MODULE *hlc*

EXTENDS *Integers*
CONSTANT N, $STOP$, $EPSILON$
ASSUME $N \in Nat \setminus \{0, 1\}$
$Procs \triangleq 1 \,..\, N$
$SetMax(S) \triangleq$ CHOOSE $i \in S : \forall\, j \in S : i \geq j$

Hybrid Logical Clocks naive algorithm

```
--algorithm hlc{
  variable pt = [j ∈ Procs ↦ 0], lc = [j ∈ Procs ↦ 0], mailboxL = [j ∈ Procs ↦ 0],
           mailboxC = [j ∈ Procs ↦ 0], c = [j ∈ Procs ↦ 0], ltmp = 0 ;
  fair process ( j ∈ Procs ) {
    J0: while ( pt[self] < STOP ) {
          either   local or receive event
          J1: {    phy clocks cannot diverge more than EPSILON
              await (∀ k ∈ Procs : pt[self] < pt[k] + EPSILON) ;
              pt[self] := pt[self] + 1 ;
              ltmp := lc[self] ;
              lc[self] := SetMax({ltmp, mailboxL[self], pt[self]}) ;
              if ( (lc[self] = ltmp) ∧ (lc[self] = mailboxL[self]) )
              { c[self] := SetMax({c[self], mailboxC[self]}) + 1 ; }
              else if ( lc[self] = ltmp )
              { c[self] := c[self] + 1 ; }
              else if ( lc[self] = mailboxL[self] )
              { c[self] := mailboxC[self] + 1 ; }
              else
              { c[self] := 0 ; }
              ;
           }
          or   send event
          J2: {    phy clocks cannot diverge more than EPSILON
              await (∀ k ∈ Procs : pt[self] < pt[k] + EPSILON) ;
              pt[self] := pt[self] + 1 ;
              ltmp := lc[self] ;
              lc[self] := SetMax({ltmp, pt[self]}) ;
              if ( lc[self] = ltmp )
              { c[self] := c[self] + 1 ; }
              else
              { c[self] := 0 ; }
              ;
              mailboxL[(self%N) + 1] := lc[self] ;
              mailboxC[(self%N) + 1] := c[self] ;
           }
        }
  }
```

}

BEGIN TRANSLATION

VARIABLES pt, lc, $mailboxL$, $mailboxC$, c, $ltmp$, pc

$vars \triangleq \langle pt, lc, mailboxL, mailboxC, c, ltmp, pc \rangle$

$ProcSet \triangleq (Procs)$

$Init \triangleq$ Global variables
$\land pt = [j \in Procs \mapsto 0]$
$\land lc = [j \in Procs \mapsto 0]$
$\land mailboxL = [j \in Procs \mapsto 0]$
$\land mailboxC = [j \in Procs \mapsto 0]$
$\land c = [j \in Procs \mapsto 0]$
$\land ltmp = 0$
$\land pc = [self \in ProcSet \mapsto \text{“J0”}]$

$J0(self) \triangleq \land pc[self] = \text{“J0”}$
$\land$ IF $pt[self] < STOP$
THEN $\land \lor \land pc' = [pc$ EXCEPT $![self] = \text{“J1”}]$
$\lor \land pc' = [pc$ EXCEPT $![self] = \text{“J2”}]$
ELSE $\land pc' = [pc$ EXCEPT $![self] = \text{“Done”}]$
$\land$ UNCHANGED $\langle pt, lc, mailboxL, mailboxC, c, ltmp \rangle$

$J1(self) \triangleq \land pc[self] = \text{“J1”}$
$\land (\forall k \in Procs : pt[self] < pt[k] + EPSILON)$
$\land pt' = [pt$ EXCEPT $![self] = pt[self] + 1]$
$\land ltmp' = lc[self]$
$\land lc' = [lc$ EXCEPT $![self] = SetMax(\{ltmp', mailboxL[self], pt'[self]\})]$
$\land$ IF $(lc'[self] = ltmp') \land (lc'[self] = mailboxL[self])$
THEN $\land c' = [c$ EXCEPT $![self] = SetMax(\{c[self], mailboxC[self]\}) + 1]$
ELSE $\land$ IF $lc'[self] = ltmp'$
THEN $\land c' = [c$ EXCEPT $![self] = c[self] + 1]$
ELSE $\land$ IF $lc'[self] = mailboxL[self]$
THEN $\land c' = [c$ EXCEPT $![self] = mailboxC[self] + 1]$
ELSE $\land c' = [c$ EXCEPT $![self] = 0]$
$\land pc' = [pc$ EXCEPT $![self] = \text{“J0”}]$
$\land$ UNCHANGED $\langle mailboxL, mailboxC \rangle$

$J2(self) \triangleq \land pc[self] = \text{“J2”}$
$\land (\forall k \in Procs : pt[self] < pt[k] + EPSILON)$
$\land pt' = [pt$ EXCEPT $![self] = pt[self] + 1]$
$\land ltmp' = lc[self]$
$\land lc' = [lc$ EXCEPT $![self] = SetMax(\{ltmp', pt'[self]\})]$
$\land$ IF $lc'[self] = ltmp'$
THEN $\land c' = [c$ EXCEPT $![self] = c[self] + 1]$
ELSE $\land c' = [c$ EXCEPT $![self] = 0]$

$\land\ mailboxL' = [mailboxL \text{ EXCEPT } ![(self\%N) + 1] = lc'[self]]$
$\land\ mailboxC' = [mailboxC \text{ EXCEPT } ![(self\%N) + 1] = c'[self]]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“J0”}]$

$j(self) \triangleq J0(self) \lor J1(self) \lor J2(self)$

$Next \triangleq (\exists\, self \in Procs : j(self))$
$\lor$ Disjunct to prevent deadlock on termination
$((\forall\, self \in ProcSet : pc[self] = \text{“Done”}) \land \text{UNCHANGED } vars)$

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

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

END TRANSLATION

$TypeOK \triangleq (\forall\, k \in Procs : lc[k] \geq pt[k])$
$Sync \triangleq (\forall\, k,\, l \in Procs : pt[k] \leq pt[l] + EPSILON)$
$Bounded \triangleq (\forall\, k \in Procs : lc[k] < pt[k] + N * (EPSILON + 1))$
$BoundedC \triangleq (\forall\, k \in Procs : c[k] \leq N * (EPSILON + 1))$

\ * Modification History
\ * Last modified *Thu Oct* 30 22:56:28 *EDT* 2014 by *Siddharth*
\ * Created *Thu Oct* 30 21:04:25 *EDT* 2014 by *Siddharth* This is the implementation of the *HLC*. Here we model check to prove the boundedness, which in the case of naive algorithm was divergent. We also model check that c is bounded.

Project submitted by *Siddharth Krishna Sinha* (*ssinha4@buffalo.edu*)
