MODULE *MCBlueRep*

EXTENDS *bluerep*

VARIABLES *prevread*,
for mapping to regular model
*last_read_val*,
*last_committed_write*

$mcblue\_newvars \triangleq \langle prevread,\ last\_read\_val,\ last\_committed\_write \rangle$
$mcbluevars \triangleq mcvars \circ mcblue\_newvars$

---

$MCBlue\_TypeInvariant \triangleq$
$\land\ Blue\_TypeInvariant$
$\land\ Print(\text{“}MCtypeInvar\text{”},\ \text{TRUE})$

$\land\ prevread \in [BObject \to Val \cup \{NoVal\}]$
$\land\ last\_read\_val \in [BObject \to Val \cup \{NoVal\}]$
$\land\ last\_committed\_write \in [BObject \to [val : Val \cup \{NoVal\},\ version : Nat]]$

$MCBlue\_Init \triangleq$
$\land\ Blue\_Init$

$\land\ prevread = [o \in BObject \mapsto NoVal]$
$\land\ last\_read\_val = [c \in BObject \mapsto NoVal]$
$\land\ last\_committed\_write = [c \in BObject \mapsto [val \mapsto NoVal,\ version \mapsto 0]]$

---

$MCBlue\_Reply(r,\ type,\ reply) \triangleq$
$\lor\ \land\ type = \text{“wrFinished”}$
All finished writes in Blue are guaranteed to be committed, even if they are committed to less than threshold replicas.
$\land\ last\_committed\_write' = [last\_committed\_write\ \text{EXCEPT}\ ![reply.object] =$
$\text{IF}\ reply.w.version \geq @.version\ \text{THEN}\ reply.w\ \text{ELSE}\ @]$
$\land\ prevread' = [prevread\ \text{EXCEPT}\ ![reply.object] = NoVal]$ invalidate the previous read
$\land\ \text{UNCHANGED}\ last\_read\_val$

$\lor\ \land\ type = \text{“rd”}$
$\land\ prevread' = [prevread\ \text{EXCEPT}\ ![reply.object] = reply.val]$
$\land\ last\_read\_val' = [last\_read\_val\ \text{EXCEPT}\ ![reply.object] = reply.val]$
$\land\ \text{UNCHANGED}\ \langle last\_committed\_write \rangle$

$MCBlue\_NoReply \triangleq$
$\text{UNCHANGED}\ mcblue\_newvars$

---

$MCBlue\_ReplicaActions \triangleq$
$\lor\ ReplicaDeath \land \text{UNCHANGED}\ mcblue\_newvars$

$\lor$ *ReadVersion* $\land$ UNCHANGED *mcblue_newvars*
$\lor$ *ProcessMessage* $\land$ UNCHANGED *mcblue_newvars*
$\lor$ *PrimaryKillsFailedRep* $\land$ UNCHANGED *mcblue_newvars*
$\lor$ *FinishWrite* a replica finishes a write it has started

*MCBlue_Next* $\triangleq$
$\lor$ *Blue_MasterActions* $\land$ UNCHANGED *mcblue_newvars*
$\lor$ *TimeActions* $\land$ UNCHANGED *mcblue_newvars*
$\lor$ *MCBlue_ReplicaActions*
$\lor$ *Blue_ClientActions*

*MCBlue_Spec* $\triangleq$ *MCBlue_Init* $\land \Box[MCBlue\_Next]_{mcbluevars}$

---

Invariants

This invariant should only hold if all reads are to the primary or if a replica never falsely declares another replica dead.

*ReadNowLastCommitted* $\triangleq$
$\forall\, o \in BObject$ :
$\forall\, r \in Rep$ :
( $\land\ cache[r][o].prim = r$ $r$ thinks it's primary
$\land\ stat[r].phase =$ "alive" $r$ is alive
$\land\ stat[r].lock[o] =$ "rdy"
$\land\ master.health[r] =$ "alive" ASSUMPTION : I never read from a stale replica (this ensured by read leases)
) $r$ is ready to answer query
$\Rightarrow$
$data[r][o] = last\_committed\_write[o].val$

*ReadLastCommitted* $\triangleq$
$\forall\, o \in BObject$ :
$\lor\ prevread[o] = NoVal$ there's been no read since the last commit
$\lor\ prevread[o] = last\_committed\_write[o].val$ OR the read value is that of the last committed write

*MCBlue_AllInvariants* $\triangleq$
$\land$ *AllInvariants*
$\land$ *ReadNowLastCommitted*
$\land$ *ReadLastCommitted*

---

Refinement mapping from Blue $->$ *SimpleStore*

*NotFailed*$(rsp)$ $\triangleq$
$\forall\, r \in Rep : rsp[r] \neq$ "badver" the only way to fail is to have a badversion response in Blue

*map_pending_wrreq*$(o)$ $\triangleq$
LET *has_wrreq*$(r)$ $\triangleq$

```
            ∧ stat[r].lock[o] = "busy"
            ∧ stat[r].in_progress[o].val ≠ NoWrite
            ∧ stat[r].in_progress[o].version ≥ last_committed_write[o].version
            ∧ NotFailed(resps[r][o])  the write is not yet failed

        F[reps ∈ SUBSET (Rep)] ≜
          IF (reps = {}) ∨ (∀ r ∈ reps : ¬has_wrreq(r))
           THEN ⟨⟩
           ELSE LET r_minver ≜ CHOOSE r ∈ reps :
                                ∧ has_wrreq(r)
                                 pick the one w/ min version number
                                ∧ ∀ r1 ∈ reps : (has_wrreq(r1) ⇒ stat[r].in_progress[o].version ≤ stat[r1].in_progr
                IN      ⟨stat[r_minver].in_progress[o].val⟩ ∘ F[reps \ {r_minver}]
    IN   ( Print("wrreq = ", F[Rep])

map_pending_wrresp(o) ≜   map the overlapping writes
    LET
        has_wrresp(r) ≜
            ∧ stat[r].lock[o] = "busy"
            ∧ stat[r].in_progress[o].val ≠ NoWrite
            ∧ stat[r].in_progress[o].version < last_committed_write[o].version
            ∧ NotFailed(resps[r][o])  the write is not yet failed

        F[reps ∈ SUBSET (Rep), ret ∈ [Val → Nat]] ≜
          IF (reps = {}) ∨ (∀ r ∈ reps : ¬has_wrresp(r))
             THEN ret
              ELSE LET r ≜ CHOOSE r ∈ reps : has_wrresp(r)
                       IN     F[reps \ {r}, [ret EXCEPT ![stat[r].in_progress[o].val] = @ + 1]]
    IN   ( Print("wrresp = ", F[Rep, [v ∈ Val ↦ 0]])

map_failed_wr(o) ≜
    LET
        G[reps ∈ SUBSET (Rep), ret ∈ [Val → Nat]] ≜
          IF reps = {} ∨ ∀ r ∈ reps : NotFailed(resps[r][o])
           THEN ret
           ELSE LET r ≜ CHOOSE r ∈ reps : ¬NotFailed(resps[r][o])  r is failed
                    IN     G[reps \ {r}, [ret EXCEPT ![stat[r].in_progress[o].val] = @ + 1]]
    IN   ( Print("failed_wr = ", G[Rep, [v ∈ Val ↦ 0]])

map_store(o) ≜   map the store value
    last_committed_write[o].val

map_last_read_val(o) ≜
    last_read_val[o]

mapped_object ≜ CHOOSE o ∈ BObject : TRUE
SS ≜ INSTANCE simplestore_quickrd WITH
```

$pending\_wrreq \leftarrow map\_pending\_wrreq(mapped\_object)$,
$pending\_wrresp \leftarrow map\_pending\_wrresp(mapped\_object)$,
$failed\_wr \leftarrow map\_failed\_wr(mapped\_object)$,
$last\_read\_val \leftarrow map\_last\_read\_val(mapped\_object)$,
$store \leftarrow map\_store(mapped\_object)$,
$pending\_rd \leftarrow 0$ 0-stage reads, so I never buffer reads in the channel

$Blue\_Implements\_SS \triangleq SS!SS1\_Spec$

---

THEOREM $MCBlue\_Spec \Rightarrow MCBlue\_AllInvariants$

---