MODULE *Zab*

This is the formal specification for the *Zab* consensus algorithm,
which means *Zookeeper* Atomic *Broadcast*.

This work is driven by *Flavio P.* Junqueira,"*Zab*: High-performance broadcast for primary-backup systems"

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

The set of server identifiers
CONSTANT *Server*

The set of requests that can go into history
CONSTANT *Value*

Server states
CONSTANTS *Follower*, *Leader*, *ProspectiveLeader*

Message types
CONSTANTS *CEPOCH*, *NEWEPOCH*, *ACKE*, *NEWLEADER*, *ACKLD*, *COMMITLD*, *PROPOSE*, *ACK*, C

the maximum round of epoch (initially {0, 1, 2})
CONSTANT *Epoches*

Return the maximum value from the set S
$Maximum(S) \triangleq$ IF $S = \{\}$ THEN -1
ELSE CHOOSE $n \in S : \forall\, m \in S : n \geq m$

Return the minimum value from the set S
$Minimum(S) \triangleq$ IF $S = \{\}$ THEN -1
ELSE CHOOSE $n \in S : \forall\, m \in S : n \leq m$

$Quorums \triangleq \{Q \in$ SUBSET $Server : Cardinality(Q) * 2 > Cardinality(Server)\}$
ASSUME $QuorumsAssumption \triangleq \land \forall\, Q \in Quorums : Q \subseteq Server$
$\land \forall\, Q1,\ Q2 \in Quorums : Q1 \cap Q2 \neq \{\}$

Messages $\triangleq$ [*mtype*:{*CEPOCH*}, msource:*Server*, mdest:*Server*, *mepoch*:*Epoches*]
$\cup$
[*mtype*:{*NEWEPOCH*}, msource:*Server*, mdest:SUBSET *Server*, *mepoch*:*Epoches*]
$\cup$
[*mtype*:{*ACK_E*}, msource:*Server*, mdest: *Server*, *lastEpoch*:*Epoches*, hf:]

$None \triangleq$ CHOOSE $v : v \notin Value$

$NullPoint \triangleq$ CHOOSE $p : p \notin Server$

The server's *state*(*Follower*, *Leader*, *ProspectiveLeader*).
VARIABLE *state*

The leader's epoch or the last new epoch proposal the follower *acknowledged*(*f.p in paper*).
VARIABLE *currentEpoch*

The last new leader proposal the follower *acknowledged*(*f.a in paper*).
VARIABLE *leaderEpoch*

The identifier of the leader for followers.
VARIABLE *leaderOracle*

The history of servers as the sequence of transactions.
VARIABLE *history*

The messages repersenting requests and responses sent from one server to another.
msgs[*i*][*j*] means the input buffer of server *j* from server *i*.
VARIABLE *msgs*

The set of followers who has successfully sent *CEPOCH* to pleader in pleader.
VARIABLE *cepochRecv*

The set of followers who has successfully sent *ACK*-E to pleader in pleader.
VARIABLE *ackeRecv*

The set of followers who has successfully sent *ACK*-LD to pleader in pleader.
VARIABLE *ackldRecv*

ackIndex[*i*][*j*] means leader *i* has received how many *ACK* messages from follower *j*.
So *ackIndex*[*i*][*i*] is not used.
VARIABLE *ackIndex*

currentCounter[*i*] means leader *i* has proposed how many transactions
VARIABLE *currentCounter*

commitIndex[*i*] means leader/follower *i* has committed how many proposals and sent *COMMIT* messages.
VARIABLE *commitIndex*

Hepler matrix for follower to stop sending *CEPOCH* to pleader in followers.
Because *CEPOCH* is the sole message which follower actively sends to pleader.
VARIABLE *cepochSent*

the biggest epoch in *CEPOCH* pleader received from followers.
VARIABLE *tempMaxEpoch*

the biggest *leaderEpoch* and most up-to-date history in *ACKE* pleader received from followers.
VARIABLE *tempMaxLastEpoch*

VARIABLE *tempInitialHistory*

$serverVars \triangleq \langle state, currentEpoch, leaderEpoch, leaderOracle, history, commitIndex \rangle$
$leaderVars \triangleq \langle cepochRecv, ackeRecv, ackldRecv, ackIndex, currentCounter \rangle$
$tempVars \triangleq \langle tempMaxEpoch, tempMaxLastEpoch, tempInitialHistory \rangle$

$vars \triangleq \langle serverVars, msgs, leaderVars, cepochSent \rangle$

$$LastZxid(his) \triangleq \text{IF } Len(his) > 0 \text{ THEN } \langle his[Len(his)].epoch,\ his[Len(his)].counter\rangle \text{ ELSE } \langle -1,\ -1\rangle$$

Add a message to $msgs$ − add a message m to $msgs[i][j]$

$$Send(m) \triangleq msgs' = msgs \cup \{m\}$$
$$Send(i,\ j,\ m) \triangleq msgs' = [msgs \text{ EXCEPT } ![i][j] = Append(msgs[i][j],\ m)]$$

Remove a message from $msgs$ − discard head of $msgs[i][j]$

$$Discard(m) \triangleq msgs' = msgs \setminus \{m\}$$
$$Discard(i,\ j) \triangleq msgs' = \text{IF } msgs[i][j] \neq \langle\rangle \text{ THEN } [msgs \text{ EXCEPT } ![i][j] = Tail(msgs[i][j])] \text{ ELSE } msgs$$

Leader/Pleader broadcasts a message to all other servers

$$Broadcast(i,\ m) \triangleq msgs' = [ii \in Server \mapsto [ij \in Server \mapsto \text{IF } ii = i \wedge ij \neq i \text{ THEN } Append(msgs[ii][ij],\ m) \text{ ELSE } msgs[ii][ij]]]$$

Combination of $Send$ and $Discard$ − discard head of $msgs[j][i]$ and add m into $msgs[i][j]$

$$Reply(response,\ request) \triangleq msgs' = (msgs \cup \{response\}) \setminus \{request\}$$
$$Reply(i,\ j,\ m) \triangleq msgs' = [msgs \text{ EXCEPT } ![j][i] = Tail(msgs[j][i]),\ ![i][j] = Append(msgs[i][j],\ m)]$$

$$\begin{aligned} TypeOK \triangleq\ & \wedge state \in [Server \to \{Follower,\ Leader,\ ProspectiveLeader\}] \\ & \wedge currentEpoch \in [Server \to Epoches] \\ & \wedge leaderEpoch \in [Server \to Epoches] \\ & \wedge leaderOracle \in [Server \to Server] \end{aligned}$$

Define initial values for all variables

$$\begin{aligned} Init \triangleq\ & \wedge state && = [s \in Server \mapsto Follower] \\ & \wedge currentEpoch && = [s \in Server \mapsto 0] \\ & \wedge leaderEpoch && = [s \in Server \mapsto 0] \\ & \wedge leaderOracle && = [s \in Server \mapsto NullPoint] \\ & \wedge history && = [s \in Server \mapsto \langle\rangle] \\ & \wedge msgs && = [i \in Server \mapsto [j \in Server \mapsto \langle\rangle]] \\ & \wedge cepochRecv && = [s \in Server \mapsto \{\}] \\ & \wedge ackeRecv && = [s \in Server \mapsto \{\}] \\ & \wedge ackldRecv && = [s \in Server \mapsto \{\}] \\ & \wedge ackIndex && = [i \in Server \mapsto [j \in Server \mapsto 0]] \\ & \wedge currentCounter && = [i \in Server \mapsto 0] \\ & \wedge commitIndex && = [s \in Server \mapsto 0] \\ & \wedge cepochSent && = [s \in Server \mapsto \text{FALSE}] \\ & \wedge tempMaxEpoch && = [s \in Server \mapsto 0] \\ & \wedge tempMaxLastEpoch && = [s \in Server \mapsto 0] \\ & \wedge tempInitialHistory && = [s \in Server \mapsto \langle\rangle] \end{aligned}$$

A server becomes pleader and a quorum servers knows that.

In phase $f11$, follower sends $f.p$ to pleader via $CEPOCH$.

$FollowerDiscovery1(i) \triangleq$
$\wedge\ state[i] = Follower$
$\wedge\ leaderOracle[i] \neq NullPoint$
$\wedge\ \neg cepochSent[i]$
$\wedge$ LET $leader \triangleq leaderOracle[i]$
IN $Send(i,\ leader,\ [mtype \mapsto CEPOCH,$
$mepoch \mapsto currentEpoch[i]])$
$\wedge\ cepochSent' = [cepochSent$ EXCEPT $![i] =$ TRUE$]$
$\wedge$ UNCHANGED $\langle serverVars,\ leaderVars,\ tempVars\rangle$

In phase $l11$, pleader receives $CEPOCH$ from a quorum, and choose a new epoch e' as its own $l.p$ and sends $NEWEPOCH$ to followers.

$LeaderHandleCEPOCH(i,\ j) \triangleq$
$\wedge\ state[i] = ProspectiveLeader$
$\wedge\ msgs[j][i] \neq \langle\rangle$
$\wedge\ msgs[j][i][1].mtype = CEPOCH$
$\wedge\ \vee$ redundant message - just discard
$\wedge\ j \in cepochRecv[i]$
$\wedge$ UNCHANGED $\langle tempMaxEpoch,\ cepochRecv\rangle$
$\vee$ new message - modify $tempMaxEpoch$ and $cepochRecv$
$\wedge\ j \notin cepochRecv[i]$
$\wedge$ LET $newEpoch \triangleq Maximum(\{tempMaxEpoch[i],\ msgs[j][i][1].mepoch\})$
IN $tempMaxEpoch' = [tempMaxEpoch$ EXCEPT $![i] = newEpoch]$
$\wedge\ cepochRecv' = [cepochRecv$ EXCEPT $![i] = cepochRecv[i] \cup \{j\}]$
$\wedge\ Discard(j,\ i)$
$\wedge$ UNCHANGED $\langle serverVars,\ ackeRecv,\ ackldRecv,\ ackIndex,$
$currentCounter,\ cepochSent,\ tempMaxLastEpoch,\ tempInitialHistory\rangle$

$LeaderDiscovery1(i) \triangleq$
$\wedge\ state[i] = ProspectiveLeader$
$\wedge\ cepochRecv[i] \in Quorums$
$\wedge\ currentEpoch' = [currentEpoch$ EXCEPT $![i] = tempMaxEpoch[i] + 1]$
$\wedge\ cepochRecv' = [cepochRecv$ EXCEPT $![i] = \{\}]$
$\wedge\ Broadcast(i,\ [mtype \mapsto NEWEPOCH,$
$mepoch \mapsto currentEpoch'[i]])$
$\wedge$ UNCHANGED $\langle state,\ leaderEpoch,\ leaderOracle,\ history,\ ackeRecv,\ ackldRecv,$
$ackIndex,\ currentCounter,\ commitIndex,\ cepochSent,\ tempVars\rangle$

In phase $f12$, follower receives $NEWEPOCH$. If $e' > f.p$ then sends back $ACKE$, and $ACKE$ contains $f.a$ and hf to help pleader choose a newer history.

$FollowerDiscovery2(i,\ j) \triangleq$
$\wedge\ state[i] = Follower$
$\wedge\ msgs[j][i] \neq \langle\rangle$
$\wedge\ msgs[j][i][1].mtype = NEWEPOCH$
$\wedge$ LET $msg \triangleq msgs[j][i][1]$
IN $\vee$ new $NEWEPOCH-$ accept and reply

$\land\ currentEpoch[i] < msg.mepoch$
$\land\ currentEpoch' = [currentEpoch \text{ EXCEPT } ![i] = msg.mepoch]$
$\land\ leaderOracle' = [leaderOracle \text{ EXCEPT } ![i] = j]$
$\land\ Reply(i, j, [mtype \mapsto ACKE,$
$mepoch \mapsto msg.mepoch,$
$mlastEpoch \mapsto leaderEpoch[i],$
$mhf \mapsto history[i]])$
$\lor$ stale *NEWEPOCH − diacard*
$\land\ currentEpoch[i] \geq msg.mepoch$
$\land\ Discard(j, i)$
$\land$ UNCHANGED $\langle currentEpoch, leaderOracle\rangle$
$\land$ UNCHANGED $\langle state, leaderEpoch, history, leaderVars, commitIndex, cepochSent, tempVars\rangle$

In phase $l12$, pleader receives $ACKE$ from a quorum,
and select the history of one most up-to-date follower to be the initial history.

$LeaderHandleACKE(i, j) \triangleq$
$\land\ state[i] = ProspectiveLeader$
$\land\ msgs[j][i] \neq \langle\rangle$
$\land\ msgs[j][i][1].mtype = ACKE$
$\land$ LET $msg \triangleq msgs[j][i][1]$
$infoOk \triangleq \lor\ msg.mlastEpoch > tempMaxLastEpoch[i]$
$\lor\ \land\ msg.mlastEpoch = tempMaxLastEpoch[i]$
$\land\ \lor\ LastZxid(msg.mhf)[1] > LastZxid(tempInitialHistory[i])[1]$
$\lor\ \land\ LastZxid(msg.mhf)[1] = LastZxid(tempInitialHistory[i])[1]$
$\land\ LastZxid(msg.mhf)[2] \geq LastZxid(tempInitialHistory[i])[2]$
IN $\lor\ \land\ currentEpoch[i] = msg.mepoch$
$\land\ \lor\ \land\ infoOk$
$\land\ tempMaxLastEpoch' = [tempMaxLastEpoch \text{ EXCEPT } ![i] = msg.mlastEpoch]$
$\land\ tempInitialHistory' = [tempInitialHistory \text{ EXCEPT } ![i] = msg.mhf]$
$\lor\ \land\ \neg infoOk$
$\land$ UNCHANGED $\langle tempMaxLastEpoch, tempInitialHistory\rangle$
$\land\ ackeRecv' = [ackeRecv \text{ EXCEPT } ![i] = \text{IF } j \notin ackeRecv[i] \text{ THEN } ackeRecv[i] \cup \{j\}$
ELSE $ackeRecv[i]]$
$\lor\ \land\ currentEpoch[i] \neq msg.mepoch$
$\land$ UNCHANGED $\langle tempMaxLastEpoch, tempInitialHistory, ackeRecv\rangle$
$\land\ Discard(j, i)$
$\land$ UNCHANGED $\langle serverVars, cepochRecv, ackldRecv, ackIndex, currentCounter, cepochSent, tempMaxl$

$LeaderDiscovery2Sync1(i) \triangleq$
$\land\ state[i] = ProspectiveLeader$
$\land\ ackeRecv[i] \in Quorums$
$\land\ history' = [history \text{ EXCEPT } ![i] = tempInitialHistory[i]]$
$\land\ commitIndex' = [commitIndex \text{ EXCEPT } ![i] = 0]$
$\land\ ackeRecv' = [ackeRecv \text{ EXCEPT } ![i] = \{\}]$
until now, $phase1(Discovery)$ ends

$\land$ $Broadcast(i, [mtype \mapsto NEWLEADER,$
$mepoch \mapsto currentEpoch[i],$
$minitialHistory \mapsto history'[i]])$
$\land$ UNCHANGED $\langle state, currentEpoch, leaderEpoch, leaderOracle, cepochRecv, ackldRecv,$
$ackIndex, currentCounter, cepochSent, tempVars\rangle$

In phase $f21$, follower receives $NEWLEADER$. The follower updates its epoch and history, and send back ACK-LD to pleader.

$FollowerSync1(i, j) \triangleq$
$\land$ $state[i] = Follower$
$\land$ $msgs[j][i] \neq \langle\rangle$
$\land$ $msgs[j][i][1].mtype = NEWLEADER$
$\land$ LET $msg \triangleq msgs[j][i][1]$
IN $\lor$ new $NEWLEADER-$ accept and reply
$\land$ $currentEpoch[i] \leq msg.mepoch$
$\land$ $currentEpoch' = [currentEpoch$ EXCEPT $![i] = msg.mepoch]$
$\land$ $leaderEpoch' = [leaderEpoch$ EXCEPT $![i] = msg.mepoch]$
$\land$ $leaderOracle' = [leaderOracle$ EXCEPT $![i] = j]$
$\land$ $history' = [history$ EXCEPT $![i] = msg.minitialHistory]$
$\land$ $commitIndex' = [commitIndex$ EXCEPT $![i] = 0]$
$\land$ $Reply(i, j, [mtype \mapsto ACKLD,$
$mepoch \mapsto msg.mepoch])$
$\lor$ stale $NEWLEADER-$ discard
$\land$ $currentEpoch[i] > msg.mepoch$
$\land$ $Discard(j, i)$
$\land$ UNCHANGED $\langle currentEpoch, leaderEpoch, leaderOracle, history, commitIndex\rangle$
$\land$ UNCHANGED $\langle state, leaderVars, tempVars, cepochSent\rangle$

In phase $l22$, pleader receives ACK-LD from a quorum of followers, and sends $COMMIT$-LD to followers.

$LeaderHandleACKLD(i, j) \triangleq$
$\land$ $state[i] = ProspectiveLeader$
$\land$ $msgs[j][i] \neq \langle\rangle$
$\land$ $msgs[j][i][1].mtype = ACKLD$
$\land$ LET $msg \triangleq msgs[j][i][1]$
IN $\lor$ new ACK-LD - accept
$\land$ $currentEpoch[i] = msg.mepoch$
$\land$ $ackldRecv' = [ackldRecv$ EXCEPT $![i] =$ IF $j \notin ackldRecv[i]$ THEN $ackldRecv[i] \cup \{j\}$
ELSE $ackldRecv[i]]$
$\lor$ stale ACK-LD - impossible
$\land$ $currentEpoch[i] \neq msg.mepoch$
$\land$ UNCHANGED $ackldRecv$
$\land$ $Discard(j, i)$
$\land$ UNCHANGED $\langle serverVars, cepochRecv, ackeRecv, ackIndex, currentCounter, tempVars, cepochSent\rangle$

$LeaderSync2(i) \triangleq$
$\land$ $state[i] = ProspectiveLeader$

$\land$ $ackldRecv[i] \in Quorums$
$\land$ $commitIndex' = [commitIndex \text{ EXCEPT } ![i] = Len(history[i])]$
$\land$ $state' = [state \text{ EXCEPT } ![i] = Leader]$
$\land$ $Broadcast(i, [mtype \mapsto COMMITLD, mepoch \mapsto currentEpoch[i]])$
$\land$ UNCHANGED $\langle currentEpoch, leaderEpoch, leaderOracle, history, leaderVars, tempVars, cepochSer$

In phase $f22$, follower receives $COMMIT$-LD and submits all unprocessed transaction.

$FollowerSync2(i, j) \triangleq$
$\land$ $state[i] = Follower$
$\land$ $msgs[j][i] \neq \langle\rangle$
$\land$ $msgs[j][i][1].mtype = COMMITLD$
$\land$ LET $msg \triangleq msgs[j][i][1]$
IN $\lor$ new $COMMIT$-LD - commit all transactions in initial history
$\land$ $currentEpoch[i] = msg.mepoch$
$\land$ $commitIndex' = [commitIndex \text{ EXCEPT } ![i] = Len(history[i])]$
$\lor$ stale $COMMIT$-LD - discard
$\land$ $currentEpoch[i] \neq msg.mepoch$
$\land$ UNCHANGED $commitIndex$
$\land$ $Discard(j, i)$
$\land$ UNCHANGED $\langle state, currentEpoch, leaderEpoch, leaderOracle, history, leaderVars, tempVars, cepoch$

In phase $l31$, leader receives client request.

$ClientRequest(i, v) \triangleq$
$\land$ $state[i] = Leader$
$\land$ LET $newTransaction \triangleq [epoch \mapsto currentEpoch[i], counter \mapsto currentCounter[i]]$

$DiscoveryLeader1(i) \triangleq$
$\land$ $state[i] = ProspectiveLeader$
$\land$ $\neg\exists\, m \in msgs:$ $\land$ $m.mtype = NEWEPOCH$
$\land$ $m.msource = i$
$\land$ $m.mepoch = currentEpoch[i]$
$\land$ $\exists\, Q \in Quorums:$
LET $mset \triangleq \{m \in msgs : \land m.mtype = CEPOCH$
$\land$ $m.msource \in Q$
$\land$ $m.mdest = i\}$
$newEpoch \triangleq Maximum(\{m.mepoch : m \in mset\}) + 1$
IN $\land \forall\, s \in Q: \exists\, m \in mset: m.msource = s$
$\land$ $currentEpoch' = [currentEpoch \text{ EXCEPT } ![i] = newEpoch]$
$\land$ $leaderEpoch' = [leaderEpoch \text{ EXCEPT } ![i] = newEpoch]$
$\land$ $Send([mtype \mapsto NEWEPOCH, msource \mapsto i, mdest \mapsto Server \setminus \{i\}, mepoch \mapsto newEpoch])$
$\land$ UNCHANGED $\langle state, leaderOracle, history\rangle$

$DiscoveryFollower1(i) \triangleq$

$\land\ state[i] = Follower$
$\land\ leaderOracle[i] \neq NullPoint$
$\land$ LET $leader \triangleq leaderOracle[i]$
IN $\land \neg\exists\, m \in msgs$: $\land\ m.mtype = CEPOCH$
$\land\ m.msource = i$
$\land\ m.mdest = leader$
$\land\ m.mepoch = currentEpoch[i]$
$\land\ Send([mtype \mapsto CEPOCH,$
$msource \mapsto i,$
$mdest \mapsto leader,$
$mepoch \mapsto currentEpoch[i]])$
$\land$ UNCHANGED $\langle state, currentEpoch, leaderEpoch, leaderOracle, history\rangle$

$DiscoveryFollower2(i) \triangleq$
$\land\ state[i] = Follower$
$\land\ \exists\, m \in msgs$: $\land\ m.mtype = NEWEPOCH$
$\land\ i \in m.mdest$
$\land\ currentEpoch[i] < m.mepoch$
$\land\ leaderOracle' = [leaderOracle$ EXCEPT $![i] = m.msource]$
$\land\ currentEpoch' = [currentEpoch$ EXCEPT $![i] = m.mepoch]$
$\land$ LET $qm \triangleq [mtype \mapsto NEWEPOCH,$
$msource \mapsto m.msource,$
$mdest \mapsto m.mdest \setminus \{i\},$
$mepoch \mapsto m.mepoch]$
IN $msgs' = (msgs \setminus \{m\}) \cup \{qm\}$
$\land\ Send([mtype \mapsto ACKE,$
$msource \mapsto i,$
$mdest \mapsto m.msource,$
$lastEpoch \mapsto leaderEpoch[i],$
$hf \mapsto history[i]])$
$\land$ UNCHANGED $\langle state, leaderEpoch, history\rangle$

Integrity $\triangleq\ \forall\, l, f \in Server, msg \in msgs$:
$\land\ state[l] = Leader \land state[f] = Follower$
$\land\ msg.type = COMMIT \land msg \in histroy[f]$
$\Rightarrow msg \in history[l]$

Consistency $\triangleq\ \exists\, i, j \in Server$: $(state[i] = Leader)\ \land\ (state[j] = Leader)$
$\Rightarrow i = j$

$LivenessProperty1 \triangleq\ \forall\, i,\ j \in Server,\ msg \in msgs$: $(state[i] = Leader)\ \land\ (msg.type = COMMIT)\ \leadsto\ (msg \in history[j])\ \land\ (state[j] = Follower)$

* Modification History
* Last modified *Mon Mar* 15 14:23:01 *CST* 2021 by Dell
* Created Sat *Dec* 05 13:32:08 *CST* 2020 by Dell