MODULE *backpressure*

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

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

$Pick(s) \triangleq \text{CHOOSE } x \in s : \text{TRUE}$
$Min(s) \triangleq \text{CHOOSE } x \in s : \forall\, y \in s \setminus \{x\} : y > x$
$Max(s) \triangleq \text{CHOOSE } x \in s : \forall\, y \in s \setminus \{x\} : y < x$
$Range(f) \triangleq \{f[x] : x \in \text{DOMAIN } f\}$
$Subsets(s, min, max) \triangleq$
  $\{x \in \text{SUBSET } s : (Cardinality(x) \geq min) \land (Cardinality(x) \leq max)\}$
RECURSIVE $Concat(\_)$
$Concat(s) \triangleq \text{IF } s = \{\} \text{ THEN } \langle\rangle \text{ ELSE LET } x \triangleq Pick(s) \text{IN } x \circ Concat(s \setminus \{x\})$

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

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

$Normal(c) \triangleq priority[c] = 0$
$Prioritized(c) \triangleq priority[c] = 1$
$Muted(c) \triangleq priority[c] = -1$

$EmptyQueue(c) \triangleq Len(queue[c]) = 0$
$Overloaded(c) \triangleq Len(queue[c]) \geq OverloadThreshold$
$Enqueue(c, m) \triangleq c :> Append(queue[c], m)$
$Dequeue(c) \triangleq c :> Tail(queue[c])$

RECURSIVE $Blockers(\_)$
$Blockers(c) \triangleq$
  IF $blocker[c] = Null$ THEN $\{\}$
  ELSE $\{blocker[c]\} \cup Blockers(blocker[c])$

$Unblock(c) \triangleq [k \in \{c\} \cup Blockers(c) \mapsto Muted(k) \lor scheduled[k]]$

$Running(c) \triangleq \exists\, k \in Cowns : running[k] \land c \in Head(queue[k])$

$AcquiredBy(a, b) \triangleq$
  $\land\, a < b$
  $\land\, \exists\, m \in Range(queue[b]) : (a \in m) \land (b \in m)$
$Acquired(c) \triangleq \exists\, k \in Cowns : AcquiredBy(c, k)$

$MutedBy(a, b) \triangleq$

$\land\ Muted(a)$
$\land\ \exists\, m \in Range(queue[b]) : (b \notin m) \land (a \in m)$

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

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

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

$Acquire(cown)\ \triangleq$
$\land\ scheduled[cown]$
$\land\ \neg running[cown]$
$\land\ \neg EmptyQueue(cown)$
$\land\ cown \in Head(queue[cown])$
$\land\ cown < Max(Head(queue[cown]))$
———
$\land$ UNCHANGED $\langle fuel,\ running,\ mutor\rangle$
Unschedule and forward the message to the next *cown*.
$\land$ LET
$msg\ \triangleq\ Head(queue[cown])$
$next\ \triangleq\ Min(\{c \in msg : c > cown\})$
IN
$\land\ queue' = Enqueue(next,\ msg)\ @@\ Dequeue(cown)\ @@\ queue$
$\land\ blocker' = (cown :> next)\ @@\ blocker$
Prioritize this *cown* and next if either are prioritized.
$\land$ IF $\exists\, c \in \{cown,\ next\} : Prioritized(c)$ THEN
Unblock next.
$\land\ priority' = (next :> 1)\ @@\ [c \in Blockers(next) \mapsto 1]\ @@\ priority$
$\land\ scheduled' = Unblock(next)\ @@\ (cown :> \text{FALSE})\ @@\ scheduled$

```
        ELSE
        ∧ UNCHANGED ⟨priority⟩
        ∧ scheduled' = (cown :> FALSE) @@ scheduled

Unmute(cown) ≜
  ∧ scheduled[cown]
  ∧ ¬running[cown]
  ∧ ¬EmptyQueue(cown)
  ∧ cown ∉ Head(queue[cown])
   ———
  ∧ UNCHANGED ⟨fuel, running, mutor, blocker⟩
   # Remove message from queue.
  ∧ queue' = Dequeue(cown) @@ queue
   # Reschedule muted cowns.
  ∧ LET muted ≜ {c ∈ Head(queue[cown]) : Muted(c)}IN
    ∧ priority' = [c ∈ muted ↦ 0] @@ priority
    ∧ scheduled' = [c ∈ muted ↦ TRUE] @@ scheduled

PreRun(cown) ≜
  ∧ scheduled[cown]
  ∧ ¬running[cown]
  ∧ ¬EmptyQueue(cown)
  ∧ cown = Max(Head(queue[cown]))
   ———
  ∧ UNCHANGED ⟨fuel, queue, scheduled, mutor, priority⟩
   # Set max cown in current message to running
  ∧ running' = (cown :> TRUE) @@ running
  ∧ blocker' = [c ∈ Head(queue[cown]) ↦ Null] @@ blocker

Send(cown) ≜
  ∧ running[cown]
  ∧ fuel > 0
   ———
  ∧ UNCHANGED ⟨running, blocker⟩
  ∧ fuel' = fuel − 1
  ∧ ∃ receivers ∈ Subsets(Cowns, 1, MaxMessageSize) :
    LET
      next ≜ Min(receivers)
      senders ≜ Head(queue[cown])
      mutors ≜ {c ∈ receivers : Overloaded(c)}
    IN
     # Place message for receivers in the first receiver's queue.
     ∧ queue' = Enqueue(next, receivers) @@ queue
     ∧ IF (∃ c ∈ receivers : Prioritized(c)) ∨ Overloaded(next) THEN
        # Unblock next.
       ∧ priority' = (next :> 1) @@ [c ∈ Blockers(next) ↦ 1] @@ priority
```

$\land\ scheduled' = Unblock(next)\ @@\ scheduled$
\# Set *mutor* if any receiver is overloaded and there are no receivers in the set of senders.
$\land$ IF
$\land\ mutors \neq \{\}$
$\land\ mutor[cown] = Null$
$\land\ (senders \cap receivers) = \{\}$
\# The priority of senders is checked before muting in *PostRun*.
THEN $mutor' = (cown :> Min(mutors))\ @@\ mutor$
ELSE UNCHANGED $\langle mutor \rangle$
ELSE
$\land$ UNCHANGED $\langle scheduled,\ mutor,\ priority \rangle$

$PostRun(cown)\ \triangleq$
$\land\ running[cown]$
———
$\land$ UNCHANGED $\langle fuel,\ blocker \rangle$
$\land\ running' = (cown :>$ FALSE$)\ @@\ running$
$\land\ mutor' = (cown :> Null)\ @@\ mutor$
$\land$ LET $msg \triangleq Head(queue[cown])$ IN
\# Mute if *mutor* is set and no running cowns are overloaded.
$\land$ IF
$\land\ mutor[cown] \neq Null$
$\land\ \forall\, c \in msg : \neg Overloaded(c) \land \neg Prioritized(c)$
THEN
$\land\ priority' = [c \in msg \mapsto -1]\ @@\ priority$
$\land\ scheduled' = [c \in msg \mapsto$ FALSE$]\ @@\ scheduled$
\# *Send* unmute message to *mutor*
$\land\ queue' = Enqueue(mutor[cown],\ msg)\ @@\ Dequeue(cown)\ @@\ queue$
ELSE
$\land$ UNCHANGED $\langle priority \rangle$
$\land\ scheduled' = [c \in msg \mapsto$ TRUE$]\ @@\ scheduled$
$\land\ queue' = Dequeue(cown)\ @@\ queue$

$RunStep(cown)\ \triangleq$
\# $\lor\ ExternalReceive(cown)$ * \# Very expensive check
$\lor\ Acquire(cown)$
$\lor\ Unmute(cown)$
$\lor\ PreRun(cown)$
$\lor\ Send(cown)$
$\lor\ PostRun(cown)$

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

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

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

Properties

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

Invariants

Ensure that the model produces finite messages.
$\mathit{MessageLimit} \triangleq \mathit{Cardinality}(\mathit{Messages}) \leq (\mathit{Cardinality}(\mathit{Cowns}) + \mathit{MaxMessageCount})$

Cowns are acquired by one running message at a time.
$\mathit{UniqueAcquisition} \triangleq$
LET $\mathit{msgs} \triangleq \mathit{Concat}(\{\langle \mathit{Head}(\mathit{queue}[c])\rangle : c \in \{k \in \mathit{Cowns} : \mathit{running}[k]\}\})$
IN $\mathit{Cardinality}(\mathit{Range}(\mathit{msgs})) = \mathit{Len}(\mathit{msgs})$

TODO: Token messages?
Each message has at least one *cown*.
$\mathit{NoTokens} \triangleq \forall\, c \in \mathit{Cowns} : \mathit{Len}(\mathit{SelectSeq}(\mathit{queue}[c], \text{LAMBDA } m : m = \{\})) = 0$

A running *cown* must be scheduled and be the *max cown* in the message at the head of its queue.
$\mathit{RunningImplication} \triangleq \forall\, c \in \mathit{Cowns} : \mathit{running}[c] \Rightarrow$
$\land\ \mathit{scheduled}[c]$
$\land\ c = \mathit{Max}(\mathit{Head}(\mathit{queue}[c]))$
$\land\ \forall\, k \in \mathit{Head}(\mathit{queue}[c]) : (k < c) \Rightarrow \mathit{AcquiredBy}(k,\ c)$

An acquired *cown* is not scheduled.
$\mathit{AcquiredImplication} \triangleq \forall\, c \in \mathit{Cowns} : \mathit{Acquired}(c) \Rightarrow$
$\land\ \neg\mathit{scheduled}[c]$

A muted *cown* is not scheduled or running.
$\mathit{MutedImplication} \triangleq \forall\, c \in \mathit{Cowns} : \mathit{Muted}(c) \equiv$
$\land\ \exists\, k \in \mathit{Cowns} : \mathit{MutedBy}(c,\ k)$
$\land\ \neg\mathit{scheduled}[c]$
$\land\ \neg\mathit{Running}(c)$

A muted *cown* exists in an unmute message in the queue of at least one *mutor*.
$\mathit{MutedInUnmuteMsg} \triangleq$
$\forall\, m \in \{c \in \mathit{Cowns} : \mathit{Muted}(c)\} :$
$\mathit{Cardinality}(\{c \in \mathit{Cowns} : \mathit{MutedBy}(m,\ c)\}) > 0$

A *cown* may be acqiured by at most one message.
$\mathit{AcquiredOnce} \triangleq$
$\forall\, a \in \{c \in \mathit{Cowns} : \mathit{Acquired}(c)\} :$
$\mathit{Cardinality}(\{c \in \mathit{Cowns} : \mathit{AcquiredBy}(a,\ c)\}) = 1$

An acquired *cown* is either acquired by a *cown* in its blocker set or it is running.
$\mathit{AcquiredByBlocker} \triangleq \forall\, \langle a,\ b\rangle \in \mathit{Cowns} \times \mathit{Cowns} :$

$AcquiredBy(a,\ b) \Rightarrow b \in Blockers(a) \lor Running(a)$

\# A prioritized *cown* is not acquired by a muted *cown*.
$PrioritizedNotAcquiredByMuted \triangleq \forall \langle o,\ m \rangle \in Cowns \times Cowns :$
$Prioritized(o) \land Muted(m) \Rightarrow \neg AcquiredBy(o,\ m)$

(*
* https://*github.com*/tlaplus/Examples/blob/master/specifications/*TransitiveClosure*/*TransitiveClosure.tla*#*L*114

```
TC(R) ≜
 LET
   S ≜ {r[1] : r ∈ R} ∪ {r[2] : r ∈ R}
   RECURSIVE TCR(_)
   TCR(T) ≜
    IF T = {} THEN R
     ELSE
      LET
        r ≜ CHOOSE s ∈ T: TRUE
        RR ≜ TCR(T \ {r})
      IN
        RR ∪ {⟨s, t⟩ ∈ S × S : ⟨s, r⟩ ∈ RR ∧ ⟨r, t⟩ ∈ RR}
 IN
   TCR(S)

CylcicTransitiveClosure(R(_, _)) ≜
 LET s ≜ {⟨a, b⟩ ∈ Cowns × Cowns : R(a, b)}
 IN  ∃ c ∈ Cowns: ⟨c, c⟩ ∈ TC(s)
```
*)