Exercises : Real-time Scheduling analysis

Frank Singhoff
University of Brest

June 2013

Exercise 1 : Fixed priority scheduling and Rate Monotonic priority assignment

Given a set of tasks defined by the following parameters : $S_1 = 0$, $P_1 = 29$, $C_1 = 7$, $S_2 = 0$, $P_2 = 5$, $C_2 = 1$, $S_3 = 0$, $P_3 = 10$ and $C_3 = 2$.

We also assume that the task deadlines are equal to their periods ($\forall\ i\ :\ D_i = P_i$). We use a preemptive fixed priority scheduling with Rate Monotonic priority assignment.

1. Apply the processor utilization factor test. Do the tasks meet their deadlines ?
2. Draw the scheduling sequence of this task set during the first 30 units of times with a preemptive scheduler. Redraw the scheduling with a non-preemptive scheduler. What can you see ?
3. Assuming we change task parameters as follow : $P_1 = 30$, $C_1 = 6$, and $C_2 = 3$. This new task set is said to be "harmonic". A **harmonic set** of periodic tasks is one in which the period of each task is an exact multiple of every task that has a shorter period. Apply the processor utilization factor test. Do the tasks meet their deadlines ?
4. Draw the scheduling sequence of this new task set during the first 30 units of times. What can you see now ?
5. Compute worst case response time of each task with the Joseph and Pandia method. You should find worst case response times that are consistent with the response times in the scheduling you drew in the previous question of this exercise.

Exercise 2 : Earliest Deadline First

Given a set of tasks defined by the following parameters : $S_1 = 0$, $P_1 = 12$, $C_1 = 5$, $S_2 = 0$, $P_2 = 6$, $C_2 = 2$, $S_3 = 0$, $P_3 = 24$ and $C_3 = 5$.

We also assume that the task deadlines are equal to their periods ($\forall\ i\ :\ D_i = P_i$). We use a preemptive Earliest Deadline First scheduler.

1. Apply the processor utilization factor test. Do the tasks meet their deadlines ?
2. Compute the hyper-period of this task set. How many idle units of time should we have during this hyper-period ?
3. Draw the scheduling sequence of this task set during the hyper-period. Is the number of idle units of time consistent with your answer of question 2 ?
4. Now assume a non-preemptive Earliest Deadline First scheduler. Re-draw the scheduling sequence of this task set during the hyper-period and compare this scheduling with the scheduling sequence of question 3. Compare the behaviors of the preemptive and non-preemptive schedulers for this task set.

5. Now assume that some aperiodic tasks are released : tasks TA_1 and TA_2. TA_1 has a capacity of 1 unit of time, is released at time 7 and has to meet an absolute deadline at time 9. TA_2 has a capacity of 3 units of time, is released at time 12 and has to meet a an absolute deadline at time 21. We assume that deadline of aperiodic tasks defined above are the dynamic priorities that the EDF must handle to schedule the task set (e.g. $D_i(t)$). Draw the scheduling sequence of this new task set for the first 30 units of time. Can we meet all the deadlines under the preemptive EDF scheduler ?

Exercise 3 : introducing POSIX 1003 scheduling

In this exercise you will practice with POSIX 1003 scheduling policies. We assume a Linux operating system that implements the POSIX 1003 standard with a scheduling model with 100 priority levels (from 0 to 99 with 99 being the highest priority). Priority level 0 is devoted to time sharing tasks. All tasks that have a 0 priority level are scheduled according to the `SCHED_OTHER` policy. The Linux `SCHED_OTHER` policy schedules tasks according to how long they waited for the processor. In priority level 0, the task to be run is the one that has waited for the longest duration to get the processor among all tasks of priority queue 0. Tasks which have a priority level between 1 and 99 are scheduled either with `SCHED_FIFO` or `SCHED_RR`. Of course, the tasks in priority queues 1 through 99 are dispatched before any tasks in the lower priority queue 0. We assume a quantum of one unit of time for the `SCHED_RR` policy. The scheduling is preemptive.

Given the following task characteristics :

Name	Capacity	Release time	Priority	Policy
other1	8	0	0	SCHED_OTHER
rr1	5	2	3	SCHED_RR
rr2	5	2	3	SCHED_RR
fifo1	3	4	5	SCHED_FIFO
fifo2	3	3	2	SCHED_FIFO
fifo3	2	4	5	SCHED_FIFO

Draw the scheduling sequence for this task set from time 0 to time 25.

Exercise 4 : POSIX 1003 scheduling with periodic tasks

In this exercise you will practice with POSIX 1003 scheduling policies. We assume a system composed of two periodic tasks defined with $S_1 = 0, S_2 = 0, C_1 = 1, C_2 = 2, P_1 = 8$ and $P_2 = 7$. We also assume that the task deadlines are equal to their periods ($\forall\ i\ :\ D_i = P_i$).

The operating system is a POSIX 1003.1b operating system with 32 priority levels (priority levels ranging from 0 to 31). Priority level 31 is the highest priority level. This operating system provides POSIX 1003.1b `SCHED_FIFO`, `SCHED_RR` and `SCHED_OTHER` policies. In the case of the `SCHED_RR`, the quantum is about one unit of time. `SCHED_OTHER` implements a time sharing scheduling. The scheduling is preemptive.

1. Check the schedulability of the periodic task set.
2. Besides T_1 and T_2, the processor owns a set of aperiodic processes defined as follows :

Name	Capacity	Release time	Priority	Policy
fifo1	4	5	20	SCHED_FIFO
other1	5	0	0	SCHED_OTHER
fifo2	3	4	5	SCHED_FIFO
rr1	3	3	7	SCHED_RR
rr2	3	4	7	SCHED_RR

Assign a priority level to tasks T_1 and T_2 which allows these periodic tasks to meet their deadlines and which is compliant to the operating system properties (related to POSIX 1003.1b scheduling model). Explain your choice.

3. Draw the scheduling of all processes (both periodic and aperiodic tasks) from time 0 to time 30.

Exercise 5 : sharing resources with PIP

In this exercise, you will investigate the schedulability of a set of tasks that share a block of memory. Given the following set of periodic tasks :

Task	Initial Release	Period	Capacity	Deadline
T_1	$S_1 = 0$	$P_1 = 6$	$C_1 = 2$	$D_1 = 6$
T_2	$S_2 = 0$	$P_2 = 8$	$C_2 = 2$	$D_2 = 8$
T_3	$S_3 = 0$	$P_3 = 12$	$C_3 = 5$	$D_3 = 12$

These three tasks are scheduled with a preemptive fixed priority scheduler and Rate Monotonic. T_1 and T_3 make use of a block of memory. This block of memory is protected by a semaphore to enforce a critical section.

T_1 requires the block of memory during the second half of its capacity while T_3 needs the block of memory during all its capacity.

1. First, we assume that no inheritance protocol is activated with the semaphore. Draw the scheduling of this task set during its hyper-period. In the scheduling sequence, find when the semaphore is allocated and released by tasks T_1 and T_3. Can you see on the graph at which time a priority inversion occurs ?
2. Now we assume that the PIP protocol is active during semaphore allocations and releases. Draw the scheduling sequence of this task set during its hyper-period. In the scheduling sequence, find when the semaphore is allocated and released. Also check that no priority inversion occurs.
3. Compute worst case blocking times B_1, B_2 and B_3.

Exercise 6 : comparing PIP with ICPP

In the previous exercise, you investigated the use of PIP to put an upper bound on the length of time a high priority task has to wait for a shared resource. Because PIP is difficult to implement, PCP protocols are more commonly used. In this exercise, we study the ICPP, a kind of PCP. Given the following set of periodic task :

Task	Initial Release	Period	Capacity	Deadline
T_1	$S_1 = 0$	$P_1 = 31$	$C_1 = 8$	$D_1 = 31$
T_2	$S_2 = 2$	$P_2 = 30$	$C_2 = 8$	$D_2 = 30$

We schedule this task set with a preemptive fixed priority scheduler and Rate Monotonic.

T_1 and T_2 require the access of two shared resources called R_1 and R_2 that are controlled by the two semaphores *mutex1* and *mutex2*.

– T_1 needs R_1 from the 2nd to the 8th units of time of its capacity.
– T_1 needs R_2 from the 4th to the 8th units of time of its capacity.
– T_2 needs R_1 from the 6th to the 8th units of time of its capacity.
– T_2 needs R_2 from the 2nd to the 8th units of time of its capacity.

1. First, we assume that PIP is activated with the semaphores. Draw the scheduling sequence of this task set during the first 30 units of time. In the scheduling sequence, find when the semaphores are allocated and released by the tasks T_1 and T_2. Describe what you see in the scheduling sequence.
2. Second, we assume that ICPP is activated for these semaphores. Draw the scheduling sequence of this task set during the first 30 units of time. On the scheduling sequence, find when the semaphores are allocated and released by the tasks T_1 and T_2. Compare the scheduling sequences of questions 1 and 2.

Exercise 7 : comparing Rate Monotonic and Earliest Deadline First

Assuming two tasks defined by the following parameters : $S_1 = S_2 = 0, P_1 = 8, P_2 = 10, C_1 = 4$ and $C_2 = 5$. We also assume that the task deadlines are equal to their periods ($\forall\ i\ :\ D_i = P_i$).

1. Draw the scheduling sequence from time 0 to time 24 with a preemptive Earliest Deadline First scheduler. Do tasks meet their deadlines ?
2. Draw the scheduling sequence from time 0 to time 24 with a preemptive fixed priority scheduler with Rate Monotonic priority assignment. Do tasks meet their deadlines ?
3. What can you see ? Is it surprising ?

Exercise 8 : comparing EDF and LLF

Assuming two tasks defined by the following parameters : $S_1 = S_2 = 0, P_1 = 9, P_2 = 8, C_1 = 4$ and $C_2 = 3$. We also assume that the task deadlines are equal to their periods ($\forall\ i\ :\ D_i = P_i$).

In this exercise, we investigate how a LLF scheduler works. LLF (Least Laxity Fist) runs the task with the smallest laxity. A task laxity is a dynamic priority. A task i laxity at time t (noted $L_i(t)$) is computed as follows :

$$L_i(t) = D_i(t) - rest(t)$$

where $D_i(t)$ is the task deadline computed as EDF does it ; and $rest(i)$ is the rest of the task capacity of task i e.g. the part of the task i capacity that the task is not yet ran.

1. Draw the scheduling sequence from time 0 to time 20 with a preemptive Earliest Deadline First scheduler. Do tasks meet their deadlines ?
2. Draw the scheduling sequence from time 0 to time 20 with a preemptive Least Laxity First scheduler. Do tasks meet their deadlines ?
3. What can you see ? Is it surprising ?

Exercise 9 : case study

This exercise summarizes the different topics presented in this lecture. You will investigate performances of an application responsible for displaying data to the driver of a car. The application is composed of 5 tasks :

- Task $SENSOR\_1$ reads a speed sensor every 10 milliseconds.
- Task $SENSOR\_2$ reads the temperature inside the car every 10 milliseconds.
- Task $SENSOR\_3$ reads the GPS location of the car every 40 milliseconds.
- Task $DISPLAY\_1$ computes every 12 milliseconds a summary of the data produced by $SENSOR\_1$, $SENSOR\_2$ and $SENSOR\_3$ and displays the results on a first screen.
- When the driver requests it, task $DISPLAY\_2$ displays, on a second screen, the road map centred around the current location of the car. This screen is refreshed every 6 milliseconds.

We have implemented this set of tasks, and, after some measurements, we have noticed that :
– Execution time of tasks $SENSOR\_2$ and $DISPLAY\_1$ is bounded by 2 milliseconds.
– Task $SENSOR\_3$ needs a processor time ranging from 2 to 4 milliseconds.
– Execution times of tasks $SENSOR\_1$ and $DISPLAY\_2$ never exceeded 1 millisecond.

We also assume that all tasks have the same first release time. The engineers would like to check that all tasks finish their current work before being released to do the next one. They have selected an operating system that provides 32 priority levels ranging from 0 to 31. Priority level 31 is the highest priority. The scheduler is a preemptive fixed priority scheduler. This system does not allow us to assign the same priority level to several tasks.

Question 1

The engineers would like you to help them with the priority assignment of each task.
Explain the common criteria that people use to assign priorities to tasks.
Assign a priority for each task of this system. Explain your choice.

Question 2

From your priority assignment, and without drawing the scheduling sequence, compute the worst case response time for tasks $SENSOR\_1$, $SENSOR\_2$, and $SENSOR\_3$.

Question 3

In practice, tasks $SENSOR\_1$, $SENSOR\_2$, and $SENSOR\_3$ share a block of memory through the semaphore *mutex*. $SENSOR\_1$ and $SENSOR\_2$ need the *mutex* during all their capacity. $SENSOR\_3$ needs the *mutex* during the two first units of time of its capacity. The *mutex* uses ICPP.

Explain why the worst case response times of question 2 are incorrect with the use of this *mutex*.

Question 4

Draw the scheduling sequence of this system up to time 25. Show when each shared resource is allocated or/and released.

Question 5

In order to decrease the response time of the system, we run the task set on an architecture with two processors, a and b. The task parameters are now defined by :

Task	Processor	Capacity	Period
$DISPLAY\_1$	b	4	6
$SENSOR\_3$	b	2	20
$DISPLAY\_2$	a	2	3
$SENSOR\_1$	a	2	5
$SENSOR\_2$	a	2	5

These tasks stay scheduled with a preemptive fixed priority scheduler. Priorities are assigned according to Rate Monotonic. We will not take the shared resources into account for this question. **You may assume that all tasks are independent.** Show that at least one task can not meet its deadline.

Question 6

In the previous question, the engineers assigned tasks to processors randomly. Each task may be run on either processor a or on processor b. However, running a given task on a or on b changes the task worst case execution time : processor b is twice as fast as processor a.

Assign each task to processor a or b in order to meet all task deadlines. Explain how you made this assignment and how you have checked deadlines. **You may again assume that all tasks are independent.**