MODULE *Galene*

Galene: is a linearizable protocol used in *ccKVS* of Scale-out *ccNUMA* [*Eurosys'18*]
This spec actually includes two variants of *Galene* one that does not accepts
concurrent writes (*SWMR*) but allows for read-modify-writes and
the one actually used in the paper that permits concurrent writes (*MWMR*).
Setting the constant *enableMWMR* TRUE or FALSE verifies either variant accordingly.

EXTENDS *Integers*, *FiniteSets*

CONSTANTS *G_NODES*, *G_MAX_VERSION*, *enableMWMR*

VARIABLES *msgs*, *nodeTS*, *nodeState*, *nodeRcvedAcks*

ASSUME $enableMWMR \in \{\text{TRUE}, \text{FALSE}\}$

all *Galene*(+ environment) variables
$gvars \triangleq \langle msgs, nodeTS, nodeState, nodeRcvedAcks \rangle$

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

$equalTS(v1, tb1, v2, tb2) \triangleq$ Timestamp equality
$\quad \land \quad v1 = v2$
$\quad \land \quad tb1 = tb2$

$greaterTS(v1, tb1, v2, tb2) \triangleq$ Timestamp comparison
$\quad \lor v1 > v2$
$\quad \lor \land \quad v1 = v2$
$\qquad \land \quad tb1 > tb2$

$GMessage \triangleq$ Messages exchanged by *Galene*
$\quad [type : \{\text{“INV”}, \text{“ACK”}\}, sender : G\_NODES,$
$\qquad\qquad\qquad\qquad\qquad\quad version : 0 \,..\, G\_MAX\_VERSION,$
$\qquad\qquad\qquad\qquad\qquad\quad tieBreaker : G\_NODES]$

$\quad \cup$

We do not send the Value w/ *UPDs* (*TS* suffices to check consistency)

$[type : \{\text{“UPD”}\}, \quad version : 0 \,..\, G\_MAX\_VERSION,$
$\quad tieBreaker : G\_NODES]$

$GTypeOK \triangleq$ The type correctness invariant
$\quad \land\ msgs \subseteq GMessage$
$\quad \land\ \forall\, n \in G\_NODES : nodeRcvedAcks[n] \subseteq (G\_NODES \setminus \{n\})$
$\quad \land\ nodeTS \in [G\_NODES \to [version : 0 \,..\, G\_MAX\_VERSION,$
$\qquad tieBreaker : G\_NODES\ ]]$
$\quad \land\ nodeState \in [G\_NODES \to \{\text{“valid”}, \text{“invalid”}, \text{“write”}\}]$

The consistent invariant: all alive nodes in valid state should have the same TS (value)
$GConsistent \triangleq$
$\quad \forall\, k, s \in G\_NODES : \lor\ nodeState[k] \neq \text{“valid”}$
$\qquad \lor\ nodeState[s] \neq \text{“valid”}$
$\qquad \lor\ nodeTS[s] = nodeTS[k]$

$GSWMR \triangleq$ veryfying exactly one write is committed per version
$\quad \lor\ enableMWMR$
$\quad \lor\ \forall\, m, l \in msgs : \lor\ m.type \neq \text{“UPD”}$
$\qquad \lor\ l.type \neq \text{“UPD”}$
$\qquad \lor\ m.version \neq l.version$
$\qquad \lor\ m.tieBreaker = l.tieBreaker$

$GInit \triangleq$ The initial predicate
$\quad \land\ msgs = \{\}$
$\quad \land\ nodeRcvedAcks = [n \in G\_NODES \mapsto \{\}]$
$\quad \land\ nodeState = [n \in G\_NODES \mapsto \text{“valid”}]$
$\quad \land\ nodeTS = [n \in G\_NODES \mapsto [version \mapsto 0,$
$\qquad tieBreaker \mapsto$
$\qquad\quad \text{CHOOSE } k \in G\_NODES :$
$\qquad\quad \forall\, m \in G\_NODES : k \leq m]]$

$g\_actions\_for\_upd(n, newVersion, newTieBreaker, newState, newAcks) \triangleq$
$\quad \land\ nodeRcvedAcks' = [nodeRcvedAcks \text{ EXCEPT } ![n] = newAcks]$
$\quad \land\ nodeState' = [nodeState \text{ EXCEPT } ![n] = newState]$
$\quad \land\ nodeTS' = [nodeTS \text{ EXCEPT } ![n].version = newVersion,$
$\qquad ![n].tieBreaker = newTieBreaker]$
$\quad \land\ send([type \mapsto \text{“INV”},$
$\qquad sender \mapsto n,$
$\qquad version \mapsto newVersion,$
$\qquad tieBreaker \mapsto newTieBreaker])$

$GRead(n) \triangleq$ Execute a read
$\land\ nodeState[n] = \text{“valid”}$
$\land$ UNCHANGED $gvars$

$GWrite(n) \triangleq$ Execute a write
$\land\ nodeState[n] = \text{“valid”}$
$\land\ nodeTS[n].version < G\_MAX\_VERSION$ 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*
$\exists\, m \in msgs :$
$\land\ m.type = \text{“ACK”}$
$\land\ nodeState[n] = \text{“write”}$
$\land\ m.sender \neq n$
$\land\ m.sender \notin nodeRcvedAcks[n]$
$\land\ equalTS(m.version,\ m.tieBreaker,$
$nodeTS[n].version,$
$nodeTS[n].tieBreaker)$
$\land\ nodeRcvedAcks' = [nodeRcvedAcks$ EXCEPT $![n] =$
$nodeRcvedAcks[n] \cup \{m.sender\}]$
$\land$ UNCHANGED $\langle msgs,\ nodeTS,\ nodeState\rangle$

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

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

$ACK\_already\_send(n,\ m) \triangleq$ only to keep the state space small (we do not re-send *ACKs* previously sent)
$\exists\, k \in msgs :$
$\land\ k.type = \text{“ACK”}$
$\land\ k.sender = n$
$\land\ k.version = m.version$
$\land\ k.tieBreaker = m.tieBreaker$

$GRcvInvSWMR(n,\ m) \triangleq$ apply and respond to invalidation iff greater ts

$\land$ $greaterTS(m.version,\ m.tieBreaker,$
$nodeTS[n].version,$
$nodeTS[n].tieBreaker)$

$\land$ $send([type \mapsto$ "ACK",
$sender \mapsto n,$
$version \mapsto m.version,$
$tieBreaker \mapsto m.tieBreaker])$

$\land$ $nodeState' = [nodeState$ EXCEPT $![n] =$ "invalid"$]$

$\land$ $nodeTS' = [nodeTS$ EXCEPT $![n].version = m.version,$
$![n].tieBreaker = m.tieBreaker]$

$\land$ UNCHANGED $\langle nodeRcvedAcks \rangle$

$GRcvInvMWMR(n,\ m) \triangleq$ (if $MWMR$ is enabled) send an ACK even if not greater ts

$\land$ $enableMWMR =$ TRUE

$\land$ $\neg greaterTS(m.version,\ m.tieBreaker,$
$nodeTS[n].version,$
$nodeTS[n].tieBreaker)$

$\land$ $\neg ACK\_already\_send(n,\ m)$

$\land$ $send([type \mapsto$ "ACK",
$sender \mapsto n,$
$version \mapsto m.version,$
$tieBreaker \mapsto m.tieBreaker])$

$\land$ UNCHANGED $\langle nodeRcvedAcks,\ nodeTS,\ nodeState \rangle$

$GRcvInv(n,\ m) \triangleq$

$\land\ m.type = $ "INV"
$\land\ m.sender \neq n$
$\land\ \lor\ GRcvInvSWMR(n,\ m)$
$\lor\ GRcvInvMWMR(n,\ m)$

$GRcvUpd(n,\ m) \triangleq$ Process received validation iff same ts

$\land\ m.type = $ "UPD"
$\land\ nodeState[n] \neq$ "valid"
$\land\ equalTS(m.version,\ m.tieBreaker,$
$nodeTS[n].version,$
$nodeTS[n].tieBreaker)$
$\land\ nodeState' = [nodeState$ EXCEPT $![n] =$ "valid"$]$
$\land$ UNCHANGED $\langle msgs,\ nodeTS,\ nodeRcvedAcks \rangle$

$GFollowerActions(n) \triangleq$ Follower actions for writes

$\exists\, m \in msgs :$
$\lor\ GRcvInv(n,\ m)$
$\lor\ GRcvUpd(n,\ m)$

$$
\begin{array}{l}
GNext \triangleq \\
\quad \exists\, n \in G\_NODES : \\
\qquad \lor\ GFollowerActions(n) \\
\qquad \lor\ GCoordinatorActions(n)
\end{array}
$$

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

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