```
┌──────────────────── MODULE VectorClocks ────────────────────┐
EXTENDS Integers, Sequences, FiniteSets, Utils

CONSTANT
    P  the set of processes

VectorClock ≜ [P → Int]
v1 ⪯ v2 ≜ ∀ p ∈ P : v1[p] ≤ v2[p]
v1 ≺ v2 ≜ v1 ⪯ v2 ∧ ∃ p ∈ P : v1[p] < v2[p]

Msg ≜ [sender : P, clock : VectorClock]

Event ≜ (P × {"send", "deliver"} × Msg) ∪ (P × {"init"})
EventOrdering ≜ SUBSET (Event × Event)

VARIABLES
    happensBefore  a ghost variable tracking the happens-before relation
,   clock  the vector clock
,   sent  the set of messages sent
,   localEvents

TypeOK ≜
    ∧ happensBefore ∈ EventOrdering
    ∧ clock ∈ [P → VectorClock]
    ∧ sent ∈ SUBSET Msg
    ∧ localEvents ∈ [P → SUBSET Event]

zero ≜ [p ∈ P ↦ 0]

Init ≜
    ∧ happensBefore = {}
    ∧ clock = [p ∈ P ↦ zero]
    ∧ sent = {}
    ∧ localEvents = [p ∈ P ↦ {⟨p, "init"⟩}]

MergeClocks(c1, c2) ≜ [p ∈ P ↦ Max(c1[p], c2[p])]
StepClock(p, vc) ≜ [vc EXCEPT ![p] = @ + 1]
DeliverableAt(m, p) ≜
    ∀ k ∈ P :
        ∧ k = m.sender ⇒ m.clock[k] = clock[p][k] + 1
        ∧ k ≠ m.sender ⇒ m.clock[k] ≤ clock[p][k]

SendEvent(m) ≜ ⟨m.sender, "send", m⟩
DeliveryEvent(p, m) ≜ ⟨p, "deliver", m⟩

Deliver(p, m) ≜
    ∧ m ∈ sent
    ∧ DeliverableAt(m, p)
    $\land$ LET $d \triangleq \mathit{DeliveryEvent}(p, m)$
                $s \triangleq \mathit{SendEvent}(m)$
        IN
         $\land$ $\mathit{localEvents}' = [\mathit{localEvents}$ EXCEPT $![p] = @ \cup \{d\}]$
         $\land$ $\mathit{happensBefore}' = \mathit{TransitiveClosure}(\{\langle s, d \rangle\}$
                  $\cup \{\langle e, d \rangle : e \in \mathit{localEvents}[p]\} \cup \mathit{happensBefore})$
    $\land$ $\mathit{clock}' = [\mathit{clock}$ EXCEPT $![p] = \mathit{MergeClocks}(@, m.\mathit{clock})]$
    $\land$ UNCHANGED $\mathit{sent}$

$\mathit{Send}(p) \triangleq$
     $\land$ $\mathit{clock}' = [\mathit{clock}$ EXCEPT $![p] = \mathit{StepClock}(p, @)]$
     $\land$ LET $m \triangleq [\mathit{sender} \mapsto p, \mathit{clock} \mapsto \mathit{clock}'[p]]$
                 $s \triangleq \mathit{SendEvent}(m)$
         IN
          $\land$ $\mathit{sent}' = \mathit{sent} \cup \{m\}$
          $\land$ $\mathit{localEvents}' = [\mathit{localEvents}$ EXCEPT $![p] = @ \cup \{s\}]$
          $\land$ $\mathit{happensBefore}' =$
                 $\mathit{TransitiveClosure}(\{\langle e, s \rangle : e \in \mathit{localEvents}[p]\} \cup \mathit{happensBefore})$

$\mathit{Next} \triangleq \exists p \in P :$
    $\lor$ $\mathit{Send}(p)$
    $\lor$ $\exists m \in \mathit{Msg} : \mathit{Deliver}(p, m)$

$\mathit{vars} \triangleq \langle \mathit{happensBefore}, \mathit{clock}, \mathit{sent}, \mathit{localEvents} \rangle$
$\mathit{Spec} \triangleq$
    $\land$ $\mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}}$
    $\land$ $\forall p \in P, m \in \mathit{Msg} : \mathrm{WF}_{\mathit{vars}}(\mathit{Deliver}(p, m))$

$\mathit{ReflectsAndPreserve} \triangleq$
   $\forall m1, m2 \in \mathit{sent} :$
      $(m1.\mathit{clock} \prec m2.\mathit{clock}) = (\langle \mathit{SendEvent}(m1), \mathit{SendEvent}(m2) \rangle \in \mathit{happensBefore})$

$\mathit{CausalDelivery} \triangleq \forall p \in P :$
   $\forall e1, e2 \in \mathit{localEvents}[p] :$
      $\land$ $e1[2] =$ “deliver”
      $\land$ $e2[2] =$ “deliver”
      $\Rightarrow$ LET $m1 \triangleq e1[3]$
                   $m2 \triangleq e2[3]$
          IN      $\langle \mathit{SendEvent}(m1), \mathit{SendEvent}(m2) \rangle \in \mathit{happensBefore}$
                   $\Rightarrow \langle e1, e2 \rangle \in \mathit{happensBefore}$

$\mathit{Liveness} \triangleq \forall m \in \mathit{Msg} : \forall p \in P :$
   $\Box(m \in \mathit{sent} \land m.\mathit{sender} \neq p \Rightarrow \Diamond(\mathit{DeliveryEvent}(p, m) \in \mathit{localEvents}[p]))$

$\mathit{Canary} \triangleq \neg($
   $\exists p \in P, q \in P : p \neq q \land \mathit{clock}[p][q] > 0$
$)$
```

$\textsc{constant}\ \mathit{IntMax}$
$\mathit{MyInts} \triangleq 0 \mathrel{..} \mathit{IntMax}$

$\mathit{Constraint} \triangleq$
$\quad \forall\, p1,\, p2 \in P : \mathit{clock}[p1][p2] < \mathit{IntMax}$