CRENN Romain

Master 2 LSE
Année universitaire 2020-2021

Monitoring des événements d'ordonnancement sur RTEMS

Projet Logiciels pour les Systèmes Embarqués

Summary

Introduction

The project

The goal of this project is to develop an algorithm which adapts to different events that occur on a device. For example, if a task takes longer than expected to complete, what should the system do. For this project, we focused on the embedded system RTEMS. RTEMS is a real-time operating system designed for embedded systems.

Technology used

To develop and test our algorithms we used Cheddar, Cheddar is an open-source real-time scheduling simulator. It helped us to quickly prototype our new scheduling policies.

State of the art

We began this project by reading different papers on how we could implement these algorithms. First few papers allow us to narrow different parameters we could survey in order to adapt scheduling.

Three parameters came out of these papers, the first one was the period with the Elastic model (“Elastic Task Model For Adaptive Rate Control” Giorgio C. Buttazzo, Giuseppe Lipari, and Luca Abeni). The principle of this model is that we have three periods T0 basic period, the one that will be used the most, Tmin the minimum period of time a task can be fulfilled and finally Tmax, if the task does not complete during this time it would fail.

The second parameter is tasks itself, with the paper Skipover (“Skip-Over: Algorithms and Complexity for Overloaded Systems that Allow Skips” Gilad Koren and Dennis Shasha). In this paper, they explain that their algorithms allow them to skip a task if it could fail.

And the last parameter is the worst case execution time (WCET) with these algorithms EDF-VD and AMC. If a task does not respect its low WCET then, it turns into a high criticality task and it becomes priority.

EDF-VD

EDF-VD (“The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems” S. Baruah, V. Bonifaci, G. D’Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster and L. Stougie) is based on well known EDF algorithm in which we add dual criticality (low and high). This algorithm shows great performance as it has an optimal speed (4/3 of an oracle algorithm) and a complexity of EDF, which is nlog2n.

AMC

AMC (“Response-Time Analysis for Mixed Criticality Systems” S.K. Baruah, A. Burns and R.I. Davis) is an algorithm in which we have a set of tasks with a fixed priority. If a task comes to fail its WCET then, its criticality grows and it becomes a priority.

Development of algorithms

EDF-VD

I started by implementing EDF-VD as it seems the easiest one for me as I had already worked with EDF. Furthermore, EDF is already implemented inside Cheddar, but I had to adapt the algorithm so it uses criticality. For this, I added a low WCET parameter inside the task model. And then compare this new WCET with the current time to react if a task exceeds this time. Then, I had to recreate a dynamic deadline (dynamic deadline is equal to wake up time plus deadline of the task) and this was the most difficult part of the project. I had to modify the parent package of EDF-VD (mixed criticality scheduler) to understand how the scheduler class worked.

```
-- We check if there is a task in critical mode
                        if (is_critical) then
-- We check if the task we are looking is critical
                           if (si.tcbs (i).tsk.criticality = 2) then
-- We calculate the smallest deadline between critical tasks
                              if
                                 (mixed_criticality_tcb_ptr (si.tcbs (i))
                                    .dynamic_deadline <
                                  smallest_deadline)
                                    then
-- We elect the task with the smallest deadline
                                       smallest_deadline := mixed_criticality_tcb_ptr (si.tcbs (i)).dynamic_deadline;
                                       elected := i;
                                       si.tcbs (i).tsk.criticality := 1;
                                    end if;
                           end if;
                        else
-- Same thing as above but there is no critical task
                           if
                              (mixed_criticality_tcb_ptr (si.tcbs (i))
                                 .dynamic_deadline <
                               smallest_deadline)
                           then
                              smallest_deadline :=
                               mixed_criticality_tcb_ptr (si.tcbs (i))
                                  .dynamic_deadline;
                              elected := i;
                           end if;
                        end if;
  -- We check if a low WCET of a task exceed its time
          if ((si.tcbs (i).tsk.capacity_low + si.tcbs (i).wake_up_time) > current_time) then
                 is_critical := True;
                 si.tcbs (i).tsk.criticality := 2;
          end if;
```

AMC

AMC was easier as I took the logic of EDF-VD and adapted it to a fixed priority algorithm. I check which task has the highest priority and elect it. Priority was already implemented in the task model.

```
-- We check if there is a task in critical mode
                if (is_critical) then
-- We check if the task we are looking is critical
                if (si.tcbs (i).tsk.criticality = 2) then
                  if (si.tcbs (i).wake_up_time <= current_time) and
                    (si.tcbs (i).rest_of_capacity /= 0)
                  then

                     if (options.with_jitters = False) or
                       (si.tcbs (i).is_jitter_ready)
                     then

                        if (options.with_offsets = False) or
                          check_offset (si.tcbs (i), current_time)
                        then

                           if (options.with_precedencies = False) or
                             check_precedencies (si, current_time, si.tcbs (i))
                           then
                                 if (options.with_resources = False) or
                                    fixed_priority_tcb_ptr (si.tcbs (i))
                                    .is_resource_ready
                                 then
                                    highiest_priority  :=  Natural (fixed_priority_tcb_ptr (si.tcbs
(i))
                                                                    .current_priority);
                                    elected := i;
                                 end if;
                           end if;
                        end if;
                     end if;
                  end if;
               end if;
```

Results

Results I obtained where satisfying as they corroborate with the ones I made before starting the project.

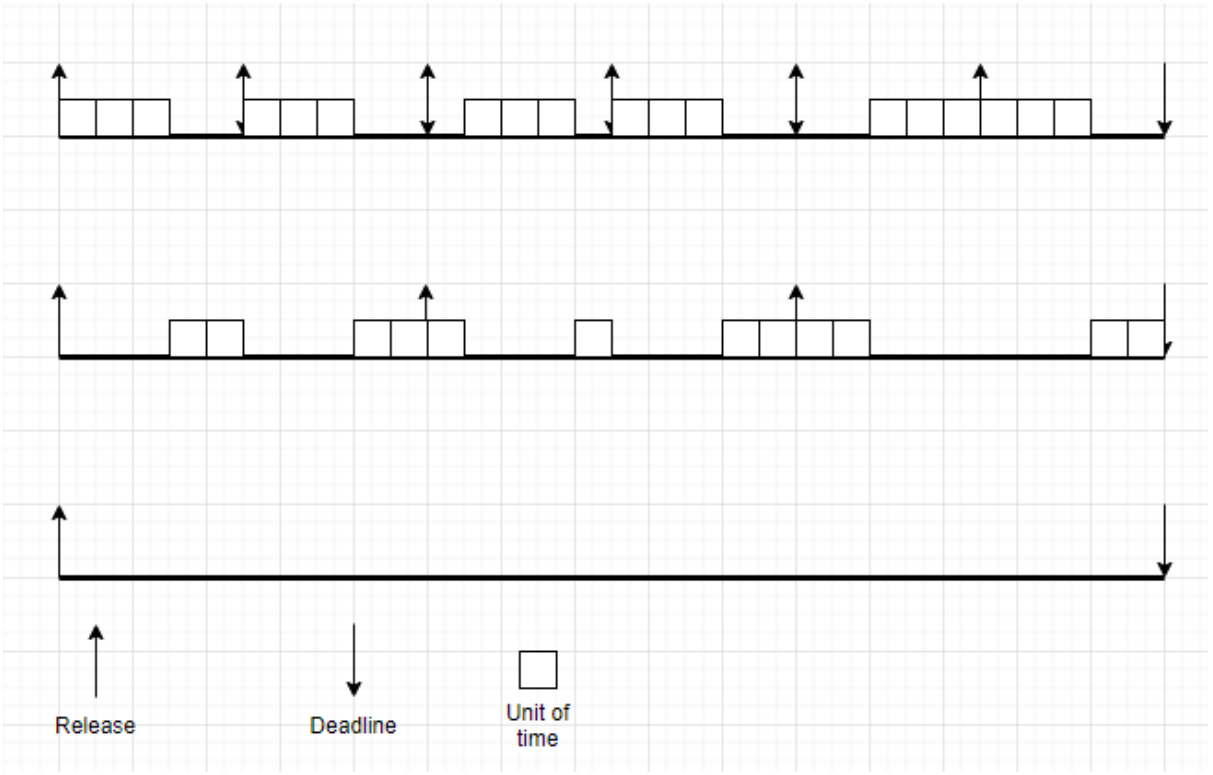
Some example :

I took this set of task :

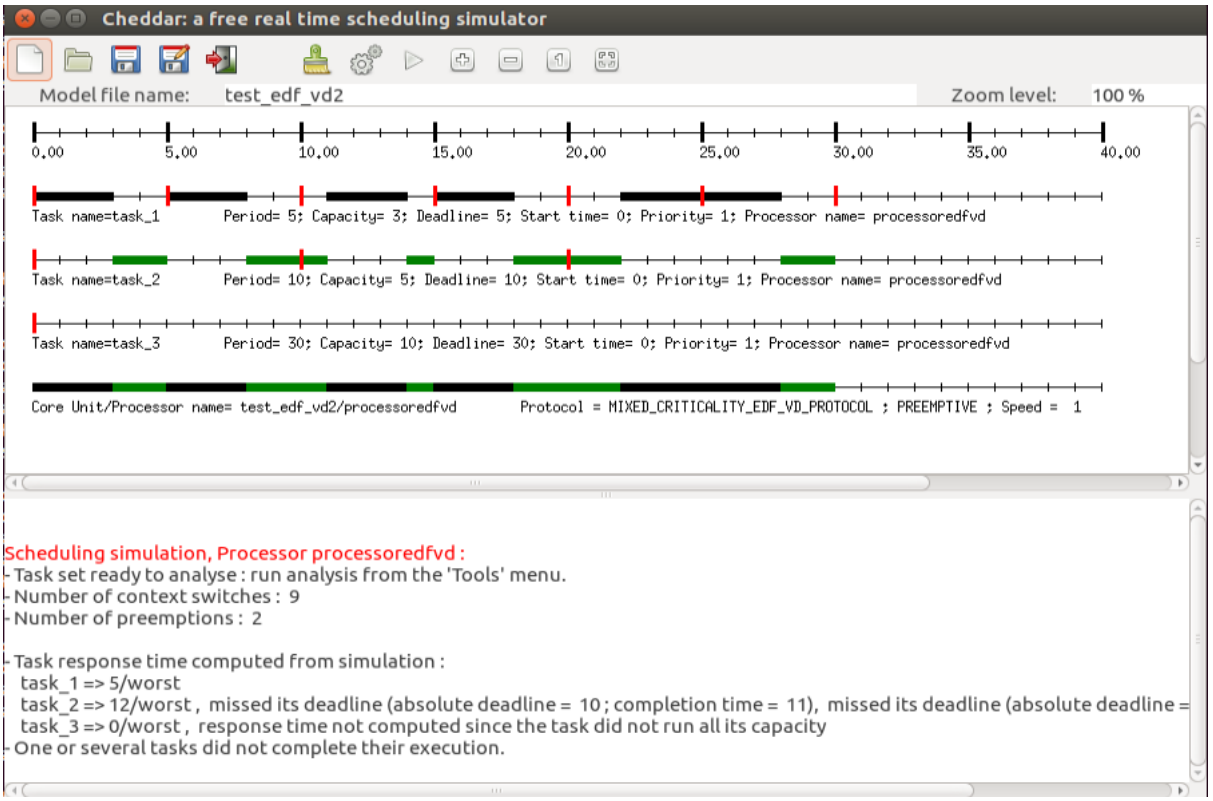
Task	Release time	Deadline	Criticality	ci(LO)	ci(HI)
T1	0	5	-	1	3
T2	0	10	-	2	5
T3	0	30	-	9	20

We assume that T1 takes 2 units of time T2 5 and T3 9.

The model I made before the project is as follow :



And the results from the algorithm is :



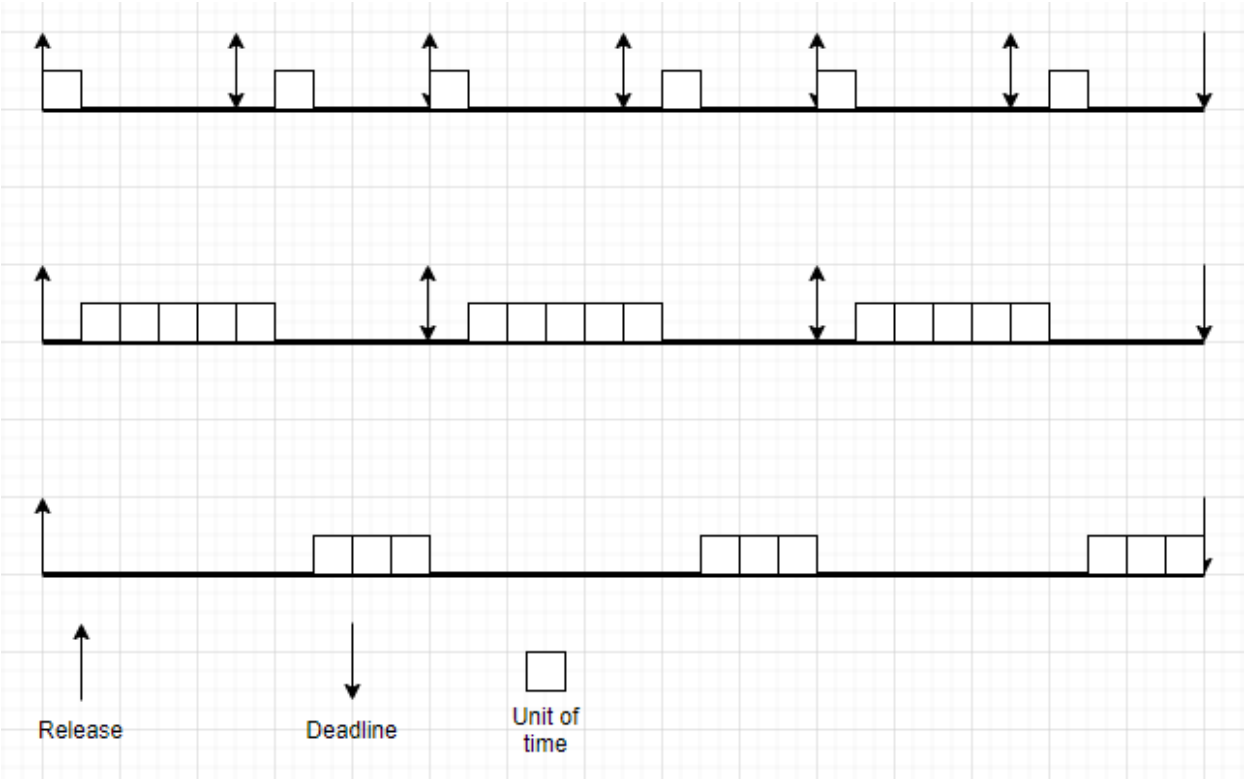
We can see that the algorithm respects the rules of EDF-VD.

For AMC I used this set of task :

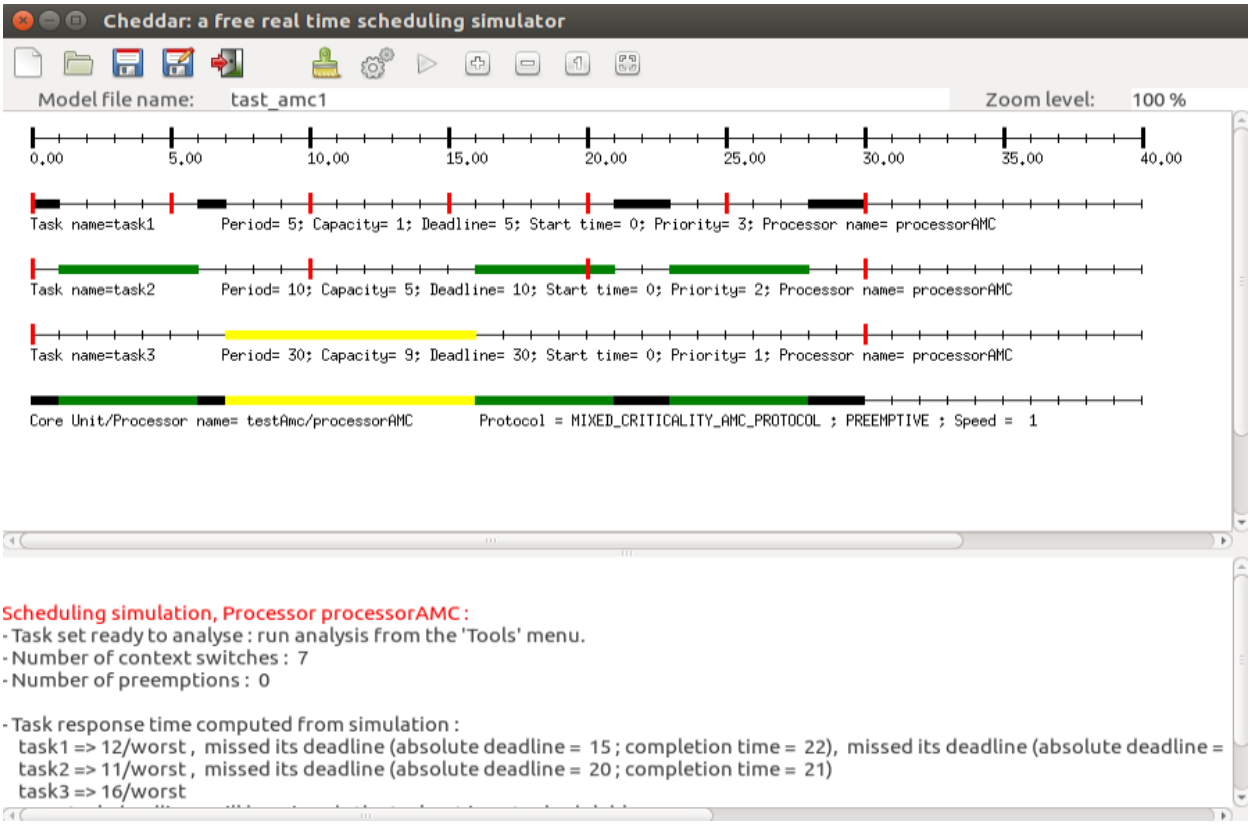
Task	Period	Deadline	ci(LO)	ci(HI)	Priority
T1	5	5	1	-	3
T2	10	10	2	5	2
T3	30	30	9	20	1

We assume that T1 takes 1 unit of time, T2 5 and T3 9.

The model I elaborate :



And the result from Cheddar :



As we can see, my algorithm does not work as intended. I think what is wrong is that the WCET fails to start at the right time and therefore tasks trigger the high criticality too early. I tried to correct that but I was not able to.

Conclusion

To conclude this project was really a challenging experience. Firstly, because it was interesting to implement these algorithms and I had to learn how to use new tools. Secondly, during this time I had to work remotely and I find it really difficult to stay involved when you do not have a work environment to rely on.

If I had a criticism to provide on my work, I would say that I did not put enough energy into it. I could have done it better, my code is not optimal and it could be cleaner.