MODULE *scheduler*

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

CONSTANTS *Subscribers*, *Servers*, *Clients*, *Workers*

$AllTask \triangleq Subscribers \cup Servers \cup Clients$

```
--algorithm scheduler
variables
    events
  wait_set = {} ;

    states of tasks
  run_queue = ⟨⟩ ;
  running = {} ;
  waiting  = AllTask ;

define
  starvation_free ≜ ∀ event ∈ AllTask : event ∈ wait_set ⇝ ◇(event ∈ running)
end define

fair  + process scheduler = "scheduler"
begin
  start_sched:
    while TRUE do
          await wait_set ≠ {} ;

            pick runnable tasks and change the states to run_queue from waiting
          with tasks = waiting ∩ wait_set,
               servers = tasks ∩ Servers,
               clients  = tasks ∩ Clients,
               subscribers = tasks ∩ Subscribers do
                 push to run_queue
              run_queue :=
                  run_queue ∘
                  SetToSeq(subscribers) ∘
                  SetToSeq(servers) ∘
                  SetToSeq(clients) ;

                  change state
              waiting := ((waiting \ subscribers) \ servers) \ clients ;
          end with ;
    end while ;
end process ;

fair process trigger_subscriber ∈ Subscribers
begin
  start_subscriber:
```

```
        while TRUE do
            wait_set := wait_set ∪ {self} ;
        end while ;
end process ;

fair process trigger_server ∈ Servers
begin
    start_server:
        while TRUE do
            wait_set := wait_set ∪ {self} ;
        end while ;
end process ;

fair process trigger_client ∈ Clients
begin
    start_client:
        while TRUE do
            wait_set := wait_set ∪ {self} ;
        end while ;
end process ;

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

            task := Head(run_queue) ;
            run_queue := Tail(run_queue) ;
            running := running ∪ {task} ;

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

end algorithm  ;
 BEGIN TRANSLATION (chksum(pcal) = "e7be2e0c" ∧ chksum(tla) = "373ff876")
CONSTANT defaultInitValue
VARIABLES wait_set, run_queue, running, waiting, pc

 define statement
```

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

VARIABLE $task$

$vars \triangleq \langle wait\_set,\ run\_queue,\ running,\ waiting,\ pc,\ task\rangle$

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

$Init \triangleq$ Global variables
$\land\ wait\_set = \{\}$
$\land\ run\_queue = \langle\rangle$
$\land\ running = \{\}$
$\land\ waiting\ = AllTask$
Process worker
$\land\ task = [self \in Workers \mapsto defaultInitValue]$
$\land\ pc = [self \in ProcSet \mapsto$ CASE $self = \text{"scheduler"} \rightarrow \text{"start\_sched"}$
$\square\ self \in Subscribers \rightarrow \text{"start\_subscriber"}$
$\square\ self \in Servers\ \rightarrow \text{"start\_server"}$
$\square\ self \in Clients\ \rightarrow \text{"start\_client"}$
$\square\ self \in Workers \rightarrow \text{"start\_worker"}]$

$start\_sched \triangleq\ \land\ pc[\text{"scheduler"}] = \text{"start\_sched"}$
$\land\ wait\_set \neq \{\}$
$\land$ LET $tasks \triangleq waiting \cap wait\_set$ IN
LET $servers \triangleq tasks \cap Servers$ IN
LET $clients \triangleq tasks \cap Clients$ IN
LET $subscribers \triangleq tasks \cap Subscribers$ IN
$\land\ run\_queue' = run\_queue \circ$
$SetToSeq(subscribers) \circ$
$SetToSeq(servers) \circ$
$SetToSeq(clients)$
$\land\ waiting' = ((waiting \setminus subscribers) \setminus servers) \setminus clients$
$\land\ pc' = [pc$ EXCEPT $![\text{"scheduler"}] = \text{"start\_sched"}]$
$\land$ UNCHANGED $\langle wait\_set,\ running,\ task\rangle$

$scheduler \triangleq start\_sched$

$start\_subscriber(self) \triangleq\ \land\ pc[self] = \text{"start\_subscriber"}$
$\land\ wait\_set' = (wait\_set \cup \{self\})$
$\land\ pc' = [pc$ EXCEPT $![self] = \text{"start\_subscriber"}]$
$\land$ UNCHANGED $\langle run\_queue,\ running,\ waiting,\ task\rangle$

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

$start\_server(self) \triangleq\ \land\ pc[self] = \text{"start\_server"}$
$\land\ wait\_set' = (wait\_set \cup \{self\})$
$\land\ pc' = [pc$ EXCEPT $![self] = \text{"start\_server"}]$
$\land$ UNCHANGED $\langle run\_queue,\ running,\ waiting,\ task\rangle$

$$trigger\_server(self) \triangleq start\_server(self)$$

$$\begin{aligned} start\_client(self) \triangleq\ & \land pc[self] = \text{“start\_client”} \\ & \land wait\_set' = (wait\_set \cup \{self\}) \\ & \land pc' = [pc \text{ EXCEPT } ![self] = \text{“start\_client”}] \\ & \land \text{UNCHANGED } \langle run\_queue, running, waiting, task\rangle \end{aligned}$$

$$trigger\_client(self) \triangleq start\_client(self)$$

$$\begin{aligned} start\_worker(self) \triangleq\ & \land pc[self] = \text{“start\_worker”} \\ & \land run\_queue \neq \langle\rangle \\ & \land task' = [task \text{ EXCEPT } ![self] = Head(run\_queue)] \\ & \land run\_queue' = Tail(run\_queue) \\ & \land running' = (running \cup \{task'[self]\}) \\ & \land pc' = [pc \text{ EXCEPT } ![self] = \text{“finish\_task”}] \\ & \land \text{UNCHANGED } \langle wait\_set, waiting\rangle \end{aligned}$$

$$\begin{aligned} finish\_task(self) \triangleq\ & \land pc[self] = \text{“finish\_task”} \\ & \land running' = running \setminus \{task[self]\} \\ & \land waiting' = (waiting \cup \{task[self]\}) \\ & \land pc' = [pc \text{ EXCEPT } ![self] = \text{“start\_worker”}] \\ & \land \text{UNCHANGED } \langle wait\_set, run\_queue, task\rangle \end{aligned}$$

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

$$\begin{aligned} Next \triangleq\ & scheduler \\ & \lor (\exists\, self \in Subscribers : trigger\_subscriber(self)) \\ & \lor (\exists\, self \in Servers : trigger\_server(self)) \\ & \lor (\exists\, self \in Clients : trigger\_client(self)) \\ & \lor (\exists\, self \in Workers : worker(self)) \end{aligned}$$

$$\begin{aligned} Spec \triangleq\ & \land Init \land \Box[Next]_{vars} \\ & \land \mathrm{SF}_{vars}(scheduler) \\ & \land \forall\, self \in Subscribers : \mathrm{WF}_{vars}(trigger\_subscriber(self)) \\ & \land \forall\, self \in Servers : \mathrm{WF}_{vars}(trigger\_server(self)) \\ & \land \forall\, self \in Clients : \mathrm{WF}_{vars}(trigger\_client(self)) \\ & \land \forall\, self \in Workers : \mathrm{SF}_{vars}(worker(self)) \end{aligned}$$

END TRANSLATION