MODULE *SimpleStore*

Blue *SimpleStore*.

EXTENDS *Naturals*, *Sequences*, *Util*, *TLC*

CONSTANT *Val*,
NoVal,
MaxReq

VARIABLE *pending_rd*,
pending_wrreq, pending / overlapping requests (*seq*)
pending_wrresp, pending responses – these are still pending requests from user's standpoint (map)
failed_wr, unsuccessful writes (map)

store, last update value acknowledged to the client
last_read_val last value returned to the last read

$SS\_TypeInvariant \triangleq$
$\land\ pending\_rd \in Nat$
$\land\ pending\_wrreq \in Seq(Val)$
$\land\ pending\_wrresp \in [Val \to Nat]$
$\land\ failed\_wr \in [Val \to Nat]$

$\land\ store \in Val \cup \{NoVal\}$
$\land\ last\_read\_val \in Val \cup \{NoVal\}$

$SS\_Init \triangleq$
$\land\ pending\_rd = 0$
$\land\ pending\_wrreq = \langle\rangle$
$\land\ pending\_wrresp = [v \in Val \mapsto 0]$
$\land\ failed\_wr = [v \in Val \mapsto 0]$

$\land\ store = NoVal$
$\land\ last\_read\_val = NoVal$

Client operations:

$SS\_CliWrite(w) \triangleq$ write request

Normally, the client inserts the request into the *pending_wrreq*

$\lor\ \land\ pending\_wrreq' = pending\_wrreq \circ \langle w \rangle$ add one more update request for value w
$\land$ UNCHANGED $\langle store, last\_read\_val, pending\_wrresp, failed\_wr, pending\_rd \rangle$

Rarely, an out-of-date client pushes its update directly to the *pending_wrresp*,
$w/$ no chance of his update to ever succeed.
This can only happen in Blue, because a replica may not know it's no longer the primary at some point.

$\lor\ \land\ pending\_wrresp' = [pending\_wrresp$ EXCEPT $![w] = @ + 1]$
$\land$ UNCHANGED $\langle store, last\_read\_val, pending\_wrreq, failed\_wr, pending\_rd \rangle$
$\lor\ \land\ pending\_wrreq' = \langle w \rangle \circ pending\_wrreq$

$\land$ UNCHANGED $\langle store, last\_read\_val, pending\_wrresp, failed\_wr, pending\_rd\rangle$

$SS\_CliRead \triangleq$ read request
$\land\ pending\_rd' = pending\_rd + 1$
$\land$ UNCHANGED $\langle store, last\_read\_val, pending\_wrreq, pending\_wrresp, failed\_wr\rangle$

$SS\_Client \triangleq$
$\lor \exists\, v \in Val : SS\_CliWrite(v)$
$\lor SS\_CliRead$

Channel operations:

$FailPendingWrReq(idx) \triangleq$
$\land\ Len(pending\_wrreq) \geq idx$
drop if from $pending\_wrreq$
$\land\ pending\_wrreq' = [i \in 1\,..\,(Len(pending\_wrreq) - 1) \mapsto$ IF $i < idx$ THEN $pending\_wrreq[i]$ ELSE $pending$
$SubSeq(pending\_wrreq, 1, idx - 1) \circ SubSeq(idx + 1, Len(pending\_wrreq))$
put it onto $failed\_wr$
$\land\ failed\_wr' = [failed\_wr$ EXCEPT $![pending\_wrreq[idx]] = @ + 1]$
$\land$ UNCHANGED $\langle pending\_wrresp\rangle$

$FailPendingWrResp(w) \triangleq$
$\land\ pending\_wrresp[w] > 0$ there's something to fail
drop if from $pending\_wrresp$
$\land\ pending\_wrresp' = [pending\_wrresp$ EXCEPT $![w] = @ - 1]$
put it onto $failed\_wr$
$\land\ failed\_wr' = [failed\_wr$ EXCEPT $![w] = @ + 1]$
$\land$ UNCHANGED $\langle pending\_wrreq\rangle$

$MapToSeq(map) \triangleq$
LET $F[set \in$ SUBSET $(Val)] \triangleq$
IF $set = \{\}$ THEN $\langle\rangle$
ELSE LET $v \triangleq$ CHOOSE $v \in set :$ TRUE
IN $[i \in 1\,..\,map[v] \mapsto v] \circ F[set \setminus \{v\}]$
IN
$F[Val]$

$\_Drop(map, seq, drop\_idxs) \triangleq$
LET $F[set \in$ SUBSET $(1\,..\,Len(seq)), ret \in [Val \to Nat]] \triangleq$
IF $set = \{\}$ THEN ret
ELSE LET $i \triangleq$ CHOOSE $i \in set :$ TRUEIN
$F[set \setminus \{i\}, [ret$ EXCEPT $![seq[i]] = @ - 1]]$
IN $F[drop\_idxs, map]$

$DropFailedWr \triangleq$
LET $seq \triangleq MapToSeq(failed\_wr)$
IN
$\land \exists\, drop\_idxs \in$ SUBSET $(1\,..\,Len(seq)) :$
$failed\_wr' = \_Drop(failed\_wr, seq, drop\_idxs)$

$\land$ UNCHANGED $\langle pending\_wrreq, pending\_wrresp \rangle$

$SS\_DropWrite \triangleq$
$\lor \exists\, idx \in 1 \,..\, Len(pending\_wrreq) : FailPendingWrReq(idx)$
$\lor \exists\, v \in Val : FailPendingWrResp(v)$
$\lor DropFailedWr$

$SS\_DropReads \triangleq$
$\land \exists\, i \in 1 \,..\, pending\_rd : pending\_rd' = pending\_rd - i$

$SS\_DropFromChannels \triangleq$
$\lor SS\_DropWrite$ either drop from write queue
$\land$ UNCHANGED $\langle last\_read\_val, store, pending\_rd \rangle$
$\lor SS\_DropReads$ or drop from read queue
$\land$ UNCHANGED $\langle last\_read\_val, store, pending\_wrresp, pending\_wrreq, failed\_wr \rangle$

$\_SS\_SendResponse(v) \triangleq$ sends one response of a write with value v
$\land pending\_wrresp[v] > 0$ there is a pending response $w/$ this value
$\land pending\_wrresp' = [pending\_wrresp$ EXCEPT $![v] = @ - 1]$ send the response
$\land$ UNCHANGED $\langle pending\_wrreq, last\_read\_val, pending\_rd, failed\_wr, store \rangle$
$SS\_SendResponse \triangleq$
$\exists\, v \in Val : \_SS\_SendResponse(v)$

$SS\_ChannelActions \triangleq$
$\lor SS\_DropFromChannels$
$\lor SS\_SendResponse$

STORE operations:

$SS\_HdlRead \triangleq$ handle one read request
$\land pending\_rd > 0$
$\land last\_read\_val' = store$ get the most current value of the store
$\land pending\_rd' = pending\_rd - 1$

$Move(from, to, idx) \triangleq$ move all writes in *pending_wrreq* up to *idx* to *pending_wrresp*
LET $F[i \in Nat, to\_prime \in [Val \to Nat]] \triangleq$
IF $i = 0$ THEN to_prime
ELSE $F[i - 1, [to\_prime$ EXCEPT $![from[i]] = @ + 1]]$
IN
$F[idx, to]$

$\_CommitWrite(idx) \triangleq$ handle one write request at index *idx* atomically
$\land Len(pending\_wrreq) \geq idx$

$\land store' = pending\_wrreq[idx]$
$\land Print(\text{“}CommitWrite : wrreq\text{”}, pending\_wrreq) \,\#\, \langle 1 \rangle$
$\land Print(\text{“}CommitWrite : wrreq'\text{”}, pending\_wrreq') \,\#\, \langle 1 \rangle$

$\land\ Print(\text{“}CommitWrite : wrresp\text{”},\ pending\_wrresp) \ \#\ \langle 1\rangle$
$\land\ Print(\text{“}CommitWrite : wrresp'\text{”},\ pending\_wrresp') \ \#\ \langle 1\rangle$
$\land\ Print(\text{“}CommitWrite : store\text{”},\ store) \ \#\ 1$
$\land\ Print(\text{“}CommitWrite : store'\text{”},\ store') \ \#\ 1$
$\land\ Print(\text{“}Committing\ write\ idx\text{”},\ idx) \geq 0$
$\land\ pending\_wrreq' = SubSeq(pending\_wrreq,\ idx + 1,\ Len(pending\_wrreq))$
drop all writes before this write from *pending_wrreq*
$\land\ pending\_wrresp' = Move(pending\_wrreq,\ pending\_wrresp,\ idx - 1)$ move all writes before *idx* from *pending_wrr*
Send response to client
$\land\ Print(\text{“}CommitWrite : wrresp\text{”},\ pending\_wrresp) \ \#\ \langle 1\rangle$
$\land\ Print(\text{“}CommitWrite : wrresp'\text{”},\ pending\_wrresp') \ \#\ \langle 1\rangle$
$\land\ Print(\text{“}CommitWrite : wrreq\text{”},\ pending\_wrreq) \ \#\ \langle 1\rangle$
$\land\ Print(\text{“}CommitWrite : wrreq'\text{”},\ pending\_wrreq') \ \#\ \langle 1\rangle$
$\land$ Print (“Committed!”, TRUE)

$SS\_CommitWrite \triangleq$
$\exists\, idx \in 1\,..\,Len(pending\_wrreq) : \_CommitWrite(idx)$

$SS\_RunStore \triangleq$
$\lor\ \land\ SS\_HdlRead$
$\land$ UNCHANGED $\langle pending\_wrreq,\ pending\_wrresp,\ failed\_wr,\ store\rangle$
$\lor\ \land\ SS\_CommitWrite$
$\land$ UNCHANGED $\langle pending\_rd,\ last\_read\_val,\ failed\_wr\rangle$

Full specification

$SS\_Next \triangleq$ Step
$\lor\ SS\_Client$ a client submits a request (query or update)
$\lor\ SS\_RunStore$ the store deals $w/$ the updates
$\lor\ SS\_ChannelActions$ the incoming channel for the store drops some requests

$ssvars \triangleq \langle store,\ pending\_wrreq,\ pending\_wrresp,\ failed\_wr,\ pending\_rd,\ last\_read\_val\rangle$
$SS\_Spec \triangleq SS\_Init \land \Box[SS\_Next]_{ssvars}$

$MaxRequests \triangleq$
$\land\ pending\_rd \leq MaxReq$
$\land\ \forall\, v \in Val : pending\_wrresp[v] \leq MaxReq \land failed\_wr[v] \leq MaxReq$
$\land\ Len(pending\_wrreq) \leq MaxReq$

Invariants

$SS\_AllInvariants \triangleq$
$\land\ SS\_TypeInvariant$

Theorem

THEOREM $SS\_Spec \Rightarrow \Box SS\_AllInvariants$