MODULE *backpressure*

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

$Null \triangleq 0$
$Cowns \triangleq 1 \,..\, 4$
$MaxMessageCount \triangleq 4$
$MaxMessageSize \triangleq 3$
$OverloadThreshold \triangleq 2$
$PriorityLevels \triangleq \{2, 1, 0\}$

$Min(s) \triangleq \text{CHOOSE } x \in s : \forall\, y \in s \setminus \{x\} : y > x$
$Max(s) \triangleq \text{CHOOSE } x \in s : \forall\, y \in s \setminus \{x\} : y < x$
$Range(f) \triangleq \{f[x] : x \in \text{DOMAIN } f\}$
$Subsets(s, min, max) \triangleq$
  $\{x \in \text{SUBSET } s : (Cardinality(x) \geq min) \land (Cardinality(x) \leq max)\}$

VARIABLES *fuel*, *queue*, *scheduled*, *running*, *priority*, *blocker*, *mutor*
$vars \triangleq \langle fuel, queue, scheduled, running, priority, blocker, mutor\rangle$

$Messages \triangleq \text{UNION } \{Range(queue[c]) : c \in Cowns\}$
$EmptyQueue(c) \triangleq Len(queue[c]) = 0$

$Init \triangleq$
  $\land\, fuel = MaxMessageCount$
  $\land\, queue = [c \in Cowns \mapsto \langle\{c\}\rangle]$
  $\land\, scheduled = [c \in Cowns \mapsto \text{TRUE}]$
  $\land\, running = [c \in Cowns \mapsto \text{FALSE}]$
  $\land\, priority = [c \in Cowns \mapsto 0]$
  $\land\, blocker = [c \in Cowns \mapsto Null]$
  $\land\, mutor = [c \in Cowns \mapsto Null]$

$Terminating \triangleq$
  $\land\, \forall\, c \in Cowns : EmptyQueue(c)$
  ———
  $\land$ UNCHANGED *vars*

$ExternalReceive(cown) \triangleq$
  $\land\, fuel > 0$
  ———
  $\land$ UNCHANGED $\langle scheduled, running, priority, blocker, mutor\rangle$
  $\land\, fuel' = fuel - 1$
  # Receive a message from an external source
  $\land\, \exists\, others \in Subsets(\{c \in Cowns : c > cown\}, 0, MaxMessageSize - 1) :$
    LET $msg \triangleq \{cown\} \cup others$ IN
    $queue' = (cown :> Append(queue[cown], msg)) \,@@\, queue$

$Acquire(cown) \triangleq$

$\land\ scheduled[cown]$
$\land\ \neg running[cown]$
$\land\ \neg EmptyQueue(cown)$
$\land\ cown < Max(Head(queue[cown]))$
———
$\land$ UNCHANGED $\langle fuel,\ scheduled,\ running,\ priority,\ blocker,\ mutor\rangle$
Forward the message to the next *cown*.
$\land$ LET
$msg \triangleq Head(queue[cown])$
$next \triangleq Min(\{c \in msg : c > cown\})$
$q \triangleq (cown :> Tail(queue[cown])) \mathbin{@@} queue$
IN
$queue' = (next :> Append(queue[next],\ msg)) \mathbin{@@} q$

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

$Send(c) \triangleq$
$\land\ running[c]$
$\land\ fuel > 0$
———
$\land$ UNCHANGED $\langle scheduled,\ running,\ priority,\ blocker,\ mutor\rangle$
$\land\ fuel' = fuel - 1$
Select set of receivers
$\land\ \exists\, receivers \in Subsets(Cowns,\ 1,\ MaxMessageSize) :$
place message for receivers in the first receiver's queue
LET $next \triangleq Min(receivers)$IN
$queue' = (next :> Append(queue[next],\ receivers)) \mathbin{@@} queue$

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

$RunStep(c) \triangleq$
$\lor\ ExternalReceive(c) \setminus$ *# Very expensive check

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

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

$Spec \triangleq$
$\land\ Init$
$\land\ \Box[Next \lor Terminating]_{vars}$
$\land\ \forall\, c \in Cowns : \mathrm{WF}_{vars}(RunStep(c))$

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.
$MessageLimit \triangleq Cardinality(Messages) \leq (Cardinality(Cowns) + MaxMessageCount)$

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.
$RunningImplication \triangleq$
$\forall\, c \in Cowns : running[c] \Rightarrow scheduled[c] \land (c = Max(Head(queue[c])))$