Individual Assignment 3

May 2019

1 Instructions

Full solutions should be turned in as a zip-file this involves the written solutions as a pdf. You turn-in the solutions in Ping-Pong. The solutions can be hand-written or generated in LATEX/Word. The solutions should be complete and well explained.

2 Problem 1

Consider two tasks to be executed periodically on a single processor, where task 1 has period $p_1 = 4$ and and task 2 has period $p_2 = 10$. Assume task 1 has execution time $e_1 = 1$, and task 2 has execution time $e_2 = 7$. The deadline for each task is equal to the period of the same task.

a) Sketch a rate-monotonic schedule (for 20 time units, the least common multiple of 4 and 10).

b) Now suppose task 1 and 2 contend for a mutex lock, assuming that the lock is acquired at the beginning of each execution and released at the end of each execution. Also, suppose that acquiring or releasing locks takes zero time and the priority inheritance protocol is used. Is the rate-monotonic schedule feasible, i.e. are the deadlines always met?

c) Assume still that tasks 1 and 2 content for a mutex lock, as in part (b). Suppose that task 2 is running an *anytime algorithm*, which is an algorithm that can be terminated early and still deliver useful results. Find the maximum value for the execution time e_2 of task 2 such that the rate-monotonic schedule is feasible.

d) For the original problem, where $e_1 = 1$ and $e_2 = 7$, and there is no mutex lock, sketch an EDF schedule for 20 time units. For tie-breaking among task executions with the same deadline, assume the execution of task 1 has higher priority than the execution of task 2. Is the schedule feasible?

e) Now consider adding a third task, task 3, which has period $p_3 = 5$ and execution time $e_3 = 2$. In addition, assume as in part c) that we can adjust execution time of task 2. Find the maximum value for the execution time e_2 of task 2 such that the EDF schedule is feasible and sketch the schedule for 20 time units. You may assume that the execution times are always positive integers. For tie-breaking among task executions with the same deadline, assume task i has higher priority than task j if $i < j$.

3 Problem 2

Consider a system with two tasks, T_1 and T_2, and a single shared resource. Both tasks perform different activities but every now and then wants to use the shared resource. We have the following atomic propositions defined, $i \in \{1, 2\}$.

- T_i.request
- T_i.use
- T_i.release

Specify in Linear Temporal Logic (LTL) the properties below. Model the specifications using the following LTL temporal operators `ALWAYS` ($\Box$), `EVENTUALLY` ($\Diamond$), `UNTIL` ($\mathcal{U}$), `NEXT` ($\bigcirc$), and the operators from propositional calculus ($\wedge$, $\vee$, $\neg$).

a) Mutual exclusion, i.e., only one task at a time can use the resource.

b) Finite time of usage, i.e., a task can only use the resource for a finite amount of time.

c) Absence of individual starvation, i.e., if a task request to use the shared resource it will eventually be able to do so.

d) Absence of blocking, i.e., a task can always request to use the resource.

e) Alternating access, i.e., tasks must strictly alternate in using the resource.