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
  $unprioritized \cup \text{UNION } \{Blockers(c) : c \in unprioritized\}$

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

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

$Terminating \triangleq$
$\land \forall\, c \in Cowns : Sleeping(c)$
$\land$ 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
$\land scheduled[cown]$
$\land$ 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 ≜ CurrentMessage(cown)IN
  ∧ running[cown]
  ∧ fuel > 0
  ∧ ∃ receivers ∈ SUBSET Cowns :
    ∧ Cardinality(receivers) > 0
    ∧ queue' =
      (Min(receivers) :> Append(queue[Min(receivers)], receivers)) @@ queue
    ∧ IF ∃ c ∈ receivers : priority[c] = 1 THEN
      LET prioritizing ≜ Prioritizing(receivers)IN
      LET unmuting ≜ LowPriority(prioritizing)IN
       ∧ priority' = [c ∈ prioritizing ↦ 1] @@ priority
       ∧ scheduled' = [c ∈ unmuting ↦ TRUE] @@ scheduled
       ∧ LET mutors ≜ {c ∈ receivers \ senders : ValidMutor(c)}IN
         IF
            ∧ mutors ≠ {}
            ∧ mutor[cown] = Null
            ∧ ∀ c ∈ senders : priority[c] = 0
            ∧ ∀ c ∈ senders : c ∉ receivers  TODO: justify
         THEN
            ∧ mutor' = (cown :> Min(mutors)) @@ mutor
         ELSE
            ∧ UNCHANGED ⟨mutor⟩
      ELSE
        ∧ UNCHANGED ⟨scheduled, priority, mutor⟩
  ∧ fuel' = fuel − 1
  ∧ UNCHANGED ⟨running, blocker, mute⟩

Complete(cown) ≜
  LET msg ≜ CurrentMessage(cown)IN
  ∧ running[cown]
  ∧ IF mutor[cown] ≠ Null THEN
      LET muting ≜ {c ∈ msg : priority[c] = 0}IN
       ∧ priority' = [c ∈ muting ↦ −1] @@ priority
       ∧ mute' = (mutor[cown] :> mute[mutor[cown]] ∪ muting) @@ mute
       ∧ scheduled' = [c ∈ msg ↦ c ∉ muting] @@ scheduled
     ELSE
       ∧ scheduled' = [c ∈ msg ↦ TRUE] @@ scheduled
       ∧ UNCHANGED ⟨priority, mute⟩
  ∧ queue' = (cown :> Tail(queue[cown])) @@ queue
  ∧ running' = (cown :> FALSE) @@ running
  ∧ mutor' = (cown :> Null) @@ mutor
  ∧ UNCHANGED ⟨fuel, blocker⟩

Unmute ≜
  LET invalid_keys ≜ {c ∈ DOMAIN mute : (priority[c] = 0) ∨ Sleeping(c)}IN
```

$$
\begin{array}{l}
\quad \text{LET } unmuting \triangleq \text{UNION } Range([k \in invalid\_keys \mapsto LowPriority(mute[k])]) \text{IN} \\
\quad \land\ unmuting \neq \{\} \\
\quad \land\ priority' = [c \in unmuting \mapsto 0] \mathbin{@@} priority \\
\quad \land\ mute' = [c \in invalid\_keys \mapsto \{\}] \mathbin{@@} mute \\
\quad \land\ scheduled' = [c \in unmuting \mapsto \text{TRUE}] \mathbin{@@} scheduled \\
\quad \land\ \text{UNCHANGED } \langle fuel,\ queue,\ running,\ blocker,\ mutor \rangle
\end{array}
$$

$$
\begin{array}{l}
Run(cown) \triangleq \\
\quad \lor\ Acquire(cown) \\
\quad \lor\ Prerun(cown) \\
\quad \lor\ Send(cown) \\
\quad \lor\ Complete(cown)
\end{array}
$$

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

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

$$
\begin{array}{l}
MessageLimit \triangleq \\
\quad \text{LET } msgs \triangleq ReduceSet(\text{LAMBDA } c,\ sum : sum + Len(queue[c]),\ Cowns,\ 0) \text{IN} \\
\quad msgs \leq (BehaviourLimit + Max(Cowns))
\end{array}
$$

$$
RunningIsScheduled \triangleq \forall\, c \in Cowns : running[c] \Rightarrow scheduled[c]
$$

$$
LowPriorityNotScheduled \triangleq \forall\, c \in Cowns : (priority[c] = -1) \Rightarrow \neg scheduled[c]
$$

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

$$
\begin{array}{l}
BehaviourAcquisition \triangleq \\
\quad \forall\, c \in Cowns : scheduled[c] \Rightarrow \\
\qquad \neg(\exists\, k \in Cowns : (k > c) \land (c \in \text{UNION } Range(queue[k])))
\end{array}
$$

$$
\begin{array}{l}
Nonblocking \triangleq \\
\quad \forall\, c \in Cowns : \forall\, m \in Range(queue[c]) : \\
\qquad \neg(\exists\, h \in HighPriority(m) : \exists\, l \in LowPriority(m) : (h < c) \land (l \leq c))
\end{array}
$$

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

$$
\begin{array}{l}
OverloadRaisesPriority \triangleq \\
\quad \forall\, c \in Cowns : (scheduled[c] \land Overloaded(c)) \Rightarrow (priority[c] = 1)
\end{array}
$$