```
──────────────────── MODULE backpressure ────────────────────
EXTENDS FiniteSets, Integers, Sequences, TLC

Null ≜ 0
Cowns ≜ 1 . . 3
BehaviourLimit ≜ 3
OverloadThreshold ≜ 2
PriorityLevels ≜ {2, 1, 0}

Min(s) ≜ CHOOSE x ∈ s : ∀ y ∈ s \ {x} : y > x
Max(s) ≜ CHOOSE x ∈ s : ∀ y ∈ s \ {x} : y < x
Range(f) ≜ {f[x] : x ∈ DOMAIN f}

VARIABLES fuel, queue, scheduled, running, priority, blocker, mutor
vars ≜ ⟨fuel, queue, scheduled, running, priority, blocker, mutor⟩

Messages ≜ UNION {Range(queue[c]) : c ∈ Cowns}
EmptyQueue(c) ≜ Len(queue[c]) = 0

Init ≜
  ∧ fuel = BehaviourLimit
  ∧ queue = [c ∈ Cowns ↦ ⟨{c}⟩]
  ∧ scheduled = [c ∈ Cowns ↦ TRUE]
  ∧ running = [c ∈ Cowns ↦ FALSE]
  ∧ priority = [c ∈ Cowns ↦ 0]
  ∧ blocker = [c ∈ Cowns ↦ Null]
  ∧ mutor = [c ∈ Cowns ↦ Null]

Terminating ≜
  ∧ ∀ c ∈ Cowns : EmptyQueue(c)
  ∧ UNCHANGED vars

Acquire(cown) ≜
   # Preconditions
  ∧ scheduled[cown]
  ∧ ¬running[cown]
  ∧ ¬EmptyQueue(cown)
  ∧ cown < Max(Head(queue[cown]))
   # Forward the message to the next cown.
  ∧ LET
      msg ≜ Head(queue[cown])
      next ≜ Min({c ∈ msg : c > cown})
    IN
      queue′ =
        (next :> Append(queue[next], msg)) @@
        (cown :> Tail(queue[cown])) @@
        $queue$

    $\land$ UNCHANGED $\langle fuel, scheduled, running, priority, blocker, mutor \rangle$

$PreRun(c) \triangleq$
     # Preconditions
    $\land scheduled[c]$
    $\land \neg running[c]$
    $\land \neg EmptyQueue(c)$
    $\land c = Max(Head(queue[c]))$
     # Set max cown in current message to running
    $\land running' = (c :> \text{TRUE}) @@ running$
    $\land$ UNCHANGED $\langle fuel, queue, scheduled, priority, blocker, mutor \rangle$

$Send(c) \triangleq$
     # Preconditions
    $\land running[c]$
    $\land fuel > 0$
     # Select set of receivers
    $\land \exists receivers \in \{cs \in \text{SUBSET } Cowns : Cardinality(cs) > 1\} :$
       # place message for receivers in the first receiver's queue
      LET $next \triangleq Min(receivers)$IN
      $queue' = (next :> Append(queue[next], receivers)) @@ queue$

    $\land fuel' = fuel - 1$
    $\land$ UNCHANGED $\langle scheduled, running, priority, blocker, mutor \rangle$

$PostRun(c) \triangleq$
     # Preconditions
    $\land running[c]$
     # Transition
    $\land running' = (c :> \text{FALSE}) @@ running$
     # Remove message from queue
    $\land queue' = (c :> Tail(queue[c])) @@ queue$
    $\land$ UNCHANGED $\langle fuel, scheduled, priority, blocker, mutor \rangle$

$RunStep(c) \triangleq$
    $\lor Acquire(c)$
    $\lor PreRun(c)$
    $\lor Send(c)$
    $\lor PostRun(c)$

$Next \triangleq \exists c \in Cowns : RunStep(c)$

$Spec \triangleq$
    $\land Init$
    $\land \Box[Next \lor Terminating]_{vars}$
```

```
  $\land \forall c \in Cowns : \mathrm{WF}_{vars}(RunStep(c))$

 # Properties

 # Ensure that the termination condition is reached by the model.
$Termination \triangleq \Diamond\Box(\forall c \in Cowns : EmptyQueue(c))$

 # Invariants

 # Ensure that the model produces finite messages.
$MessageLimit \triangleq Cardinality(Messages) \leq (Cardinality(Cowns) + BehaviourLimit)$

 # A message must contain at least one *cown*.
$MessagesAreNonEmpty \triangleq \forall m \in Messages : m \neq \{\}$

 # A running *cown* must be scheduled and be the max *cown* in the message at the head of its queue.
$RunningImplication \triangleq$
  $\forall c \in Cowns : running[c] \Rightarrow scheduled[c] \land (c = Max(Head(queue[c])))$
```