MODULE *Galene*

Galene: is a linearizable protocol used in *ccKVS* of Scale-out *ccNUMA* [*Eurosys*′18]

```
EXTENDS     Integers,
            FiniteSets

CONSTANTS   G_NODES,
            G_MAX_VERSION

VARIABLES   msgs,
            nodeTS,
            nodeState,
            nodeRcvedAcks,
            nodeLastWriteTS
```

all *Galene*(+ environment) variables

$gvars \triangleq \langle msgs, nodeTS, nodeState, nodeRcvedAcks, nodeLastWriteTS \rangle$

A buffer maintaining all network messages. Messages are only appended to this variable (not removed once delivered) intentionally to check protocols tolerance in dublicates and reorderings

$send(m) \triangleq msgs' = msgs \cup \{m\}$

Check if all acknowledgments for a write have been received

$receivedAllAcks(n) \triangleq (G\_NODES \setminus \{n\}) \subseteq nodeRcvedAcks[n]$

```
equalTS(v1, tb1, v2, tb2) ≜     Timestamp equality
    ∧  v1 = v2
    ∧  tb1 = tb2

greaterTS(v1, tb1, v2, tb2) ≜     Timestamp comparison
    ∨ v1 > v2
    ∨ ∧  v1 = v2
      ∧ tb1 > tb2
```

```
GMessage ≜     Messages exchanged by the Protocol
    [type : {"INV", "ACK"}, sender     : G_NODES,
                            version    : 0 .. G_MAX_VERSION,
                            tieBreaker : G_NODES]
        ∪
      Note that we need not send Value w/ VALs, timestamp suffice to check consistency
    [type : {"VAL"},        version    : 0 .. G_MAX_VERSION,
                            tieBreaker : G_NODES]

GTypeOK ≜     The type correctness invariant
    ∧     msgs          ⊆ GMessage
```

$$
\begin{array}{ll}
\land \forall n \in G\_NODES : nodeRcvedAcks[n] \subseteq (G\_NODES \setminus \{n\}) & \\
\land\ nodeLastWriteTS \in [G\_NODES \rightarrow [version : 0 \,..\, G\_MAX\_VERSION, & \\
\qquad\qquad\qquad\qquad\qquad\qquad tieBreaker : G\_NODES]] & \\
\land\ nodeTS \in [G\_NODES \rightarrow [version : 0 \,..\, G\_MAX\_VERSION, & \\
\qquad\qquad\qquad\qquad\qquad\qquad tieBreaker : G\_NODES]] & \\
\land\ nodeState \in [G\_NODES \rightarrow \{\text{“valid”}, \text{“invalid”}, \text{“invalid\_write”}, \text{“write”}\}] &
\end{array}
$$

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

$$
\begin{array}{l}
GConsistent \triangleq \\
\quad \forall k, s \in G\_NODES \ : \ \lor\ nodeState[k] \neq \text{“valid”} \\
\qquad\qquad\qquad\qquad\qquad\quad \lor\ nodeState[s] \neq \text{“valid”} \\
\qquad\qquad\qquad\qquad\qquad\quad \lor\ nodeTS[k] = nodeTS[s]
\end{array}
$$

$GInit \triangleq$ The initial predicate

$$
\begin{array}{lcl}
\land\ msgs & = & \{\} \\
\land\ nodeRcvedAcks & = & [n \in G\_NODES \mapsto \{\}] \\
\land\ nodeState & = & [n \in G\_NODES \mapsto \text{“valid”}] \\
\land\ nodeTS & = & [n \in G\_NODES \mapsto [version \mapsto 0, \\
 & & \qquad tieBreaker \mapsto \\
 & & \qquad \textsc{choose}\ k \in G\_NODES : \\
 & & \qquad\ \forall m \in G\_NODES : k \leq m]] \\
\land\ nodeLastWriteTS & = & [n \in G\_NODES \mapsto [version \mapsto 0, \\
 & & \qquad tieBreaker \mapsto \\
 & & \qquad \textsc{choose}\ k \in G\_NODES : \\
 & & \qquad\ \forall m \in G\_NODES : k \leq m]]
\end{array}
$$

$$
\begin{array}{l}
g\_upd\_state(n, newVersion, newTieBreaker, newState, newAcks) \triangleq \\
\quad \land\ nodeRcvedAcks' = [nodeRcvedAcks\ \textsc{except}\ ![n] = newAcks] \\
\quad \land\ nodeState' = [nodeState\ \textsc{except}\ ![n] = newState] \\
\quad \land\ nodeTS' = [nodeTS\ \textsc{except}\ ![n].version = newVersion, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad ![n].tieBreaker = newTieBreaker] \\
\quad \land\ nodeLastWriteTS' = [nodeLastWriteTS\ \textsc{except}\ ![n].version = newVersion, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![n].tieBreaker = newTieBreaker]
\end{array}
$$

$$
\begin{array}{l}
g\_send\_inv\_or\_ack(n, newVersion, newTieBreaker, msgType) \triangleq \\
\quad \land\ send([type \mapsto msgType, \\
\qquad\qquad\quad sender \mapsto n, \\
\qquad\qquad\quad version \mapsto newVersion, \\
\qquad\qquad\quad tieBreaker \mapsto newTieBreaker])
\end{array}
$$

$$
\begin{array}{l}
g\_actions\_for\_upd(n, newVersion, newTieBreaker, newState, newAcks) \triangleq \\
\quad \land\ g\_upd\_state(n, newVersion, newTieBreaker, newState, newAcks) \\
\quad \land\ g\_send\_inv\_or\_ack(n, newVersion, newTieBreaker, \text{“INV”})
\end{array}
$$

```
$GRead(n) \triangleq$   Execute a read
    $\land\ nodeState[n] = \text{“valid”}$
    $\land\ \text{UNCHANGED}\ \langle msgs, nodeTS, nodeState, nodeRcvedAcks, nodeLastWriteTS \rangle$

$GWrite(n) \triangleq$   Execute a write
    $\land\ nodeState[n] = \text{“valid”}$
    $\land\ nodeTS[n].version < G\_MAX\_VERSION$   Only to configurably terminate the model checking
    $\land\ g\_actions\_for\_upd(n, nodeTS[n].version + 1, n, \text{“write”}, \{\})$

$GRcvAck(n) \triangleq$   Process received $Ack$
   $\exists\, m \in msgs :$
      $\land\ m.type \quad = \text{“ACK”}$
      $\land\ m.sender \ \neq n$
      $\land\ m.sender \ \notin nodeRcvedAcks[n]$
      $\land\ equalTS(m.version, m.tieBreaker,$
                    $nodeLastWriteTS[n].version,$
                    $nodeLastWriteTS[n].tieBreaker)$
      $\land\ nodeState[n] \in \{\text{“write”}, \text{“invalid\_write”}\}$
      $\land\ nodeRcvedAcks' = [nodeRcvedAcks\ \text{EXCEPT}\ ![n] =$
                                             $nodeRcvedAcks[n] \cup \{m.sender\}]$
      $\land\ \text{UNCHANGED}\ \langle msgs, nodeLastWriteTS, nodeTS, nodeState \rangle$

$GSendVals(n) \triangleq$   Send validations once acknowledments are received from all alive nodes
    $\land\ nodeState[n] \in \{\text{“write”}\}$
    $\land\ receivedAllAcks(n)$
    $\land\ nodeState' \qquad = [nodeState\ \text{EXCEPT}\ ![n] = \text{“valid”}]$
    $\land\ send([type \quad\ \mapsto \text{“VAL”},$
              $version \quad \mapsto nodeTS[n].version,$
              $tieBreaker \mapsto nodeTS[n].tieBreaker])$
    $\land\ \text{UNCHANGED}\ \langle nodeTS, nodeLastWriteTS, nodeRcvedAcks \rangle$

$GCoordinatorActions(n) \triangleq$   Coordinator actions for reads or writes
    $\lor\ GRead(n)$
    $\lor\ GWrite(n)$
    $\lor\ GRcvAck(n)$
    $\lor\ GSendVals(n)$
```

```
$GRcvInv(n) \triangleq$   Process received invalidation
   $\exists\, m \in msgs :$
      $\land\ m.type \quad = \text{“INV”}$
      $\land\ m.sender \ \neq n$
       always acknowledge a received invalidation (irrelevant to the timestamp)
```

```
        ∧ $send([type \mapsto$ "ACK",
                  $sender \mapsto n$,
                  $version \mapsto m.version$,
                  $tieBreaker \mapsto m.tieBreaker])$
        ∧ IF $greaterTS(m.version, m.tieBreaker,$
                        $nodeTS[n].version, nodeTS[n].tieBreaker)$
           THEN  ∧ $nodeTS' = [nodeTS$ EXCEPT $![n].version = m.version,$
                                              $![n].tieBreaker = m.tieBreaker]$
                 ∧ IF $nodeState[n] \in \{$"valid", "invalid"$\}$
                     THEN
                       $nodeState' = [nodeState$ EXCEPT $![n] =$ "invalid"$]$
                     ELSE
                       $nodeState' = [nodeState$ EXCEPT $![n] =$ "invalid_write"$]$
           ELSE
                 UNCHANGED $\langle nodeState, nodeTS \rangle$
        ∧ UNCHANGED $\langle nodeLastWriteTS, nodeRcvedAcks \rangle$

$GRcvVal(n) \triangleq$   Process received validation
    $\exists m \in msgs$ :
        ∧ $nodeState[n] \neq$ "valid"
        ∧ $m.type =$ "VAL"
        ∧ $equalTS(m.version, m.tieBreaker,$
                   $nodeTS[n].version,$
                   $nodeTS[n].tieBreaker)$
        ∧ $nodeState' = [nodeState$ EXCEPT $![n] =$ "valid"$]$
        ∧ UNCHANGED $\langle msgs, nodeTS, nodeLastWriteTS, nodeRcvedAcks \rangle$

$GFollowerActions(n) \triangleq$   Follower actions for writes
    ∨ $GRcvInv(n)$
    ∨ $GRcvVal(n)$
```

```
$GNext \triangleq$   Coordinator and Follower actions
    $\exists n \in G\_NODES$ :
            ∨ $GFollowerActions(n)$
            ∨ $GCoordinatorActions(n)$

$G\_Spec \triangleq GInit \land \Box[GNext]_{gvars}$

THEOREM $G\_Spec \Rightarrow (\Box GTypeOK) \land (\Box GConsistent)$
```