1 Ejercicio 5:

1.1 a:

Frente al avance que se venia produciendo en esos años del uso de las computadoras en distintos procesos industriales. Los autores establecieron que esto se iría masificiando aún más. Y que una correcta aplicación solo es posible cuando se tiene un scheduler eficiente y factible. Sobre todo para aquellos usos en los cuales se requerían que las tareas llevadas al cabo se ejecuten dentro de un lapso de tiempo determinado (a lo cual lo llaman *"hard-real-time"*). Ya que en caso contrario, una respuesta tardía podría provocar un efecto no deseado en una aplicación o no sería de interés. Además intentaron desarollar este tema. Ya que como describen, la mayoría de paper, hasta entonces, estaban orientados a contextos en lo que no había una cota de tiempo para llevar a cabo todas las tareas. Y los que habían, hacían uso de varios supuestos, llevando en algunos casos a situaciones irreales. Por lo que plantearon dos modelos de scheduler para tareas en un entorno de *hard-real-time*. Ambos se basan en fijar prioridades para cada una de las tareas pero, uno de manera estática y el otro dinámicamente. Además para mejorar el rendimiento del mismo buscan establecer cotas (cuando es posible) para determinar si es aplicable alguno de los modelos. De manera que no se produzca *overflow* (alguna de las tareas finalice su ejecución después del tiempo límite).

1.2 b:

En la sección 7 se introduce un nuevo algoritmo, el cual es para un scheduler basado en asignaciones de prioridades dinámicamente (llamado *The deadline Driven Scheduling*). Lo introducen como una variante al algoritmo presentado anteriormente (para un scheduler con prioridades fijas). Intentando evitar o disminuir algunos de los problemas presentados para dicho modelo.
Una de los problemas a resolver es mejorar las asignaciones de las prioridades. De manera de aprovechar el procesador a un 100%. De esta forma, este nuevo algoritmo establece que el nuevo modelo nunca va a ejecutar la idle sino, que siempre se va a estar ejecutando alguna tarea. A diferencia del modelo de prioridades fijas, donde la utilización del procesador podía variar de un 70 a un 100%. Para conseguir esto, se va a aumentar o disminuir la prioridad de una tarea conforme se acerca a su deadline, es decir las prioridades de una misma tarea van a ser modificadas a lo largo de su ejecución. Además, buscaron determinar si un conjunto de tareas evitara producir overflow, con este algoritmo. En el caso del scheduler con prioridades fijas, se estableció una condición para tal motivo (*teorema 4* [1]). Pero, la misma solo es necesaria. Para el scheduler presentado en esta sección se desarrollo una nueva condición, *teorema 7*, la cual garantiza que si se cumple entonces es aplicable el algoritmo y en caso contrario, no lo es (el mismo sera explicado en el siguiente punto).

1.3 c:

Teorema 7: *For a given set of m tasks, the deadline driven scheduling algorithm is feasible if and only if*

$$(C_1/T_1) + (C_2/T_2) + \ldots + (C_m/T_m) \leq 1$$ [2]

Siendo C_i el tiempo de ejecución para una tarea i, $1 \leq i \leq m$, y T_i su respectivo período.

En las secciones anteriores, al presente teorema, se determina que la fracción de CPU que utiliza una tarea en el sistema esta dado por C_i/T_i (*the utilization factor*). Lo que se plantea en este teorema es que un scheduler, para una cantidad m de tareas, es factible y por lo tanto realizable sin que se produzca overflow. Cuando la suma del costo de CPU de cada tarea es menor que uno o igual a 1. Es decir que el ejecutar todas las tareas este dentro de la capacidades del CPU (no supere el 100% del rendimiento del mismo). Ya que en caso contrario se estaria exigiendo que el CPU trabaje a más límite, lo cual es imposible.

1

2

[1] Scheduling Algorithms for Multiprogramming in a HardReal-Time Environment. C. L. LIU Project MAC, Massachusetts Institute of Technology AND JAMES W. LAYLAND Jet Propulsion Laboratory, California Institute of Technology. pág 6

[2] Scheduling Algorithms for Multiprogramming in a HardReal-Time Environment. C. L. LIU Project MAC, Massachusetts Institute of Technology AND JAMES W. LAYLAND Jet Propulsion Laboratory, California Institute of Technology. pág 10