Laboratory 1

Olafur Jonsson Konstantinos Xyderos

September 15, 2020

3.3.1 Periodic tasks in Ada

TODO

3.3.2 The rate monotonic schedule.

1) Calculating utilization

The utilization of a periodic task $\tau_i(T_i, C_i)$ is defined as:

$$u_i = \frac{C_i}{T_i}$$

The total utilization for a set of tasks is defined as:

$$U = \sum_{i=1}^{n} u_i$$

The 3 tasks given are:

$$\tau_i(\phi_i, T_i, C_i, D_i)$$

$$\tau_1(100, 300, 100, 300)$$

$$\tau_2(100, 400, 100, 400)$$

$$\tau_3(100, 600, 100, 600)$$

Their respective utilizations are:

$$u_1 = \frac{C_1}{D_1} = \frac{100}{300} = \frac{1}{3}$$

$$u_2 = \frac{C_2}{D_2} = \frac{100}{400} = \frac{1}{4}$$
$$u_3 = \frac{C_3}{D_3} = \frac{100}{600} = \frac{1}{6}$$

The total utilization U can thus be calculated:

$$U = \frac{1}{3} + \frac{1}{4} + \frac{1}{6} = \frac{9}{12} = \frac{3}{4} = 0.75$$

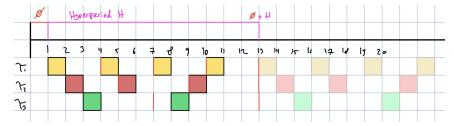
The hyperperiod H is defined as the least common multiplier from the set of task periods:

$$H = lcm(T_1, T_2, T_3) = lcm(300, 400, 600) = 1200$$

According to RMS the priorities should therefore be:

$$P_1 > P_2 > P_3$$

For one hyperperiod of 1200ms the schedule should look like this:



2) Calibrating execution time

Constraining the execution of the program to use only a single CPU core using taskset was made possible on a macOS system by using a virtual machine. The calibration parameter was set to **1208** which produced the following output:

```
vagrant@buster:/vagrant ♪

vagrant@buster:/vagrant ♪

vagrant@buster:/vagrant ↑

taskset -c 0 ./periodictasks_priority

Task 1- Release: 0.100, Completion: 1.094, Response: 0.994, WCRT: 0.994, Next Release: 2.100

Task 1- Release: 2.100, Completion: 5.090, Response: 0.994, WCRT: 0.994, Next Release: 4.100

Task 1- Release: 4.100, Completion: 5.090, Response: 0.990, WCRT: 0.994, Next Release: 6.100

Task 1- Release: 6.100, Completion: 7.092, Response: 0.992, WCRT: 0.994, Next Release: 8.100

Task 1- Release: 8.100, Completion: 9.103, Response: 1.003, WCRT: 1.003, Next Release: 10.100

Task 1- Release: 10.100, Completion: 11.093, Response: 0.993, WCRT: 1.003, Next Release: 12.100

Task 1- Release: 12.100, Completion: 13.094, Response: 0.994, WCRT: 1.003, Next Release: 14.100

Task 1- Release: 14.100, Completion: 15.104, Response: 0.994, WCRT: 1.004, Next Release: 16.100

Task 1- Release: 16.100, Completion: 17.095, Response: 0.995, WCRT: 1.004, Next Release: 18.100

Task 1- Release: 18.100, Completion: 19.094, Response: 0.994, WCRT: 1.004, Next Release: 20.100

Task 1- Release: 20.100, Completion: 21.094, Response: 0.994, WCRT: 1.004, Next Release: 22.100

Task 1- Release: 22.100, Completion: 21.094, Response: 0.994, WCRT: 1.004, Next Release: 22.100

Task 1- Release: 24.100, Completion: 23.099, Response: 0.999, WCRT: 1.004, Next Release: 24.100

Task 1- Release: 24.100, Completion: 27.099, Response: 0.999, WCRT: 1.020, Next Release: 28.100

Task 1- Release: 28.100, Completion: 27.095, Response: 0.995, WCRT: 1.020, Next Release: 30.100

Task 1- Release: 28.100, Completion: 29.092, Response: 0.995, WCRT: 1.020, Next Release: 30.100

Task 1- Release: 30.100, Completion: 29.092, Response: 0.995, WCRT: 1.020, Next Release: 32.100

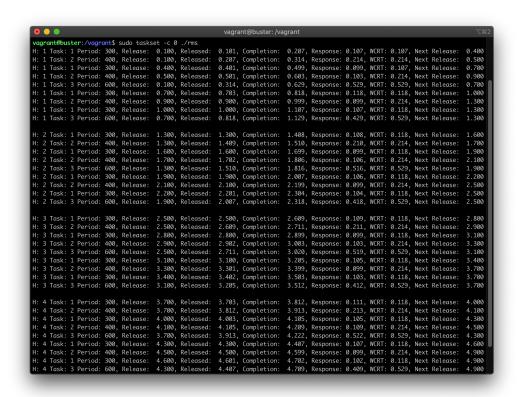
Task 1- Release: 28.100, Completion: 29.092, Response: 0.995, WCRT: 1.020, Next Release: 30.100

Task 1- Release: 30.100, Completion: 31.095, Response: 0.995, WCRT: 1.020, Next Release: 30.100

Task 1- Release: 30.100, Completion: 31.095, Response: 0.995, WCRT: 1.020, Next Release: 30.100
```

3) Implementing the periodic task set Γ_1

Running the rms on a single core produces the following output:



4) Adding the additional task τ_4

RMS2 output:

