MODULE *backpressure*

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

$Null \triangleq 0$
$Cowns \triangleq 1 \,..\, 4$
$BehaviourLimit \triangleq 4$
$OverloadThreshold \triangleq 2$
$PriorityLevels \triangleq \{-1, 0, 1\}$

$Min(s) \triangleq \text{CHOOSE } x \in s : \forall\, y \in s \setminus \{x\} : y > x$
$Max(s) \triangleq \text{CHOOSE } x \in s : \forall\, y \in s \setminus \{x\} : y < x$

$Range(f) \triangleq \{f[x] : x \in \text{DOMAIN } f\}$

$Pick(s) \triangleq \text{CHOOSE } x \in s : \text{TRUE}$

$ReduceSet(op(\_, \_), set, acc) \triangleq$
  LET $f[s \in \text{SUBSET } set] \triangleq$
    IF $s = \{\}$ THEN acc ELSE LET $x \triangleq Pick(s)$IN $op(x, f[s \setminus \{x\}])$
  IN $f[set]$

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

$Sleeping(c) \triangleq scheduled[c] \land (Len(queue[c]) = 0)$

$Available(c) \triangleq scheduled[c] \land (Len(queue[c]) > 0)$

$Overloaded(c) \triangleq Len(queue[c]) > OverloadThreshold$

$Muted(c) \triangleq c \in \text{UNION } Range(mute)$

$CurrentMessage(c) \triangleq \text{IF } Len(queue[c]) > 0 \text{ THEN } Head(queue[c]) \text{ ELSE } \{\}$

$LowPriority(cs) \triangleq \{c \in cs : priority[c] = -1\}$

$HighPriority(cs) \triangleq \{c \in cs : priority[c] = 1\}$

$RequiresPriority(c) \triangleq$
  $\lor\, Overloaded(c)$
  $\lor \exists\, m \in Range(queue[c]) : \exists\, k \in m \setminus \{c\} : priority[k] = 1$

RECURSIVE $Blockers(\_)$
$Blockers(c) \triangleq$
  IF $blocker[c] = Null$ THEN $\{\}$ ELSE $\{blocker[c]\} \cup Blockers(blocker[c])$
$Prioritizing(cs) \triangleq$
  LET $unprioritized \triangleq \{c \in cs : priority[c] < 1\}$IN
  $unprioritized \cup \text{UNION } \{Blockers(c) : c \in unprioritized\}$

$ValidMutor(c) \triangleq$
$\lor (priority[c] = 1) \land Overloaded(c)$
$\lor (priority[c] = -1)$

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

$Terminating \triangleq$
$\land \forall\, c \in Cowns : Sleeping(c)$
$\land$ UNCHANGED $vars$

$Acquire(cown) \triangleq$
LET $msg \triangleq CurrentMessage(cown)$IN
$\land Available(cown)$
$\land cown < Max(msg)$
$\land$ IF $\exists\, c \in msg : priority[c] = 1$ THEN
LET $prioritizing \triangleq Prioritizing(\{c \in msg : c > cown\})$IN
LET $unmuting \triangleq LowPriority(prioritizing)$IN
$\land priority' = [c \in prioritizing \mapsto 1] \mathbin{@@} priority$
$\land scheduled' = (cown :> \text{FALSE}) \mathbin{@@} [c \in unmuting \mapsto \text{TRUE}] \mathbin{@@} scheduled$
ELSE
$\land scheduled' = (cown :> \text{FALSE}) \mathbin{@@} scheduled$
$\land$ UNCHANGED $\langle priority, mute\rangle$
$\land$ LET $next \triangleq Min(\{c \in msg : c > cown\})$IN
$\land blocker' = (cown :> next) \mathbin{@@} blocker$
$\land$ LET $q \triangleq (cown :> Tail(queue[cown])) \mathbin{@@} queue$IN
$queue' = (next :> Append(queue[next], msg)) \mathbin{@@} q$
$\land$ UNCHANGED $\langle fuel, running, mutor, mute\rangle$

$Prerun(cown) \triangleq$
LET $msg \triangleq CurrentMessage(cown)$IN
$\land scheduled[cown]$
$\land \neg running[cown]$
$\land$ IF $msg = \{\}$ THEN FALSE ELSE $cown = Max(msg)$
$\land priority' = (cown :>$ IF $RequiresPriority(cown)$ THEN 1 ELSE $0) \mathbin{@@} priority$
$\land running' = (cown :> \text{TRUE}) \mathbin{@@} running$
$\land blocker' = [c \in msg \mapsto Null] \mathbin{@@} blocker$
$\land$ UNCHANGED $\langle fuel, queue, scheduled, mutor, mute\rangle$

$Send(cown) \triangleq$
 LET $senders \triangleq CurrentMessage(cown)$IN
  $\land\ running[cown]$
  $\land\ fuel > 0$
  $\land\ \exists\, receivers \in$ SUBSET $Cowns$:
   $\land\ Cardinality(receivers) > 0$
   $\land\ queue' =$
    $(Min(receivers) :> Append(queue[Min(receivers)],\ receivers))$ @@ $queue$
   $\land$ IF $\exists\, c \in receivers : priority[c] = 1$ THEN
    LET $prioritizing \triangleq Prioritizing(receivers)$IN
    LET $unmuting \triangleq LowPriority(prioritizing)$IN
     $\land\ priority' = [c \in prioritizing \mapsto 1]$ @@ $priority$
     $\land\ scheduled' = [c \in unmuting \mapsto$ TRUE$]$ @@ $scheduled$
     $\land$ LET $mutors \triangleq \{c \in receivers \setminus senders : ValidMutor(c)\}$IN
      IF
       $\land\ mutors \neq \{\}$
       $\land\ mutor[cown] = Null$
       $\land\ \forall\, c \in senders : priority[c] = 0$
       $\land\ \forall\, c \in senders : c \notin receivers$ $TODO$: justify
      THEN
       $\land\ mutor' = (cown :> Min(mutors))$ @@ $mutor$
      ELSE
       $\land$ UNCHANGED $\langle mutor \rangle$
    ELSE
     $\land$ UNCHANGED $\langle scheduled,\ priority,\ mutor \rangle$
  $\land\ fuel' = fuel - 1$
  $\land$ UNCHANGED $\langle running,\ blocker,\ mute \rangle$

$Complete(cown) \triangleq$
 LET $msg \triangleq CurrentMessage(cown)$IN
  $\land\ running[cown]$
  $\land$ IF $mutor[cown] \neq Null$ THEN
    LET $muting \triangleq \{c \in msg : priority[c] = 0\}$IN
     $\land\ priority' = [c \in muting \mapsto -1]$ @@ $priority$
     $\land\ mute' = (mutor[cown] :> mute[mutor[cown]] \cup muting)$ @@ $mute$
     $\land\ scheduled' = [c \in msg \mapsto c \notin muting]$ @@ $scheduled$
    ELSE
     $\land\ scheduled' = [c \in msg \mapsto$ TRUE$]$ @@ $scheduled$
     $\land\ priority' =$
      $(cown :>$ IF $Len(queue[cown]) = 1$ THEN 0 ELSE $priority[cown])$ @@
      $[c \in msg \setminus \{cown\} \mapsto$ IF $Len(queue[c]) = 0$ THEN 0 ELSE $priority[c]]$ @@
      $priority$
     $\land$ UNCHANGED $\langle mute \rangle$
  $\land\ queue' = (cown :> Tail(queue[cown]))$ @@ $queue$
  $\land\ running' = (cown :>$ FALSE$)$ @@ $running$

$\land\ mutor' = (cown :> Null)\ @@\ mutor$
$\land$ UNCHANGED $\langle fuel,\ blocker\rangle$

$Unmute\ \triangleq$
LET $invalid\_keys\ \triangleq\ \{c \in$ DOMAIN $mute : priority[c] = 0\}$IN
LET $unmuting\ \triangleq$ UNION $Range([k \in invalid_keys \mapsto LowPriority(mute[k])])$IN
$\land\ unmuting \neq \{\}$
$\land\ priority' = [c \in unmuting \mapsto 0]\ @@\ priority$
$\land\ mute' = [c \in invalid\_keys \mapsto \{\}]\ @@\ mute$
$\land\ scheduled'\ = [c \in unmuting \mapsto$ TRUE$]\ @@\ scheduled$
$\land$ UNCHANGED $\langle fuel,\ queue,\ running,\ blocker,\ mutor\rangle$

$Run(cown)\ \triangleq$
$\lor\ Acquire(cown)$
$\lor\ Prerun(cown)$
$\lor\ Send(cown)$
$\lor\ Complete(cown)$

$Next\ \triangleq\ Terminating \lor \exists\, c \in Cowns : Run(c) \lor Unmute$

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

$MessageLimit\ \triangleq$
LET $msgs\ \triangleq\ ReduceSet($LAMBDA $c,\ sum : sum + Len(queue[c]),\ Cowns,\ 0)$IN
$msgs \leq (BehaviourLimit + Max(Cowns))$

$RunningIsScheduled\ \triangleq$
$\forall\, c \in Cowns : running[c] \Rightarrow scheduled[c] \land (c = Max(CurrentMessage(c)))$

$CownNotMutedBySelf\ \triangleq\ \forall\, c \in Cowns : c \notin mute[c]$

$LowPriorityMuted\ \triangleq\ \forall\, c \in Cowns : (priority[c] = -1) \Rightarrow Muted(c)$

$WillScheduleCown\ \triangleq\ \exists\, c \in Cowns :$
$\lor\ scheduled[c]$
$\lor$
$\land\ priority[c] = -1$
$\land\ \exists\, k \in$ DOMAIN $mute : (c \in mute[k]) \land (priority[k] = 0)$

$Nonblocking\ \triangleq$
$\forall\, c \in Cowns : \forall\, m \in Range(queue[c]) :$
$\neg(\exists\, h \in HighPriority(m) : \exists\, l \in LowPriority(m) : (h < c) \land (l \leq c))$

$RunningNotBlocked\ \triangleq$

$$\forall\, c \in Cowns : running[c] \Rightarrow (\forall\, k \in CurrentMessage(c) : blocker[k] = Null)$$

$$Acquired(c) \triangleq \exists\, k \in Cowns : (k > c) \wedge (c \in \text{UNION}\ Range(queue[k]))$$
$$UnscheduledByMuteOrAcquire \triangleq$$
$$\forall\, c \in Cowns : \neg((priority[c] = -1) \vee Acquired(c)) \equiv scheduled[c]$$

$$BehaviourAcquisition \triangleq$$
$$\forall\, c \in Cowns : \forall\, k \in \text{UNION}\ Range(queue[c]) : (k < c) \Rightarrow \neg scheduled[k]$$

$$AcquiredBy(a,\, b) \triangleq (a < b) \wedge (a \in \text{UNION}\ Range(queue[b]))$$
$$AcquiredOnce \triangleq$$
$$\forall\, a \in Cowns : \forall\, b \in Cowns : \forall\, c \in Cowns :$$
$$(AcquiredBy(a,\, b) \wedge AcquiredBy(a,\, c)) \Rightarrow (b = c)$$

$$SelfInCurrentMessage \triangleq$$
$$\forall\, c \in Cowns : (Len(queue[c]) > 0) \Rightarrow (c \in CurrentMessage(c))$$

$$HighPriorityInQueue \triangleq$$
$$\forall\, c \in Cowns : (priority[c] = 1) \Rightarrow$$
$$\exists\, k \in Cowns : c \in \text{UNION}\ Range(queue[k])$$

$$Required(c) \triangleq \exists\, k \in Cowns : (k < c) \wedge (c \in \text{UNION}\ Range(queue[k]))$$
$$SleepingIsNormalOrRequired \triangleq$$
$$\forall\, c \in Cowns : Sleeping(c) \Rightarrow ((priority[c] = 0) \vee Required(c))$$

$$MuteSetsDisjoint \triangleq$$
$$\forall\, c \in Cowns : \forall\, k \in Cowns :$$
$$((mute[c] \cap mute[k]) \neq \{\}) \Rightarrow (c = k)$$

$$Termination \triangleq \Diamond\Box(\forall\, c \in Cowns : Sleeping(c))$$

$$SomeCownWillBeScheduled \triangleq \Box\Diamond(\exists\, c \in Cowns : scheduled[c])$$