EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science

Examination of Real-time Systems (2IMN20)
on Friday, June 29, 2018, 18:00h-21:00h.

First read the entire examination. There are 5 exercises in total. Grades are included between parentheses at all parts and sum up to 10 points. *Motivate all your answers.* Good luck!

1. *Cyclic Executives*
The task set $\mathcal{T}_1$ given in Table 1 is schedulable under fixed-priority pre-emptive scheduling (FPPS), but not under fixed-priority non-pre-emptive scheduling (FPNS). Suppose we want to use a cyclic executive to schedule this task set.

	$T_i = D_i$	C_i
τ_1	3	2
τ_2	10	3

Table 1: Task characteristics of $\mathcal{T}_1$.

(a) (1.0) How can the task set be made schedulable under FPNS? Describe the resulting task set $\mathcal{T}_1'$, and draw a schedule illustrating schedulability.
Answer: Split τ_2 in three sub-tasks, τ_{2a}, τ_{2b}, and τ_{2c}, each with a period of 10 and a computation time of 1. Guarantee a precedence relation between the sub-tasks by using a relative phasing; see Figure 1.

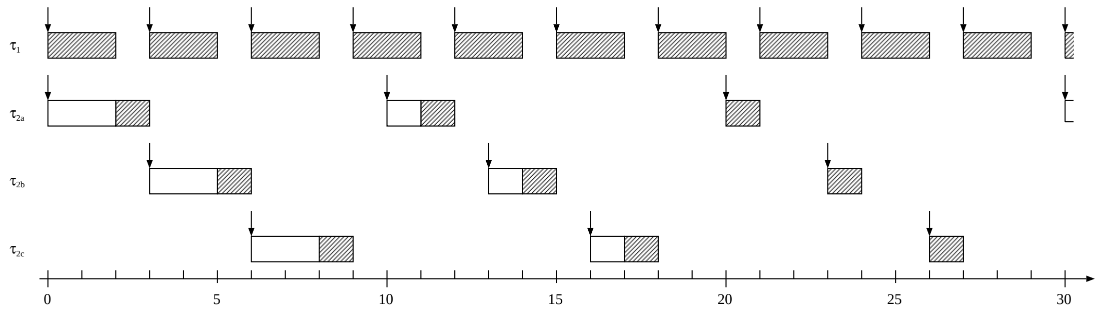

Figure 1: A timeline with a schedule of the task set $\mathcal{T}_1'$

.

See also `RTS.B3-Cyclic Executives` Exercise 2 "*Splitting a task*".

(b) (0.5) Is it always possible to use the specific solution you proposed?
Answer: No, because tasks τ_1 and τ_2 may share a resource R, and the required critical section to guarantee mutual exclusive access to R may become "split" as well.

(c) (0.5) Which type of cyclic executive can be used to schedule the resulting task set?
Answer: Multi-rate periodic, to allow proper (i.e. explicit) activation of the tasks.

(d) (1.0) Show the pseudo-code of the cyclic executive for the task set.
Answer: Considering Figure 1, we can simply create a multi-rate executive with a major cycle of length lcm (least common multiple) of the periods, i.e. 30, and using a timer with the gcd (greatest common divider) of the periods, i.e. $T_{\text{timer}} = 1$.

```
int k; /* cycle counter */

k = -1;
while(1){
  k = k + 1;
  sleep( k ); /* await timer expiration */
  if( k % 3 == 0)
  { Task_1();
  } else if( ((k-2) % 30 == 0) || ((k-11) % 30 == 0) || ((k-20) % 30 == 0) )
  { Task_2a();
  } else if( ((k-5) % 30 == 0) || ((k-14) % 30 == 0) || ((k-23) % 30 == 0) )
  { Task_2b();
  } else if( ((k-8) % 30 == 0) || ((k-17) % 30 == 0) || ((k-26) % 30 == 0) )
  { Task_2c();
  }
}
```

2. *Servers*
Consider the task set $\mathcal{T}_2'$ in Table 2 and a deferrable server S with period $T_S = 5$ and capacity $C_S = 1$. Assume rate-monotonic scheduling for the task set and the server.

	$T_i = D_i$	C_i
τ_1	3	1
τ_2	7	1

Table 2: Task characteristics of $\mathcal{T}_2$.

(a) (1.0) Draw a schedule with the worst-case response time of τ_2.
Answer: See Figure 2. Note that the server S provided its capacity as late as possible in the period ending at time 2 and as early as possible in the next period.

From an *analysis* point of view, a deferrable server can be treated as a regular task with activation jitter. From a *behavioral* point of view (illustrated by a time-line) it is slightly different, as indicated by the activation at time $t = 2$. In particular, there is no simultaneous release of the deferrable server with the other tasks, and the activation jitter interval is not shown for a deferrable server.

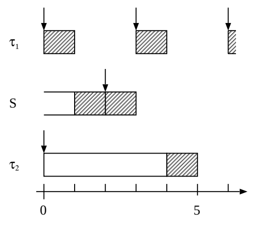

Figure 2: A timeline with a worst-case response time $WR_2 = 5$ of task τ_2.

(b) Let a soft-task τ^{soft} with a computation time of 2 be served by S.

i. (0.5) Draw a schedule with the best-case response time of the soft-task.
Answer: The timeline can be identical to the timeline in Figure 2, with an activation of τ^{soft} at time $t = 1$, i.e. S provided $2 \times C_S$ contiguously, yielding a best-case response time $BR^{\text{soft}} = 2$.

ii. (0.5) Draw a schedule with the worst-case response time of the soft-task, assuming no pending load of other soft-tasks in the system.
Answer: The formulation "*assuming no pending load of other soft-tasks in the*

system" within the question may give rise to (at least) two interpretations. On the one hand, it could be interpreted as "*there are no other soft-tasks.* On the other hand, it could be interpreted as "*when the soft-task is activated, there is no pending load of other soft-tasks*". Below, we consider both interpretations.

Alternative 1 ("*no other soft-tasks in the system*"):
We first observe that $C^{\text{soft}} = 2 \times C_S$, i.e. the soft-task needs capacity of (at least) two server periods. Because a deferrable server keeps its capacity when there are no requests, a worst-case provisioning of capacity happens when the capacity is provided as early as possible in the first and as late as possible in the last server period. As late as possible implies a maximum interference, i.e. due to task τ_1. This is illustrated in Figure 3.

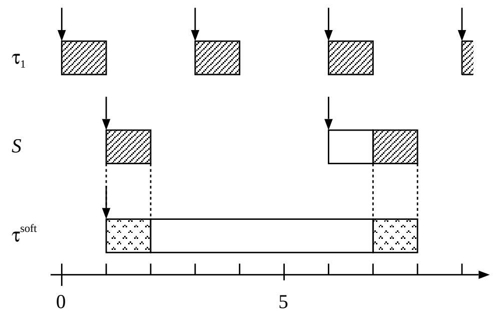

Figure 3: A timeline with a worst-case response time $WR^{\text{soft}} = 7$ of task τ^{soft} when simultaneously activated with the server at time $t = 1$, i.e. without other tasks using the server.

Alternative 2 ("*when the soft-task is activated, there is no pending load of other soft-tasks*"):
In this case, the deferrable server may be depleted as early as possible in the first period by other soft-tasks. The resulting time-line is illustrated in Figure 4.

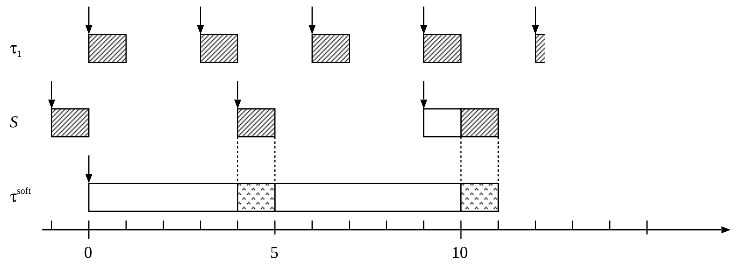

Figure 4: A timeline with a worst-case response time $WR^{\text{soft}} = 11$ of task τ^{soft} when other soft-tasks may have depleted the server.

This question is similar to Question 3 of the exam of 2018, April 13^{th}. The main difference being that the deferrable server may experience interference from task τ_1.

3. *Resource Access Protocols*
Chained blocked is a situation where a task may have to wait multiple times for shared resources.

(a) (0.5) Illustrate chained blocking, e.g. draw a time-line, with at three tasks and two shared resources R_1 and R_2, where the shared resources are protected using semaphores with priority queues. Make the assumptions on which your drawing is

based explicit, e.g. the scheduling mechanism.
Answer: Figure 5 shows an example with chained blocking.

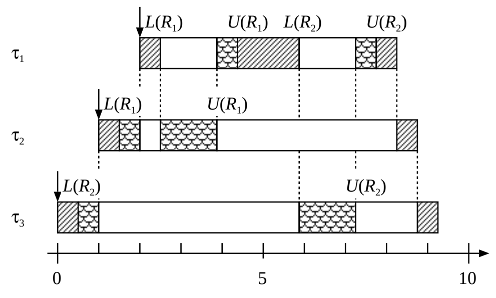

Figure 5: A timeline with chained blocking, assuming semaphores for shared resource protection.

(b) (1.0) Give at least two examples of resource access protocols that prevent chained blocking, and illustrate the resulting behavior for both examples based on the illustration drawn for question 3a.
Answer: Figure 6 shows an example with chained blocking for both the stack resource policy (SRP) as well as the highest locker protocol (HLP).

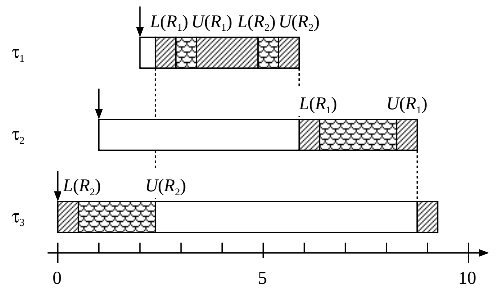

Figure 6: A timeline with chained blocking, assuming semaphores for shared resource protection.

See also `RTS.B3-Policies-3-RAP` Exercise 2.

4. *Mechanisms*
For the analysis, we assume we know the characteristics of tasks, such as minimal inter-arrival times, and (worst-case and best-case) computation times. Moreover, we may like to guarantee precedence constraints between tasks. Briefly describe a mechanism to guarantee (or enforce)

(a) (0.5) minimal inter-arrival times;
Answer: A *release guard*, preventing a release when the minimal inter-arrival time has not been passed yet.

(b) (0.5) that other tasks are not hampered when a task exceeds it worst-case computation time;
Answer: A dedicated *resource reservation* per task, e.g. a budget with a capacity equal to the task its worst-case computation time.

(c) (0.5) that other tasks are not hampered when a task executes *shorter* than its best-case computation time;
Answer: An additional *resource reservation* per task, e.g. an idling periodic server for its best-case computation time and a gain-time providing periodic server with a capacity equal to the difference of the worst-case and best-case computation time.

(d) (0.5) precedence constraints.
Answer: *Semaphores.*

5. *Level-i active period*
It may be required to consider all jobs in a so-called level-i active period to determine the worst-case response time of a task τ_i.

(a) (0.5) Describe the notion of level-i active period in your own words.
Answer: The notion of level-i active period is defined in terms of pending load, i.e. it is an interval $[t_s, t_e)$, where the pending load $P_i(t)$ of tasks with a priority higher than or equal to the priority of task τ_i is to zero for $t = t_e$ and $t = t_s$, and where the pending load is higher than zero throughout the interval. Pending load $P_i(t)$ at time t is the amount of work that still needs to be performed due to activations prior to time t.
See also Exercise 6b of the exam of 2016, June 29^{th}.

(b) Why is it required to consider all jobs in a level-i active period

i. (0.5) for fixed-priority pre-emptive scheduling and deadlines larger than periods?
Answer: In addition to delays caused by jobs of tasks with a higher priority, a next job of a task may also be *directly* delayed by its previous job.

ii. (0.5) for limited pre-emptive scheduling, e.g. fixed-priority scheduling with deferred preemptions, even for deadlines at most equal to periods?
Answer: In addition to delays caused by jobs of tasks with a higher priority, a next job of a task may also be *indirectly* delayed by its previous job, because that previous job delayed jobs of tasks with a higher priority.

See also Exercise 6c of the exam of 2016, June 29^{th} and Exercise 2 of 2012, January 23^{rd}.