MODULE *orchestrator*

```
EXTENDS $TLC$, $Naturals$, $Integers$, $Sequences$

CONSTANT $Workers$, $Manager$, $Clients$

$Messages \triangleq$ [$type$ : {"task"}, $s$ : $Manager$, $r$ : $Workers$] $\cup$
             [$type$ : {"working", "completed", "waiting"}, $s$ : $Workers$, $r$ : $Manager$] $\cup$
             [$type$ : {"inprogress", "finished"}, $s$ : $Manager$, $r$ : $Clients$] $\cup$
             [$type$ : {"doWork"}, $s$ : $Clients$, $r$ : $Manager$]

$Actors \triangleq \{Workers \cup Manager \cup Clients\}$

 --algorithm $orchestrator$

variables $msgs = \{\}$, $wState = [w \in Workers \mapsto$ "waiting"],
          $mState = [m \in Manager \mapsto$ "ready"],
          $cState = [c \in Clients \mapsto$ "idle"],
          $queues = [q \in Actors \mapsto \langle\rangle]$ ;

macro $send(id, msg)$ begin
    $queues := Append(queues[id], msg)$ ;
end macro ;

macro $receive(msg)$ begin
    await $Len(queues[self] > 0)$ ;
    $msg := Head(queues[self])$ ;
    $queues := Tail(queues[self])$ ;
end macro ;

process $worker \in Workers$
variable $workQueue = \langle\rangle$ ;
begin
    $WaitForWork$:
        skip ;
    $PerformWork$:
        skip ;


end process ;

process $client \in Clients$
variable $msg = \langle\rangle$ ;
begin
    $SendTaskToManager$:
        if $msg = \langle\rangle$ then
            with $m \in Manager$ do
                $send(self, [type \mapsto$ "doWork", $s \mapsto self, r \mapsto m])$ ;
            **end with** ;
        **end if** ;
    $ReceiveTaskFinish$:
        **with** $m \in Manager$ **do**
            **if**
                $\land\ msgs.type\ = \text{“finished”}$
                $\land\ mState[m] = \text{“done”}$
             **then**
                $receive(msg)$ ;
             **else**
                **goto** $SendTaskToManager$ ;
            **end if** ;
        **end with** ;
**end process** ;

**process** $manager \in Manager$
**begin**
    $NotifyClientOfCompleteJob$:
        **if** $msgs.type = \text{“completed”}$ **then**
            **with** $c \in Clients$ **do**
                $send(self, [type \mapsto \text{“finished”}, s \mapsto self, r \mapsto c])$ ;
            **end with** ;
        **end if** ;
    $ReceiveTaskFromClient$:
        **skip** ;
    $GiveTaskToWorker$:
        **skip** ;

**end process** ;


**end algorithm** ;
```

BEGIN TRANSLATION $(chksum(pcal) = \text{“901}ddb8f\text{”} \land chksum(tla) = \text{“580405}ba\text{”})$

VARIABLES $msgs, wState, mState, cState, queues, pc, workQueue, msg$

$vars \triangleq \langle msgs, wState, mState, cState, queues, pc, workQueue, msg \rangle$

$ProcSet \triangleq (Workers) \cup (Clients) \cup (Manager)$

```
$Init \triangleq$  Global variables
        $\land\ msgs = \{\}$
        $\land\ wState\ = [w \in Workers \mapsto \text{“waiting”}]$
        $\land\ mState = [m \in Manager \mapsto \text{“ready”}]$
        $\land\ cState\ = [c \in Clients \mapsto \text{“idle”}]$
        $\land\ queues\ = [q \in Actors \mapsto \langle\rangle]$
         Process worker
        $\land\ workQueue = [self \in Workers \mapsto \langle\rangle]$
```

```
    Process client
    $\land\ msg = [self \in Clients \mapsto \langle\rangle]$
    $\land\ pc = [self \in ProcSet \mapsto$ CASE $self \in Workers \rightarrow$ "WaitForWork"
                                          $\Box$ $self \in Clients \rightarrow$ "SendTaskToManager"
                                          $\Box$ $self \in Manager \rightarrow$ "NotifyClientOfCompleteJob" ]

$WaitForWork(self) \triangleq$ $\land\ pc[self] =$ "WaitForWork"
                           $\land$ TRUE
                           $\land\ pc' = [pc$ EXCEPT $![self] =$ "PerformWork" ]
                           $\land$ UNCHANGED $\langle msgs, wState, mState, cState, queues,$
                                         $workQueue, msg\rangle$

$PerformWork(self) \triangleq$ $\land\ pc[self] =$ "PerformWork"
                           $\land$ TRUE
                           $\land\ pc' = [pc$ EXCEPT $![self] =$ "Done" ]
                           $\land$ UNCHANGED $\langle msgs, wState, mState, cState, queues,$
                                         $workQueue, msg\rangle$

$worker(self) \triangleq WaitForWork(self) \lor PerformWork(self)$

$SendTaskToManager(self) \triangleq$ $\land\ pc[self] =$ "SendTaskToManager"
                                 $\land$ IF $msg[self] = \langle\rangle$
                                       THEN $\land\ \exists\, m \in Manager :$
                                                 $queues' = Append(queues[self], ([type \mapsto$ "doWork", $s \mapsto s$
                                       ELSE $\land$ TRUE
                                            $\land$ UNCHANGED $queues$
                                 $\land\ pc' = [pc$ EXCEPT $![self] =$ "ReceiveTaskFinish" ]
                                 $\land$ UNCHANGED $\langle msgs, wState, mState, cState,$
                                               $workQueue, msg\rangle$

$ReceiveTaskFinish(self) \triangleq$ $\land\ pc[self] =$ "ReceiveTaskFinish"
                                 $\land\ \exists\, m \in Manager :$
                                     IF $\land\ msgs.type =$ "finished"
                                        $\land\ mState[m] =$ "done"
                                      THEN $\land\ Len(queues[self] > 0)$
                                           $\land\ msg' = [msg$ EXCEPT $![self] = Head(queues[self])]$
                                           $\land\ queues' = Tail(queues[self])$
                                           $\land\ pc' = [pc$ EXCEPT $![self] =$ "Done" ]
                                      ELSE $\land\ pc' = [pc$ EXCEPT $![self] =$ "SendTaskToManager" ]
                                           $\land$ UNCHANGED $\langle queues, msg\rangle$
                                 $\land$ UNCHANGED $\langle msgs, wState, mState, cState,$
                                               $workQueue\rangle$

$client(self) \triangleq SendTaskToManager(self) \lor ReceiveTaskFinish(self)$

$NotifyClientOfCompleteJob(self) \triangleq$ $\land\ pc[self] =$ "NotifyClientOfCompleteJob"
                                         $\land$ IF $msgs.type =$ "completed"
```

```
                                   THEN  $\land\ \exists\, c \in Clients :$
                                                $queues' = Append(queues[self],\ ([type \mapsto \text{“finished”},$
                                   ELSE  $\land$ TRUE
                                         $\land$ UNCHANGED $queues$
                             $\land\ pc' = [pc$ EXCEPT $![self] = \text{“ReceiveTaskFromClient”}]$
                             $\land$ UNCHANGED $\langle msgs,\ wState,\ mState,$
                                                $cState,\ workQueue,\ msg\rangle$

$ReceiveTaskFromClient(self) \triangleq$ $\land\ pc[self] = \text{“ReceiveTaskFromClient”}$
                                   $\land$ TRUE
                                   $\land\ pc' = [pc$ EXCEPT $![self] = \text{“GiveTaskToWorker”}]$
                                   $\land$ UNCHANGED $\langle msgs,\ wState,\ mState,\ cState,$
                                                  $queues,\ workQueue,\ msg\rangle$

$GiveTaskToWorker(self) \triangleq$ $\land\ pc[self] = \text{“GiveTaskToWorker”}$
                              $\land$ TRUE
                              $\land\ pc' = [pc$ EXCEPT $![self] = \text{“Done”}]$
                              $\land$ UNCHANGED $\langle msgs,\ wState,\ mState,\ cState,\ queues,$
                                             $workQueue,\ msg\rangle$

$manager(self) \triangleq$ $NotifyClientOfCompleteJob(self)$
                      $\lor\ ReceiveTaskFromClient(self)$
                      $\lor\ GiveTaskToWorker(self)$

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

$Next \triangleq$ $(\exists\, self \in Workers : worker(self))$
          $\lor\ (\exists\, self \in Clients\ \ : client(self))$
          $\lor\ (\exists\, self \in Manager : manager(self))$
          $\lor\ Terminating$

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

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

 END TRANSLATION

$TypeOK \triangleq$
          $\land\ wState \in [Workers \rightarrow \{\text{“waiting”},\ \text{“working”}\}]$
          $\land\ mState \in [Manager \rightarrow \{\text{“ready”},\ \text{“busy”},\ \text{“jobComplete”}\}]$
          $\land\ cState\ \ \in [Clients\ \ \ \rightarrow \{\text{“assignTask”},\ \text{“idle”}\}]$
          $\land\ msgs \subseteq Messages$
```

\ * Modification History
\ * Last modified *Mon Mar* 11 10:40:22 *CET* 2024 by lee
\ * Created *Fri Mar* 08 22:22:11 *CET* 2024 by lee