MODULE *tem*

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

CONSTANTS *Server*

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

CONSTANTS *Nil*

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

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

VARIABLE *log*
$logVars \triangleq \langle log \rangle$

VARIABLE *syncTrack*
$leaderVars \triangleq \langle syncTrack \rangle$

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

VARIABLE *allLogs*
VARIABLE *allEntries*
VARIABLE *allSynced*

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

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

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

$$\begin{aligned}
&Value \triangleq Nat\\
&Index \triangleq Nat\\
&Term \triangleq Nat
\end{aligned}$$

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

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

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

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

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

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

$$Entries \triangleq [term : Nat, index : Nat, value : Value]$$

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

```
                    $\land$ $halfElections \in [eterm : Nat, eleaderCandidate : Server, esync : Nat,$
                                        $evotes : Quorums, elog : \text{SUBSET } Entries]$
                    $\land$ $elections \in [eterm : Nat, eleader : Server, evotes : Quorums, elog : \text{SUBSET } Entries]$

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

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

$Timeout(i) \triangleq$
    $\land$ $currentState[i] \in \{Follower, Candidate\}$
    $\land$ $currentState' = [currentState \text{ EXCEPT } ![i] = Candidate]$
    $\land$ $currentTerm' = [currentTerm \text{ EXCEPT } ![i] = currentTerm[i] + 1]$
    $\land$ $currentTerm[i] < 4$
    $\land$ $votedFor' = [votedFor \text{ EXCEPT } ![i] = Nil]$
    $\land$ $\text{UNCHANGED } \langle messages, leaderVars, sync, endPoint, logVars, syncTrack,$
                    $electionVars, allSynced\rangle$

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

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

$i$: recipient
$HandleRequestVoteRequest(i) \triangleq$
    $\land \exists m \in messages :$
        LET $j \triangleq m.msource$
            $syncOK \triangleq \land m.msync \geq sync[i]$
            grant ≜  ∧ syncOK
                     ∧ 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, 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}
            $allLog \triangleq \text{UNION } \{m.mlog : m \in voteResponded\}$
            $end \triangleq \text{LET } allPoint \triangleq \{m.mend : m \in voteResponded\}$
                            $e \triangleq \text{CHOOSE } e1 \in allPoint \quad : (\forall\, e2 \in allPoint : e1[1] \geq e2[1])$
                        $\text{IN} \quad \text{IF } e[1] = -1 \text{ THEN } Max(\{e1[1] : e1 \in allLog\})$
                                $\text{ELSE } \; e[2]$
            $toRecover \triangleq \{n \in 0 \mathrel{..} end : log[i][n].committed = \text{FALSE}\}$
            $toSync \triangleq \{\langle n,\, Merge(\{l[2] : l \in \{t \in allLog : t[1] = n\}\},\, sync[i],\, currentTerm[i])\rangle$
                                        $: n \in toRecover\}$
        $\text{IN}$
        $\land \forall\, q \in Q : \exists\, m \in voteGranted : m.msource = q$
        $\land log' = [log \text{ EXCEPT } ![i] = \text{IF } end = -1 \text{ THEN } [n \in Index \mapsto \text{IF } log[i][n].term = sync[i] \text{ THEN}$
                                                                                        $[term \mapsto -1,$
                                                                                        $\;date \mapsto -1,$
                                                                                        $\;value \mapsto Nil,$
                                                                                        $\;committed \mapsto \text{FALSE}]$
                                                                                $\text{ELSE } \; log[i][n]]$
                                            $\text{ELSE } \; [n \in Index \mapsto \; \text{IF } n \in toRecover \text{ THEN}$
                                                                    $(\text{CHOOSE } e \in toSync : e[1] = n)[2]$
                                                                $\text{ELSE IF } (n > end) \text{ THEN}$
                                                                    $[term \mapsto -1,$
                                                                    $\;date \mapsto -1,$
                                                                    $\;value \mapsto Nil,$
                                                                    $\;committed \mapsto \text{FALSE}]$
                                                                $\text{ELSE } \; log[i][n]]]$
        $\land endPoint' = [endPoint \text{ EXCEPT } ![i][sync[i]] = \langle currentTerm[i],\, end\rangle]$
        $\land halfElections' = halfElections \cup \{[eterm \mapsto currentTerm[i],$
                                                    $eleaderCandidate \mapsto i,$
                                                    $esync \mapsto sync[i],$
                                                    $evotes \mapsto Q,$
                                                    $elog \mapsto log[i]]\}$
    $\land currentState' = [currentState \text{ EXCEPT } ![i] = LeaderCandidate]$
    $\land syncTrack' = [syncTrack \text{ EXCEPT } ![i] = [j \in Server \mapsto sync[i]]]$
    $\land \text{UNCHANGED } \langle messages,\, currentTerm,\, votedFor,\, sync,\, elections,\, allSynced\rangle$

$RequestSync(i) \triangleq$
    $\land currentState[i] \in \{LeaderCandidate,\, Leader\}$
    $\land \exists\, s \in 0 \mathrel{..} sync[i] :$
        $\text{LET } start \triangleq Min(\{n \in Index : log[i][n].term = s\})$
            $end \triangleq Max(\{n \in Index : log[i][n].term = s\})$
        $\text{IN}$
            $\land Send([mtype \mapsto RequestSyncRequest,$
                    $mterm \mapsto currentTerm[i],$
                    $msync \mapsto s,$
                    $mstart \mapsto start,$
```

```
                  mend ↦ end,
                  mentries ↦ IF start = −1 THEN Nil ELSE [n ∈ start .. end ↦ log[i][n]],
                  msource ↦ i,
                  mdest ↦ Nil])
     ∧ UNCHANGED ⟨serverVars, logVars, 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,
                                                                 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) ≜
```

```
    $\land \exists\, m \in messages :$
       LET $j \triangleq m.msource$IN
       $\land\ m.mtype = RequestSyncResponse$
       $\land\ m.mdest = i$
       $\land\ currentTerm[i] = m.mterm$
       $\land\ currentState[i] \in \{Leader,\ LeaderCandidate\}$
       $\land\ syncTrack' = [syncTrack$ EXCEPT $![i][j] = m.msync]$
       $\land\ \lor\ \land\ m.msyncGranted$
                 $\land\ m.msync < sync[i]$
                 $\land\ Send([mtype \mapsto UpdateSyncRequest,$
                          $mterm \mapsto currentTerm[i],$
                          $msync \mapsto Min(\{sync[i]\} \cup \{k \in Nat : k > m.msync \land$
                                   $Cardinality(\{n \in Index : log[i][n].term = k\}) > 0\}),$
                          $msource \mapsto i,$
                          $mdest \mapsto \{j\}])$
           $\lor\ \land\ \neg m.msyncGranted$
              $\land$ UNCHANGED $messages$
    $\land$ UNCHANGED $\langle serverVars,\ logVars,\ electionVars,\ allSynced\rangle$

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

$HandleUpdateSyncRequest(i) \triangleq$
    $\exists\, m \in messages :$
```

```
      LET $grant \triangleq \land currentTerm[i] = m.mterm$
                     $\land m.msync > sync[i]$
          $j \triangleq m.msource$
      IN
       $\land m.mtype = UpdateSyncRequest$
       $\land i \in m.mdest$
       $\land m.mterm \leq currentTerm[i]$
       $\land \lor \land grant$
             $\land sync' = [sync$ EXCEPT $![i] = m.msync]$
             $\land log' = [log$ EXCEPT $![i] = [n \in Index \mapsto$
                                   IF $log[i][n].term = sync[i]$ THEN
                                        $[term \mapsto log[i][n].term,$
                                         $value \mapsto log[i][n].value,$
                                         $committed \mapsto$ TRUE$]$
                                    ELSE $log[i][n]]]$
          $\lor \land \neg grant$
             $\land$ 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 $\langle messages, serverVars, logVars, electionVars, allSynced \rangle$

$BecomeLeader(i) \triangleq$
    $\land currentState[i] = LeaderCandidate$
    $\land \exists Q \in Quorums : \forall q \in Q : (q = i \lor syncTrack[i][q] = currentTerm[i])$
                        $\land elections' = elections \cup \{[eterm \mapsto currentTerm[i],$
                                                    $esync \mapsto sync[i],$
                                                    $eleader \mapsto i,$
                                                    $evotes \mapsto Q,$
```

```
                                              $evoterLog \mapsto \{log[k] : k \in Q\},$
                                              $elog \mapsto log[i]]\}$
    $\land\ sync' = [sync\ \text{EXCEPT}\ ![i] = currentTerm[i]]$
    $\land\ currentState' = [currentState\ \text{EXCEPT}\ ![i] = Leader]$
    $\land\ log' = [log\ \text{EXCEPT}\ ![i] = [n \in Index \mapsto$
                              $\text{IF}\ log[i][n].term = sync[i]\ \text{THEN}$
                                  $[term \mapsto log[i][n].term,$
                                   $value \mapsto log[i][n].value,$
                                   $committed \mapsto \text{TRUE}]$
                              $\text{ELSE}\ \ log[i][n]]]$
    $\land\ \text{UNCHANGED}\ \langle messages, currentTerm, votedFor, endPoint, leaderVars, halfElections, allSynced\rangle$

$ClientRequest(i, v) \triangleq$
    $\text{LET}\ nextIndex \triangleq logTail(log[i]) + 1$
           $entry \triangleq [term \mapsto currentTerm[i],$
                      $value \mapsto v,$
                      $committed \mapsto \text{FALSE}]$
    $\text{IN}$
    $\land\ currentState[i] = Leader$
    $\land\ logTail(log[i]) < 3$
    $\land\ log' = [log\ \text{EXCEPT}\ ![i][nextIndex] = entry]$
    $\land\ \text{UNCHANGED}\ \langle messages, serverVars, electionVars, syncTrack, allSynced\rangle$

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

$Next \triangleq\ \ \land$
            $\lor\ \exists\, i \in Server : Restart(i)$
            $\lor\ \exists\, i \in Server : Timeout(i)$
            $\lor\ \exists\, i \in Server : UpdateTerm(i)$
            $\lor\ \exists\, i \in Server : RequestVote(i)$
            $\lor\ \exists\, i \in Server : HandleRequestVoteRequest(i)$
            $\lor\ \exists\, i \in Server : BecomeLeaderCandidate(i)$
            $\lor\ \exists\, i \in Server : BecomeLeader(i)$
            $\lor\ \exists\, i \in Server, v \in Value : ClientRequest(i, v)$
```

```
        $\lor \exists i, j \in Server : RequestSync(i)$
        $\lor \exists i \in Server : HandleRequestSyncRequest(i)$
        $\lor \exists i \in Server : HandleRequestSyncResponse(i)$
        $\lor \exists i, j \in Server : UpdateSync(i)$
        $\lor \exists i \in Server : HandleUpdateSyncRequest(i)$
        $\lor \exists i \in Server : HandleUpdateSyncResponse(i)$
     $\land\ allLogs' = allLogs \cup \{log[i] : i \in Server\}$
     $\land\ \text{LET } entries(i) \triangleq \{\langle n, log[i][n] \rangle : n \in Index\}$
        $\text{IN}$
        $allEntries' = allEntries \cup \text{UNION } \{entries(i) : i \in Server\}$

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

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

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

\ * Modification History
\ * Last modified *Tue Apr* 21 22:42:24 *CST* 2020 by *assstriker*
\ * Created *Mon Apr* 06 11:17:40 *CST* 2020 by *assstriker*