MODULE *scheduler*

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

CONSTANTS *Subscribers*, *Timers*, *Workers*, *LoopTimer*, *LoopSubscriber*

$AllTask \triangleq Subscribers \cup Subscribers$

--algorithm *scheduler*
variables
events
$wait\_set = \{\}$;

states of tasks
$run\_queue = \langle\rangle$;
$running = \{\}$;
$waiting = AllTask$;

$finish\_subscriber =$ FALSE ;
$finish\_timer =$ FALSE ;
$is\_finish\_sched =$ FALSE ;

define
$is\_finish\_event \triangleq finish\_subscriber \wedge finish\_timer$
$starvation\_free \triangleq \forall\, event \in AllTask : event \in wait\_set \rightsquigarrow \Diamond(event \in running)$
end define

fair + **process** *scheduler* = "scheduler"
begin
start_sched:
while TRUE **do**
await $wait\_set \neq \{\} \vee is\_finish\_event$;

if $wait\_set \neq \{\}$ **then**
pick runnable tasks and change the states to *run_queue* from waiting
with $tasks = waiting \cap wait\_set$,
$timers = tasks \cap Timers$,
$subscribers = tasks \cap Subscribers$ **do**
push to *run_queue*
$run\_queue := run\_queue \circ SetToSeq(timers) \circ SetToSeq(subscribers)$;

change state
$waiting := (waiting \setminus timers) \setminus subscribers$;
end with ;
else
goto *end_sched* ;
end if ;
end while ;
end_sched:

```
        is_finish_sched := TRUE ;
end process ;

fair + process trigger_subscriber ∈ Subscribers
variables
    cnt = 0 ;
begin
    start_subscriber:
        while cnt < LoopSubscriber do
            wait_set := wait_set ∪ {self} ;
            cnt := cnt + 1 ;
        end while ;
    end_subscriber:
        finish_subscriber := TRUE ;
end process ;

fair + process trigger_timer ∈ Timers
variables
    cnt = 0 ;
begin
    start_timer:
        while cnt < LoopTimer do
            wait_set := wait_set ∪ {self} ;
            cnt := cnt + 1 ;
        end while ;
    end_timer:
        finish_timer := TRUE ;
end process ;

 worker thread
fair + process worker ∈ Workers
variables
    task ;
begin
     work-stealing
    start_worker:
        while TRUE do
            await run_queue ≠ ⟨⟩ ∨ is_finish_sched ;

            if run_queue = ⟨⟩ then
                goto end_worker ;
             else
                task := Head(run_queue) ;
                run_queue := Tail(run_queue) ;
                running := running ∪ {task} ;
            end if ;
```

```
            finish_task:
                running := running \ {task} ;
                waiting  := waiting ∪ {task} ;
        end while ;
    end_worker:
        skip ;
end process ;

end algorithm ;
```

BEGIN TRANSLATION ($chksum(pcal) =$ "$993d6fff$" $\land\ chksum(tla) =$ "$e3181fcf$")

Process variable *cnt* of process *trigger_subscriber* at line 54 col 5 changed to *cnt_*

CONSTANT *defaultInitValue*

VARIABLES *wait_set*, *run_queue*, *running*, *waiting*, *finish_subscriber*, *finish_timer*, *is_finish_sched*, *pc*

define statement

$is\_finish\_event \triangleq finish\_subscriber \land finish\_timer$

$starvation\_free \triangleq \forall\, event \in AllTask : event \in wait\_set \rightsquigarrow \Diamond(event \in running)$

VARIABLES *cnt_*, *cnt*, *task*

$vars \triangleq \langle wait\_set,\ run\_queue,\ running,\ waiting,\ finish\_subscriber,\ finish\_timer,\ is\_finish\_sched,\ pc,\ cnt\_,\ cnt,\ task\rangle$

$ProcSet \triangleq \{\text{"scheduler"}\} \cup (Subscribers) \cup (Timers) \cup (Workers)$

$Init \triangleq$ Global variables

$\land\ wait\_set = \{\}$

$\land\ run\_queue = \langle\rangle$

$\land\ running = \{\}$

$\land\ waiting\ = AllTask$

$\land\ finish\_subscriber = \text{FALSE}$

$\land\ finish\_timer = \text{FALSE}$

$\land\ is\_finish\_sched = \text{FALSE}$

Process *trigger_subscriber*

$\land\ cnt\_ = [self \in Subscribers \mapsto 0]$

Process *trigger_timer*

$\land\ cnt = [self \in Timers \mapsto 0]$

Process worker

$\land\ task = [self \in Workers \mapsto defaultInitValue]$

$\land\ pc = [self \in ProcSet \mapsto \text{CASE } self = \text{"scheduler"} \to \text{"start\_sched"}$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \square\ \ self \in Subscribers \to \text{"start\_subscriber"}$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \square\ \ self \in Timers \to \text{"start\_timer"}$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \square\ \ self \in Workers \to \text{"start\_worker"}]$

$start\_sched \triangleq\ \land\ pc[\text{"scheduler"}] = \text{"start\_sched"}$

$\qquad\qquad\land\ wait\_set \neq \{\} \lor is\_finish\_event$

$\land$ IF $wait\_set \neq \{\}$
THEN $\land$ LET $tasks \triangleq waiting \cap wait\_set$ IN
LET $timers \triangleq tasks \cap Timers$ IN
LET $subscribers \triangleq tasks \cap Subscribers$ IN
$\land\ run\_queue' = run\_queue \circ SetToSeq(timers) \circ SetToSeq(subscribers$
$\land\ waiting' = (waiting \setminus timers) \setminus subscribers$
$\land\ pc' = [pc$ EXCEPT $![\text{“scheduler”}] = \text{“start\_sched”}]$
ELSE $\land\ pc' = [pc$ EXCEPT $![\text{“scheduler”}] = \text{“end\_sched”}]$
$\land$ UNCHANGED $\langle run\_queue, waiting\rangle$
$\land$ UNCHANGED $\langle wait\_set, running, finish\_subscriber,$
$finish\_timer, is\_finish\_sched, cnt\_, cnt, task\rangle$

$end\_sched \triangleq \land\ pc[\text{“scheduler”}] = \text{“end\_sched”}$
$\land\ is\_finish\_sched' = \text{TRUE}$
$\land\ pc' = [pc$ EXCEPT $![\text{“scheduler”}] = \text{“Done”}]$
$\land$ UNCHANGED $\langle wait\_set, run\_queue, running, waiting,$
$finish\_subscriber, finish\_timer, cnt\_, cnt, task\rangle$

$scheduler \triangleq start\_sched \lor end\_sched$

$start\_subscriber(self) \triangleq \land\ pc[self] = \text{“start\_subscriber”}$
$\land$ IF $cnt\_[self] < LoopSubscriber$
THEN $\land\ wait\_set' = (wait\_set \cup \{self\})$
$\land\ cnt\_' = [cnt\_$ EXCEPT $![self] = cnt\_[self] + 1]$
$\land\ pc' = [pc$ EXCEPT $![self] = \text{“start\_subscriber”}]$
ELSE $\land\ pc' = [pc$ EXCEPT $![self] = \text{“end\_subscriber”}]$
$\land$ UNCHANGED $\langle wait\_set, cnt\_\rangle$
$\land$ UNCHANGED $\langle run\_queue, running, waiting,$
$finish\_subscriber, finish\_timer,$
$is\_finish\_sched, cnt, task\rangle$

$end\_subscriber(self) \triangleq \land\ pc[self] = \text{“end\_subscriber”}$
$\land\ finish\_subscriber' = \text{TRUE}$
$\land\ pc' = [pc$ EXCEPT $![self] = \text{“Done”}]$
$\land$ UNCHANGED $\langle wait\_set, run\_queue, running, waiting,$
$finish\_timer, is\_finish\_sched, cnt\_,$
$cnt, task\rangle$

$trigger\_subscriber(self) \triangleq start\_subscriber(self) \lor end\_subscriber(self)$

$start\_timer(self) \triangleq \land\ pc[self] = \text{“start\_timer”}$
$\land$ IF $cnt[self] < LoopTimer$
THEN $\land\ wait\_set' = (wait\_set \cup \{self\})$
$\land\ cnt' = [cnt$ EXCEPT $![self] = cnt[self] + 1]$
$\land\ pc' = [pc$ EXCEPT $![self] = \text{“start\_timer”}]$
ELSE $\land\ pc' = [pc$ EXCEPT $![self] = \text{“end\_timer”}]$
$\land$ UNCHANGED $\langle wait\_set, cnt\rangle$

$\land$ UNCHANGED $\langle run\_queue,\ running,\ waiting,$
$finish\_subscriber,\ finish\_timer,$
$is\_finish\_sched,\ cnt\_,\ task\rangle$

$end\_timer(self) \triangleq$
$\land\ pc[self] = \text{“end\_timer”}$
$\land\ finish\_timer' = \text{TRUE}$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“Done”}]$
$\land$ UNCHANGED $\langle wait\_set,\ run\_queue,\ running,\ waiting,$
$finish\_subscriber,\ is\_finish\_sched,\ cnt\_,$
$cnt,\ task\rangle$

$trigger\_timer(self) \triangleq start\_timer(self) \lor end\_timer(self)$

$start\_worker(self) \triangleq$
$\land\ pc[self] = \text{“start\_worker”}$
$\land\ run\_queue \neq \langle\rangle \lor is\_finish\_sched$
$\land$ IF $run\_queue = \langle\rangle$
THEN $\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“end\_worker”}]$
$\land$ UNCHANGED $\langle run\_queue,\ running,\ task\rangle$
ELSE $\land\ task' = [task \text{ EXCEPT } ![self] = Head(run\_queue)]$
$\land\ run\_queue' = Tail(run\_queue)$
$\land\ running' = (running \cup \{task'[self]\})$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“finish\_task”}]$
$\land$ UNCHANGED $\langle wait\_set,\ waiting,\ finish\_subscriber,$
$finish\_timer,\ is\_finish\_sched,\ cnt\_,\ cnt\rangle$

$finish\_task(self) \triangleq$
$\land\ pc[self] = \text{“finish\_task”}$
$\land\ running' = running \setminus \{task[self]\}$
$\land\ waiting' = (waiting \cup \{task[self]\})$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“start\_worker”}]$
$\land$ UNCHANGED $\langle wait\_set,\ run\_queue,\ finish\_subscriber,$
$finish\_timer,\ is\_finish\_sched,\ cnt\_,\ cnt,$
$task\rangle$

$end\_worker(self) \triangleq$
$\land\ pc[self] = \text{“end\_worker”}$
$\land$ TRUE
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“Done”}]$
$\land$ UNCHANGED $\langle wait\_set,\ run\_queue,\ running,\ waiting,$
$finish\_subscriber,\ finish\_timer,$
$is\_finish\_sched,\ cnt\_,\ cnt,\ task\rangle$

$worker(self) \triangleq start\_worker(self) \lor finish\_task(self) \lor end\_worker(self)$

Allow infinite stuttering to prevent deadlock on termination.
$Terminating \triangleq$
$\land\ \forall\, self \in ProcSet : pc[self] = \text{“Done”}$
$\land$ UNCHANGED $vars$

$Next \triangleq scheduler$

$$
\begin{array}{l}
\quad \lor (\exists\, self \in Subscribers : trigger\_subscriber(self)) \\
\quad \lor (\exists\, self \in Timers : trigger\_timer(self)) \\
\quad \lor (\exists\, self \in Workers : worker(self)) \\
\quad \lor Terminating
\end{array}
$$

$$
\begin{array}{lll}
Spec & \triangleq & \land\, Init \land \Box[Next]_{vars} \\
 & & \land\, \mathrm{SF}_{vars}(scheduler) \\
 & & \land\, \forall\, self \in Subscribers : \mathrm{SF}_{vars}(trigger\_subscriber(self)) \\
 & & \land\, \forall\, self \in Timers : \mathrm{SF}_{vars}(trigger\_timer(self)) \\
 & & \land\, \forall\, self \in Workers : \mathrm{SF}_{vars}(worker(self))
\end{array}
$$

$$
Termination \triangleq \Diamond(\forall\, self \in ProcSet : pc[self] = \text{“Done”})
$$

END TRANSLATION