INFOF404 : Uniprocessor scheduling project

Imane Moussaoui 496221

November 13, 2022

1 Introduction

For this project, we have to implement two kinds of scheduler : a rate monotonic and an earliest deadline first. Also, we only consider synchronous implicit deadline task sets.

We have to implement a tool to visualize the execution of a task set and we also need to show the deadline misses. For that, I used a package in python which is matplotlib.

In this report, we are going to talk about the difficulties of the project, the implementation that I choose and show some graphical examples of rate monitonic and earliest deadline first.

2 How to run the program

The program is writing in python and it accepts two mandatory arguments : the name of the algorithm you want to use and the name of the task file. You have to use this following command :

python3 project.py rm|edf $< task\_file >$

3 Difficult during this projet

The first difficulty of this project was to find the first task to execute. For example, if we have this following file:

2 4
3 6

First, I looked at the number of jobs that a task needed then I created a list that was going to contained all the tasks but the tasks all had the same size in this list which was 1. For example, the first task in the file has 2 WCET so in my list this task will be indicated twice with size 1 :

listTasks = [(T1, 1, 0, 4), (T1, 1, 0, 4)]

We can see it's a list of tuple with four element : the first element is the name of the task, the second element is the size of the task, the third element is when the task can be start and the last element is when the task has to finish.

With this solution, it was easy to start to find the priority for the two algorithm, we will start with the rate monotonic because it's not the same priority with the earliest deadline first.

For the rate monotonic, the priority depends on the task which will have the smallest period so the first task is the task with 2 WCET and period 4. My solution was to order the list with respect to the smallest period. I created another list because I don't use the listTasks list for the tasks which can be executed before a time x. So with the example, we will have :

ongoingList = [(T1, 1, 0, 4), (T2, 1, 0, 6)]

After sorting the list according to the period, we will execute the first element of the list which will be : (T1, 1, 0, 4).

For the earliest deadline first, it's a little be more complicated to handle the priority because it assigns priorities to the task according to the absolute deadline. The task whose deadline is closest gets the highest priority. When we have two task with the same priority, we can have a lot of possibility to handle this but I choose to order by index when I have this case.

Another problem to handle was to verify if there is a deadline miss and for that I was to check if there was a task in my ongoing list that had not been done before the time x.

4 Implementation

I noticed that both algorithm are similar but were very different in terms of the priority so I created three class : a class contain the similitude with the both algorithm and two class (a RM and an EDF class) which contain the different. Also, a tool class.

For the feasibility, when the $U(t) < 1$ I don't use the same formula for the both algorithm. I use the worst case for the rate monotonic :

$$\begin{cases} w_0 = C_i \\ w_{k+1} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{w_k}{T_j} \right\rceil \cdot C_j \text{ till } w_{k+1} = w_k \text{ or } w_k > D_i \text{(system is not feasible)} \end{cases}$$

and this is the formula for the earliest deadline first :

$$\begin{cases} L_0 = \sum_{i=1}^{n} C_i \\ L_{k+1} = \sum_{i=1}^{n} \left\lceil \frac{L_k}{T_i} \right\rceil \cdot C_i \end{cases}$$

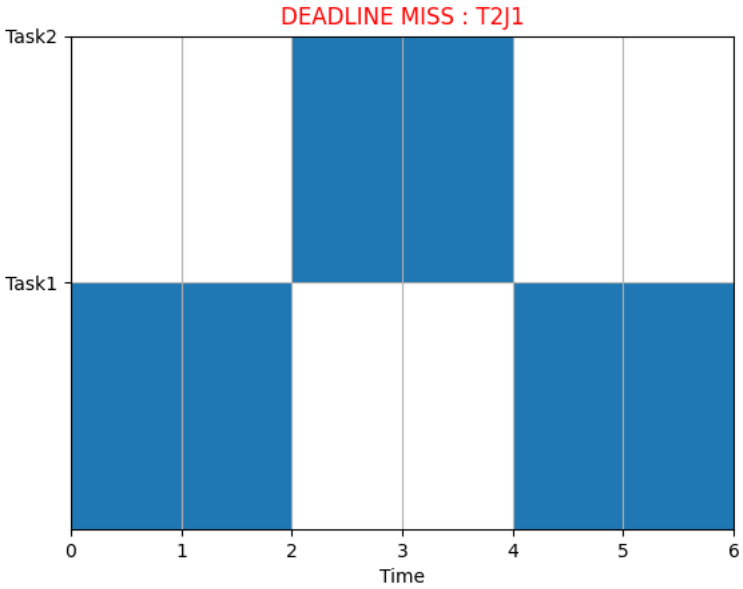
When the $U(t) > 1$, we know there is a deadline miss then I used the max deadline : $[0, maxDi | i = 1, ..., n))$.

5 Example

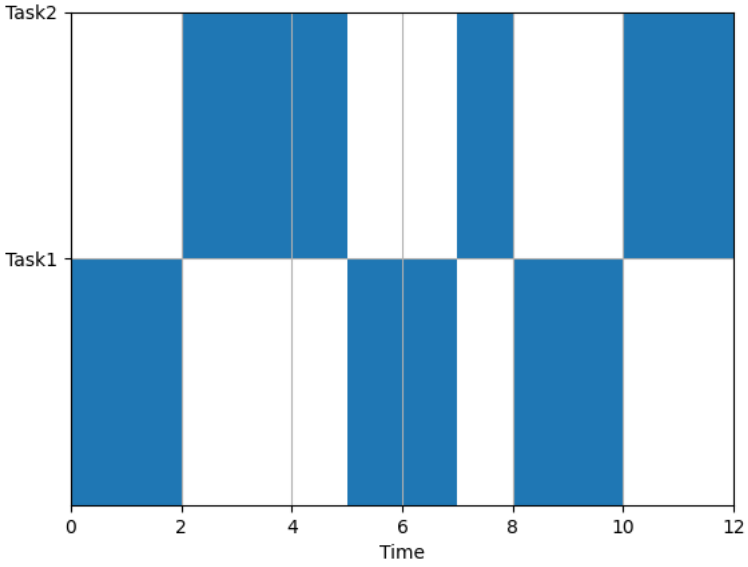
Here is an example of a graph with the following file:

2 4
3 6

For the rate monotonic, we have a deadline miss for task 2 job 1 :



The deadline miss is writing in red at the top of the graph. And for the earliest deadline first, we don't have the deadline miss :



We can see that the earliest deadline first is better here if we want to have a system scheduled.