
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 {
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*
}

zk_vars \triangleq {*zk_leader*, *zk_epoch*, *zk_leader_epoch*, *zk_status*,
old_leaders, *zk_catchup_index*}

db_vars \triangleq {*db*, *db_leader*, *db_replicated*, *db_epoch*, *db_status*}

client_vars \triangleq {
next_req, *client_leader*, *client_success*,
pending, *pending_db*, *client_leader_epoch*}

healer_vars \triangleq {*healer_status*, *healer_epoch*, *healer_replicas*}

max_next_req \triangleq 3

max_change_leader \triangleq 3

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 DOMAIN *f*}

$replication_factor \triangleq 2$

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

$TypeOK \triangleq$

$\wedge zk_leader \in Replica$
 $\wedge zk_epoch \in 1 \dots 30$
 $\wedge zk_leader_epoch \in 1 \dots max_change_leader$
 $\wedge zk_status \in \{ \text{"Normal"}, \text{"ChangingLeader"}, \text{"WaitReplicaLog"} \}$
 $\wedge old_leaders \subseteq Replica$
 $\wedge zk_catchup_index \in LogOffset$

 $\wedge db \in [Replica \rightarrow Seq(ReqSet)]$
 $\wedge db_leader \in [Replica \rightarrow Replica]$
 $\wedge db_replicated \in [Replica \rightarrow [Replica \rightarrow LogOffset]]$
 $\wedge db_epoch \in [Replica \rightarrow Epoch]$
 $\wedge db_status \in [Replica \rightarrow \{ \text{"Writable"}, \text{"Replica"}, \text{"Frozen"} \}]$

 $\wedge next_req \in ReqSet$
 $\wedge client_leader \in [Client \rightarrow Replica]$
 $\wedge client_success \in [Client \rightarrow Seq(ReqSet)]$
 $\wedge pending \in [Client \rightarrow NullReqSet]$
 $\wedge pending_db \in [Client \rightarrow NullReplica]$
 $\wedge client_leader_epoch \in [Client \rightarrow Epoch]$

 $\wedge healer_status \in \{ \text{"Init"}, \text{"UpdatingLeader"}, \text{"WaitReplica"} \}$
 $\wedge healer_epoch \in Epoch$
 $\wedge healer_replicas \in [Replica \rightarrow NullLogOffset]$

$Init \triangleq$

$\wedge zk_leader \in Replica$
 $\wedge zk_epoch = 1$
 $\wedge zk_leader_epoch = 1$
 $\wedge zk_status = \text{"Normal"}$
 $\wedge old_leaders = \{ \}$
 $\wedge zk_catchup_index = 0$

 $\wedge db = [r \in Replica \mapsto \langle \rangle]$
 $\wedge db_leader = [r \in Replica \mapsto zk_leader]$
 $\wedge db_replicated = [r \in Replica \mapsto [r1 \in Replica \mapsto 0]]$
 $\wedge db_epoch = [r \in Replica \mapsto zk_epoch]$
 $\wedge db_status = [r \in Replica \mapsto \text{IF } zk_leader = r \text{ THEN "Writable" ELSE "Replica"}]$

 $\wedge next_req = 60$
 $\wedge client_leader = [c \in Client \mapsto zk_leader]$
 $\wedge client_success = [c \in Client \mapsto \langle \rangle]$

$$\begin{aligned}
& \wedge \text{pending} = [c \in \text{Client} \mapsto \text{nil}] \\
& \wedge \text{pending_db} = [c \in \text{Client} \mapsto \text{nil}] \\
& \wedge \text{client_leader_epoch} = [c \in \text{Client} \mapsto \text{zk_leader_epoch}] \\
& \wedge \text{healer_status} = \text{"init"} \\
& \wedge \text{healer_epoch} = 1 \\
& \wedge \text{healer_replicas} = [r \in \text{Replica} \mapsto \text{nil}]
\end{aligned}$$

$$\begin{aligned}
\text{StartRequest}(c) & \triangleq \\
& \wedge \text{pending}[c] = \text{nil} \\
& \wedge \text{next_req} < 60 + \text{max_next_req} \\
& \wedge \text{db_status}[\text{client_leader}[c]] = \text{"Writable"} \\
& \wedge \text{next_req}' = \text{next_req} + 1 \\
& \wedge \text{pending}' = [\text{pending} \text{ EXCEPT } ![c] = \text{next_req}'] \\
& \wedge \text{pending_db}' = [\text{pending_db} \text{ EXCEPT } ![c] = \text{client_leader}[c]] \\
& \wedge \text{LET } \text{leader} \triangleq \text{client_leader}[c] \text{ IN} \\
& \quad \wedge \text{db}' = [\text{db} \text{ EXCEPT } ![\text{leader}] = \text{Append}(@, \text{next_req}')] \\
& \quad \wedge \text{db_replicated}' = [\\
& \quad \quad \text{db_replicated} \text{ EXCEPT } ![\text{leader}][\text{leader}] = \text{Len}(\text{db}'[\text{leader}])] \\
& \wedge \text{UNCHANGED } \langle \text{client_leader}, \text{client_success}, \text{client_leader_epoch} \rangle \\
& \wedge \text{UNCHANGED } \langle \text{db_leader}, \text{db_epoch}, \text{db_status} \rangle \\
& \wedge \text{UNCHANGED } \text{zk_vars} \\
& \wedge \text{UNCHANGED } \text{healer_vars}
\end{aligned}$$

$$\begin{aligned}
\text{Replicate}(r) & \triangleq \\
& \wedge r \neq \text{db_leader}[r] \\
& \wedge \text{db_status}[r] = \text{"Replica"} \\
& \wedge \text{LET } \text{leader_data} \triangleq \text{db}[\text{db_leader}[r]] \\
& \quad \text{new_len} \triangleq \text{Len}(\text{db}[r]) + 1 \\
& \quad \text{leader} \triangleq \text{db_leader}[r] \\
& \text{ IN} \\
& \quad \wedge \text{Len}(\text{db}[r]) < \text{Len}(\text{leader_data}) \\
& \quad \wedge \text{db}' = [\text{db} \text{ EXCEPT } ![r] = \text{Append}(@, \text{leader_data}[\text{new_len}])] \\
& \quad \wedge \text{db_replicated}' = [\text{db_replicated} \text{ EXCEPT } ![\text{leader}][r] = \text{new_len}] \\
& \wedge \text{UNCHANGED } \langle \text{db_leader}, \text{db_epoch}, \text{db_status} \rangle \\
& \wedge \text{UNCHANGED } \text{client_vars} \\
& \wedge \text{UNCHANGED } \text{zk_vars} \\
& \wedge \text{UNCHANGED } \text{healer_vars}
\end{aligned}$$

$$\text{new_repl} \triangleq [r \in \text{Replica} \mapsto 0]$$

$$\begin{aligned}
\text{initReplicated}(r) & \triangleq \\
& \wedge \text{db_replicated}' = [\\
& \quad \text{db_replicated} \text{ EXCEPT } ![r] = [
\end{aligned}$$

$new_repl \text{ EXCEPT } ![r] = Len(db[r])$
 $]]$

$dbStatusFromZK(r) \triangleq$
 IF $zk_status \in \{ \text{"Normal"}, \text{"WaitReplicaLog"} \} \wedge \neg(r \in old_leaders)$
 THEN IF $zk_leader = r$
 THEN "Writable"
 ELSE "Replica"
 ELSE "Frozen"

$DBUpdateLeader(r) \triangleq$
 $\wedge db_epoch[r] < zk_epoch$
 $\wedge db_epoch' = [db_epoch \text{ EXCEPT } ![r] = zk_epoch]$
 $\wedge db_leader' = [db_leader \text{ EXCEPT } ![r] = zk_leader]$
 $\wedge db_status' = [db_status \text{ EXCEPT } ![r] = dbStatusFromZK(r)]$
 \wedge IF $db_leader[r] \neq zk_leader$
 THEN $initReplicated(r)$
 ELSE UNCHANGED $db_replicated$
 \wedge UNCHANGED $\langle db \rangle$
 \wedge UNCHANGED zk_vars
 \wedge UNCHANGED $client_vars$
 \wedge UNCHANGED $healer_vars$

$minOfSet(S) \triangleq \text{CHOOSE } x \in S : \forall x1 \in S : x \leq x1$

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

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

$DBResponse(c) \triangleq$
 $\wedge pending[c] \neq nil$
 \wedge LET $leader \triangleq pending_db[c]$ IN
 $\wedge \exists index \in \text{DOMAIN } db[leader], Q \in Quorum :$
 $\wedge pending[c] = db[leader][index]$
 $\wedge index \leq minReplicate(leader, Q)$
 $\wedge pending' = [pending \text{ EXCEPT } ![c] = nil]$
 $\wedge pending_db' = [pending_db \text{ EXCEPT } ![c] = nil]$
 $\wedge client_success' = [client_success \text{ EXCEPT } ![c] = Append(@, pending[c])]$
 \wedge UNCHANGED db_vars
 \wedge UNCHANGED $next_req$
 \wedge UNCHANGED $\langle client_leader, client_leader_epoch \rangle$
 \wedge UNCHANGED zk_vars
 \wedge UNCHANGED $healer_vars$

$$\begin{aligned}
\text{ClientUpdateLeader}(c) &\triangleq \\
&\wedge \text{client_leader_epoch}[c] < \text{zk_leader_epoch} \\
&\wedge \text{client_leader_epoch}' = [\text{client_leader_epoch} \text{ EXCEPT } ![c] = \text{zk_leader_epoch}] \\
&\wedge \text{client_leader}' = [\text{client_leader} \text{ EXCEPT } ![c] = \text{zk_leader}] \\
&\wedge \text{pending}' = [\text{pending} \text{ EXCEPT } ![c] = \text{nil}] \\
&\wedge \text{pending_db}' = [\text{pending_db} \text{ EXCEPT } ![c] = \text{nil}] \\
&\wedge \text{UNCHANGED } \text{zk_vars} \\
&\wedge \text{UNCHANGED } \text{healer_vars} \\
&\wedge \text{UNCHANGED } \text{db_vars} \\
&\wedge \text{UNCHANGED } \langle \text{next_req}, \text{client_success} \rangle
\end{aligned}$$

$$\begin{aligned}
\text{ReadyToChangeZKLeader} &\triangleq \\
&\wedge \text{zk_leader_epoch} < \text{max_change_leader} \\
&\wedge \text{Cardinality}(\text{Replica} \setminus \text{old_leaders}) > \text{replication_factor} \\
&\wedge \text{zk_status} = \text{"Normal"} \\
&\wedge \text{zk_status}' = \text{"ChangingLeader"} \\
&\wedge \text{zk_epoch}' = \text{zk_epoch} + 1 \\
&\wedge \text{old_leaders}' = \text{old_leaders} \cup \{\text{zk_leader}\} \\
&\wedge \text{UNCHANGED } \langle \text{zk_leader}, \text{zk_leader_epoch}, \text{zk_catchup_index} \rangle \\
&\wedge \text{UNCHANGED } \text{client_vars} \\
&\wedge \text{UNCHANGED } \text{db_vars} \\
&\wedge \text{UNCHANGED } \text{healer_vars}
\end{aligned}$$

$$\begin{aligned}
\text{zkStatusToHealerStatus} &\triangleq \\
&\text{IF } \text{zk_status} = \text{"ChangingLeader"} \\
&\quad \text{THEN } \text{"UpdatingLeader"} \\
&\quad \text{ELSE IF } \text{zk_status} = \text{"WaitReplicaLog"} \\
&\quad \quad \text{THEN } \text{"WaitReplica"} \\
&\quad \quad \text{ELSE } \text{"Init"}
\end{aligned}$$

$$\begin{aligned}
\text{HealerUpdateState} &\triangleq \\
&\wedge \text{healer_epoch} < \text{zk_epoch} \\
&\wedge \text{healer_epoch}' = \text{zk_epoch} \\
&\wedge \text{healer_replicas}' = [r \in \text{Replica} \mapsto \text{nil}] \\
&\wedge \text{healer_status}' = \text{zkStatusToHealerStatus} \\
&\wedge \text{UNCHANGED } \text{zk_vars} \\
&\wedge \text{UNCHANGED } \text{client_vars} \\
&\wedge \text{UNCHANGED } \text{db_vars}
\end{aligned}$$

$$\begin{aligned}
\text{HealerGetDBLog}(r) &\triangleq \\
&\wedge \text{healer_status} = \text{"UpdatingLeader"}
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{healer_replicas}[r] = \text{nil} \\
& \wedge \neg(r \in \text{old_leaders}) \\
& \wedge \text{db_epoch}[r] = \text{healer_epoch} \\
& \wedge \text{healer_replicas}' = [\text{healer_replicas} \text{ EXCEPT } ![r] = \text{Len}(\text{db}[r])] \\
& \wedge \text{UNCHANGED } \langle \text{healer_status}, \text{healer_epoch} \rangle \\
& \wedge \text{UNCHANGED } \text{db_vars} \\
& \wedge \text{UNCHANGED } \text{client_vars} \\
& \wedge \text{UNCHANGED } \text{zk_vars}
\end{aligned}$$

$$\text{collectedDB} \triangleq \{r \in \text{Replica} : \text{healer_replicas}[r] \neq \text{nil}\}$$

$$\begin{aligned}
\text{HealerUpdateLeader} & \triangleq \\
& \wedge \text{healer_status} = \text{"UpdatingLeader"} \\
& \wedge \text{healer_epoch} = \text{zk_epoch} \\
& \wedge \text{Cardinality}(\text{collectedDB}) \geq \text{replication_factor} \\
& \wedge \exists r \in \text{collectedDB} : \\
& \quad \wedge \forall r1 \in \text{collectedDB} : \text{healer_replicas}[r] \geq \text{healer_replicas}[r1] \\
& \quad \wedge \text{zk_leader}' = r \\
& \quad \wedge \text{zk_catchup_index}' = \text{healer_replicas}[r] \\
& \wedge \text{zk_status}' = \text{"WaitReplicaLog"} \\
& \wedge \text{zk_epoch}' = \text{zk_epoch} + 1 \\
& \wedge \text{zk_leader_epoch}' = \text{zk_leader_epoch} + 1 \\
& \wedge \text{UNCHANGED } \text{old_leaders} \\
& \wedge \text{UNCHANGED } \text{healer_vars} \\
& \wedge \text{UNCHANGED } \text{db_vars} \\
& \wedge \text{UNCHANGED } \text{client_vars}
\end{aligned}$$

$$\begin{aligned}
\text{HealerUpdateToNormal} & \triangleq \\
& \wedge \text{healer_status} = \text{"WaitReplica"} \\
& \wedge \text{zk_status} = \text{"WaitReplicaLog"} \\
& \wedge \text{healer_epoch} = \text{zk_epoch} \\
& \wedge \exists Q \in \text{Quorum} : \\
& \quad \wedge \neg(\text{old_leaders} \subseteq Q) \\
& \quad \wedge \forall r \in Q : \text{Len}(\text{db}[r]) \geq \text{zk_catchup_index} \\
& \wedge \text{zk_status}' = \text{"Normal"} \\
& \wedge \text{zk_epoch}' = \text{zk_epoch} + 1 \\
& \wedge \text{UNCHANGED } \langle \text{old_leaders}, \text{zk_leader_epoch}, \text{zk_leader}, \text{zk_catchup_index} \rangle \\
& \wedge \text{UNCHANGED } \text{healer_vars} \\
& \wedge \text{UNCHANGED } \text{db_vars} \\
& \wedge \text{UNCHANGED } \text{client_vars}
\end{aligned}$$

$$\begin{aligned}
\text{JoinOldOlder}(r) & \triangleq \\
& \wedge r \in \text{old_leaders} \\
& \wedge \text{zk_status} = \text{"Normal"}
\end{aligned}$$

$$\begin{aligned}
& \wedge db_epoch[r] = zk_epoch \\
& \wedge Len(db[r]) \leq zk_catchup_index \\
& \wedge old_leaders' = old_leaders \setminus \{r\} \\
& \wedge zk_epoch' = zk_epoch + 1 \\
& \wedge \text{UNCHANGED } db_vars \\
& \wedge \text{UNCHANGED } \langle zk_leader_epoch, zk_leader, zk_status, zk_catchup_index \rangle \\
& \wedge \text{UNCHANGED } client_vars \\
& \wedge \text{UNCHANGED } healer_vars
\end{aligned}$$

$$\begin{aligned}
TruncateOldLeader(r) & \triangleq \\
& \wedge r \in old_leaders \\
& \wedge zk_status = \text{"Normal"} \\
& \wedge db_status[r] = \text{"Frozen"} \\
& \wedge db[r] \neq \langle \rangle \\
& \wedge db' = [db \text{ EXCEPT } ![r] = \langle \rangle] \\
& \wedge \text{UNCHANGED } \langle db_epoch, db_leader, db_replicated, db_status \rangle \\
& \wedge \text{UNCHANGED } zk_vars \\
& \wedge \text{UNCHANGED } client_vars \\
& \wedge \text{UNCHANGED } healer_vars
\end{aligned}$$

$$checked_max_epoch \triangleq 14$$

$$\begin{aligned}
TerminateCond & \triangleq \\
& \wedge next_req = 60 + max_next_req \\
& \wedge zk_status = \text{"Normal"} \\
& \wedge zk_leader_epoch = max_change_leader \\
& \wedge zk_epoch \geq checked_max_epoch - 1 \\
& \wedge \forall c \in Client : pending[c] = nil \wedge pending_db[c] = nil \\
& \wedge \forall c \in Client : client_leader_epoch[c] = zk_leader_epoch \\
& \wedge \forall r \in Replica : db_epoch[r] = zk_epoch
\end{aligned}$$

$$\begin{aligned}
Terminated & \triangleq \\
& \wedge TerminateCond \\
& \wedge \text{UNCHANGED } vars
\end{aligned}$$

$$\begin{aligned}
Next & \triangleq \\
& \vee \exists c \in Client : \\
& \quad \vee StartRequest(c) \\
& \quad \vee DBResponse(c) \\
& \quad \vee ClientUpdateLeader(c) \\
& \vee \exists r \in Replica : \\
& \quad \vee Replicate(r) \\
& \quad \vee DBUpdateLeader(r) \\
& \quad \vee JoinOldOlder(r)
\end{aligned}$$

$$\vee \text{TruncateOldLeader}(r)$$

$$\vee \text{ReadyToChangeZKLeader}$$

$$\vee \text{HealerUpdateState}$$

$$\vee \exists r \in \text{Replica} : \text{HealerGetDBLog}(r)$$

$$\vee \text{HealerUpdateLeader}$$

$$\vee \text{HealerUpdateToNormal}$$

$$\vee \text{Terminated}$$

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

$$\text{FairSpec} \triangleq \text{Spec} \wedge \text{WF}_{\text{vars}}(\text{Next})$$

$$\text{Consistent} \triangleq$$

$$\wedge \forall c \in \text{Client} :$$

$$\wedge \text{Len}(\text{client_success}[c]) \leq \text{Len}(\text{db}[\text{zk_leader}])$$

$$\wedge \forall x \in \text{Range}(\text{client_success}[c]) : x \in \text{Range}(\text{db}[\text{zk_leader}])$$

$$\text{Perms} \triangleq \text{Permutations}(\text{Replica})$$

$$\text{Inv} \triangleq$$

$$\wedge \text{zk_epoch} < \text{checked_max_epoch}$$

$$\wedge \text{zk_leader_epoch} \leq \text{max_change_leader}$$

$$\wedge (\text{zk_leader_epoch} \geq 2) \Rightarrow (\forall c \in \text{Client} : \text{Len}(\text{client_success}[c]) < 4)$$

$$\text{Finish} \triangleq \Diamond \text{TerminateCond}$$

$$\text{CanRequestCond} \triangleq$$

$$\wedge \text{zk_leader_epoch} \geq \text{max_change_leader}$$

$$\wedge \text{next_req} < 60 + \text{max_next_req}$$

$$\text{CanStartReq}(c) \triangleq$$

$$\wedge \text{pending}[c] = \text{nil}$$

$$\wedge \text{db_status}[\text{client_leader}[c]] = \text{"Writable"}$$

$$\text{CanRequest} \triangleq$$

$$\text{CanRequestCond} \Rightarrow \exists c \in \text{Client} : \text{CanStartReq}(c)$$

$$\text{StillCanRequest} \triangleq \Box \Diamond \text{CanRequest}$$

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