EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science

Examination Real-time Architectures (2YN26)
on Monday, April 11^{th} 2011, 9.00h-10.30h.

First read the entire examination. There are 6 exercises in total. Grades are included between parentheses at all parts and sum up to 10 points. Good luck!

1. Consider two periodic tasks τ_1 and τ_2, with characteristics as given in the following table, which are scheduled by means of fixed-priority scheduling with deferred pre-emption (FPDS), where τ_1 has a higher priority than τ_2.

	$T = D$	C
τ_1	3	$1\frac{1}{2}$
τ_2	5	$1 + 1\frac{1}{4}$

   (a) (1.5) Determine the worst-case length WL_2 of the level-2 active period.
   Answer: Draw a timeline with a simultaneous release of both tasks at time $t = 0$. Because the pending load becomes zero at time $t = 9$, the worst-case length $WL_2 = 9$.

   (b) (1.0) Determine the worst-case response times WR_1^{D} of τ_1 and WR_2^{D} of τ_2 using

$$WR_{ik}^{\mathrm{D}} = \begin{cases} WR_i^{\mathrm{P}}(B_i^{\mathrm{D}} + kC_i - F_i) + F_i - (k-1)T_i & \text{for } i < n \\ WO_i^{\mathrm{P}}(kC_n - F_n) + F_n - (k-1)T_n & \text{for } i = n \end{cases}, \tag{1}$$

   and the following recursive equations for $WR_i^{\mathrm{P}}(C)$ and $WO_i^{\mathrm{P}}(C)$, respectively.

$$x = C + \sum_{j<i} \left\lceil \frac{x}{T_j} \right\rceil C_j, \tag{2}$$

$$x = C + \sum_{j<i} \left(\left\lfloor \frac{x}{T_j} \right\rfloor + 1 \right) C_j. \tag{3}$$

   Answer: For task τ_1, it is only necessary to consider the first job: $WR_1^{\mathrm{D}} = 2\frac{3}{4}$. Because the worst-case length WL_2 of the level-2 active period contains $\left\lceil \frac{WL_2}{T_2} \right\rceil = 2$ jobs, we need to consider two jobs for task τ_2. $WR_{2,1}^{\mathrm{D}} = 3\frac{3}{4}$ and $WR_{2,2}^{\mathrm{D}} = 4$, hence $WR_2^{\mathrm{D}} = 4$.

2. Assume an RTOS (Real-Time Operating System) supporting fixed-priority pre-emptive scheduling FPPS. An RTOS 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 a priority equal to that of τ_i.

   (a) (0.5) Describe how such an RTOS may schedule multiple tasks sharing a priority level.
   Answer: By means of, for example, first-come first-served (FCFS) or round robin (RR).

(b) (0.5) Determine a critical instant for task τ_i.
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 $\tau_i \notin ep(i)$.

(c) (0.5) A typical recursive equation to determine the worst-case response time of a task τ_i under FPPS for tasks having unique priorities 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. Extend the recursive equation to account for multiple tasks sharing a priority level. Motivate your answer.
Answer: The easiest approach to analyse the system is to treat all tasks in $ep(i)$ as higher priority tasks, i.e.

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

For FCFS this is pessimistic, however, because *at most one job* of every task $\tau_j \in ep(i)$ can delay the execution of τ_i when multiple tasks sharing a priority level, and we therefore get

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

3. For *tick scheduling*, the *scheduler* only executes at clock interrupts.

   (a) (0.5) Describe what will happen when a task *finishes* its current job.
   Hint: remember that a *scheduler* and a *dispatcher* have different roles.
   Answer: The *scheduler* determines the *order* in which tasks are executed. The *dispatcher* allocates the CPU based on that order. When a task finishes its job, the dispatcher simply allocates the next task in the ready-queue to the CPU.

   (b) (0.5) Describe what will happen when a task exits a non-preemptable section.
   Answer: The task is allowed to continue, because the scheduler is *not* invoked. As a result, when the task is allowed to enter a next non-preemptable section *before* the scheduler is invoked, that task may prolong blocking higher priority tasks.

4. One of the lectures concerned "*A QoS approach for multimedia consumer terminals with media processing in software*". The aim of the QoS approach was *cost-effective high-quality video processing in software for multimedia consumer terminals*, motivated by the requirements for *openness* and *flexibility* of these systems, and having as boundary condition that *the existing system qualities should be preserved*.

   (a) (0.5) Explain which real-time problems were addressed.

   (b) (1.0) Explain how these problems have been solved.

   Answers: See slides.

5. The following questions concern Hierarchical Scheduling Frameworks (HSFs) and independent applications.

(a) (1.0) In "[Shin et al 03] I. Shin and I. Lee, *Periodic resource model for compositional real-time guarantees*, In: Proc. 24^{th} IEEE Real-Time Systems Symposium (RTSS), pp. 2-13, December 2003", a *resource supply bound function* $\texttt{sbf}_\Gamma(t)$ of a time interval of length t is defined that calculates the minimum resource supply of Γ during t units. Describe how this function is used to determine the schedulability of a set $\mathcal{T}$ of independent strictly periodic, hard real-time tasks under FPPS.
Hint: Describe a so-called *demand bound function* $\texttt{dbf}_\mathcal{T}(i,t)$ of task τ_i of $\mathcal{T}$ in a time interval of length t, determine the worst-case response time of a task τ_i, and give a schedulability condition.
Answer: Every job of task $\tau_i \in \mathcal{T}$ needs to finish before its deadline D_i, i.e. $WR_i \leq D_i$. The worst-case response time WR_i of task τ_i is given by the smallest positive solution of

$$\texttt{sbf}_\Gamma(t) = \texttt{dbf}_\mathcal{T}(i,t), \tag{4}$$

where

$$\texttt{dbf}_\mathcal{T}(i,t) = C_i + \sum_{j \in hp(i)} \left\lceil \frac{t}{T_j} \right\rceil C_j. \tag{5}$$

Note that (4) is similar to the classical recursive equation for worst-case response time analysis, where the left-hand side represents the amount of resource provided in an interval of length t and the right-hand side represents the amount of resources requested (or demanded) in that interval.

Rather than determining the worst-case response time explicitly for every task, it is also possible to use

$$\underset{i}{\forall} \underset{0<t\leq D_i}{\exists} \texttt{dbf}_\mathcal{T}(i,t) \leq \texttt{sbf}_\Gamma(t). \tag{6}$$

(b) (1.0) The worst-case response time analysis of tasks presented in "[Davis et al 05] R.I. Davis and A. Burns, *Hierarchical Fixed Priority Pre-Emptive Scheduling*, In: Proc. 26^{th} IEEE Real-Time Systems Symposium (RTSS), pp. 389-398, December 2005." improves on the analysis in [Shin et al 03]. Explain *why*.
Answer: The analysis in [Davis et al 05] takes the worst-case interference of higher priority subsystems into account, given the characteristics of these subsystems. The *resource supply bound function* $\texttt{sbf}_\Gamma(t)$ assumes a worst-case interference of other subsystems, without knowing their characteristics or even their existance, and is therefore pessimistic.

6. In "[Bertogna et al 07] M. Bertogna, N. Fisher, and S. Baruah. *Static-priority scheduling and resource hold times*, In: Proc. 15^{th} International Workshop on Parallel and Distributed Real-Time Systems, pp. 1-8, March 2007.", the notion of *resource hold time* is defined for fixed-priority scheduling using the Stack Resource Policy (SRP) and algorithms are presented to (*i*) calculate resource hold times and (*ii*) to minimize resource hold times.

   (a) (0.5) Describe the notion of *resource hold time* in your own words.

   (b) (1.0) Describe *how* a resource hold time can be reduced and *what* determines its minimum.

   Answer: See paper.