MODULE *ZeusReliableCommit*

Specification of *Zeus*'s reliable commit protocol presented in the *Zeus* paper
that appears in *Eurosys*'21.
This module includes everything but the pipelining optimization presented in the paper.

Model check passed [@ 21*st* of *Jan* 2021] with the following parameters:
$R\_NODES = \{0, 1, 2\}$
$R\_MAX\_EPOCH = 4$
$R\_MAX\_VERSION = 4$

EXTENDS *Integers*

```
CONSTANTS R_NODES,
          R_MAX_EPOCH,
          R_MAX_VERSION

VARIABLES rMsgs,
          rKeyState,
          rKeySharers,
          rKeyVersion,
          rKeyRcvedACKs,
          rKeyLastWriter,
          rNodeEpochID,
          rAliveNodes,
          rEpochID

vars ≜ ⟨rMsgs, rKeyState, rKeySharers, rKeyVersion, rKeyRcvedACKs,
        rKeyLastWriter, rNodeEpochID, rAliveNodes, rEpochID⟩
```

The consistent invariant: all alive nodes in valid state should have the same value / *TS*

```
RConsistentInvariant ≜
    ∀ k, s ∈ rAliveNodes :  ∨ rKeyState[k] ≠ "valid"
                            ∨ rKeyState[s] ≠ "valid"
                            ∨ rKeyVersion[k] = rKeyVersion[s]

RMaxVersionDistanceInvariant ≜     this does not hold w/ the pipelining optimization
    ∀ k, s ∈ rAliveNodes :
                            ∨ rKeyVersion[k] ≤ rKeyVersion[s] + 1
                            ∨ rKeyVersion[s] ≤ rKeyVersion[k] + 1

RSingleOnwerInvariant ≜
    ∀ k, s ∈ rAliveNodes :
                            ∨ rKeySharers[k] ≠ "owner"
                            ∨ rKeySharers[s] ≠ "owner"
                            ∨ k = s

ROnwerOnlyWriterInvariant ≜
```

```
    $\forall k \in rAliveNodes :$
                        $\lor rKeyState[k] \neq$ "write"
                        $\lor rKeySharers[k] =$ "owner"

$ROnwerHighestVersionInvariant \triangleq$   owner has the highest version among alive nodes
    $\forall k, s \in rAliveNodes :$
                        $\lor \land rKeySharers[s] \neq$ "owner"
                          $\land rKeySharers[k] \neq$ "owner"
                        $\lor$
                          $\land rKeySharers[k] =$ "owner"
                          $\land rKeyVersion[k] \geq rKeyVersion[s]$
                        $\lor$
                          $\land rKeySharers[s] =$ "owner"
                          $\land rKeyVersion[s] \geq rKeyVersion[k]$
```

```
$RMessage \triangleq$   Messages exchanged by the Protocol
    $[type : \{$"INV", "ACK"$\},$ $sender$   $: R\_NODES,$
                         $epochID$  $: 0 \mathrel{..} R\_MAX\_EPOCH,$
                         $version$  $: 0 \mathrel{..} R\_MAX\_VERSION]$
        $\cup$
    $[type : \{$"VAL"$\},$    $epochID$  $: 0 \mathrel{..} R\_MAX\_EPOCH,$
                         $version$  $: 0 \mathrel{..} R\_MAX\_VERSION]$

$RTypeOK \triangleq$   The type correctness invariant
    $\land$  $rMsgs$          $\subseteq RMessage$
    $\land$  $rAliveNodes$    $\subseteq R\_NODES$
    $\land$  $\forall n \in R\_NODES : rKeyRcvedACKs[n] \subseteq (R\_NODES \setminus \{n\})$
    $\land$  $rNodeEpochID$   $\in [R\_NODES \rightarrow 0 \mathrel{..} R\_MAX\_EPOCH]$
    $\land$  $rKeyLastWriter$ $\in [R\_NODES \rightarrow R\_NODES]$
    $\land$  $rKeyVersion$    $\in [R\_NODES \rightarrow 0 \mathrel{..} R\_MAX\_VERSION]$
    $\land$  $rKeySharers$    $\in [R\_NODES \rightarrow \{$"owner", "reader", "non-sharer"$\}]$
    $\land$  $rKeyState$      $\in [R\_NODES \rightarrow \{$"valid", "invalid", "write", "replay"$\}]$

$RInit \triangleq$   The initial predicate
    $\land$ $rMsgs$          $= \{\}$
    $\land$ $rEpochID$       $= 0$
    $\land$ $rAliveNodes$    $= R\_NODES$
    $\land$ $rKeyVersion$    $= [n \in R\_NODES \mapsto 0]$
    $\land$ $rNodeEpochID$   $= [n \in R\_NODES \mapsto 0]$
    $\land$ $rKeyRcvedACKs$  $= [n \in R\_NODES \mapsto \{\}]$
    $\land$ $rKeySharers$    $= [n \in R\_NODES \mapsto$ "reader"$]$
    $\land$ $rKeyState$      $= [n \in R\_NODES \mapsto$ "valid"$]$
    $\land$ $rKeyLastWriter$ $= [n \in R\_NODES \mapsto$ CHOOSE $k \in R\_NODES :$
```

```
                                        $\forall\, m \in R\_NODES : k \leq m]$
```

```
$RNoChanges\_in\_membership \triangleq$ UNCHANGED $\langle rAliveNodes,\ rEpochID \rangle$

$RNoChanges\_but\_membership \triangleq$
    UNCHANGED $\langle rMsgs,\ rKeyState,\ rKeyVersion,$
              $rKeyRcvedACKs,\ rKeyLastWriter,$
              $rKeySharers,\ rNodeEpochID \rangle$

$RNoChanges \triangleq$
    $\land\ RNoChanges\_in\_membership$
    $\land\ RNoChanges\_but\_membership$
```

```
 A buffer maintaining all network messages. Messages are only appended to
 this variable (not \* removed once delivered) intentionally to check
 protocol's tolerance in dublicates and reorderings
$RSend(m) \triangleq rMsgs' = rMsgs \cup \{m\}$

 Check if all acknowledgments for a write have been received
$RAllACKsRcved(n) \triangleq (rAliveNodes \setminus \{n\}) \subseteq rKeyRcvedACKs[n]$

$RIsAlive(n) \triangleq n \in rAliveNodes$

$RNodeFailure(n) \triangleq$   Emulate a node failure
   Make sure that there are atleast 3 alive nodes before killing a node
    $\land \exists\, k,\ m \in rAliveNodes :$ $\land\ k \neq n$
                                $\land\ m \neq n$
                                $\land\ m \neq k$
    $\land\ rEpochID' = rEpochID + 1$
    $\land\ rAliveNodes' = rAliveNodes \setminus \{n\}$
    $\land\ RNoChanges\_but\_membership$
```

```
$RNewOwner(n) \triangleq$
    $\land \forall\, k \in rAliveNodes :$
        $\land\ rKeySharers[k] \neq$ "owner"
        $\land \lor \land\ rKeyState[k] =$ "valid"          all alive replicas are in valid state
              $\land\ rKeySharers[k] =$ "reader"       and there is not alive owner
           $\lor \land\ rKeySharers[k] =$ "non-sharer"   and there is not alive owner
    $\land\ rKeySharers' = [rKeySharers$ EXCEPT $![n] =$ "owner"$]$
    $\land$ UNCHANGED $\langle rMsgs,\ rKeyState,\ rKeyVersion,\ rKeyRcvedACKs,$
                $rKeyLastWriter,\ rAliveNodes,\ rNodeEpochID,\ rEpochID \rangle$

$ROverthrowOwner(n) \triangleq$
    $\exists\, k \in rAliveNodes :$
```

```
        $\land$ $rKeyState[k]$     = "valid"
        $\land$ $rKeySharers[k]$ = "owner"
        $\land$ $rKeySharers'$     = [$rKeySharers$ EXCEPT ![$n$] = "owner",
                                                            ![$k$] = "reader"]
        $\land$ UNCHANGED $\langle rMsgs, rKeyState, rKeyVersion, rKeyRcvedACKs,$
                          $rKeyLastWriter, rAliveNodes, rNodeEpochID, rEpochID \rangle$

$RGetOwnership(n) \triangleq$
    $\land$ $rKeySharers[n] \neq$ "owner"
    $\land$ $\forall x \in rAliveNodes : rNodeEpochID[x] = rEpochID$   TODO may move this to $RNewOwner$
    $\land$ $\lor$ $ROverthrowOwner(n)$
      $\lor$ $RNewOwner(n)$
```

```
$RRead(n) \triangleq$   Execute a read
    $\land$ $rNodeEpochID[n] = rEpochID$
    $\land$ $rKeyState[n]$       = "valid"
    $\land$ $RNoChanges$

$RRcvInv(n) \triangleq$   Process a received invalidation
 $\exists m \in rMsgs$ :
        $\land$ $m.type$        = "INV"
        $\land$ $m.epochID$  = $rEpochID$
        $\land$ $m.sender$  $\neq n$
        $\land$ $m.sender$  $\in rAliveNodes$
         always acknowledge a received invalidation (irrelevant to the timestamp)
        $\land$ $RSend$([$type$          $\mapsto$ "ACK",
                    $epochID$   $\mapsto rEpochID$,
                    $sender$     $\mapsto n$,
                    $version$    $\mapsto m.version$])
        $\land$ $\lor$ $m.version$          $> rKeyVersion[n]$
              $\land$ $rKeyState[n]$   $\in$ {"valid", "invalid", "replay"}
              $\land$ $rKeyState'$         = [$rKeyState$ EXCEPT ![$n$] = "invalid"]
              $\land$ $rKeyVersion'$      = [$rKeyVersion$ EXCEPT ![$n$]  = $m.version$]
              $\land$ $rKeyLastWriter'$ = [$rKeyLastWriter$ EXCEPT ![$n$] = $m.sender$]
           $\lor$ $m.version$           $\leq rKeyVersion[n]$
              $\land$ UNCHANGED $\langle rKeyState, rKeyVersion, rKeyLastWriter \rangle$
        $\land$ UNCHANGED $\langle rAliveNodes, rKeySharers, rKeyRcvedACKs, rNodeEpochID, rEpochID \rangle$

$RRcvVal(n) \triangleq$   Process a received validation
    $\exists m \in rMsgs$ :
        $\land$ $rKeyState[n] \neq$ "valid"
        $\land$ $m.type$          = "VAL"
        $\land$ $m.epochID$     = $rEpochID$
        $\land$ $m.version$      = $rKeyVersion[n]$
        $\land$ $rKeyState'$      = [$rKeyState$ EXCEPT ![$n$] = "valid"]
    $\land$ UNCHANGED $\langle rMsgs, rKeyVersion, rKeyLastWriter, rKeySharers,$
                      $rAliveNodes, rKeyRcvedACKs, rNodeEpochID, rEpochID\rangle$

$RReaderActions(n) \triangleq$    Actions of a write follower
    $\lor RRead(n)$
    $\lor RRcvInv(n)$
    $\lor RRcvVal(n)$
```

```
$RWrite(n) \triangleq$
    $\land$ $rNodeEpochID[n] = rEpochID$
    $\land$ $rKeySharers[n] \in \{$"owner"$\}$
    $\land$ $rKeyState[n] \in \{$"valid"$\}$  May add invalid state here as well
    $\land$ $rKeyVersion[n] < R\_MAX\_VERSION$
    $\land$ $rKeyLastWriter' = [rKeyLastWriter$ EXCEPT $![n] = n]$
    $\land$ $rKeyRcvedACKs' = [rKeyRcvedACKs$ EXCEPT $![n] = \{\}]$
    $\land$ $rKeyState' = [rKeyState$ EXCEPT $![n] =$ "write"$]$
    $\land$ $rKeyVersion' = [rKeyVersion$ EXCEPT $![n] = rKeyVersion[n] + 1]$
    $\land$ $RSend([type \mapsto$ "INV",
               $epochID \mapsto rEpochID,$
               $sender \mapsto n,$
               $version \mapsto rKeyVersion[n] + 1])$
    $\land$ UNCHANGED $\langle rAliveNodes, rKeySharers, rNodeEpochID, rEpochID\rangle$

$RRcvAck(n) \triangleq$    Process a received acknowledment
    $\exists m \in rMsgs :$
        $\land m.type =$ "ACK"
        $\land m.epochID = rEpochID$
        $\land m.sender \neq n$
        $\land m.version = rKeyVersion[n]$
        $\land m.sender \notin rKeyRcvedACKs[n]$
        $\land rKeyState[n] \in \{$"write", "replay"$\}$
        $\land rKeyRcvedACKs' = [rKeyRcvedACKs$ EXCEPT $![n] =$
                                       $rKeyRcvedACKs[n] \cup \{m.sender\}]$
        $\land$ UNCHANGED $\langle rMsgs, rKeyState, rKeyVersion, rKeyLastWriter,$
                          $rAliveNodes, rKeySharers, rNodeEpochID, rEpochID\rangle$

$RSendVals(n) \triangleq$    Send validations once received acknowledments from all alive nodes
    $\land rKeyState[n] \in \{$"write", "replay"$\}$
    $\land RAllACKsRcved(n)$
    $\land rKeyState' = [rKeyState$ EXCEPT $![n] =$ "valid"$]$
    $\land RSend([type \mapsto$ "VAL",
               $epochID \mapsto rEpochID,$
               $version \mapsto rKeyVersion[n]])$
    $\land$ UNCHANGED $\langle rKeyRcvedACKs, rKeyVersion, rKeyLastWriter,$
```

$$\qquad rAliveNodes, rKeySharers, rNodeEpochID, rEpochID\rangle$$

$ROwnerActions(n) \triangleq$ Actions of a read/write coordinator

$$\begin{aligned}
&\lor RRead(n)\\
&\lor RWrite(n)\\
&\lor RRcvAck(n)\\
&\lor RSendVals(n)
\end{aligned}$$

$RWriteReplay(n) \triangleq$ Execute a write-replay

$$\begin{aligned}
&\land\ rKeyLastWriter' = [rKeyLastWriter\ \text{EXCEPT}\ ![n] = n]\\
&\land\ rKeyRcvedACKs' = [rKeyRcvedACKs\ \text{EXCEPT}\ ![n] = \{\}]\\
&\land\ rKeyState' = [rKeyState\ \text{EXCEPT}\ ![n] = \text{“replay”}]\\
&\land\ RSend([type \mapsto \text{“INV”},\\
&\qquad\qquad sender \mapsto n,\\
&\qquad\qquad epochID \mapsto rEpochID,\\
&\qquad\qquad version \mapsto rKeyVersion[n]])\\
&\land\ \text{UNCHANGED}\ \langle rKeyVersion, rKeySharers, rAliveNodes, rNodeEpochID, rEpochID\rangle
\end{aligned}$$

$RLocalWriteReplay(n) \triangleq$

$$\begin{aligned}
&\land\ \lor rKeySharers[n] = \text{“owner”}\\
&\quad\ \lor rKeyState[n] = \text{“replay”}\\
&\land\ RWriteReplay(n)
\end{aligned}$$

$RFailedNodeWriteReplay(n) \triangleq$

$$\begin{aligned}
&\land\ \neg RIsAlive(rKeyLastWriter[n])\\
&\land\ rKeyState[n] = \text{“invalid”}\\
&\land\ RWriteReplay(n)
\end{aligned}$$

$RUpdateLocalEpochID(n) \triangleq$

$$\begin{aligned}
&\land rKeyState[n] = \text{“valid”}\\
&\land rNodeEpochID' = [rNodeEpochID\ \text{EXCEPT}\ ![n] = rEpochID]\\
&\land \text{UNCHANGED}\ \langle rMsgs, rKeyState, rKeyVersion, rKeyRcvedACKs,\\
&\qquad\qquad rKeyLastWriter, rKeySharers, rAliveNodes, rEpochID\rangle
\end{aligned}$$

$RReplayActions(n) \triangleq$

$$\begin{aligned}
&\land rNodeEpochID[n] < rEpochID\\
&\land\ \lor RLocalWriteReplay(n)\\
&\quad\ \lor RFailedNodeWriteReplay(n)\\
&\quad\ \lor RUpdateLocalEpochID(n)
\end{aligned}$$

$RNext \triangleq$ Modeling protocol (Owner and Reader actions while emulating failures)

$$\begin{aligned}
&\exists\, n \in rAliveNodes :\\
&\qquad \lor RReaderActions(n)\\
&\qquad \lor ROwnerActions(n)
\end{aligned}$$

$\lor RReplayActions(n)$
$\lor RGetOwnership(n)$
$\lor RNodeFailure(n)$ emulate node failures

The complete definition of the algorithm

$Spec \triangleq RInit \land \Box[RNext]_{vars}$

$$Invariants \triangleq \begin{array}{l} \land (\Box RTypeOK) \\ \land (\Box RConsistentInvariant) \\ \land (\Box RSingleOnwerInvariant) \\ \land (\Box ROnwerOnlyWriterInvariant) \\ \land (\Box RMaxVersionDistanceInvariant) \\ \land (\Box ROnwerHighestVersionInvariant) \end{array}$$

THEOREM $Spec \Rightarrow Invariants$