```
──────────────────────── MODULE ParallelRaftCE ────────────────────────
EXTENDS Integers, FiniteSets, Sequences, TLC, Naturals

CONSTANTS Server, Follower, Candidate, Leader, LeaderCandidate, Nil, Value

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

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

serverVars ≜ ⟨currentTerm, currentState, votedFor, sync, endPoint⟩

VARIABLE log


VARIABLE syncTrack
leaderVars ≜ ⟨syncTrack⟩


VARIABLE halfElections
VARIABLE elections
electionVars ≜ ⟨halfElections, elections⟩


VARIABLE allLogs
VARIABLE allEntries
VARIABLE allSynced


vars ≜ ⟨messages, allLogs, allEntries, log, serverVars, leaderVars, allSynced, electionVars⟩

Quorums ≜ {i ∈ SUBSET (Server) : Cardinality(i) ∗ 2 > Cardinality(Server)}

Send(m) ≜ messages′ = messages ∪ {m}


Index ≜ Nat
Term ≜ Nat

Min(s) ≜ IF s = {} THEN −1 ELSE CHOOSE i ∈ s : ∀ j ∈ s : j ≥ i
Max(s) ≜ IF s = {} THEN −1 ELSE CHOOSE i ∈ s : ∀ j ∈ s : i ≥ j
```

$$\begin{aligned}
\mathit{InitServerVars} \triangleq\ & \text{LET } k \triangleq \text{CHOOSE } x \in \mathit{Server} : x \in \mathit{Server} \\
& \text{IN} \\
& \land\ \mathit{currentTerm} = [i \in \mathit{Server} \mapsto 0] \\
& \land\ \mathit{sync} = [i \in \mathit{Server} \mapsto 0] \\
& \land\ \mathit{currentState} = [i \in \mathit{Server} \mapsto \mathit{Follower}] \\
& \land\ \mathit{endPoint} = [i \in \mathit{Server} \mapsto [n \in \mathit{Term} \mapsto \langle -1,\ -1 \rangle]] \\
& \land\ \mathit{votedFor} = [i \in \mathit{Server} \mapsto \mathit{Nil}]
\end{aligned}$$

$$\mathit{InitLeaderVars} \triangleq\ \land\ \mathit{syncTrack} = [i \in \mathit{Server} \mapsto [j \in \mathit{Server} \mapsto 0]]$$

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

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

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

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

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

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

```
$\mathit{Restart}(i) \triangleq$
    $\land\ \mathit{currentState}' = [\mathit{currentState}\ \textsc{except}\ ![i] = \mathit{Follower}]$
    $\land\ \mathit{syncTrack}' = [\mathit{syncTrack}\ \textsc{except}\ ![i] = [j \in \mathit{Server} \mapsto 0]]$
    $\land\ \textsc{unchanged}\ \langle \mathit{messages}, \mathit{currentTerm}, \mathit{endPoint}, \mathit{sync}, \mathit{votedFor}, \mathit{log},$
                              $\mathit{electionVars}, \mathit{allSynced} \rangle$

$\mathit{Timeout}(i) \triangleq$
    $\land\ \mathit{currentState}[i] \in \{\mathit{Follower}, \mathit{Candidate}\}$
    $\land\ \mathit{currentState}' = [\mathit{currentState}\ \textsc{except}\ ![i] = \mathit{Candidate}]$
    $\land\ \mathit{currentTerm}' = [\mathit{currentTerm}\ \textsc{except}\ ![i] = \mathit{currentTerm}[i] + 1]$
    $\land\ \mathit{currentTerm}[i] + 1 \in \mathit{Term}$
    $\land\ \mathit{votedFor}' = [\mathit{votedFor}\ \textsc{except}\ ![i] = \mathit{Nil}]$
    $\land\ \textsc{unchanged}\ \langle \mathit{messages}, \mathit{leaderVars}, \mathit{sync}, \mathit{endPoint}, \mathit{log}, \mathit{syncTrack},$
                            $\mathit{electionVars}, \mathit{allSynced} \rangle$

$\mathit{UpdateTerm}(i) \triangleq$
    $\land\ \exists\, m \in \mathit{messages} :$
            $\land\ m.\mathit{mterm} > \mathit{currentTerm}[i]$
            $\land\ \lor\ m.\mathit{mdest} = i$
                $\lor\ m.\mathit{mdest} = \mathit{Nil}$
            $\land\ \mathit{currentTerm}' = [\mathit{currentTerm}\ \textsc{except}\ ![i] = m.\mathit{mterm}]$
            $\land\ \mathit{currentState}' = [\mathit{currentState}\ \textsc{except}\ ![i] = \mathit{Follower}]$
            $\land\ \mathit{votedFor}' = [\mathit{votedFor}\ \textsc{except}\ ![i] = \mathit{Nil}]$
    $\land\ \textsc{unchanged}\ \langle \mathit{messages}, \mathit{sync}, \mathit{log}, \mathit{leaderVars}, \mathit{electionVars}, \mathit{allSynced}, \mathit{endPoint} \rangle$


$\mathit{RequestVote}(i) \triangleq$
    $\land\ \mathit{currentState}[i] = \mathit{Candidate}$
    $\land\ \mathit{Send}([\mathit{mtype} \mapsto \mathit{RequestVoteRequest},$
               $\mathit{mterm} \mapsto \mathit{currentTerm}[i],$
               $\mathit{msync} \mapsto \mathit{sync}[i],$
               $\mathit{msource} \mapsto i,$
               $\mathit{mdest} \mapsto \mathit{Nil}])$
    $\land\ \textsc{unchanged}\ \langle \mathit{serverVars}, \mathit{leaderVars}, \mathit{log}, \mathit{electionVars}, \mathit{allSynced} \rangle$

$\mathit{HandleRequestVoteRequest}(i) \triangleq$
    $\land\ \exists\, m \in \mathit{messages} :$
        $\textsc{let}\ j \triangleq m.\mathit{msource}$
            $\mathit{syncOK} \triangleq \land\ m.\mathit{msync} \geq \mathit{sync}[i]$
            $\mathit{grant} \triangleq \land\ \mathit{syncOK}$
                       $\land\ \mathit{votedFor}[i] \in \{\mathit{Nil}, j\}$
                       $\land\ \mathit{currentTerm}[i] = m.\mathit{mterm}$
        $\textsc{in}$
            $\land\ m.\mathit{mterm} \leq \mathit{currentTerm}[i]$
            $\land\ m.\mathit{mtype} = \mathit{RequestVoteRequest}$
        ∧ ∨ 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, log, leaderVars,
                         electionVars, allSynced, endPoint⟩

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.date
                                 □    committed ≠ {} → CHOOSE x ∈ committed : TRUE
                             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}
                          e ≜ CHOOSE e1 ∈ allPoint    : (∀ e2 ∈ allPoint : e1[1] ≥ e2[1])
                      IN  IF e[1] = −1 THEN Max({e1[1] : e1 ∈ allLog})
                            ELSE e[2]
            toRecover ≜ {n ∈ 0 . . end : log[i][n].committed = FALSE}
```

```
              toSync ≜  {⟨n, Merge({l[2] : l ∈ {t ∈ allLog : t[1] = n}}, sync[i], currentTerm[i])⟩
                                      : n ∈ toRecover}
          IN
          ∧ ∀ q  ∈ Q : ∃ m ∈ voteGranted : m.msource          = q
          ∧ log′ = [log EXCEPT ![i] = IF end = −1 THEN [n ∈ Index ↦ IF log[i][n].term = sync[i] THEN
                                                                               [term ↦ −1,
                                                                                date ↦ −1,
                                                                                value ↦ Nil,
                                                                                committed ↦ FALSE]
                                                                          ELSE log[i][n]]
                                     ELSE [n ∈ Index ↦  IF n ∈ toRecover THEN
                                                              (CHOOSE e ∈ toSync : e[1] = n)[2]
                                                         ELSE IF (n > end) THEN
                                                                [term ↦ −1,
                                                                 date ↦ −1,
                                                                 value ↦ Nil,
                                                                 committed ↦ FALSE]
                                                         ELSE log[i][n]]]
          ∧ endPoint′ = [endPoint EXCEPT ![i][sync[i]] = ⟨currentTerm[i], end⟩]
          ∧ halfElections′ = halfElections ∪ {[eterm ↦ currentTerm[i],
                                                eleaderCandidate ↦ i,
                                                esync ↦ sync[i],
                                                evotes ↦ Q,
                                                elog ↦ log[i]]}
      ∧ currentState′ = [currentState EXCEPT ![i] = LeaderCandidate]
      ∧ syncTrack′ = [syncTrack EXCEPT ![i] = [j ∈ Server ↦ sync[i]]]
      ∧ UNCHANGED ⟨messages, currentTerm, votedFor, sync, elections, allSynced⟩

RequestSync(i) ≜
      ∧ currentState[i] ∈ {LeaderCandidate, Leader}
      ∧ ∃ s ∈ 0 . . sync[i] :
          LET start ≜ Min({n ∈ Index : log[i][n].term = s})
              end ≜ Max({n ∈ Index : log[i][n].term = s})
          IN
              ∧ Send([mtype ↦ RequestSyncRequest,
                      mterm ↦ currentTerm[i],
                      msync ↦ s,
                      mstart ↦ start,
                      mend ↦ end,
                      mentries ↦ IF start = −1 THEN Nil ELSE [n ∈ start . . end ↦ log[i][n]],
                      msource ↦ i,
                      mdest ↦ Nil])
      ∧ UNCHANGED ⟨serverVars, log, electionVars, syncTrack, allSynced⟩
```

```
HandleRequestSyncRequest(i) ≜
    ∧ ∃ m ∈ messages :
                LET  j ≜ m.msource
                     grant ≜  ∧ m.mterm = currentTerm[i]
                              ∧ 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 = sync[i] THEN
                                                                   [term ↦ −1,
                                                                    date ↦ −1,
                                                                    value ↦ Nil,
                                                                    committed ↦ FALSE]
                                                               ELSE
                                                                 log[i][n]]
                                               ELSE
                                                 [n ∈ Index ↦  IF n < m.mstart THEN log[i][n]
                                                               ELSE  IF n ∈ m.mstart .. m.mend
                                                                       THEN m.mentries[n]
                                                               ELSE  [term ↦ −1,
                                                                      date ↦ −1,
                                                                      value ↦ Nil,
                                                                      committed ↦ FALSE]]]
                   ∧ 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.msourceIN
        ∧ m.mtype = RequestSyncResponse
        ∧ m.mdest = i
        ∧ currentTerm[i] = m.mterm
        $\land\ currentState[i] \in \{Leader,\ LeaderCandidate\}$
        $\land\ syncTrack' = [syncTrack\ \text{EXCEPT}\ ![i][j] = m.msync]$
        $\land\ \lor\ \land\ m.msyncGranted$
                $\land\ m.msync < sync[i]$
                $\land\ Send([mtype \mapsto UpdateSyncRequest,$
                        $mterm \mapsto currentTerm[i],$
                        $msync \mapsto Min(\{sync[i]\} \cup \{k \in Nat : k > m.msync\ \land$
                                $Cardinality(\{n \in Index : log[i][n].term = k\}) > 0\}),$
                        $msource \mapsto i,$
                        $mdest \mapsto \{j\}])$
            $\lor\ \land\ \neg m.msyncGranted$
                $\land\ \text{UNCHANGED}\ messages$
    $\land\ \text{UNCHANGED}\ \langle serverVars,\ log,\ electionVars,\ allSynced \rangle$

$UpdateSync(i)\ \triangleq$
    $\land\ currentState[i] = LeaderCandidate$
    $\land\ \exists\, Q \in Quorums :$
            $\text{LET}\ syncUpdated\ \triangleq\ \{m \in messages :\ \land\ m.mtype\ = RequestSyncResponse$
                                                    $\land\ m.mterm = currentTerm[i]$
                                                    $\land\ m.msyncGranted = \text{TRUE}$
                                                    $\land\ m.msync = sync[i]$
                                                    $\land\ m.msource \in Q$
                                                    $\land\ m.mdest = i\}$
              $\text{IN}$
            $\land\ \forall\, q \in Q : (\exists\, m \in syncUpdated : m.msource = q) \lor q = i$
            $\land\ allSynced' = \text{LET}\ indexes\ \triangleq\ \{n \in Index : log[i][n].term = sync[i]\}$
                                    $entries\ \triangleq\ \{\langle n,\ [term \mapsto log[i][n].term,$
                                                    $date\ \mapsto log[i][n].date,$
                                                    $value \mapsto log[i][n].value,$
                                                    $committed \mapsto \text{TRUE}] \rangle : n \in indexes\}$
                                $\text{IN}\ \ allSynced \cup \{\langle sync[i],\ endPoint[i][sync[i]][2],\ entries \rangle\}$
            $\land\ Send([mtype\ \mapsto UpdateSyncRequest,$
                    $mterm \mapsto currentTerm[i],$
                    $msync\ \mapsto currentTerm[i],$
                    $msource \mapsto i,$
                    $mdest \mapsto Q])$
    $\land\ \text{UNCHANGED}\ \langle serverVars,\ log,\ leaderVars,\ electionVars \rangle$

$HandleUpdateSyncRequest(i)\ \triangleq$
    $\exists\, m \in messages :$
      $\text{LET}\ grant\ \triangleq\ \land\ currentTerm[i] = m.mterm$
                        $\land\ m.msync > sync[i]$
            $j\ \triangleq\ m.msource$
      $\text{IN}$
       $\land\ m.mtype = UpdateSyncRequest$
```

```
        $\land\ i \in m.mdest$
        $\land\ m.mterm \leq currentTerm[i]$
        $\land\ \lor\ \land\ grant$
                  $\land\ sync' = [sync\ \text{EXCEPT}\ ![i] = m.msync]$
                  $\land\ log' = [log\ \text{EXCEPT}\ ![i] = [n \in Index \mapsto$
                                      $\text{IF}\ log[i][n].term = sync[i]\ \text{THEN}$
                                          $[term \mapsto log[i][n].term,$
                                           $date \mapsto log[i][n].date,$
                                           $value \mapsto log[i][n].value,$
                                           $committed \mapsto \text{TRUE}]$
                                       $\text{ELSE}\ \ log[i][n]]]$
           $\lor\ \land\ \neg grant$
                  $\land\ \text{UNCHANGED}\ \langle log,\ sync\rangle$
        $\land\ Send([\ mtype \mapsto UpdateSyncResponse,$
                      $mterm \mapsto currentTerm[i],$
                      $mupdateSyncGranted \mapsto grant,$
                      $msync \mapsto sync'[i],$
                      $msource \mapsto i,$
                      $mdest \mapsto j])$
    $\land\ \text{UNCHANGED}\ \langle currentTerm,\ currentState,\ votedFor,\ endPoint,\ leaderVars,\ electionVars,\ allSynced\rangle$

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

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

ClientRequest(i, v) ≜
   LET nextIndex ≜  logTail(log[i]) + 1
           entry  ≜ [term ↦ currentTerm[i],
                     date  ↦ currentTerm[i],
                     value ↦ v,
                     committed ↦ FALSE]
   IN
    ∧ currentState[i] = Leader
    ∧ nextIndex ∈ Nat
    ∧ log′ = [log EXCEPT ![i][nextIndex] = entry]
    ∧ UNCHANGED ⟨messages, serverVars, electionVars, syncTrack, allSynced⟩

CommitEntry(i, n) ≜
    ∧ ∃ Q ∈ Quorums :
      LET succ ≜ {m ∈ messages : ∧ m.type = RequestSyncResponse
                                 ∧ m.msyncGranted = TRUE
                                 ∧ m.mdest  = i
                                 ∧ m.mterm = currentTerm[i]
                                 ∧ m.msource ∈ Q
                                 ∧ n ∈ m.mstart . . m.mend}
      IN    ∧ ∀ q  ∈ Q : ∃ m ∈ succ : (m.msource = q ∨ q = i)
            ∧ log′ = [log EXCEPT ![i][n].committed = TRUE]
    ∧ currentState[i] = Leader
    ∧ UNCHANGED ⟨messages, serverVars, log, syncTrack, electionVars, allSynced⟩


Next ≜        ∧
                ∨ ∃ i ∈ Server : Restart(i)
                ∨ ∃ i ∈ Server : Timeout(i)
                ∨ ∃ i ∈ Server : UpdateTerm(i)
                ∨ ∃ i ∈ Server : RequestVote(i)
                ∨ ∃ i ∈ Server : HandleRequestVoteRequest(i)
                ∨ ∃ i ∈ Server : BecomeLeaderCandidate(i)
                ∨ ∃ i ∈ Server : BecomeLeader(i)
                ∨ ∃ i ∈ Server, v ∈ Value : ClientRequest(i, v)
                ∨ ∃ i, j ∈ Server : RequestSync(i)
                ∨ ∃ i ∈ Server : HandleRequestSyncRequest(i)
```

$$
\begin{array}{ll}
& \quad \lor \exists\, i \in Server : HandleRequestSyncResponse(i) \\
& \quad \lor \exists\, i, j \in Server : UpdateSync(i) \\
& \quad \lor \exists\, i \in Server : HandleUpdateSyncRequest(i) \\
& \quad \lor \exists\, i \in Server : HandleUpdateSyncResponse(i) \\[6pt]
\land & allLogs' = allLogs \cup \{log[i] : i \in Server\} \\
\land & \textsc{let}\ entries(i) \triangleq \{\langle n,\ log[i][n]\rangle : n \in Index\} \\
& \textsc{in} \\
& allEntries' = allEntries \cup \textsc{union}\ \{entries(i) : i \in Server\}
\end{array}
$$

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

$Lemma1 \triangleq \forall\, i \in Server : sync[i] \leq currentTerm[i]$

$Lemma2 \triangleq \forall\, i \in Server : currentState[i] = Leader \Rightarrow sync[i] = currentTerm[i]$

$Lemma3 \triangleq \forall\, e, f \in halfElections : e.eterm = f.eterm \Rightarrow e.eleaderCandidate = f.eleaderCandidate$

$Lemma4 \triangleq \forall\, e \in elections : \exists f \in halfElections : e.eterm = f.eterm$
$\qquad\qquad\qquad\qquad\qquad\qquad \land\ e.eleader = f.eleaderCandidate$

$Lemma5 \triangleq \forall\, e, f \in elections : e.eterm = f.eterm \Rightarrow e.eleader = f.eleader$

$Lemma6 \triangleq \forall\, i \in Server \qquad : currentState[i] = Leader \Rightarrow currentTerm[i] = sync[i]$

$Lemma7 \triangleq \forall\, e \in halfElections : e.esync < e.eterm$

$Lemma8 \triangleq \forall\, i, j \in Server,\ n \in Index : log[i][n].term = log[j][n].term \Rightarrow$
$\qquad\qquad\qquad\qquad\qquad\qquad\quad log[i][n].value = log[j][n].value$

$Lemma9 \triangleq \forall\, s1,\ s2 \in allSynced : s1[1] = s2[1] \Rightarrow s1 = s2$

$Lemma10 \triangleq \forall\, e1,\ e2 \in elections : e1.eterm < e2.eterm \Rightarrow$
$\qquad\qquad\quad \exists\, s \in allSynced : s[1] = e1.term$

$$
\begin{array}{l}
Lemma11 \triangleq \textsc{let}\ indexes(i,\ t) \triangleq \{n \in Index : log[i][n].term = t\} \\
\qquad\qquad\qquad\quad entries(i,\ t) \triangleq \{\langle n,\ log[i][n]\rangle : n \in indexes(i,\ t)\}\textsc{in} \\
\qquad\qquad\quad \forall\, i \in Server : \forall\, t \in Term : \\
\qquad\qquad\quad t < sync[i] \land (\exists\, e \in elections : e.eterm = t) \Rightarrow \exists\, s \in allSynced : s[1] = t\ \land \\
\qquad\qquad\quad\ entries(i,\ t) = s[3]
\end{array}
$$

$Lemma12 \triangleq \forall\, i \in Server : \forall\, e \in AllEntries(i) : \ e[2].term \leq sync[i]$

$Lemma13 \triangleq \forall\, e \in halfElections : \forall\, f \in elections : f.eterm \leq e.esync \lor f.eterm \geq e.eterm$

$$
\begin{array}{l}
syncCompleteness \triangleq \forall\, i, j \in Server : \\
\qquad \{e \in AllEntries(i) : e[2].term \geq 0 \land e[2].term < Min(\{sync[i],\ sync[j]\})\} = \\
\qquad \{e \in AllEntries(j) : e[2].term \geq 0 \land e[2].term < Min(\{sync[i],\ sync[j]\})\}
\end{array}
$$

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

\ * Modification History

\ * Last modified *Fri Sep* 11 15:41:13 *CST* 2020 by 15150