EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science

Examination Real-time Architectures (2IN25)
on Friday, June 13^{th} *2008, 14.00h-17.00h.*

First read the entire examination. There are 6 exercises in total. Grades are included between parentheses at all parts and sum up to 9 points. Good luck!

1. A typical recursive equation to determine the worst-case response time of a task τ_i under fixed-priority pre-emptive scheduling (FPPS) is given by

$$x = C_i + \sum_{j \in hp(i)} \left\lceil \frac{x}{T_j} \right\rceil C_j,$$

where $hp(i)$ denotes the set of tasks with a *higher priority* than τ_i and tasks have distinct priorities.

(a) (0.5) Give at least two additional assumptions that need to hold to use this equation.
Answer Examples: (i) tasks are independent, (ii) deadlines are at most equal to periods, i.e. $D_i \leq T_i$, and (iii) tasks do not suspend themselves. Note that tasks need not be strictly periodic, e.g. the equation is also valid for elastic and sporadic tasks.

An RTOS (Real-Time Operating System) typically only provides a limited number of priority levels. Hence, multiple tasks may have to *share* a priority level. Let $ep(i)$ denote the set of tasks with priority equal to that of τ_i.

(b) (0.5) Extend the recursive equation given above to account for multiple tasks sharing a priority level. Motivate your answer.
Answer The worst-case response time is assumed when a task τ_i is simultaneously released with all tasks in $hp(i) \cup ep(i)$, and is the last task from $ep(i) \cup \{\tau_i\}$ that is allowed to execute. Note that it is assumed that $i \notin ep(i)$.
The easiest approach to analyse the system is to treat all tasks in $ep(i)$ as higher priority tasks, i.e.

$$x = C_i + \sum_{j \in hp(i) \cup ep(i)} \left\lceil \frac{x}{T_j} \right\rceil C_j.$$

This is pessimistic, however, because *at most one job* of every task $\tau_j \in ep(i)$ can delay the execution of τ_i, and we therefore get

$$x = C_i + \sum_{j \in ep(i)} C_j + \sum_{j \in hp(i)} \left\lceil \frac{x}{T_j} \right\rceil C_j.$$

2. Consider (fixed-priority) servers.

(a) (0.5) Describe how a server compares to background scheduling.
Answer See book.

(b) (0.5) Under which conditions does a deferrable server behave as a periodic task without self-suspension?
Answer When there is always work pending for the deferrable server if its capacity is larger than zero.

3. (2.0) Let tasks τ_1 and τ_2 both use resources r_1 and r_2. Task τ_1 first locks r_1 and subsequently r_2, and τ_2 first locks r_2 and subsequently r_1, which may give rise to a deadlock without a resource access protocol; see Figure 1. Discuss what happens when a resource

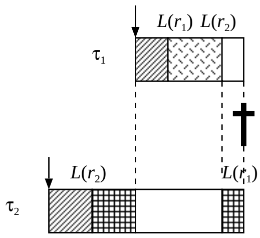

Figure 1: A deadlock situation.

access protocol is used and illustrate the behavior by means of appropriate drawings for PIP, HLP, PCP, and SRP.
Answer This exercise has been taken from RTA.Exercises-4 (2007/2008).

4. (1.0) Give an example of a task set that is schedulable with fixed-priority non-pre-emptive scheduling (FPNS) but is *not* schedulable with fixed-priority pre-emptive scheduling (FPPS).
Answer Compared to FPPS, FPNS can *decrease* the worst-case response time of lower priority tasks. Hence, we could aim at creating a task-set where

- a lower priority task is not schedulable under FPPS;
- that lower priority task is schedulable under FPNS;
- all other tasks are also schedulable under FPNS.

Table 1 provides an example of such a task-set (assuming arbitrary phasing), where τ_1 has highest and τ_3 has lowest priority.

	T	C
τ_1	9	2
τ_2	11	2
τ_3	12	6

Table 1: A task-set schedulable under FPNS but not under FPPS.

Alternative approaches: assume a specific phasing or assume resource sharing between a highest and lowest priority task *without* a resource access protocol (and a middle priority task that preempts the lowest priority task during resource access under FPPS).

5. (2.0) Consider the system configuration shown in Figure 2, with characteristics as given in Table 2. The input streams are (strictly) periodic and have arbitrary phasing. The tasks are scheduled by means of FPPS, where τ_1 has highest priority and task τ_3 has lowest priority. Determine the worst-case end-to-end delay of an event of input stream I_2. (*Hint:*

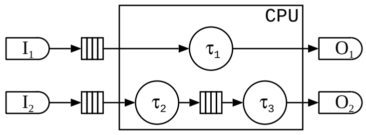

Figure 2: System configuration.

	T		C
I_1	80	τ_1	21
I_2	50	τ_2	20
		τ_3	10

Table 2: Characteristics of input streams and tasks.

determine a critical instant and use a timeline.)
Answer A worst-case situation is characterized by a simultaneous arrival of both events I_1 and I_2. The behavior of the system under worst-case conditions is shown in Figure 3.

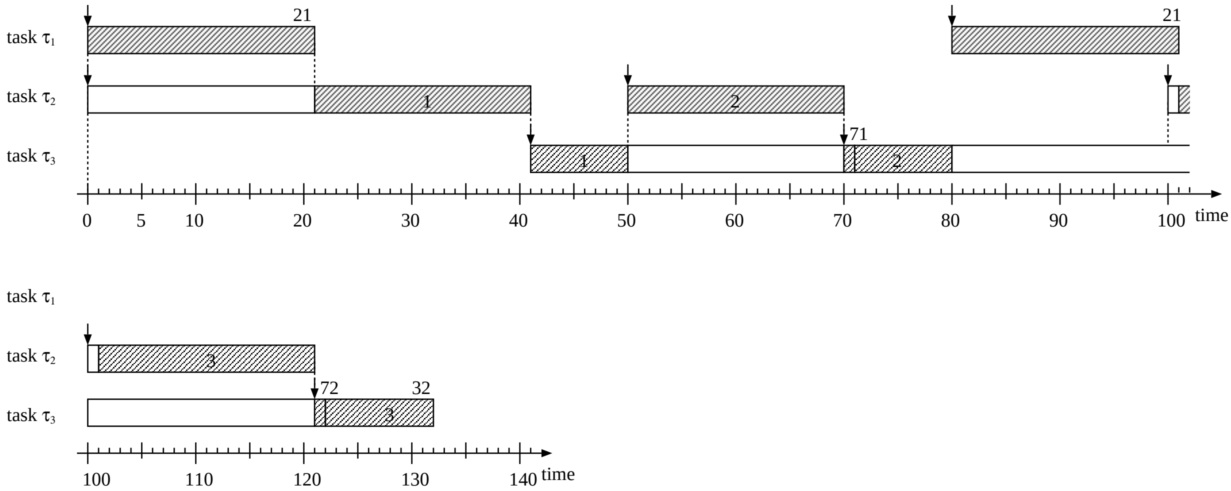

Figure 3: Timeline of tasks, where the numbers at the top-right corner of the boxes represent end-to-end response times of I_1 and I_2.

The end-to-end response times of $I_{2,k}$ are given in Table 3, where $R_{I_{2,k}} = R_{2,k} + R_{3,k}$. The worst-case end-to-end response time $WR_{I_2} = 72$. Note that $WR_{I_2} < \max_k R_{2,k} + \max_k R_{3,k} = 41 + 52 = 93$.

Task τ_3 is triggered by τ_2, and the *activation* of τ_3 therefore coincides with the *completion* of τ_2. When the temporal offset between the tasks is not taken into account, the analysis becomes very pessimistic. As an example, assuming an activation of task τ_3 at time $t = 0$ rather than $t = 41$ would yield a response time $R_{3,1} = 71$ rather than 30, and an end-to-end delay of 112.

6. The lecture of Dr. Alina Weffers-Albu concerned *Behavioral Analysis of Real-Time Systems with Interdependent Tasks.*

   (a) (1.0) Explain which real-time problems were addressed.

	$k = 1$	2	3
$R_{2,k}$	41	20	21
$R_{3,k}$	30	52	11
$R_{I_{2,k}}$	71	72	32

Table 3: Response times of tasks τ_2 and τ_3 and end-to-end response times of the input stream I_2.

(b) (0.5) Explain how these problems have been solved.

(c) (0.5) Explain why these approaches were taken.

Answers See slides of the lecture.