EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science

Examination Real-time Architectures (2IN20)
on Wednesday, June 23, 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. A schedule can be constructed using methods like rate monotonic scheduling and earliest deadline first. Indicate whether the following statements are true and explain your answer briefly.

   (a) (0.5) With EDF a higher utilization is possible than with RMS.
   Answer. EDF has a maximum utilization of 100%; for RMS there are tasksets with a lower utilization that are not feasible. Hence, the statement is true, though the actual utilization depends on the taskset. The argument that RMS has a lower utilization bound is not acceptable as such, since this bound is a pessimistic one.

   (b) (0.5) RMS is more efficient than EDF.
   Answer. Efficiency has many dimensions. Good arguments are:
   - When used on-line, EDF needs more advanced data structures (logarithmic access). RMS is then more efficient.
   - EDF assigns different priorities to jobs and therefore needs dynamic priority assigment which is generally more costly. Again, RMS is then more efficient.
   - EDF can have a higher utilization. It makes more efficient use of the resources.

   (c) (0.5) RMS is more general than EDF (i.e., applicable in more situations).
   Answer.
   - RMS admits to separate importance and schedulability in a systematic way.
   - RMS performs more predictable under overload conditions.

   An argument against it would be to refer to the utilization once again.

   (d) (0.5) Explain the difference between deadline monotonic scheduling and earliest deadline first.
   Answer. The first is a fixed priority policy that refers to the relative deadline. The second is a dynamic priority policy that refers to the absolute deadline.

2. Consider the following taskset.

Name	Period $T_j (= D_j)$	Computation time C_j	Phasing
τ_1	4	2	0
τ_2	6	1	0
τ_3	12	1	0

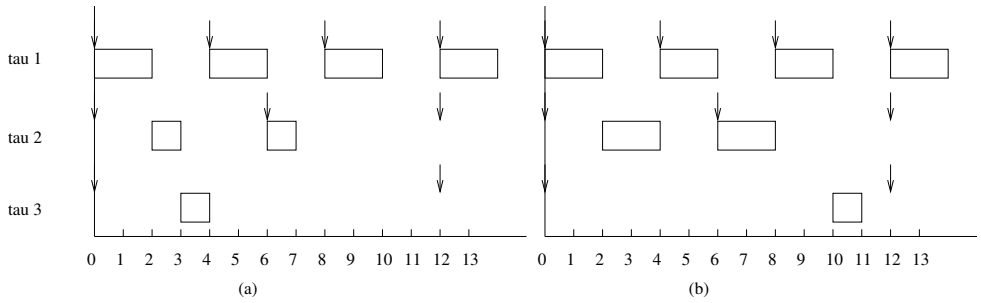

Figure 1: *Timelines for exercise 2(a) and 2(b). Timelines for EDF and RMS are identical. The hyperperiod is 12 (=lcm(4,6,12)) hence it suffices to consider just this part.*

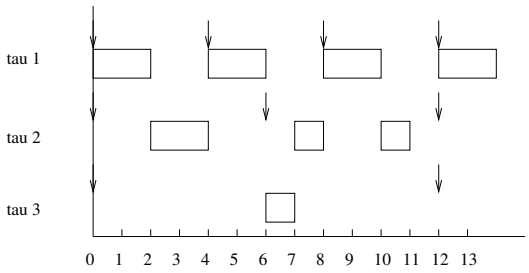

Figure 2: *Alternate timeline for EDF in exercise 2(b).*

In the following, motivate your answer, when necessary through a computation.

(a) (1) Is this taskset schedulable with RMS? With EDF? Draw timelines in both cases.
Answer. See Figure 1(a). The schedulability follows from the existence of the timing diagram. Alternatively, it follows from the utilization that falls under the Liu & Layland bound.

(b) (1) The same question if $C_2 = 2$.
Answer. See Figure 1(b). For EDF there are more options now, one of them is given in Figure 2. Notice that in Figure 1 there is no preemption needed while in Figure 2 there is indeed preemption. Notice also that this preemption is not necessary; there are options without preemption (in fact, the RMS schedule is also possible with EDF).

(c) (1) Determine a phasing and a value for C_2 larger than 2 such that RMS still yields a feasible schedule. What is the utilization?
Answer. Regardless of the phasing, the maximum value for C_2 is 2.5, as this leads to a utilization of 100%. Putting a value of 2.5 without changing the phasing leads to a deadline miss for τ_2 at t=6 (see Figure 1(b)). A phasing of 0.5 resolves this problem. Schedulability follows from Figure 3(a).

(d) (1) Assume that τ_2 generates the output of the system and assume again that $C_2 = 2$. What is the output jitter? How can this jitter be removed?
Answer. The output jitter is the maximum difference in response times. According

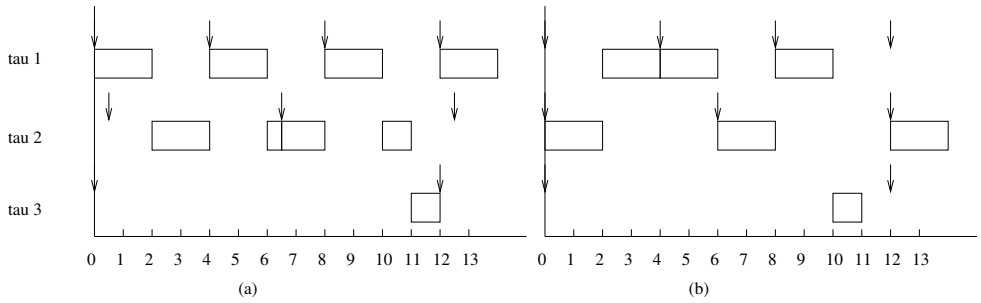

Figure 3: *Left: RMS timeline with the phasing of τ_2 set to 0.5 and C_2=2.5. Right: timeline with the parameters of (b) and with τ_2 having the highest priority.*

to Figure 1(b) this is 2 (reponse times are 4 for the first instance and 2 for the second). It can be removed by giving τ_2 the highest priority. This is no longer an RMS priority assignment but just a fixed priority assignment. According to Figure 3(b) this solution is feasible.

3. Consider the following taskset.

Name	Priority	Computation time	Resources
τ_1	1	C_1	Ra, Rb
τ_2	2	C_2	none
τ_3	3	C_3	Ra
τ_4	4	C_4	Rb

The priorities are fixed with lower numbers representing higher priorities. The tasks need resources 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) Discuss scenarios for priority inversion. Is unbounded inversion possible?
Answer. We mention two characteristic scenario's.

- τ_3 reserves Ra, τ_1 blocks on Ra, τ_2 preempts for C_2 seconds.
- τ_4 reserves Rb, τ_1 blocks on Rb, τ_2 preempts for C_2 seconds.
- Not possible is: τ_4 reserves Rb, τ_1 blocks on Rb, τ_3 preempts and executes for C_3 seconds. This is because while τ_1 is blocked on Rb τ_1 also has reserved Ra.

Against unbounded inversion there are two arguments. First, if this is a schedulable set then at some point in time the lowest priority job will be able to proceed (otherwise, the set would have a utilization above 1). Second, assume that this just is a set of prioritized tasks without real-time guarantees. One form of unbounded inversion in the case of the above scenario's is that τ_2 never gives up the processor. If we disregard this as an unrealistic case then the only other situation is that two or

more tasks alternate to block access of the low priority task. However, as mentioned above τ_2 and τ_3 cannot cooperate to block τ_4.

(b) (1) Assume that we use the priority inheritance protocol to resolve this. What is the maximum blocking time for τ_1 expressed in the given computation times? And what is this value if we use the priority ceiling protocol?
Answer. In priority inheritance τ_1 has to wait for at most the completion of a critical section per resource. This amounts to a worst-case waiting period of $C_3 + C_4$ seconds. In the priority ceiling protocol τ_1 has to wait for at most one critical section to complete. This yields the maximum of C_1 and C_2 as the worst-case waiting time.

4. Two processors p and q have their own clocks. The times returned by these clocks on world time t are denoted by functions $C_p(t)$ and $C_q(t)$ respectively. The drift of both clocks is bounded by $\rho \ll 1$, i.e.,

$$(1-\rho)(t_2 - t_1) < C_i(t_2) - C_i(t_1) < (1+\rho)(t_2 - t_1)$$

for $t_2 > t_1$ and $i = p, q$.

Processor p synchronizes its clock with that of processor q as follows.

(a) at time t_1, p sends a message to q that arrives after precisely δ seconds, at time t_2.

(b) at time t after receiving the message, q reads its clock. The value is sent back at time t_3. It arrives at p at time t_4, exactly δ seconds later.

(c) It is given that t is precisely in the middle of t_2 and t_3. The difference $t_3 - t$ is called d.

Answer the following questions.

(a) (0.5) Draw a diagram.
Answer. See Figure 4.

(b) (1) Express $C_p(t) - C_q(t)$ in terms of $C_q(t), C_p(t_4), \rho, d$ and δ by giving upper and lower bounds.
Answer. We only have to get rid of the term $C_p(t)$.

$$\begin{aligned} & C_p(t) - C_q(t) \\ = & \\ & C_p(t) - C_p(t_4) + C_p(t_4) - C_q(t) \end{aligned}$$

Hence, using the time bound for C_p at times t and t_4 (multiply by -1), viz.

$$(1+\rho)(t - t_4) < C_p(t) - C_p(t_4) < (1-\rho)(t - t_4)$$

we obtain

$$\begin{aligned} & (1+\rho)(t-t_4) + C_p(t_4) - C_q(t) < C_p(t) - C_q(t) < \\ & \quad (1-\rho)(t-t_4) + C_p(t_4) - C_q(t) \\ = & \quad \{t - t_4 = -(\delta + d)\} \\ & -(1+\rho)(\delta+d) + C_p(t_4) - C_q(t) < C_p(t) - C_q(t) \\ & \quad < -(1+\rho)(\delta+d) + C_p(t_4) - C_q(t) \end{aligned}$$

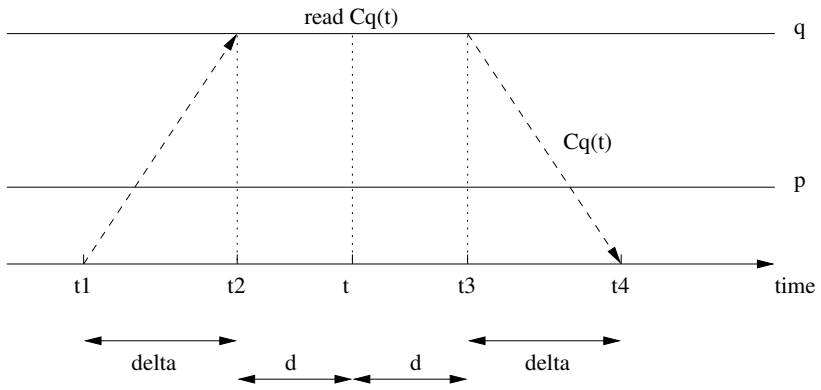

Figure 4: *Timing diagram for exercise 4. Clocks at p and q are inspected at given points in real time.*

(c) (1.5) What are upper and lower bounds for $C_p(t_4) - C_q(t_4)$ expressed in $C_p(t_1), C_p(t_4), C_q(t)$ and ρ? Discuss the relevance of this result.
Answer. Now we need to get rid of $C_q(t_4)$ in a similar way.

$$\begin{aligned} & C_p(t_4) - C_q(t_4) \\ = & \\ & C_p(t_4) - C_q(t) + C_q(t) - C_q(t_4) \end{aligned}$$

By defining $X = C_p(t_4) - C_q(t)$ we obtain in exactly the same way as above,

$$-(1+\rho)(\delta + d) + X < C_p(t_4) - C_q(t_4) < -(1-\rho)(\delta + d) + X$$

Next we need an estimate for $\delta + d$. From figure 4 we obtain

$$t_4 - t_1 = 2(\delta + d)$$

Hence,

$$2(1-\rho)(\delta + d) < C_p(t_4) - C_p(t_1) < 2(1+\rho)(\delta + d)$$

This gives, assuming $0 \leq \rho < 1$

$$(\delta + d) < \frac{C_p(t_4) - C_p(t_1)}{2(1-\rho)} \text{ and } \frac{C_p(t_4) - C_p(t_1)}{2(1+\rho)} < (\delta + d)$$

Combining these two,

$$-(1+\rho)\frac{C_p(t_4) - C_p(t_1)}{2(1-\rho)} + X < C_p(t_4) - C_q(t_4) < -(1-\rho)\frac{C_p(t_4) - C_p(t_1)}{2(1+\rho)} + X$$

Relevance: the results are formulated entirely using values known by p and it does not need to know the delays.