EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science

Examination Real-time Architectures (2IN20)
on Friday, August 13, 14.00h-17.00h.

First read the entire examination. There are 4 exercises in total. Grades are included between parentheses at all parts and sum up to 11 points. You may use slides, papers and books for reference purposes. Good luck!

1. Following are some statements which you are expected to comment on, confirming or rejecting them.

   (a) (0.5) In order to synchronize clocks in a distributed system to an accuracy of 0.1 milliseconds, it is necessary to exchange at least 20000 messages per second.

   Answer: The necessary exchange frequency depends on the *drift* of the clocks, and a small drift allows a low exchange frequency. The statement is therefore rejected.

   (b) (0.5) The priority of a task in RMS is determined by the criticality level of that task.

   Answer: The priority of tasks in RMS is determined by the *periods* of the tasks, not their criticality level. In order to take the criticality level into account, one has to change, i.e. transform, the periods. The statement is therefore rejected.

   (c) (0.5) Preemptive scheduling techniques can be applied to all types of resources.

   Answer: Resources, such as buses, memory, and co-processors, are typically not preemptive. The statement is therefore rejected.

2. A real-time system has to monitor an environment in which critical conditions X and Y can occur. These conditions may exist for at most x and y time units respectively. The conditions come into effect instantaneously, and they remain in effect until they are handled. Once handled, they may come into effect again immediately. The system architect considers two alternatives to deal with this.

   (a) The use of sensor devices that generate an event when the condition occurs. This event gives an overhead of e units (i.e., switching control and executing the interrupt handler takes e units) and handling it takes h units.

   (b) The use of sensor devices that just allow observing the conditions X and Y. Observing the state as well as recording it takes also e units; handling the critical condition if it occurs is the same as in the previous case and takes h units.

   Answer the following questions.

(a) (1) The architect knows about polling tasks. Determine periods and deadlines for periodic tasks to solve case (b).

Answer: The problem is equivalent to that of the water vessel (see lecture *Real-Time Scheduling*), and we can therefore reuse its solution. Let task τ_x and a period T_x and a deadline D_x, and task τ_y have a period T_y and a deadline D_y. We now get: $T_x + D_x < x$ and $T_y + D_y < y$. For $T_x = D_x$ and $T_y = D_y$, this yields $T_x < \frac{x}{2}$ and $T_y < \frac{y}{2}$.

(b) (2) Adopt the following numbers: $e = 1, h = 4, x = 20, y = 32$. Introduce tasks for case (a) as well. Discuss the two cases with respect to the scheduling policies EDF and RMS (e.g., explain which one is most appropriate). Draw timelines to explain and validate your answers.

Answer: We assume that the event is entirely handled by the task, i.e. $C_x = C_y = e + h = 5$. The *minimal* (or lowerbound for the) interarrival time of both events is therefore 5, i.e. the time needed to process the event. For case (a), the deadlines of tasks τ_x and τ_y are $D_x = x = 20$ and $D_y = y = 32$, respectively.

For case (a), i.e. the event-triggered alternative, EDF is the most appropriate scheduling policy, and it is 'unnatural' to use RMS.

> *Motivation* (optional): Firstly, we observe that the deadlines are fixed, but the interarrival time is only characterized by a lower bound. RMS assumes deadlines and periods to be equal. Next, although the *minimal* interarrival time of the events is known, it is not obvious which static priority assignment to use for RMS. Assume that τ_x receives a higher priority than τ_y, and that both critical conditions X and Y become into effect simultaneously. A 'burst' of $n_x > (D_y - C_y)/C_x = \frac{27}{5}$ critical conditions X will cause τ_y to miss its deadline. A similar argument holds when τ_y receives a higher priority than τ_x. Hence, the only way out would be to give both tasks the same priority, and assume a first-come-first-serve implementation of the ready-queue. On the other hand, the event-triggered approach fits quite naturally for EDF.

Because EDF assigns priorities dynamically based on absolute deadlines, neither of the tasks can preempt the other task long enough to cause a deadline miss. E.g. task τ_x can delay the execution of τ_y for at most $\lfloor (D_y - D_x)/C_x \rfloor + 1 = 3$ times, and because $D_y - C_y - 3C_x = 32 - 5 - 15 \geq 0$, τ_y will never miss its deadline; see Figure 1. Hence, EDF prevents overloads caused by bursts of critical conditions.

For case (b), i.e. the time-triggered alternative, RMS is the most 'natural' scheduling policy, although EDF will work equally well. From Figure 2, we derive that tasks are schedulable under RMS. Note that the deadlines for handling the critical condition, i.e. x and y, are much larger than the deadlines $D_x = T_x$ and $D_y = T_y$ of the tasks.

(c) (1) Determine, if possible, utilizations in the previous case. What are utilizations if in addition it is given that there will be at least 100 units between occurrences of critical conditions?

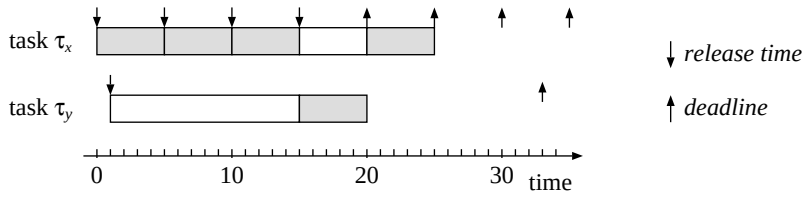

Figure 1: Timeline for tasks τ_x and τ_y under EDF, with a burst of critical conditions X.

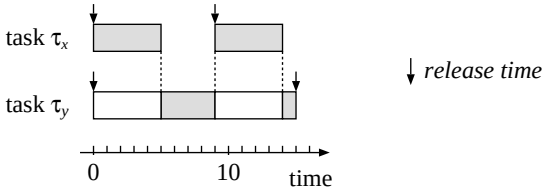

Figure 2: Timeline for τ_x and τ_y under RMS, with a critical instant at time 0. Task τ_y just meets its deadline $D_y = T_y$ for $T_y = 15$.

Answer: For case (a), let the critical conditions come into effect again immediately after being handled. In such a situation, the tasks will occupy the processor completely, causing a processor utilization of 1. For case (b), assume $T_x = 9$ and $T_y = 15$. In that case, tasks τ_x and τ_y can utilize the processor for at most $U_x^{upb} = \frac{C_x}{T_x} = \frac{5}{9}$ and $U_y^{upb} = \frac{C_y}{T_y} = \frac{5}{15}$, respectively. Together, they can utilize the processor for at most $U^{upb} = U_x^{upb} + U_y^{upb} \approx 0.89$.

Now assume that the critical conditions become into effect at most once every 100 units of time. For case (a), both tasks are activated at most once every 100 units of time, yielding an upperbound for the total utilization of $2(e + h)/100 = 1/10$.

For case (b), assume $T_x = 9$ and $T_y = 15$. For task τ_x we get at most $\left\lceil \frac{100}{T_x} \right\rceil = 12$ activations during 100 units, of which at most one activation finds a critical condition X. Hence, an upperbound for the utilization U_x^{upb} of task τ_x is given by $U_x^{upb} = \frac{12e+h}{12T_x} = \frac{16}{108} = \frac{4}{27}$. Similarly, we find an upperbound U_y^{upb} for task τ_y: $U_x^{upb} = \frac{8e+h}{8T_y} = \frac{12}{112} = \frac{3}{28}$. The upperbound for the total utilization is $U_x^{upb} + U_y^{upb} \approx 0.26$.

3. Consider the following three tasksets, given as sets of (period, computation time, deadline).

   (a) $\{(8, 3, 8), (9, 3, 9), (15, 3, 15)\}$

   (b) $\{(8, 4, 8), (12, 4, 12), (20, 4, 20)\}$

   (c) $\{(8, 4, 8), (10, 2, 10), (12, 3, 12)\}$

   Phasings are all 0. Answer the following questions.

   (a) (1) Which of the tasksets are schedulable by RMS? Motivate your answer.

   Answer: First, we determine the utilizations U_i of the three tasksets Γ_1, Γ_2, and

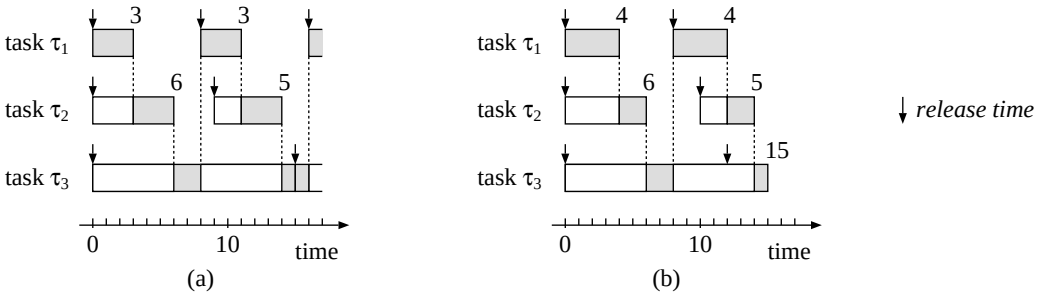

Figure 3: Timeline for Γ_1 (a) and Γ_3 (b) under RMS, with a critical instant at time 0.

Γ_3: $U_1 = \frac{3}{8} + \frac{3}{9} + \frac{3}{15} = \frac{3 \cdot 45 + 3 \cdot 40 + 3 \cdot 24}{360} = \frac{109}{120}$, $U_2 = \frac{4}{8} + \frac{4}{12} + \frac{4}{20} = \frac{4 \cdot 15 + 4 \cdot 10 + 4 \cdot 6}{120} = \frac{31}{30}$, $U_3 = \frac{4}{8} + \frac{2}{10} + \frac{3}{12} = \frac{4 \cdot 15 + 2 \cdot 12 + 3 \cdot 10}{120} = \frac{19}{20}$.

Because $U_2 > 1$, Γ_2 is not schedulable. For the other two tasksets, we now compare their utilization with the Liu and Layland bound $\mathrm{LL}(n) = n(2^{1/n} - 1)$, where $n = 3$ for both tasksets, i.e. $\mathrm{LL}(3) = 3(2^{1/3} - 1) \approx 0.78$. Because both U_1 and U_3 are larger than $\mathrm{LL}(3)$, we have to use other means to determine schedulability.

From Figure 3, we immediately see that Γ_1 is schedulable, whereas Γ_3 is not (because τ_3 misses its deadline).

(b) (1) For RMS, give an upperbound for the response jitter of the third task in the above sets, but only for the sets that are schedulable.

Answer: Only Γ_1 is schedulable, hence we only need to consider that taskset. In order to determine an upperbound for the release jitter RJ_3, we apply the $RJ_3 \leq WR_3 - BR_3$. From Figure 3 (a), we derive $WR_3 = 15$. We determine the best-case response time for τ_3 by means of the iterative procedure, although it is equally well valid to use a timeline.

The iterative procedure stops after four iterations, yielding a value of 3, i.e. $BR_3^{(0)} = WR_3 = 15$, $BR_3^{(1)} = C_3 + \sum_{j<3} \left(\left\lceil \frac{BR_3^{(0)}}{T_j} \right\rceil - 1 \right) C_j = 3 + \left(\left\lceil \frac{15}{8} \right\rceil - 1 \right) 3 + \left(\left\lceil \frac{15}{9} \right\rceil - 1 \right) 3 = 9$, $BR_3^{(2)} = 6$, $BR_3^{(3)} = 3$, and $BR_3^{(4)} = 3$.

Hence, $RJ_3 \leq 12$.

(c) (1) For taskset (c), show through a timeline the first time that there is slack (i.e., idle processing) in EDF.

Answer: From Figure 4, we see that the first time that there is slack is at time 46.

4. Consider the following taskset.

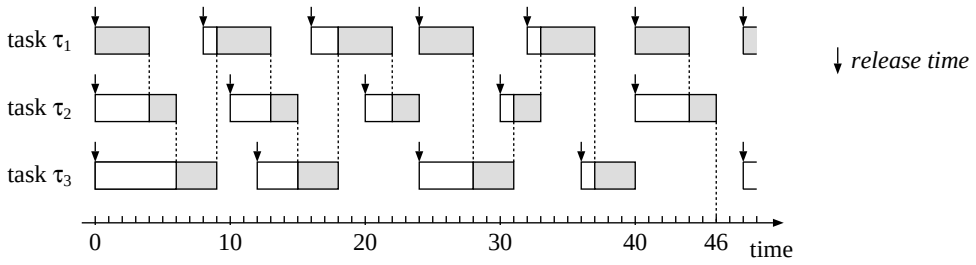

Figure 4: Timeline for Γ_3 under EDF.

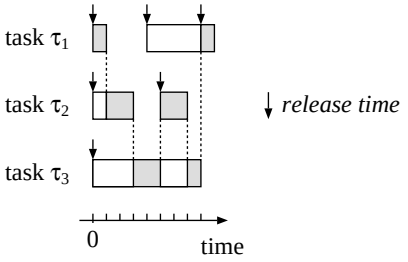

Figure 5: Timeline for Γ_3 under RMS.

Name	Priority	Period = deadline	Computation time	Resources
τ_1	1	4	1	Ra, Rb
τ_2	2	5	2	Rb
τ_3	3	11	3	Ra

The priorities are fixed with lower numbers representing higher priorities. The phasings are 0. The tasks need resources for their entire computation times that they reserve through a regular locking mechanism that is not of our concern here. For example, task τ_1 needs resources Ra and Rb for its operations and acquires them in that order.

(a) (1) Show through a timeline that priority inversion leads to a deadline miss. When does the miss occur?

Answer: Assuming RMS, a timeline is shown in Figure 5. Because task τ_3 acquired Ra, it can only be preempted by τ_1 for an amount of time to determine that Ra is in use at time 4. This checking time is neglected in Figure 5. However, task τ_3 *can* be preempted by τ_2, causing a deadline miss of τ_1 at time 8.

(b) (0.5) Show also that priority inheritance removes this particular instant of a miss. Give an argument why this might not be enough to resolve deadline misses entirely.

Answer: With priority inheritance, the priority of τ_3 is raised to the priority of τ_1, inhibiting τ_2 to preempt τ_3 at time 5. As a result, τ_3 completes at time 6. Task τ_1 can subsequently run, and completes at time 7. Hence, τ_1 does not miss its deadline at time 8.

With priority inheritance, the worst-case blocking time time B_1 of τ_1 is $C_2 + C_3 = 5$. Because $B_1 + C_1 = 6 > T_1 = 4$, τ_1 may still miss its deadline under worst-case conditions.

(c) (1) Is a deadline miss of τ_1 possible under priority ceiling? What about τ_2? What type of blocking plays a role in that case?

Answer: With priority ceiling, the worst-case blocking time B_1 for τ_1 is $max(C_2, C_3) = 3$. As a result, $B_1 + C_1 \leq T_1$, and τ_1 will therefore not miss a deadline.

The worst-case blocking time B_2 of task τ_2 is $C_3 = 3$. This type of blocking is called *push-through blocking*. Next to a worst-case blocking time B_2, task τ_2 can be preempted by task τ_1. The combination of blocking and preemption may cause τ_2 to miss its deadline. As an example, let task τ_1 and task τ_2 be released simultaneously at time 0, and task τ_3 be released an infinitsimal ε before that simultaneous release. In such a situation, τ_3 is allowed to execute first, immediately followed by task τ_1, causing an total interference for task τ_2 of 4. Because task τ_2 has a computation time of 2, it will miss its deadline at time 5.