TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

# IN4343 – Real Time Systems

June 23rd 2016, from 13:30 to 16:30

Koen Langendoen

| Question: | 1 | 2 | 3 | 4 | 5 | 6 | Total |
|---|---|---|---|---|---|---|---|
| Points: | 15 | 15 | 15 | 25 | 10 | 10 | 90 |
| Score: | | | | | | | |

- This is a closed book exam
- You may use a **simple** calculator only (i.e. graphical calculators are not permitted)
- Write your answers with a black or blue pen, not with a pencil
- **Always justify your answers, unless stated otherwise**

- The exam covers the following material:
  (a) chapters 1-6, 8-9 of the book "Hard Real-Time Computing Systems (3rd ed)" by G. Buttazzo
  (b) the paper "The Worst-Case Execution-Time Problem" by Wilhelm et al. (except Section 6)
  (c) the paper "Transforming Execution-Time Boundable Code into Temporally Predictable Code" by P. Puschner
  (d) the paper "Best-case response times and jitter analysis of real-time tasks" by R.J. Bril, E.F.M. Steffens, and W.F.J. Verhaegh

| | |
|---|---|
| **Liu and Layland (LL)** bound | $U_{lub}^{RM} = n(2^{1/n} - 1)$ |
| **Hyperbolic (HB)** bound | $\prod_{i=1}^{n}(U_i + 1) \leq 2$ |
| **Response Time Analysis** | $WR_i = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{WR_i + AJ_k}{T_k} \right\rceil C_k$ |
| | $BR_i = C_i + \sum_{k=1}^{i-1} \left( \left\lceil \frac{BR_i - AJ_k}{T_k} \right\rceil - 1 \right)^+ C_k$ |
| | $w^+ = \max(w, 0)$ |
| **Processor Demand** | $g(t_1, t_2) = \sum_{r_i \geq t_1}^{d_i \leq t_2} C_i \qquad g(0, L) = \sum_{i=1}^{n} \left\lfloor \frac{L + T_i - D_i}{T_i} \right\rfloor C_i$ |
| schedulability | $\forall L \in D, \quad g(0, L) \leq L$ |
| | $D = \{d_k \mid d_k \leq \min(H, \max(D_{max}, L^*))\}$ |
| | $H = lcm(T_1, \ldots, T_n)$ |
| | $L^* = \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1 - U}$ |
| **Polling Server** schedulability | $U_{lub}^{RM+PS} = U_s + n \left[ \left( \frac{2}{U_s + 1} \right)^{1/n} - 1 \right]$ |
| | $\prod_{i=1}^{n}(U_i + 1) \leq \frac{2}{U_s + 1}$ |
| response time | $R_a = C_a + \Delta_a + F_a(T_s - C_s)$ |
| | $\Delta_a = \left\lceil \frac{r_a}{T_s} \right\rceil T_s - r_a \qquad F_a = \left\lceil \frac{C_a}{C_s} \right\rceil - 1$ |
| **Deferrable Server** schedulability | $U_{lub}^{RM+DS} = U_s + n \left[ \left( \frac{U_s + 2}{2U_s + 1} \right)^{1/n} - 1 \right]$ |
| | $\prod_{i=1}^{n}(U_i + 1) \leq \frac{U_s + 2}{2U_s + 1}$ |
| response time | $R_a = C_a + \Delta_a - C_0 + F_a(T_s - C_s)$ |
| | $C_0 = \min(C_s(r_a), \Delta_a)$ |
| | $\Delta_a = \left\lceil \frac{r_a}{T_s} \right\rceil T_s - r_a \qquad F_a = \left\lceil \frac{C_a - C_0}{C_s} \right\rceil - 1$ |
| **NP scheduling** level-i busy period | $L_i = B_i + \sum_{h=1}^{i} \left\lceil \frac{L_i}{T_h} \right\rceil C_h \qquad N_i = \left\lceil \frac{L_i}{T_i} \right\rceil$ |
| | $B_i = \max_{j>i}\{C_j\}$ |
| response time | $s_{ik} = B_i + (k-1)C_i + \sum_{h=1}^{i-1} \left( \left\lfloor \frac{s_{ik}}{T_h} \right\rfloor + 1 \right) C_h$ |
| | $R_{ik} = (s_{ik} + C_i) - (k-1)T_i$ |
| | $R_i = \max_{k \in [1, N_i]} \{R_{ik}\}$ |
| **Elastic Model** utilization | $\forall i \quad U_i = U_{i0} - (U_0 - U_d)\frac{E_i}{E_S} \qquad \text{where } E_S = \sum_{i=1}^{n} E_i$ |

## Question 1 [15 points]

**Bratley's algorithm** is part of the class of scheduling algorithms handling aperiodic tasks. It is a heuristic algorithm that prunes (large) parts of the search tree.

(a) $\boxed{\text{3 points}}$ Bratley's algorithm can be classified as $(1|no-preem|L_{max})$ according to Graham's notation. Explain what the classification denotes.

**Solution:** Uniprocessor algorithm for non-preemptive tasks that minimizes the maximum lateness.

(b) $\boxed{\text{4 points}}$ Explain how Bratley's algorithm works.

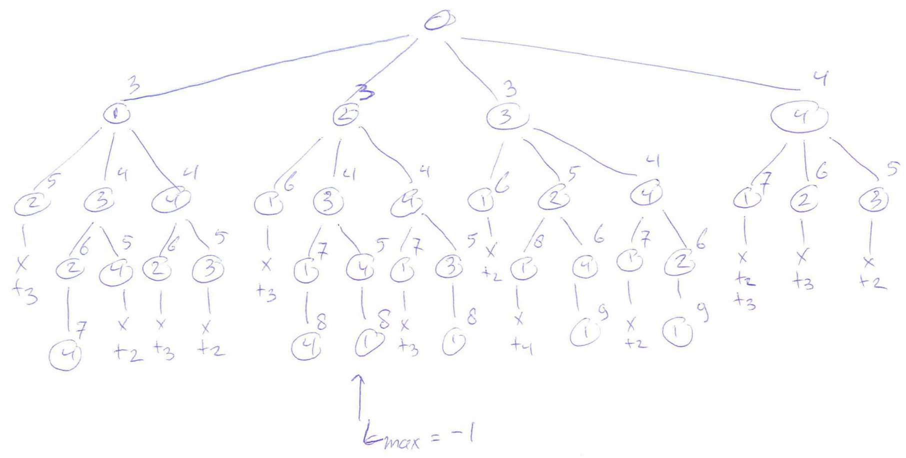
**Solution:** It expands the tree in depth-first order. A node, however, is only expanded when the partial schedule is found to be **strongly feasible**, i.e. if it remains feasible when adding any of the remaining nodes.

(c) $\boxed{\text{8 points}}$ Apply Bratley's algorithm to find the feasible schedules for the following task set:

| | $a_i$ | $C_i$ | $d_i$ |
|---|---|---|---|
| $\tau_1$ | 0 | 3 | 9 |
| $\tau_2$ | 1 | 2 | 6 |
| $\tau_3$ | 2 | 1 | 5 |
| $\tau_4$ | 3 | 1 | 8 |

Draw the resulting search tree, and determine the optimal $L_{max}$. Finally, compute the gain over expanding the tree fully.

**Solution:**



6 feasible schedules, with $\tau_2 - \tau_3 - \tau_4 - \tau_1$ achieving an $L_{max} = -1$. A fully expanded tree consists of $1 + 4 + 12 + 24 + 24 = 65$ nodes. Bratley only expands $1 + 4 + 12 + 12 + 6 = 35$ nodes, so the gain is $65/35 = 1.86$.

## Question 2 [15 points]

|  | $C_i$ | $T_i$ |
|---|---|---|
| $\tau_1$ | 2 | 5 |
| $\tau_2$ | 3 | 9 |
| $\tau_3$ | 3 | 15 |

(i)

|  | $C_i$ | $T_i$ |
|---|---|---|
| $\tau_1$ | 2 | 5 |
| $\tau_2$ | 3 | 9 |
| $\tau_3$ | 4 | 15 |

(ii)

|  | $C_i$ | $T_i$ |
|---|---|---|
| $\tau_1$ | 2 | 5 |
| $\tau_2$ | 3 | 9 |
| $\tau_3$ | 5 | 15 |

(iii)

(a) $\boxed{\text{3 points}}$ Which task sets are feasible under Earliest Deadline First scheduling?

**Solution:**

(i) Feasible as processor utilization $(42/45) \leq 1$

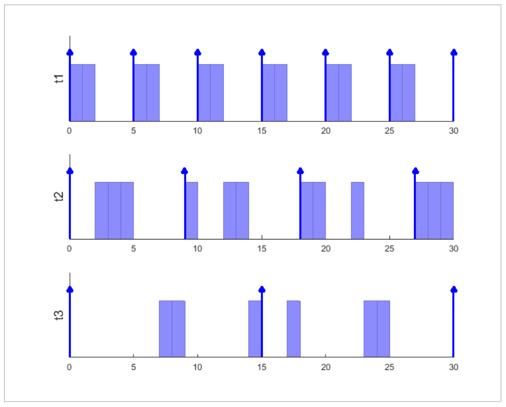
(ii) Feasible as processor utilization $(45/45) \leq 1$

(iii) Unfeasible as processor utilization $(48/45) > 1$

(b) $\boxed{\text{7 points}}$ Which task sets are feasible under Rate Monotonic scheduling?

**Solution:**

(i) Feasible



(ii) Unfeasible as (i) has no slack in the critical instant of task $\tau_3$

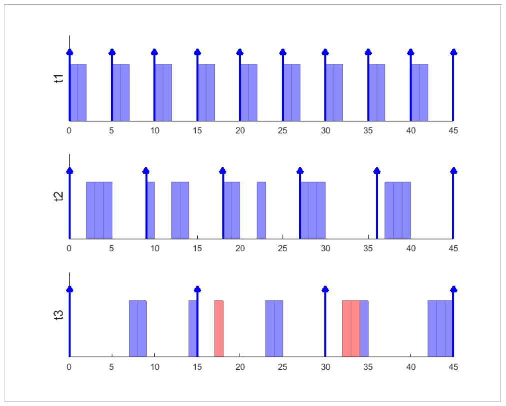
(iii) Unfeasible as EDF can't schedule it

(c) $\boxed{\text{5 points}}$ Determine the maximum lateness for the given task sets under RM scheduling. Hint: standard techniques may not apply to unfeasible task sets.

**Solution:**

(i) $L_{max} = 0$, see figure above

(ii) Task $\tau_3$ misses its first deadline, and continues running in the $2^{nd}$ period pushing that instance to the right. Thus simply looking at the critical instant is not enough. A safe bet is to check the completer hyper period, i.e. until t=45.



$L_{max} = 4$

(iii) As the system is overloaded $L_{max}$ grows unbounded with time.

## Question 3 [15 points]

| | $C_i$ | $D_i$ | $T_i$ |
|---|---|---|---|
| $\tau_1$ | 2 | 3 | 5 |
| $\tau_2$ | 4 | 6 | 9 |
| $\tau_3$ | 4 | 30 | 40 |

Deadlines complicate matters. For EDF for example, we may need to revert to the processor demand criterion to determine the feasibility of a schedule. The processor demand is defined as

$$g(0, L) = \sum_{i=1}^{n} \left\lfloor \frac{L + T_i - D_i}{T_i} \right\rfloor C_i$$

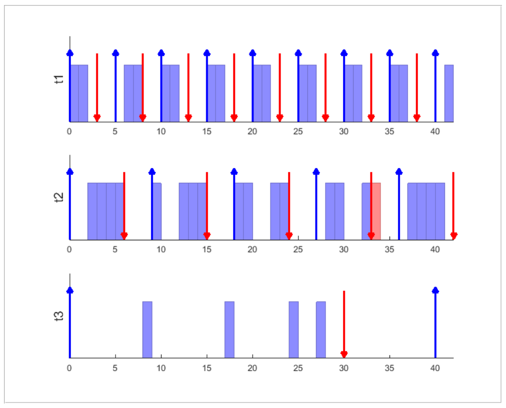
(a) $\boxed{\text{5 points}}$ Explain what $g(0, L)$ denotes, and why the term $T_i - D_i$ is part of the numerator.

**Solution:** $g(0, L)$ captures the amount of work that has to be executed by time $L$ when starting at time 0. When counting the number of instances of task $\tau_i$, we know that the demand goes up as soon as the next deadline is reached. This deadline occurs before the next period $T_i$, so we must bump the counter early by $T_i - D_i$. See the drawing in the lecture notes.

(b) $\boxed{\text{10 points}}$ Determine if the task set is feasible under EDF scheduling.

**Solution:** Processor utilization is $340/360 \leq 1$, so we need to check the processor demand until $min(H, max(D_{max}, L^*))$ with $H = 360$, $D_{max} = 30$, and $L^* = 56$, so until 56.

| | demand |
|---|---:|
| g(0,3) | 2 |
| g(0,6) | 6 |
| g(0,8) | 8 |
| g(0,13) | 10 |
| g(0,15) | 14 |
| g(0,18) | 16 |
| g(0,23) | 18 |
| g(0,24) | 22 |
| g(0,28) | 24 |
| g(0,30) | 28 |
| g(0,33) | 34 |



Task set is **un**feasible!

## Question 4 [25 points]

When mixing periodic and aperiodic tasks one can make use of a fixed priority server to schedule the aperiodic tasks. Consider the following periodic tasks and aperiodic jobs (under RM):

| | $C_i$ | $T_i$ |
|---|---|---|
| $\tau_1$ | 2 | 5 |
| $\tau_2$ | 2 | 10 |

| | $a_i$ | $C_i$ |
|---|---|---|
| $J_1$ | 3 | 3 |
| $J_2$ | 7 | 4 |

(a) $\boxed{\text{4 points}}$ Dimension both a polling and a deferrable server by means of the hyperbolic bound guaranteeing schedulability.

**Solution:** $P = \prod_{i=1}^{n}(U_i + 1) = \frac{7}{5} \cdot \frac{12}{10} = \frac{84}{50}$

Polling server: $U_s = \frac{2-P}{P} = \frac{4}{21}$, $< C_s = \frac{20}{21}, T_s = 5 >$

Deferrable server: $U_s = \frac{2-P}{2P-1} = \frac{8}{59}$, $< C_s = \frac{40}{59}, T_s = 5 >$

(b) $\boxed{\text{7 points}}$ Consider a server dimensioned with $<C_s = 2, T_s = 5>$. Determine if this configuration is feasible (i.e. no deadline violations) to be used for a polling server. Determine the same for a deferrable server.

**Solution:** Polling server: utilization is 1 (!), we need to check the critical instant. Task $\tau_1$ has a lateness of -1, task $\tau_2$ has a lateness of 0, hence, it is a feasible task set.

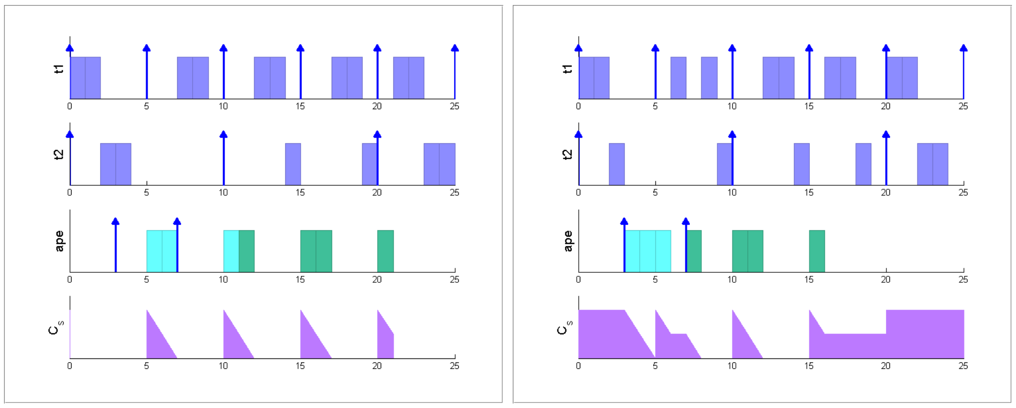
Deferrable server: utilization is 1 (!), we need to check the critical instant, **but** with jitter for the server ($AJ_S = 5 - 2 = 3$). As the schedule was already tight for the polling server, now tasks are getting into troubles ($R_1 = 6, R_2 = 20$) and violates their deadline. Thus the schedule is **un**feasible.

(c) $\boxed{\text{10 points}}$ Obtain the response times of jobs $J_1$ and $J_2$ for a polling server, as well as for a deferrable server when configured as $<C_s = 2, T_s = 5>$.

**Solution:** Response times

| | PS | DS |
|---|---|---|
| $J_1$ | 8 | 3 |
| $J_2$ | 14 (15*) | 9 |

*the equations in the book for PS ignore the case that spare capacity can be used when a job is blocked by a predecessor, see picture below, so the computed response time is too high.



(d) $\boxed{\text{4 points}}$ Determine the response times of the jobs when **slack stealing** would be used.

**Solution:** From the DS plot we can see that job $J_1$ is already optimal. Regarding job $J_2$ we can see that there is no slack until $t = 10$, but there is room in the 3rd period of task $\tau_1$ and the 2nd period of $\tau_2$. Shifting both 1 unit to the right allows job $J_2$ to finish by $t = 13$, reducing its response time to 6.

## Question 5 [10 points]

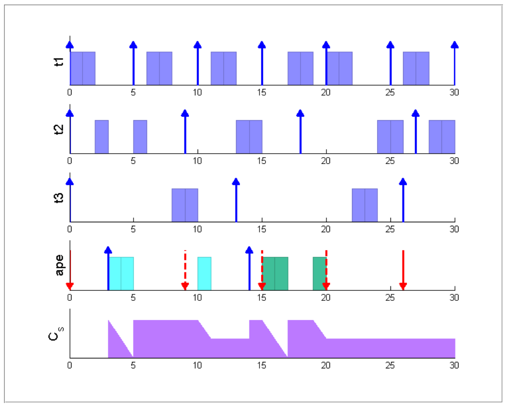
To protect periodic tasks from overruns by a-periodic jobs a Constant Bandwidth Server can be used in combination with an EDF scheduler.

| | $C_i$ | $T_i$ |
|---|---|---|
| $\tau_1$ | 2 | 5 |
| $\tau_2$ | 2 | 9 |
| $\tau_3$ | 2 | 13 |

| | $a_i$ | $C_i$ |
|---|---|---|
| $J_1$ | 3 | 3 |
| $J_2$ | 14 | 3 |

(a) $\boxed{\text{7 points}}$ Consider the case that the CBS server is dimensioned as $<C_S = 2, T_S = 6>$, determine the response times of jobs $J_1$ and $J_2$.

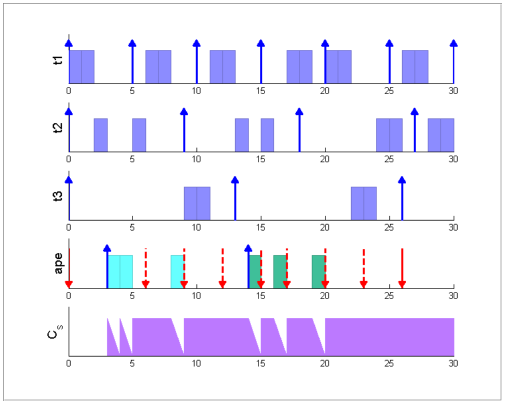
**Solution:**



$RJ_1 = 8, RJ_2 = 6$

(b) $\boxed{\text{3 points}}$ Argue the effect (if any) of keeping the server utilization constant, but halving its period (i.e. $C_S = 1, T_S = 3$).

**Solution:**
Things can only get better when the server period is reduced because there will be more intermediate deadlines (with in-between values) allowing shorter jobs to attain a higher priority (earlier deadline). Proof by example (not required for full score):



$RJ_1 = 6, RJ_2 = 6$

## Question 6 [10 points]

One approach to handle permanent overloads is to resort to value-based scheduling, in which the least important tasks are simply rejected. The performance of a scheduling algorithm $A$ can be evaluated through its cumulative value $\Gamma_A(T) < \Gamma_{max}(T) = \sum_i V_i$ for task set $T$ with $V_i$ denoting the value of task $i$.

(a) $\boxed{\text{4 points}}$ Show that, in case of overload, an **online** scheduling algorithm can not be optimal. That is, achieve the same performance as an offline (clairvoyant) algorithm in all cases.

**Solution:** Construct a counter example showing that knowledge of the future is required to decide to skip a task or not. See lecture notes for a concrete example.

(b) $\boxed{\text{6 points}}$ Explain what the **competitive factor** is, and demonstrate that for EDF this equals to zero.

**Solution:** The competitive factor $\phi$ captures how well (bad) a scheduling policy $A$ does in comparison to the optimal (offline) scheduler.
$\phi_a = min_T \frac{\Gamma_A(T)}{\Gamma_{maz}(T)}$
For EDF we can construct an example with just two tasks of value $\epsilon$ and 1, where the deadline of the former is earlier than the latter, and the unit task has no slack (deadline = computation time). See lecture notes.