```
MODULE scheduler
EXTENDS $TLC$, $Sequences$, $SequencesExt$, $FiniteSets$

CONSTANTS $Subscribers$, $Servers$, $Clients$, $Workers$

$AllTask \triangleq Subscribers \cup Subscribers$

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

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

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

fair + process $scheduler =$ "scheduler"
begin
   $start\_sched$:
      while TRUE do
            await $wait\_set \neq \{\}$ ;

             pick runnable tasks and change the states to $run\_queue$ from waiting
            with $tasks = waiting \cap wait\_set$,
                 $servers = tasks \cap Servers$,
                 $clients = tasks \cap Clients$,
                 $subscribers = tasks \cap Subscribers$ do
                   push to $run\_queue$
                  $run\_queue := run\_queue \circ SetToSeq(subscribers) \circ SetToSeq(servers) \circ SetToSeq($

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

fair process $trigger\_subscriber \in Subscribers$
begin
   $start\_subscriber$:
      while TRUE do
         $wait\_set := wait\_set \cup \{self\}$ ;
      end while ;
end process ;
```

```
fair process $trigger\_server \in Servers$
begin
    $start\_server$:
        while TRUE do
            $wait\_set := wait\_set \cup \{self\}$ ;
        end while ;
end process ;

fair process $trigger\_client \in Clients$
begin
    $start\_client$:
        while TRUE do
            $wait\_set := wait\_set \cup \{self\}$ ;
        end while ;
end process ;

 worker thread
fair  + process $worker \in Workers$
variables
    $task$ ;
begin
     work-stealing
    $start\_worker$:
        while TRUE do
            await $run\_queue \neq \langle\rangle$      ;

            $task := Head(run\_queue)$ ;
            $run\_queue := Tail(run\_queue)$ ;
            $running := running \cup \{task\}$ ;

            $finish\_task$:
                $running := running \setminus \{task\}$ ;
                $waiting \ := waiting \cup \{task\}$ ;
        end while ;
end process ;

end algorithm   ;
 BEGIN TRANSLATION ($chksum(pcal) =$ "$f261a5cb$" $\land chksum(tla) =$ "$14a8e3c0$")
CONSTANT $defaultInitValue$
VARIABLES $wait\_set, run\_queue, running, waiting, pc$

 define statement
$starvation\_free \triangleq \forall event \in AllTask : event \in wait\_set \leadsto \Diamond(event \in running)$

VARIABLE $task$

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

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

$$\begin{array}{ll}
Init \triangleq & \text{Global variables} \\
& \land\ wait\_set = \{\} \\
& \land\ run\_queue = \langle\rangle \\
& \land\ running = \{\} \\
& \land\ waiting\ = AllTask \\
& \text{Process worker} \\
& \land\ task = [self \in Workers \mapsto defaultInitValue] \\
& \land\ pc = [self \in ProcSet \mapsto \text{CASE } self = \text{“scheduler”} \rightarrow \text{“start\_sched”} \\
& \qquad\qquad\qquad\qquad\qquad\quad \Box\ \ self \in Subscribers \rightarrow \text{“start\_subscriber”} \\
& \qquad\qquad\qquad\qquad\qquad\quad \Box\ \ self \in Servers \ \ \rightarrow \text{“start\_server”} \\
& \qquad\qquad\qquad\qquad\qquad\quad \Box\ \ self \in Clients \ \ \rightarrow \text{“start\_client”} \\
& \qquad\qquad\qquad\qquad\qquad\quad \Box\ \ self \in Workers \rightarrow \text{“start\_worker”}]
\end{array}$$

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

$$scheduler \triangleq start\_sched$$

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

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

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

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

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

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

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

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

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

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

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

END TRANSLATION