

Real-Time Systems

Solutions to Exercise 7: Scheduling

1 a. The tasks are given the following priorities:

Task name	T_i	Priority
A	10	Medium
B	5	High
C	20	Low

The schedule is shown in Figure 1. The worst-case response times of the tasks are $R_A = 3$, $R_B = 2$, $R_C = 9$, i.e., task A will not meet its deadlines.

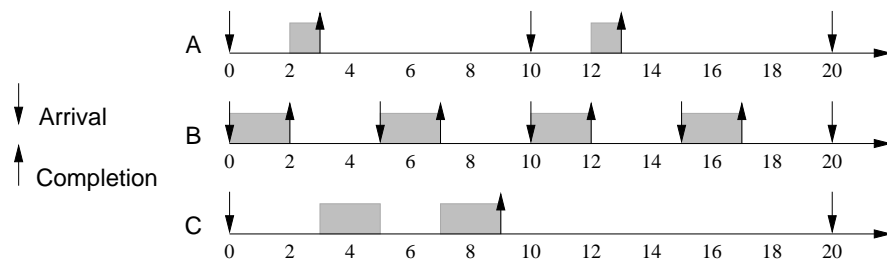


Figure 1 Schedule with rate-monotonic priority assignments.

b. The tasks are given the following priorities:

Task name	D_i	Priority
A	2	High
B	4	Medium
C	10	Low

The schedule is shown in Figure 2. The worst-case response times of the tasks are $R_A = 1$, $R_B = 3$, $R_C = 9$, i.e. all tasks will meet their deadlines.

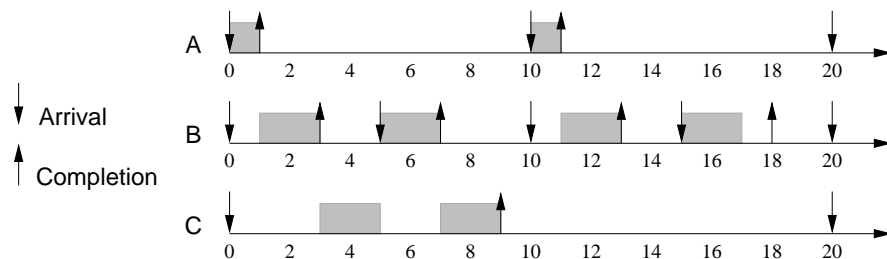


Figure 2 Schedule with deadline-monotonic priority assignments.

2 a.

$$U = \sum_{i=1}^3 \frac{C_i}{T_i} = \frac{0.6}{3} + \frac{1.2}{4} + \frac{1.5}{5} = 0.8$$

b. A sufficient condition for all the tasks to meet their deadlines is

$$\sum_{i=1}^n \frac{C_i}{T_i} \leq n(2^{1/n} - 1).$$

In this case $n(2^{1/n} - 1) = 0.7798 < 0.8$, and we cannot conclude from this criterion whether the tasks are schedulable or not.

c. Here A has high priority, B has medium priority, and C has low priority. The worst-case response time R_i of task i satisfies

$$R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

where $hp(i)$ is the set of tasks of higher priority than i , and $\lceil \cdot \rceil$ is the ceiling function. The equation can be solved by fix-point iterations.

Start with calculating the worst-case response time for A (R_A): as A has the highest priority it will not be interrupted, thus

$$R_A = C_A = 0.6.$$

Task B can be interrupted by task A. Start the fix-point iterations with the initial value $R_B^1 = C_B = 1.2$:

$$R_B^2 = C_B + \left\lceil \frac{R_B^1}{T_A} \right\rceil C_A = C_B + C_A = 1.8.$$

$$R_B^3 = C_B + \left\lceil \frac{R_B^2}{T_A} \right\rceil C_A = C_B + C_A = 1.8.$$

Thus $R_B = 1.8$.

Task C can be interrupted both by A and B. Start with the initial value $R_C^1 = 1.5$.

$$R_C^2 = C_C + \left\lceil \frac{R_C^1}{T_A} \right\rceil C_A + \left\lceil \frac{R_C^1}{T_B} \right\rceil C_B = C_C + C_A + C_B = 3.3.$$

$$R_C^3 = C_C + \left\lceil \frac{R_C^2}{T_A} \right\rceil C_A + \left\lceil \frac{R_C^2}{T_B} \right\rceil C_B = C_C + 2C_A + C_B = 3.9.$$

$$R_C^4 = C_C + \left\lceil \frac{R_C^3}{T_A} \right\rceil C_A + \left\lceil \frac{R_C^3}{T_B} \right\rceil C_B = C_C + 2C_A + C_B = 3.9.$$

Thus $R_C = 3.9$.

As $R_i \leq D_i$ for all i , all the tasks will meet their deadlines with rate monotonic scheduling.

- 3 a.** The utilization is 90%. This is above the limit $(n(2^{1/n} - 1) = [n=2] = 0.828)$ for the sufficient condition and the exact analysis must be applied. The tasks will meet their deadlines if and only if

$$R_i \leq D_i, \quad \forall i$$

where

$$R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

where $hp(i)$ is the set with all tasks of higher priority than task i and $\lceil x \rceil$ is the ceiling function that returns the smallest integer $\geq x$.

The calculation of R_i is a recurrence equation that can be solved by the iteration

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j, \quad R_i^0 = 0$$

For task B the response time becomes equal to the execution time, i.e., $= 1$. For task A the following results are obtained.

$$\begin{aligned} R_A^0 &= 0, \quad R_A^1 = C_A = 2 \\ R_A^2 &= C_A + \left\lceil \frac{R_A^1}{T_B} \right\rceil C_B = C_A + C_B = 3 \\ R_A^3 &= C_A + \left\lceil \frac{R_A^2}{T_B} \right\rceil C_B = C_A + 2C_B = 4 \\ R_A^4 &= C_A + \left\lceil \frac{R_A^3}{T_B} \right\rceil C_B = C_A + 2C_B = 4 \end{aligned}$$

Hence, the response times are smaller than the deadlines for both tasks, and the task set is schedulable.

- b.** Task A has higher priority than task B. We draw the schedule for the worst-case when both tasks are released simultaneously, at time 0. The maximum response time is obtained for the first invocation that

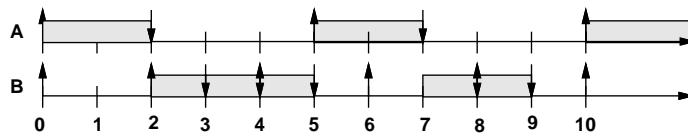


Figure 3 Schedule

is released at time 0 and does not become finished until time 3, i.e., the maximum response time is 3, well above the deadline that is 2.

- c.** For task B with the highest priority the worst-case and the best-case input-output latencies are both equal to the execution time, i.e., $= 1$. Consider now task A. The worst case occurs when task A is preempted by task B immediately after it has performed the sampling. Then first task B executes for 1 time unit, then task A is resumed and continues for another time unit until it again is preempted by task B. After yet

another time unit task A may continue and finish. The worst case latency for task A is thus equal to 4. Using similar reasoning one can easily see that the best-case input latency for task A is equal to 3.

4 a.

$$U = 2/5 + 4/7 = 0.9714 > 2(2^{1/2} - 1) = 0.8284$$

It is not possible to prove schedulability using the sufficient RM schedulability test.

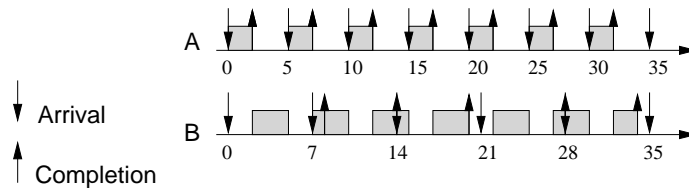
b. Since task A has highest priority one immediately gets $R_A = C_A = 2$. When calculating R_B one gets the following:

$$R_B^0 = 0, R_B^1 = C_B = 4, R_B^2 = C_B + \lceil \frac{4}{5} \rceil 2 = 6$$

$$R_B^3 = C_B + \lceil \frac{6}{5} \rceil 2 = 8$$

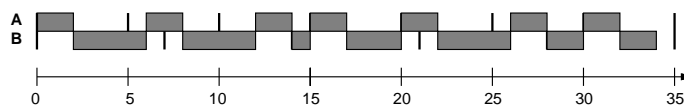
Since R_B^3 is larger than the deadline for task B, the task set is not schedulable using rate-monotonic scheduling.

c.



d. Since $U < 1$ the task set is schedulable with EDF scheduling.

e.



Here, we assume that when two processes are equally close to their deadlines, the task that was added to the ready queue latest has priority. This occurs at time=30 when both tasks have 5 time units to their deadlines. Here, task A interrupts task B. The task set would, however, be schedulable also with the opposite policy of letting the currently executing task finish.

5 a. The tasks will not meet their deadlines. When all the threads have been created and executed their SetPriority, process B will execute, calculate the next execution time and do a waituntil. The worst case execution time is 0.8 msec. At that time thread A will start executing. The first thing it does is that it measures the current time which will return the same tick as for thread B. This means that thread B and

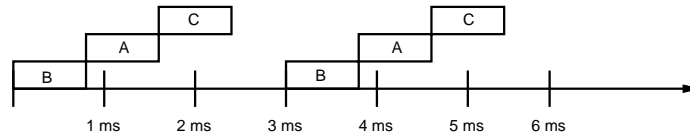


Figure 4 Clock interrupts

A both will try to run in the same clock time tick. The situation is shown in Fig. 4.

Thread A and B will both be scheduled for execution at time $t = 0, 3, 6, 9, \dots$. Thread C will be scheduled for execution at time $t = 1, 4, 7, 10, \dots$. The response time of B will be 0.8 msec. The response time of A will be 1.6 msec, i.e., larger than the deadline. The response time of C will be 1.4 msec, also larger than the deadline.

- b.** The load on the CPU is balanced if the execution of the three threads are spread out evenly in time. A simple way of ensuring this for this simple case is to do a simple `WaitTime` between the calls to `CreateProcess`.

```
BEGIN
    ...
    ...
    CreateProcess(TaskprocessA,1000,"A");
    WaitTime(1);
    CreateProcess(TaskprocessB,1000,"B");
    WaitTime(1);
    CreateProcess(TaskprocessC,1000,"C");
    Wait(Terminate);
END Main.
```

Process A will now be scheduled for execution at $t = 0, 3, 6, 9, \dots$ process B will be scheduled for execution at $t = 1, 4, 7, 10, \dots$ and process C will be scheduled for execution at $t = 2, 5, 8, 11, \dots$. This means that all processes will meet their deadlines.

(In the scheduling theory, this corresponds to introducing *offsets* to the tasks. Introducing offsets can increase the schedulability of a task set.)

6 a. Evaluating M terms in the sum takes

$$(4M + 4M + 20M + 40M + 400M) f_c^{-1} = 468M f_c^{-1}$$

seconds.

For $f_c^{-1} = 4 \cdot 10^{-8}$ s and $M = 10$, this evaluates to

$$0.187 \text{ ms}$$

and stability can therefore not be guaranteed.

- b.** For rate monotonic scheduling, it holds that all deadlines will be met if the CPU utilization is less than 69%.

Since the CPU utilization with the original control algorithm is 60%, the additional compensation may use at maximum 9% of the CPU time. We thus have

$$U_{comp} = \frac{C}{T} = \frac{468Mf_c^{-1}}{0.001} \leq 0.09$$

from which we obtain

$$M \leq 4.8$$

Conclusion: No more than 4 terms can be included in the compensation.