TECHNISCHE UNIVERSITÄT DRESDEN

CFAED CENTER FOR ADVANCING ELECTRONICS DRESDEN

Networked Embedded Systems Group

Dr. Marco Zimmerling

Networked Embedded Systems WS 2016/17

Sample Solutions to Exercise 1: Real-time Scheduling

Discussion date: December 9, 2016

Task 1: Scheduling Function and Parameters of Real-time Tasks

A real-time system needs to execute four tasks J_1, J_2, J_3, and J_4. Their arrival times a_i and absolute deadlines d_i are listed in Table 1. The scheduling function $\sigma(t)$ is shown in Figure 1.

Table 1: Task set for Task 1.

	J_1	J_2	J_3	J_4
arrival time a_i	0	6	4	2
absolute deadline d_i	9	18	22	7

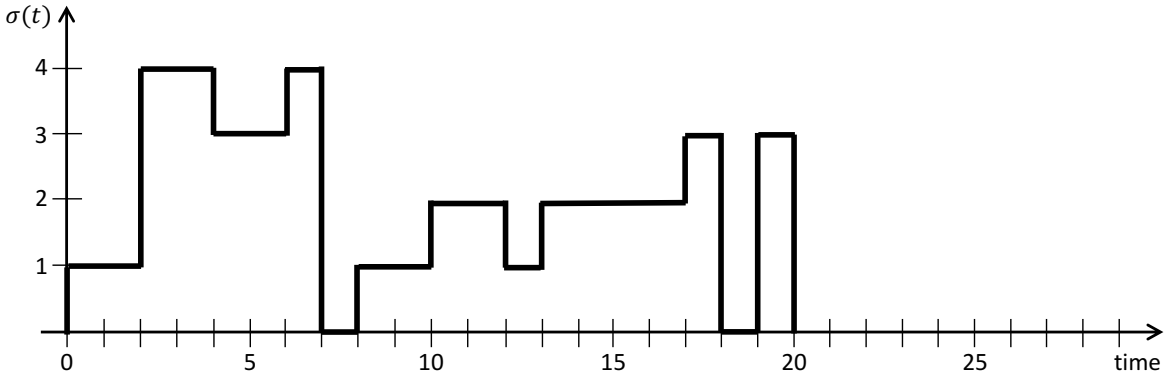

Figure 1: Scheduling function $\sigma(t)$ for Task 1.

(a) Determine the maximum lateness when tasks are executed according to the given scheduling function.

Solution: The lateness of each individual task is

- $L_1 = f_1 - d_1 = 13 - 9 = 4$
- $L_2 = f_2 - d_2 = 17 - 18 = -1$
- $L_3 = f_3 - d_3 = 20 - 22 = -2$
- $L_4 = f_4 - d_4 = 7 - 7 = 0$

Thus, the maximum lateness is 4. It is induced by task J_2, which is the only task that violates the specified timing constraints and hence has a lateness greater than zero.

(b) Determine the laxity of each task.

Solution: The tasks' computation times are $C_1 = 5$, $C_2 = 6$, $C_3 = 4$, and $C_4 = 3$. Based on those, the laxity of each individual task can be computed as follows:

- $X_1 = d_1 - a_1 - C_1 = 9 - 0 - 5 = 4$
- $X_2 = d_2 - a_2 - C_2 = 18 - 6 - 6 = 6$
- $X_3 = d_3 - a_3 - C_3 = 22 - 4 - 4 = 14$
- $X_4 = d_4 - a_4 - C_4 = 7 - 2 - 3 = 2$

(c) Compute the processor utilization U for the interval between time $t = 0$ and time $t = 20$.

Solution: $U = 18/20 = 0.9$

(d) Is this schedule feasible? If not, modify the scheduling function so that the task set is schedulable.

Solution: A schedule is *feasible* if all tasks can be completed according to a set of specified constraints. However, the given scheduling function does not complete all tasks by their deadlines, as visible in Figure 2.

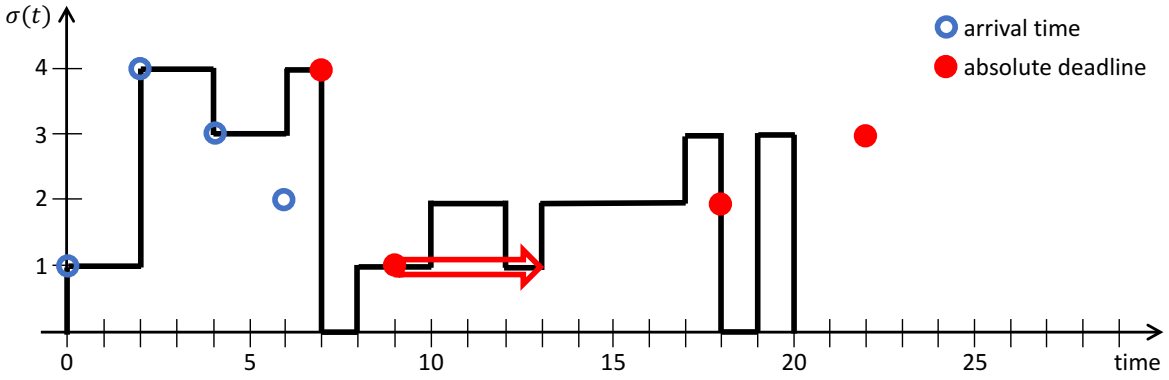

Figure 2: Scheduling function $\sigma(t)$ from Task 1 with the violation of task J_1's deadline.

Figure 3 shows one of the many possible modified scheduling functions that completes all tasks by their deadlines.

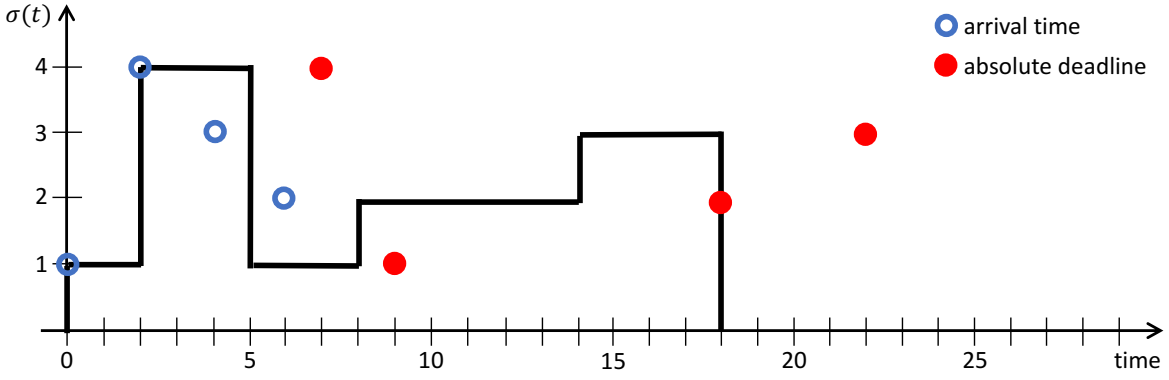

Figure 3: Modified scheduling function $\sigma(t)$ for Task 1 that meets all deadlines.

Task 2: Earliest Deadline Due (EDD)

Check whether the Earliest Deadline Due (EDD) algorithm produces a feasible schedule for the task set in Table 2. The tasks are independent and arrive synchronously at time $t = 0$. Determine the schedule.

Table 2: Task set for Task 2.

	J_1	J_2	J_3	J_4
execution time C_i	6	2	4	3
relative deadline D_i	16	7	5	12

Solution: EDD can schedule independent tasks with the same arrival time by following Jackson's rule: Given a set of n real-time tasks, any algorithm that executes the tasks in order of non-decreasing deadline is optimal with respect to minimizing the maximum lateness of the task set. When applied to the given task set, EDD produces a feasible schedule, as shown in Figure 4. As visible from the figure, EDD schedules the four tasks in order of non-decreasing deadline as follows: J_3, J_2, J_4, J_1.

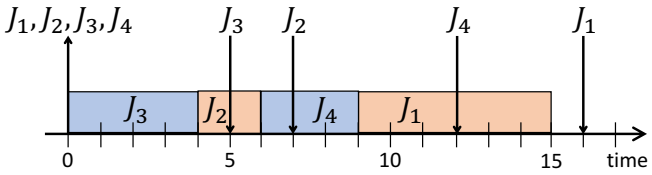

Figure 4: Feasible schedule produced by EDD for the task set from Task 2.

Task 3: Earliest Deadline First (EDF)

Given is a set of five tasks as shown in Table 3.

Table 3: Task set for Task 3.

	J_1	J_2	J_3	J_4	J_5
arrival time a_i	0	13	3	0	9
execution time C_i	3	4	2	5	2
absolute deadline d_i	17	19	6	8	15

(a) Determine the Earliest Deadline First (EDF) schedule. Is the schedule feasible?

Solution: EDF can schedule independent tasks with arbitrary arrival times in a preemptive fashion by following Horn's rule: Given a set of n real-time tasks with arbitrary arrival times, any algorithm that at any point in time executes the task with the earliest absolute deadline among all ready tasks is optimal with respect to minimizing the maximum lateness. When applied to the given task set, EDF produces a feasible schedule, as shown in Figure 5.

(b) At time $t = 2$, a new task J_n arrives with execution time $C_n = 3$ and absolute deadline $d_n = 11$. Is the new task set (including J_n) still schedulable or do you need to reject the newly arrived task?

Solution: The new task J_n that arrives at time $t = 2$ can be accepted, because the resulting task set that includes J_n remains schedulable; the new schedule is shown in Figure 6. As described in the lecture, this can be checked by computing at certain time points the worst-case finishing times of the tasks and comparing them with the absolute deadlines. In the following, we perform this check in an online fashion; that is, we perform the EDF schedulability test each time a new task arrives, and consider only those tasks that are present in the system at this point in time. We process those tasks in order of increasing absolute deadline.

From (a) we know that before task J_n arrives at time $t = 2$ all active tasks, J_1 and J_4, are feasible.

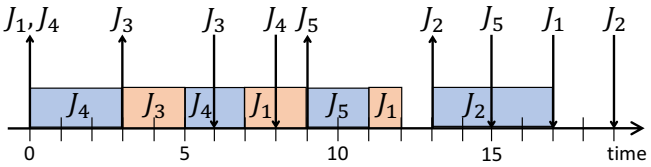

Figure 5: Feasible schedule produced by EDF for the task set from Task 3.

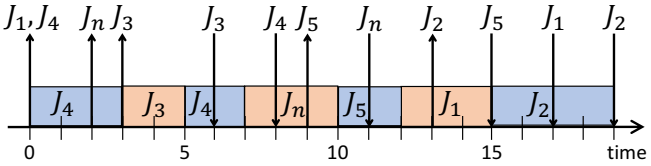

Figure 6: Feasible schedule produced by EDF after adding the new task J_n to the task set from Task 3.

At time $t = 2$, we have three tasks in the system: J_1, J_4, and the new task J_n. For these three tasks we perform the EDF schedulability test in order of increasing absolute deadline: Set $f_0 = t = 2$.

- Task J_4: $f_1 = f_0 + c_4(2) = 2 + 3 = 5 \leq 8 = d_4$ (OK)
- Task J_n: $f_2 = f_1 + c_n(2) = 5 + 3 = 8 \leq 11 = d_n$ (OK)
- Task J_1: $f_3 = f_2 + c_1(2) = 8 + 3 = 11 \leq 17 = d_1$ (OK)

Thus, at time $t = 2$, all tasks in the system are feasible.

At time $t = 3$, the next task, J_3, arrives. We now have four active tasks in the system: J_1, J_3, J_4, and J_n. The schedulability test proceeds as follows: Set $f_0 = t = 3$.

- Task J_3: $f_1 = f_0 + c_3(3) = 3 + 2 = 5 \leq 6 = d_3$ (OK)
- Task J_4: $f_2 = f_1 + c_4(3) = 5 + 2 = 7 \leq 8 = d_4$ (OK)
- Task J_n: $f_3 = f_2 + c_n(3) = 7 + 3 = 10 \leq 11 = d_n$ (OK)
- Task J_1: $f_4 = f_3 + c_1(3) = 10 + 3 = 13 \leq 17 = d_1$ (OK)

Thus, at time $t = 2$, all tasks in the system are feasible.

The next task to arrive is J_5. It arrives as $t = 8$. At this time, we have three active tasks in the system: J_1, J_5, and J_n. The schedulability test proceeds as follows: Set $f_0 = t = 8$.

- Task J_n: $f_1 = f_0 + c_n(8) = 8 + 2 = 10 \leq 11 = d_n$ (OK)
- Task J_5: $f_2 = f_1 + c_5(8) = 10 + 2 = 12 \leq 15 = d_5$ (OK)
- Task J_1: $f_3 = f_2 + c_1(8) = 12 + 3 = 15 \leq 17 = d_1$ (OK)

Thus, at time $t = 8$, all tasks in the system are feasible.

Finally, task J_2 arrives at $t = 13$. At this time, we have two active tasks in the system: J_1 and J_2. The schedulability test proceeds as follows: Set $f_0 = t = 13$.

- Task J_1: $f_1 = f_0 + c_1(13) = 13 + 2 = 15 \leq 17 = d_1$ (OK)
- Task J_2: $f_2 = f_1 + c_2(13) = 15 + 4 = 19 \leq 19 = d_2$ (OK)

Thus, we can conclude that the whole schedule, as shown in Figure 6, is feasible.

Task 4: Fixed-priority Scheduling: Rate Monotonic (RM)

Consider the set of periodic tasks given in Table 4; assume that the first instance of each task arrives at time $t = 0$ (i.e., the phases Φ_i are 0) and that relative deadlines are equal to periods (i.e., $D_i = T_i$).

Table 4: Task set for Task 4.

	τ_1	τ_2	τ_3
execution time C_i	1	2	3
period T_i	4	6	8

(a) Use the sufficient test to check if the task set is schedulable under Rate-monotonic (RM) scheduling.

RM assigns static priorities to tasks before execution, so that tasks with smaller periods have higher priorities. RM scheduling is preemptive (i.e., higher-priority tasks preempt lower-priority tasks) and optimal; that is, if a tast set cannot be scheduling using RM, it cannot be scheduling by any other fixed-priority scheduling algorithm.

For a set of n periodic tasks, the sufficient schedulability test based on the processor utilization U is

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$$

For the task set given in Table 4, we have $U = 1/4 + 2/6 + 3/8 \approx 0.958 \nleq 3(2^{1/3} - 1) \approx 0.779$. Thus, the sufficient schedulability test failed.

(b) Construct the schedule using RM for the interval $[0, 20]$. Identify deadline misses if they exist.

The schedule produced by RM in the interval $[0, 20]$ is shown in Figure 7. In this interval, there is one deadline miss: the first instance of task τ_3 completes only at time $t = 10$, but the absolute deadline of that instance is at time $t = 8$.

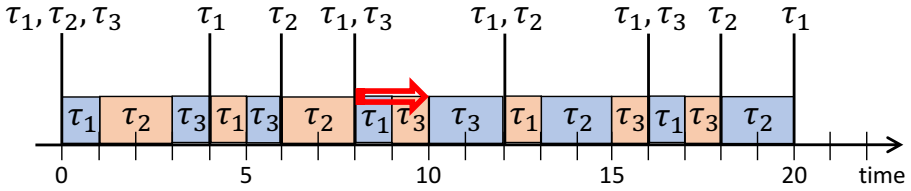

Figure 7: Schedule produced by RM for the task set from Task 4, including a deadline miss.

Task 5: Dynamic-priority Scheduling: Earliest Deadline First (EDF)

(a) Check if the task set from Task 4 in Table 4 is schedulable under EDF scheduling.

EDF assigns dynamic priorities to tasks during execution, so that tasks with earlier deadline have higher priority. EDF is preemptive and optimal; that is, no other algorithm can schedule a set of periodic real-time tasks if it cannot be scheduled by EDF.

For a set of n periodic tasks, the necesary and sufficient schedulability test is

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

For the task set given in Table 4, we have $U = 1/4 + 2/6 + 3/8 \approx 0.958 \leq 1$. Thus, the EDF schedule meets all deadlines.

(b) Construct the schedule using EDF for the interval $[0, 20]$. Identify deadline misses if they exist.

The schedule produced by EDF in the interval $[0, 20]$ is shown in Figure 8. Different from the schedule produced by RM shown in Figure 7, EDF meets all deadlines. Note that, for example, at time $t = 4$ when the second instance of task τ_1 arrives, the running task τ_3 is *not* preempted, because both have the same absolute deadline ($d_{1,2} = d_{3,1} = 8$) and hence the same priority.

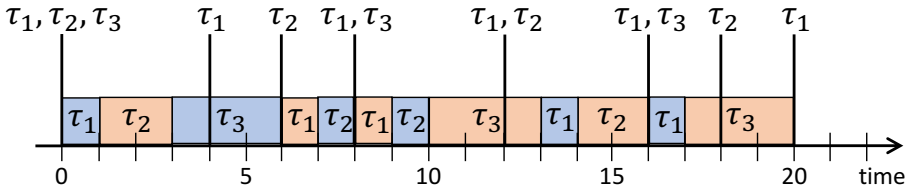

Figure 8: Feasible schedule produced by EDF for the task set from Task 4.

(c) Assume the necessary and sufficient EDF schedulability test fails for a task set with given execution times C_i and periods $T_i = D_i$, so there are definitely deadline misses expected when scheduling this task set using EDF. Does this imply that always the same task(s) will miss their deadline?

Schedulability test based on execution times and periods does not say which tasks are going to miss their deadline. Depending on their arrival times (i.e., phases), different tasks may miss deadlines.