technische universität dortmund
fakultät für informatik
computer science 12

Prof. Jian-Jia Chen

Real-Time Systems (SS 2014)

Exercise 1: Concept of Real-Time Systems and Static-Priority Scheduling

Discussion Date: 30, April 2014

Exercise 1.1

What are the main differences between general purpose computing and real-time computing? List some applications for different levels of supports of real-time systems.

Solution 1.1

General purpose computing

- Generally, the tasks do not have timing constraints.
- Correctness of system behavior only depends on the correct results of computing.
- The objective is typically to optimize the performance which is usually estimated by the average response time.
- Applications: mathematical computing, word processing etc.

Real-time computing

- Tasks have timing requirements: soft, firm or hard deadlines, periods, etc.
- Correctness depends on the correct computing result and the time at which the results are produced.
- The goal is to guarantee each task will meet the timing constraints under all the execution scenarios.
- Systems may have to react within precise timing constraints to events from the environment.
- Applications: engine controls, robot controls, flight management systems, etc.

Exercise 1.2

For real-time systems, it is important to know the maximum (worst-case) execution time of each task a priori. What are the definition and difference between the worst-case execution time and the worst-case response time? Even if the worst-case execution time of a task is given, there are several other problems that may be encountered during the design of a scheduling algorithm for a real-time system. Can you think of some difficulties? What are possible solutions?

Solution 1.2

- **The definition of the worst-case execution time:** The worst-case execution time is the upper bound on the execution times of a single program, which can be known by analyzing the program. This is independent from the scheduling algorithms.
- **The definition of the worst-case response time:** The worst-case response time is the upper bound of the response time, in which the concurrent execution and multi-tasking have to be considered. This depends on the scheduling algorithms.

If the worst-case execution time of a task is given, there are several constraints might be considered while designing a real-time scheduling algorithm, for example:

1. Timing constraints: the tasks might have alternative execution time range or a special activation patterns. The execution time may not be known in advance and be given as a probability distribution.
2. Precedence relations between the tasks: the tasks should be scheduled according to their dependencies: for example, one task cannot be executed, since the output of another task is not deliverd yet as the next input.
3. Resources sharing problem: several tasks are competing a critical resource such that there is only one task can access the resource in one-time: for example, all the tasks are sharing a information bus as the communication channel. Without a proper management of resources, there might be a strange preemption with priority inversion scenario, such that the higher priority task is preempted by the lower priority task.
4. Preemption costs: the worst-case execution time is analyzed without considering multi-tasking. When a task is preempted, the architectural structure, e.g., the cache states, may be changed by another task, the worst-case execution time is no longer a safe upper bound.

The possible solutions to the problems mentioned above are:

1. Timing constraints: Define a different task model to describe the specific execution behavior in a formal way, which is good for generalizing the problem.
2. Precedence relations between the tasks: The precedence relations betweeen each tasks should be taken into consideration by a suitable scheduling algorithm.

3. Resources sharing problem: A trivial way is to disallow the preemption during the critical sections. Another way might be dynamically changing the priority with a reasonable policy.

4. Preemption costs: The worst-case response time analysis has to also consider such preemption costs or the worst-case execution time analysis should also account for such situations.

Hint: more information on this subject is available in the book of G. Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, pp.41, Ch.2.

Exercise 1.3

Suppose that the following set of jobs is given:

	J_1	J_2	J_3	J_4	J_5
a_j	0	2	8	10	15
C_j	4	3	6	3	4
d_j	6	8	20	14	22

(a) What is the resulting schedule of the shortest-job-first (SJF) scheduling policy?

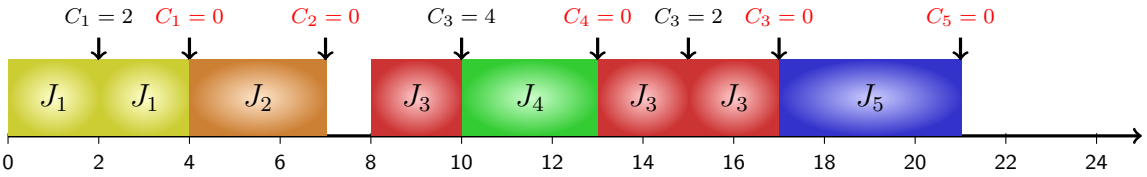
(b) What is the resulting schedule of the earliest-deadline-first (EDF) scheduling policy?

(c) What is the average response time of SJF and EDF, respectively?

(d) Mr. S claims that SJF is optimal for his system, and Miss E claims that EDF is optimal for her system. Is it possible that both of them are correct? Please make their descriptions more clear.

Solution 1.3

(a) schedule for SJF



(b) schedule for EDF and schedule for SJF are the same in this job set.

(c) Coincidentally, the average response time of SJF and EDF are totally the same in this exercise:

$$\frac{\sum_{j=1}^{5}(f_j - a_j)}{5} = 27/5 = 5.4$$

(d) Yes, it is possible that both of them are correct.

In order to make the descriptions better, Mr. S should say SJF scheduling policy is optimal w.r.t the system performance, which is estimated by the average response time. Miss E has to emphasize EDF scheduling policy is optimal with respect to the feasibility of deadline satisfactions, where all the jobs should be finished no later than their deadlines.

Exercise 1.4

Suppose that we are given the following 3 sporadic real-time tasks with implicit deadlines.

	τ_1	τ_2	τ_3
C_i	1	2	3
T_i	4	6	10

(a) What are their priority levels? Is the rate-monotonic (RM) schedule feasible?

(b) What happens if we change the minimum inter-arrival time of task τ_3 from 10 to 8.

Solution 1.4

(a) Their priority levels are assigned by the scheduling policy. For instance, according to the rate-monotonic scheduling policy, the priority levels will be $\tau_1 > \tau_2 > \tau_3$, where τ_1 is the highest priority task with the shortest period $T_1 = 4$, τ_2 has the second priority, and τ_3 has the lowest priority.

With the time-demand schedulability test, we can find that when time $t = 10$, the time demand function will be $W_3^4(10) \leq 10$ under rate-monotonic scheduling. It presents that the tasks are schedulable and WCRT = 10 under rate-monotonic scheduling policy.

$$
\begin{aligned}
W_3^0(0) &= 6 \\
W_3^1(6) &= \left\lceil \frac{6}{4} \right\rceil \cdot 1 + \left\lceil \frac{6}{6} \right\rceil \cdot 2 + 3 = 7 \\
W_3^2(7) &= \left\lceil \frac{7}{4} \right\rceil \cdot 1 + \left\lceil \frac{7}{6} \right\rceil \cdot 2 + 3 = 9 \\
W_3^3(9) &= \left\lceil \frac{9}{4} \right\rceil \cdot 1 + \left\lceil \frac{9}{6} \right\rceil \cdot 2 + 3 = 10 \\
W_3^4(10) &= \left\lceil \frac{10}{4} \right\rceil \cdot 1 + \left\lceil \frac{10}{6} \right\rceil \cdot 2 + 3 = 10
\end{aligned}
$$

(b) If the minimum inter-arrival time of task τ_3 is changed from 10 to 8, according to the time-demand schedulability test, we can only check to $t = 8$, such that $W_3(8) = 9 > t = 8$. Then it is clear that if the minimum inter-arrival time of task τ_3 is changed from 10 to 8, it is not feasible.

Exercise 1.5

Explain how to use the time-demand schedulability test to prove that rate-monotonic scheduling is an optimal static-priority scheduling policy (with respect to schedulability).

Solution 1.5

Suppose that there exists a static-priority scheduling policy that produces feasible schedules for a given task set. We can show that the rate-monotonic scheduling policy is also feasible for the given task set. Without loss of generality, we index the tasks according the priority levels of the feasible static-priority policy, in which τ_1 is the task with the highest priority and τ_n is the task with the lowest priority.

We prove this statement by using swapping properties. Suppose that τ_k and τ_{k+1} do not follow the rate-monotonic prioritization. That is, $T_k > T_{k+1}$. Then, we can show that swapping the priority levels of τ_k and τ_{k+1} remains feasible, provided that the original ordering is also feasible.

The feasibility of the given priority levels implies that

$$\exists t \text{ with } 0 < t \leq T_i \ \ and \ \ C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t$$

for $i = 1, 2, \ldots, n$.

By swapping the priority levels of τ_k and τ_{k+1}, the feasibility of tasks $\tau_1, \ldots, \tau_{k-1}$ remains, as tasks τ_k and τ_{k+1} are not accounted anyway for the schedulabiliy tests, and the feasibility of $\tau_{k+2}, \ldots, \tau_n$ remains as the term $\sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j$ does not care the priority levels of τ_k and τ_{k+1}.

The only problem is whether τ_k and τ_{k+1} remain feasible. It is clear that after swapping τ_{k+1} is feasible, since the existence of $0 < t \leq T_{k+1} \ \ with \ \ C_{k+1} + \sum_{j=1}^{k} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t$ implies the existence of $0 < t \leq T_{k+1} \ \ with \ \ C_{k+1} + \sum_{j=1}^{k-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t$.

After swapping τ_k is also feasible, since

$$\exists 0 < t \leq T_{k+1} < T_k \ \ with \ \ C_{k+1} + \sum_{j=1}^{k} \left\lceil \frac{t}{T_j} \right\rceil C_j = C_{k+1} + C_k + \sum_{j=1}^{k-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t.$$

That is, the worst-case response time of τ_k in such a case is at most T_{k+1}, which is less than T_k in the above definition.

Therefore, by swapping the priority levels repeatedly, the optimality of RM scheduling is achieved.

Challenge 1.6

Mr. Smart suggests the following schedulability test of static-priority scheduling for sporadic real-time tasks, as defined in the course. He claims that task τ_i can meet its its relative deadline under the static-priority scheduling if and only if the following mixed-integer linear programming has a solution.

$$C_i + \sum_{j=1}^{i-1} n_j \cdot C_j \leq t \tag{1}$$

$$n_j \cdot T_j \geq t \qquad \forall j = 1, 2, \ldots, i-1 \tag{2}$$

$$n_j \in \mathbf{N} \qquad \forall j = 1, 2, \ldots, i-1 \tag{3}$$

$$0 < t \leq D_i, \tag{4}$$

where t is a positive variable, described in (4), and n_j is a positive integer number, described in (3). Please either explain/prove or disprove his argument.

Solution 1.6

The time-demand schedulability test is a necessary and sufficient schedulability (exact) test for static-priority scheduling for sporadic real-time tasks. One way to prove Mr. Smart's argument is to prove the argument is equivalent to the time-demand schedulability test.

If-part If the argument holds with a certain t^*, the time-demand schedulability test says schedulable under the static-priority scheduling.

$$\begin{aligned}
&\text{If } \exists t^*, n_j^* \qquad \text{s.t. (1), (2), (3), (4) hold,} \\
&\text{then we know that} \\
&n_j^* \cdot T_j \geq t^* \\
\Rightarrow &\left\lceil \frac{t^*}{T_j} \right\rceil \leq n_j^* \\
\Rightarrow &\text{by (1)}, C_i + \left\lceil \frac{t^*}{T_j} \right\rceil \cdot C_j \leq t^*.
\end{aligned}$$

Only-If-part If there exists a certain t^* such that the time-demand schedulability test holds, which also holds (4), we can find a certain n_j^*, which holds (2), (3), greater than or equal to the ceiling function $\left\lceil \frac{t^*}{T_j} \right\rceil$ in the test. At the end, (1) also holds with a certain t^*.

The argument of Mr. Smart is correct, since the mixed-integer linear programming is totally the same as the time-demand schedulability test.