MODULE *syncCon*1

EXTENDS *Integers*, *Sequences*, *FiniteSets*, *TLC*
CONSTANTS N, *FAILNUM*
ASSUME $N \leq 5 \land 0 \leq FAILNUM \land FAILNUM \leq 4$
$Nodes \triangleq 1 \,..\, N$

```
--algorithm syncCon1
{ variables 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
     }

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

    fair process ( n ∈ Nodes )
    variable v = 0, Q = {} ;
    {
P: if ( up[self] ) {
    v := self ;
    Q := Nodes ;
PS: while ( up[self] ∧ Q ≠ {} ) {
        with ( p ∈ Q ) {
          MaybeFail() ;   In process n, each time we add v to mb[p], this process might be fail, and once the process
                          fail, we set up[self] fail and after that, all operations will be invalid.
          if ( up[self] ) mb[p] := mb[p] ∪ {v} ;   In process n, add v to mb[1] to mb[N]
          Q := Q \ {p} ;   For each element p in Q, we have to broadcast value v to mb[p], after adding v to mb[p],
                           remove p from Q in case of adding v to mb[p] again.

        } ;
      } ;
     if ( up[self] ) pt[self] := pt[self] + 1 ;
PR: await (up[self] ∧ (∀ i ∈ Nodes : pt[i] = pt[self])) ;   If no operations are failed and all process enter in round 1
     d[self] := SetMin(mb[self]) ;   Set the minimum of mb[p] to d[self], if success, all d[p] will be the same.
     t[self] := TRUE ;   The process is terminated
```

```
   } ;
 }
}
```

BEGIN TRANSLATION

VARIABLES $FailNum,\ up,\ pt,\ t,\ d,\ mb,\ pc$

define statement

$SetMin(S) \triangleq$ CHOOSE $i \in S : \forall\, j \in S : i \leq j$

VARIABLES $v,\ Q$

$vars \triangleq \langle FailNum,\ up,\ pt,\ t,\ d,\ mb,\ pc,\ v,\ Q \rangle$

$ProcSet \triangleq (Nodes)$

$Init \triangleq$ Global variables
$\land\ FailNum = FAILNUM$
$\land\ up = [n \in Nodes \mapsto \text{TRUE}]$
$\land\ pt = [n \in Nodes \mapsto 0]$
$\land\ t = [n \in Nodes \mapsto \text{FALSE}]$
$\land\ d = [n \in Nodes \mapsto -1]$
$\land\ mb = [n \in Nodes \mapsto \{\}]$
Process n
$\land\ v = [self \in Nodes \mapsto 0]$
$\land\ Q = [self \in Nodes \mapsto \{\}]$
$\land\ pc = [self \in ProcSet \mapsto \text{“P”}]$

$P(self) \triangleq \land\ pc[self] = \text{“P”}$
$\land$ IF $up[self]$
THEN $\land\ v' = [v$ EXCEPT $![self] = self]$
$\land\ Q' = [Q$ EXCEPT $![self] = Nodes]$
$\land\ pc' = [pc$ EXCEPT $![self] = \text{“PS”}]$
ELSE $\land\ pc' = [pc$ EXCEPT $![self] = \text{“Done”}]$
$\land$ UNCHANGED $\langle v,\ Q \rangle$
$\land$ UNCHANGED $\langle FailNum,\ up,\ pt,\ t,\ d,\ mb \rangle$

$PS(self) \triangleq \land\ pc[self] = \text{“PS”}$
$\land$ IF $up[self] \land Q[self] \neq \{\}$
THEN $\land \exists\, p \in Q[self] :$
$\land$ IF $FailNum > 0 \land up[self]$
THEN $\land\ \lor\ \land\ up' = [up$ EXCEPT $![self] = \text{FALSE}]$
$\land\ FailNum' = FailNum - 1$
$\lor\ \land$ TRUE
$\land$ UNCHANGED $\langle FailNum,\ up \rangle$
ELSE $\land$ TRUE
$\land$ UNCHANGED $\langle FailNum,\ up \rangle$
$\land$ IF $up'[self]$

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{THEN} \ \land\, mb' = [mb \ \text{EXCEPT} \ ![p] = mb[p] \cup \{v[self]\}] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{ELSE} \ \land \text{TRUE} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \land\, mb' = mb \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \land\, Q' = [Q \ \text{EXCEPT} \ ![self] = Q[self] \setminus \{p\}] \\
\qquad\qquad\qquad\qquad\qquad\quad \land\, pc' = [pc \ \text{EXCEPT} \ ![self] = \text{“PS”}] \\
\qquad\qquad\qquad\qquad\qquad\quad \land\, pt' = pt \\
\qquad\qquad\qquad\qquad \text{ELSE} \ \land \text{IF} \ up[self] \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{THEN} \ \land\, pt' = [pt \ \text{EXCEPT} \ ![self] = pt[self] + 1] \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{ELSE} \ \land \text{TRUE} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \land\, pt' = pt \\
\qquad\qquad\qquad\qquad\qquad\quad \land\, pc' = [pc \ \text{EXCEPT} \ ![self] = \text{“PR”}] \\
\qquad\qquad\qquad\qquad\qquad\quad \land \text{UNCHANGED} \ \langle FailNum,\, up,\, mb,\, Q\rangle \\
\qquad\qquad\qquad\quad \land \text{UNCHANGED} \ \langle t,\, d,\, v\rangle
\end{array}
$$

$$
\begin{array}{l}
PR(self) \ \triangleq \ \land\, pc[self] = \text{“PR”} \\
\qquad\qquad\quad \land\, (up[self] \land (\forall\, i \in Nodes : pt[i] = pt[self])) \\
\qquad\qquad\quad \land\, d' = [d \ \text{EXCEPT} \ ![self] = SetMin(mb[self])] \\
\qquad\qquad\quad \land\, t' \ = [t \ \text{EXCEPT} \ ![self] \ = \text{TRUE}] \\
\qquad\qquad\quad \land\, pc' = [pc \ \text{EXCEPT} \ ![self] = \text{“Done”}] \\
\qquad\qquad\quad \land \text{UNCHANGED} \ \langle FailNum,\, up,\, pt,\, mb,\, v,\, Q\rangle
\end{array}
$$

$$
n(self) \ \triangleq \ P(self) \lor PS(self) \lor PR(self)
$$

$$
\begin{array}{l}
Next \ \triangleq \ (\exists\, self \in Nodes : n(self)) \\
\qquad\quad \lor \ \text{Disjunct to prevent deadlock on termination} \\
\qquad\quad\ ((\forall\, self \in ProcSet : pc[self] = \text{“Done”}) \land \text{UNCHANGED} \ vars)
\end{array}
$$

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

$$
Termination \ \triangleq \ \Diamond(\forall\, self \in ProcSet : pc[self] = \text{“Done”})
$$

END TRANSLATION

$$
agreement \ \triangleq \ (\forall\, i,\, j \in Nodes : t[i] \land t[j] \Rightarrow d[i] = d[j])
$$

*When no crash happened, the program will excute without any problems and all $d[i]$
*will be the same. However, if some nodes crash, the crashed process will be in round 0
*and other uncrashed processes will be in round 1, so other uncrashed processes will
*not execute $SetMin(S)$, so all $d[i]$ will be -1. Only if all processes are not crashed can
*we get minimum value.

* Modification History
* Last modified *Tue Oct* 24 21:46:27 *EDT* 2017 by *lenovo*
* Created *Wed Oct* 11 00:01:22 *EDT* 2017 by *lenovo*