Embedded Systems - Lab Report

Axel Alvin, Lukas Axelborn

October 21 2021

1 Introduction

In this assignment the group were to implement a set of scheduling algorithms including regular Rate Monotonic and Earliest Deadline First and also the corresponding algorithms but with a hard deadline implementation. The assignment folder contained a set of object files which included functions to simulate tasks in a processor. A skeleton code for the students to work from were also included.

The task was then to implement the different algorithms in the functions provided and then test them by comparing the output to the correct output-files provided in the project folder. If the outputs matched the algorithm could be considered correct. When all four were implemented, presumably in a correct way, they were to be implemented and ran in a different program and then be analyzed given the different outputs.

2 Analysis

For the analyse a Python program was implemented which was recommended in the lab specification. In this program the output files from the additional programs were to be analyses to determine why the tasks in some of these programs did not run properly. Firstly though, it had to be determined what the problems in these programs were, as part of the assignment. Just by looking at the output files with the eyes, it could be noticed that, once all tasks had finished running, the total time were not the same in all four. The first two programs seemed to run perfectly with exactly 20 000 time-clicks passed. The last two were a bit different though, seemingly passing the total deadline of the program by a few clicks, indicating something went wrong.

Now, having some idea of what to look for, the Python program with some functions calculating some averages for the different programs was implemented, which were as follows;

First, looking at the average processing time there were no significant differences. Looking at response time, it showed, basically the same. Both algorithms in program 2 showed a lower response time compared to program 1 but since both differs the conclusion should be that the programs themselves were different. Although when looking at the average time a process was late, there were some interesting results. For the first two, program 1 algorithm EDF and RM the avg late time was 0 and also, the counter indicating number of late tasks showed 0. But for program 2, RM and EDF, it showed that some tasks were

late. When looking at the counter, it showed that most of the tasks were late many times. Why is this though?

To figure this out, we started looking at the actual output to see if there were any irregularities to be spotted. And pretty quickly, we noticed that task one looked a bit weird. It seemed that it got a set processing time every time it arrived, but when it finished running the observed processing time many times (if not every) differed from the stated processing time. Looking at documentation for RM and EDF, this is a problem. In the documentation, one of the demands for these algorithms is that "every process must require the same amount of CPU time on each burst". To confirm this suspicion, a function to calculate the difference between the stated processing time and the observed processing time was implemented to get it in to numbers. This showed, as suspected that there were no differences in the first programs, but in program two the difference was present. Looking at the results from this diff analysis, task 1 requires a different amount of CPU time at each burst, making this algorithm unstable in some way. To get an even deeper understanding a new function was implemented that calculates the average observed proc time for each tasks and sure enough, the result for task 2 in RM and EDF program 2 shows that the observed proc time was 19.512 and not 20, as was stated.

3 Results

analysis.txt

```
Processing time:
Avg proc time for edf 1:        17.78
Avg proc time for  rm 1:        17.78
Avg proc time for edf 2:        17.78
Avg proc time for  rm 2:        17.78

Response time:
Avg resp time for edf 1:        34.44
Avg resp time for  rm 1:        35.56
Avg resp time for edf 2:        36.40
Avg resp time for  rm 2:        40.09

Lateness
Avg  lateness  for edf 1:         0.00
Task 0 was late 0 times for edf 1
Task 1 was late 0 times for edf 1
Task 2 was late 0 times for edf 1
Avg earliness for edf 1:        18.89


Avg  lateness  for  rm 1:         0.00
Task 0 was late 0 times for  rm 1
Task 1 was late 0 times for  rm 1
Task 2 was late 0 times for  rm 1
Avg earliness for  rm 1:        17.78


Avg  lateness  for edf 2:         1.81
Task 0 was late 43 times for edf 2
Task 1 was late 89 times for edf 2
Task 2 was late 26 times for edf 2
Avg earliness for edf 2:        18.73
```

```
Avg  lateness  for  rm 2:        10.27
Task 0 was late 0 times for  rm 2
Task 1 was late 19 times for  rm 2
Task 2 was late 87 times for  rm 2
Avg earliness for  rm 2:        23.52



Obs_proc time proc_time diff
Total diff for task 0 is 0 for edf 1
Total diff for task 1 is 0 for edf 1
Total diff for task 2 is 0 for edf 1


Total diff for task 0 is 0 for rm 1
Total diff for task 1 is 0 for rm 1
Total diff for task 2 is 0 for rm 1


Total diff for task 0 is 0 for edf 2
Total diff for task 1 is 2350 for edf 2
Total diff for task 2 is 0 for edf 2


Total diff for task 0 is 0 for rm 2
Total diff for task 1 is 2350 for rm 2
Total diff for task 2 is 0 for rm 2



Avg obs proc time
Avg obs proc time for task 0 is 10.0 for edf 1, stated: 10
Avg obs proc time for task 1 is 20.0 for edf 1, stated: 20
Avg obs proc time for task 2 is 40.0 for edf 1, stated: 40


Avg obs proc time for task 0 is 10.0 for rm 1, stated: 10
Avg obs proc time for task 1 is 20.0 for rm 1, stated: 20
Avg obs proc time for task 2 is 40.0 for rm 1, stated: 40


Avg obs proc time for task 0 is 10.0 for edf 2, stated: 10
Avg obs proc time for task 1 is 19.512 for edf 2, stated: 20
Avg obs proc time for task 2 is 40.0 for edf 2, stated: 40


Avg obs proc time for task 0 is 10.0 for rm 2, stated: 10
Avg obs proc time for task 1 is 19.512 for rm 2, stated: 20
Avg obs proc time for task 2 is 40.0 for rm 2, stated: 40
```

---

4 Conclusion

Our results show that in program 2, both for RM and EDF, task 2 had a diff when looking at the observed processing time compared to the stated processing time, causing lateness in almost all tasks during program 2. So why is this a

problem? Our theory is that when the task is scheduled by it's processing time and that processing time is not correct, this in turn leads the algorithm prioritization being out of line. Say, for instance, that task 1 has 10 clicks left to process, and it has 12 clicks left before it's deadline. Then, the program will run it thinking that it is still time. When the real processing time then is, say 15, it runs late for 3 clicks. This in turn, can lead to the next task to also run late since it has to start late. Because the task deadline is now changed the algorithm prioritisation can be wrong, both for RM and EDF since they both priorities according to task deadline. Judging by the average lateness EDF handles this problem better then RM, 1,81 compare to 10,27 respectively. The reason behind this is because EDF uses a dynamic prioritization process, since the absolute deadline depends on the arrival time of the task, and therefore can change dynamically. RM on the other hand uses static prioritisation since the period of a task never changes.

Further analysis on lateness shows that the tasks are not late often, considering all tasks run hundreds of times. This makes sense because most of the times it is not a problem for the program, since the tasks periods are much longer than the actual processing time, meaning there are some space for errors. The same can not be applied to a task that finishes earlier than it should, which happens more often according to the avg observed processing time being less than 20, since this will never have an impact on the task deadline. The average observed time calculated is a misleading indicator, because even if more tasks are early then late, it is the late ones that has an effect on the deadline and thereby prioritisation.