```
──────────────────── MODULE syncCon1 ────────────────────
EXTENDS Integers, Sequences, FiniteSets, TLC
CONSTANT N, FAILNUM
ASSUME N ≤ 5  ∧ 0 ≤ FAILNUM ∧ FAILNUM ≤ 2
Nodes ≜ 1 . . N
--algorithm syncCon1
{
     variable FailNum = FAILNUM,
              up = [n ∈ Nodes ↦ TRUE],
              pt  = [n ∈ Nodes ↦ 0],
              t  = [n ∈ Nodes ↦ FALSE],
              d = [n ∈ Nodes ↦  − 1],
              mb = [n ∈ Nodes ↦ {}] ;

    define {
    SetMin(S) ≜ CHOOSE i ∈ S : ∀ j ∈ S : i ≤ j
    AllUpNodes ≜ {n ∈ Nodes : up[n] = TRUE}                A set of nodes that have not failed
     }

    macro MaybeFail(  ) {
        if ( FailNum > 0 ∧ up[self] )
            { either
                { up[self] := FALSE ; FailNum := FailNum − 1 ;  }
              or skip ;  } ;
     }

    fair process ( n ∈ Nodes )
    variable v = 0, pv = 0, Q = {} ;
    {
P:  if ( up[self] ) {
       v  := self ;
       Q := Nodes ;
PS: while ( up[self] ∧ Q ≠ {} ) {
        with ( p ∈ Q ) {
              mb[p] := mb[p] ∪ {v} ;
              Q := Q \ {p} ;
              MaybeFail()  ;
         } ;
     } ;  end_while

    if ( up[self] ) pt[self] := pt[self] + 1 ;

PR: await (up[self] = TRUE ∧ ∀ k ∈ AllUpNodes : pt[self] = pt[k]) ;
        d[self] := SetMin(mb[self]) ;
        t[self]  := TRUE ;

     }  end_if
```

```
        }    end_process

        }
 }
 BEGIN TRANSLATION
VARIABLES FailNum, up, pt, t, d, mb, pc

 define statement
SetMin(S) ≜ CHOOSE i ∈ S : ∀ j ∈ S : i ≤ j
AllUpNodes ≜ {n ∈ Nodes : up[n] = TRUE}

VARIABLES v, pv, Q

vars ≜ ⟨FailNum, up, pt, t, d, mb, pc, v, pv, Q⟩

ProcSet ≜ (Nodes)

Init ≜  Global variables
         ∧ FailNum = FAILNUM
         ∧ up = [n  ∈ Nodes ↦ TRUE]
         ∧ pt  = [n  ∈ Nodes ↦ 0]
         ∧ t  = [n ∈ Nodes ↦ FALSE]
         ∧ d = [n ∈ Nodes ↦  − 1]
         ∧ mb = [n ∈ Nodes ↦ {}]
         Process n
         ∧ v = [self ∈ Nodes ↦ 0]
         ∧ pv = [self ∈ Nodes ↦ 0]
         ∧ Q = [self ∈ Nodes ↦ {}]
         ∧ pc = [self ∈ ProcSet ↦ "P"]

P(self) ≜ ∧ pc[self] = "P"
           ∧ IF up[self]
                 THEN ∧ v′  = [v EXCEPT ![self]  = self]
                      ∧ Q′ = [Q EXCEPT ![self] = Nodes]
                      ∧ pc′ = [pc EXCEPT ![self] = "PS"]
                 ELSE ∧ pc′ = [pc EXCEPT ![self] = "Done"]
                      ∧ UNCHANGED ⟨v, Q⟩
           ∧ UNCHANGED ⟨FailNum, up, pt, t, d, mb, pv⟩

PS(self) ≜ ∧ pc[self] = "PS"
            ∧ IF up[self] ∧ Q[self] ≠ {}
                  THEN ∧ ∃ p ∈ Q[self] :
                              ∧ mb′ = [mb EXCEPT ![p] = mb[p] ∪ {v[self]}]
                              ∧ Q′ = [Q EXCEPT ![self] = Q[self] \ {p}]
                              ∧ IF FailNum > 0 ∧ up[self]
                                    THEN ∧ ∨ ∧ up′ = [up EXCEPT ![self] = FALSE]
                                             ∧ FailNum′ = FailNum − 1
```

```
                                        ∨  ∧ TRUE
                                           ∧ UNCHANGED ⟨FailNum, up⟩
                                  ELSE  ∧ TRUE
                                        ∧ UNCHANGED ⟨FailNum, up⟩
                        ∧ pc' = [pc EXCEPT ![self] = "PS"]
                        ∧ pt' = pt
               ELSE  ∧ IF up[self]
                          THEN  ∧ pt' = [pt EXCEPT ![self] = pt[self] + 1]
                          ELSE  ∧ TRUE
                                ∧ pt' = pt
                     ∧ pc' = [pc EXCEPT ![self] = "PR"]
                     ∧ UNCHANGED ⟨FailNum, up, mb, Q⟩
            ∧ UNCHANGED ⟨t, d, v, pv⟩

PR(self) ≜  ∧ pc[self] = "PR"
            ∧ (up[self] = TRUE ∧ ∀ k ∈ AllUpNodes : pt[self] = pt[k])
            ∧ d' = [d EXCEPT ![self] = SetMin(mb[self])]
            ∧ t' = [t EXCEPT ![self] = TRUE]
            ∧ pc' = [pc EXCEPT ![self] = "Done"]
            ∧ UNCHANGED ⟨FailNum, up, pt, mb, v, pv, Q⟩

n(self) ≜ P(self) ∨ PS(self) ∨ PR(self)

Next ≜ (∃ self ∈ Nodes : n(self))
          ∨  Disjunct to prevent deadlock on termination
            ((∀ self ∈ ProcSet : pc[self] = "Done") ∧ UNCHANGED vars)

Spec ≜  ∧ Init ∧ □[Next]_vars
        ∧ ∀ self ∈ Nodes : WF_vars(n(self))

Termination ≜ ◇(∀ self ∈ ProcSet : pc[self] = "Done")

 END TRANSLATION

Inv ≜ ∀ i, j ∈ Nodes : (t[i] ∧ t[j]) ⇒ (d[i] = d[j])
```

Violation of Agreement property: The decision value, *d*□, is set to the minimum of *mb*[*self*] value. Now, when there is no crash node, all nodes send their value to *mb*□ and minimum is selected. But, when one or more nodes fail, they are not able to send their value (which could be minimum of all values) to the mailbox of other nodes. Hence, when the other nodes terminate, they may not able to correctly determine the minimum value. *i.e min*(*node i*) may not be equal to *min*(*node j*)

This is submission for following students:

Name: *Piyush Saravagi*	Name: *Abhishek Krishna*
UB Name: piyushsu	*UB* Name: *krishna*7
UB ID: 50246596	*UB ID*: 50246436