jian-jia.chen [☺] tu-dortmund.de
lea.schoenberger [☺] tu-dortmund.de

Exercise for the lecture
Embedded Systems
Wintersemester 17/18

Written Exercise Sheet 5

Hints: These assignments will be discussed at E23 OH14, from 10:15 am - 11:45 am on 20. Dez. 2017. You are not obligated to turn in the solutions. T_i indicates the period of task τ_i, D_i its relative deadline, and C_i its WCET.

1 Schedulability Analysis

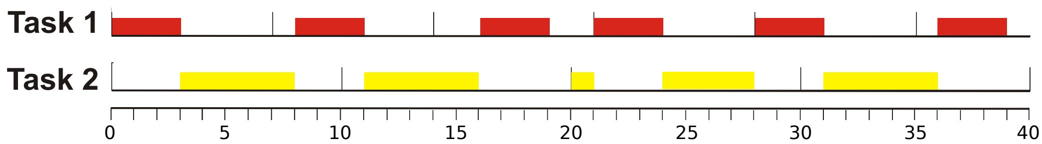
Consider the following parameters for periodic tasks with $T_i = D_i$ under *rate-monotonic scheduling*:

	T_i	C_i
τ_1	5	1
τ_2	8	3
τ_3	9	2

Assume that the tasks shall be executed on a single core system. Which is the hyperperiod? Examine formally, if a feasible schedule with the given parameters exists such that all deadlines are met, and verify your result. What are your findings? Why?

2 Scheduling of independent periodic tasks

The following task diagram is given:



Indicate the parameters of the tasks (C_i, $T_i = D_i$). Which scheduling policy was possibly used? Why? Draw the diagram for the case that the tasks are ordered according to a *rate monotonic scheduling policy*. What do you notice?

3 Harmonic Task Systems

Consider the following parameters for periodic tasks with $T_i = D_i$:

C_i	0.2	2	2	1.5	1	14	28.8
T_i	2	6	12	24	24	72	288
D_i	2	6	12	24	24	72	288
U_i	0.1	1/3	1/6	0.0625	0.0417	0.195	0.1

Assume the tasks shall be executed on a single core system. Determine formally if the rate monotonic schedule is *feasible*. Determine formally if the earliest deadline first schedule is *feasible*.

4 Critical Instant Theorem

Explain the critical instant theorem for uniprocessor fixed-priority scheduling in your words. As mentioned in the lecture, the critical instant theorem for uniprocessor fixed-priority scheduling is very fragile if the assumptions are not met. To apply the critical instant theorem, quite a few conditions have to be satisfied. Please indicate which of the following conditions are correct and which of them are incorrect. If a condition is incorrect, please correct it.

- The task set consists of only independent tasks.
- The task set must be *strictly* periodic.
- The scheduling algorithm is fixed-priority preemptive scheduling.
- Early completion of jobs is not possible. A job has to spin till its worst-case execution time if it finishes earlier.
- No task voluntarily suspends itself. That is, a job cannot suspend itself during its execution.
- The relative deadline of a task can be larger than its period.
- Scheduling overheads (context switch overheads) are zero.
- All periodic/sporadic tasks have zero release jitter (the time from the task arriving to it becoming ready to execute).

5 Periodic Tasks

Is the following program a periodic task with period T? Explain your answer.

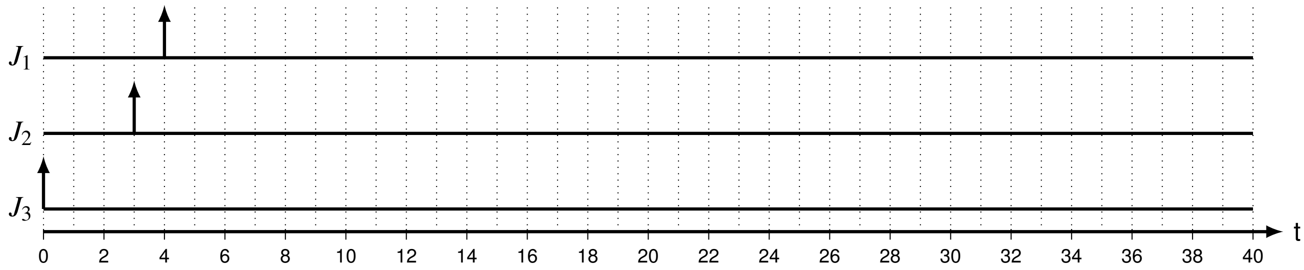
```
while (true)
   start := get the system tick;
   perform analog-to-digital conversion to get y;
   compute control output u;
   output u and do digital-to-analog conversion;
   end := get the system tick;
   timeToSleep := T-(end-start);
   sleep timeToSleep;
end while
```

6 Priority Inversion

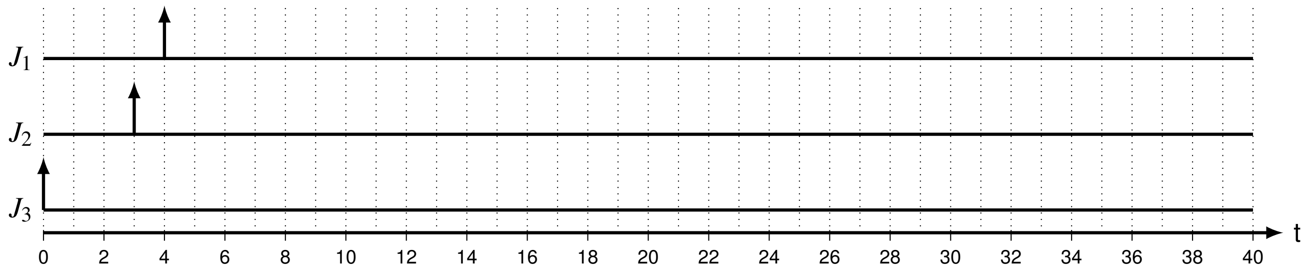
We are considering a system with three jobs J_1, J_2 and J_3. The priority of job 1 is assumed to be highest, the priority of job 3 is assumed to be lowest. These jobs become available as indicated in the following diagram.

- Job J_3 executes P(S) 2 time units after it becomes available and V(S) after 4 time units of execution time. (J_3 runs for 2 time units first and then enters the critical section. Its critical section length is 2 time units.) Its total execution time is 8 units of time.
- Job J_2 has a total execution time of 25 time units.
- Job J_1 executes P(S) after 2 units of execution time and V(S) after 5 units of execution time. (J_1 runs for 2 time units first and then enters the critical section. Its critical section length is 3 time units.) Its total execution time is 6 units of time.

What is the schedule of the system if we use pre-emptive, priority-based scheduling? Include calls to P(S) and V(S). Mark time intervals of priority inversion.



What is the schedule of the system if we use pre-emptive, priority-inheritance protocol (PIP)? Include calls to P(S) and V(S). Mark time intervals of priority inversion.



7 PCP Example

Draw the current priority ceiling $\Pi'(t)$ of the system and the current priority of the jobs in the two examples of PCP (i.e., x axis with respect to time and y axis with respect to the priority levels) given in the lecture (i.e., Pages 19 and 20 in es-chen-4.4.pdf).

8 Distributed PCP (DPCP) (advanced**)

Consider a system with multiple processors. A task is statically assigned on one processor. One additional processor, P_0, is allocated as a *synchronization processor*. That is, all the critical sections are executed on processor P_0 by using PCP under the rate-monotonic priority assignment. This protocol, called Distributed PCP (DPCP), was proposed by Raj Rajkumar in 1990.

In the following concrete example, we have three tasks, each of them assigned on one processor and all the critical sections are protected by using *one binary semaphore*. Therefore, multi-tasking only takes place on P_0. Task τ_1 is on processor P_1, task τ_2 is on processor P_2, and task τ_3 is on processor P_3. In Tabelle 1, for a task τ_k, C_k is the worst-case execution time (including the critical section length), T_k is the period, N_k is the number of critical sections per job invocation, and L_k is the worst-case critical section length (per critical section). Note that early completions are possible.

To analyze whether task τ_k can meet its deadline, we need to analyze its *remote blocking time* B_k on P_0. In the above simple example, since there is only one task per processor (except P_0), we can then simply validate whether $B_k + C_k \leq T_k$. By using the critical instant theorem, Mr. Smart argues that the additional delay due to PCP on P_0 for task τ_k is upper bounded by $L_k \cdot (\max_{j>k} L_j) + \sum_{i=1}^{k-1} \left\lceil \frac{T_k}{T_i} \right\rceil L_i N_i = t$. The first term $L_k \cdot (\max_{j>k} L_j)$ is due to the fact that each critical section access can be blocked by a lower-priority task. The second term $\sum_{i=1}^{k-1} \left\lceil \frac{T_k}{T_i} \right\rceil L_i N_i$ is due to the interference from the higher-priority tasks under the critical instant theorem. Therefore, he concludes that

- B_1 is upper bounded by 4,

τ_k	$P(\tau_k)$	C_k	T_k $(= D_k)$	N_k	L_k
τ_1	P_1	6	10	1	2
τ_2	P_2	11	18	1	4
τ_3	P_3	8	20	3	1

Tabelle 1: Task parameters for the counterexample.

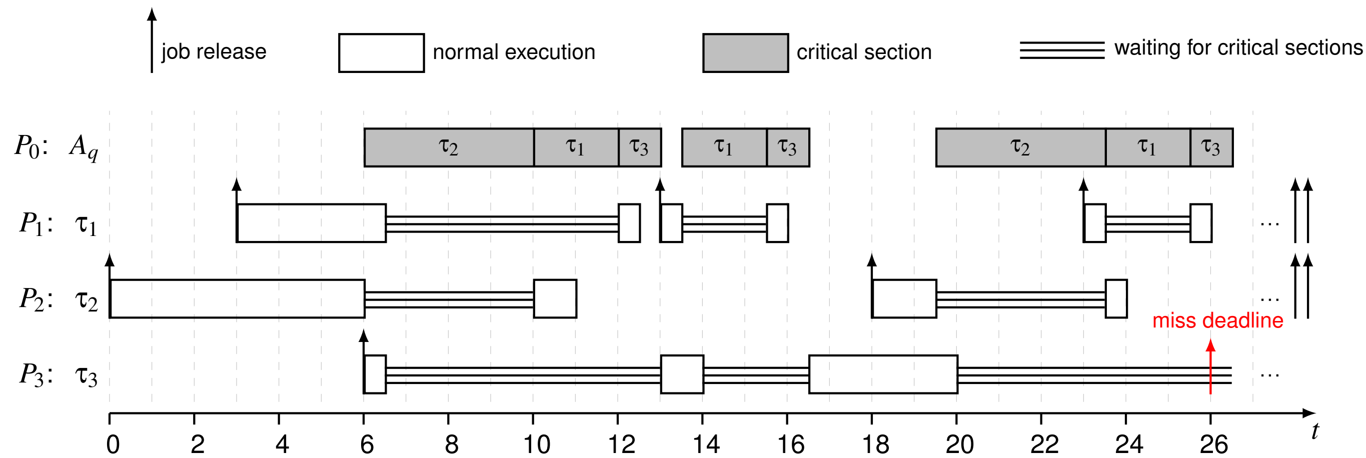


Abbildung 1: An example schedule where τ_3 misses a deadline

- B_2 is upper bounded by $1 + 2 \cdot 2 = 5$, and
- B_3 is upper bounded by $0 + 2 \cdot 2 + 4 \cdot 2 = 12$.

Since $C_k + B_k \leq T_k \forall k = 1, 2, 3$, he concludes that this task set is feasible under DPCP.

However, there is a concrete counterexample in Abbildung 1, showing that task τ_3 misses the deadline. What went wrong in the above analysis?