POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Schedsim

Algorithms for critical real-time systems

A.Y.: 2024/2025

Name/Surname: Bashkim Jançe

Contents

1. Introduction

1.1. Purpose

Schedsim is a program written in python that works as a simulator for task scheduling implementing different algorithms. Till now Schedsim was able to simulate task scheduling using two categories of algorithms:

- Non-pre-emptive: FIFO, SJF, HRRN
- Pre-emptive: SRTF, RR.

The goal of this project was to add another important category of task scheduling algorithms which deals with mixed-criticality systems where the tasks may have two criticality levels: low and high. Precisely it was requested to implement the two state-of-art policies: OBCP(Own Best Criticality Protocol) and EDF-VD(Earliest Deadline First-Virtual Deadline).

1.2. Implemented Functionalities

1.2.1. Front-end

In this section I am going to mention all the added/updated functionalities in front-end of the Schedsim application.

1. New Task: In the New Task section two fields were added to the form used to create a new task. More precisely the field of WCET_HIGH(Worst case execution time at high criticality) was added which has to be greater than WCET. Also, the field Criticality was added to choose between the different

criticality levels that the task will have. Notice that if non-mixed-criticality algorithm is chosen those values are ignored by the scheduler.

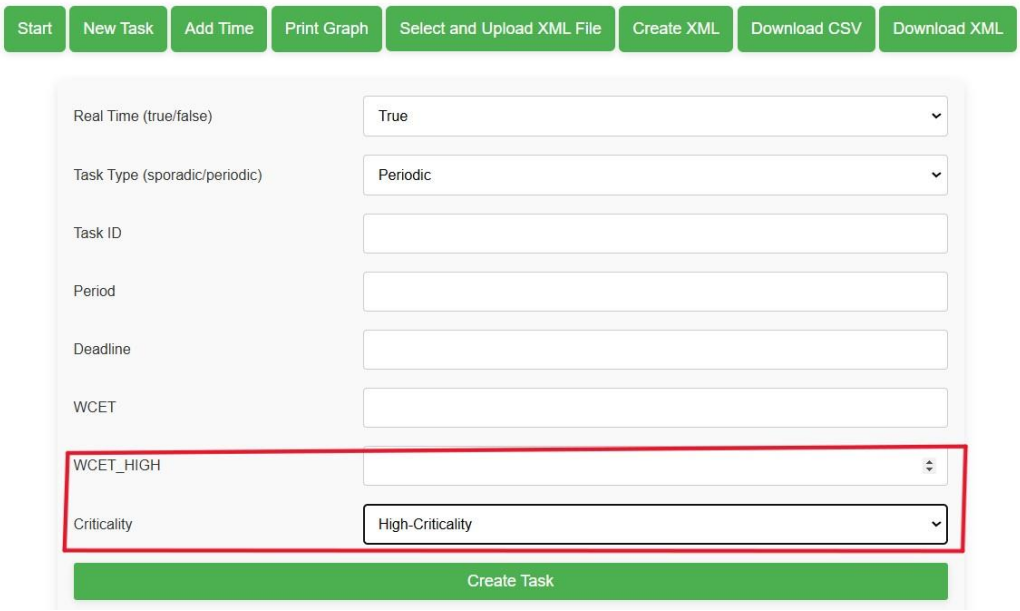


Figure 1

2. Create XML: In this section two additional values were added for the Scheduling Algorithm Field: OBCP and EDF-VD. In case the user chooses between these two values an additional field appear asking the user to choose between the two possible switching-to-low mode policies: Idel and Hyperperiod.

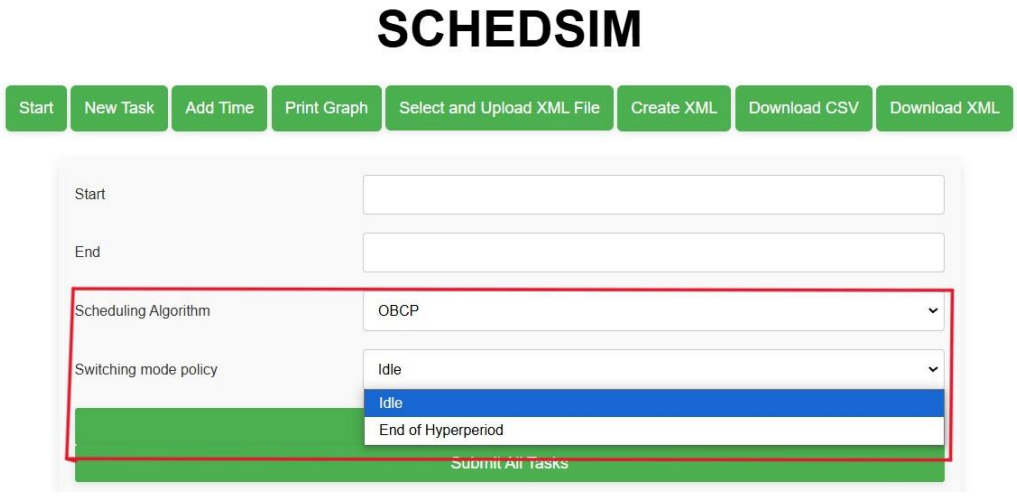


Figure 2

3. Additional Considerations: When choosing between the two possible new algorithms it is possible to have only real time tasks otherwise the system will generate an error while trying to create the XML file. Also, in case we want to execute the EDF-VD algorithm(Hyperperiod policy) the xml file must contain at least one periodic task otherwise and error will occur. This because the hyperperiod is calculated as the LCM of the periods of all the periodic tasks.

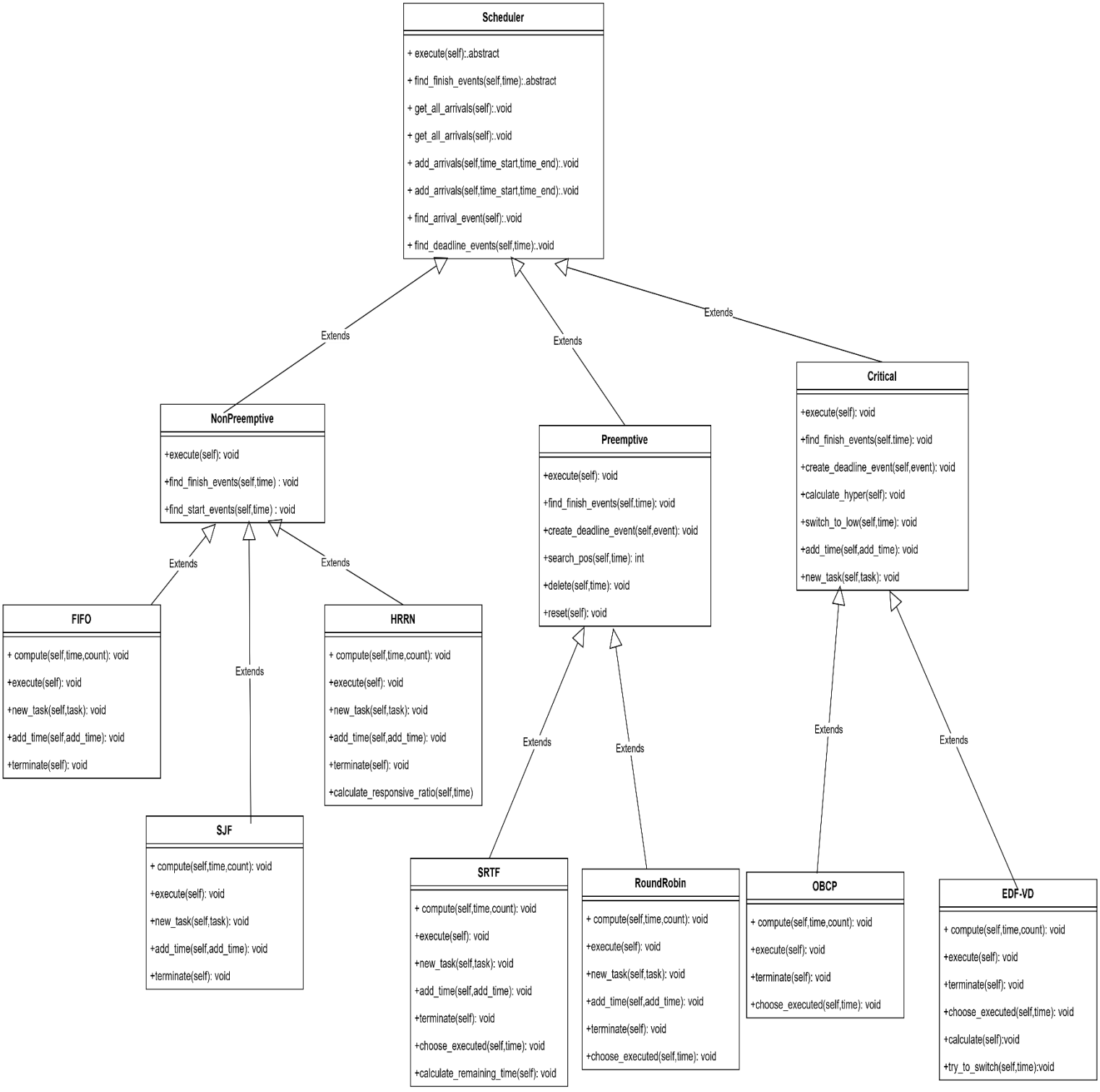
1.2.2. Back-end

In this section I will shortly describe the updated classes of the application:

1. Task: The task object was updated in order to support additional attributes that describe it in mixed-criticality system. More precisely, a wcet_high, criticality and virtual deadline attributes were added.
2. Scheduler: Two additional classes were added in order to support the two new added algorithms to the schedule. For more information on this see the next sessions of this document.
3. SchedulerController: The method create_task was fixed. Actually, it had a bug from the previous branch of the project. Every time this method was called to create a new periodic task, the task itself was being created with a deadline of 0 and to the value of wcet was being assigned the value of the deadline of the task inserted by the user.
4. Additional changes were made to support the parameter control during the creation of a task or a xml file.

2. Algorithms' Description

2.1. UML Diagram



As we can see in the previous page the UML class diagram represent the domain of our application. This project in fact, took part in the implementation of the Critical class extending the Scheduler one. On the other hand, the Critical class itself has extended by the OBCP and EDF-VD classes that implement the corresponding algorithms.

2.2. OBCP Algorithm

In this section I will describe how the implemented OBCP algorithm works in order to make things clearer. So, the scheduler will work in two possible modes: in high and low mode.

Low Mode:

In this mode the scheduler sorts the arrival tasks based on their absolute deadline. This is done in order to have a fair criterion that schedules the corresponding tasks of high and low criticality. The low-criticality tasks will be executed until completion or until the wcet is met and after that the finishing event for the task is being created.

Transition Low->High Mode:

Tasks with high criticality are the ones that decide the transition. A high-criticality task will be executed until completion and if the wcet is met the Scheduler will pass from low mode to high mode. The current high criticality task running will remain in execution.

High Mode:

In this mode the scheduler tasks are sorted based on their criticality so tasks with high criticality will be always executed before the ones with low criticality. Also, the high criticality tasks are left to execute until wect_high and only after they reach this level the corresponding finish event will be created.

Transition High->Low Mode:

The transition from high to low mode is based on the two main policies: the Idle and Hyperperiod one. In the idle policy if no current task is in execution the scheduler automatically passes in low mode. Meanwhile in the Hyperperiod policy every time the

end of an hyperpriod arrives and if no high criticality task is running the scheduler passes to low mode. Also, if no remaining high criticality tasks are left the scheduler switches to low mode.

2.3. EDF-VD Algorithm

In this section the EDF-VD algorithm is going to be described.

Low-Mode:

In this mode the scheduler sorts the arrival tasks based on the virtual deadline parameter. This parameter is calculated for each task at the initial execution of the algorithm. Below the formula for the virtual deadline is given.

Low criticality Tasks: $VD = D$ where VD = virtual deadline and D = deadline

High criticality Tasks: $VD = x*D$

$$x = \min (1, (1-UL) /ULH)$$

$$UL = \sum \text{task.wcet / task.deadline where task.criticality = low}$$

$$ULH = \sum \text{task.wcet / task.deadline where task.criticality = high}$$

The low criticality tasks act in the same way as in the OBCP algorithm. The only difference is that EDF-VD has a preemptive nature. So, if an arrival task has a lower VD in respect to the one executing the tasks switch between them.

Transition Low->High mode:

The transition happens every time a high criticality task meets its wcet_high. Similarly to the previous algorithm the current high critical task remains in execution.

High Mode:

The tasks with high criticality are prioritized compared to low ones. The high-criticality tasks on the other hand are sorted between them using the deadline parameter. In this case the algorithm has a non-preemptive nature.

Transition High->Low mode:

The transition is the as the one described for the OBCP algorithm using two policies chosen by the user.

3. Conclusion

So, the implementation of these two algorithms gives the possibility to visualize and to better understand how they really work. By using the Schedsim application now it is possible to create task with mixed criticality and execute the OBCP or EDF-VD algorithm and see in detail how they schedule the execution of each task. For future implementation of other algorithms similar to these two the application certainly has a design that will allow to add new algorithms of this kind. For more details on the implementation, I will recommend directly inspecting the code and the commented documentation of each method or class.