```
MODULE callback
EXTENDS TLC, Sequences, SequencesExt, FiniteSets, Integers

CONSTANTS Timers, DeltaRange, Servers, Clients, Subscribers

Tasks ≜ Subscribers ∪ Servers ∪ Clients

   --algorithm callback
variables
     list for timer
     example: ⟨[delta ↦ 3, name ↦ "timer1"], [delta ↦ 2, name ↦ "timer2"]⟩
    delta_list = SetToSeq({[delta ↦ random_num(0, DeltaRange), name ↦ x] : x ∈ Timers}) ;

     events
    wait_set = {} ;

     tasks
    running = {} ;
    waiting  = Tasks ;

define
    random_num(min, max) ≜ CHOOSE i ∈ min .. max : TRUE
    pick_task(set) ≜ CHOOSE x ∈ set : TRUE

    starvation_free ≜ ∀ x ∈ (Timers ∪ Tasks) :
        LET delta_set ≜ {y.name : y ∈ ToSet(delta_list)} IN
        (((x ∈ delta_set) ∨ (x ∈ wait_set)) ⇝ ◇(x ∈ running))
    running_xor_waiting ≜ ∀ x ∈ Tasks :
        (x ∈ running ∧ x ∉ waiting) ∨ (x ∉ running ∧ x ∈ waiting)
    running_then_not_delta_list ≜ ∀ x ∈ Timers :
        LET delta_set ≜ {y.name : y ∈ ToSet(delta_list)} IN
        x ∈ running ⇒ x ∉ delta_set
    type_check ≜
        LET delta_set ≜ {y.name : y ∈ ToSet(delta_list)} IN
         ∧ waiting ⊆ Tasks
         ∧ running ⊆ (Tasks ∪ Timers)
         ∧ delta_set ⊆ Timers
end define

 To emulate incrementing clock, decrement the delta of the head of the delta_list.
macro increment_clock()
begin
    if delta_list ≠ ⟨⟩ ∧ delta_list[1].delta > 0 then
        delta_list[1].delta := delta_list[1].delta − 1 ;
    end if ;
end macro ;

 execute a callback function
```

```
procedure $callback(name)$
begin
    $start\_callback$:
        $increment\_clock()$ ;
        $running := running \cup \{name\}$ ;
        $waiting := waiting \setminus \{name\}$ ;

    $end\_callback$:
        $running := running \setminus \{name\}$ ;
        if $name \in Tasks$ then
            $waiting := waiting \cup \{name\}$
        end if ;
        return ;
end procedure ;

 reenable timer with at random delay
procedure $reload\_timer(name)$
variables
    $idx$ ;
    $delta$ ;
begin
    $start\_reload\_timer$:
        $increment\_clock()$ ;

         choose insertion point
        $idx := random\_num(1, Len(delta\_list) + 1)$ ;
        if $idx \leq Len(delta\_list)$ then
             insert to middle
            $delta := random\_num(0, delta\_list[idx].delta)$ ;

            $reload\_insert1$:
                 update delta and insert
                $delta\_list[idx].delta := delta\_list[idx].delta - delta$ ;

            $reload\_insert2$:
                $delta\_list := InsertAt(delta\_list, idx, [delta \mapsto delta, name \mapsto name])$ ;
        else
             insert to the end
            $delta := random\_num(0, DeltaRange)$ ;

            $reload\_insert\_end$:
                $delta\_list := Append(delta\_list, [delta \mapsto delta, name \mapsto name])$ ;
            skip ;
        end if ;

    $end\_reload\_timer$:
        return ;
```

```
end procedure ;

 execute a task
procedure $execute\_task(runnable)$
variables
    $task$ ;
begin
    $start\_task$:
        while $runnable \neq \{\}$ do
            $task := pick\_task(runnable)$ ;
            $runnable := runnable \setminus \{task\}$ ;
            call $callback(task)$ ;
        end while ;

        return;
end procedure ;

fair process $trigger\_event \in Tasks$
begin
    $fire\_event$:
        while TRUE do
            $wait\_set := wait\_set \cup \{self\}$ ;
        end while ;
end process ;

fair + process $executor = \text{"executor"}$
variables
    $head$ ;
    $to\_be\_reloaded = \langle\rangle$ ;
begin
    $start\_executor$:
        while TRUE do
            $increment\_clock()$ ;

            $execute$:
                while $delta\_list \neq \langle\rangle \wedge delta\_list[1].delta = 0$ do
                     pop front
                    $head := Head(delta\_list)$ ;
                    $delta\_list := Tail(delta\_list)$ ;

                     call the callback function
                    call $callback(head.name)$ ;

                     reenable timer later
                    $save\_timer$:
                        $to\_be\_reloaded := Append(to\_be\_reloaded, head.name)$ ;
                end while ;
            reload:
                reenable timer
                while $to\_be\_reloaded \neq \langle\rangle$ do
                    call $reload\_timer(to\_be\_reloaded[1])$ ;

                    reload2:
                        $to\_be\_reloaded := Tail(to\_be\_reloaded)$ ;
                end while ;

            execute_tasks:
                pick $wait\_set$ tasks up
                with $tmp\_wait\_set = wait\_set$ do
                    $wait\_set := \{\}$ ;
                    call $execute\_task(tmp\_wait\_set)$ ;
                end with ;

        end while ;
end process ;
end algorithm ;
```

BEGIN TRANSLATION ($chksum(pcal) =$ "86a9cce3" $\land\ chksum(tla) =$ "1a4cbaba")

Parameter name of procedure *callback* at line 49 col 20 changed to *name_*

```
CONSTANT $defaultInitValue$
VARIABLES $delta\_list, wait\_set, running, waiting, pc, stack$

 define statement
$random\_num(min, max) \triangleq$ CHOOSE $i \in min\ ..\ max$ : TRUE
$pick\_task(set) \triangleq$ CHOOSE $x \in set$ : TRUE

$starvation\_free \triangleq \forall x \in (Timers \cup Tasks)$ :
    LET $delta\_set \triangleq \{y.name : y \in ToSet(delta\_list)\}$ IN
    $(((x \in delta\_set) \lor (x \in wait\_set)) \leadsto \Diamond(x \in running))$
$running\_xor\_waiting \triangleq \forall x \in Tasks$ :
    $(x \in running \land x \notin waiting) \lor (x \notin running \land x \in waiting)$
$running\_then\_not\_delta\_list \triangleq \forall x \in Timers$ :
    LET $delta\_set \triangleq \{y.name : y \in ToSet(delta\_list)\}$ IN
    $x \in running \Rightarrow x \notin delta\_set$
$type\_check \triangleq$
    LET $delta\_set \triangleq \{y.name : y \in ToSet(delta\_list)\}$ IN
    $\land\ waiting \subseteq Tasks$
    $\land\ running \subseteq (Tasks \cup Timers)$
    $\land\ delta\_set \subseteq Timers$

VARIABLES $name\_, name, idx, delta, runnable, task, head, to\_be\_reloaded$

$vars \triangleq \langle delta\_list, wait\_set, running, waiting, pc, stack, name\_, name,$
          $idx, delta, runnable, task, head, to\_be\_reloaded\rangle$
```

```
ProcSet ≜ (Tasks) ∪ {"executor"}

Init ≜  Global variables
        ∧ delta_list = SetToSeq({[delta ↦ random_num(0, DeltaRange), name ↦ x] : x ∈ Timers})
        ∧ wait_set = {}
        ∧ running = {}
        ∧ waiting  = Tasks
         Procedure callback
        ∧ name_ = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure reload_timer
        ∧ name = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ idx = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ delta = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure execute_task
        ∧ runnable = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ task = [self ∈ ProcSet ↦ defaultInitValue]
         Process executor
        ∧ head = defaultInitValue
        ∧ to_be_reloaded = ⟨⟩
        ∧ stack = [self ∈ ProcSet ↦ ⟨⟩]
        ∧ pc = [self ∈ ProcSet ↦ CASE self ∈ Tasks → "fire_event"
                                  □   self = "executor" → "start_executor"]

start_callback(self) ≜ ∧ pc[self] = "start_callback"
                       ∧ IF delta_list ≠ ⟨⟩ ∧ delta_list[1].delta > 0
                            THEN ∧ delta_list' = [delta_list EXCEPT ![1].delta = delta_list[1].delta − 1]
                            ELSE ∧ TRUE
                                 ∧ UNCHANGED delta_list
                       ∧ running' = (running ∪ {name_[self]})
                       ∧ waiting'  = waiting \ {name_[self]}
                       ∧ pc' = [pc EXCEPT ![self] = "end_callback"]
                       ∧ UNCHANGED ⟨wait_set, stack, name_, name, idx,
                                    delta, runnable, task, head,
                                    to_be_reloaded⟩

end_callback(self) ≜ ∧ pc[self]   = "end_callback"
                     ∧ running' = running \ {name_[self]}
                     ∧ IF name_[self] ∈ Tasks
                          THEN ∧ waiting' = (waiting ∪  {name_[self]})
                          ELSE ∧ TRUE
                               ∧ UNCHANGED waiting
                     ∧ pc' = [pc EXCEPT ![self] = Head(stack[self]).pc]
                     ∧ name_' = [name_ EXCEPT ![self] = Head(stack[self]).name_]
                     ∧ stack'   = [stack EXCEPT ![self]   = Tail(stack[self])]
                     ∧ UNCHANGED ⟨delta_list, wait_set, name, idx, delta,
                                  runnable, task, head, to_be_reloaded⟩
```

```
$callback(self) \triangleq start\_callback(self) \lor end\_callback(self)$

$start\_reload\_timer(self) \triangleq$ $\land\ pc[self] = $ "start_reload_timer"
    $\land$ IF $delta\_list \neq \langle\rangle \land delta\_list[1].delta > 0$
        THEN $\land\ delta\_list' = [delta\_list$ EXCEPT $![1].delta = delta\_list[1].delta$ -
        ELSE $\land$ TRUE
             $\land$ UNCHANGED $delta\_list$
    $\land\ idx' = [idx$ EXCEPT $![self] = random\_num(1,\ Len(delta\_list') + 1)]$
    $\land$ IF $idx'[self] \leq Len(delta\_list')$
        THEN $\land\ delta' = [delta$ EXCEPT $![self] = random\_num(0,\ delta\_list'[id$
             $\land\ pc' = [pc$ EXCEPT $![self] = $ "reload_insert1"$]$
        ELSE $\land\ delta' = [delta$ EXCEPT $![self] = random\_num(0,\ DeltaRange)]$
             $\land\ pc' = [pc$ EXCEPT $![self] = $ "reload_insert_end"$]$
    $\land$ UNCHANGED $\langle wait\_set,\ running,\ waiting,\ stack,$
                   $name\_,\ name,\ runnable,\ task,\ head,$
                   $to\_be\_reloaded\rangle$

$reload\_insert1(self) \triangleq$ $\land\ pc[self] = $ "reload_insert1"
    $\land\ delta\_list' = [delta\_list$ EXCEPT $![idx[self]].delta = delta\_list[idx[self]].delta -$
    $\land\ pc' = [pc$ EXCEPT $![self] = $ "reload_insert2"$]$
    $\land$ UNCHANGED $\langle wait\_set,\ running,\ waiting,\ stack,$
                   $name\_,\ name,\ idx,\ delta,\ runnable,$
                   $task,\ head,\ to\_be\_reloaded\rangle$

$reload\_insert2(self) \triangleq$ $\land\ pc[self] = $ "reload_insert2"
    $\land\ delta\_list' = InsertAt(delta\_list,\ idx[self],\ [delta \mapsto delta[self],\ name \mapsto name[s$
    $\land\ pc' = [pc$ EXCEPT $![self] = $ "end_reload_timer"$]$
    $\land$ UNCHANGED $\langle wait\_set,\ running,\ waiting,\ stack,$
                   $name\_,\ name,\ idx,\ delta,\ runnable,$
                   $task,\ head,\ to\_be\_reloaded\rangle$

$reload\_insert\_end(self) \triangleq$ $\land\ pc[self] = $ "reload_insert_end"
    $\land\ delta\_list' = Append(delta\_list,\ [delta \mapsto delta[self],\ name \mapsto name[self]])$
    $\land$ TRUE
    $\land\ pc' = [pc$ EXCEPT $![self] = $ "end_reload_timer"$]$
    $\land$ UNCHANGED $\langle wait\_set,\ running,\ waiting,\ stack,$
                   $name\_,\ name,\ idx,\ delta,\ runnable,$
                   $task,\ head,\ to\_be\_reloaded\rangle$

$end\_reload\_timer(self) \triangleq$ $\land\ pc[self] = $ "end_reload_timer"
    $\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
    $\land\ idx' = [idx$ EXCEPT $![self] = Head(stack[self]).idx]$
    $\land\ delta' = [delta$ EXCEPT $![self] = Head(stack[self]).delta]$
    $\land\ name' = [name$ EXCEPT $![self] = Head(stack[self]).name]$
    $\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
    $\land$ UNCHANGED $\langle delta\_list,\ wait\_set,\ running,$
```

```
                                        $waiting, name\_, runnable, task, head,$
                                        $to\_be\_reloaded \rangle$

$reload\_timer(self) \triangleq start\_reload\_timer(self) \lor reload\_insert1(self)$
                  $\lor reload\_insert2(self) \lor reload\_insert\_end(self)$
                  $\lor end\_reload\_timer(self)$

$start\_task(self) \triangleq \land pc[self] = \text{“start\_task”}$
               $\land \textsc{if}\ runnable[self] \neq \{\}$
                   $\textsc{then}\ \land task' = [task\ \textsc{except}\ ![self] = pick\_task(runnable[self])]$
                         $\land runnable' = [runnable\ \textsc{except}\ ![self] = runnable[self] \setminus \{task'[self]\}]$
                         $\land \land name\_' = [name\_\ \textsc{except}\ ![self] = task'[self]]$
                           $\land stack' = [stack\ \textsc{except}\ ![self] = \langle [procedure \mapsto \text{“callback”},$
                                                                    $pc \mapsto \text{“start\_task”},$
                                                                    $name\_ \mapsto name\_[self]] \rangle$
                                                                    $\circ\ stack[self]]$
                         $\land pc' = [pc\ \textsc{except}\ ![self] = \text{“start\_callback”}]$
                   $\textsc{else}\ \land pc' = [pc\ \textsc{except}\ ![self] = Head(stack[self]).pc]$
                         $\land task' = [task\ \textsc{except}\ ![self] = Head(stack[self]).task]$
                         $\land runnable' = [runnable\ \textsc{except}\ ![self] = Head(stack[self]).runnable]$
                         $\land stack' = [stack\ \textsc{except}\ ![self] = Tail(stack[self])]$
                         $\land name\_' = name\_$
               $\land \textsc{unchanged}\ \langle delta\_list, wait\_set, running, waiting,$
                              $name, idx, delta, head, to\_be\_reloaded \rangle$

$execute\_task(self) \triangleq start\_task(self)$

$fire\_event(self) \triangleq \land pc[self] = \text{“fire\_event”}$
               $\land wait\_set' = (wait\_set \cup \{self\})$
               $\land pc' = [pc\ \textsc{except}\ ![self] = \text{“fire\_event”}]$
               $\land \textsc{unchanged}\ \langle delta\_list, running, waiting, stack, name\_,$
                              $name, idx, delta, runnable, task, head,$
                              $to\_be\_reloaded \rangle$

$trigger\_event(self) \triangleq fire\_event(self)$

$start\_executor \triangleq \land pc[\text{“executor”}] = \text{“start\_executor”}$
              $\land \textsc{if}\ delta\_list \neq \langle \rangle \land delta\_list[1].delta > 0$
                  $\textsc{then}\ \land delta\_list' = [delta\_list\ \textsc{except}\ ![1].delta = delta\_list[1].delta - 1]$
                  $\textsc{else}\ \land \text{TRUE}$
                        $\land \textsc{unchanged}\ delta\_list$
              $\land pc' = [pc\ \textsc{except}\ ![\text{“executor”}] = \text{“execute”}]$
              $\land \textsc{unchanged}\ \langle wait\_set, running, waiting, stack, name\_,$
                             $name, idx, delta, runnable, task, head,$
                             $to\_be\_reloaded \rangle$

$execute \triangleq \land pc[\text{“executor”}] = \text{“execute”}$
```

```
        ∧ IF delta_list ≠ ⟨⟩ ∧ delta_list[1].delta = 0
             THEN ∧ head' = Head(delta_list)
                  ∧ delta_list' = Tail(delta_list)
                  ∧ ∧ name_' = [name_ EXCEPT ![“executor”] = head'.name]
                    ∧ stack'  = [stack EXCEPT ![“executor”]  = ⟨[procedure ↦  “callback”,
                                                                 pc        ↦  “save_timer”,
                                                                 name_     ↦  name_[“executor”]]
                                                                 ∘ stack[“executor”]]
                  ∧ pc' = [pc EXCEPT ![“executor”] = “start_callback”]
             ELSE ∧ pc' = [pc EXCEPT ![“executor”] = “reload”]
                  ∧ UNCHANGED ⟨delta_list, stack, name_, head⟩
        ∧ UNCHANGED ⟨wait_set, running, waiting, name, idx, delta,
                     runnable, task, to_be_reloaded⟩

save_timer ≜ ∧ pc[“executor”] = “save_timer”
             ∧ to_be_reloaded' = Append(to_be_reloaded, head.name)
             ∧ pc' = [pc EXCEPT ![“executor”] = “execute”]
             ∧ UNCHANGED ⟨delta_list, wait_set, running, waiting, stack,
                          name_, name, idx, delta, runnable, task, head⟩

reload ≜ ∧ pc[“executor”] = “reload”
         ∧ IF to_be_reloaded ≠ ⟨⟩
              THEN ∧ ∧ name' = [name EXCEPT ![“executor”] = to_be_reloaded[1]]
                     ∧ stack' = [stack EXCEPT ![“executor”] = ⟨[procedure ↦  “reload_timer”,
                                                               pc        ↦  “reload2”,
                                                               idx       ↦  idx[“executor”],
                                                               delta     ↦  delta[“executor”],
                                                               name      ↦  name[“executor”]]⟩
                                                               ∘ stack[“executor”]]
                   ∧ idx' = [idx EXCEPT ![“executor”] = defaultInitValue]
                   ∧ delta' = [delta EXCEPT ![“executor”] = defaultInitValue]
                   ∧ pc' = [pc EXCEPT ![“executor”] = “start_reload_timer”]
              ELSE ∧ pc' = [pc EXCEPT ![“executor”] = “execute_tasks”]
                   ∧ UNCHANGED ⟨stack, name, idx, delta⟩
         ∧ UNCHANGED ⟨delta_list, wait_set, running, waiting, name_,
                      runnable, task, head, to_be_reloaded⟩

reload2 ≜ ∧ pc[“executor”] = “reload2”
          ∧ to_be_reloaded' = Tail(to_be_reloaded)
          ∧ pc' = [pc EXCEPT ![“executor”] = “reload”]
          ∧ UNCHANGED ⟨delta_list, wait_set, running, waiting, stack,
                       name_, name, idx, delta, runnable, task, head⟩

execute_tasks ≜ ∧ pc[“executor”] = “execute_tasks”
                ∧ LET tmp_wait_set ≜ wait_setIN
                     ∧ wait_set' = {}
        $\land\ \land\ runnable' = [runnable\ \text{EXCEPT}\ ![\text{“executor”}] = tmp\_wait\_set]$
           $\land\ stack' = [stack\ \text{EXCEPT}\ ![\text{“executor”}] = \langle[procedure \mapsto \text{“execute\_task”},$
                                                              $pc \mapsto \text{“start\_executor”},$
                                                              $task \mapsto task[\text{“executor”}],$
                                                              $runnable \mapsto runnable[\text{“executor”}]]\rangle$
                                                              $\circ\ stack[\text{“executor”}]]$
        $\land\ task' = [task\ \text{EXCEPT}\ ![\text{“executor”}] = defaultInitValue]$
        $\land\ pc' = [pc\ \text{EXCEPT}\ ![\text{“executor”}] = \text{“start\_task”}]$
     $\land\ \text{UNCHANGED}\ \langle delta\_list, running, waiting, name\_, name,$
                          $idx, delta, head, to\_be\_reloaded\rangle$

$executor \triangleq start\_executor \lor execute \lor save\_timer \lor reload \lor reload2$
              $\lor\ execute\_tasks$

$Next \triangleq executor$
           $\lor\ (\exists\, self \in ProcSet : \lor\ callback(self) \lor reload\_timer(self)$
                                    $\lor\ execute\_task(self))$
           $\lor\ (\exists\, self \in Tasks : trigger\_event(self))$

$Spec \triangleq \land\ Init \land \Box[Next]_{vars}$
         $\land\ \forall\, self \in Tasks : \text{WF}_{vars}(trigger\_event(self))$
         $\land\ \land\ \text{SF}_{vars}(executor)$
            $\land\ \text{SF}_{vars}(callback(\text{“executor”}))$
            $\land\ \text{SF}_{vars}(reload\_timer(\text{“executor”}))$
            $\land\ \text{SF}_{vars}(execute\_task(\text{“executor”}))$

 END TRANSLATION
```