
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*
 $Maximum(S) \triangleq \text{IF } S = \{\} \text{ THEN } -1$
 $\text{ELSE CHOOSE } n \in S : \forall m \in S : n \geq m$

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

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

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

$None \triangleq \text{CHOOSE } v : v \notin Value$

$NullPoint \triangleq \text{CHOOSE } p : p \notin Server$

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 representing 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 *ceepochRecv*

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 the count of transactions client requests leader.
VARIABLE *currentCounter*

sendCounter[i] means the count of transactions leader has broadcast.
VARIABLE *sendCounter*

initialHistory[i] means the initial history of leader *i* in epoch *currentEpoch[i]*.
VARIABLE *initialHistory*

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

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

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 *ceepochSent*

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

the maximum *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 $\triangleq \langle cepochRecv, ackeRecv, ackldRecv, ackIndex, currentCounter, sendCounter, initialHistory, com \rangle$

tempVars $\triangleq \langle tempMaxEpoch, tempMaxLastEpoch, tempInitialHistory \rangle$

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

LastZxid(*his*) \triangleq IF *Len*(*his*) > 0 THEN $\langle his[Len(his)].epoch, his[Len(his)].counter \rangle$
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*) $\triangleq msgs' =$ IF *msgs*[*i*][*j*] $\neq \langle \rangle$ THEN $[msgs \text{ EXCEPT } ![i][j] = Tail(msgs[i][j])]$
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*) $\triangleq msgs' = [msgs \text{ EXCEPT } ![j][i] = Tail(msgs[j][i]),$
 $![i][j] = Append(msgs[i][j], m)]$

Reply2(*i*, *j*, *m1*, *m2*) $\triangleq msgs' = [msgs \text{ EXCEPT } ![j][i] = Tail(msgs[j][i]),$
 $![i][j] = Append(Append(msgs[i][j], m1), m2)]$

TypeOK $\triangleq \wedge state \in [Server \rightarrow \{Follower, Leader, ProspectiveLeader\}]$

$\wedge currentEpoch \in [Server \rightarrow Epoches]$

$\wedge leaderEpoch \in [Server \rightarrow Epoches]$

$\wedge leaderOracle \in [Server \rightarrow Server]$

Define initial values for all variables

Init $\triangleq \wedge state = [s \in Server \mapsto Follower]$

$\wedge currentEpoch = [s \in Server \mapsto 0]$

$\wedge leaderEpoch = [s \in Server \mapsto 0]$

$\wedge leaderOracle = [s \in Server \mapsto NullPoint]$

$\wedge history = [s \in Server \mapsto \langle \rangle]$

$\wedge msgs = [i \in Server \mapsto [j \in Server \mapsto \langle \rangle]]$

$\wedge cepochRecv = [s \in Server \mapsto \{\}]$

$\wedge ackeRecv = [s \in Server \mapsto \{\}]$

$\wedge ackldRecv = [s \in Server \mapsto \{\}]$

$\wedge ackIndex = [i \in Server \mapsto [j \in Server \mapsto 0]]$

$$\begin{aligned}
\wedge \text{currentCounter} &= [s \in \text{Server} \mapsto 0] \\
\wedge \text{sendCounter} &= [s \in \text{Server} \mapsto 0] \\
\wedge \text{commitIndex} &= [s \in \text{Server} \mapsto 0] \\
\wedge \text{committedIndex} &= [s \in \text{Server} \mapsto 0] \\
\wedge \text{initialHistory} &= [s \in \text{Server} \mapsto \langle \rangle] \\
\wedge \text{cepocheSent} &= [s \in \text{Server} \mapsto \text{FALSE}] \\
\wedge \text{tempMaxEpoch} &= [s \in \text{Server} \mapsto 0] \\
\wedge \text{tempMaxLastEpoch} &= [s \in \text{Server} \mapsto 0] \\
\wedge \text{tempInitialHistory} &= [s \in \text{Server} \mapsto \langle \rangle]
\end{aligned}$$

A server becomes pleader and a quorum servers knows that.

$$\begin{aligned}
\text{Election}(i, Q) &\triangleq \\
&\wedge i \in Q \\
&\wedge \text{state}' = [s \in \text{Server} \mapsto \text{IF } s = i \text{ THEN } \text{ProspectiveLeader} \\
&\hspace{15em} \text{ELSE IF } s \in Q \text{ THEN } \text{Follower} \\
&\hspace{15em} \text{ELSE } \text{state}[s]] \\
&\wedge \text{cepocheRecv}' = [\text{cepocheRecv} \text{ EXCEPT } ![i] = \{i\}] \\
&\wedge \text{ackeRecv}' = [\text{ackeRecv} \text{ EXCEPT } ![i] = \{i\}] \\
&\wedge \text{ackldRecv}' = [\text{ackldRecv} \text{ EXCEPT } ![i] = \{i\}] \\
&\wedge \text{ackIndex}' = [ii \in \text{Server} \mapsto [ij \in \text{Server} \mapsto \\
&\hspace{10em} \text{IF } ii = i \text{ THEN } 0 \\
&\hspace{10em} \text{ELSE } \text{ackIndex}[ii][ij]]] \\
&\wedge \text{committedIndex}' = [\text{committedIndex} \text{ EXCEPT } ![i] = 0] \\
&\wedge \text{initialHistory}' = [\text{initialHistory} \text{ EXCEPT } ![i] = \langle \rangle] \\
&\wedge \text{tempMaxEpoch}' = [\text{tempMaxEpoch} \text{ EXCEPT } ![i] = \text{currentEpoch}[i]] \\
&\wedge \text{tempMaxLastEpoch}' = [\text{tempMaxLastEpoch} \text{ EXCEPT } ![i] = \text{currentEpoch}[i]] \\
&\wedge \text{tempInitialHistory}' = [\text{tempInitialHistory} \text{ EXCEPT } ![i] = \text{history}[i]] \\
&\wedge \text{leaderOracle}' = [s \in \text{Server} \mapsto \text{IF } s \in Q \text{ THEN } i \\
&\hspace{10em} \text{ELSE } \text{leaderOracle}[s]] \\
&\wedge \text{leaderEpoch}' = [s \in \text{Server} \mapsto \text{IF } s \in Q \text{ THEN } \text{currentEpoch}[s] \\
&\hspace{10em} \text{ELSE } \text{leaderEpoch}[s]] \\
&\wedge \text{cepocheSent}' = [s \in \text{Server} \mapsto \text{IF } s \in Q \text{ THEN } \text{FALSE} \\
&\hspace{10em} \text{ELSE } \text{cepocheSent}[s]] \\
&\wedge \text{msgs}' = [ii \in \text{Server} \mapsto [ij \in \text{Server} \mapsto \\
&\hspace{10em} \text{IF } ii \in Q \wedge ij \in Q \text{ THEN } \langle \rangle \\
&\hspace{10em} \text{ELSE } \text{msgs}[ii][ij]]] \\
&\wedge \text{UNCHANGED } \langle \text{currentEpoch}, \text{history}, \text{commitIndex}, \text{currentCounter}, \text{sendCounter} \rangle \\
\text{Restart}(i) &\triangleq \\
&\wedge \text{state}' = [\text{state} \text{ EXCEPT } ![i] = \text{Follower}] \\
&\wedge \text{leaderOracle}' = [\text{leaderOracle} \text{ EXCEPT } ![i] = \text{NullPoint}] \\
&\wedge \text{cepocheSent}' = [\text{cepocheSent} \text{ EXCEPT } ![i] = \text{FALSE}] \\
&\wedge \text{UNCHANGED } \langle \text{currentEpoch}, \text{leaderEpoch}, \text{history}, \text{commitIndex}, \text{leaderVars}, \text{tempVars}, \text{msgs} \rangle
\end{aligned}$$

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.

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

Here I decide to change leader's epoch in *l12*&*l21*, otherwise there may exist an old leader and a new leader who share the same epoch. So here I just change *leaderEpoch*, and use it in handling *ACK-E*.

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

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.

$FollowerDiscovery2(i, j) \triangleq$
 $\wedge state[i] = Follower$
 $\wedge msgs[j][i] \neq \langle \rangle$

$\wedge \text{msgs}[j][i][1].\text{mtype} = \text{NEWPOCH}$
 $\wedge \text{LET } \text{msg} \triangleq \text{msgs}[j][i][1]$
 IN \vee $\text{new NEWPOCH} - \text{accept and reply}$
 $\wedge \text{currentEpoch}[i] \leq \text{msg.mepoch}$ Here use \leq , because one follower may send CEPOCH more than o
 $\wedge \text{currentEpoch}' = [\text{currentEpoch} \text{ EXCEPT } ![i] = \text{msg.mepoch}]$
 $\wedge \text{leaderOracle}' = [\text{leaderOracle} \text{ EXCEPT } ![i] = j]$
 $\wedge \text{Reply}(i, j, [\text{mtype} \mapsto \text{ACKE},$
 $\text{mepoch} \mapsto \text{msg.mepoch},$
 $\text{mlastEpoch} \mapsto \text{leaderEpoch}[i],$
 $\text{mhf} \mapsto \text{history}[i]])$
 \vee $\text{stale NEWPOCH} - \text{diacard}$
 $\wedge \text{currentEpoch}[i] > \text{msg.mepoch}$
 $\wedge \text{Discard}(j, i)$
 $\wedge \text{UNCHANGED } \langle \text{currentEpoch}, \text{leaderOracle} \rangle$
 $\wedge \text{UNCHANGED } \langle \text{state}, \text{leaderEpoch}, \text{history}, \text{leaderVars}, \text{commitIndex}, \text{ceepochSent}, \text{tempVars} \rangle$

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.

$\text{LeaderHandleACKE}(i, j) \triangleq$
 $\wedge \text{state}[i] = \text{ProspectiveLeader}$
 $\wedge \text{msgs}[j][i] \neq \langle \rangle$
 $\wedge \text{msgs}[j][i][1].\text{mtype} = \text{ACKE}$
 $\wedge \text{LET } \text{msg} \triangleq \text{msgs}[j][i][1]$
 $\text{infoOk} \triangleq \vee \text{msg.mlastEpoch} > \text{tempMaxLastEpoch}[i]$
 $\vee \wedge \text{msg.mlastEpoch} = \text{tempMaxLastEpoch}[i]$
 $\wedge \vee \text{LastZxid}(\text{msg.mhf})[1] > \text{LastZxid}(\text{tempInitialHistory}[i])[1]$
 $\vee \wedge \text{LastZxid}(\text{msg.mhf})[1] = \text{LastZxid}(\text{tempInitialHistory}[i])[1]$
 $\wedge \text{LastZxid}(\text{msg.mhf})[2] \geq \text{LastZxid}(\text{tempInitialHistory}[i])[2]$
 IN $\vee \wedge \text{leaderEpoch}[i] = \text{msg.mepoch}$
 $\wedge \vee \wedge \text{infoOk}$
 $\wedge \text{tempMaxLastEpoch}' = [\text{tempMaxLastEpoch} \text{ EXCEPT } ![i] = \text{msg.mlastEpoch}]$
 $\wedge \text{tempInitialHistory}' = [\text{tempInitialHistory} \text{ EXCEPT } ![i] = \text{msg.mhf}]$
 $\vee \wedge \neg \text{infoOk}$
 $\wedge \text{UNCHANGED } \langle \text{tempMaxLastEpoch}, \text{tempInitialHistory} \rangle$
 $\wedge \text{ackRecv}' = [\text{ackRecv} \text{ EXCEPT } ![i] = \text{IF } j \notin \text{ackRecv}[i] \text{ THEN } \text{ackRecv}[i] \cup \{j\}$
 $\text{ELSE } \text{ackRecv}[i]]$
 $\vee \wedge \text{leaderEpoch}[i] \neq \text{msg.mepoch}$
 $\wedge \text{UNCHANGED } \langle \text{tempMaxLastEpoch}, \text{tempInitialHistory}, \text{ackRecv} \rangle$
 $\wedge \text{Discard}(j, i)$
 $\wedge \text{UNCHANGED } \langle \text{serverVars}, \text{ceepochRecv}, \text{ackldRecv}, \text{ackIndex}, \text{currentCounter},$
 $\text{sendCounter}, \text{initialHistory}, \text{committedIndex}, \text{ceepochSent}, \text{tempMaxEpoch} \rangle$

$\text{LeaderDiscovery2Sync1}(i) \triangleq$
 $\wedge \text{state}[i] = \text{ProspectiveLeader}$
 $\wedge \text{ackRecv}[i] \in \text{Quorums}$

$$\begin{aligned}
& \wedge \text{currentEpoch}' = [\text{currentEpoch} \text{ EXCEPT } ![i] = \text{leaderEpoch}[i]] \\
& \wedge \text{history}' = [\text{history} \text{ EXCEPT } ![i] = \text{tempInitialHistory}[i]] \\
& \wedge \text{initialHistory}' = [\text{initialHistory} \text{ EXCEPT } ![i] = \text{tempInitialHistory}[i]] \\
& \wedge \text{commitIndex}' = [\text{commitIndex} \text{ EXCEPT } ![i] = 0] \\
& \wedge \text{ackRecv}' = [\text{ackRecv} \text{ EXCEPT } ![i] = \{\}] \\
& \wedge \text{ackIndex}' = [\text{ackIndex} \text{ EXCEPT } ![i] = \text{Len}(\text{tempInitialHistory}[i])] \\
& \text{until now, phase1(Discovery) ends} \\
& \wedge \text{Broadcast}(i, [\text{mtype} \mapsto \text{NEWLEADER}, \\
& \quad \text{mepoch} \mapsto \text{currentEpoch}[i], \\
& \quad \text{minitialHistory} \mapsto \text{history}'[i]]) \\
& \wedge \text{UNCHANGED } \langle \text{state}, \text{leaderEpoch}, \text{leaderOracle}, \text{ceepochRecv}, \text{ackldRecv}, \\
& \quad \text{currentCounter}, \text{sendCounter}, \text{committedIndex}, \text{ceepochSent}, \text{tempVars} \rangle
\end{aligned}$$

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

$$\begin{aligned}
& \text{FollowerSync1}(i, j) \triangleq \\
& \quad \wedge \text{state}[i] = \text{Follower} \\
& \quad \wedge \text{msgs}[j][i] \neq \langle \rangle \\
& \quad \wedge \text{msgs}[j][i][1].\text{mtype} = \text{NEWLEADER} \\
& \quad \wedge \text{LET } \text{msg} \triangleq \text{msgs}[j][i][1] \\
& \quad \text{IN } \vee \text{ new NEWLEADER - accept and reply} \\
& \quad \quad \wedge \text{currentEpoch}[i] \leq \text{msg.mepoch} \\
& \quad \quad \wedge \text{currentEpoch}' = [\text{currentEpoch} \text{ EXCEPT } ![i] = \text{msg.mepoch}] \\
& \quad \quad \wedge \text{leaderEpoch}' = [\text{leaderEpoch} \text{ EXCEPT } ![i] = \text{msg.mepoch}] \\
& \quad \quad \wedge \text{leaderOracle}' = [\text{leaderOracle} \text{ EXCEPT } ![i] = j] \\
& \quad \quad \wedge \text{history}' = [\text{history} \text{ EXCEPT } ![i] = \text{msg.minitialHistory}] \\
& \quad \quad \wedge \text{commitIndex}' = [\text{commitIndex} \text{ EXCEPT } ![i] = 0] \\
& \quad \quad \wedge \text{Reply}(i, j, [\text{mtype} \mapsto \text{ACKLD}, \\
& \quad \quad \quad \text{mepoch} \mapsto \text{msg.mepoch}, \\
& \quad \quad \quad \text{mhistory} \mapsto \text{msg.minitialHistory}]) \\
& \quad \vee \text{ stale NEWLEADER - discard} \\
& \quad \quad \wedge \text{currentEpoch}[i] > \text{msg.mepoch} \\
& \quad \quad \wedge \text{Discard}(j, i) \\
& \quad \quad \wedge \text{UNCHANGED } \langle \text{currentEpoch}, \text{leaderEpoch}, \text{leaderOracle}, \text{history}, \text{commitIndex} \rangle \\
& \quad \wedge \text{UNCHANGED } \langle \text{state}, \text{leaderVars}, \text{tempVars}, \text{ceepochSent} \rangle
\end{aligned}$$

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

$$\begin{aligned}
& \text{LeaderHandleACKLD}(i, j) \triangleq \\
& \quad \wedge \text{state}[i] = \text{ProspectiveLeader} \\
& \quad \wedge \text{msgs}[j][i] \neq \langle \rangle \\
& \quad \wedge \text{msgs}[j][i][1].\text{mtype} = \text{ACKLD} \\
& \quad \wedge \text{LET } \text{msg} \triangleq \text{msgs}[j][i][1] \\
& \quad \text{IN } \vee \text{ new ACK-LD - accept} \\
& \quad \quad \wedge \text{currentEpoch}[i] = \text{msg.mepoch}
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{ackIndex}' = [\text{ackIndex} \text{ EXCEPT } ![i][j] = \text{Len}(\text{initialHistory}[i])] \\
& \wedge \text{ackldRecv}' = [\text{ackldRecv} \text{ EXCEPT } ![i] = \text{IF } j \notin \text{ackldRecv}[i] \text{ THEN } \text{ackldRecv}[i] \cup \{j\} \\
& \hspace{15em} \text{ELSE } \text{ackldRecv}[i]] \\
& \vee \text{stale ACK-LD - impossible} \\
& \wedge \text{currentEpoch}[i] \neq \text{msg.mepoch} \\
& \wedge \text{UNCHANGED } \langle \text{ackldRecv}, \text{ackIndex} \rangle \\
& \wedge \text{Discard}(j, i) \\
& \wedge \text{UNCHANGED } \langle \text{serverVars}, \text{ceepochRecv}, \text{ackRecv}, \text{currentCounter}, \\
& \hspace{10em} \text{sendCounter}, \text{initialHistory}, \text{committedIndex}, \text{tempVars}, \text{ceepochSent} \rangle \\
\text{LeaderSync2}(i) & \triangleq \\
& \wedge \text{state}[i] = \text{ProspectiveLeader} \\
& \wedge \text{ackldRecv}[i] \in \text{Quorums} \\
& \wedge \text{commitIndex}' = [\text{commitIndex} \text{ EXCEPT } ![i] = \text{Len}(\text{history}[i])] \\
& \wedge \text{committedIndex}' = [\text{committedIndex} \text{ EXCEPT } ![i] = \text{Len}(\text{history}[i])] \\
& \wedge \text{state}' = [\text{state} \text{ EXCEPT } ![i] = \text{Leader}] \\
& \wedge \text{currentCounter}' = [\text{currentCounter} \text{ EXCEPT } ![i] = 0] \\
& \wedge \text{sendCounter}' = [\text{sendCounter} \text{ EXCEPT } ![i] = 0] \\
& \wedge \text{ackldRecv}' = [\text{ackldRecv} \text{ EXCEPT } ![i] = \{\}] \\
& \wedge \text{Broadcast}(i, [\text{mtype} \mapsto \text{COMMITLD}, \\
& \hspace{10em} \text{mepoch} \mapsto \text{currentEpoch}[i]]) \\
& \wedge \text{UNCHANGED } \langle \text{currentEpoch}, \text{leaderEpoch}, \text{leaderOracle}, \text{history}, \text{ceepochRecv}, \\
& \hspace{10em} \text{ackRecv}, \text{ackIndex}, \text{initialHistory}, \text{tempVars}, \text{ceepochSent} \rangle
\end{aligned}$$

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

$$\begin{aligned}
\text{FollowerSync2}(i, j) & \triangleq \\
& \wedge \text{state}[i] = \text{Follower} \\
& \wedge \text{msgs}[j][i] \neq \langle \rangle \\
& \wedge \text{msgs}[j][i][1].\text{mtype} = \text{COMMITLD} \\
& \wedge \text{LET } \text{msg} \triangleq \text{msgs}[j][i][1] \\
& \text{IN } \vee \text{new COMMIT-LD - commit all transactions in initial history} \\
& \hspace{10em} \wedge \text{currentEpoch}[i] = \text{msg.mepoch} \\
& \hspace{10em} \wedge \text{commitIndex}' = [\text{commitIndex} \text{ EXCEPT } ![i] = \text{Len}(\text{history}[i])] \\
& \hspace{10em} \wedge \text{leaderOracle}' = [\text{leaderOracle} \text{ EXCEPT } ![i] = j] \\
& \vee \text{stale COMMIT-LD - discard} \\
& \hspace{10em} \wedge \text{currentEpoch}[i] \neq \text{msg.mepoch} \\
& \hspace{10em} \wedge \text{UNCHANGED } \langle \text{commitIndex}, \text{leaderOracle} \rangle \\
& \wedge \text{Discard}(j, i) \\
& \wedge \text{UNCHANGED } \langle \text{state}, \text{currentEpoch}, \text{leaderOracle}, \text{history}, \text{leaderVars}, \text{tempVars}, \text{ceepochSent} \rangle
\end{aligned}$$

In phase *l31*, leader receives client request and broadcasts *PROPOSE*.

$$\begin{aligned}
\text{ClientRequest}(i, v) & \triangleq \\
& \wedge \text{state}[i] = \text{Leader} \\
& \wedge \text{currentCounter}' = [\text{currentCounter} \text{ EXCEPT } ![i] = \text{currentCounter}[i] + 1] \\
& \wedge \text{LET } \text{newTransaction} \triangleq [\text{epoch} \mapsto \text{currentEpoch}[i],
\end{aligned}$$

$$\begin{aligned}
& \text{counter} \mapsto \text{currentCounter}'[i], \\
& \text{value} \mapsto v] \\
\text{IN } & \wedge \text{history}' = [\text{history} \text{ EXCEPT } ![i] = \text{Append}(\text{history}[i], \text{newTransaction})] \\
& \wedge \text{ackIndex}' = [\text{ackIndex} \text{ EXCEPT } ![i] = \text{Len}(\text{history}'[i])] \\
& \wedge \text{UNCHANGED } \langle \text{msgs}, \text{state}, \text{currentEpoch}, \text{leaderEpoch}, \text{leaderOracle}, \text{commitIndex}, \text{ceepochRecv}, \\
& \quad \text{ackRecv}, \text{ackldRecv}, \text{sendCounter}, \text{initialHistory}, \text{committedIndex}, \text{tempVars}, \text{ceepoch} \rangle \\
\text{LeaderBroadcast1}(i) \triangleq & \\
& \wedge \text{state}[i] = \text{Leader} \\
& \wedge \text{sendCounter}[i] < \text{currentCounter}[i] \\
& \wedge \text{LET } \text{toBeSentCounter} \triangleq \text{sendCounter}[i] + 1 \\
& \quad \text{toBeSentIndex} \triangleq \text{Len}(\text{initialHistory}[i]) + \text{toBeSentCounter} \\
& \quad \text{toBeSentEntry} \triangleq \text{history}[i][\text{toBeSentIndex}] \\
\text{IN } & \wedge \text{Broadcast}(i, [\text{mtype} \mapsto \text{PROPOSE}, \\
& \quad \text{mepoch} \mapsto \text{currentEpoch}[i], \\
& \quad \text{mproposal} \mapsto \text{toBeSentEntry}]) \\
& \wedge \text{sendCounter}' = [\text{sendCounter} \text{ EXCEPT } ![i] = \text{toBeSentCounter}] \\
& \wedge \text{UNCHANGED } \langle \text{serverVars}, \text{ceepochRecv}, \text{ackRecv}, \text{ackldRecv}, \text{ackIndex}, \\
& \quad \text{currentCounter}, \text{initialHistory}, \text{committedIndex}, \text{tempVars}, \text{ceepochSent} \rangle
\end{aligned}$$

In phase *f31*, follower accepts proposal and append it to history.

$$\begin{aligned}
\text{FollowerBroadcast1}(i, j) \triangleq & \\
& \wedge \text{state}[i] = \text{Follower} \\
& \wedge \text{msgs}[j][i] \neq \langle \rangle \\
& \wedge \text{msgs}[j][i][1].\text{mtype} = \text{PROPOSE} \\
& \wedge \text{LET } \text{msg} \triangleq \text{msgs}[j][i][1] \\
\text{IN } & \vee \text{ It should be that } \text{msg.mproposal.counter} = 1 \vee \text{msg.mproposal.counter} = \text{history}[\text{Len}(\text{history})].\text{counter} + 1 \\
& \wedge \text{currentEpoch}[i] = \text{msg.mepoch} \\
& \wedge \text{history}' = [\text{history} \text{ EXCEPT } ![i] = \text{Append}(\text{history}[i], \text{msg.mproposal})] \\
& \wedge \text{leaderOracle}' = [\text{leaderOracle} \text{ EXCEPT } ![i] = j] \\
& \wedge \text{Reply}(i, j, [\text{mtype} \mapsto \text{ACK}, \\
& \quad \text{mepoch} \mapsto \text{currentEpoch}[i], \\
& \quad \text{mindex} \mapsto \text{Len}(\text{history}'[i])]) \\
& \vee \text{ If happens, } \neq \text{ must be } >, \text{ namely a stale leader sends it.} \\
& \wedge \text{currentEpoch}[i] \neq \text{msg.mepoch} \\
& \wedge \text{Discard}(j, i) \\
& \wedge \text{UNCHANGED } \langle \text{history}, \text{leaderOracle} \rangle \\
& \wedge \text{UNCHANGED } \langle \text{state}, \text{currentEpoch}, \text{leaderEpoch}, \text{commitIndex}, \text{leaderVars}, \text{tempVars}, \text{ceepochSent} \rangle
\end{aligned}$$

In phase *l32*, leader receives ack from a quorum of followers to a certain proposal, and commits the proposal.

$$\begin{aligned}
\text{LeaderHandleACK}(i, j) \triangleq & \\
& \wedge \text{state}[i] = \text{Leader} \\
& \wedge \text{msgs}[j][i] \neq \langle \rangle \\
& \wedge \text{msgs}[j][i][1].\text{mtype} = \text{ACK} \\
& \wedge \text{LET } \text{msg} \triangleq \text{msgs}[j][i][1]
\end{aligned}$$

IN \vee $\text{There should be } \text{ackIndex}[i][j] + 1 \triangleq \text{msg.mindex}$
 $\wedge \text{currentEpoch}[i] = \text{msg.mepoch}$
 $\wedge \text{ackIndex}' = [\text{ackIndex} \text{ EXCEPT } ![i][j] = \text{Maximum}(\{\text{ackIndex}[i][j], \text{msg.mindex}\})]$
 \vee $\text{If happens, } \neq \text{ must be } >, \text{ namely a stale follower sends it.}$
 $\wedge \text{currentEpoch}[i] \neq \text{msg.mepoch}$
 $\wedge \text{UNCHANGED } \text{ackIndex}$
 $\wedge \text{Discard}(j, i)$
 $\wedge \text{UNCHANGED } \langle \text{serverVars}, \text{ceepochRecv}, \text{ackRecv}, \text{ackldRecv}, \text{currentCounter},$
 $\text{sendCounter}, \text{initialHistory}, \text{committedIndex}, \text{tempVars}, \text{ceepochSent} \rangle$

$\text{LeaderAdvanceCommit}(i) \triangleq$
 $\wedge \text{state}[i] = \text{Leader}$
 $\wedge \text{commitIndex}[i] < \text{Len}(\text{history}[i])$
 $\wedge \text{LET } \text{Agree}(\text{index}) \triangleq \{i\} \cup \{k \in \text{Server} : \text{ackIndex}[i][k] \geq \text{index}\}$
 $\text{agreeIndexes} \triangleq \{\text{index} \in (\text{commitIndex}[i] + 1) \dots \text{Len}(\text{history}[i]) : \text{Agree}(\text{index}) \in \text{Quorum}\}$
 $\text{newCommitIndex} \triangleq \text{IF } \text{agreeIndexes} \neq \{\} \text{ THEN } \text{Maximum}(\text{agreeIndexes})$
 $\text{ELSE } \text{commitIndex}[i]$
 IN $\text{commitIndex}' = [\text{commitIndex} \text{ EXCEPT } ![i] = \text{newCommitIndex}]$
 $\wedge \text{UNCHANGED } \langle \text{state}, \text{currentEpoch}, \text{leaderEpoch}, \text{leaderOracle}, \text{history},$
 $\text{msgs}, \text{leaderVars}, \text{tempVars}, \text{ceepochSent} \rangle$

$\text{LeaderBroadcast2}(i) \triangleq$
 $\wedge \text{state}[i] = \text{Leader}$
 $\wedge \text{committedIndex}[i] < \text{commitIndex}[i]$
 $\wedge \text{LET } \text{newCommittedIndex} \triangleq \text{committedIndex}[i] + 1$
 IN $\wedge \text{Broadcast}(i, [\text{mtype} \mapsto \text{COMMIT},$
 $\text{mepoch} \mapsto \text{currentEpoch}[i],$
 $\text{mindex} \mapsto \text{newCommittedIndex},$
 $\text{mcounter} \mapsto \text{history}[\text{newCommittedIndex}].\text{counter}])$
 $\wedge \text{committedIndex}' = [\text{committedIndex} \text{ EXCEPT } ![i] = \text{committedIndex}[i] + 1]$
 $\wedge \text{UNCHANGED } \langle \text{serverVars}, \text{ceepochRecv}, \text{ackRecv}, \text{ackldRecv}, \text{ackIndex}, \text{currentCounter},$
 $\text{sendCounter}, \text{initialHistory}, \text{tempVars}, \text{ceepochSent} \rangle$

In phase $f32$, follower receives COMMIT and commits transaction.

$\text{FollowerBroadcast2}(i, j) \triangleq$
 $\wedge \text{state}[i] = \text{Follower}$
 $\wedge \text{msgs}[j][i] \neq \langle \rangle$
 $\wedge \text{msgs}[j][i][1].\text{mtype} = \text{COMMIT}$
 $\wedge \text{LET } \text{msg} \triangleq \text{msgs}[j][i][1]$
 IN \vee $\text{new } \text{COMMIT} - \text{commit transaction in history}$
 $\wedge \text{currentEpoch}[i] = \text{msg.mepoch}$
 $\wedge \text{commitIndex}' = [\text{commitIndex} \text{ EXCEPT } ![i] = \text{Maximum}(\{\text{commitIndex}[i], \text{msg.mindex}\})]$
 $\wedge \text{leaderOracle}' = [\text{leaderOracle} \text{ EXCEPT } ![i] = j]$
 \vee $\text{stale } \text{COMMIT} - \text{discard}$
 $\wedge \text{currentEpoch}[i] \neq \text{msg.mepoch}$
 $\wedge \text{UNCHANGED } \langle \text{commitIndex}, \text{leaderOracle} \rangle$

$\wedge \text{Discard}(j, i)$
 $\wedge \text{UNCHANGED } \langle \text{state}, \text{currentEpoch}, \text{leaderEpoch}, \text{history},$
 $\text{leaderVars}, \text{tempVars}, \text{epochSent} \rangle$

There may be two ways to make sure all followers as up-to-date as the leader.
way1: choose *Send* not *Broadcast* when leader is going to send *PROPOSE* and *COMMIT*.
way2: When one follower receives *PROPOSE* or *COMMIT* which misses some entries between
 its history and the newest entry, the follower send *CEPOCH* to catch pace.
 Here I choose *way2*, which I need not to rewrite *PROPOSE* and *COMMIT*, but need to
 modify the code when follower receives *NEWLEADER* and *COMMIT*.

In phase *l33*, upon receiving *CEPOCH*, leader *l* proposes back *NEWEPOCH* and *NEWLEADER*.
 $\text{LeaderHandleCEPOCHinPhase3}(i, j) \triangleq$

$\wedge \text{state}[i] = \text{Leader}$
 $\wedge \text{msgs}[j][i] \neq \langle \rangle$
 $\wedge \text{msgs}[j][i][1].\text{mtype} = \text{CEPOCH}$
 $\wedge \text{LET } \text{msg} \triangleq \text{msgs}[j][i][1]$
 IN $\vee \wedge \text{currentEpoch}[i] \geq \text{msg.mepoch}$
 $\wedge \text{Reply2}(i, j, [\text{mtype} \mapsto \text{NEWEPOCH},$
 $\text{mepoch} \mapsto \text{currentEpoch}[i],$
 $[\text{mtype} \mapsto \text{NEWLEADER},$
 $\text{mepoch} \mapsto \text{currentEpoch}[i],$
 $\text{minitialHistory} \mapsto \text{history}[i]])$
 $\vee \wedge \text{currentEpoch}[i] < \text{msg.mepoch}$
 $\wedge \text{UNCHANGED } \text{msgs}$
 $\wedge \text{UNCHANGED } \langle \text{serverVars}, \text{leaderVars}, \text{tempVars}, \text{epochSent} \rangle$

In phase *l34*, upon receiving ack from *f* of the *NEWLEADER*, it sends a commit message to *f*.
 Leader *l* also makes $Q := Q \cup \{f\}$.

$\text{LeaderHandleACKLDinPhase3}(i, j) \triangleq$
 $\wedge \text{state}[i] = \text{Leader}$
 $\wedge \text{msgs}[j][i] \neq \langle \rangle$
 $\wedge \text{msgs}[j][i][1].\text{mtype} = \text{ACKLD}$
 $\wedge \text{LET } \text{msg} \triangleq \text{msgs}[j][i][1]$
 $\text{aimCommitIndex} \triangleq \text{Minimum}(\{\text{commitIndex}[i], \text{Len}(\text{msg.mhistory})\})$
 IN $\vee \wedge \text{currentEpoch}[i] = \text{msg.mepoch}$
 $\wedge \text{ackIndex}' = [\text{ackIndex} \text{ EXCEPT } ![i][j] = \text{Len}(\text{msg.mhistory})]$
 $\wedge \text{Reply}(i, j, [\text{mtype} \mapsto \text{COMMIT},$
 $\text{mepoch} \mapsto \text{currentEpoch}[i],$
 $\text{mindex} \mapsto \text{aimCommitIndex},$
 $\text{mcounter} \mapsto \text{history}[\text{aimCommitIndex}].\text{counter}])$
 $\vee \wedge \text{currentEpoch}[i] \neq \text{msg.mepoch}$
 $\wedge \text{Discard}(j, i)$
 $\wedge \text{UNCHANGED } \langle \text{ackIndex} \rangle$
 $\wedge \text{UNCHANGED } \langle \text{serverVars}, \text{epochRecv}, \text{ackRecv}, \text{ackldRecv}, \text{currentCounter}, \text{sendCounter},$

$initialHistory, committedIndex, tempVars, cepochSent\rangle$

To ensure any follower can find the correct leader, the follower should modify *leaderOracle* anytime when it receive messages from leader, because a server may restart and join the cluster *Q* halfway and receive the first message which is not *NEWEPOCH*. But we can delete this restriction when we ensure *Broadcast* function acts on the followers in the cluster not any servers in the whole system, then one server must has correct *leaderOracle* before it receives messages.

Let me suppose two conditions when one follower sends *CEPOCH* to leader:

0. Usually, the server becomes follower in election and sends *CEPOCH* before receiving *NEWEPOCH*.
1. The follower wants to join the cluster halfway and get the newest history.
2. The follower has received *COMMIT*, but there exists the gap between its own history and *mindex*, which means there are some transactions before *mindex* miss. Here we choose to send *CEPOCH* again, to receive the newest history from leader.

$$\begin{aligned}
 BecomeFollower(i) &\triangleq \\
 &\wedge \exists j \in Server \setminus \{i\} : \wedge msgs[j][i] \neq \langle \rangle \\
 &\quad \wedge LET \ msg \triangleq \ msgs[j][i][1] \\
 &\quad IN \quad \wedge currentEpoch[i] < msg.mepoch \\
 &\quad \quad \wedge \vee msg.mtype = NEWEPOCH \\
 &\quad \quad \vee msg.mtype = NEWLEADER \\
 &\quad \quad \vee msg.mtype = COMMITLD \\
 &\quad \quad \vee msg.mtype = PROPOSE \\
 &\quad \quad \vee msg.mtype = COMMIT \\
 &\quad \wedge state' = [state \quad EXCEPT ![i] = Follower] \\
 &\quad \wedge currentEpoch' = [currentEpoch \quad EXCEPT ![i] = msg.mepoch] \\
 &\quad \wedge leaderOracle' = [leaderOracle \quad EXCEPT ![i] = j] \\
 &\quad \quad \text{Here we should not use Discard.} \\
 &\wedge UNCHANGED \langle leaderEpoch, history, commitIndex, msgs, leaderVars, tempVars, cepochSent \rangle
 \end{aligned}$$

$$\begin{aligned}
 DiscardStaleMessage(i) &\triangleq \\
 &\wedge \exists j \in Server \setminus \{i\} : \wedge msgs[j][i] \neq \langle \rangle \\
 &\quad \wedge LET \ msg \triangleq \ msgs[j][i][1] \\
 &\quad IN \quad \vee \wedge state[i] = Follower \\
 &\quad \quad \wedge \vee msg.mepoch < currentEpoch[i] \\
 &\quad \quad \vee msg.mtype = CEPOCH \\
 &\quad \quad \vee msg.mtype = ACKE \\
 &\quad \quad \vee msg.mtype = ACKLD \\
 &\quad \quad \vee msg.mtype = ACK \\
 &\quad \vee \wedge state[i] = Leader \\
 &\quad \quad \wedge msg.mtype \neq CEPOCH \\
 &\quad \quad \wedge \vee msg.mepoch < currentEpoch[i] \\
 &\quad \quad \quad \vee msg.mtype = ACKE \quad \text{response of NEWEPOCH} \\
 &\quad \vee \wedge state[i] = ProspectiveLeader \\
 &\quad \quad \wedge msg.mtype \neq CEPOCH \\
 &\quad \quad \wedge \vee msg.mepoch < currentEpoch[i]
 \end{aligned}$$

$$\begin{aligned}
& \vee msg.mtype = ACK \\
& \wedge Discard(j, i) \\
& \wedge \text{UNCHANGED } \langle serverVars, leaderVars, tempVars, cepochSent \rangle
\end{aligned}$$

Defines how the variables may transition.

$Next \triangleq$

$$\begin{aligned}
& \vee \exists i \in Server : Restart(i) \\
& \vee \exists i \in Server, Q \in Quorums : Election(i, Q) \\
& \vee \exists i \in Server : FollowerDiscovery1(i) \\
& \vee \exists i, j \in Server : LeaderHandleCEPOCH(i, j) \\
& \vee \exists i \in Server : LeaderDiscovery1(i) \\
& \vee \exists i, j \in Server : FollowerDiscovery2(i, j) \\
& \vee \exists i, j \in Server : LeaderHandleACKE(i, j) \\
& \vee \exists i \in Server : LeaderDiscovery2Sync1(i) \\
& \vee \exists i, j \in Server : FollowerSync1(i, j) \\
& \vee \exists i, j \in Server : LeaderHandleACKLD(i, j) \\
& \vee \exists i \in Server : LeaderSync2(i) \\
& \vee \exists i, j \in Server : FollowerSync2(i, j) \\
& \vee \exists i \in Server, v \in Value : ClientRequest(i, v) \\
& \vee \exists i \in Server : LeaderBroadcast1(i) \\
& \vee \exists i, j \in Server : FollowerBroadcast1(i, j) \\
& \vee \exists i, j \in Server : LeaderHandleACK(i, j) \\
& \vee \exists i \in Server : LeaderAdvanceCommit(i) \\
& \vee \exists i \in Server : LeaderBroadcast2(i) \\
& \vee \exists i, j \in Server : FollowerBroadcast2(i, j) \\
& \vee \exists i, j \in Server : LeaderHandleCEPOCHinPhase3(i, j) \\
& \vee \exists i, j \in Server : LeaderHandleACKLDinPhase3(i, j) \\
& \vee \exists i \in Server : DiscardStaleMessage(i) \\
& \vee \exists i \in Server : BecomeFollower(i)
\end{aligned}$$

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

Defines some variants, safety propoties, and liveness propoties of zab consensus algorithm.

$Consistency \triangleq$

$$\begin{aligned}
& \exists i, j \in Server : \\
& \quad \wedge state[i] = Leader \\
& \quad \wedge state[j] = Leader \\
& \quad \wedge currentEpoch[i] = currentEpoch[j] \\
& \quad \Rightarrow i = j
\end{aligned}$$

$DiscoveryLeader1(i) \triangleq$

$$\begin{aligned}
& \wedge state[i] = ProspectiveLeader \\
& \wedge \neg \exists m \in msgs : \wedge m.mtype = NEWPOCH \\
& \quad \wedge m.msource = i
\end{aligned}$$

$$\begin{aligned}
& \wedge m.mepoch = currentEpoch[i] \\
& \wedge \exists Q \in Quorums: \\
& \quad LET \ mset \triangleq \{m \in msgs : \wedge m.mtype = CEPOCH \\
& \quad \quad \wedge m.msource \in Q \\
& \quad \quad \wedge m.mdest = i\} \\
& \quad newEpoch \triangleq Maximum(\{m.mepoch : m \in mset\}) + 1 \\
& IN \quad \wedge \forall s \in Q: \exists m \in mset: m.msource = s \\
& \quad \wedge currentEpoch' = [currentEpoch \text{ EXCEPT } ![i] = newEpoch] \\
& \quad \wedge leaderEpoch' = [leaderEpoch \text{ EXCEPT } ![i] = newEpoch] \\
& \quad \wedge Send([mtype \mapsto NEWEPOCH, \\
& \quad \quad msource \mapsto i, \\
& \quad \quad mdest \mapsto Server \setminus \{i\}, \\
& \quad \quad mepoch \mapsto newEpoch]) \\
& \wedge UNCHANGED \langle state, leaderOracle, history \rangle \\
\\
DiscoveryFollower1(i) & \triangleq \\
& \wedge state[i] = Follower \\
& \wedge leaderOracle[i] \neq NullPoint \\
& \wedge LET \ leader \triangleq leaderOracle[i] \\
& IN \quad \wedge \neg \exists m \in msgs: \wedge m.mtype = CEPOCH \\
& \quad \quad \wedge m.msource = i \\
& \quad \quad \wedge m.mdest = leader \\
& \quad \quad \wedge m.mepoch = currentEpoch[i] \\
& \quad \wedge Send([mtype \mapsto CEPOCH, \\
& \quad \quad msource \mapsto i, \\
& \quad \quad mdest \mapsto leader, \\
& \quad \quad mepoch \mapsto currentEpoch[i]]) \\
& \wedge UNCHANGED \langle state, currentEpoch, leaderEpoch, leaderOracle, history \rangle \\
\\
DiscoveryFollower2(i) & \triangleq \\
& \wedge state[i] = Follower \\
& \wedge \exists m \in msgs: \wedge m.mtype = NEWEPOCH \\
& \quad \wedge i \in m.mdest \\
& \quad \wedge currentEpoch[i] < m.mepoch \\
& \quad \wedge leaderOracle' = [leaderOracle \text{ EXCEPT } ![i] = m.msource] \\
& \quad \wedge currentEpoch' = [currentEpoch \text{ EXCEPT } ![i] = m.mepoch] \\
& \quad \wedge LET \ qm \triangleq [mtype \mapsto NEWEPOCH, \\
& \quad \quad msource \mapsto m.msource, \\
& \quad \quad mdest \mapsto m.mdest \setminus \{i\}, \\
& \quad \quad mepoch \mapsto m.mepoch] \\
& IN \quad msgs' = (msgs \setminus \{m\}) \cup \{qm\} \\
& \quad \wedge Send([mtype \mapsto ACKE, \\
& \quad \quad msource \mapsto i, \\
& \quad \quad mdest \mapsto m.msource, \\
& \quad \quad lastEpoch \mapsto leaderEpoch[i], \\
& \quad \quad hf \mapsto history[i]]) \\
& \wedge UNCHANGED \langle state, leaderEpoch, history \rangle \\
\\
Integrity & \triangleq \forall l, f \in Server, msg \in msgs: \\
& \quad \wedge state[l] = Leader \wedge state[f] = Follower
\end{aligned}$$

```

       $\wedge msg.type = COMMIT \wedge msg \in histroy[f]$ 
       $\Rightarrow msg \in history[l]$ 
Consistency  $\triangleq \exists i, j \in Server: (state[i] = Leader) \wedge (state[j] = Leader)$ 
       $\Rightarrow i = j$ 
LivenessProperty1  $\triangleq \forall i, j \in Server, msg \in msgs: (state[i] = Leader) \wedge (msg.type =$ 
       $COMMIT) \leadsto (msg \in history[j]) \wedge (state[j] = Follower)$ 

```

```

\ * Modification History
\ * Last modified Tue Mar 30 22:14:50 CST 2021 by Dell
\ * Created Sat Dec 05 13:32:08 CST 2020 by Dell

```