TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

IN4343 – Real Time Systems

June 25th 2018, from 09:00 to 12:00

Koen Langendoen

Question:	1	2	3	4	5	6	7	Total
Points:	10	15	20	15	10	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

IN4343 Real-Time Systems *Cheat Sheet*

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 [10 points]

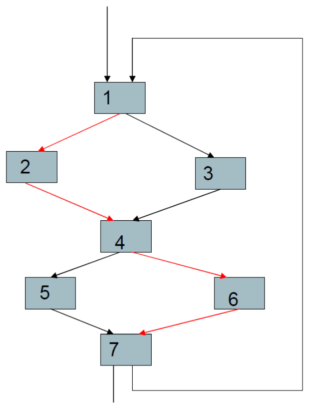
Determining the worst-case execution time of a task lies at the heart of real-time systems. (Wilhelm:2008) describes several methods, including Integer Linear Programming (ILP), for estimating the WCET of a task. ILP is formally defined as

$$\begin{aligned} \textbf{maximize} \quad & c^T x \\ \textbf{subject to} \quad & Ax \leq b, \\ & x \geq 0, \\ \textbf{and} \quad & x \in \mathbb{Z}^n \end{aligned}$$

(a) $\boxed{\text{4 points}}$ Describe how the WCET of a control-flow graph (program) can be instantiated as an ILP problem.

Solution: The ILP solver can be set to work to find the maximum execution path in the given control-flow graph (a set of connected basic blocks) as follows: The WCET is simply the sum-of-product of the number of times each basic block is executed (x_i) times the cost (c_i) counted in machine instructions (clock cycles), subject to constraints imposed by the control-flow graph. Example constraints are (i) the count of incoming executions must equal the outgoing count, and (ii) the maximum number of loop iterations.

(b) $\boxed{\text{6 points}}$



Apply your approach from (a) to the control-flow graph above, with the additional information that the maximum loop count is 97 and that the red path is infeasible.

Solution: First formulate the function to maximize: WCET $= \sum_{i=1}^{7} c_i \times x_i$. The basic set of constraints (in = out) is as follows:

$x_1 = 1 + x_{loop}$
$x_1 = x_2 + x_3$
$x_4 = x_2 + x_3$
$x_4 = x_5 + x_6$
$x_7 = x_5 + x_6$
$x_7 = 1 + x_{loop}$
$x_{loop} \leq 97$

(Note that we have introduced an extra ILP variable to keep track of the number of loops to avoid copy&pasting the constant 97 twice.) The infeasible path can be expressed as:

$x_6 \leq x_3$

Question 2 [15 points]

Given the following set of aperiodic, preemptable tasks

	A	B	C	D	E	F
r_i	0	7	3	1	6	5
C_i	4	1	1	2	2	4
d_i	11	12	7	9	10	14

with precedent constraints:

A→B, A→D, B→E, B→F, C→B, C→D, D→E. D→F,

To determine if there is a feasible schedule EDF* can be employed, which transforms the precedence constraints into timing constraints by updating the release times and deadlines of the tasks.

The idea behind EDF* is that for precedence constraint $P \rightarrow Q$ the release time of Q must be set after the earliest time that P can finish, and the deadline of P must be set before the latest time that Q can start:

$$r_Q^* = r_P + C_P \tag{1}$$
$$d_P^* = d_Q - C_Q \tag{2}$$

(a) $\boxed{\text{2 points}}$ Argue that Equation 1 can be simplified to $r_Q^* = r_P$ **without** compromising the precedence relation $P \rightarrow Q$.

Solution: When EDF has the option of scheduling both P and Q it will pick P as it has a smaller deadline due to Equation 2.

(b) $\boxed{\text{3 points}}$ Argue that simplifying instead Equation 2 to $d_P^* = d_Q$ **could** compromise the precedence relation $P \rightarrow Q$.

Solution: When task P is released EDF might still be busy with running another task (say R) that has an earlier deadline than P. Once R completes after t time units, P may be scheduled. However at the time task Q is released P can not have completed its computation as it was delayed by t units. Thus for those t units EDF has the freedom to select either P (fine) or Q (bad) as their deadlines are equal. (Note that if the time t is larger than the computation time of P, EDF could even start executing Q first!)

(c) $\boxed{\text{3 points}}$ Transform the release times according to the precedence constraints using the simplification from (a) above.

Solution: Apply the max rule starting at the roots.

	A	B	C	D	E	F
r_i^*	0	7	3	3	7	7

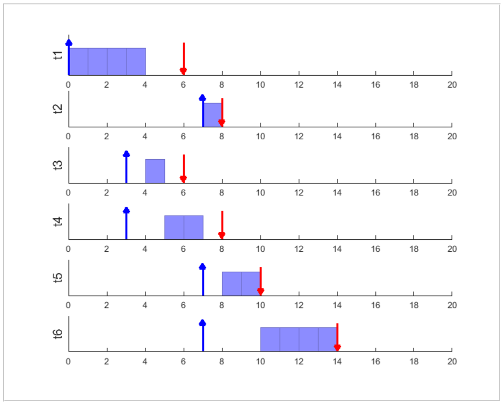
(d) $\boxed{\text{3 points}}$ Transform the deadlines according to the precedence constraints.

Solution: Apply the min rule starting at the leaves.

	A	B	C	D	E	F
d_i^*	6	8	6	8	10	14

(e) $\boxed{\text{4 points}}$ Determine the resulting response times of EDF* and wether or not the task set is feasible.

Solution: The schedule is shown below.



As all tasks meet their (transformed) deadlines the task set is feasible. With regards to the response times please note that the picture above shows the transformed release times while the tasks' response times must be computed using the original release times, resulting in the following:

	A	B	C	D	E	F
R_i	4	1	2	6	4	9

Question 3 [20 points]

Consider the following task sets under RM scheduling:

(i)

	C_i	T_i
τ_1	1	4
τ_2	2	6
τ_3	2	12
τ_4	1	35

(ii)

	C_i	T_i
τ_1	1	5
τ_2	2	6
τ_3	2	12
τ_4	1	35

(iii)

	C_i	T_i
τ_1	2	5
τ_2	2	6
τ_3	2	12
τ_4	1	35

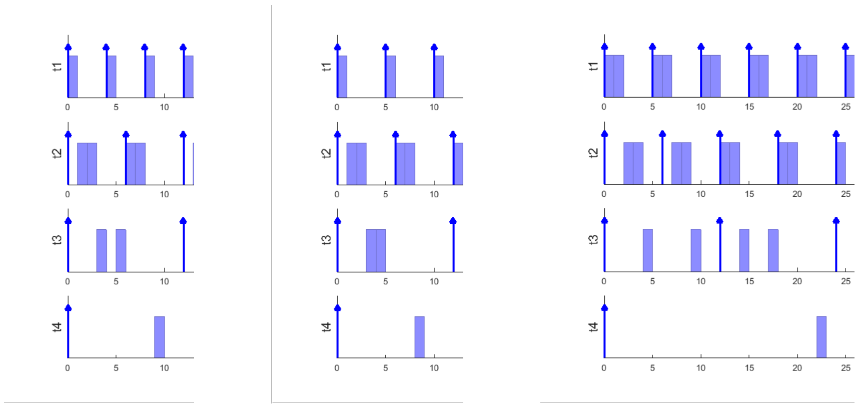
(a) $\boxed{\text{10 points}}$ Compute the LL and HB bounds for the three task sets. What can you say about the feasibility in each case?

Solution:

	$U_p < 0.76 = LL$	$HB \leq 2$	Feasibility
(i)	0.78	2	yes
(ii)	0.73	(1.92)	yes
(iii)	0.93	2.24	**maybe**

(b) $\boxed{\text{10 points}}$ Determine the maximum lateness (L_{max}) of each task set. What can you say about the feasibility in each case?

Solution: To determine L_{max} we must know the worst-case response times of the individual tasks, which can be obtained by drawing the critical instant.



	L_{max}
(i)	-3
(ii)	-3
(iii)	-2

Since L_{max} is negative for each task set, they are all feasible.

Question 4 [15 points]

Consider the following task set:

	C_i	D_i	T_i
τ_1	2	4	5
τ_2	2	5	7
τ_3	3	10	12

(a) $\boxed{\text{8 points}}$ Use the processor demand criterion to determine the feasibility of the above task set under EDF scheduling.

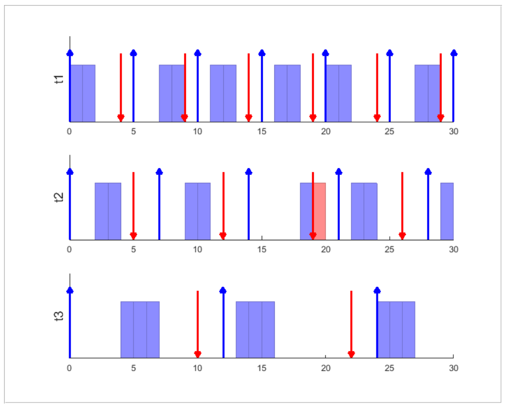
Solution: D$_{max}$ = 10, the hyperperiod = 420, and $L^* = \frac{206}{9} = 22.9$, so we need to check as far as L = 22.

	demand
g(0,4)	2
g(0,5)	4
g(0,9)	6
g(0,10)	9
g(0,12)	11
g(0,14)	13
g(0,19)	17
g(0,22)	20

No deadline violations, so task set is feasible under EDF.

(b) $\boxed{\text{7 points}}$ Now consider the same task set run in **non-preemptive** mode. Will that result in a feasible schedule under **EDF**?

Solution: No theory to help us here, so in principle we need to check the whole hyperperiod, i.e. up to 420 `:-(`



The time line shows a deadline violation at t=19, so the task set has become unfeasible.

Question 5 [10 points]

Consider the following task set under RM scheduling:

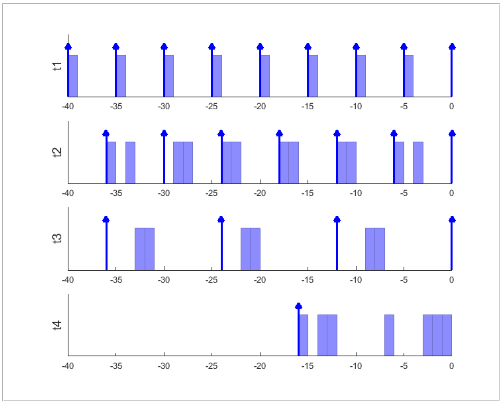
	C_i	T_i
τ_1	1	5
τ_2	2	6
τ_3	2	12
τ_4	7	35

(a) $\boxed{\text{2 points}}$ Compute the worst-case response time of task τ_4.

Solution: $\mathrm{WR}_4^0 = 12$, $\mathrm{WR}_4^1 = 7+3+4+2 = 16$, $\mathrm{WR}_4^2 = 7+4+6+4 = 21$, $\mathrm{WR}_4^3 = 7+5+8+4 = 24$, $\mathrm{WR}_4^4 = 7+5+8+4 = 24$. Thus the worst-case response time is 24.

(b) $\boxed{\text{4 points}}$ Draw the optimal instant for task τ_4 and report its best-case response time.

Solution:



The best-case response time is 16.

(c) $\boxed{\text{4 points}}$ Determine the upperbound for the response jitter of task τ_4 and tell wether or not this bound will change when task τ_1 incurs an activation jitter of $\frac{1}{2}$.

Solution: $\mathrm{RJ}_4 \leq \mathrm{WR}_4 - \mathrm{BR}_4 = 24 - 16 = 8$. Jitter can increase the worst-case, and decrease the best-case response time. The term $\left\lceil \frac{WR_4 + AJ_1}{T_1} \right\rceil = \left\lceil \frac{24+AJ_1}{5} \right\rceil$ is the same for $AJ_1 = 0$ as for $AJ_1 = \frac{1}{2}$ (the rounding would jump for $AJ_1 > 1$), so the worst-case response time is not affected. Likewise the best-case response time is also not affected as the term $\left\lceil \frac{BR_4 - AJ_1}{T_1} \right\rceil = \left\lceil \frac{16-AJ_1}{5} \right\rceil$ is the same for $AJ_1 = 0$ as for $AJ_1 = \frac{1}{2}$ (the rounding would jump for $AJ_1 \geq 1$). Thus the jitter bound stays the same.

Question 6 [10 points]

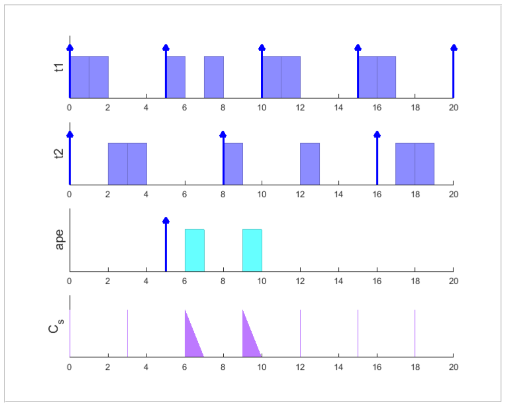
When mixing periodic and aperiodic tasks one can make use of a fixed priority server to schedule the aperiodic tasks. Consider the following periodic tasks and aperiodic job (under RM):

	C_i	T_i
τ_1	2	5
τ_2	2	8

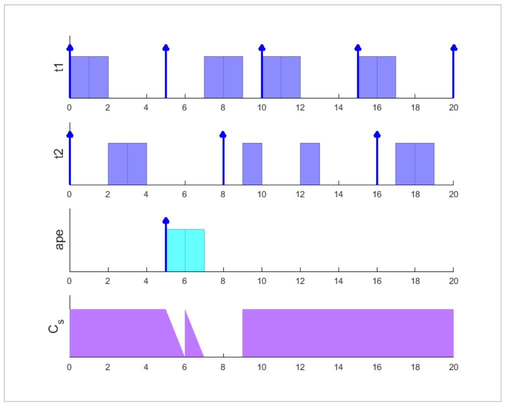
	a_i	C_i
J_1	5	2

(a) $\boxed{\text{4 points}}$ Determine the response time of job J_1 for both a Polling Server and a Deferable Server when these are dimensioned as $<C_s = 1, T_s = 3>$.

Solution:



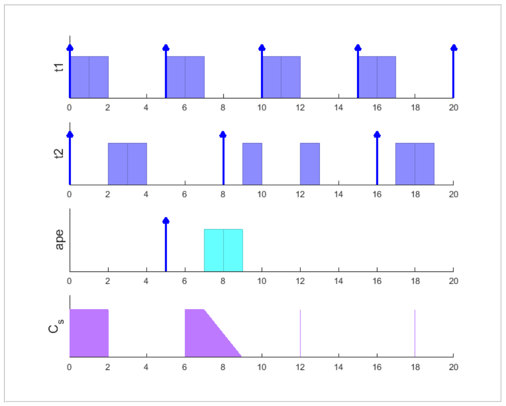
Polling server: response time = 5



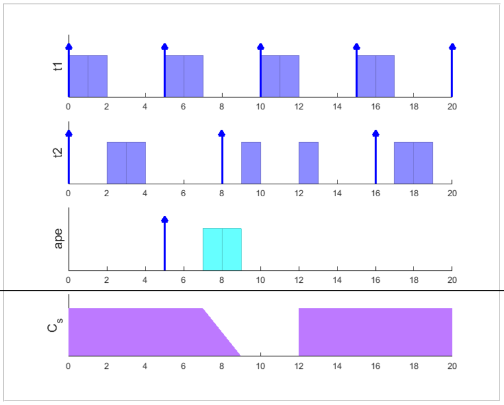
Deferable server: response time = 2

(b) $\boxed{\text{6 points}}$ Check if the response times degrade when we **lower** the priority of the Polling and Deferable server by dimensioning them as $<C_s = 2, T_s = 6>$. Explain your findings.

Solution:



Polling server: response time = 4



Deferable server: response time = 4

As we can see running the servers at a lower priority (by keeping the utilization constant, while scaling the capacity and period) can both degrade (DS) as well as improve (PS) the response time. This is due to two opposite effects: (1) by lengthening the server periods, task τ_1 now has higher priority introducing additional waiting (bad), but (2) the capacity has been increased, so job J_1 can now complete in a single run of the server and does not have to wait for the second run to come along (good).

Question 7 [10 points]

One approach to handle permanent overloads is to resort to value-based scheduling, in which the least important tasks are simply rejected. Consider the following task set:

	a_i	C_i	d_i	V_i
τ_1	4	3	9	7
τ_2	3	4	12	10
τ_3	1	5	13	5
τ_4	0	3	16	2
τ_5	2	4	17	3

(a) $\boxed{\text{4 points}}$ Describe how **robust EDF** functions.

Solution: Robust EDF maintains an additional reject queue. When an overload is detected, the least valued task(s) is placed in the reject queue. When the load drops below 1, then the highest valued task(s) in the reject queue that "fits" (i.e. new schedule is feasible) is put back into the ready queue.

(b) $\boxed{\text{6 points}}$ Apply robust EDF to the given task set and report what actions are taken at time $t = 4$, and at $t = 6$ when task τ_1 finishes early after computing for only 2 time units.

Solution: Short answer: At $t = 4$ robust EDF will put task τ_4 in the reject queue. At $t = 6$ robust EDF will check if it can revive task τ_4, but as it needs 2 more units to finish it cannot go back in the ready queue.