Real-Time and Embedded Systems

Problem 1

Adrián Gómez Llorente

0904327G

Introduction

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

Question 1

A **real time system** is a system that must deliver services quickly and in a predictable period of time, examples of this kind of systems are vehicle's brakes or an aircraft error checking

The **difference between hard and soft real-time systems** are based in job's deadline. In a hard real-time system a job must never miss its deadline but in a soft real-time system some deadlines can be missed occasionally with an acceptable low probability.

This difference is more notable if we use an example. For instance an anti-lock brakes is a hard real-time system because jobs must be completed in an exact period of time because a little delay can produce an accident. Otherwise a DVD Player is a soft real-time system because it exists the possibility of a job to be delayed but this don't produce important problems.

Question 2

Hyper-period is the pattern of job release/execution time start to repeat. To get it we only have to calculate the LCM of the periods of each task.

In the example given we have next tasks:

- **T_1:** $p_1=5$, $e_1=1$
- **T_2:** $p_2=10$, $e_2=3$
- **T_3:** $p_3=3$, $e_3=1$

So the hyper-period is:

$$H = lcm(p_i) = lcm(p_1, p_2, p_3) = 30$$

Before explaining what the total utilizations of a system is, we have to speak about the ratio. The ratio is the utilization of a task and is expressed as:

$$u_i = \frac{e_i}{p_i}$$

The total utilization is the sum of the utilizations of all tasks in a system, so in the system given is:

$$U = \sum u_i = u_1 + u_2 + u_3 = \frac{e_1}{p_1} + \frac{e_2}{p_2} + \frac{e_3}{p_3} = \frac{1}{5} + \frac{3}{10} + \frac{1}{3} = \frac{5}{6}$$

Question 3

A **locally optimal** decision gives an optimal solution within a neighbouring set of solutions, however a **globally optimal** decision gives an optimal solution among all possible solutions.

A priority-driven scheduling gives algorithm place jobs in one or more queues and the ready job with the highest priority is executed so priorities are assigned to jobs and this priorities are calculated using locally optimal decisions.

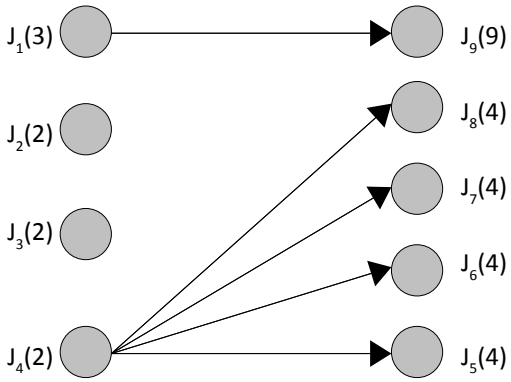
This means that all algorithms only compare one job to other existing jobs to set a priority getting a locally optimal solution, but it's not compared with all possible solutions so many times we don't get a globally optimal solution.

Question 4

Scheduling anomalies are based in the next theorem

> *If a task set is optimally scheduled on a multiprocessor with some priority assignment, a fixed number of processors, fixed execution times and precedence constraints, then increasing the number of processors, reducing execution times, or weakening the precedence constraints can increase the schedule length.*

We are going to show some situations where some scheduling anomalies are found using the next precedence graph of a task set:



Next we are going to show an optimal schedule of task on a three-processor machine

P1	J1			J9											
P2	J2		J4		J5			J7							
P3	J3				J6			J8							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

As can be seen we need 12 periods to complete all the tasks. Now we are going to use a four-processor system to complete the tasks:

P1	J1			J8											
P2	J2		J5				J9								
P3	J3		J6												
P4	J4		J7												
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

In the previous example the number of processors has been increased in one unit and we are getting 15 periods to complete all the tasks.