```
MODULE SemiSync
EXTENDS TLC, Integers, Sequences, FiniteSets

CONSTANTS Replica, nil

VARIABLES
    next_req,
    db, db_leader, db_epoch, db_leader_epoch,
    db_replicas, db_replicated,
    zk_leader, zk_epoch, zk_leader_epoch,
    zk_replicas, zk_status, zk_deleted, zk_num_change,
    zk_num_remove,
    ctl_pc, ctl_epoch, ctl_leader, ctl_replicas, ctl_offset,
    client_log

zk_vars ≜ ⟨zk_leader, zk_epoch, zk_leader_epoch,
    zk_replicas, zk_status, zk_deleted, zk_num_change, zk_num_remove⟩
db_vars ≜ ⟨db, db_leader, db_epoch, db_leader_epoch, db_replicas, db_replicated⟩
ctl_vars ≜ ⟨ctl_pc, ctl_epoch, ctl_leader, ctl_replicas, ctl_offset⟩
global_vars ≜ ⟨next_req, client_log⟩

vars ≜ ⟨global_vars, db_vars, zk_vars, ctl_vars⟩

LogEntry ≜ 41 .. 50
EpochNumber ≜ 11 .. 30

NullLogEntry ≜ LogEntry ∪ {nil}
NullReplica ≜ Replica ∪ {nil}

max_req ≜ 40 + 2
max_change_leader ≜ 2
max_remove_replica ≜ 1


replicationFactor(n) ≜ (n + 2) ÷ 2

quorumOfSet(S) ≜
    {Q ∈ SUBSET S : Cardinality(Q) = replicationFactor(Cardinality(S))}

quorumOf(r) ≜ quorumOfSet(db_replicas[r])


 ───────────────────────────────────────
 Map Functions
 ───────────────────────────────────────

Range(f) ≜ {f[x] : x ∈ DOMAIN f}

mapPut(f, k, v) ≜ LET newDomain ≜ (DOMAIN f) ∪ {k}IN
    ∧ f' = [x ∈ newDomain ↦ IF x = k THEN v ELSE f[x]]
```

$mapExist(f, k) \triangleq k \in \text{DOMAIN } f$

$mapDelete(f, k) \triangleq \text{LET } newDomain \triangleq (\text{DOMAIN } f) \setminus \{k\} \text{IN}$
$\quad \land f' = [x \in newDomain \mapsto f[x]]$

$MapOf(f, K, V) \triangleq$
$\quad \land \text{DOMAIN } f \subseteq K$
$\quad \land Range(f) \subseteq V$

$TypeOK \triangleq$
$\quad \land \quad next\_req \in LogEntry \cup \{40\}$

$\quad \land \quad zk\_leader \in NullReplica$
$\quad \land \quad zk\_epoch \in EpochNumber$
$\quad \land \quad zk\_leader\_epoch \in EpochNumber$
$\quad \land \quad zk\_replicas \subseteq Replica$
$\quad \land \quad zk\_status \in \{\text{“Normal”}, \text{“WaitReplicate”}, \text{“FindNewLeader”}\}$
$\quad \land \quad MapOf(zk\_deleted, Replica, 0 \,..\, 10)$
$\quad \land \quad zk\_num\_change \in 0 \,..\, 20$
$\quad \land \quad zk\_num\_remove \in 0 \,..\, 20$

$\quad \land \quad zk\_leader \neq nil \Rightarrow (zk\_leader \in zk\_replicas)$

$\quad \land \quad db \in [Replica \rightarrow Seq(LogEntry)]$
$\quad \land \quad db\_leader \in [Replica \rightarrow NullReplica]$
$\quad \land \quad db\_epoch \in [Replica \rightarrow EpochNumber]$
$\quad \land \quad db\_leader\_epoch \in [Replica \rightarrow EpochNumber]$
$\quad \land \quad db\_replicas \in [Replica \rightarrow \text{SUBSET } Replica]$
$\quad \land \quad db\_replicated \in [Replica \rightarrow [Replica \rightarrow 0 \,..\, 10]]$

$\quad \land \quad ctl\_pc \in \{$
$\qquad \text{“Init”}, \text{“CtlReadLeaderLog”}, \text{“CtlReadReplicaLog”},$
$\qquad \text{“CtlFindNewLeader”}\}$
$\quad \land \quad ctl\_epoch \in EpochNumber$
$\quad \land \quad ctl\_leader \in NullReplica$
$\quad \land \quad ctl\_replicas \subseteq Replica$
$\quad \land \quad ctl\_offset \in [Replica \rightarrow 0 \,..\, 10 \cup \{-1\}]$

$\quad \land \quad client\_log \in Seq(LogEntry)$

$Init \triangleq$
$\quad \land next\_req = 40$

$\quad \land zk\_leader \in Replica$
$\quad \land zk\_epoch = 11$
$\quad \land zk\_leader\_epoch = zk\_epoch$
$\quad \land zk\_replicas = \{zk\_leader\}$

```
    $\land\ zk\_status = $ “Normal”
    $\land\ zk\_deleted = \langle\rangle$
    $\land\ zk\_num\_change = 0$
    $\land\ zk\_num\_remove = 0$

    $\land\ db = [r \in Replica \mapsto \langle\rangle]$
    $\land\ db\_leader = [r \in Replica \mapsto zk\_leader]$
    $\land\ db\_epoch = [r \in Replica \mapsto zk\_epoch]$
    $\land\ db\_leader\_epoch = [r \in Replica \mapsto zk\_epoch]$
    $\land\ db\_replicas = [r \in Replica \mapsto zk\_replicas]$
    $\land\ db\_replicated = [r \in Replica \mapsto [r2 \in Replica \mapsto 0]]$

    $\land\ ctl\_pc = $ “Init”
    $\land\ ctl\_epoch = zk\_epoch$
    $\land\ ctl\_leader = zk\_leader$
    $\land\ ctl\_replicas = zk\_replicas$
    $\land\ ctl\_offset = [r \in Replica \mapsto 0]$

    $\land\ client\_log = \langle\rangle$

$AppendLog(r) \triangleq$
    $\land\ next\_req < max\_req$
    $\land\ db\_leader[r] = r$
    $\land\ next\_req' = next\_req + 1$
    $\land\ db' = [db \text{ EXCEPT } ![r] = Append(@, next\_req')]$
    $\land\ db\_replicated' = [db\_replicated \text{ EXCEPT } ![r][r] = Len(db'[r])]$
    $\land\ \text{UNCHANGED } \langle db\_leader, db\_epoch, db\_leader\_epoch, db\_replicas\rangle$
    $\land\ \text{UNCHANGED } client\_log$
    $\land\ \text{UNCHANGED } zk\_vars$
    $\land\ \text{UNCHANGED } ctl\_vars$

$AppendClientLog(r) \triangleq$
    $\land\ db\_leader[r] = r$
    $\land\ \text{LET } n \triangleq Len(client\_log) \text{ IN}$
        $\land\ n < Len(db[r])$
        $\land\ \exists\, Q \in quorumOf(r) :$
             $\forall\, r2 \in Q : db\_replicated[r][r2] > n$
        $\land\ client\_log' = Append(client\_log, db[r][n+1])$
    $\land\ \text{UNCHANGED } next\_req$
    $\land\ \text{UNCHANGED } db\_vars$
    $\land\ \text{UNCHANGED } zk\_vars$
    $\land\ \text{UNCHANGED } ctl\_vars$

$ZkAddReplica(r) \triangleq$
    $\land\ \neg(r \in zk\_replicas)$
    $\land \lnot(r \in \text{DOMAIN } zk\_deleted)$
    $\land zk\_status = \text{“Normal”} \lor zk\_status = \text{“WaitReplicate”}$
    $\land zk\_replicas' = zk\_replicas \cup \{r\}$
    $\land zk\_epoch' = zk\_epoch + 1$
    $\land zk\_status' = \text{“WaitReplicate”}$
    $\land \text{UNCHANGED } \langle zk\_leader, zk\_leader\_epoch, zk\_deleted, zk\_num\_change \rangle$
    $\land \text{UNCHANGED } zk\_num\_remove$
    $\land \text{UNCHANGED } global\_vars$
    $\land \text{UNCHANGED } db\_vars$
    $\land \text{UNCHANGED } ctl\_vars$

$ZkRemoveReplica(r) \triangleq$
    $\land zk\_num\_remove < max\_remove\_replica$
    $\land zk\_num\_remove' = zk\_num\_remove + 1$

    $\land r \in zk\_replicas$
    $\land r \neq zk\_leader$
    $\land zk\_status = \text{“Normal”} \lor zk\_status = \text{“WaitReplicate”}$
    $\land zk\_epoch' = zk\_epoch + 1$
    $\land zk\_replicas' = zk\_replicas \setminus \{r\}$
    $\land \text{IF } Cardinality(zk\_replicas') = 1$
        $\text{THEN } \land zk\_status' = \text{“Normal”}$
              $\land \text{UNCHANGED } zk\_deleted$
        $\text{ELSE } \land zk\_status' = \text{“WaitReplicate”}$
              $\land mapPut(zk\_deleted, r, Len(db[zk\_leader]))$
    $\land \text{UNCHANGED } \langle zk\_leader, zk\_leader\_epoch \rangle$
    $\land \text{UNCHANGED } zk\_num\_change$
    $\land \text{UNCHANGED } db\_vars$
    $\land \text{UNCHANGED } global\_vars$
    $\land \text{UNCHANGED } ctl\_vars$

$ZkPrepareChangeLeader \triangleq$
    $\land zk\_num\_change < max\_change\_leader$
    $\land zk\_num\_change' = zk\_num\_change + 1$

    $\land Cardinality(zk\_replicas) > 1$
    $\land zk\_status = \text{“Normal”}$
    $\land zk\_replicas' = zk\_replicas \setminus \{zk\_leader\}$
    $\land \exists r2 \in zk\_replicas' :$
        $\land mapPut(zk\_deleted, zk\_leader, Len(db[r2]))$
    $\land zk\_epoch' = zk\_epoch + 1$
    $\land \text{IF } Cardinality(zk\_replicas') > 1$
        $\text{THEN } \land zk\_status' = \text{“FindNewLeader”}$
              $\land zk\_leader' = nil$
              $\land \text{UNCHANGED } \langle zk\_leader\_epoch \rangle$
        ELSE  $\land$ UNCHANGED $zk\_status$
              $\land$ $zk\_leader' \in zk\_replicas'$
              $\land$ $zk\_leader\_epoch' = zk\_epoch'$
    $\land$ UNCHANGED $zk\_num\_remove$
    $\land$ UNCHANGED $db\_vars$
    $\land$ UNCHANGED $global\_vars$
    $\land$ UNCHANGED $ctl\_vars$

$newReplicated \triangleq [r \in Replica \mapsto 0]$

$DBUpdateZKInfo(r) \triangleq$
    $\land$ $db\_epoch[r] < zk\_epoch$
    $\land$ $db\_epoch' = [db\_epoch$ EXCEPT $![r] = zk\_epoch]$
    $\land$ $db\_leader\_epoch' = [db\_leader\_epoch$ EXCEPT $![r] = zk\_leader\_epoch]$
    $\land$ $db\_leader' = [db\_leader$ EXCEPT $![r] = zk\_leader]$
    $\land$ $db\_replicas' = [db\_replicas$ EXCEPT $![r] = zk\_replicas]$
    $\land$ IF $zk\_leader\_epoch = db\_leader\_epoch[r]$
        THEN UNCHANGED $db\_replicated$
        ELSE $db\_replicated' = [$
           $db\_replicated$ EXCEPT $![r] = [newReplicated$ EXCEPT $![r] = Len(db[r])]]$
    $\land$ UNCHANGED $db$
    $\land$ UNCHANGED $zk\_vars$
    $\land$ UNCHANGED $global\_vars$
    $\land$ UNCHANGED $ctl\_vars$

$DBReceveFromLeader(r) \triangleq$ LET $leader \triangleq db\_leader[r]$IN
    $\land$ $leader \neq r$
    $\land$ $leader \neq nil$
    $\land$ $r \in db\_replicas[r]$
    $\land$ $r \in db\_replicas[leader]$
    $\land$ $db\_leader\_epoch[r] = db\_leader\_epoch[leader]$
    $\land$ $Len(db[r]) < Len(db[leader])$
    $\land$ LET $n \triangleq Len(db[r])$IN
        $\land$  $db' = [db$ EXCEPT $![r] = Append(@, db[leader][n+1])]$
    $\land$ UNCHANGED $\langle db\_leader, db\_epoch, db\_leader\_epoch, db\_replicas, db\_replicated\rangle$
    $\land$ UNCHANGED $zk\_vars$
    $\land$ UNCHANGED $global\_vars$
    $\land$ UNCHANGED $ctl\_vars$

$DBUpdateReplicated(r, r1) \triangleq$
    $\land$ $db\_leader[r] = r$
    $\land$ $r1 \neq r$
    $\land$ $r1 \in db\_replicas[r1]$
    $\land$ $db\_leader\_epoch[r] = db\_leader\_epoch[r1]$
    $\land\ db\_replicated[r][r1] < Len(db[r1])$
    $\land\ db\_replicated' = [db\_replicated\ \text{EXCEPT}\ ![r][r1] = Len(db[r1])]$
    $\land\ \text{UNCHANGED}\ \langle db,\ db\_epoch,\ db\_leader,\ db\_leader\_epoch,\ db\_replicas\rangle$
    $\land\ \text{UNCHANGED}\ global\_vars$
    $\land\ \text{UNCHANGED}\ zk\_vars$
    $\land\ \text{UNCHANGED}\ ctl\_vars$


$zkStatusToCtlPC \triangleq$
   $\text{IF}\ zk\_status = \text{"WaitReplicate"}$
       $\text{THEN}\ \text{"CtlReadLeaderLog"}$
       $\text{ELSE}\ \text{IF}\ zk\_status = \text{"FindNewLeader"}$
           $\text{THEN}\ \text{"CtlFindNewLeader"}$
           $\text{ELSE}\ \text{"Init"}$

$initCtlOffset \triangleq ctl\_offset' = [r \in Replica \mapsto -1]$

$CtlUpdateZKInfo \triangleq$
    $\land\ ctl\_epoch < zk\_epoch$
    $\land\ ctl\_epoch' = zk\_epoch$
    $\land\ ctl\_leader' = zk\_leader$
    $\land\ ctl\_pc' = zkStatusToCtlPC$
    $\land\ ctl\_replicas' = zk\_replicas$
    $\land\ initCtlOffset$
    $\land\ \text{UNCHANGED}\ global\_vars$
    $\land\ \text{UNCHANGED}\ db\_vars$
    $\land\ \text{UNCHANGED}\ zk\_vars$

$CtlReadLeaderLog \triangleq$
    $\land\ ctl\_pc = \text{"CtlReadLeaderLog"}$
    $\land\ ctl\_epoch = db\_epoch[ctl\_leader]$
    $\land\ ctl\_pc' = \text{"CtlReadReplicaLog"}$
    $\land\ ctl\_offset' = [ctl\_offset\ \text{EXCEPT}\ ![ctl\_leader] = Len(db[ctl\_leader])]$
    $\land\ \text{UNCHANGED}\ \langle ctl\_epoch,\ ctl\_replicas,\ ctl\_leader\rangle$
    $\land\ \text{UNCHANGED}\ zk\_vars$
    $\land\ \text{UNCHANGED}\ global\_vars$
    $\land\ \text{UNCHANGED}\ db\_vars$

$CtlReadReplicaLog(r) \triangleq$
    $\land\ ctl\_pc = \text{"CtlReadReplicaLog"}$
    $\land\ r \neq ctl\_leader$
    $\land\ r \in ctl\_replicas$
    $\land\ ctl\_offset[r] < Len(db[r])$
    $\land\ ctl\_offset' = [ctl\_offset\ \text{EXCEPT}\ ![r] = Len(db[r])]$
    $\land$ UNCHANGED $\langle ctl\_pc, ctl\_replicas \rangle$
    $\land$ UNCHANGED $\langle ctl\_epoch, ctl\_leader \rangle$
    $\land$ UNCHANGED $zk\_vars$
    $\land$ UNCHANGED $db\_vars$
    $\land$ UNCHANGED $global\_vars$

$CtlSetZkNormal \triangleq$
    $\land ctl\_pc =$ "CtlReadReplicaLog"
    $\land ctl\_epoch = zk\_epoch$
    $\land \exists Q \in quorumOfSet(ctl\_replicas) :$
        $\forall r \in Q : ctl\_offset[r] \geq ctl\_offset[ctl\_leader]$

    $\land zk\_status' =$ "Normal"
    $\land zk\_epoch' = zk\_epoch + 1$
    $\land$ UNCHANGED $\langle zk\_leader\_epoch, zk\_leader, zk\_replicas, zk\_deleted \rangle$
    $\land$ UNCHANGED $\langle zk\_num\_change, zk\_num\_remove \rangle$

    $\land ctl\_pc' =$ "Init"
    $\land ctl\_epoch' = zk\_epoch'$
    $\land initCtlOffset$
    $\land$ UNCHANGED $\langle ctl\_leader, ctl\_replicas \rangle$

    $\land$ UNCHANGED $db\_vars$
    $\land$ UNCHANGED $global\_vars$

$CtlFindNewLeader(r) \triangleq$
    $\land ctl\_pc =$ "CtlFindNewLeader"
    $\land r \in ctl\_replicas$
    $\land db\_epoch[r] = ctl\_epoch$
    $\land ctl\_offset[r] < Len(db[r])$
    $\land ctl\_offset' = [ctl\_offset$ EXCEPT $![r] = Len(db[r])]$
    $\land$ UNCHANGED $\langle ctl\_epoch, ctl\_leader, ctl\_pc, ctl\_replicas \rangle$
    $\land$ UNCHANGED $zk\_vars$
    $\land$ UNCHANGED $global\_vars$
    $\land$ UNCHANGED $db\_vars$

$CtlSetNewLeader(r) \triangleq$
    $\land ctl\_pc =$ "CtlFindNewLeader"
    $\land r \in ctl\_replicas$
    $\land zk\_epoch = ctl\_epoch$
    $\land \exists Q \in quorumOfSet(ctl\_replicas) :$
        $\land r \in Q$
        $\land \forall r1 \in Q :$
            $\land ctl\_offset[r1] \geq 0$
            $\land ctl\_offset[r] \geq ctl\_offset[r1]$
    $\land\ zk\_epoch' = zk\_epoch + 1$
    $\land\ zk\_leader' = r$
    $\land\ zk\_leader\_epoch' = zk\_epoch'$

    $\land\ ctl\_epoch' = zk\_epoch'$
    $\land\ ctl\_leader' = zk\_leader'$
    $\land\ initCtlOffset$

    $\land\ zk\_status' =$ "WaitReplicate"
    $\land\ ctl\_pc' =$ "CtlReadLeaderLog"

    $\land$ UNCHANGED $ctl\_replicas$
    $\land$ UNCHANGED $\langle zk\_replicas, zk\_deleted, zk\_num\_change, zk\_num\_remove \rangle$
    $\land$ UNCHANGED $db\_vars$
    $\land$ UNCHANGED $global\_vars$


$subSeqMin(S, n) \triangleq$
    IF $Len(S) < n$
        THEN $S$
        ELSE $SubSeq(S, 1, n)$

$TruncateDeletedDB(r) \triangleq$
    $\land\ r \in$ DOMAIN $zk\_deleted$
    $\land\ db\_epoch[r] = zk\_epoch$
    $\land\ mapDelete(zk\_deleted, r)$
    $\land\ zk\_epoch' = zk\_epoch + 1$
    $\land\ db' = [db$ EXCEPT $![r] = subSeqMin(@, zk\_deleted[r])]$
    $\land$ UNCHANGED $\langle db\_epoch, db\_leader, db\_leader\_epoch, db\_replicas, db\_replicated \rangle$
    $\land$ UNCHANGED $\langle zk\_leader, zk\_leader\_epoch, zk\_replicas, zk\_status \rangle$
    $\land$ UNCHANGED $\langle zk\_num\_change, zk\_num\_remove \rangle$
    $\land$ UNCHANGED $ctl\_vars$
    $\land$ UNCHANGED $global\_vars$


$TerminateCond \triangleq$
    $\land\ next\_req = max\_req$
    $\land\ zk\_num\_change = max\_change\_leader$
    $\land\ zk\_num\_remove = max\_remove\_replica$
    $\land\ zk\_replicas = Replica$
    $\land\ zk\_leader \neq nil$
    $\land\ Len(client\_log) = Len(db[zk\_leader])$
    $\land\ zk\_status =$ "Normal"

$Terminated \triangleq$
    $\land\ TerminateCond$
    $\land$ UNCHANGED $vars$
```

```
$Next \triangleq$
    $\lor \exists\, r \in Replica :$
        $\lor AppendLog(r)$
        $\lor AppendClientLog(r)$
        $\lor ZkAddReplica(r)$
        $\lor ZkRemoveReplica(r)$
        $\lor DBUpdateZKInfo(r)$
        $\lor DBReceveFromLeader(r)$
        $\lor \exists\, r1 \in Replica : DBUpdateReplicated(r,\ r1)$
        $\lor CtlReadReplicaLog(r)$
        $\lor CtlFindNewLeader(r)$
        $\lor CtlSetNewLeader(r)$
        $\lor TruncateDeletedDB(r)$
    $\lor CtlUpdateZKInfo$
    $\lor CtlReadLeaderLog$
    $\lor CtlSetZkNormal$
    $\lor ZkPrepareChangeLeader$
    $\lor Terminated$

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

$FairSpec \triangleq Spec \land \mathrm{WF}_{vars}(Next)$

$AlwaysFinish \triangleq \Diamond TerminateCond$

$CanRecvReqAfterFailed \triangleq$
    $\land zk\_num\_change = max\_change\_leader$
    $\land zk\_replicas = Replica$
    $\land zk\_status =$ "Normal"
    $\land next\_req = 40$
    $\leadsto client\_log = \langle 41,\ 42 \rangle$


$ConsistentWhenLeaderValid \triangleq$
    $\land Len(db[zk\_leader]) \geq Len(client\_log)$
    $\land client\_log = SubSeq(db[zk\_leader],\ 1,\ Len(client\_log))$

$Consistent \triangleq$
    $\land zk\_leader \neq nil \Rightarrow ConsistentWhenLeaderValid$
    $\land zk\_status =$ "Normal" $\Rightarrow zk\_leader \neq nil$
    $\land \forall\, r \in \textsc{domain}\ zk\_deleted : \neg(r \in zk\_replicas)$
   $\land TerminateCond \Rightarrow client\_log \neq \langle 41,\ 42 \rangle$ \* $Couter$ Condition

$Perms \triangleq Permutations(Replica)$
```