARIABLE *syncTrack*
$leaderVars \triangleq syncTrack$

VARIABLES *halfElections*,
elections
$electionVars \triangleq \langle halfElections, elections \rangle$

VARIABLES *allLogs*,
allEntries,
allSynced

$vars \triangleq \langle messages, serverVars, logVars, leaderVars, electionVars, allLogs, allEntries, allSynced \rangle$

$Quorums \triangleq \{i \in \text{SUBSET } Server : Cardinality(i) * 2 > Cardinality(Server)\}$

$Send(m) \triangleq messages' = messages \cup \{m\}$

$Index \triangleq Nat$
$Term \triangleq Nat$

$Min(s) \triangleq \text{IF } s = \{\} \text{ THEN } -1$
$\text{ELSE CHOOSE } i \in s : \forall j \in s : i \leq j$

$$Max(s) \triangleq \text{IF } s = \{\} \text{ THEN } -1 \text{ ELSE CHOOSE } i \in s : \forall j \in s : i \geq j$$

$$\begin{aligned}
InitServerVars \triangleq\ & \wedge currentTerm = [i \in Server \mapsto 0] \\
& \wedge sync = [i \in Server \mapsto 0] \\
& \wedge currentState = [i \in Server \mapsto Follower] \\
& \wedge endPoint = [i \in Server \mapsto [n \in Term \mapsto \langle -1, -1 \rangle]] \\
& \wedge votedFor = [i \in Server \mapsto Nil]
\end{aligned}$$

$$\begin{aligned}
InitLogVars \triangleq\ & log = [i \in Server \mapsto [n \in Index \mapsto [term \mapsto -1, \\
& \qquad\qquad\qquad\qquad\qquad\qquad date \mapsto -1, \\
& \qquad\qquad\qquad\qquad\qquad\qquad value \mapsto Nil, \\
& \qquad\qquad\qquad\qquad\qquad\qquad committed \mapsto \text{FALSE}]]]
\end{aligned}$$

$$InitLeaderVars \triangleq syncTrack = [i \in Server \mapsto [j \in Server \mapsto 0]]$$

$$\begin{aligned}
InitHistoryVars \triangleq\ & \wedge halfElections = \{\} \\
& \wedge elections = \{\} \\
& \wedge allLogs = \{\} \\
& \wedge allEntries = \{\} \\
& \wedge allSynced = \{\}
\end{aligned}$$

$$\begin{aligned}
Init \triangleq\ & \wedge messages = \{\} \\
& \wedge InitServerVars \\
& \wedge InitLogVars \\
& \wedge InitLeaderVars \\
& \wedge InitHistoryVars
\end{aligned}$$

$$Entries \triangleq [term : Term \cup \{-1\},\ date : Term \cup \{-1\},\ value : Value \cup \{Nil\},\ committed : \{\text{TRUE}, \text{FALSE}\}]$$

$$logTail(s) \triangleq Max(\{i \in Index : s[i].term \neq -1\})$$

$$\begin{aligned}
TypeSafety \triangleq\ & \wedge allLogs \in \text{SUBSET } (\text{SUBSET } allEntries) \\
& \wedge currentTerm \in [Server \to Nat] \\
& \wedge currentState \in [Server \to \{Follower, Candidate, LeaderCandidate, Leader\}] \\
& \wedge votedFor \in [Server \to Server \cup \{Nil\}] \\
& \wedge sync \in [Server \to Nat \cup \{-1\}] \\
& \wedge endPoint \in [Server \to [Term \to ((Term \cup \{-1\}) \times (Index \cup \{-1\}))]] \\
& \wedge log \in [Server \to [Index \to [term : Index \cup \{-1\}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad date : Term \cup \{-1\}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad value : Value \cup \{Nil\}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad committed : \{\text{TRUE}, \text{FALSE}\}]]] \\
& \wedge syncTrack \in [Server \to [Server \to Nat]] \\
& \wedge halfElections \subseteq [eterm : Nat, \\
& \qquad\qquad\qquad\qquad\quad eleaderCandidate : Server, \\
& \qquad\qquad\qquad\qquad\quad esync : Nat, \\
& \qquad\qquad\qquad\qquad\quad evotes : Quorums,
\end{aligned}$$

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad\qquad\qquad elog : [Index \to Entries]] \\
\quad \land\ elections \subseteq [eterm : Term, \\
\qquad\qquad\qquad\qquad\qquad\qquad esync : Term, \\
\qquad\qquad\qquad\qquad\qquad\qquad eleader : Server, \\
\qquad\qquad\qquad\qquad\qquad\qquad evotes : Quorums, \\
\qquad\qquad\qquad\qquad\qquad\qquad evoterLog : \text{SUBSET } [Index \to Entries], \\
\qquad\qquad\qquad\qquad\qquad\qquad elog : [Index \to Entries]]
\end{array}
$$

$$
\begin{array}{l}
Restart(i) \triangleq \\
\quad \land\ currentState' = [currentState \text{ EXCEPT } ![i] = Follower] \\
\quad \land\ syncTrack' = [syncTrack \text{ EXCEPT } ![i] = [j \in Server \mapsto 0]] \\
\quad \land \text{UNCHANGED } \langle messages, currentTerm, votedFor, sync, endPoint, log, electionVars, allSynced \rangle
\end{array}
$$

$$
\begin{array}{l}
Timeout(i) \triangleq \\
\quad \land\ currentState[i] \in \{Follower, Candidate\} \\
\quad \land\ currentState' = [currentState \text{ EXCEPT } ![i] = Candidate] \\
\quad \land\ currentTerm' = [currentTerm \text{ EXCEPT } ![i] = currentTerm[i] + 1] \\
\quad \land\ (currentTerm[i] + 1) \in Term \\
\quad \land\ votedFor' = [votedFor \text{ EXCEPT } ![i] = Nil] \\
\quad \land \text{UNCHANGED } \langle messages, sync, endPoint, logVars, leaderVars, electionVars, allSynced \rangle
\end{array}
$$

$$
\begin{array}{l}
UpdateTerm(i) \triangleq \\
\quad \land \exists\, m \in messages : \\
\qquad \land\ m.mterm > currentTerm[i] \\
\qquad \land\ \lor\ m.mdest = i \\
\qquad\quad \lor\ m.mdest = Nil \\
\qquad \land\ currentState' = [currentState \text{ EXCEPT } ![i] = Follower] \\
\qquad \land\ currentTerm' = [currentTerm \text{ EXCEPT } ![i] = m.mterm] \\
\qquad \land\ votedFor' = [votedFor \text{ EXCEPT } ![i] = Nil] \\
\quad \land \text{UNCHANGED } \langle messages, sync, logVars, leaderVars, electionVars, allSynced, endPoint \rangle
\end{array}
$$

$$
\begin{array}{l}
RequestVote(i) \triangleq \\
\quad \land\ \ currentState[i] = Candidate \\
\quad \land\ \ Send([mtype \mapsto RequestVoteRequest, \\
\qquad\qquad\quad mterm \mapsto currentTerm[i], \\
\qquad\qquad\quad msync \mapsto sync[i], \\
\qquad\qquad\quad msource \mapsto i, \\
\qquad\qquad\quad mdest \mapsto Nil]) \\
\quad \land\ \ \text{UNCHANGED } \langle serverVars, leaderVars, logVars, electionVars, allSynced \rangle
\end{array}
$$

$$
\begin{array}{l}
HandleRequestVoteRequest(i) \triangleq \\
\quad \land \exists\, m \in messages : \\
\qquad \text{LET } j \triangleq m.msource \\
\qquad\quad syncOK \triangleq m.msync \geq sync[i] \\
\qquad\quad grant \triangleq \land\ syncOK
\end{array}
$$

```
                    ∧ votedFor[i] ∈ {Nil, j}
                    ∧ currentTerm[i] = m.mterm
    IN    ∧ m.mterm ≤ currentTerm[i]
          ∧ m.mtype  = RequestVoteRequest
          ∧ ∨ ∧ grant
              ∧ votedFor' = [votedFor EXCEPT ![i] = j]
            ∨ ∧ ¬grant
              ∧ UNCHANGED votedFor
          ∧ Send([mtype       ↦ RequestVoteResponse,
                  mterm       ↦ currentTerm[i],
                  mvoteGranted ↦ grant,
                  mlog        ↦ LET C ≜ {n ∈ Index : log[i][n].term = sync[i]}
                                IN  {⟨n, log[i][n]⟩ : n ∈ C},
                  mend        ↦ endPoint[i][m.msync],
                  msource     ↦ i,
                  mdest       ↦ j])
    ∧ UNCHANGED ⟨currentTerm, currentState, sync, logVars, leaderVars, electionVars, allSynced, endPo

Merge(entries, term, date) ≜ IF entries = {} THEN [term       ↦ term,
                                                   date       ↦ date,
                                                   value      ↦ Nil,
                                                   committed ↦ FALSE]
                           ELSE LET committed ≜ {e ∈ entries : e.committed = TRUE}
                                    chosen ≜ CASE committed =  {} → CHOOSE x ∈ entries :
                                                                    ∀ y ∈ entries : x.date ≥ y.
                                              □      committed ≠  {} → CHOOSE x ∈ committed : TRU
                                IN  [term       ↦ chosen.term,
                                     date       ↦ date,
                                     value      ↦ chosen.value,
                                     committed ↦ chosen.committed]

BecomeLeaderCandidate(i) ≜
    ∧ currentState[i] = Candidate
    ∧ ∃ P, Q ∈ Quorums :
        LET voteResponded ≜ {m ∈ messages : ∧ m.mtype = RequestVoteResponse
                                            ∧ m.mdest = i
                                            ∧ m.msource ∈ P
                                            ∧ m.mterm = currentTerm[i]}
            voteGranted   ≜ {m ∈ voteResponded : ∧ m.mvoteGranted = TRUE
                                                 ∧ m.msource ∈ Q}
            allLog        ≜ UNION {m.mlog : m ∈ voteResponded}
            end           ≜ LET allPoint ≜ {m.mend : m ∈ voteResponded}        end, endPoint
                                e        ≜ CHOOSE e1 ∈ allPoint : ∀ e2 ∈ allPoint : e1[1] ≥ e2[1]
                            IN  IF e[1] = −1 THEN Max({e1[1] : e1 ∈ allLog})
                                             ELSE e[2]
```

$$
\begin{array}{ll}
& toRecover \triangleq \{n \in 0 \,..\, end : log[i][n].committed = \text{FALSE}\} \\
& toSync \triangleq \{\langle n,\, Merge(\{l[2] : l \in \{t \in allLog : t[1] = n\}\},\, sync[1],\, currentTerm[i])\rangle : n \\
\text{IN} & \wedge \forall\, q \in Q : \exists\, m \in voteGranted : m.msource = q \\
& \wedge log' = [log \text{ EXCEPT } ![i] = \text{IF } end = -1 \text{ THEN } [n \in Index \mapsto \text{IF } log[i][n].term = sync[i] \text{ THE}
\end{array}
$$

$$
\begin{array}{r}
\text{ELS} \\
\text{ELSE } [n \in Index \mapsto \text{IF } n \in toRecover \text{ THEN } (\text{CHOC} \\
\text{ELSE IF } n > end \text{ THEN } [term \\
date \\
valu \\
com \\
\text{ELSE } log[i]
\end{array}
$$

$$
\begin{array}{ll}
& \wedge endPoint' = [endPoint \text{ EXCEPT } ![i][sync[i]] = \langle currentTerm[i],\, end\rangle] \\
& \wedge halfElections' = halfElections \cup \{[eterm \mapsto currentTerm[i], \\
& \qquad eleaderCandidate \mapsto i, \\
& \qquad esync \mapsto sync[i], \\
& \qquad evotes \mapsto Q, \\
& \qquad elog \mapsto log[i]]\} \\
& \wedge currentState' = [currentState \text{ EXCEPT } ![i] = LeaderCandidate] \\
& \wedge syncTrack' = [syncTrack \text{ EXCEPT } ![i] = [j \in Server \mapsto sync[i]]] \text{ here} \\
& \wedge \text{UNCHANGED } \langle messages,\, currentTerm,\, votedFor,\, sync,\, elections,\, allSynced\rangle
\end{array}
$$

$$
\begin{array}{ll}
RequestSync(i) \triangleq & \\
\quad \wedge & currentState[i] \in \{LeaderCandidate,\, Leader\} \\
\quad \wedge & \exists\, s \in 0 \,..\, sync[i] : \\
& \quad \text{LET } start \triangleq Min(\{n \in Index : log[i][n].term = s\}) \\
& \qquad end \triangleq Max(\{n \in Index : log[i][n].term = s\}) \qquad \text{here} \\
& \quad \text{IN } \wedge Send([mtype \mapsto RequestSyncRequest, \\
& \qquad\qquad mterm \mapsto currentTerm[i], \\
& \qquad\qquad msync \mapsto s, \\
& \qquad\qquad mstart \mapsto start, \\
& \qquad\qquad mend \mapsto end, \\
& \qquad\qquad mentries \mapsto \text{IF } start = -1 \text{ THEN } Nil \\
& \qquad\qquad\qquad\qquad \text{ELSE } [n \in start \,..\, end \mapsto log[i][n]], \\
& \qquad\qquad msource \mapsto i, \\
& \qquad\qquad mdest \mapsto Nil]) \qquad \text{here} \\
\quad \wedge & \text{UNCHANGED } \langle serverVars,\, logVars,\, electionVars,\, syncTrack,\, allSynced\rangle
\end{array}
$$

$$
\begin{array}{ll}
HandleRequestSyncRequest(i) \triangleq & \\
\quad \wedge \exists\, m \in messages : & \\
\quad \text{LET } j \triangleq m.msource & \\
\qquad grant \triangleq \wedge m.mterm = currentTerm[i] &
\end{array}
$$

```
                    ∧ m.msync = sync[i]
    IN  ∧ m.mtype = RequestSyncRequest
        ∧ m.mterm ≤ currentTerm[i]
        ∧ j ≠ i
        ∧ ∨ ∧ grant
            ∧ log' = [log EXCEPT ![i] = IF m.mstart = −1 THEN [n ∈ Index ↦ IF log[i][n].term = s

                                                             ELSE [n ∈ Index ↦ IF n < m.mstart TH
                                                                                   ELSE IF n ∈ m.ms
                                                                                   ELSE [term ↦ −1
                                                                                         date ↦ −1
                                                                                         value ↦ Nil
                                                                                         committed ↦
            ∧ endPoint' = [endPoint EXCEPT ![i][sync[i]] = ⟨currentTerm[i], m.mend⟩]
          ∨ ∧ ¬grant
            ∧ UNCHANGED ⟨log, endPoint⟩
        ∧ Send([mtype          ↦ RequestSyncResponse,
                mterm          ↦ currentTerm[i],
                msyncGranted   ↦ grant,
                msync          ↦ sync[i],
                mstart         ↦ m.mstart,
                mend           ↦ m.mend,
                msource        ↦ i,
                mdest          ↦ j])
  ∧ UNCHANGED ⟨currentTerm, currentState, sync, votedFor, electionVars, syncTrack, allSynced⟩

HandleRequestSyncResponse(i) ≜
  ∧ ∃ m ∈ messages :
    LET j ≜ m.msource
    IN  ∧ m.mtype = RequestSyncResponse
        ∧ m.mdest = i
        ∧ currentTerm[i] = m.mterm
        ∧ currentState[i] ∈ {LeaderCandidate, Leader}
        ∧ syncTrack' = [syncTrack EXCEPT ![i][j] = m.msync]
        ∧ ∨ ∧ m.msyncGranted
            ∧ m.msync < sync[i]
            ∧ Send([mtype    ↦ UpdateSyncRequest,
                    mterm    ↦ currentTerm[i],
                    msync    ↦ Min({sync[i]} ∪ {k ∈ Nat : ∧ k > m.msync
                                                          ∧ Cardinality({n ∈ Index : log[i][n].term
                    msource  ↦ i,
                    mdest    ↦ {j}])
```

```
                    ∨ ∧ ¬m.msyncGranted
                      ∧ UNCHANGED messages
          ∧ UNCHANGED ⟨serverVars, logVars, electionVars, allSynced⟩

──────────────────────────────────────────────────────────────────────────────

UpdateSync(i) ≜
    ∧ currentState[i] = LeaderCandidate
    ∧ ∃ Q ∈ Quorums :
        LET syncUpdated ≜ {m ∈ messages : ∧ m.mtype  = RequestSyncResponse
                                          ∧ m.mterm = currentTerm[i]
                                          ∧ m.msyncGranted = TRUE
                                          ∧ m.msync = sync[i]
                                          ∧ m.msource ∈ Q
                                          ∧ m.mdest = i}
        IN  ∧ ∀ q ∈ Q : ∨ ∃ m ∈ syncUpdated : m.msource = q
                        ∨ q = i
            ∧ allSynced' = LET indexes ≜ {n ∈ Index : log[i][n].term = sync[i]}
                               entries ≜ {⟨n, [term      ↦ log[i][n].term,
                                               date      ↦ log[i][n].date,
                                               value     ↦ log[i][n].value,
                                               committed ↦ TRUE]⟩ : n ∈ indexes}
                           IN  allSynced ∪ {⟨sync[i], endPoint[i][sync[i]][2], entries⟩}
            ∧ Send([mtype    ↦ UpdateSyncRequest,
                    mterm    ↦ currentTerm[i],
                    msync    ↦ currentTerm[i],    here 2
                    msource  ↦ i,
                    mdest    ↦ Q])
    ∧ UNCHANGED ⟨serverVars, logVars, leaderVars, electionVars⟩

HandleUpdateSyncRequest(i) ≜
    ∃ m ∈ messages :
        LET j     ≜ m.msource
            grant ≜ ∧ currentTerm[i] = m.mterm
                    ∧ m.msync > sync[i]
        IN  ∧ m.mtype = UpdateSyncRequest
            ∧ i ∈ m.mdest
            ∧ m.mterm ≤ currentTerm[i]
            ∧ ∨ ∧ grant
                ∧ sync' = [sync EXCEPT ![i] = m.msync]                    here
                ∧ log'  = [log EXCEPT ![i] = [n ∈ Index ↦ IF log[i][n].term = sync[i] THEN [term ↦ log[
                                                                                            date ↦ log[
                                                                                            value ↦ log
                                                                                            committed ↦
                                                                                     ELSE log[i][n]]]
              ∨ ∧ ¬grant
```

$\land$ UNCHANGED $\langle log, sync \rangle$
$\land Send([mtype \mapsto UpdateSyncResponse,$
$mterm \mapsto currentTerm[i],$
$mupdateSyncGranted \mapsto grant,$
$msync \mapsto sync'[i],$
$msource \mapsto i,$
$mdest \mapsto j])$
$\land$ UNCHANGED $\langle currentTerm, currentState, votedFor, endPoint, leaderVars, electionVars, allSynced \rangle$

$HandleUpdateSyncResponse(i) \triangleq$
$\land \exists\, m \in messages :$
LET $j \triangleq m.msource$
IN $\land m.mtype = UpdateSyncResponse$
$\land m.mdest = i$
$\land currentTerm[i] = m.mterm$
$\land currentState[i] \in \{Leader, LeaderCandidate\}$
$\land \lor \land m.mupdateSyncGranted$
$\land syncTrack' = [syncTrack$ EXCEPT $![i][j] = m.msync]$
$\lor \land \neg m.mupdateSyncGranted$
$\land$ UNCHANGED $syncTrack$
$\land$ UNCHANGED $\langle messages, serverVars, logVars, electionVars, allSynced \rangle$

$BecomeLeader(i) \triangleq$
$\land currentState[i] = LeaderCandidate$
$\land \exists\, Q \in Quorums :$
$\land \forall\, q \in Q : \lor q = i$
$\lor syncTrack[i][q] = currentTerm[i]$
$\land elections' = elections \cup \{[eterm \mapsto currentTerm[i],$
$esync \mapsto sync[i],$ here
$eleader \mapsto i,$
$evotes \mapsto Q,$
$evoterLog \mapsto \{log[k] : k \in Q\},$
$elog \mapsto log[i]]\}$
$\land sync' = [sync$ EXCEPT $![i] = currentTerm[i]]$
$\land currentState' = [currentState$ EXCEPT $![i] = Leader]$
$\land log' = [log$ EXCEPT $![i] = [n \in Index \mapsto$ IF $log[i][n].term = sync[i]$ THEN $[term \mapsto log[i][n].term,$
$date \mapsto log[i][n].date,$
$vallue \mapsto log[i][n].value,$
$committed \mapsto$ TRUE$]$
ELSE $log[i][n]]]$
$\land$ UNCHANGED $\langle messages, currentTerm, votedFor, endPoint, leaderVars, halfElections, allSynced \rangle$

$ClientRequest(i, v) \triangleq$
$\land$ LET $nextIndex \triangleq logTail(log[i]) + 1$

$$
\begin{array}{l}
\quad\quad\quad\quad entry \triangleq [term \mapsto currentTerm[i], \\
\quad\quad\quad\quad\quad\quad\quad\quad date \mapsto currentTerm[i], \\
\quad\quad\quad\quad\quad\quad\quad\quad value \mapsto v, \\
\quad\quad\quad\quad\quad\quad\quad\quad committed \mapsto \text{FALSE}] \\
\quad\quad\quad \text{IN} \quad \land\, currentState[i] = Leader \\
\quad\quad\quad\quad\quad \land\, nextIndex \in Nat \\
\quad\quad\quad\quad\quad \land\, log' = [log \text{ EXCEPT } ![i][nextIndex] = entry] \\
\quad\quad\quad\quad\quad \land \text{UNCHANGED } \langle messages,\, serverVars,\, electionVars,\, syncTrack,\, allSynced \rangle
\end{array}
$$

$$
\begin{array}{l}
CommitEntry(i,\, n) \triangleq \\
\quad\quad \land \quad \exists\, Q \in Quorums : \\
\quad\quad\quad\quad \text{LET } succ \triangleq \{m \in messages : \land\, m.mtype = RequestSyncResponse \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \land\, m.msyncGranted = \text{TRUE} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \land\, m.mdest \;\; = i \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \land\, m.mterm = currentTerm[i] \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \land\, m.msource \in Q \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \land\, n \in m.mstart \,..\, m.mend\} \\
\quad\quad\quad\quad \text{IN} \quad \land \forall\, q \;\in Q : \exists\, m \in succ : \lor\, q = i \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \lor\, m.msource = q \\
\quad\quad\quad\quad\quad\quad\quad \land\, log' = [log \text{ EXCEPT } ![i][n].committed = \text{TRUE}] \\
\quad\quad \land \quad currentState[i] = Leader \\
\quad\quad \land \quad \text{UNCHANGED } \langle messages,\, serverVars,\, log,\, syncTrack,\, electionVars,\, allSynced \rangle
\end{array}
$$

$$
\begin{array}{l}
Next \triangleq \quad \land \\
\quad\quad\quad\quad \lor \exists\, i \in Server : Restart(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server : Timeout(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server : UpdateTerm(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server : RequestVote(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server : HandleRequestVoteRequest(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server : BecomeLeaderCandidate(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server : BecomeLeader(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server,\, v \in Value : ClientRequest(i,\, v) \\
\quad\quad\quad\quad \lor \exists\, i,\, j \in Server : RequestSync(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server : HandleRequestSyncRequest(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server : HandleRequestSyncResponse(i) \\
\quad\quad\quad\quad \lor \exists\, i,\, j \in Server : UpdateSync(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server : HandleUpdateSyncRequest(i) \\
\quad\quad\quad\quad \lor \exists\, i \in Server : HandleUpdateSyncResponse(i) \\
\\
\quad\quad\quad \land \quad allLogs' = allLogs \cup \{log[i] \quad\; : i \in Server\} \\
\quad\quad\quad \land \text{LET } entries(i) \triangleq \{\langle n,\, log[i][n] \rangle : n \in Index\} \\
\quad\quad\quad\quad\quad \text{IN} \quad allEntries' = allEntries \cup \text{UNION } \{entries(i) : i \in Server\}
\end{array}
$$

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

$$AllEntries(i) \triangleq \{\langle n, log[i][n]\rangle : n \in Index\}$$

$$Lemma1 \triangleq \forall\, i \in Server : sync[i] \leq currentTerm[i]$$
$$Lemma2 \triangleq \forall\, i \in Server : currentState[i] = Leader \Rightarrow sync[i] = currentTerm[i]$$
$$Lemma3 \triangleq \forall\, e, f \in halfElections : e.eterm = f.eterm \Rightarrow e.eleaderCandidate = f.eleaderCandidate$$
$$Lemma4 \triangleq \forall\, e \in elections : \exists f \in halfElections : e.eterm = f.eterm \land e.eleader = f.eleaderCandidate$$
$$Lemma5 \triangleq \forall\, e, f \in elections : e.eterm = f.eterm \Rightarrow e.eleader = f.eleader$$
$$Lemma6 \triangleq \forall\, i \in Server : currentState[i] = Leader \Rightarrow currentTerm[i] = sync[i]$$
$$Lemma7 \triangleq \forall\, e \in halfElections : e.esync < e.eterm$$
$$Lemma8 \triangleq \forall\, i, j \in Server, n \in Index : log[i][n].term = log[j][n].term \Rightarrow log[i][n].value = log[j][n].value$$
$$Lemma9 \triangleq \forall\, s1, s2 \in allSynced : s1[1] = s2[1] \Rightarrow s1 = s2$$
$$Lemma10 \triangleq \forall\, e1, e2 \in elections : e1.eterm < e2.eterm \Rightarrow \exists\, s \in allSynced : s[1] = e1.term$$
$$Lemma11 \triangleq \text{LET } indexes(i, t) \triangleq \{n \in Index : log[i][n].term = t\}$$
$$entries(i, t) \triangleq \{\langle n, log[i][n]\rangle : n \in indexes(i, t)\} \text{IN}$$
$$\forall\, i \in Server : \forall\, t \in Term :$$
$$t < sync[i] \land (\exists\, e \in elections : e.eterm = t) \Rightarrow \exists\, s \in allSynced : s[1] = t \land entries(i, t) = s[3]$$
$$Lemma12 \triangleq \forall\, i \in Server : \forall\, e \in AllEntries(i) : e[2].term \leq sync[i]$$
$$Lemma13 \triangleq \forall\, e \in halfElections : \forall f \in elections : f.eterm \leq e.esync \lor f.eterm \geq e.eterm$$
$$syncCompleteness \triangleq \forall\, i, j \in Server :$$
$$\{e \in AllEntries(i) : e[2].term \geq 0 \land e[2].term < Min(\{sync[i], sync[j]\})\} =$$
$$\{e \in AllEntries(j) : e[2].term \geq 0 \land e[2].term < Min(\{sync[i], sync[j]\})\}$$

* Modification History
* Last modified *Wed* May 12 22:25:17 *CST* 2021 by Dell
* Created *Tue* May 11 22:35:25 *CST* 2021 by Dell