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*
$\rangle$

$zk\_vars \triangleq \langle zk\_leader, zk\_epoch, zk\_leader\_epoch, zk\_status,$
$old\_leaders, zk\_catchup\_index\rangle$
$db\_vars \triangleq \langle db, db\_leader, db\_replicated, db\_epoch, db\_status\rangle$
$client\_vars \triangleq \langle$
$next\_req, client\_leader, client\_success,$
$pending, pending\_db, client\_leader\_epoch\rangle$
$healer\_vars \triangleq \langle healer\_status, healer\_epoch, healer\_replicas\rangle$

$max\_next\_req \triangleq 4$

$max\_change\_leader \triangleq 4$

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

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

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

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

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

$NullLogOffset \triangleq LogOffset \cup \{nil\}$

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

$replication\_factor \triangleq 2$

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

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

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

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

- $\land \quad healer\_status \in \{\text{“Init”}, \text{“UpdatingLeader”}, \text{“WaitReplica”}\}$
- $\land \quad healer\_epoch \in Epoch$
- $\land \quad healer\_replicas \in [Replica \to NullLogOffset]$

$Init \triangleq$
- $\land\ zk\_leader \in Replica$
- $\land\ zk\_epoch = 1$
- $\land\ zk\_leader\_epoch = 1$
- $\land\ zk\_status = \text{“Normal”}$
- $\land\ old\_leaders = \{\}$
- $\land\ zk\_catchup\_index = 0$

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

- $\land\ next\_req = 60$
- $\land\ client\_leader = [c \in Client \mapsto zk\_leader]$
- $\land\ client\_success = [c \in Client \mapsto \langle\rangle]$

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

$\land\ healer\_status = \text{“Init”}$
$\land\ healer\_epoch = 1$
$\land\ healer\_replicas = [r \in Replica \mapsto nil]$

$StartRequest(c) \triangleq$
$\land\ pending[c] = nil$
$\land\ next\_req < 60 + max\_next\_req$
$\land\ db\_status[client\_leader[c]] = \text{“Writable”}$
$\land\ next\_req' = next\_req + 1$
$\land\ pending' = [pending \text{ EXCEPT } ![c] = next\_req']$
$\land\ pending\_db' = [pending\_db \text{ EXCEPT } ![c] = client\_leader[c]]$
$\land$ LET $leader \triangleq client\_leader[c]$ IN
$\quad\land\ db' = [db \text{ EXCEPT } ![leader] = Append(@, next\_req')]$
$\quad\land\ db\_replicated' = [$
$\qquad db\_replicated \text{ EXCEPT } ![leader][leader] = Len(db'[leader])]$
$\land$ 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 $healer\_vars$

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

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

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

```
            new_repl EXCEPT ![r] = Len(db[r])
        ]]

dbStatusFromZK(r) ≜
    IF zk_status ∈ {"Normal", "WaitReplicaLog"} ∧ ¬(r ∈ old_leaders)
        THEN IF zk_leader = r
            THEN "Writable"
            ELSE "Replica"
        ELSE "Frozen"

DBUpdateLeader(r) ≜
    ∧ db_epoch[r] < zk_epoch
    ∧ db_epoch' = [db_epoch EXCEPT ![r] = zk_epoch]
    ∧ db_leader' = [db_leader EXCEPT ![r] = zk_leader]
    ∧ db_status' = [db_status EXCEPT ![r] = dbStatusFromZK(r)]
    ∧ IF db_leader[r] ≠ zk_leader
        THEN initReplicated(r)
        ELSE UNCHANGED db_replicated
    ∧ UNCHANGED ⟨db⟩
    ∧ UNCHANGED zk_vars
    ∧ UNCHANGED client_vars
    ∧ UNCHANGED healer_vars

minOfSet(S) ≜ CHOOSE x ∈ S : ∀ x1 ∈ S : x ≤ x1

replicatedSet(r, Q) ≜ {db_replicated[r][r1] : r1 ∈ Q}

minReplicate(r, Q) ≜ minOfSet(replicatedSet(r, Q))

DBResponse(c) ≜
    ∧ pending[c] ≠ nil
    ∧ LET leader ≜ pending_db[c]IN
        ∧ ∃ index ∈ DOMAIN db[leader], Q ∈ Quorum :
            ∧ pending[c] = db[leader][index]
            ∧ index ≤ minReplicate(leader, Q)
    ∧ pending' = [pending EXCEPT ![c] = nil]
    ∧ pending_db' = [pending_db EXCEPT ![c] = nil]
    ∧ client_success' = [client_success EXCEPT ![c] = Append(@, pending[c])]
    ∧ UNCHANGED db_vars
    ∧ UNCHANGED next_req
    ∧ UNCHANGED ⟨client_leader, client_leader_epoch⟩
    ∧ UNCHANGED zk_vars
    ∧ 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"

$\land\ healer\_replicas[r] = nil$
$\land\ \neg(r \in old\_leaders)$
$\land\ db\_epoch[r] = healer\_epoch$
$\land\ healer\_replicas' = [healer\_replicas \text{ EXCEPT } ![r] = Len(db[r])]$
$\land$ UNCHANGED $\langle healer\_status, healer\_epoch\rangle$
$\land$ UNCHANGED db_vars
$\land$ UNCHANGED $client\_vars$
$\land$ UNCHANGED zk_vars

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

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

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

$JoinOldOlder(r) \triangleq$
$\land\ r \in old\_leaders$
$\land\ zk\_status =$ "Normal"

$\land\ db\_epoch[r] = zk\_epoch$
$\land\ Len(db[r]) \leq zk\_catchup\_index$
$\land\ old\_leaders' = old\_leaders \setminus \{r\}$
$\land\ zk\_epoch' = zk\_epoch + 1$
$\land$ UNCHANGED db_vars
$\land$ UNCHANGED $\langle zk\_leader\_epoch,\ zk\_leader,\ zk\_status,\ zk\_catchup\_index\rangle$
$\land$ UNCHANGED $client\_vars$
$\land$ UNCHANGED $healer\_vars$

$TruncateOldLeader(r)\ \triangleq$
$\land\ r \in old\_leaders$
$\land\ zk\_status =$ "Normal"
$\land\ db\_status[r] =$ "Frozen"
$\land\ db[r] \neq \langle\rangle$
$\land\ db' = [db$ EXCEPT $![r] = \langle\rangle]$
$\land$ UNCHANGED $\langle db\_epoch,\ db\_leader,\ db\_replicated,\ db\_status\rangle$
$\land$ UNCHANGED zk_vars
$\land$ UNCHANGED $client\_vars$
$\land$ UNCHANGED $healer\_vars$

$checked\_max\_epoch\ \triangleq\ 14$

$TerminateCond\ \triangleq$
$\land\ next\_req = 60 + max\_next\_req$
$\land\ zk\_status =$ "Normal"
$\land\ zk\_leader\_epoch = max\_change\_leader$
$\land\ zk\_epoch \geq checked\_max\_epoch - 1$
$\land\ \forall\, c \in Client : pending[c] = nil \land pending\_db[c] = nil$
$\land\ \forall\, c \in Client : client\_leader\_epoch[c] = zk\_leader\_epoch$
$\land\ \forall\, r \in Replica : db\_epoch[r] = zk\_epoch$

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

$Next\ \triangleq$
$\lor\ \exists\, c \in Client :$
$\quad\lor\ StartRequest(c)$
$\quad\lor\ DBResponse(c)$
$\quad\lor\ ClientUpdateLeader(c)$
$\lor\ \exists\, r \in Replica :$
$\quad\lor\ Replicate(r)$
$\quad\lor\ DBUpdateLeader(r)$
$\quad\lor\ JoinOldOlder(r)$

$\lor\ TruncateOldLeader(r)$

$\lor\ ReadyToChangeZKLeader$
$\lor\ HealerUpdateState$
$\lor\ \exists\, r \in Replica : HealerGetDBLog(r)$
$\lor\ HealerUpdateLeader$
$\lor\ HealerUpdateToNormal$

$\lor\ Terminated$

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

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

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

$Perms \triangleq Permutations(Replica)$

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

$Finish \triangleq \Diamond TerminateCond$

$CanRequestCond \triangleq$
$\land\ zk\_leader\_epoch \geq max\_change\_leader$
$\land\ next\_req < 60 + max\_next\_req$

$CanStartReq(c) \triangleq$
$\land\ pending[c] = nil$
$\land\ db\_status[client\_leader[c]] =$ "Writable"

$CanRequest \triangleq$
$CanRequestCond \Rightarrow \exists\, c \in Client : CanStartReq(c)$

$StillCanRequest \triangleq \Box\Diamond CanRequest$

$WeakFairnessOfStartReq \triangleq \mathrm{WF}_{vars}(\exists\, c \in Client : StartRequest(c))$