Real-Time and Embedded Systems

Problem 3

Adrián Gómez Llorente
0904327G

Introduction

In this document third problem questions of Real Time and Embedded Systems are answered and explained.

Question 1

To check if a system is schedulable using Rate Monotonic algorithm we have to check next theorem:

> *A system of n independent preemptable periodic tasks with* $D_i = p_i$ *can be feasibly scheduled on one processor using RM if*

$$U \leq n*(2^{1/n}-1)$$

But this condition is sufficient but not necessary.

So we are going to calculate the total utilization of the system:

$$U = u_1 + u_2 + u_3 = \frac{1}{4} + \frac{1}{5} + \frac{2}{10} = 0.65$$

In this system we have 3 tasks so we are going to calculate U_{RM} with the parameter n=3.

$$U_{RM}(3) = 3(2^{\frac{1}{3}} - 1) = 0.779$$

Using previous theorem we get that a **feasible monotonic schedule is guaranteed** as can be seen below:

$$U \leq U_{RM} \Rightarrow 0.65 \leq 0.779$$

Next theorem demonstrate when a system is schedulable using EDF:

> *A system of independent preemptable periodic tasks with* $D_i = p_i$ *can be feasibily scheduled on one processor using EDF if and only if* $U \leq 1$

We have calculated U before and it was:

$$U = u_1 + u_2 + u_3 = \frac{1}{4} + \frac{1}{5} + \frac{2}{10} = 0.65$$

So we have to check if the theorem is true

$$U \leq 1 \Rightarrow 0.65 \leq 1$$

So **previous system can be scheduled using EDF**

Question 2

To check if a system is schedulable using Rate Monotonic algorithm we have to check next theorem:

A system of n independent preemptable periodic tasks with $D_i = p_i$ can be feasibly scheduled on one processor using RM if

$$U \leq n*(2^{1/n}-1)$$

We calculate the total utilization as before:

$$U = u_1 + u_2 + u_3 + u_4 = \frac{1}{4} + \frac{1}{5} + \frac{2}{10} + \frac{1}{6} = 0.81$$

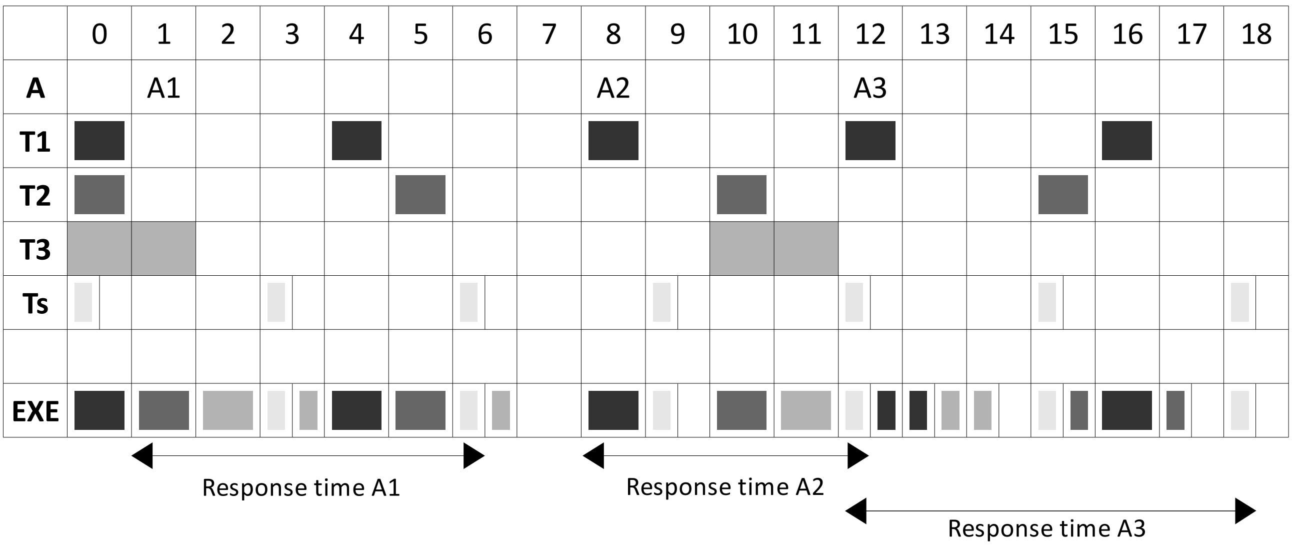
And we have to calculate U_{RM} with the parameter n=4 because there is 4 tasks:

$$U_{RM}(4) = 4(2^{\frac{1}{4}}-1) = 0.75$$

As the previous theorem says

$$U \leq U_{RM} \rightarrow 0.81 \leq 0.75$$

So in this case we can be sure if tasks can be scheduled using Rate Monotonic so we have to make draw an execution plan to check it. Next the execution plan is drawn:



In the previous plan we can see that **tasks are schedulable using Rate Monotonic** and response times for aperiodic jobs are:

- **A1:** 5.5
- **A2:** 4.5
- **A3:** 6.5

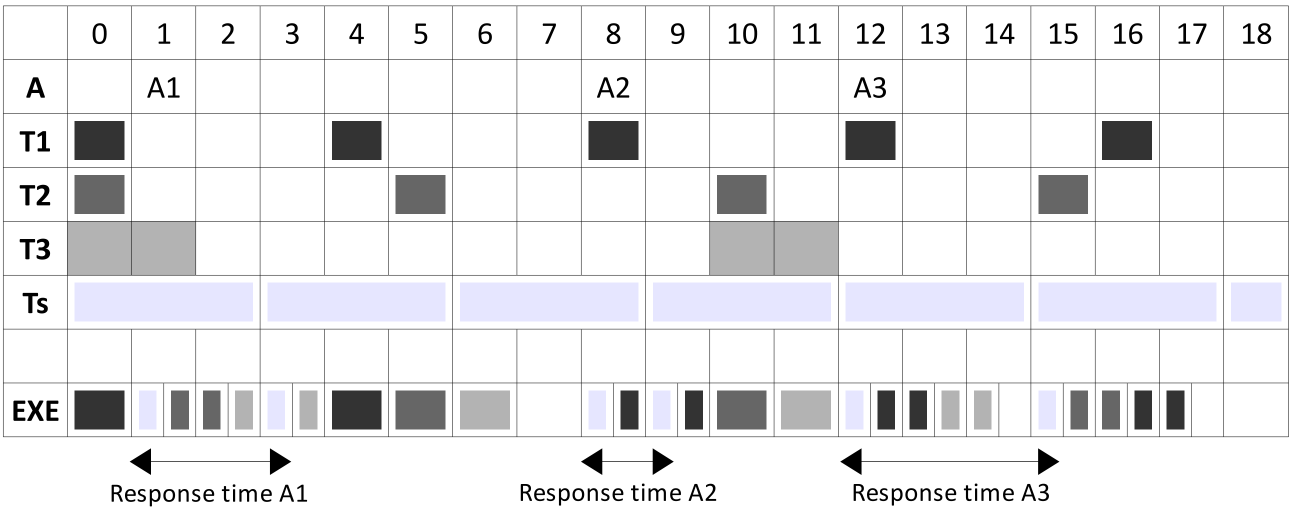
Question 3

Deferrable server improves times of aperiodic jobs, compared to polling servers because:

- The budget is consumed at the rate of one per unit time whenever the server executes
- Unused budget is retained throughout the period, to be used whenever there are aperiodic jobs to execute

So previous execution will be like shown next:

- Ts is executed with a duration of 0.5 each 3 cycles.



As we can see in the previous execution plan the systems **can be scheduled with a deferrable server** moreover time responses are:

- **A1:** 2.5
- **A2:** 1.5
- **A3:** 3.5

So **response times have been reduced** and the total execution time is one cycle less.

Question 4

To check if a system is schedulable using an sporadic server we can use next theorem:

An sporadic server (p_s, e_s) can be treated in a fixed-priority system exactly the same as any other task T_i with $p_i = p_s$ and $e_i = e_s$

So we are going to do the same as done in question 2 with a monotonic algorithm but with 4 next tasks:

- $T_1 = (4,1)$
- $T_2 = (5,1)$
- $T_3 = (10,2)$
- $T_4 = (3,0.5)$ This one is the sporadic server

There is an algorithm to check if a system is possible to be scheduled using monotonic algorithm:

A system of n independent preemptable periodic tasks with $D_i = p_i$ can be feasibly scheduled on one processor using RM if

$$U \leq n*(2^{1/n} - 1)$$

Operations needed to check this theorem are solved next:

$$U = u_1 + u_2 + u_3 + u_4 = \frac{1}{4} + \frac{1}{5} + \frac{2}{10} + \frac{1}{6} = 0.81$$

$$U_{RM}(4) = 4(2^{\frac{1}{4}} - 1) = 0.75$$

With results obtained we have to check the theorem

$$U \leq U_{RM} \rightarrow 0.81 \leq 0.75$$

With this result **we can't assume that this system can be scheduled using sporadic server** so an execution planned will be needed but this require keeping track of a lot of data and several cases to consider is impossible to do it.

Question 5

Differences between the consumption and replenishment rules for a simple sporadic server in a deadline driven system compared to rate monotonic system are shown next:

Consuption

Deadline driven system	Rate monotonic system
Consumption rule at time t: when either • Server is executing • Server has executed since being replenished and there have been no busy intervals since t	Consumption rule at time t: when either • Server is executing • *d* defined, server idle, no job with deadline before *d* ready

Replenishment

Deadline driven system	Rate monotonic system

Replenishment rule at time *t*
• Replenished with e_s
• First time executing after replenishment $(t=t_f)$;
  ○ If busy interval ends at t_f, $t_e = max(t_r, busy\,interval\,start)$
  ○ If no busy interval, then $t_e = t$, $next\ replenishment\ time = t_e + p_s$
• Next replenishment time except when:
  ○ $t_e + p_s < t_f$, then replenish when exhausted
  ○ if system idle before next replenishment, $t_e + p_s$, and then busy at t_b, budget replenished at $min(t_e + p_s, t_b)$
	Replenishment rule at time *t*
	• Initially and $replenishment\ budget = e_s$, $t_r = current$. Initially t_e and *d* undefined.
	• If t_e defined, $d = t_e + p_s$ and $next = t_e + p_s$. Otherwise (t_e and *d* undefined) t_e is defined as
	  ○ Aperiodic arrives at *t (queue empty)*, and busy interval since t_r then $t_e = t_r$. otherwise (lower priority job has executed in interval) $t_e = t$.
	  ○ At t_r if backlogged, $t_e = t_r$, or if idle then t_e and *d* are undefined.
	• Replenishment at next time except
	  ○ If next $t_e + p_s < t_f$, replenish when exhausted
	  ○ End of each idle interval of the periodic task system