```
──────────────────────── MODULE hvc ────────────────────────
EXTENDS Integers, TLC
CONSTANT N, STOP, EPS
ASSUME N ∈ Nat
Procs ≜ 1 . . N

SetMax(S)  ≜ CHOOSE i ∈ S : ∀ j ∈ S : i ≥ j
Max(X, Y) ≜ IF X > Y THEN X ELSE  Y

 Hybrid Vector Clocks algorithm
--algorithm hvc{
  variable pt = [j ∈ Procs ↦ 0] ; msg = [j ∈ Procs ↦ [k ∈ Procs ↦ 0]] ;

  fair process ( j ∈ Procs )
  variable v = [k  ∈ Procs ↦ 0] ;
  { J0: while ( pt[self] < STOP ) {
              either
              Recv: {  local or receive event
                       await (∀ k ∈ Procs : pt[self] < pt[k] + EPS) ;  NTP clock sync
                       pt[self] := pt[self] + 1 ;
                       v := [k ∈ Procs ↦ IF  k = self  THEN  pt[self]  ELSE  Max(v[k], msg[self][k])] ;
                   }
              or
              Send: {  send event
                       await (∀ k ∈ Procs : pt[self] < pt[k] + EPS) ;  NTP clock sync
                       pt[self] := pt[self] + 1 ;
                       v[self] := pt[self] ;
                       with ( k ∈ Procs \ {self} ) {
                           msg[k] := v ;
                        }
                   }
          }
   }
 }
 BEGIN TRANSLATION
VARIABLES pt, msg, pc, v

vars ≜ ⟨pt, msg, pc, v⟩

ProcSet ≜ (Procs)

Init ≜  Global variables
        ∧ pt = [j ∈ Procs ↦ 0]
        ∧ msg = [j ∈ Procs ↦ [k ∈ Procs ↦ 0]]
         Process j
        ∧ v = [self ∈ Procs ↦ [k ∈ Procs ↦ 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 $![self] = \text{"Recv"}]$
                         $\lor \land\ pc' = [pc$ EXCEPT $![self] = \text{"Send"}]$
                  ELSE $\land\ pc' = [pc$ EXCEPT $![self] = \text{"Done"}]$
             $\land$ UNCHANGED $\langle pt, msg, v \rangle$

$Recv(self) \triangleq$ $\land\ pc[self] = \text{"Recv"}$
               $\land\ (\forall k \in Procs : pt[self] < pt[k] + EPS)$
               $\land\ pt' = [pt$ EXCEPT $![self] = pt[self] + 1]$
               $\land\ v' = [v$ EXCEPT $![self] = [k \in Procs \mapsto$ IF $k = self$ THEN $pt'[self]$ ELSE $Max(v[self][k], msg$
               $\land\ pc' = [pc$ EXCEPT $![self] = \text{"J0"}]$
               $\land\ msg' = msg$

$Send(self) \triangleq$ $\land\ pc[self] = \text{"Send"}$
               $\land\ (\forall k \in Procs : pt[self] < pt[k] + EPS)$
               $\land\ pt' = [pt$ EXCEPT $![self] = pt[self] + 1]$
               $\land\ v' = [v$ EXCEPT $![self][self] = pt'[self]]$
               $\land\ \exists k \in Procs \setminus \{self\} :$
                    $msg' = [msg$ EXCEPT $![k] = v'[self]]$
               $\land\ pc' = [pc$ EXCEPT $![self] = \text{"J0"}]$

$j(self) \triangleq J0(self) \lor Recv(self) \lor Send(self)$

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

 Boundedness
$TypeOK \triangleq (\forall k \in Procs : v[k][k] = pt[k])$
$Sync \triangleq (\forall k, m \in Procs : pt[k] \leq pt[m] + EPS)$
$Safety1 \triangleq (\forall k \in Procs : v[k][k] \geq pt[k] \land v[k][k] \leq pt[k] + EPS)$
$Safety2 \triangleq (\forall k, l \in Procs : v[k][k] \geq v[l][k])$
 Stabilization
```