Homework 2

FILIP STENBECK 930702-4530
ALEXANDER RAMM 931005-8418

Febuary 2016

Part 1:

Rate Monotonic scheduling

1

Rate monotonic scheduling is a scheduling algorithm with fixed priorities. The highest priority is designated to the task with lowest period T_i.

2

The rate monotonic algorithm is schedulable if the Utilization factor (U) is less than $ln(2)$.

$$U = \sum_{i=1}^{3} \frac{C_i}{T_i} < ln(2)$$

for these tasks with $C_1 = C_2 = C_3 = 6ms$ the utilization factor is 0.6783 which is less than $ln(2)$ thus schedulable.

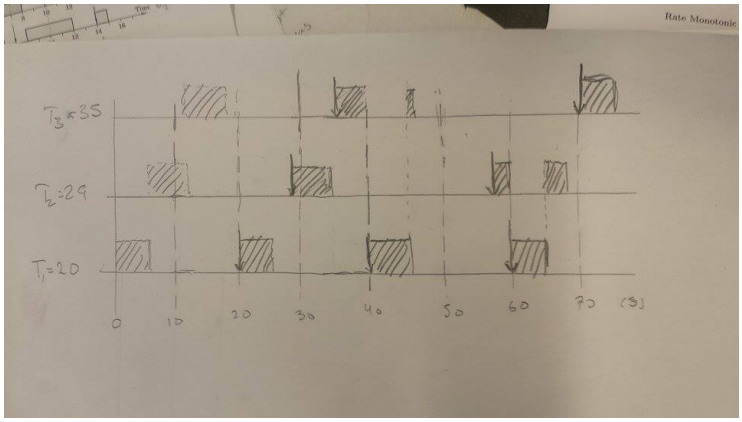


Figure 1: The schedule for the three pendulums, for the first 70 ms.

3

The three pendulums have similar performances but we can see that there is no delay in the pendulum with highest priority (top plot) whilst the two other pendulums have a 6, 12 ms delay since they have lower priority. Due to that the system is schedulable we note that all the task are performed though with a maximum delay of 35 ms for the pendulum with lowest priority.

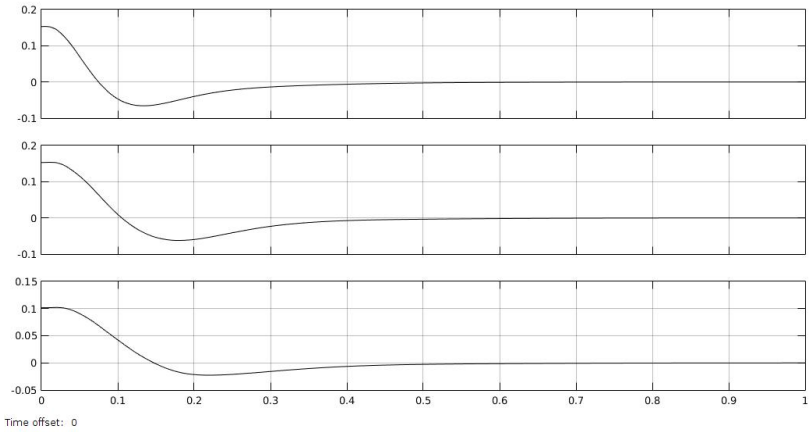


Figure 2: All three pendulums seems to be stable at execution time 6 ms.

4

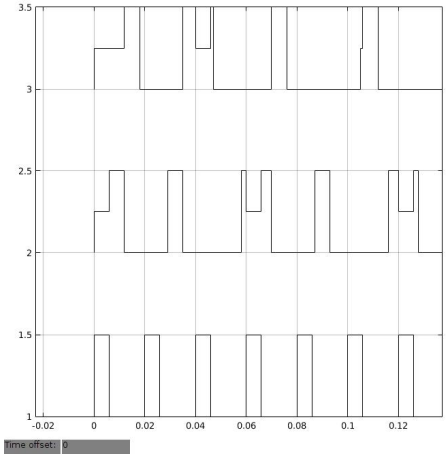


Figure 3: The schedule for the three pendulums, highest priority is the bottom plot, lowest priority is the top plot. As we can see the analytic plot does seem to map to the schedule plot from simulink over the first 80 ms.

5

For the tasks with $C_1 = C_2 = C_3 = 10ms$ the utilization factor is 1.13 which is higher than $ln(2)$ thus not schedulable. Note: Only the two tasks with the highest priority results in a utilization factor of 0.84 which is still not schedulable, though this does not necessarily mean task 2 should be unstable as seen in figure but only that some data loss will happen.

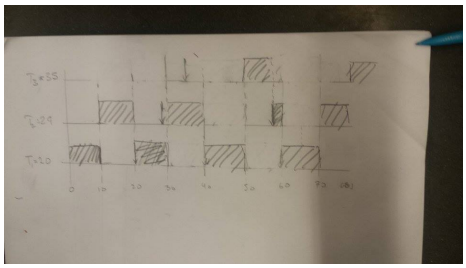


Figure 4: The schedule by hand, RM algorithm.

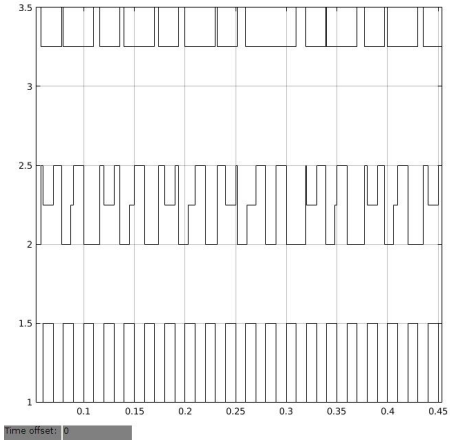


Figure 5: The schedule for the three pendulums execution time 10 ms. We can see that task 3 almost never get executed, though task 2 and 1 always get executed.

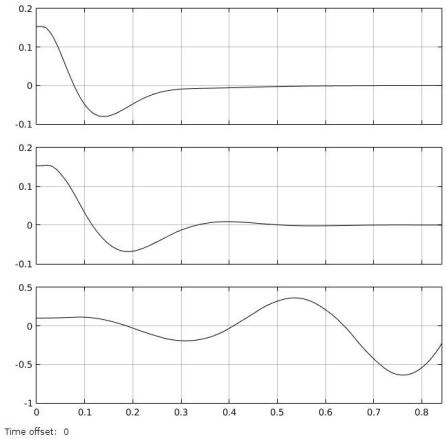


Figure 6: The pendulum with the lowest priority is now unstable at execution time 10 ms.

Earliest deadline first

1

Earliest deadline first (EDF) is a dynamic scheduling algorithm. It calculates the closest deadline in absolute time and sets the priority according to that. This optimizes the computer time to the task in most need of it, thus making systems stable as long as the utilization factor is lower than 1. Since the algorithm is dynamic the computer need to continuously calculate the priorities of the tasks which does take some computation power and is slightly more complex than the predefined priorities from the rate monotonic algorithm.

2

The utilization factor is still $0.6783 < 1$ thus schedulable with this method.

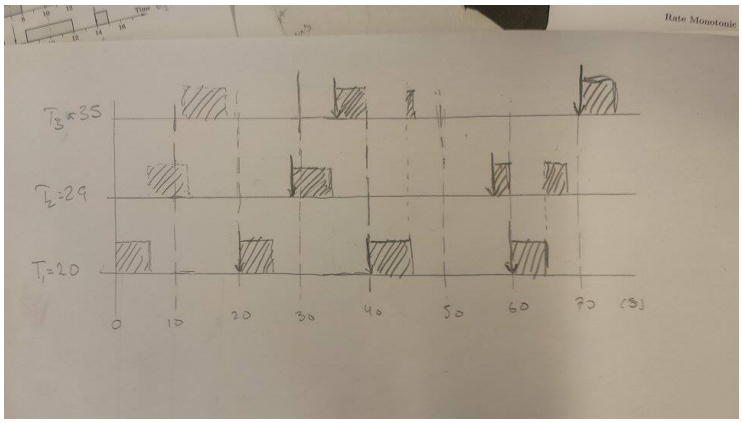


Figure 7: The schedule for the three pendulums, for the first 70 ms. This is just the same as the rate monotonic algorithm for this interval.

3

Just like for the Rate monotonic algorithm for these execution times the three pendulums are stable. The shortest pendulum seem to have a little bit quicker response time. This is due to that it will be executed first at time $0(s)$.

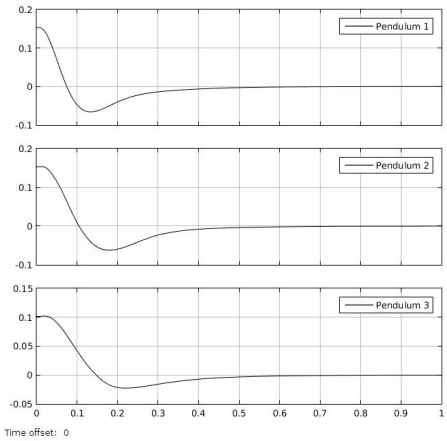


Figure 8: All three pendulums seems to be stable at execution time 6 ms with the EDF algorithm.

4

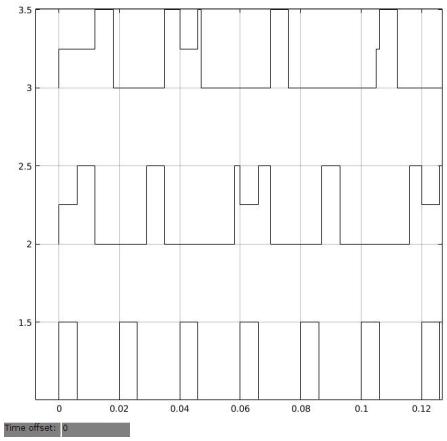


Figure 9: The schedule for the three pendulums,longest pendulum in the bottom plot, shortest in the top plot. As we can see the analytic plot maps to the schedule plot from simulink over the first 80 ms.

5

The utilization factor is still $1.13 > 1$ thus not schedulable with this method.

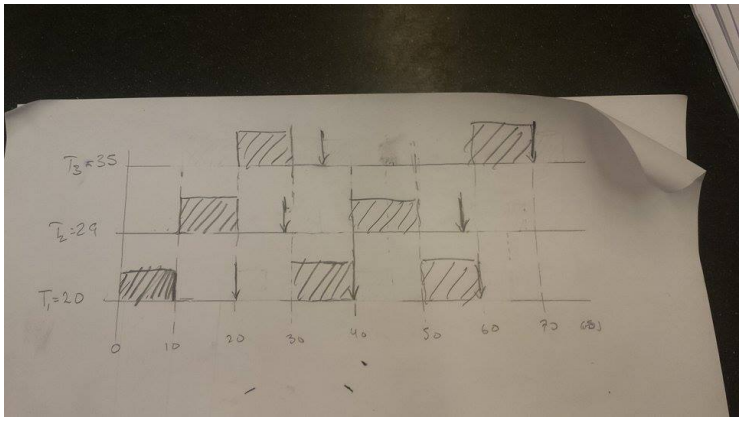


Figure 10: The schedule for the three pendulums, for the first 70 ms. Execution time $C_i = 10$ ms.

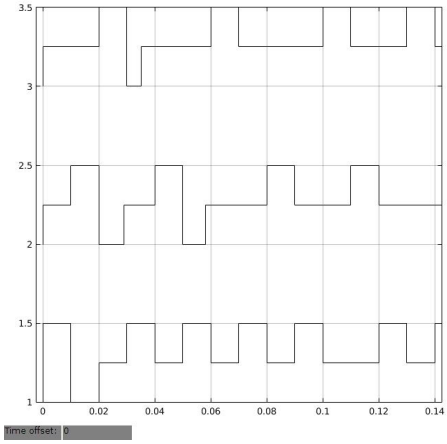


Figure 11: The schedule for the three pendulums, at $C_i = 10$ ms. We see that sometimes the first pendulum does not have the highest priority unlike the RM algorithm

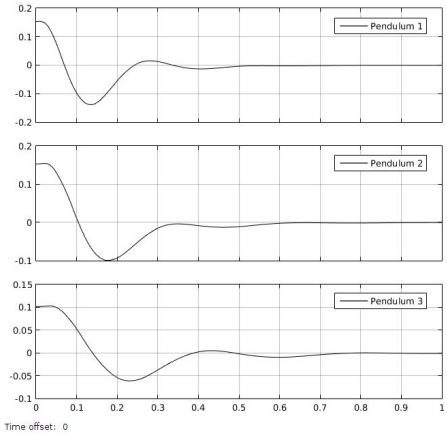


Figure 12: The step response for the three pendulums with execution time 10 ms. We see that all three systems are stable unlike the RM algorithm, though we see that pendulum 1 has far worse performance.

6

in this case the control performances of all of the pendulums were stable for the EDF algorithm, whilst in the RM algorithm only two of the pendulums where stable for the long execution tie case. We would say that in this case the EDF is a better choice. Though keeping in mind that the different algorithms are just ways of distributing the brain power of the computer differently over the tasks. For example if the execution time would be even higher that the EDF algorithm would have none of the systems be stable might be a worse algorithm than having the RM algorithm stabilizing the one pendulum with highest priority.

Part 2:

1. " Calculate analytically the closed-loop equations of the system."

The following network control system is given.

$$\dot{x} = Ax + Bu \tag{1}$$

Where the matrices A, B are set to be 0, I since the system is a simple integrator. The input u is set to be.

$$u = -Kx$$

In the network two delays are introduced. One before and one after the controller. This results in that the input of the system is delayed $\tau_{sc} + \tau_{ea}$ seconds. Since $\tau_{sc} + \tau_{ea} < h$ the input signal can be written.

$$u = \begin{cases} u_0 = -Kx(kh - h) & kh < t < kh + \tau_{sc} + \tau_{ea} \\ u_1 = -Kx(kh) & kh + \tau_{sc} + \tau_{ea} < t < kh + h \end{cases} \tag{2}$$

With this we can formulate the closed loop equation.

$$x(kh + h) = \mathrm{e}^{Ah} + \int_{kh}^{kh+h} \mathrm{e}^{AS}\,\mathrm{d}SBu$$

$$= \mathrm{e}^{Ah}x(kh) + \int_{kh}^{kh+\tau_{sc}+\tau_{ea}} \mathrm{e}^{AS}\,\mathrm{d}SBu_0 + \int_{kh+\tau_{sc}+\tau_{ea}}^{kh+h} \mathrm{e}^{AS}\,\mathrm{d}SBu_1 \tag{3}$$

Setting the time delay $\tau = \tau_{sc} + \tau_{ea}$ and combining the equations 2 with 3 we result in the following closed loop equation.

$$x(kh + h) = \begin{bmatrix} 1 + K(\tau - h) & 0 \\ 0 & -K\tau \end{bmatrix} \begin{bmatrix} x(kh) \\ x(kh - 1) \end{bmatrix}$$

2

Since the continuous plant is a simple integrator $G(s) = \frac{1}{s}$ we get that the zero-order hold of the continuous plant will have the following pulse-transfer function.

$$G(z) = \frac{h}{z - 1}$$

With the controller as $C(z) = -K$ we will have a stable system if.

$$\left|\frac{G(z)C(z)}{1 + G(z)C(z)}\right| < \frac{1}{\tau z}, z \in R$$

This can be written as.

$$hK(1 - z\tau) < z - 1, z \in R$$

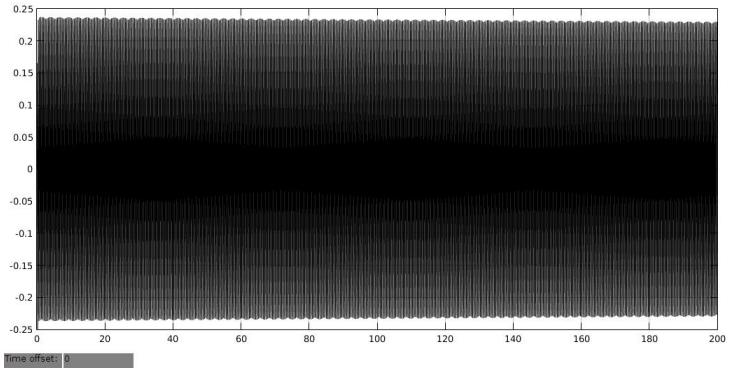


Figure 13: The system becomes unstable approximately at $\tau = 0.0398775$. Length of simulation 200 seconds. (Signal in black on white background.)

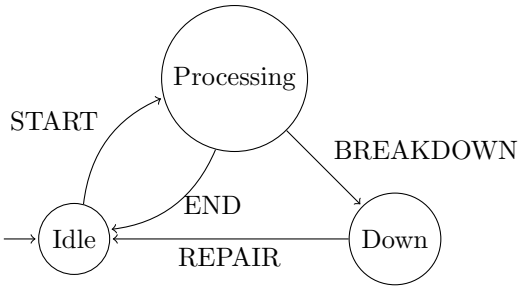
3

The system becomes unstable approximately at $\tau = 0.0398775$. This can we see through the output of the system that is plotted in figure below.

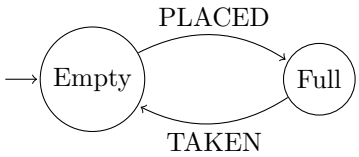
Task 3:

1

the machine M_1 and M_2 have the same statechart model in fig below.

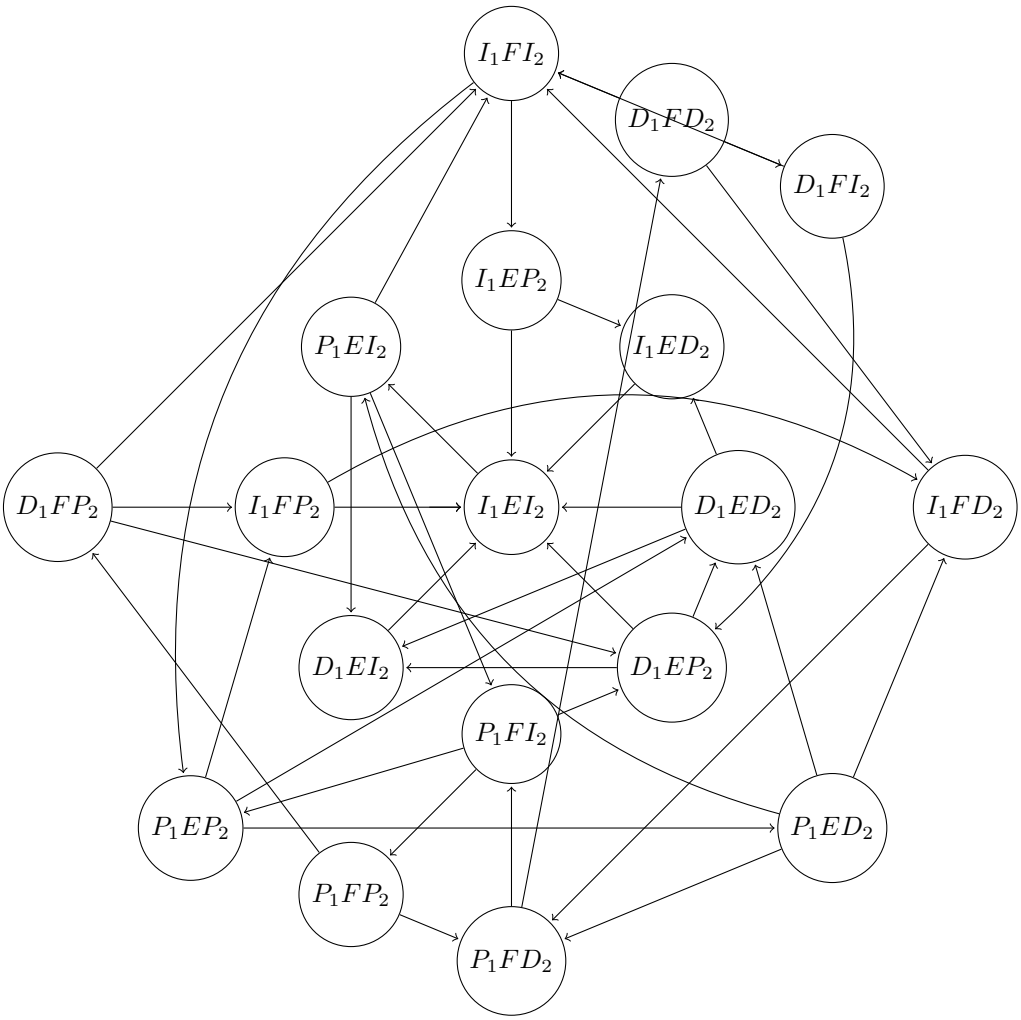


The buffer has only two states:



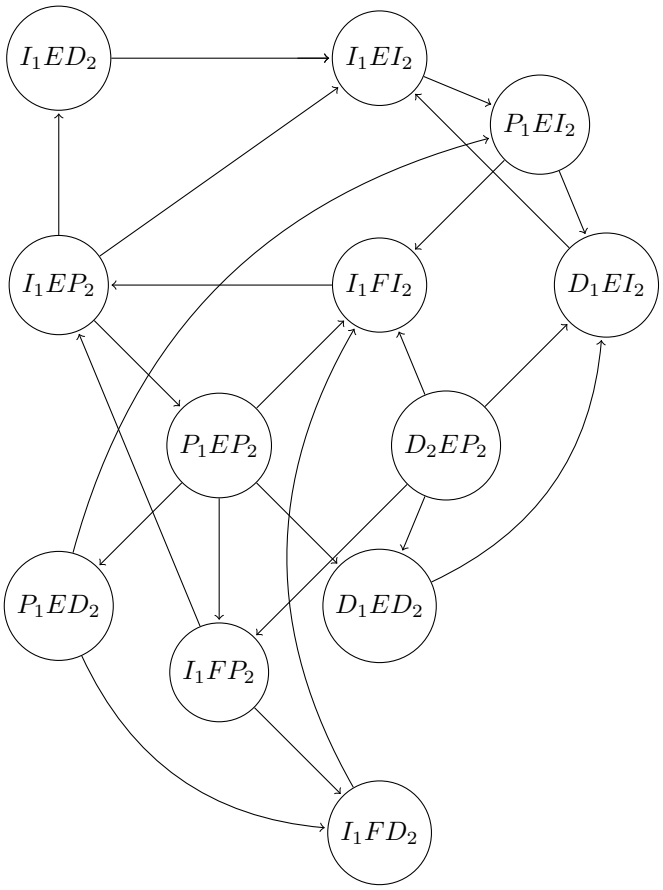
2

The state chart for the entire system can be seen below. It consists of all 18 possible states. Edges are not plotted due to space limitations.



3

When applying the new rules some states becomes unreachable, and thus the model becomes simpler. All PF* states are no longer allowed and therefore DF* states can never be reached. Also some transitions have been removes to comply with the new rule set (DED− > IED for example), further simplifying the system. The new DES can be seen below:



4

If controlling M_1 and M_2 separately both controllers needs to be aware of the entire system state to be able to reach the allowed states and only the allowed states. If M_1 is down the controller need to know that M_2 is not down, therefore the two controllers must communicate with each other.