Operating systems, Project 1
EDF

Arnaud Rosette
Denis Steckelmacher

December 12, 2014

Abstract

The first project of Operating Systems consists of designing a simulator for the EDF[1] scheduler. This simulator is then used to study the behavior of the EDF algorithm under a wide range of conditions.

1 Implementation

The project is implemented in C++ and consists of three main parts:

1. The core of the project are the *schedulers*. `AbstractScheduler` is responsible for managing the jobs of every task and scheduling them accordingly to a specific priority assignation algorithm. It also keeps track of the time and checks that no deadline is missed. Figure 2 shows that RM[2], DM[3], EDF, LLF[4] and Fixed[5] are implemented. Every algorithm just has to provide an ordering of the jobs according to its scheduling criterion.

2. In order to debug and generate nice graphs, several *loggers* were implemented. A logger is notified by the scheduling of what happens at every time step and can for instance draw a

[1]Earliest Deadline First
[2]Rate Monotonic
[3]Deadline Monotonic
[4]Least Laxity First
[5]$\tau_1 \succ \tau_2 \succ \tau_3 \succ$ ... regardless of the rate, deadline, period, laxity or any other property of the tasks or jobs.

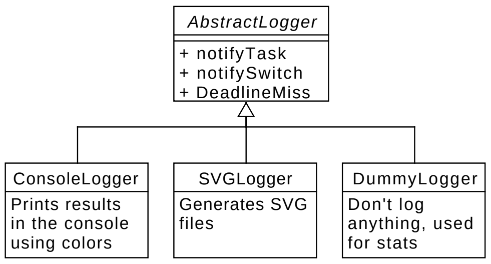


Figure 1: Different loggers supported by the project

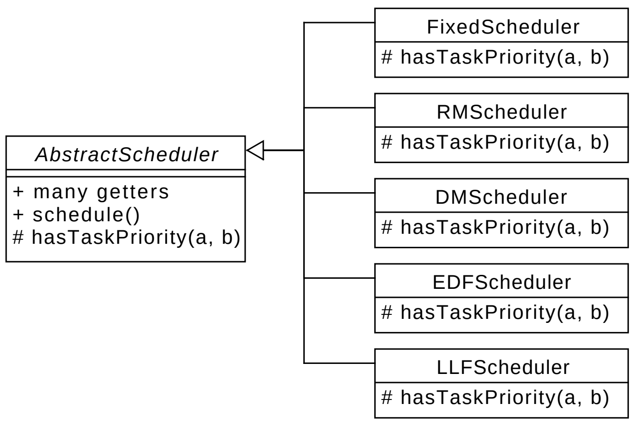


Figure 2: The project implements all the single-processor schedulers seen during the lectures

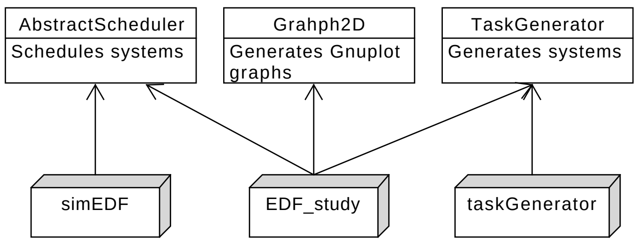


Figure 3: Classes used by the three executables of the project

SVG file or print color bars in the console. The *dummy* logger does not do anything and is used when doing many simulations in order to generate statistics. Figure 1 summarizes this description.

3. Some utility classes are also available and are used by the additional tools provided by the project (generation of system descriptions and automated statistics gathering). Figure 3 shows the dependencies between the executables of the project and the main components described here.

1.1 Generating systems

A system is a set of tasks, each one having an offset, period, deadline and worst-case execution time. The `taskGenerator` tool can be used to generate a set of N tasks whose utilization factors sum to U. Generating those tasks requires 3 steps:

- The offset and period of each task can be set randomly, the other factors will be adapted to those values.
- The total utilization factor U is randomly divided into N parts. This is done by considering a line segment of size U on which $N - 1$ sticks are randomly placed. The length $[0S_1]$ is the utilization factor of the first task, $[S_1S_2]$ is the one of the second task, and so on.
- The utilization factor of each task is multiplied by its period in order to get its worst-case execution time. The deadline of the task is randomly placed between the worst-case execution time of a task and its period (the deadlines are therefore constrained but not implicit).

1.2 Scheduling

At each time step, `AbstractScheduler` asks its subclass to order the schedulable tasks by decreasing priority and takes the first one. An idle time slot is inserted if and only if there is no schedulable task (the scheduler is therefore work-conservative).

Once the task to be scheduled has been chosen, a possible switching time has to be computed:

- If $T_{old} = T_{new}$, no switching time is required
- If $T_{old} = \emptyset$, then a *load time* of $T_i * S$ is used (where T_i is the worst-case execution time of the task being loaded and S the switching time percent specified by the user)
- If $T_{old} \neq T_{new}$, then a switching time of $(T_i + T_j) * S$ is used

The switching time is applied as is and the scheduler becomes blind during it. This means that if there are 8 time slots of switching time, no scheduling decision will be taken during this time (a switch cannot be canceled, in other words). The effect of this behavior can be seen on Figure 4.

2 Difficulties encountered

The EDF algorithm itself is quite easy understand, and even if all the infrastructure supporting it (managing the tasks, keeping track of the time to the next deadline, etc) was a bit complex, there was no real problem here. What was more complicated was the algorithm used to split a

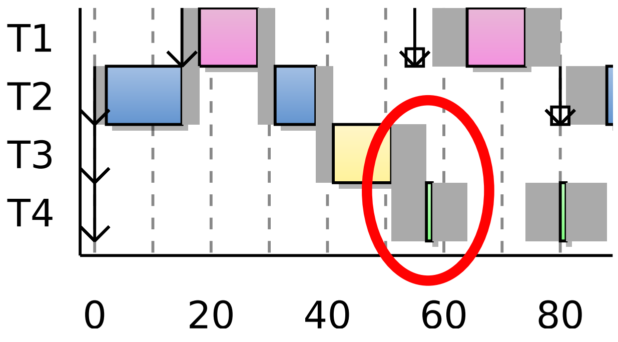


Figure 4: Switch to a job that is immediately preempted because a job with higher priority was released during the switching time

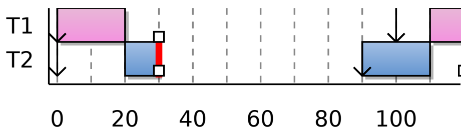


Figure 5: A deadline is missed even though the utilization factor is very low

total utilization factor between tasks. Several people of the class discussed this problem during a midday break and sketched the beginning of a solution, then other discussions and personal work allowed everyone to find a solution very close to the one presented in Section 1.1.

3 Simulation results

Having an EDF simulator allows to perform some tests and see how the algorithm behaves. The tests consist of aggregating information about the schedulability of a system and its idle time depending on the number of tasks in the system, the total utilization factor and the switching time percentage.

The results are presented in two ways. The schedulability being a boolean value, a grid is used to say whether a given set of properties yield a system that is schedulable or not. The idle time is continuous and is therefore plotted in a line graph.

The parameters that vary from simulation to simulation are as follow:

1. The *total utilization* takes a value in 50%, 90% and 110%. This last value is used to show that no system can be scheduled if its total utilization factor is larger than 100%.
2. The *switching time* takes a value in 0%, 5% and 10% in order to show the impact of an increased switching time.
3. The *number of tasks* varies between 2 and 4.

3.1 Schedulability

Figure 6 shows that any system having an utilization factor greater than 100% is not schedulable (according to the theory). What is interesting is that some systems having an utilization lower than 100% were also not schedulable. There are two reasons to that:

	50%	90%	110%
0%	Yes	Yes	**No**
5%	Yes	Yes	**No**
10%	Yes	**No**	**No**

(a) 2 tasks

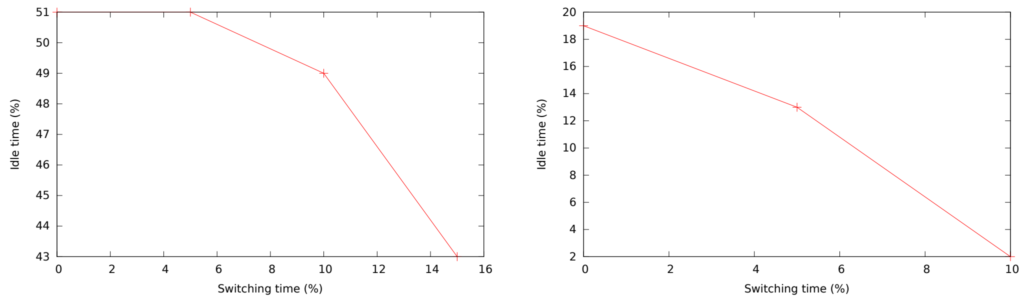
	50%	90%	110%
0%	Yes	No	**No**
5%	Yes	No	**No**
10%	Yes	**No**	**No**

(b) 3 tasks

	50%	90%	110%
0%	Yes	No	**No**
5%	Yes	**No**	**No**
10%	Yes	**No**	**No**

(c) 4 tasks

Figure 6: Schedulability of systems given a set of varying parameters



(a) 50% utilization

(b) 80% utilization

Figure 7: Idle time for given switch times (4 tasks)

1. The deadlines are constrained but not implicit, which means that jobs having a very short deadline can be released at the same time and will not have time to finish before their deadline, as shown in Figure 5. This explains why some tests where the utilization factor is 90% and without any switching time are failing.

2. When the switching time increases, time is lost between tasks. The effect is even more present as the number of task increases, because there will be more context switches. In Figure 6, failures caused by the switching time are highlighted in bold.

3.2 Idle time

Figure 7 shows that the idle time of a system has a maximum (the plateau in a)) that corresponds to $100\% - U$ where U is the utilization factor. This means that if there is no task switch (or task switches with a switching time of 0), all the time slots that will be used by the jobs will be used to execute them, and the idle time will be the remaining time slots.

If the switching time increases, the idle time decreases because time slots will be consumed by task switches. It is interesting to note that a low utilization factor requires that the switching time becomes high enough before showing signs of switching activity. This comes from the fact that context switches are rare when the utilization factor is low: a job usually has time to finish before another one acquires a greater priority and requires a task switch.

4 Conclusion

This report presented the implementation of a simulator for the EDF scheduling algorithm (along with other ones used for testing and for generating graphs for the syllabus of Operating Systems). Three tools have been developed: one that performs the actual simulation, one that generates systems of a given number of tasks and total utilization factor, and one that helps generating graphs and tables as shown in the previous section.

Some tests followed, and they have shown that constrained deadlines and the switching time can impair the ability of EDF to schedule any system having an utilization factor lower than or equal to 100%. The effect of the switching time on the idle time has also been studied and has shown that an increased switching time lowers the idle time.