TELECOM Paris

IP PARIS

SETI-CPS-ES Option
Critical Real-Time Systems - Exam

Florian Brandner - Laurent Pautet

2024-2025

2h00 – **Handwritten documents allowed**
Electronic communication devices (phones, ...) are not allowed
You can reply in only one language, either English or French
The clarity and style of your explanations will be considered during grading
Only readable, precise, and sufficiently explained answers will be considered
Points for grading may vary

The exam is composed of independent parts that can be treated in any order. You are asked to write each part on a separate copy.

Part 1 : real-time scheduling

1 Real-time scheduling for mono-processor (5 points)

Consider the following set of independent, synchronous, fixed-priority tasks. It runs on a mono-core processor. Tasks have constrained deadlines ($D \leq T$). We make no assumption on the fixed priority assignment policy.

Task	Budget	Period	Deadline
T0	3	12	7
T1	3	15	15
T2	4	20	7
T3	4	30	28
T4	8	60	60

Question 1. (0.5 point) Necessary condition [course]
Recall the necessary scheduling condition for mono-core processors. Is it satisfied for this task set?

> The utilization rate must be less than 1.
> $U = \frac{3}{12} + \frac{3}{15} + \frac{4}{20} + \frac{4}{30} + \frac{8}{60} = \frac{11}{12} \leq 1$.
> Potentially, the task system can be scheduled on a mono-core processor.

Question 2. (1 point) Worst Case Response Time [course]
Recall and explain the formula for calculating the worst-case response time. Recall the notations.

$$R_i^n = \sum_{j \in hp(i)} \lceil \frac{R_i^{n-1}}{T_j} \rceil \times C_j \leq D_i$$

We take into account the execution times of tasks with a higher priority than priority of task i, whose worst response time we want to calculate iteratively. $hp(i)$ designates tasks with a priority greater than or equal to that of task i.

Question 3. (1.5 points) Optimal Priority Assignment [course]

Describe the OPA (Optimal Priority Assignment) algorithm for calculating all the priority assignments that make this task set schedulable. Be sure to detail all the properties to be checked.

We look for all the tasks to which we can assign the lowest priority by calculating their worst-case response times and checking that they meet their deadlines. Then, for each of these tasks, t_i, we define a new reduced task set from which we have removed t_i and apply the same approach, i.e. we find all the tasks to which we can assign the lowest priority. This way, we find all the possible priority assignments that make this task set schedulable.

Question 4. (2 points) Valid Priority Assignments [course]

Using the previous questions, find the priority assignments that make this task system schedulable. Explain your reasoning.

We can systematically explore the five possibilities to assign the lowest priority. And then reiterate with the subsets of tasks of high priorities. But here, we deduce the final result thanks to some basic considerations.

The workload at $t = 0$ (when all the tasks start synchronously) is $3 + 3 + 4 + 4 + 8 = 22$. Thus, only T3 and T4 may have the lowest priority. But at $t = 22$, T0 and T1 which have high priorities have been reactivated. Thus the workload is now $3 * 2 + 3 * 2 + 4 + 4 + 8 = 28$. Only T4 can have the lowest priority. We check that by computing the response time of T4, its deadline is satisfied.

Once T4 is removed from the taskset, the workload at $t = 0$ is $3 + 3 + 4 + 4 = 14$. Thus, only T1 and T3 may have the lowest priority. But at $t = 12$, T0 which has a high priority has been reactivated and the workload is at least $14 + 3 = 17$ and thus only T3 can have the lowest priority. We check that by computing the response time of T3, its deadline is satisfied.

Once T3 is also removed from the taskset, the workload at $t = 0$ is $3 + 3 + 4 = 10$. Thus, only T1 may have the lowest priority. We check that by computing the response time of T1, its deadline is satisfied.

Once T1 is also removed from the taskset, the workload at $t = 0$ is $3 + 4 = 7$. Thus, both T0 and T2 can have have the lowest priority.

Tâche	Budget	Période	Échéance	Réponse
T0	3	12	7	3
T2	4	20	7	7
T1	3	15	15	10
T3	4	30	28	20
T4	8	60	55	55

–

Tâche	Budget	Période	Échéance	Réponse
T2	4	20	7	4
T0	3	12	7	7
T1	3	15	15	10
T3	4	30	28	20
T4	8	60	60	55

Note that these assignments correspond to the assignments that DMS would give, T0 and T2 having the same deadline. However, it was not possible to conclude. Indeed, the condition $\sum \frac{C_i}{D_i} \leq n(2^{\frac{1}{n}} - 1)$ is not satisfied ($\frac{11}{12} \not\leq 5(2^{\frac{1}{5}} - 1) \approx 0.74$ and this condition is only sufficient.

2 Code Coverage (2 points)

Question 1. (1 point) Code coverage [course]

Indicate what should be done by a test suite to guarantee the code coverage according to the approaches 1) Statement Coverage(SC), 2) Decision Coverage (DC) and 3) Modified Condition / Decision Coverage (MC/DC). Particular attention will be paid to the clarity of your explanations.

course

Question 2. (1 point) Applying approaches SC, DC, MC/DC

Consider the code below. Illustrate each of these approaches by providing possible values of a, b, and c for the conditions. Particular attention will be paid to readability and clarity.

```
1 if ((a == 0) || ((b != 1) || (c == 2))) {F1(a, b, c)}; F2(a, b, c) ;
```

Trivial with only *or*.

A	B	C	D
F	F	F	F
T	F	F	T
F	T	F	T
F	F	T	T

3 Multi-core processor scheduling (3 points)

The following set of independent, synchronous, deadline implicit tasks is executed on a **dual-core processor** according to a Global Deadline Monotonic scheduling policy.

Task	**Budget**	**Period**
T0	1	2
T1	2	3
T2	2	4

Question 1. (1 point) Critical Instant [course]

Recall why the critical instant is important when calculating the response time, and under what circumstances it occurs on a **mono-core processor**.

course

Question 2. (1 point) Applying Global Deadline Monotonic scheduling

Explain how a Global Deadline Monotonic scheduler works on a **multi-core processor** and give the chronogram of the given task set executed by a Global Deadline Monotonic scheduler on a **dual-core processor**. Justify the time interval on which the chronogram is based.

> course

Question 3. (1 point) Worst Case Response Time

Give the worst-case response time for the different tasks and conclude on what this example highlights.

> $R_0 = 1, R_1 = 2, R_2^1 = 3, R_2^2 = 4$.
> It is a scheduling anomaly, the critical instant does not occur when the tasks are all released at the same instant.

Part 2 - Program and WCET Analysis

4 Static Program Analysis(2 points)

Question 1. (2 points) Explain the three elements that define the value range analysis, i.e., the abstract domain, the transfer function, and meet operator. You do not have to provide precise formulas, but rather explain the key ideas. Also explain the partial order on the abstract domain. Provide and explain brief examples for each element of your answer.

- The abstract domain:
  Associates with each variable in the program an upper and a lower bound – the so-called ranges. Example: $(\texttt{x}, [5, 10])$, variable `x` is known to have a value larger or equal to 5, but smaller or equal to 10.
- Partial order:
  Is based on inclusion, i.e., a range $a = [la, ua]$ is smaller/greater than a range $b = [lb, ub]$ in the order if $lb \leq la \leq ua \leq ub$. Example: $[3, 4] \sqsubseteq [0, 10]$.
- The transfer function:
  Apply interval arithmetic for operations of the underlying programming language. Example: Addition is modeled as $[a, b] + [c, d] = [a + c, b + d]$.
- The meet operator:
  Takes the maximum range. Example: $[a, b] \sqcap [c, d] = [min(a, c), max(b, d)]$.

5 Worst-Case Execution Time Analysis (6 points)

Question 1. (2 points) In the lecture you have seen that the *Worst-Case Execution Time* (WCET) of a program can be determined by measurements. Explain how this could be done in practice. Explain potential advantages/disadvantages of this approach.

- Execute the program under *realistic* conditions of its actual execution environment – when deployed.
- Exercise all possible inputs, program paths, hardware states – this is really difficult.
- Take maximum observed execution time (MOET).
- Advantage:
  - Provides real numbers from the real system.
  - Intractable in practice.
- Disadvantages:
  - Impossible to be certain that the worst case was actually found.
  - Compensate uncertainty be a safety margin (i.e., multiply MOET by some factor x).
  - Can be used to validate other WCET analysis techniques.

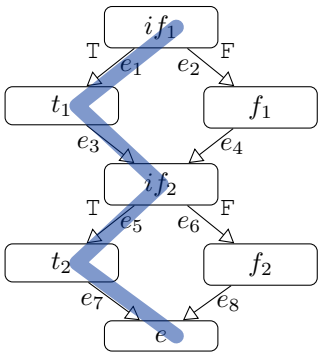
Question 2. (2 points) Explain what *loop bounds* are and why they are needed during WCET analysis. Provide a short C code example explaining the concept and show how you can indicate the loop bounds to the `Otawa` WCET analysis tool.

- A loop bounds specifies how many iterations a program may spend inside a loop, i.e., how often the body of the loop is executed once the program enters the loop.

- Without loop bounds a WCET analyser cannot limit the number of iterations a program spends in a given loop, which may thus stay in that loop indefinitely, resulting in an unbounded WCET.
- Example:

```
1  int x[100];
2  int min = INT_MAX;
3  for(unsigned int i = 0; i < 100; i++) { // Label of loop header: LH
4     if (x[i] < min)
5        min = x[i];
6
7  }
8
9  loop bound @LH is 100;
```

Question 3. (2 points) Consider the *Control-Flow Graph* (CFG) on the right, which consists of two consecutive `if`-statements. The highlighted path indicates that the two `if`-conditions are correlated. More precisely, whenever the condition of the first `if` is `true` the condition of the second `if` will also evaluate to `true`. Provide a *flow fact* for IPET that expresses this correlation. Use the edge names (e_1, e_2, ...) as names for the flow variables. Justify your answer.



$$e_1 \leq e_5$$

Whenever the execution enters e_1 it will continue with e_3 and from there has to continue with e_5 – due to the correlation of the two `if`-conditions. Consequently, e_5 has to be at least as large as e_1, or $e_1 \leq e_5$.

6 Cache Analysis (2 points)

Question 1. (1.5 points) In the lecture you have seen that memory accesses can be classified as *always hit*, *always miss*, or *not classified*. Explain the basic idea behind the data-flow analysis (abstract domain, transfer function, and meet operator) that allows to classify an access as *always miss*. You do not have to provide detailed formulas, it is sufficient to explain the basic idea.

The always miss classification is computed by the `MUST` analysis.

- Abstract domain:
  Associate each memory block mapping to a given cache set with a number in $[0, A]$, where A represents the cache's associativity.
- Transfer function:
  On every cache access update the age of every memory block of the considered cache set.
  - The currently accesses memory block's age is set to 0 in all cases
  - The age of other memory blocks may increment depending on their own age
  - The age may also stay the same
- Meet operator:
  The maximum age is retained.

Question 2. (0.5 points) The picture below illustrates the analysis information with regard to a *Memory Block* m for *Cache Hit/Miss Classification*. It shows a single *Cache Set* of a 4-way set-associative cache to which m maps. Which of the two illustrations (a) or (b) belongs to the analysis from the previous question, i.e., allows to classify accesses as *always miss*? Justify your answer. What is the meaning of the cells that are highlighted in orange?

(a) (b)

Illustration (a) concerns `MUST` analysis. The orange cells indicate the possible places in the cache set (when its content is ordered by age) where the memory block m could be placed, i.e., it might have an age of 0, 1, or 3. It thus describes the maximum age of the memory block.