MODULE *SemiSync*

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

CONSTANT *Client*, *Replica*, *nil*

VARIABLE
zk_leader, *zk_epoch*, *zk_leader_epoch*, *zk_status*,
old_leaders, *zk_catchup_index*,
db, *db_leader*, *db_replicated*, *db_epoch*, *db_status*,
next_req, *client_leader*, *client_success*,
pending, *pending_db*, *client_leader_epoch*,
healer_status, *healer_epoch*, *healer_replicas*

$vars \triangleq \langle$
zk_leader, *zk_epoch*, *zk_leader_epoch*, *zk_status*,
old_leaders, *zk_catchup_index*,
db, *db_leader*, *db_replicated*, *db_epoch*, *db_status*,
next_req, *client_leader*, *client_success*,
pending, *pending_db*, *client_leader_epoch*,
healer_status, *healer_epoch*, *healer_replicas*
$\rangle$

$zk\_vars \triangleq \langle zk\_leader, zk\_epoch, zk\_leader\_epoch, zk\_status,$
old_leaders, *zk_catchup_index*$\rangle$
$db\_vars \triangleq \langle db, db\_leader, db\_replicated, db\_epoch, db\_status\rangle$
$client\_vars \triangleq \langle$
next_req, *client_leader*, *client_success*,
pending, *pending_db*, *client_leader_epoch*$\rangle$
$healer\_vars \triangleq \langle healer\_status, healer\_epoch, healer\_replicas\rangle$

$max\_next\_req \triangleq 4$

$max\_change\_leader \triangleq 4$

$ReqSet \triangleq 60 \,..\, (60 + max\_next\_req)$

$Epoch \triangleq 0 \,..\, 20$

$NullReqSet \triangleq ReqSet \cup \{nil\}$

$NullReplica \triangleq Replica \cup \{nil\}$

$LogOffset \triangleq 0 \,..\, 20$

$Range(f) \triangleq \{f[x] : x \in \text{DOMAIN } f\}$

$replication\_factor \triangleq 2$

$Quorum \triangleq \{x \in \text{SUBSET } Replica : Cardinality(x) = replication\_factor\}$

$TypeOK \triangleq$
- $\wedge$ $zk\_leader \in Replica$
- $\wedge$ $zk\_epoch \in 1 \,..\, 30$
- $\wedge$ $zk\_leader\_epoch \in 1 \,..\, max\_change\_leader$
- $\wedge$ $zk\_status \in \{\text{“Normal”}, \text{“ChangingLeader”}, \text{“WaitReplicaLog”}\}$
- $\wedge$ $old\_leaders \subseteq Replica$
- $\wedge$ $zk\_catchup\_index \in (LogOffset \cup \{nil\})$

- $\wedge$ $db \in [Replica \to Seq(ReqSet)]$
- $\wedge$ $db\_leader \in [Replica \to Replica]$
- $\wedge$ $db\_replicated \in [Replica \to [Replica \to LogOffset]]$
- $\wedge$ $db\_epoch \in [Replica \to Epoch]$
- $\wedge$ $db\_status \in [Replica \to \{\text{“Writable”}, \text{“Replica”}, \text{“Frozen”}\}]$

- $\wedge$ $next\_req \in ReqSet$
- $\wedge$ $client\_leader \in [Client \to Replica]$
- $\wedge$ $client\_success \in [Client \to Seq(ReqSet)]$
- $\wedge$ $pending \in [Client \to NullReqSet]$
- $\wedge$ $pending\_db \in [Client \to NullReplica]$
- $\wedge$ $client\_leader\_epoch \in [Client \to Epoch]$

- $\wedge$ $healer\_status \in \{\text{“Init”}, \text{“UpdatingLeader”}, \text{“WaitReplica”}\}$
- $\wedge$ $healer\_epoch \in Epoch$
- $\wedge$ $healer\_replicas \in [Replica \to LogOffset \cup \{nil\}]$

$Init \triangleq$
- $\wedge$ $zk\_leader \in Replica$
- $\wedge$ $zk\_epoch = 1$
- $\wedge$ $zk\_leader\_epoch = 1$
- $\wedge$ $zk\_status = \text{“Normal”}$
- $\wedge$ $old\_leaders = \{\}$
- $\wedge$ $zk\_catchup\_index = nil$

- $\wedge$ $db = [r \in Replica \mapsto \langle\rangle]$
- $\wedge$ $db\_leader = [r \in Replica \mapsto zk\_leader]$
- $\wedge$ $db\_replicated \;= [r \in Replica \mapsto [r1 \in Replica \mapsto 0]]$
- $\wedge$ $db\_epoch = [r \in Replica \mapsto zk\_epoch]$
- $\wedge$ $db\_status = [r \in Replica \mapsto \text{IF } zk\_leader = r \text{ THEN “Writable” ELSE “Replica”}]$

- $\wedge$ $next\_req = 60$
- $\wedge$ $client\_leader = [c \in Client \mapsto zk\_leader]$
- $\wedge$ $client\_success = [c \in Client \mapsto \langle\rangle]$
- $\wedge$ $pending = [c \in Client \mapsto nil]$
- $\wedge$ $pending\_db = [c \in Client \mapsto nil]$

$\land$ $client\_leader\_epoch = [c \in Client \mapsto zk\_leader\_epoch]$

$\land$ $healer\_status =$ "Init"
$\land$ $healer\_epoch = 1$
$\land$ $healer\_replicas = [r \in Replica \mapsto nil]$

$StartRequest(c) \triangleq$
$\land$ $pending[c] = nil$
$\land$ $next\_req < 60 + max\_next\_req$
$\land$ $db\_status[client\_leader[c]] =$ "Writable"
$\land$ $next\_req' = next\_req + 1$
$\land$ $pending' = [pending \text{ EXCEPT } ![c] = next\_req']$
$\land$ $pending\_db' = [pending\_db \text{ EXCEPT } ![c] = client\_leader[c]]$
$\land$ LET $leader \triangleq client\_leader[c]$ IN
  $\land$ $db' = [db \text{ EXCEPT } ![leader] = Append(@, next\_req')]$
  $\land$ $db\_replicated' = [$
    $db\_replicated \text{ EXCEPT } ![leader][leader] = Len(db'[leader])]$
$\land$ UNCHANGED $\langle client\_leader, client\_success, client\_leader\_epoch\rangle$
$\land$ UNCHANGED $\langle db\_leader, db\_epoch, db\_status\rangle$
$\land$ UNCHANGED zk_vars
$\land$ UNCHANGED $healer\_vars$

$Replicate(r) \triangleq$
$\land$ $r \neq db\_leader[r]$
$\land$ $db\_status[r] =$ "Replica"
$\land$ LET $leader\_data \triangleq db[db\_leader[r]]$
    $new\_len \triangleq Len(db[r]) + 1$
    $leader \triangleq db\_leader[r]$
  IN
    $\land$ $Len(db[r]) < Len(leader\_data)$
    $\land$ $db' = [db \text{ EXCEPT } ![r] = Append(@, leader\_data[new\_len])]$
    $\land$ $db\_replicated' = [db\_replicated \text{ EXCEPT } ![leader][r] = new\_len]$
$\land$ UNCHANGED $\langle db\_leader, db\_epoch, db\_status\rangle$
$\land$ UNCHANGED $client\_vars$
$\land$ UNCHANGED zk_vars
$\land$ UNCHANGED $healer\_vars$

$new\_repl \triangleq [r \in Replica \mapsto 0]$

$initReplicated(r) \triangleq$
$\land$ $db\_replicated' = [$
  $db\_replicated \text{ EXCEPT } ![r] = [$
    $new\_repl \text{ EXCEPT } ![r] = Len(db[r])$
  $]]$

$dbStatusFromZK(r) \triangleq$
IF $zk\_status \in \{$"Normal", "WaitReplicaLog"$\} \land \neg(r \in old\_leaders)$
THEN IF $zk\_leader = r$
THEN "Writable"
ELSE "Replica"
ELSE "Frozen"

$DBUpdateLeader(r) \triangleq$
$\land\ db\_epoch[r] < zk\_epoch$
$\land\ db\_epoch' = [db\_epoch \text{ EXCEPT } ![r] = zk\_epoch]$
$\land\ db\_leader' = [db\_leader \text{ EXCEPT } ![r] = zk\_leader]$
$\land\ db\_status' = [db\_status \text{ EXCEPT } ![r] = dbStatusFromZK(r)]$
$\land$ IF $db\_leader[r] \neq zk\_leader$
THEN $initReplicated(r)$
ELSE UNCHANGED $db\_replicated$
$\land$ UNCHANGED $\langle db \rangle$
$\land$ UNCHANGED zk_vars
$\land$ UNCHANGED $client\_vars$
$\land$ UNCHANGED $healer\_vars$

$minOfSet(S) \triangleq$ CHOOSE $x \in S : \forall\, x1 \in S : x \leq x1$

$replicatedSet(r,\ Q) \triangleq \{db\_replicated[r][r1] : r1 \in Q\}$

$minReplicate(r,\ Q) \triangleq minOfSet(replicatedSet(r,\ Q))$

$DBResponse(c) \triangleq$
$\land\ pending[c] \neq nil$
$\land$ LET $leader \triangleq pending\_db[c]$ IN
$\land\ \exists\, index \in$ DOMAIN $db[leader],\ Q \in Quorum :$
$\land\ pending[c] = db[leader][index]$
$\land\ index \leq minReplicate(leader,\ Q)$
$\land\ pending' = [pending \text{ EXCEPT } ![c] = nil]$
$\land\ pending\_db' = [pending\_db \text{ EXCEPT } ![c] = nil]$
$\land\ client\_success' = [client\_success \text{ EXCEPT } ![c] = Append(@,\ pending[c])]$
$\land$ UNCHANGED db_vars
$\land$ UNCHANGED $next\_req$
$\land$ UNCHANGED $\langle client\_leader,\ client\_leader\_epoch \rangle$
$\land$ UNCHANGED zk_vars
$\land$ UNCHANGED $healer\_vars$

$ClientUpdateLeader(c) \triangleq$
$\land\ client\_leader\_epoch[c] < zk\_leader\_epoch$
$\land\ client\_leader\_epoch' = [client\_leader\_epoch \text{ EXCEPT } ![c] = zk\_leader\_epoch]$

$\land\ client\_leader' = [client\_leader \text{ EXCEPT } ![c] = zk\_leader]$
$\land\ pending' = [pending \text{ EXCEPT } ![c] = nil]$
$\land\ pending\_db' = [pending\_db \text{ EXCEPT } ![c] = nil]$
$\land$ UNCHANGED zk_vars
$\land$ UNCHANGED $healer\_vars$
$\land$ UNCHANGED db_vars
$\land$ UNCHANGED $\langle next\_req, client\_success \rangle$

$ReadyToChangeZKLeader \triangleq$
$\land\ zk\_leader\_epoch < max\_change\_leader$
$\land\ Cardinality(Replica \setminus old\_leaders) > replication\_factor$
$\land\ zk\_status =$ "Normal"

$\land\ zk\_status' =$ "ChangingLeader"
$\land\ zk\_epoch' = zk\_epoch + 1$
$\land\ old\_leaders' = old\_leaders \cup \{zk\_leader\}$

$\land$ UNCHANGED $\langle zk\_leader, zk\_leader\_epoch, zk\_catchup\_index \rangle$
$\land$ UNCHANGED $client\_vars$
$\land$ UNCHANGED db_vars
$\land$ UNCHANGED $healer\_vars$

$zkStatusToHealerStatus \triangleq$
IF $zk\_status =$ "ChangingLeader"
THEN "UpdatingLeader"
ELSE IF $zk\_status =$ "WaitReplicaLog"
THEN "WaitReplica"
ELSE "Init"

$HealerUpdateState \triangleq$
$\land\ healer\_epoch < zk\_epoch$
$\land\ healer\_epoch' = zk\_epoch$
$\land\ healer\_replicas' = [r \in Replica \mapsto nil]$
$\land\ healer\_status' = zkStatusToHealerStatus$
$\land$ UNCHANGED zk_vars
$\land$ UNCHANGED $client\_vars$
$\land$ UNCHANGED db_vars

$HealerGetDBLog(r) \triangleq$
$\land\ healer\_status =$ "UpdatingLeader"
$\land\ healer\_replicas[r] = nil$
$\land\ \neg(r \in old\_leaders)$
$\land\ db\_epoch[r] = healer\_epoch$

$\land\ healer\_replicas' = [healer\_replicas \text{ EXCEPT } ![r] = Len(db[r])]$
$\land\ \text{UNCHANGED } \langle healer\_status, healer\_epoch \rangle$
$\land\ \text{UNCHANGED } db\_vars$
$\land\ \text{UNCHANGED } client\_vars$
$\land\ \text{UNCHANGED } zk\_vars$

$collectedDB \triangleq \{r \in Replica : healer\_replicas[r] \neq nil\}$

$HealerUpdateLeader \triangleq$
$\land\ healer\_status = \text{“UpdatingLeader”}$
$\land\ healer\_epoch = zk\_epoch$
$\land\ Cardinality(collectedDB) \geq replication\_factor$
$\land\ \exists\, r \in collectedDB :$
$\quad\land\ \forall\, r1 \in collectedDB : healer\_replicas[r] \geq healer\_replicas[r1]$
$\quad\land\ zk\_leader' = r$
$\quad\land\ zk\_catchup\_index' = healer\_replicas[r]$
$\land\ zk\_status' = \text{“WaitReplicaLog”}$
$\land\ zk\_epoch' = zk\_epoch + 1$
$\land\ zk\_leader\_epoch' = zk\_leader\_epoch + 1$
$\land\ \text{UNCHANGED } old\_leaders$
$\land\ \text{UNCHANGED } healer\_vars$
$\land\ \text{UNCHANGED } db\_vars$
$\land\ \text{UNCHANGED } client\_vars$

$HealerUpdateToNormal \triangleq$
$\land\ healer\_status = \text{“WaitReplica”}$
$\land\ zk\_status = \text{“WaitReplicaLog”}$
$\land\ \exists\, Q \in Quorum :$
$\quad\land\ \neg(old\_leaders \subseteq Q)$
$\quad\land\ \forall\, r \in Q : Len(db[r]) \geq zk\_catchup\_index$
$\land\ zk\_status' = \text{“Normal”}$
$\land\ zk\_epoch' = zk\_epoch + 1$
$\land\ \text{UNCHANGED } \langle old\_leaders, zk\_leader\_epoch, zk\_leader, zk\_catchup\_index \rangle$
$\land\ \text{UNCHANGED } healer\_vars$
$\land\ \text{UNCHANGED } db\_vars$
$\land\ \text{UNCHANGED } client\_vars$

$RecoverOldLeader(r) \triangleq$
$\land\ r \in old\_leaders$
$\land\ zk\_status = \text{“Normal”}$
$\land\ db' = [db \text{ EXCEPT } ![r] = \langle\rangle]$
$\land\ db\_status' = [db\_status \text{ EXCEPT } ![r] = \text{“Replica”}]$
$\land\ db\_epoch' = [db\_epoch \text{ EXCEPT } ![r] = zk\_epoch]$
$\land\ db\_leader' = [db\_leader \text{ EXCEPT } ![r] = zk\_leader]$

$\land\ db\_replicated' = [db\_replicated \text{ EXCEPT } ![r] = new\_repl]$
$\land\ old\_leaders' = old\_leaders \setminus \{r\}$
$\land$ UNCHANGED $\langle zk\_epoch,\ zk\_leader,\ zk\_leader\_epoch,\ zk\_status,\ zk\_catchup\_index\rangle$
$\land$ UNCHANGED $client\_vars$
$\land$ UNCHANGED $healer\_vars$

$TerminateCond \triangleq$
$\land\ next\_req = 60 + max\_next\_req$
$\land\ zk\_status =$ "Normal"
$\land\ zk\_leader\_epoch = max\_change\_leader$
$\land\ zk\_epoch \geq 10$
$\land \forall\ c \in Client : pending[c] = nil \land pending\_db[c] = nil$
$\land \forall\ c \in Client : client\_leader\_epoch[c] = zk\_leader\_epoch$
$\land \forall\ r \in Replica : db\_epoch[r] = zk\_epoch$

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

$Next \triangleq$
$\lor \exists\ c \in Client :$
$\lor\ StartRequest(c)$
$\lor\ DBResponse(c)$
$\lor\ ClientUpdateLeader(c)$
$\lor \exists\ r \in Replica :$
$\lor\ Replicate(r)$
$\lor\ DBUpdateLeader(r)$
$\lor\ RecoverOldLeader(r)$
$\lor\ ReadyToChangeZKLeader$
$\lor\ HealerUpdateState$
$\lor \exists\ r \in Replica : HealerGetDBLog(r)$
$\lor\ HealerUpdateLeader$
$\lor\ HealerUpdateToNormal$
$\lor\ Terminated$

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

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

$Consistent \triangleq$
$\land \forall\ c \in Client :$
$\land\ Len(client\_success[c]) \leq Len(db[zk\_leader])$

$$\land \forall\, x \in Range(client\_success[c]) : x \in Range(db[zk\_leader])$$

$Perms \triangleq Permutations(Replica)$

$Inv \triangleq$
$\land\ zk\_epoch < 11$
$\land\ zk\_leader\_epoch \leq max\_change\_leader$
$\land\ (zk\_leader\_epoch \geq 2) \Rightarrow (\forall\, c \in Client:\ Len(client\_success[c]) < 4)$

$Finish \triangleq \Diamond TerminateCond$