Tentamen 12 April 2018

Real-time Systems (Technische Universiteit Delft)

TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

IN4343 – Real Time Systems

April 12th 2018, from 09:00 to 12:00

Koen Langendoen

Question:	1	2	3	4	5	6	Total
Points:	10	15	20	20	10	15	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	$\forall i \quad U_i = U_{i0} - (U_0 - U_d)\frac{E_i}{E_S} \qquad$ 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).

(a) $\boxed{\text{4 points}}$ Succinctly describe how ILP functions.

Solution: Quoting Section 2.3.5 of (Wilhelm:2008) "Linear programming is a generic methodology to code the requirements of a system in the form of a system of linear constraints. In addition, a goal function that has to be maximized to obtain an optimal assignment of values to the system's variables is given. One speaks of integer linear programming if these values are required to be integers."

(b) $\boxed{\text{6 points}}$ Describe in which **two** ways ILP can be used to assist in WCET analysis.

Solution: To determine the WCET of a task one needs to know the longest path and the cost of it. ILP can be used for both. For the former, constraints from the control flow graph can be modeled as linear relations (sum of incoming executions equals sum of outgoing executions), and WCET $= C_i * X_i$ where C_i models cost, and X_i the number of executions (ILP variable). C_i can then be derived through ILP by modeling the micro-architecture in detail. For example, the effect of caches can be modeled by differentiating context (Always Hit, Always Miss, Persistent, etc.)

Question 2 [15 points]

Given the following set of aperiodic, preemptable tasks simultaneously released at t=0

	A	B	C	D	E	F
C_i	2	3	5	5	1	2
d_i	15	20	7	18	8	8

with precedent constraints:

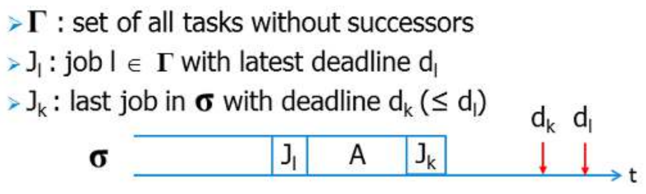
A→B, A→D, C→D, C→F, E→F.

(a) $\boxed{\text{5 points}}$ Derive the schedule produced by the Latest Deadline First (LDF) policy. Report the response times and lateness of each task.

Solution: C, E, F, A, D, B.

	A	B	C	D	E	F
R_i	10	18	5	15	6	8
L_i	-5	-2	-2	-3	-2	0

(b) $\boxed{\text{4 points}}$ The proof that LDF is optimal (i.e. minimizes the maximum lateness) when tasks with precedence constraints are activated synchronously (i.e. all at once) is based on an interchange argument using the following constellation:



- Γ : set of all tasks without successors
- J_l : job $l \in \Gamma$ with latest deadline d_l
- J_k : last job in σ with deadline d_k ($\leq d_l$)

Explain why job J_l and J_k cannot be simply swapped to make the schedule σ one step closer to an LDF schedule.

Solution: There might be dependencies from tasks within A towards job J_k that must be preserved.

(c) $\boxed{\text{6 points}}$ Complete the proof by drawing the correct interchange and providing the reasoning that the latency has not increased due to this transformation.

Solution: See the notes of lecture 6. A and J_k are being pulled forward so their latency decreases. Job J_l is pushed backwards, but as its deadline is past that of job J_k the lateness must be less (or equal when $d_k = d_l$) than that of J_k in the original schedule, which was less or equal to L_{max}. QED.

Question 3 [20 points]

Consider the following task set under RM scheduling:

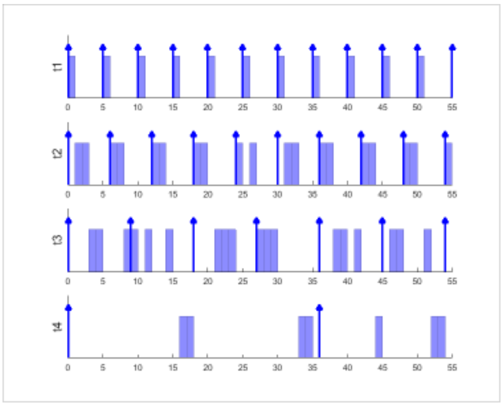
	C_i	T_i
τ_1	1	5
τ_2	2	6
τ_3	3	9
τ_4	4	36

(a) 3 points Compute the LL and HB bounds. What can you say about the feasibility of the task set?

Solution: LL-bound = 0.757, HB-bound = 2.37. As neither bound is satisfied, but the processor load (0.98) is below 1 the task set **could** be feasible.

(b) 5 points Draw the critical instant for the task set and report the worst-case response times of the tasks.

Solution:



	worst-case response
τ_1	1
τ_2	3
τ_3	9
τ_4	35

(c) 4 points Compute the best-case response times of the tasks.

Solution:

	best-case response	iterations
τ_1	1	1
τ_2	2	3, 2
τ_3	3	9, 6, 4, 3
τ_4	19	35, 29, 26, 23, 20, 19

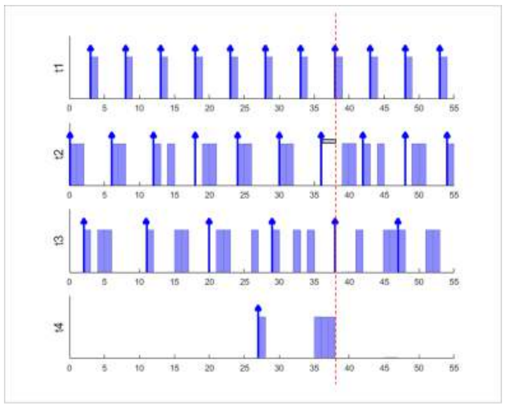
(d) 2 points Compute the response jitter of the tasks.

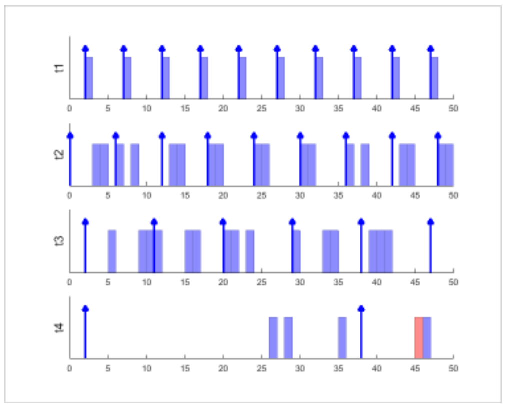
Solution:

	response jitter
τ_1	0
τ_2	1
τ_3	6
τ_4	16

(e) $\boxed{\text{6 points}}$ Consider the effects of task τ_2 incurring an activation jitter of 2. Determine the best-case response of task τ_4 by drawing its optimal instant. Is the task set still feasible?

Solution:





The activation jitter thus improves the best-case response of τ_4 to 11. However, its worst-case response increases to 44, which exceeds the deadline rendering the task set infeasible.

Question 4 [20 points]

Consider the following task set:

	C_i	D_i	T_i
τ_1	2	3	5
τ_2	4	6	8
τ_3	1	15	20

(a) $\boxed{\text{10 points}}$ Use the processor demand criterion to determine the feasibility of the above task set under EDF scheduling.

Solution: D$_{max}$ = 16, the hyperperiod = 40, and $L^* = 41$, so we need to check as far as L = 40.

	demand
g(0,3)	2
g(0,6)	6
g(0,8)	8
g(0,13)	10
g(0,14)	14
g(0,15)	15
g(0,18)	17
g(0,22)	21
g(0,23)	23
g(0,28)	25
g(0,30)	29
g(0,33)	31
g(0,35)	32
g(0,38)	38

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

(b) $\boxed{\text{5 points}}$ Argue that

$$U_p \leq 1 - \frac{\min\limits_i (T_i - D_i)}{H} \tag{1}$$

is a necessary condition for feasibility under EDF, where H denotes the hyperperiod of the task set. (In other words, violating Equation 1 leads to an unfeasible EDF schedule.)

Solution: At the end of a hyperperiod no task is ready for execution when the tasks' deadlines are strictly less than their respective periods ($D_i < T_i$). This idle period lasts for at least $\min\limits_i (T_i - D_i)$. Thus tasks can only execute for at most $H - \min\limits_i (T_i - D_i)$ units. In other words the demand ($H \times U_p$) must be less than that. Dividing by H leads to Equation 1.

(c) $\boxed{\text{5 points}}$ Determine the feasibility when we change task τ_3 into $< C_3{=}3, D_3{=}35, T_3{=}40 >$.

Solution: The hyperperiod stays the same ($H = 40$), but the utilization of task τ_3 increases ($\frac{1}{20} = \frac{2}{40} < \frac{3}{40}$) so trouble may be ahead. Let's check Equation 1. $U_p = \frac{39}{40}$, while $\min\limits_i (T_i - D_i) = 2$, so the processor load exceeds the bound ($1 - \frac{2}{40} = \frac{38}{40}$) rendering the task set unfeasible.

Question 5 [10 points]

Consider the following hard real-time, preemptable tasks with deadlines:

	C_i	D_i	T_i
τ_1	2	6	8
τ_2	3	5	9
τ_3	3	10	12

Slack stealing is known to be the best **on-line** policy for handling additional (soft real-time) aperiodic jobs.

(a) $\boxed{\text{3 points}}$ Explain why slack stealing is **not** the best among all scheduling policies. *Hint: give a specific counter example.*

Solution: Slack stealing cannot look into the future and has to make a decision as soon as a job becomes available. See slide 41 'Scheduling dilemma' from the lecture on fixed priority servers for an example.

(b) $\boxed{\text{4 points}}$ Consider the combination of DM scheduling and slack stealing. Report the response times (3x) of a job J arriving at $t = 3$ and having a compute time of C_J ranging from 1 to 3. *Hint: a single picture says it all.*

Solution:

C_J	response time
1	6
2	20
3	21

(c) $\boxed{\text{3 points}}$ Report the response times (3x) for the same jobs J when slack stealing is combined with EDF scheduling.

Solution:

C_J	response time
1	1
2	12
3	21

Question 6 [15 points]

Consider the following task set under **non-preemptive** RM scheduling:

	C_i	T_i
τ_1	2	8
τ_2	4	9
τ_3	3	17
τ_4	2	20

Due to the self-pushing phenomenon the analysis may need to be extended beyond the first period of a task. This requires the computation of the level-i busy periods using the following equation:

$$L_i = B_i + \sum_{h=1}^{i} \left\lceil \frac{L_i}{T_h} \right\rceil C_h \tag{2}$$

(a) $\boxed{\text{5 points}}$ Prove that the inequality '$L_{i+1} \geq L_i$' holds.

Solution: Insight: when moving from L_i to L_{i+1} the interfernce of higher priority tasks will increase, but the blocking time may decrease ($B_{i+1} \leq B_i$) so we have to show that the decrease in blocking time (at most C_{i+1}) is offset by the increase in busy time (at least C_{i+1}). Here we go:

$$\begin{aligned}
L_{i+1} &= B_{i+1} + \sum_{h=1}^{i+1} \left\lceil \frac{L_{i+1}}{T_h} \right\rceil C_h \\
&= B_{i+1} + \sum_{h=1}^{i} \left\lceil \frac{L_{i+1}}{T_h} \right\rceil C_h + \left\lceil \frac{L_{i+1}}{T_{i+1}} \right\rceil C_{i+1} \\
&\geq B_{i+1} + \sum_{h=1}^{i} \left\lceil \frac{L_{i+1}}{T_h} \right\rceil C_h + C_{i+1} \\
&= \max_{j>i+1}\{C_j\} + C_{i+1} + \sum_{h=1}^{i} \left\lceil \frac{L_{i+1}}{T_h} \right\rceil C_h \\
&\geq \max_{j>i}\{C_j\} + \sum_{h=1}^{i} \left\lceil \frac{L_{i+1}}{T_h} \right\rceil C_h \\
&= B_i + \sum_{h=1}^{i} \left\lceil \frac{L_{i+1}}{T_h} \right\rceil C_h \\
&= L_i
\end{aligned}$$

(b) $\boxed{\text{5 points}}$ The above result can be used to start the fixed-point computation using the level-i-busy period of the previous task. How many iterations can be "saved" when computing the level-i busy period for task τ_4 knowing that $L_3 = 32$?

Solution: Note: there is **no** need to compute L_4 as we can determine the number of saved iterations when both methods arrive at the same intermediate result! Thus we are looking for the old method to arrive at 32.

$$L_4^{(0)} = B_4 + \sum_{h=1}^{4} C_h = 0 + 2 + 4 + 3 + 2 = 11$$

$$L_4^{(1)} = B_4 + \sum_{h=1}^{4} \left\lceil \frac{L_4^{(0)}}{T_h} \right\rceil C_h$$

$$= \left\lceil \frac{11}{8} \right\rceil 2 \ + \left\lceil \frac{11}{9} \right\rceil 4 \ + \left\lceil \frac{11}{17} \right\rceil 3 \ + \left\lceil \frac{11}{20} \right\rceil 2 = 2 \times 2 \ + 2 \times 4 \ + 1 \times 3 \ + 1 \times 2 = 17$$

$$L_4^{(2)} = \left\lceil \frac{17}{8} \right\rceil 2 \ + \left\lceil \frac{17}{9} \right\rceil 4 \ + \left\lceil \frac{17}{17} \right\rceil 3 \ + \left\lceil \frac{17}{20} \right\rceil 2 = 3 \times 2 \ + 2 \times 4 \ + 1 \times 3 \ + 1 \times 2 = 19$$

$$L_4^{(3)} = \left\lceil \frac{19}{8} \right\rceil 2 \ + \left\lceil \frac{19}{9} \right\rceil 4 \ + \left\lceil \frac{19}{17} \right\rceil 3 \ + \left\lceil \frac{19}{20} \right\rceil 2 = 3 \times 2 \ + 3 \times 4 \ + 2 \times 3 \ + 1 \times 2 = 26$$

$$L_4^{(4)} = \left\lceil \frac{26}{8} \right\rceil 2 \ + \left\lceil \frac{26}{9} \right\rceil 4 \ + \left\lceil \frac{26}{17} \right\rceil 3 \ + \left\lceil \frac{26}{20} \right\rceil 2 = 4 \times 2 \ + 3 \times 4 \ + 2 \times 3 \ + 2 \times 2 = 30$$

$$L_4^{(5)} = \left\lceil \frac{30}{8} \right\rceil 2 \ + \left\lceil \frac{30}{9} \right\rceil 4 \ + \left\lceil \frac{30}{17} \right\rceil 3 \ + \left\lceil \frac{30}{20} \right\rceil 2 = 4 \times 2 \ + 4 \times 4 \ + 2 \times 3 \ + 2 \times 2 = 34$$

Oops, we overshot. Let's see what happens when we do the new method starting with 32.

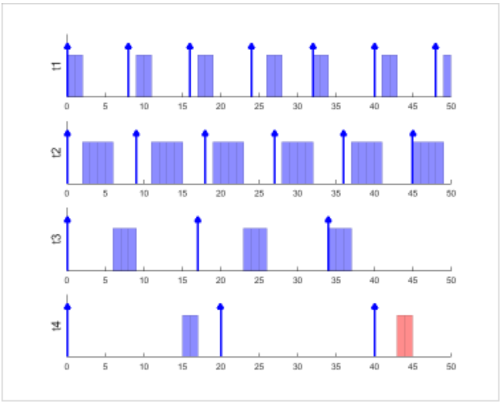
$$L_4^{(0)} = L_3 = 32$$

$$L_4^{(1)} = \left\lceil \frac{32}{8} \right\rceil 2 \ + \left\lceil \frac{32}{9} \right\rceil 4 \ + \left\lceil \frac{32}{17} \right\rceil 3 \ + \left\lceil \frac{32}{20} \right\rceil 2 = 4 \times 2 \ + 4 \times 4 \ + 2 \times 3 \ + 2 \times 2 = 34$$

We arrive at the same value, so we saved $5 - 1 = 4$ iterations.

(c) $\boxed{\text{5 points}}$ Demonstrate that the task set is unfeasible as task τ_4 misses a deadline.

Solution:



Task τ_4 misses the 2^{nd} deadline.