MODULE *BenOr*

* *BenOr* algorithm

* *Copr*. (c) Ani & *Pratik*, *Nov* 22, 2019

EXTENDS *Integers*, *TLC*, *Sequences*

CONSTANT N, F, $MAXROUND$, $INPUT$

* N Nodes, F Failures

ASSUME $N \in Nat \land F \in Nat \land F \leq N$

* Assuming $INPUT$ is a valid Sequence in the form $\langle x, \ldots, N \rangle$

$Procs \triangleq 1 \ldots N$

```
--algorithm BenOr
{
    variable msgBrd = {};

    macro SendP1( i, r )
    {
        \* Sends initial value i
        msgBrd := msgBrd ∪ {[type ↦ "p1v", value ↦ i, round ↦ r]}
    }

    macro RvcP1( )
    {
        skip;
    }

    fair process ( p ∈ Procs )
    variable r = 1, p1v = INPUT[self], p2v = -1, decided = -1;
    {
        broadcast: while ( TRUE )
        {
            \* SendP1 → macro which will post the value of that node to the message board as p1v
            print p1v;
            P1S: SendP1(p1v, r);
            \* RvcP1 → Get n-f values with p1v
            P1R: RvcP1(r);
            \* Logic can be included here or in the above macro. Basically we need to finalize b[p] ≜ v if there is a ma
            \* SendP2 → Macro which sends the b value of the node to the message board as p2v
            \* RvcP2 → Get n-f values with p2v
            \* Logic can be included here or in the above macro. Basically need to finalize
            decided[p] = v if there is a majority v in (n-f),
            else pick random b if some value is not -1 else if all are undecided pick uniformly
            between (0,1)
            r := r + 1;
        }
    }
}
```

BEGIN TRANSLATION

VARIABLES $msgBrd$, pc, r, $p1v$, $p2v$, $decided$

$vars \triangleq \langle msgBrd,\ pc,\ r,\ p1v,\ p2v,\ decided \rangle$

$ProcSet \triangleq (Procs)$

$Init \triangleq$ Global variables
$\land\ msgBrd = \{\}$
Process p
$\land\ r = [self \in Procs \mapsto 1]$
$\land\ p1v = [self \in Procs \mapsto INPUT[self]]$
$\land\ p2v = [self \in Procs \mapsto -1]$
$\land\ decided = [self \in Procs \mapsto -1]$
$\land\ pc = [self \in ProcSet \mapsto \text{“broadcast”}]$

$broadcast(self) \triangleq$ $\land\ pc[self] = \text{“broadcast”}$
$\land\ PrintT(p1v[self])$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“P1S”}]$
$\land$ UNCHANGED $\langle msgBrd,\ r,\ p1v,\ p2v,\ decided \rangle$

$P1S(self) \triangleq$ $\land\ pc[self] = \text{“P1S”}$
$\land\ msgBrd' = (msgBrd \cup \{[type \mapsto \text{“p1v”},\ value \mapsto p1v[self],\ round \mapsto r[self]]\})$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“P1R”}]$
$\land$ UNCHANGED $\langle r,\ p1v,\ p2v,\ decided \rangle$

$P1R(self) \triangleq$ $\land\ pc[self] = \text{“P1R”}$
$\land$ TRUE
$\land\ r' = [r \text{ EXCEPT } ![self] = r[self] + 1]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“broadcast”}]$
$\land$ UNCHANGED $\langle msgBrd,\ p1v,\ p2v,\ decided \rangle$

$p(self) \triangleq broadcast(self) \lor P1S(self) \lor P1R(self)$

$Next \triangleq (\exists\, self \in Procs : p(self))$

$Spec \triangleq$ $\land\ Init \land \Box[Next]_{vars}$
$\land\ \forall\, self \in Procs : \mathrm{WF}_{vars}(p(self))$

END TRANSLATION