MODULE *ZeusOwnershipFaults*

EXTENDS *ZeusOwnership*

$sharing\_ok \triangleq \forall\, nn \in APP\_LIVE\_NODES : \lor \neg has\_data(nn)$
$\lor\ tState[nn] = \text{“valid”}$

$arb\_replay(n) \triangleq$
$\land\ upd\_rEID(n)$
$\land\ upd\_o\_meta\_driver(n,\ oTS[n].ver,\ oTS[n].tb)$
$\land\ o\_send\_inv(n,\ n,\ oTS[n],\ oVector[n],\ rTS[n],\ rID[n],\ rType[n])$

Requester replays msg after a failure

$OFRequester \triangleq$
$\exists\, n \in APP\_LIVE\_NODES :$
$\land\ oState[n] = \text{“request”}$
$\land\ rEID[n] < mEID$
$\land\ upd\_rEID(n)$
$\land\ o\_send\_req(rTS[n],\ rID[n],\ rType[n])$
$\land\ unchanged\_mtco$

$OFDriverRequester \triangleq$ waits for sharing-ok + computes next *oVec* transitions to *valid* + send vals with the proper changes in the *oVec*

$\exists\, n \in LB\_LIVE\_NODES :$
$\land\ oState[n] = \text{“drive”}$
$\land\ has\_rcved\_all\_ACKs(n)$
$\land\ \neg requester\_is\_alive(n)$
$\land\ o\_send\_val(oTS[n])$
$\land\ upd\_o\_meta(n,\ oTS[n].ver,\ oTS[n].tb,\ \text{“valid”},\ 0,\ post\_oVec(n,\ 0,\ oVector[n]),\ \{\})$
$\land\ unchanged\_mtrc$

$OFArbReplay \triangleq$ drivers resets acks and replays msg on arbiter failures
if the failed arbiter was an owner we need to wait for sharing-ok
for convinience arb-replays happen on any failure (*e.g.*, requester)

$\exists\, n \in mAliveNodes :$
$\land\ (oState[n] = \text{“drive”} \quad \lor\ oState[n] = \text{“invalid”})$
$\land\ (n \in LB\_LIVE\_NODES \lor oVector[n].owner = n)$
$\land\ rEID[n] < mEID$
$\land\ \lor\ oVector[n].owner \in mAliveNodes$
$\lor\ sharing\_ok$
$\land\ arb\_replay(n)$
$\land\ unchanged\_mtc$

$OLBArbiterACK \triangleq$ *ACK* an *INV* message which has the same as local *s_ts* but wasn't applied
$\exists\, n \in LB\_LIVE\_NODES : \exists\, m \in oMsgs :$

$\land \neg inv\_to\_be\_applied(n,\ m)$
$\land o\_rcv\_inv\_equal\_ts(m,\ n)$
$\land o\_send\_ack(n,\ m.oTS,\ 0)$
$\land unchanged\_mtrco$

INV response to an owner who did an arb-replay due to a lost val

$OFMOArbiterLostVALOldReplay \triangleq$
$\exists\, l \in LB\_LIVE\_NODES : \exists\, a \in APP\_LIVE\_NODES :$
$\land oState[a] =$ "drive"
$\land oVector[a].owner = a$
$\land is\_greaterTS(oTS[l],\ oTS[a])$
$\land o\_send\_inv(l,\ l,\ oTS[l],\ oVector[l],\ rTS[l],\ rID[l],\ rType[l])$
$\land unchanged\_mtrco$

message failures

$OFMOArbiterINVLostVAL \triangleq$ An *INV* is received (w/ higher ts) to a non-valid owner who lost a *VAL* for the message that demoted him

$\exists\, n \in APP\_LIVE\_NODES : \exists\, m \in oMsgs :$
$\land oVector[n].owner = n$
$\land o\_rcv\_inv\_greater\_ts(m,\ n)$
$\land m.oVector.owner \neq n$
$\land upd\_o\_meta\_apply\_val\_n\_reset\_o\_state(n)$
$\land o\_send\_ack(n,\ m.oTS,\ tVersion[n])$
$\land unchanged\_mtrc$

$OFMRequesterVALReplay \triangleq$ Requester receives a *RESP* (already applied) and re-sends a *VAL* to arbiters

$\exists\, n \in APP\_LIVE\_NODES : \exists\, m \in oMsgs :$
$\land o\_rcv\_resp(m,\ n)$
$\land m.rTS.tb = n$
$\land m.oTS \quad = oTS[n]$
$\land o\_send\_val(m.oTS)$
$\land unchanged\_mtrco$

$block\_owner\_failures\_if\_not\_in\_tx\_valid\_state(n) \triangleq$
$\lor has\_valid\_data(n)$
$\lor \neg is\_valid\_owner(n)$

Emulate a node failure if there more than 2 alive nodes in *LIVE_NODE_SET*

$nodeFailure(n,\ LIVE\_NODE\_SET) \triangleq$
$\land n \in LIVE\_NODE\_SET$
$\land block\_owner\_failures\_if\_not\_in\_valid\_state(n)$
$\land Cardinality(LIVE\_NODE\_SET) > 2$
Update Membership and epoch *id*

$\land\ mEID' = mEID + 1$
$\land\ mAliveNodes' = mAliveNodes \setminus \{n\}$
Remove failed node from *oVectors*
$\land\ oVector' = [l \in O\_NODES \mapsto [readers \mapsto oVector[l].readers \setminus \{n\},$
$owner \mapsto \text{IF } oVector[l].owner = n$
$\text{THEN } 0$
$\text{ELSE } oVector[l].owner\ ]]$
$\land\ unchanged\_Mtrc$
$\land\ \text{UNCHANGED } \langle oState,\ oDriver,\ oRcvACKs,\ oTS \rangle$

$FNext \triangleq$
$\lor\ OFRequester$
$\lor\ OFDriverRequester$
$\lor\ OFArbReplay$
$\lor\ OLBArbiterACK$
$\lor\ OFMOArbiterINVLostVAL$
$\lor\ OFMRequesterVALReplay$
$\lor\ OFMOArbiterLostVALOldReplay$

$OFNext \triangleq$
$\lor\ ONext$
$\lor\ FNext$
$\lor\ \exists\, n \in mAliveNodes :$
$\lor\ nodeFailure(n,\ LB\_LIVE\_NODES)$ emulate *LB* node failures
$\lor\ nodeFailure(n,\ APP\_LIVE\_NODES)$ emulate application node failures

The complete definition of the algorithm

$SFSpec \triangleq\ OInit \land \Box[OFNext]_{vars}$

THEOREM $SFSpec \Rightarrow Invariants$
