```
──────────────────────── MODULE voldemort ────────────────────────
EXTENDS Integers, Sequences, FiniteSets, TLC
CONSTANTS N, C, STOP, ReadQ, WriteQ, FAILNUM
ASSUME N = 5 ∧ C = 1 ∧ STOP < 10 ∧ 1 ≤ ReadQ ∧ ReadQ ≤ 3 ∧ 1 ≤ WriteQ ∧ WriteQ ≤ 3 ∧ 0 ≤ FAILNU
Nodes ≜ 1 . . N
Clients ≜ N + 1 . . N + C
--algorithm voldemort{
    variable FailNum = FAILNUM,
                 HVAL = 0, CVAL = 0, CVER = 0,
                 up = [n ∈ Nodes ↦ TRUE],
                 db = [n ∈ Nodes ↦ {[ver ↦ 0, val ↦ 0]}] ;
    define
    {
        UpNodes ≜ {i ∈ 1 . . N : up[i] = TRUE}
        ReturnReadQ ≜ CHOOSE i ∈ SUBSET (UpNodes) : Cardinality(i) = ReadQ
        ReturnWriteQ ≜ CHOOSE i ∈ SUBSET (UpNodes) : Cardinality(i) = WriteQ
     }

    procedure maxVal( tempQ )
    variable temp = 0, x = 0 ;
    {
         L1: while ( tempQ ≠ {} ) {
             x := CHOOSE k ∈ tempQ : TRUE ;
             tempQ := tempQ \ {x} ;
             if ( x > temp )
                temp := x ;
          } ;
         HVAL := temp ;
         return ;
     }

    fair process ( c ∈ Clients )
    variable cntr = 0, hver = 0, q = 0, Q = {}, nodeVersions = {}, writeQ = {}, data = 0, t = 0, i = 0, ver
    {
        CL: while ( cntr ≤ STOP )
        {
            cntr := cntr + 1     ;
            Q := ReturnReadQ ;

            L2: while ( Q ≠ {} ) {
              q  := CHOOSE k ∈ Q : TRUE ;
              Q := Q \ {q} ;
              ver := db[q]   ;

               L3: while ( ver ≠ {} ) {
                     r := CHOOSE k ∈ ver : TRUE ;
```

```
                    $ver := ver \setminus \{r\}$ ;
                    $nodeVersions := nodeVersions \cup \{r.ver\}$ ;
                }
            } ;

            get the highest version number from $RQ$
            call $maxVal(nodeVersions)$ ;
            $X1$: $hver := HVAL + 1$ ;

            write $val = cntr$ to $writeQuorum$ with higher version number
            $writeQ := ReturnWriteQ$ ;

            $L4$: while ( $writeQ \neq \{\}$ ) {
              $v :=$ CHOOSE $m \in writeQ$ : TRUE ;
              $writeQ := writeQ \setminus \{v\}$ ;
              $data\ \ \ := [ver \mapsto hver,\ val \mapsto cntr]$ ;
              $CVAL := cntr$ ;
              $CVER := hver$ ;
              $db[v] := db[v] \cup \{data\}$ ;
            }
        }
    }


   fair process ( $n \in Nodes$ )
   variable $x = 0$ ;
   {
       $L5$: while ( TRUE )
       {
           if ( $FailNum > 0 \land up[self] =$ TRUE )   Storage node can fail
           {
               $up[self] :=$ FALSE ;
               $FailNum := FailNum - 1$ ;
            }
           else if ( $up[self] =$ FALSE )     Or recover
           {
               $up[self] :=$ TRUE ;
               $FailNum := FailNum + 1$ ;
            }
    }
}
}
```

BEGIN TRANSLATION

Process variable x of process n at line 74 col 14 changed to $x\_$

CONSTANT $defaultInitValue$

VARIABLES $FailNum$, $HVAL$, $CVAL$, $CVER$, up, db, pc, $stack$

define statement

$UpNodes \triangleq \{i \in 1 \,..\, N : up[i] = \text{TRUE}\}$
$ReturnReadQ \triangleq \text{ CHOOSE } i \in \text{SUBSET } (UpNodes) : Cardinality(i) = ReadQ$
$ReturnWriteQ \triangleq \text{ CHOOSE } i \in \text{SUBSET } (UpNodes) : Cardinality(i) = WriteQ$

$\text{VARIABLES } tempQ, temp, x, cntr, hver, q, Q, nodeVersions, writeQ, data, t, i,$
$\quad ver, r, v, x\_$

$vars \triangleq \langle FailNum, HVAL, CVAL, CVER, up, db, pc, stack, tempQ, temp, x, cntr,$
$\quad hver, q, Q, nodeVersions, writeQ, data, t, i, ver, r, v, x\_\rangle$

$ProcSet \triangleq (Clients) \cup (Nodes)$

$Init \triangleq$ Global variables

$\land FailNum = FAILNUM$
$\land HVAL = 0$
$\land CVAL = 0$
$\land CVER = 0$
$\land up = [n \in Nodes \mapsto \text{TRUE}]$
$\land db = [n \in Nodes \mapsto \{[ver \mapsto 0, val \mapsto 0]\}]$

Procedure *maxVal*

$\land tempQ = [self \in ProcSet \mapsto defaultInitValue]$
$\land temp = [self \in ProcSet \mapsto 0]$
$\land x = [self \in ProcSet \mapsto 0]$

Process *c*

$\land cntr = [self \in Clients \mapsto 0]$
$\land hver = [self \in Clients \mapsto 0]$
$\land q = [self \in Clients \mapsto 0]$
$\land Q = [self \in Clients \mapsto \{\}]$
$\land nodeVersions = [self \in Clients \mapsto \{\}]$
$\land writeQ = [self \in Clients \mapsto \{\}]$
$\land data = [self \in Clients \mapsto 0]$
$\land t = [self \in Clients \mapsto 0]$
$\land i = [self \in Clients \mapsto 0]$
$\land ver = [self \in Clients \mapsto \{\}]$
$\land r = [self \in Clients \mapsto defaultInitValue]$
$\land v = [self \in Clients \mapsto 0]$

Process *n*

$\land x\_ = [self \in Nodes \mapsto 0]$
$\land stack = [self \in ProcSet \mapsto \langle\rangle]$
$\land pc = [self \in ProcSet \mapsto \text{CASE } self \in Clients \rightarrow \text{“CL”}$
$\qquad \Box \quad self \in Nodes \rightarrow \text{“L5”}]$

$L1(self) \triangleq \land pc[self] = \text{“L1”}$
$\quad \land \text{IF } tempQ[self] \neq \{\}$
$\qquad \text{THEN } \land x' = [x \text{ EXCEPT } ![self] = \text{CHOOSE } k \in tempQ[self] : \text{TRUE}]$

```
                    ∧ tempQ' = [tempQ EXCEPT ![self] = tempQ[self] \ {x'[self]}]
                    ∧ IF x'[self] > temp[self]
                          THEN ∧ temp' = [temp EXCEPT ![self] = x'[self]]
                          ELSE ∧ TRUE
                               ∧ temp' = temp
                    ∧ pc' = [pc EXCEPT ![self] = "L1"]
                    ∧ UNCHANGED ⟨HVAL, stack⟩
               ELSE ∧ HVAL' = temp[self]
                    ∧ pc' = [pc EXCEPT ![self] = Head(stack[self]).pc]
                    ∧ temp' = [temp EXCEPT ![self] = Head(stack[self]).temp]
                    ∧ x' = [x EXCEPT ![self] = Head(stack[self]).x]
                    ∧ tempQ' = [tempQ EXCEPT ![self] = Head(stack[self]).tempQ]
                    ∧ stack'  = [stack EXCEPT ![self]  = Tail(stack[self])]
          ∧ UNCHANGED ⟨FailNum, CVAL, CVER, up, db, cntr, hver, q, Q,
                       nodeVersions, writeQ, data, t, i, ver, r, v, x_⟩

maxVal(self) ≜ L1(self)

CL(self) ≜ ∧ pc[self] = "CL"
           ∧ IF cntr[self] ≤ STOP
                 THEN ∧ cntr' = [cntr EXCEPT ![self] = cntr[self] + 1]
                      ∧ Q' = [Q EXCEPT ![self] = ReturnReadQ]
                      ∧ pc' = [pc EXCEPT ![self] = "L2"]
                 ELSE ∧ pc' = [pc EXCEPT ![self] = "Done"]
                      ∧ UNCHANGED ⟨cntr, Q⟩
           ∧ UNCHANGED ⟨FailNum, HVAL, CVAL, CVER, up, db, stack, tempQ,
                        temp, x, hver, q, nodeVersions, writeQ, data, t, i,
                        ver, r, v, x_⟩

L2(self) ≜ ∧ pc[self] = "L2"
           ∧ IF Q[self] ≠ {}
                 THEN ∧ q'  = [q EXCEPT ![self]  = CHOOSE k ∈ Q[self] : TRUE]
                      ∧ Q' = [Q EXCEPT ![self] = Q[self] \ {q'[self]}]
                      ∧ ver' = [ver EXCEPT ![self] = db[q'[self]]]
                      ∧ pc' = [pc EXCEPT ![self] = "L3"]
                      ∧ UNCHANGED ⟨stack, tempQ, temp, x⟩
                 ELSE ∧ ∧ stack'  = [stack EXCEPT ![self]  = ⟨[procedure ↦  "maxVal",
                                                              pc        ↦ "X1",
                                                              temp      ↦  temp[self],
                                                              x         ↦  x[self],
                                                              tempQ     ↦  tempQ[self]]⟩
                                                             ∘ stack[self]]
                        ∧ tempQ' = [tempQ EXCEPT ![self] = nodeVersions[self]]
                      ∧ temp' = [temp EXCEPT ![self] = 0]
                      ∧ x' = [x EXCEPT ![self] = 0]
                      ∧ pc' = [pc EXCEPT ![self] = "L1"]
```

```
                $\land$ UNCHANGED $\langle q, Q, ver \rangle$
        $\land$ UNCHANGED $\langle FailNum, HVAL, CVAL, CVER, up, db, cntr, hver,$
                          $nodeVersions, writeQ, data, t, i, r, v, x\_\rangle$

$L3(self) \triangleq$ $\land\ pc[self] = $ "L3"
        $\land$ IF $ver[self] \neq \{\}$
            THEN $\land\ r' = [r$ EXCEPT $![self] =$ CHOOSE $k \in ver[self]$ : TRUE$]$
                 $\land\ ver' = [ver$ EXCEPT $![self] = ver[self] \setminus \{r'[self]\}]$
                 $\land\ nodeVersions' = [nodeVersions$ EXCEPT $![self] = nodeVersions[self] \cup \{r'[self].ve$
                 $\land\ pc' = [pc$ EXCEPT $![self] = $ "L3"$]$
            ELSE $\land\ pc' = [pc$ EXCEPT $![self] = $ "L2"$]$
                 $\land$ UNCHANGED $\langle nodeVersions, ver, r\rangle$
        $\land$ UNCHANGED $\langle FailNum, HVAL, CVAL, CVER, up, db, stack, tempQ,$
                          $temp, x, cntr, hver, q, Q, writeQ, data, t, i, v,$
                          $x\_\rangle$

$X1(self) \triangleq$ $\land\ pc[self] = $ "X1"
        $\land\ hver' = [hver$ EXCEPT $![self] = HVAL + 1]$
        $\land\ writeQ' = [writeQ$ EXCEPT $![self] = ReturnWriteQ]$
        $\land\ pc' = [pc$ EXCEPT $![self] = $ "L4"$]$
        $\land$ UNCHANGED $\langle FailNum, HVAL, CVAL, CVER, up, db, stack, tempQ,$
                          $temp, x, cntr, q, Q, nodeVersions, data, t, i, ver,$
                          $r, v, x\_\rangle$

$L4(self) \triangleq$ $\land\ pc[self] = $ "L4"
        $\land$ IF $writeQ[self] \neq \{\}$
            THEN $\land\ v' = [v$ EXCEPT $![self] =$ CHOOSE $m \in writeQ[self]$ : TRUE$]$
                 $\land\ writeQ' = [writeQ$ EXCEPT $![self] = writeQ[self] \setminus \{v'[self]\}]$
                 $\land\ data' \quad = [data$ EXCEPT $![self] = [ver \mapsto hver[self], val \mapsto cntr[self]]]$
                 $\land\ CVAL' = cntr[self]$
                 $\land\ CVER' = hver[self]$
                 $\land\ db' = [db$ EXCEPT $![v'[self]] = db[v'[self]] \cup \{data'[self]\}]$
                 $\land\ pc' = [pc$ EXCEPT $![self] = $ "L4"$]$
            ELSE $\land\ pc' = [pc$ EXCEPT $![self] = $ "CL"$]$
                 $\land$ UNCHANGED $\langle CVAL, CVER, db, writeQ, data, v\rangle$
        $\land$ UNCHANGED $\langle FailNum, HVAL, up, stack, tempQ, temp, x, cntr,$
                          $hver, q, Q, nodeVersions, t, i, ver, r, x\_\rangle$

$c(self) \triangleq CL(self) \lor L2(self) \lor L3(self) \lor X1(self) \lor L4(self)$

$L5(self) \triangleq$ $\land\ pc[self] = $ "L5"
        $\land$ IF $FailNum > 0 \land up[self] =$ TRUE
            THEN $\land\ up' = [up$ EXCEPT $![self] =$ FALSE$]$
                 $\land\ FailNum' = FailNum - 1$
            ELSE $\land$ IF $up[self] =$ FALSE
                     THEN $\land\ up' = [up$ EXCEPT $![self] =$ TRUE$]$
```

$$
\begin{array}{lll}
 & & \land\ \mathit{FailNum}' = \mathit{FailNum} + 1 \\
 & \text{ELSE} & \land\ \text{TRUE} \\
 & & \land\ \text{UNCHANGED}\ \langle \mathit{FailNum},\ \mathit{up} \rangle \\
\end{array}
$$

$$
\begin{array}{l}
\land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\mathit{self}] = \text{“L5”}] \\
\land\ \text{UNCHANGED}\ \langle \mathit{HVAL},\ \mathit{CVAL},\ \mathit{CVER},\ \mathit{db},\ \mathit{stack},\ \mathit{tempQ},\ \mathit{temp},\ x,\ \mathit{cntr}, \\
\qquad\qquad \mathit{hver},\ q,\ Q,\ \mathit{nodeVersions},\ \mathit{writeQ},\ \mathit{data},\ t,\ i,\ \mathit{ver}, \\
\qquad\qquad r,\ v,\ x\_ \rangle
\end{array}
$$

$$n(\mathit{self}) \triangleq L5(\mathit{self})$$

$$
\begin{array}{l}
\mathit{Next} \triangleq (\exists\, \mathit{self} \in \mathit{ProcSet} : \mathit{maxVal}(\mathit{self})) \\
\qquad\quad \lor (\exists\, \mathit{self} \in \mathit{Clients} : c(\mathit{self})) \\
\qquad\quad \lor (\exists\, \mathit{self} \in \mathit{Nodes} : n(\mathit{self}))
\end{array}
$$

$$
\begin{array}{l}
\mathit{Spec} \triangleq\ \land\ \mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}} \\
\qquad\quad \land\ \forall\, \mathit{self} \in \mathit{Clients} : \mathrm{WF}_{\mathit{vars}}(c(\mathit{self})) \land \mathrm{WF}_{\mathit{vars}}(\mathit{maxVal}(\mathit{self})) \\
\qquad\quad \land\ \forall\, \mathit{self} \in \mathit{Nodes} : \mathrm{WF}_{\mathit{vars}}(n(\mathit{self}))
\end{array}
$$

END TRANSLATION

$$
\begin{array}{l}
\mathit{Termination} \triangleq \Diamond(\mathit{CVER} = \mathit{STOP}) \\
\mathit{invariant} \triangleq \mathit{CVER} = \mathit{CVAL}
\end{array}
$$

Members − *Anand Sankar Bhagavandas − UB id* − 50208048
Rohit *Joseph Sebastian − UB id* − 50204806

The code given above is an implementation of the *Voldermort* single copy consistency.

The invariant we have specified in the above code is that the value of the version number and the value of the data written in a round is the same. In the ideal case with perfect copy consistency, the value of the version number and data should be the same.

Given below are the results we got for some of the values of *ReadQ*, *WriteQ* and *FAILNUM* that we tested our system on

Case 1: When $\mathit{ReadQ} = 1,\ \mathit{WriteQ} = 1$ and $\mathit{FAILNUM} = 0$

This is the ideal case where no node fails and the system runs without any hiccups. Our system runs successfully for this case satisfiying the invariant. The same result can be expected for higher values of *ReadQ* and *WriteQ* with *FailNum* remaining as 0.

Case 2: When $\mathit{ReadQ} = 1,\ \mathit{WriteQ} = 1$ and $\mathit{FAILNUM} = 1$

The system fails in this case as the invariant property is violated ie the value of the version number and the value of the data entered is not the same.

Case 3: When $\mathit{ReadQ} = 2,\ \mathit{WriteQ} = 1$ and $\mathit{FAILNUM} = 1$

The system fails for this case too as the invariant property is violated.

Case 4: When $\mathit{ReadQ} = 1,\ \mathit{WriteQ} = 2$ and $\mathit{FAILNUM} = 1$

The system fails for this case too as the invariant property is violated.

Case 5: When $ReadQ = 2$, $WriteQ = 2$ and $FAILNUM = 1$

The system runs to completion in this configuration of $ReadQ$, $WriteQ$ and $FAILNUM$.

Case 6: When $ReadQ = 3$, $WriteQ = 2$ and $FAILNUM = 2$

The system fails for this case as the invariant property is violated.

Case 7: When $ReadQ = 2$, $WriteQ = 3$ and $FAILNUM = 2$

The system fails for this case as the invariant property is violated.

Case 8: When $ReadQ = 3$, $WriteQ = 3$ and $FAILNUM = 2$

The system runs to completion in this configuration of $ReadQ$, $WriteQ$ and $FAILNUM$.

From the analysis we have done above we have come to the conclusion that the value of $ReadQ$ and $WriteQ$ should be one greater than $FAILNUM$. This will ensure that at a time the most current value written to the database will always be present in a node that has not failed. This will ensure single copy consistency.

Note - The initial value of $defaultInitValue$ should be given as 0 or as the model value