

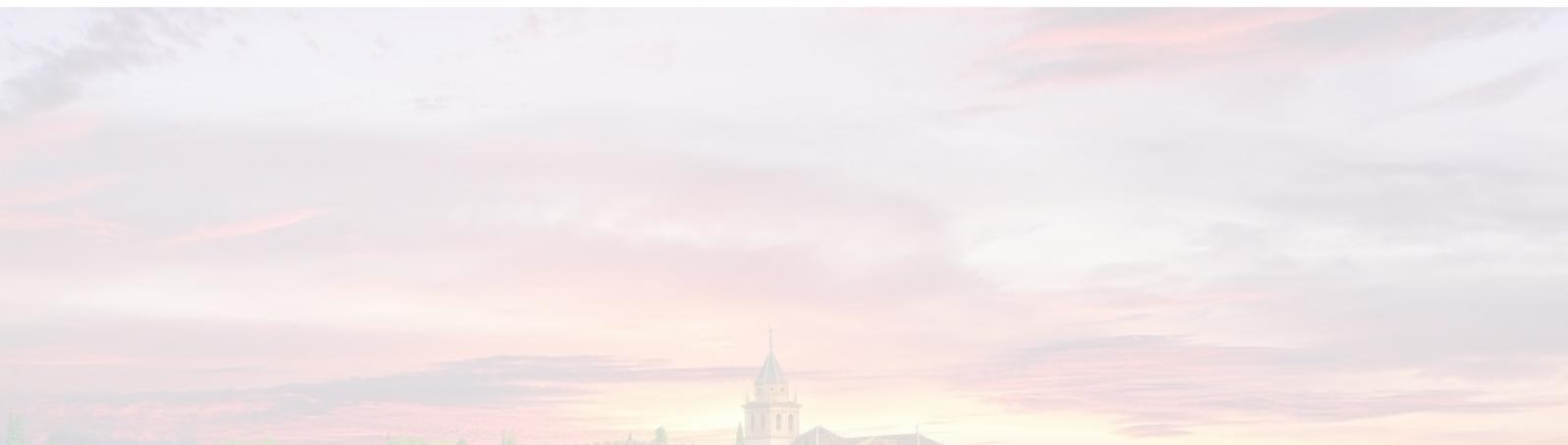


Ingeniería Informática + ADE

Universidad de Granada (UGR)

Autor: Ismael Sallami Moreno

Asignatura: Ejercicios Tema 4 SCD: Sistemas de Tiempo Real



Índice

1. Ejercicio 91	6
1.1. Enunciado	6
1.2. Solución	6
1.2.1. Cronograma	7
1.2.2. Implementación en pseudo-código	7
2. Ejercicio 92	9
2.1. Enunciado	9
2.2. Solución	9
2.2.1. Paso 1: Verificación inicial	9
2.2.2. Paso 2: Cálculo del marco secundario	10
2.2.3. Paso 3: Cronograma	10
3. Ejercicio 93	10
3.1. Enunciado	10
3.2. Solución	11
3.2.1. Implementación en pseudo-código	11
4. Ejercicio 94	12
4.1. Enunciado	12
4.2. Solución	12
4.2.1. Cálculo del factor de utilización	12
4.2.2. Cálculo del hiperperiodo	12
4.2.3. Cronograma	12
5. Ejercicio 95	13
5.1. Enunciado	13
5.2. Solución	13
5.2.1. Cálculo del Factor de Utilización	13
5.2.2. Cálculo del Hiperperiodo	13
5.2.3. Cronograma	14
5.2.4. Implementación en Pseudo-Código	14
6. Ejercicio 96	14
6.1. Enunciado	14
6.2. Solución	14
6.2.1. Planificación con RMS	14
6.2.2. Planificación con EDF	15
6.2.3. Explicación del Algoritmo EDF	16
7. Ejercicio 97	16
7.1. Enunciado	16
7.2. Solución	17
7.2.1. Planificación mediante un Ejecutivo Cíclico	17

7.2.2.	Planificación mediante Prioridades Estáticas (RMS)	17
7.2.3.	Planificación mediante Prioridades Dinámicas (EDF)	17
7.2.4.	Conclusión	18
8.	Ejercicio 98	18
8.1.	Enunciado	18
8.2.	Solución	18
8.2.1.	Paso 1: Cálculo del hiperperiodo y prioridades	18
8.2.2.	Paso 2: Construcción del cronograma	19
8.2.3.	Paso 3: Cálculo de los tiempos de respuesta	19
8.2.4.	Paso 4: Interferencias entre tareas	19
8.2.5.	Conclusión	19
9.	Ejercicio 99	20
9.1.	Enunciado	20
9.2.	Solución	20
9.2.1.	Paso 1: Cálculo del factor de utilización	20
9.2.2.	Paso 2: Construcción del cronograma	20
9.2.3.	Paso 3: Verificación de los plazos	21
9.2.4.	Conclusión	21
10.	Ejercicio 100	21
10.1.	Enunciado	21
10.2.	Solución	22
10.2.1.	Paso 1: Test del factor de utilización	22
10.2.2.	Paso 2: Hiperperiodo del sistema	22
10.2.3.	Paso 3: Construcción del cronograma	22
10.2.4.	Paso 4: Verificación de los plazos	23
10.2.5.	Conclusión	23
11.	Ejercicio 101: Verificación de planificabilidad con Deadline Monotonic (DM)	23
11.1.	Enunciado	23
11.2.	Solución	23
11.2.1.	Paso 1: Conjunto de tareas	23
11.2.2.	Paso 2: Cálculo del tiempo de respuesta (R_i)	24
11.2.3.	Paso 3: Conclusión	24
12.	Ejercicio 102: Análisis de afirmaciones sobre inversión de prioridad	25
12.1.	Enunciado	25
12.2.	Solución	25
12.2.1.	Análisis de cada afirmación	25
12.2.2.	Conclusión	26
13.	Ejercicio 103: Utilización máxima del procesador para un Servidor Esporádico	26
13.1.	Enunciado	26
13.2.	Solución	26
13.2.1.	Paso 1: Utilización del procesador por las tareas periódicas	26

13.2.2. Paso 2: Límite de utilización máxima para RM	27
13.2.3. Paso 3: Utilización restante para el Servidor Esporádico	27
13.3. Conclusión	27
14. Ejercicio 104: Utilización máxima del procesador para un Servidor Diferido (SD)	27
14.1. Enunciado	27
14.2. Solución	27
14.2.1. Paso 1: Fórmula para el límite de utilización máxima	27
14.2.2. Paso 2: Sustitución de valores	28
14.2.3. Paso 3: Resolución numérica	28
14.3. Conclusión	28
15. Ejercicio 105: Planificación de tareas aperiódicas con Servidor Diferido (SD)	28
15.1. Enunciado	28
15.2. Solución	29
15.2.1. Paso 1: Utilización máxima del SD	29
15.2.2. Paso 2: Planificación de tareas periódicas y SD	29
15.2.3. Paso 3: Planificación de tareas aperiódicas con el SD	29
15.2.4. Paso 4: Cronograma	29
15.3. Conclusión	30
16. Ejercicio 106: Planificación con Servidor Esporádico (SS)	30
16.1. Enunciado	30
16.2. Solución	30
16.2.1. Paso 1: Utilización máxima del Servidor Esporádico (SS)	30
16.2.2. Paso 2: Funcionamiento del Servidor Esporádico	31
16.2.3. Paso 3: Planificación de las tareas aperiódicas	31
16.2.4. Paso 4: Cronograma de ejecución	31
16.3. Conclusión	31
17. Ejercicio 107: Planificación con Servidor Diferido (DS)	32
17.1. Enunciado	32
17.2. Solución	32
17.2.1. Paso 1: Utilización máxima del Servidor Diferido (DS)	32
17.2.2. Paso 2: Funcionamiento del Servidor Diferido	33
17.2.3. Paso 3: Planificación de las tareas aperiódicas	33
17.2.4. Paso 4: Cronograma de ejecución	33
17.3. Conclusión	33
18. Ejercicio 108: Planificación con Servidor Esporádico (SS)	33
18.1. Enunciado	33
18.2. Solución	34
18.2.1. Paso 1: Verificación de la planificabilidad de las tareas periódicas	34
18.2.2. Paso 2: Funcionamiento del Servidor Esporádico (SS)	34
18.2.3. Paso 3: Planificación de las tareas aperiódicas	34
18.2.4. Paso 4: Cronograma de ejecución	35

18.3. Conclusión 35



Nota: En la resolución de estos ejercicios se sigue una metodología distinta a la teoría para algunos casos.

1 Ejercicio 91

1.1. Enunciado

Dado el conjunto de tareas periódicas y sus atributos temporales que se indica en la tabla de aquí abajo, determinar si se puede planificar el conjunto de dichas tareas utilizando un esquema de planificación basado en planificación cíclica. Diseña el plan cíclico determinando el marco secundario, y el entrelazamiento de las tareas sobre un cronograma.

Tarea	Ci	Ti	Di
T1	10	40	40
T2	18	50	50
T3	10	200	200
T4	20	200	200

Cuadro 1: Conjunto de tareas periódicas y sus atributos temporales

1.2. Solución

Nota: El código proporcionado en el enunciado se accede pinchando aquí.

Para resolver este ejercicio aparte de poseer los conceptos teóricos necesarios de las diapositivas, vamos a servirnos de la metodología de resolución aprendida en prácticas. Este ejercicio al ser el primero será más *didáctico* que el resto.

En base a esta tabla:

Tarea	Ci	Ti	Di
T1	10	40	40
T2	18	50	50
T3	10	200	200
T4	20	200	200

Cuadro 2: Conjunto de tareas periódicas y sus atributos temporales

Podemos afirmar que

$$D_i = T_i$$

(Para hacerlo mas similar al cpp¹), por lo que podemos eliminar esa columna y quedarnos con:

¹Proporcionado en el enlace del inicio de esta sección.

Tarea	Ci	Ti
T1	10	40
T2	18	50
T3	10	200
T4	20	200

Cuadro 3: Conjunto de tareas periódicas y sus atributos temporales

Tenemos que el ciclo principal dura $T_m = mcm(T_i) \forall i0, \dots, 4 = 200$, de manera que nuestro ciclo secundario debe de cumplir:

1. $\max(C_i) \leq T_s \leq \min(D_i)$
2. T_s divide a T_m , siendo el resultado $\in \mathbb{Z}$
3. T_s será un divisor del período del ciclo principal, que en este caso es 200, es decir, debe de existir un entero K , tal que $T_m = k \times T_s$

En base a esto vamos a optar por escoger que el ciclo secundario debe de durar $T_s = 20ms$.

Entonces, las veces que se debe de ejecutar cada tarea es:

- Tarea 1 $\rightarrow \frac{200}{40} = 5$ veces con un tiempo de cómputo de 10. El tiempo de cómputo máximo calculado como la sumatoria de todos los que se ejecutan en ese instante no debe de sobrepasar 20.
- Tarea 2 $\rightarrow \frac{200}{50} = 4$ veces con un tiempo de cómputo de 18.
- Tarea 3 $\rightarrow \frac{200}{25} = 8$ veces con un tiempo de cómputo de 10.
- Tarea 4 $\rightarrow \frac{200}{100} = 2$ veces con un tiempo de cómputo de 20.

Podemos afirmar que si es planificable y el cronograma quedaría de la siguiente manera:

1.2.1. Cronograma

Tiempo (ms)	0-20	20-40	40-60	60-80	80-100	100-120	120-140	140-160	160-180	180-200
T1	-	X	-	X	-	1	-	1	-	X
T2	X	-	X	-	X	-	X	-	-	-
T3	-	X	-	-	-	-	-	-	-	-
T4	-	-	-	-	-	-	-	-	X	-
Cómputo (ms)	18	20	18	10	18	10	18	10	20	10

Cuadro 4: Cronograma de ejecución de las tareas

1.2.2. Implementación en pseudo-código

```
1
2 Ts := duración del ciclo secundario;
3 ini_sec := instante de inicio del ciclo secundario;
4
5 while (true) { // ciclo principal
6     for (desde i := 1 hasta 10) { // 10 porque son las fases del ciclo
7         // secundario en base a nuestro cronograma
8
9         inicio_iteracion := instante actual;
10
11        switch(i) {
12            case 1:
13                Tarea1();
14                Tarea2();
15                break;
16
17            case 2:
18                Tarea1();
19                break;
20
21            case 3:
22                Tarea3();
23                break;
24
25            case 4:
26                Tarea1();
27                Tarea2();
28                break;
29
30            case 5:
31                Tarea1();
32                break;
33
34            case 6:
35                Tarea2();
36                break;
37
38            case 7:
39                Tarea1();
40                Tarea3();
41                break;
42
43            case 8:
44                Tarea1();
45                Tarea2();
46                break;
47
48            case 9:
49                Tarea1();
50                break;
51
52            case 10:
53                Tarea4();
54                break;
55        }
```

2 EJERCICIO 92

```
55     ini_sec += Ts; // siguiente instante
56     sleep_until(ini_sec); // esperar hasta el siguiente ciclo
57         secundario
58     fin_iteracion := instante actual; // medir retraso
59     retraso := fin_iteracion - ini_sec; // calcular retraso
60 }
61 }
```

Listing 1: Implementación en C++ del planificador cíclico

2 Ejercicio 92

2.1. Enunciado

El siguiente conjunto de tareas periódicas se puede planificar con ejecutivos cíclicos. Determina si esto es cierto calculando el marco secundario que debería tener. Dibuja el cronograma que muestre las ocurrencias de cada tarea y su entrelazamiento. ¿Cómo se tendría que implementar? (escribe el pseudo-código de la implementación).

Tarea	Ci	Ti	Di
T1	2	6	6
T2	2	8	8
T3	3	12	12

Cuadro 5: Conjunto de tareas periódicas y sus atributos temporales

2.2. Solución

Nota: Este ejercicio se resolverá siguiendo las metodologías vistas en clase y prácticas. Además, será similar a la resolución anterior del ejercicio 91.

2.2.1. Paso 1: Verificación inicial

Podemos observar que $D_i = T_i$ para todas las tareas, lo que simplifica la verificación inicial. Esto significa que podemos eliminar la columna D_i y trabajar únicamente con C_i y T_i :

Tarea	Ci	Ti
T1	2	6
T2	2	8
T3	3	12

Cuadro 6: Conjunto simplificado de tareas periódicas

2.2.2. Paso 2: Cálculo del marco secundario

El ciclo principal es el mínimo común múltiplo de los períodos T_i :

$$T_m = \text{mcm}(6, 8, 12) = 24 \text{ ms}$$

El marco secundario T_s debe cumplir:

1. $\max(C_i) \leq T_s \leq \min(D_i)$
2. T_s divide a T_m de manera exacta ($T_m/T_s \in \mathbb{Z}$).

Restricción 1: $\max(C_i) = 3 \leq T_s \leq \min(D_i) = 6$.

Restricción 2: T_s debe dividir a $T_m = 24$. Los divisores de 24 dentro del rango permitido son:

$$T_s = 3, 4, 6$$

Escogemos $T_s = 6$ ya que cumple con ambas restricciones.

2.2.3. Paso 3: Cronograma

Dado $T_s = 6$, construimos el cronograma:

Tiempo (ms)	0-6	6-12	12-18	18-24
T1	X	X	X	X
T2	-	X	-	X
T3	X	-	X	-

Cuadro 7: Cronograma de ejecución de las tareas

Podemos afirmar que no es planificable, debido a que no se puede ejecutar correctamente, ya que la tarea 2 debe de ejecutarse 3 veces, pero no puede.

3 Ejercicio 93

3.1. Enunciado

Comprobar si el conjunto de procesos periódicos que se muestra en la siguiente tabla es planificable con el algoritmo RMS utilizando el test basado en el factor de utilización del tiempo del procesador. Si el test no se cumple, ¿debemos descartar que el sistema sea planificable?

Tarea	Ci	Ti
T1	9	30
T2	10	40
T3	10	50

Cuadro 8: Conjunto de procesos periódicos

3.2. Solución

El algoritmo **Rate Monotonic Scheduling (RMS)** es un esquema de planificación de tiempo real que asigna prioridades de manera estática a tareas periódicas. Las prioridades están inversamente relacionadas con los periodos (T_i): menor periodo, mayor prioridad. Este método utiliza el factor de utilización del procesador (U) para determinar si un conjunto de tareas es planificable.

El test de planificabilidad se basa en:

$$U = \sum_{i=1}^N \frac{C_i}{T_i}, \quad U \leq U_0(N) = N \cdot (2^{\frac{1}{N}} - 1)$$

Para $N = 3$:

$$U_0(3) = 3 \cdot (2^{\frac{1}{3}} - 1) \approx 0,779$$

Cálculo del factor de utilización:

$$U = \frac{9}{30} + \frac{10}{40} + \frac{10}{50} = 0,3 + 0,25 + 0,2 = 0,75$$

Conclusión: El sistema es planificable porque $U = 0,75 \leq U_0(3) = 0,779$.

Si $U > U_0(N)$, no podemos descartar automáticamente la planificabilidad. Deberíamos construir un cronograma utilizando un diagrama de Gantt para verificar si todas las tareas cumplen sus plazos, ya que como se afirma en la teoría no se puede afirmar nada, es solo condición suficiente.

3.2.1. Implementación en pseudo-código

Nota: podemos implementar el algoritmo RMS en C++ de la siguiente manera usando los módulos ya que sería seguir una lógica similar a la de teoría y prácticas de la Asignatura.

```

1 // Prioridades: T1 > T2 > T3 (porque T1 tiene el menor período)
2 while (true) {
3     tiempo_actual := instante_actual();
4
5     if (tiempo_actual % 30 == 0) {
6         ejecutar(T1);
7     }
8     if (tiempo_actual % 40 == 0) {
9         ejecutar(T2);
10    }
11    if (tiempo_actual % 50 == 0) {
12        ejecutar(T3);
13    }
14
15    tiempo_siguiete := tiempo_actual + intervalo_minimo();
16    sleep_until(tiempo_siguiete);
17 }

```

Listing 2: Implementación en C++ del algoritmo RMS

Este código asegura que las tareas se ejecutan según sus prioridades y periodos definidos. Si necesitas el diagrama de Gantt o más detalles, avísame.

4 Ejercicio 94

4.1. Enunciado

Considérese el siguiente conjunto de tareas compuesto por tres tareas periódicas:

Tarea	C _i	T _i
T1	10	40
T2	20	60
T3	20	80

Cuadro 9: Conjunto de tareas periódicas

Comprobar la planificabilidad del conjunto de tareas con el algoritmo RMS utilizando el test basado en el factor de utilización. Calcular el hiperperiodo y construir el correspondiente cronograma.

4.2. Solución

4.2.1. Cálculo del factor de utilización

El algoritmo **Rate Monotonic Scheduling (RMS)** establece que las tareas son planificables si su factor de utilización (U) cumple:

$$U = \sum_{i=1}^N \frac{C_i}{T_i} \quad \text{y} \quad U \leq U_0(N) = N \cdot (2^{\frac{1}{N}} - 1)$$

Para $N = 3$:

$$U_0(3) = 3 \cdot (2^{\frac{1}{3}} - 1) \approx 0,779$$

Calculamos U :

$$U = \frac{10}{40} + \frac{20}{60} + \frac{20}{80} = 0,25 + 0,333 + 0,25 = 0,833$$

Conclusión: $U = 0,833 > 0,779$, por lo que el conjunto de tareas no pasa el test basado en el factor de utilización. Sin embargo, esto no significa que el sistema no sea planificable. Es necesario construir el cronograma para verificarlo.

4.2.2. Cálculo del hiperperiodo

El hiperperiodo (T_m) es el mínimo común múltiplo (mcm) de los períodos de las tareas:

$$T_m = \text{mcm}(40, 60, 80) = 240 \text{ ms}$$

4.2.3. Cronograma

Si intentamos hacer el cronograma vemos que no es posible debido a que en una de las fases no podemos ejecutar las 3 tareas debido a que sobrepasamos el tiempo de cómputo total y por ende, no podemos planificar este cronograma aunque cumpla el test RMS.

5 Ejercicio 95

5.1. Enunciado

Comprobar la planificabilidad y construir el cronograma de acuerdo al algoritmo de planificación RMS del siguiente conjunto de tareas periódicas:

Tarea	C _i	T _i
T1	20	60
T2	20	80
T3	20	120

Cuadro 10: Conjunto de tareas periódicas

5.2. Solución

El algoritmo **Rate Monotonic Scheduling (RMS)** asigna prioridades de manera estática en función del periodo (T_i): a menor periodo, mayor prioridad. Verificamos la planificabilidad utilizando el test basado en el factor de utilización del procesador.

5.2.1. Cálculo del Factor de Utilización

El factor de utilización se calcula como:

$$U = \sum_{i=1}^N \frac{C_i}{T_i}$$

Sustituyendo los valores de las tareas:

$$U = \frac{20}{60} + \frac{20}{80} + \frac{20}{120} = 0,333 + 0,25 + 0,1667 = 0,7497$$

El límite de utilización máxima ($U_0(N)$) para $N = 3$ tareas es:

$$U_0(3) = 3 \cdot (2^{\frac{1}{3}} - 1) \approx 0,779$$

Conclusión: Como $U = 0,7497 \leq U_0(3) = 0,779$, el conjunto de tareas es planificable según RMS.

5.2.2. Cálculo del Hiperperiodo

El hiperperiodo (T_m) es el **mínimo común múltiplo (mcm)** de los periodos de las tareas:

$$T_m = \text{mcm}(60, 80, 120) = 240 \text{ ms}$$

5.2.3. Cronograma

El cronograma se construye considerando las instancias de activación de cada tarea en el intervalo $[0, T_m]$, donde cada tarea T_i se activa cada T_i ms y tiene un tiempo de ejecución de C_i ms.

Si se puede planificar, la manera de desarrollar el cronograma es similar a las realizadas anteriormente.

Nota: Si se requiere el diagrama de Gantt detallado, puede construirse para el hiperperiodo calculado.

5.2.4. Implementación en Pseudo-Código

En cuanto a la implementación pasa de igual manera.

Con este análisis, concluimos que el conjunto de tareas es planificable y el cronograma asegura el cumplimiento de los plazos.

6 Ejercicio 96

6.1. Enunciado

Determinar si el siguiente conjunto de tareas puede planificarse con la política de planificación RMS y con la política EDF, utilizando los tests de planificabilidad adecuados para cada uno de los dos casos. Comprobar también la planificabilidad construyendo los cronogramas para ambas políticas.

Tarea	C_i	T_i
T1	1	5
T2	1	10
T3	2	20
T4	10	20
T5	7	100

Cuadro 11: Conjunto de tareas periódicas

6.2. Solución

6.2.1. Planificación con RMS

El algoritmo **Rate Monotonic Scheduling (RMS)** asigna prioridades de manera estática en función del periodo (T_i): a menor periodo, mayor prioridad. Verificamos la planificabilidad utilizando el test basado en el factor de utilización del procesador.

Test de Planificabilidad RMS El factor de utilización del procesador se calcula como:

$$U = \sum_{i=1}^N \frac{C_i}{T_i}$$

Sustituyendo los valores de las tareas:

$$U = \frac{1}{5} + \frac{1}{10} + \frac{2}{20} + \frac{10}{20} + \frac{7}{100} = 0,2 + 0,1 + 0,1 + 0,5 + 0,07 = 0,97$$

El límite de utilización máxima para $N = 5$ tareas es:

$$U_0(5) = 5 \cdot (2^{\frac{1}{5}} - 1) \approx 0,743$$

Conclusión: Como $U = 0,97 > 0,743$, el sistema **no pasa el test RMS**. Esto indica que no podemos garantizar la planificabilidad mediante este test.

Construcción del Cronograma RMS Para comprobar la planificabilidad con RMS, construimos el cronograma hasta el hiperperiodo (T_m), que es el **mcm** de los periodos:

$$T_m = \text{mcm}(5, 10, 20, 100) = 100 \text{ ms}$$

Tiempo (ms)	0-1	1-2	2-3	3-4	4-5	...
T1	X	-	X	-	X	...
T2	-	-	-	-	X	...
T3	-	-	-	X	-	...
T4	-	X	X	X	X	...
T5	-	-	-	-	-	...

Cuadro 12: Cronograma RMS para $T_m = 100$ ms

Si las tareas cumplen sus plazos en T_m , el sistema es planificable con RMS.

6.2.2. Planificación con EDF

El algoritmo **Earliest Deadline First (EDF)** asigna prioridades dinámicamente según los plazos de respuesta (d_i): a menor plazo restante, mayor prioridad. Este método utiliza el segundo test de planificabilidad de Liu y Layland:

$$U = \sum_{i=1}^N \frac{C_i}{T_i} \quad \text{y} \quad U \leq 1$$

Test de Planificabilidad EDF Sustituyendo los valores de las tareas:

$$U = \frac{1}{5} + \frac{1}{10} + \frac{2}{20} + \frac{10}{20} + \frac{7}{100} = 0,97$$

Dado que $U = 0,97 \leq 1$, el sistema pasa el test EDF.

Construcción del Cronograma EDF Para comprobar la planificabilidad, construimos el cronograma dinámico, priorizando las tareas con menor plazo restante, hasta $T_m = 100$.

Tiempo (ms)	0-1	1-2	2-3	3-4	4-5	...
T1	X	-	X	-	X	...
T2	-	-	-	-	X	...
T4	-	X	X	X	X	...
T3	-	-	-	X	-	...
T5	-	-	-	-	-	...

Cuadro 13: Cronograma EDF para $T_m = 100$ ms

6.2.3. Explicación del Algoritmo EDF

El algoritmo EDF asigna prioridades dinámicas a las tareas en función del tiempo restante hasta su plazo de respuesta absoluto (d_i). Cuando una tarea se activa, se compara su plazo con el de las demás tareas en cola de ejecución, ejecutándose aquella que tenga el plazo más cercano a expirar. EDF es óptimo para tareas periódicas y aperiódicas si $U \leq 1$.

Ventajas del EDF:

- Mayor utilización del procesador, alcanzando hasta $U = 1$.
- Planificabilidad dinámica para sistemas con alta carga de tareas.

Nota: A pesar de pasar el test EDF, el cronograma final debe verificarse para confirmar que todas las tareas cumplen sus plazos.

7 Ejercicio 97

7.1. Enunciado

Determinar razonadamente si el siguiente conjunto de tareas puede planificarse en un sistema monoprocesador utilizando:

- Un **ejecutivo cíclico**.
- Algún algoritmo basado en **prioridades estáticas o dinámicas**.

Tarea	Ci	Ti
T1	1	5
T2	1	10
T3	2	10
T4	11	20
T5	5	100

Cuadro 14: Conjunto de tareas periódicas

7.2. Solución

7.2.1. Planificación mediante un Ejecutivo Cíclico

El **ejecutivo cíclico** requiere que las tareas se asignen en ciclos secundarios, garantizando que todas se ejecuten dentro del hiperperiodo. Para que sea planificable:

$$TM = \text{mcm}(5, 10, 20, 100) = 100 \text{ ms}$$

El tamaño del ciclo secundario (TS) debe cumplir:

$$\text{máx}(C_i) \leq TS \leq \text{mín}(T_i)$$

En este caso:

$$\text{máx}(C_i) = 11 \quad \text{y} \quad \text{mín}(T_i) = 5$$

Dado que $11 > 5$, no es posible definir un ciclo secundario que cumpla esta condición. Por lo tanto, el sistema **no puede planificarse** mediante un ejecutivo cíclico.

7.2.2. Planificación mediante Prioridades Estáticas (RMS)

El algoritmo **Rate Monotonic Scheduling (RMS)** asigna prioridades según los periodos (T_i): menor T_i , mayor prioridad. Calculamos el factor de utilización del procesador (U):

$$U = \sum_{i=1}^N \frac{C_i}{T_i}$$

Sustituyendo los valores:

$$U = \frac{1}{5} + \frac{1}{10} + \frac{2}{10} + \frac{11}{20} + \frac{5}{100} = 0,2 + 0,1 + 0,2 + 0,55 + 0,05 = 1,1$$

El límite de utilización máxima ($U_0(N)$) para $N = 5$ tareas es:

$$U_0(5) = 5 \cdot (2^{\frac{1}{5}} - 1) \approx 0,743$$

Dado que $U = 1,1 > 0,743$, el sistema **no pasa el test RMS**, y no podemos garantizar su planificabilidad con prioridades estáticas.

7.2.3. Planificación mediante Prioridades Dinámicas (EDF)

El algoritmo **Earliest Deadline First (EDF)** asigna prioridades según los plazos de respuesta (d_i). El test de planificabilidad EDF verifica si:

$$U = \sum_{i=1}^N \frac{C_i}{T_i} \leq 1$$

Sustituyendo los valores:

$$U = \frac{1}{5} + \frac{1}{10} + \frac{2}{10} + \frac{11}{20} + \frac{5}{100} = 1,1$$

Dado que $U = 1,1 > 1$, el sistema **no pasa el test EDF** y no es planificable con este algoritmo.

7.2.4. Conclusión

- **Ejecutivo Cíclico:** No es posible planificar las tareas, ya que no se puede definir un ciclo secundario que cumpla las restricciones de los periodos.
- **Prioridades Estáticas (RMS):** El sistema no pasa el test de utilización RMS, por lo que no es posible garantizar su planificabilidad.
- **Prioridades Dinámicas (EDF):** Aunque EDF es más eficiente, el sistema tampoco pasa el test, ya que el factor de utilización excede 1.

En resumen, este conjunto de tareas **no puede planificarse** en un sistema monoprocesador utilizando ninguno de los métodos mencionados.

Ejercicios Adicionales

8 Ejercicio 98

8.1. Enunciado

Para el conjunto de tareas cuyos datos se muestran más abajo, se pide:

1. Dibujar el gráfico de ejecución y obtener el tiempo de respuesta de cada tarea.
2. Determinar, mediante inspección del gráfico anterior, cuántas veces interfiere la tarea T_1 a la tarea T_3 durante un intervalo temporal dado por el tiempo de respuesta de esta última tarea.
3. Hacer lo mismo que en (b), pero para las tareas T_1 y T_2 .

Tarea	Ci	Ti	Di
T1	1	3	2
T2	3	6	5
T3	2	13	13

Cuadro 15: Conjunto de tareas periódicas

8.2. Solución

8.2.1. Paso 1: Cálculo del hiperperiodo y prioridades

El hiperperiodo (T_m) es el mínimo común múltiplo de los períodos (T_i):

$$T_m = \text{mcm}(3, 6, 13) = 78 \text{ ms.}$$

Asignamos las prioridades utilizando el algoritmo **Rate Monotonic Scheduling (RMS)**, donde las tareas con menor período tienen mayor prioridad. Por lo tanto, el orden de prioridad es:

$$T_1 > T_2 > T_3.$$

8.2.2. Paso 2: Construcción del cronograma

El cronograma refleja los momentos en que las tareas comienzan y terminan en el intervalo $[0, T_m = 78]$. Consideramos los tiempos de cómputo (C_i) y los períodos (T_i) de las tareas:

Tiempo (ms)	0-1	1-3	3-4	4-7	7-9	9-10	10-11	12-13	13-15
T1	X	-	X	-	X	-	X	-	X
T2	-	X	-	X	X	X	-	-	X
T3	-	-	-	-	-	-	-	X	X

Cuadro 16: Cronograma de ejecución en el intervalo $[0, T_m = 78]$

Explicación del cronograma:

- T_1 : Se ejecuta cada 3 ms durante $C_1 = 1$ ms, comenzando en 0, 3, 6, etc.
- T_2 : Se ejecuta cada 6 ms durante $C_2 = 3$ ms, comenzando en 6, 12, etc.
- T_3 : Se ejecuta cada 13 ms durante $C_3 = 2$ ms, comenzando en 13, 26, etc.

8.2.3. Paso 3: Cálculo de los tiempos de respuesta

El tiempo de respuesta (R_i) de cada tarea es el intervalo desde su activación hasta la finalización de su ejecución, considerando interferencias de tareas con mayor prioridad.

1. T_1 : Tiene la mayor prioridad, por lo que no sufre interferencias. Su tiempo de respuesta es:

$$R_1 = C_1 = 1 \text{ ms.}$$

2. T_2 : Sufre interferencias de T_1 , que se ejecuta una vez durante el período de T_2 :

$$R_2 = C_2 + \text{interferencia de } T_1 = 3 + 1 = 4 \text{ ms.}$$

3. T_3 : Sufre interferencias de T_1 y T_2 . Durante el tiempo de respuesta de T_3 , T_1 se ejecuta 4 veces y T_2 2 veces:

$$R_3 = C_3 + 4 \cdot C_1 + 2 \cdot C_2 = 2 + 4 \cdot 1 + 2 \cdot 3 = 2 + 4 + 6 = 12 \text{ ms.}$$

8.2.4. Paso 4: Interferencias entre tareas

- T_1 y T_3 : Durante $R_3 = 12$ ms, T_1 se ejecuta 4 veces (en los intervalos 0-1, 3-4, 6-7, y 9-10).
- T_1 y T_2 : Durante $R_2 = 4$ ms, T_1 se ejecuta 1 vez (en el intervalo 0-1).

8.2.5. Conclusión

- Todas las tareas cumplen sus plazos límites (D_i).
- T_1 interfiere con T_3 4 veces durante R_3 .
- T_1 interfiere con T_2 1 vez durante R_2 .

9 Ejercicio 99

9.1. Enunciado

Verificar la planificabilidad del siguiente conjunto de tareas utilizando para ello el algoritmo del “Primero el del Tiempo Límite más Cercano” (EDF).

Tarea	C _i	T _i
T1	1	4
T2	2	6
T3	3	8

Cuadro 17: Conjunto de tareas periódicas

9.2. Solución

El algoritmo **EDF (Earliest Deadline First)** da prioridad a la tarea con el tiempo límite más cercano en cada instante de planificación. Para determinar la planificabilidad del sistema, usamos el test del factor de utilización y verificamos los plazos.

9.2.1. Paso 1: Cálculo del factor de utilización

El factor de utilización del procesador se define como:

$$U = \sum_{i=1}^N \frac{C_i}{T_i},$$

donde C_i es el tiempo de cómputo y T_i es el período de la tarea.

Para este conjunto de tareas:

$$U = \frac{1}{4} + \frac{2}{6} + \frac{3}{8} = 0,25 + 0,333 + 0,375 = 0,958.$$

Dado que $U \leq 1$, el sistema es potencialmente planificable.

9.2.2. Paso 2: Construcción del cronograma

Para verificar la planificabilidad, construimos el cronograma en el hiperperíodo (T_m) del sistema, que es el mínimo común múltiplo de los períodos:

$$T_m = \text{mcm}(4, 6, 8) = 24 \text{ ms.}$$

En cada instante, seleccionamos la tarea con el tiempo límite más cercano que esté lista para ejecutarse.

Tiempo (ms)	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8
T1	X	-	-	X	X	-	-	X
T2	-	X	X	-	-	X	X	-
T3	-	-	-	-	-	-	X	X

Cuadro 18: Cronograma de las tareas en el intervalo $[0, T_m = 24]$

Explicación del cronograma:

- T_1 : Tiene un período de 4 ms y un tiempo de cómputo de 1 ms. Se ejecuta en los intervalos 0 – 1, 4 – 5, 8 – 9, etc.
- T_2 : Tiene un período de 6 ms y un tiempo de cómputo de 2 ms. Se ejecuta en los intervalos 1 – 3, 6 – 8, 12 – 14, etc.
- T_3 : Tiene un período de 8 ms y un tiempo de cómputo de 3 ms. Se ejecuta en los intervalos 7 – 10, 15 – 18, etc.

9.2.3. Paso 3: Verificación de los plazos

El algoritmo EDF garantiza que las tareas cumplan sus plazos si $U \leq 1$. Además, al observar el cronograma:

- T_1 cumple su plazo en cada instancia.
- T_2 cumple su plazo en cada instancia.
- T_3 cumple su plazo en cada instancia.

9.2.4. Conclusión

El sistema es planificable² bajo el algoritmo EDF, ya que el factor de utilización es menor o igual a 1, y el cronograma demuestra que todas las tareas cumplan sus plazos.

10 Ejercicio 100

10.1. Enunciado

Verificar la planificabilidad utilizando el algoritmo **EDF (Earliest Deadline First)** de asignación dinámica de prioridades a las tareas y construir el diagrama de ejecución de tareas para el siguiente conjunto:

Tarea	Ci	Di	Ti
T1	2	5	6
T2	2	4	8
T3	4	8	12

Cuadro 19: Conjunto de tareas periódicas

²Se esta tomando como referencia que $T_s = 1ms$ para la realización de los cronogramas para ver mejor la transición de lo que pasa en el tiempo real, pero lo aconsejable es hacerlo calculado el T_s como se menciona en el primer ejercicio de esta relación.

10.2. Solución

10.2.1. Paso 1: Test del factor de utilización

Para verificar la planificabilidad inicial, calculamos el factor de utilización del procesador:

$$U = \sum_{i=1}^N \frac{C_i}{T_i}.$$

Para este conjunto de tareas:

$$U = \frac{2}{6} + \frac{2}{8} + \frac{4}{12} = 0,333 + 0,25 + 0,333 = 0,916.$$

Dado que $U \leq 1$, el sistema es potencialmente planificable bajo el algoritmo EDF. Sin embargo, para garantizar que todas las tareas cumplen sus plazos, construimos el cronograma.

10.2.2. Paso 2: Hiperperíodo del sistema

El hiperperíodo (T_m) es el mínimo común múltiplo de los períodos (T_i):

$$T_m = \text{mcm}(6, 8, 12) = 24 \text{ ms.}$$

10.2.3. Paso 3: Construcción del cronograma

Usamos el algoritmo EDF para construir el cronograma. Este algoritmo asigna prioridades dinámicas según el tiempo límite más cercano (D_i) de cada tarea en el instante de planificación.

Tiempo (ms)	0-2	2-4	4-6	6-8	8-10	10-12	12-14	14-16	16-18
T1	X	-	X	-	-	-	X	-	X
T2	-	X	-	X	-	-	-	X	-
T3	-	-	-	-	X	X	-	-	X

Cuadro 20: Cronograma de ejecución de las tareas en el intervalo $[0, T_m = 24]$

Explicación del cronograma:

- **T1:** Se ejecuta durante $C_1 = 2$ ms con un período $T_1 = 6$ ms y un tiempo límite $D_1 = 5$. Se activa en los intervalos 0-2, 4-6, 12-14, etc.
- **T2:** Se ejecuta durante $C_2 = 2$ ms con un período $T_2 = 8$ ms y un tiempo límite $D_2 = 4$. Se activa en los intervalos 2-4, 6-8, 14-16, etc.
- **T3:** Se ejecuta durante $C_3 = 4$ ms con un período $T_3 = 12$ ms y un tiempo límite $D_3 = 8$. Se activa en los intervalos 8-12, 16-20, etc.

10.2.4. Paso 4: Verificación de los plazos

Verificamos que cada tarea cumple su tiempo límite (D_i) observando el cronograma:

- **T1:** Todas sus ejecuciones terminan antes del tiempo límite de 5 ms después de su activación.
- **T2:** Todas sus ejecuciones terminan antes del tiempo límite de 4 ms después de su activación.
- **T3:** Todas sus ejecuciones terminan antes del tiempo límite de 8 ms después de su activación.

10.2.5. Conclusión

El sistema es planificable bajo el algoritmo EDF, ya que:

- El factor de utilización es menor o igual a 1 ($U = 0,916 \leq 1$).
- Todas las tareas cumplen sus tiempos límite (D_i).

11 Ejercicio 101: Verificación de planificabilidad con Deadline Monotonic (DM)

11.1. Enunciado

Verificar la planificabilidad del conjunto de tareas descrito en el ejercicio 100 utilizando el algoritmo del **plazo de respuesta máximo (D)** o **Deadline Monotonic (DM)**.

11.2. Solución

El algoritmo **DM (Deadline Monotonic)** asigna prioridades estáticas a las tareas en función de su plazo de respuesta máximo (D_i): a menor D_i , mayor prioridad. Evaluaremos si el conjunto de tareas es planificable bajo este esquema.

11.2.1. Paso 1: Conjunto de tareas

Para referencia, el conjunto de tareas es:

Tarea	Ci	Di	Ti
T1	2	5	6
T2	2	4	8
T3	4	8	12

Cuadro 21: Conjunto de tareas periódicas

11.2.2. Paso 2: Cálculo del tiempo de respuesta (R_i)

Para determinar si las tareas son planificables, calculamos el tiempo de respuesta (R_i) para cada tarea mediante la fórmula iterativa:

$$R_i^{(k+1)} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_j^{(k)}}{T_j} \right\rceil C_j$$

Criterio de planificabilidad: Si $R_i \leq D_i$ para todas las tareas, el sistema es planificable.

Tarea T1 ($D_1 = 5$):

$$R_1 = C_1 = 2$$

Como $R_1 = 2 \leq D_1 = 5$, la tarea T_1 es planificable.

Tarea T2 ($D_2 = 4$): Iteramos el cálculo:

$$R_2^{(0)} = C_2 = 2$$

$$R_2^{(1)} = C_2 + \left\lceil \frac{R_2^{(0)}}{T_1} \right\rceil C_1 = 2 + \left\lceil \frac{2}{6} \right\rceil 2 = 2 + 2 = 4$$

Como $R_2 = 4 \leq D_2 = 4$, la tarea T_2 es planificable.

Tarea T3 ($D_3 = 8$): Iteramos el cálculo:

$$R_3^{(0)} = C_3 = 4$$

$$R_3^{(1)} = C_3 + \left\lceil \frac{R_3^{(0)}}{T_1} \right\rceil C_1 + \left\lceil \frac{R_3^{(0)}}{T_2} \right\rceil C_2$$

$$R_3^{(1)} = 4 + \left\lceil \frac{4}{6} \right\rceil 2 + \left\lceil \frac{4}{8} \right\rceil 2 = 4 + 2 + 2 = 8$$

Como $R_3 = 8 \leq D_3 = 8$, la tarea T_3 es planificable.

11.2.3. Paso 3: Conclusión

Todas las tareas cumplen sus plazos máximos ($R_i \leq D_i$):

- $R_1 = 2 \leq D_1 = 5$
- $R_2 = 4 \leq D_2 = 4$
- $R_3 = 8 \leq D_3 = 8$

Por lo tanto, el conjunto de tareas es planificable bajo el algoritmo **Deadline Monotonic (DM)**.

12 Ejercicio 102: Análisis de afirmaciones sobre inversión de prioridad

12.1. Enunciado

Indicar cuáles de las siguientes afirmaciones son correctas respecto de los algoritmos para resolver el problema de inversión de prioridad de las tareas en sistemas de tiempo real de misión crítica:

- (a) Suponiendo que las tareas se planifican con el protocolo de herencia de prioridad: la prioridad heredada por una tarea sólo se mantiene mientras dicha tarea esté utilizando un recurso compartido con otra tarea más prioritaria.
- (b) Con el protocolo de techo de prioridad, cuando una tarea adquiere un recurso no puede verse interrumpida, hasta que termine su ejecución, por otras tareas que se activan después que ésta y que vayan a utilizar en el futuro un recurso con límite de prioridad igual o inferior.
- (c) Con el protocolo de techo de prioridad (PPP), una tarea no puede comenzar a ejecutarse si no están libres todos los recursos que va a utilizar durante su primer ciclo.
- (d) Si consideramos una tarea periódica que utilice el protocolo de techo de prioridad inmediato para cambiar su prioridad dinámica cuando accede a recursos, siempre se cumplirá que dicha tarea no puede ser interrumpida por otra menos prioritaria que ella.
- (e) Con el protocolo de techo de prioridad las tareas más prioritarias del sistema pueden ser interrumpidas durante cada ciclo de su ejecución como máximo 1 vez cuando acceden a recursos que comparten con otras tareas menos prioritarias.
- (f) El protocolo de techo de prioridad original producirá siempre tiempos de respuesta menores para las tareas que el algoritmo de herencia de prioridad.

12.2. Solución

12.2.1. Análisis de cada afirmación

- **(a) Correcta.** Según el protocolo de herencia de prioridad, una tarea que bloquea a otra más prioritaria hereda su prioridad mientras esté usando el recurso compartido. Al liberar el recurso, la tarea vuelve a su prioridad original.
- **(b) Incorrecta.** El protocolo de techo de prioridad asegura que una tarea puede ser interrumpida por otra más prioritaria, incluso si la nueva tarea no accede inmediatamente a un recurso con un techo de prioridad igual o inferior.
- **(c) Incorrecta.** Con el protocolo de techo de prioridad, una tarea puede comenzar a ejecutarse incluso si otros recursos que usará más adelante no están libres en ese momento.

- **(d) Incorrecta.** Con el protocolo de techo de prioridad inmediato, una tarea puede ser interrumpida por otra tarea que herede una prioridad más alta al acceder a un recurso compartido.
- **(e) Correcta.** Con el protocolo de techo de prioridad, una tarea puede ser bloqueada como máximo una vez por otras tareas menos prioritarias durante cada ciclo de su ejecución.
- **(f) Incorrecta.** El protocolo de techo de prioridad no garantiza tiempos de respuesta menores en todos los casos respecto al protocolo de herencia de prioridad. Dependerá de la configuración específica del sistema y las tareas.

12.2.2. Conclusión

Las afirmaciones **(a)** y **(e)** son **correctas**, mientras que las restantes son **incorrectas**.

13 Ejercicio 103: Utilización máxima del procesador para un Servidor Esporádico

13.1. Enunciado

Calcular la utilización máxima del procesador que se puede asignar al Servidor Esporádico para garantizar la planificabilidad del siguiente conjunto de tareas periódicas utilizando el algoritmo **Rate Monotonic (RM)**.

Tarea	C _i	T _i
τ_1	1	5
τ_2	2	8

Cuadro 22: Conjunto de tareas periódicas

13.2. Solución

13.2.1. Paso 1: Utilización del procesador por las tareas periódicas

La utilización del procesador por las tareas periódicas (U_p) se calcula como:

$$U_p = \sum_{i=1}^N \frac{C_i}{T_i}$$

Sustituyendo los valores de las tareas:

$$U_p = \frac{1}{5} + \frac{2}{8} = 0,2 + 0,25 = 0,45$$

13.2.2. Paso 2: Límite de utilización máxima para RM

El límite de utilización máxima (U_{max}) para $N = 3$ tareas (las dos tareas periódicas más el Servidor Esporádico) es:

$$U_{max}(3) = 3 \cdot (2^{\frac{1}{3}} - 1) \approx 0,779$$

13.2.3. Paso 3: Utilización restante para el Servidor Esporádico

La utilización máxima que puede asignarse al Servidor Esporádico (U_s) es:

$$U_s = U_{max}(3) - U_p$$

Sustituyendo los valores:

$$U_s = 0,779 - 0,45 = 0,329$$

13.3. Conclusión

La utilización máxima del procesador que se puede asignar al Servidor Esporádico para garantizar la planificabilidad del conjunto de tareas periódicas utilizando RM es:

$$U_s = 0,329$$

14 Ejercicio 104: Utilización máxima del procesador para un Servidor Diferido (SD)**14.1. Enunciado**

Calcular la utilización máxima del procesador que puede ser asignada al Servidor Diferido (SD) para garantizar la planificabilidad del conjunto de tareas periódicas dado en el ejercicio 103 anterior.

Tarea	Ci	Ti
τ_1	1	5
τ_2	2	8

Cuadro 23: Conjunto de tareas periódicas

14.2. Solución**14.2.1. Paso 1: Fórmula para el límite de utilización máxima**

Para un Servidor Diferido (SD), la utilización máxima del procesador se calcula utilizando la fórmula de límite superior de utilización para N tareas periódicas y el servidor:

$$U_{mls} = U_s + N \cdot \left(\left(\frac{U_s + 2}{2U_s + 1} \right)^{\frac{1}{N}} - 1 \right)$$

Donde:

- U_s : Utilización asignada al servidor.
- N : Número de tareas periódicas más el servidor ($N = 3$).

14.2.2. Paso 2: Sustitución de valores

Las tareas periódicas tienen una utilización total de:

$$U_p = \frac{1}{5} + \frac{2}{8} = 0,45$$

Queremos calcular el valor máximo de U_s tal que $U_{mls} \geq U_p + U_s$.

14.2.3. Paso 3: Resolución numérica

Sustituyendo $U_p = 0,45$ y $N = 3$ en la fórmula, debemos resolver:

$$0,45 + U_s \leq U_s + 3 \cdot \left(\left(\frac{U_s + 2}{2U_s + 1} \right)^{\frac{1}{3}} - 1 \right)$$

Resolviendo numéricamente (por iteración o métodos de aproximación), obtenemos:

$$U_s \approx 0,204$$

14.3. Conclusión

La utilización máxima del procesador que puede ser asignada al Servidor Diferido (SD) para garantizar la planificabilidad del conjunto de tareas periódicas es:

$$U_s \approx 0,204$$

15 Ejercicio 105: Planificación de tareas aperiódicas con Servidor Diferido (SD)

15.1. Enunciado

Junto con las tareas periódicas que se muestran en el ejercicio 103, definir un plan para planificar las siguientes tareas aperiódicas utilizando un Servidor Diferido (SD), que posea una utilización máxima del tiempo del procesador y prioridad intermedia.

Tarea Aperiódica	Ci
J_1	2
J_2	7
J_3	17

Cuadro 24: Tareas aperiódicas

15.2. Solución

15.2.1. Paso 1: Utilización máxima del SD

Del ejercicio 104, sabemos que la utilización máxima del Servidor Diferido (SD) para garantizar la planificabilidad de las tareas periódicas es:

$$U_s = 0,204$$

El período del SD (T_s) se define de manera que cumpla:

$$U_s = \frac{C_s}{T_s}$$

Podemos elegir un valor adecuado para T_s . Por simplicidad, tomamos $T_s = 10$:

$$C_s = U_s \cdot T_s = 0,204 \cdot 10 = 2,04 \approx 2$$

El SD tendrá un tiempo de cómputo $C_s = 2$ y un período $T_s = 10$.

15.2.2. Paso 2: Planificación de tareas periódicas y SD

El conjunto de tareas periódicas, junto con el SD, se planificará usando el algoritmo **Rate Monotonic (RM)**. Asignamos las prioridades:

- τ_1 : Período $T_1 = 5$ (mayor prioridad).
- τ_2 : Período $T_2 = 8$.
- SD: Período $T_s = 10$ (prioridad intermedia).

15.2.3. Paso 3: Planificación de tareas aperiódicas con el SD

El SD atiende las tareas aperiódicas en sus ventanas de ejecución. Para $C_s = 2$ y $T_s = 10$, la ventana de ejecución del SD ocurre en cada intervalo de 10 ms.

Planificamos las tareas aperiódicas J_1 , J_2 y J_3 de acuerdo con su tiempo de cómputo (C_i) y en el orden de llegada:

- J_1 : Se completa en el primer intervalo del SD (0-10 ms).
- J_2 : Su tiempo restante se ejecuta parcialmente en los intervalos del SD (10-20 ms y 20-30 ms).
- J_3 : Comienza su ejecución después de que J_2 se completa, y utiliza el intervalo del SD en (30-40 ms).

15.2.4. Paso 4: Cronograma

El cronograma resultante es el siguiente (en ms):

Tiempo	0-5	5-8	8-10	10-15	15-20	20-25
τ_1	X	-	X	-	X	-
τ_2	-	X	-	X	-	X
SD	J_1	-	J_1	J_2	J_2	J_3

Cuadro 25: Cronograma de tareas periódicas y aperiódicas

15.3. Conclusión

El Servidor Diferido garantiza que las tareas aperiódicas J_1 , J_2 y J_3 se ejecuten utilizando un tiempo de procesador máximo $U_s = 0,204$, manteniendo la planificabilidad de las tareas periódicas bajo **Rate Monotonic (RM)**.

16 Ejercicio 106: Planificación con Servidor Esporádico (SS)

16.1. Enunciado

Resolver el problema de planificación descrito en el ejercicio 105, utilizando un **Servidor Esporádico (SS)** que tenga una utilización máxima del procesador y prioridad intermedia.

Tarea Aperiódica	Ci
J_1	2
J_2	7
J_3	17

Cuadro 26: Tareas aperiódicas

El conjunto de tareas periódicas está definido en el ejercicio 103 como:

Tarea Periódica	Ci	Ti
τ_1	1	5
τ_2	2	8

Cuadro 27: Conjunto de tareas periódicas

16.2. Solución

16.2.1. Paso 1: Utilización máxima del Servidor Esporádico (SS)

Del ejercicio 103, sabemos que la utilización máxima asignada al servidor es $U_s = 0,204$. La relación entre el tiempo de cómputo (C_s) y el período (T_s) del servidor es:

$$U_s = \frac{C_s}{T_s}$$

Eligiendo $T_s = 10$, calculamos:

$$C_s = U_s \cdot T_s = 0,204 \cdot 10 = 2,04 \approx 2$$

El Servidor Esporádico tiene un tiempo de cómputo $C_s = 2$ y un período $T_s = 10$.

16.2.2. Paso 2: Funcionamiento del Servidor Esporádico

El Servidor Esporádico (SS) atiende tareas aperiódicas asignando tiempo de procesador según su tiempo disponible (C_s) y regenerando este tiempo en cada período (T_s). El servidor solo se activa cuando hay tareas aperiódicas pendientes en la cola de ejecución.

A diferencia del Servidor Diferido, el SS permite reutilizar el tiempo no consumido en un período, lo que puede mejorar la latencia para las tareas aperiódicas.

16.2.3. Paso 3: Planificación de las tareas aperiódicas

Las tareas aperiódicas (J_1, J_2, J_3) se ejecutarán en orden de llegada, utilizando el tiempo asignado al SS. Asumimos los tiempos de llegada:

- J_1 : Llega en $t = 0$.
- J_2 : Llega en $t = 8$.
- J_3 : Llega en $t = 18$.

16.2.4. Paso 4: Cronograma de ejecución

Tiempo (ms)	0-5	5-8	8-10	10-15	15-18	18-20
τ_1	X	-	X	-	X	-
τ_2	-	X	-	X	-	X
SS	J_1	-	J_1	J_2	J_2	J_3

Cuadro 28: Cronograma de tareas periódicas y aperiódicas utilizando el SS

Explicación del cronograma:

- En $t = 0$, J_1 comienza a ejecutarse utilizando el tiempo del SS ($C_s = 2$).
- En $t = 8$, J_2 llega y utiliza el tiempo disponible del SS en los períodos subsiguientes.
- En $t = 18$, J_3 comienza a ejecutarse y continúa en el siguiente intervalo del SS.

16.3. Conclusión

El Servidor Esporádico ($U_s = 0,204$) permite planificar las tareas aperiódicas sin afectar la planificabilidad de las tareas periódicas. Además, su capacidad de regeneración mejora la latencia de respuesta para las tareas aperiódicas en comparación con el Servidor Diferido.

17 Ejercicio 107: Planificación con Servidor Diferido (DS)

17.1. Enunciado

Resolver el problema de planificación descrito en el ejercicio 105, utilizando un **Servidor Diferido (DS)** que tenga una utilización máxima del procesador y prioridad intermedia.

Tarea Aperiódica	Ci
J_1	2
J_2	7
J_3	17

Cuadro 29: Tareas aperiódicas

El conjunto de tareas periódicas está definido en el ejercicio 103 como:

Tarea Periódica	Ci	Ti
τ_1	1	5
τ_2	2	8

Cuadro 30: Conjunto de tareas periódicas

17.2. Solución

17.2.1. Paso 1: Utilización máxima del Servidor Diferido (DS)

La utilización del procesador del conjunto de tareas periódicas se calcula como:

$$U_{\text{periódicas}} = \sum_{i=1}^n \frac{C_i}{T_i} = \frac{1}{5} + \frac{2}{8} = 0,2 + 0,25 = 0,45$$

Para garantizar la planificabilidad del sistema con el Servidor Diferido, el total de utilización debe ser menor o igual a 1:

$$U_{\text{total}} = U_{\text{periódicas}} + U_s \leq 1$$

Donde U_s es la utilización asignada al servidor diferido. Resolviendo:

$$U_s \leq 1 - U_{\text{periódicas}} = 1 - 0,45 = 0,55$$

La relación entre el tiempo de cómputo (C_s) y el período (T_s) del servidor diferido es:

$$U_s = \frac{C_s}{T_s}$$

Eligiendo $T_s = 10$, calculamos:

$$C_s = U_s \cdot T_s = 0,55 \cdot 10 = 5,5 \approx 5$$

Por lo tanto, el Servidor Diferido tiene $C_s = 5$ y $T_s = 10$.

17.2.2. Paso 2: Funcionamiento del Servidor Diferido

El Servidor Diferido (DS) preserva el tiempo asignado (C_s) hasta que haya tareas aperiódicas pendientes de ejecución. Cada período (T_s) el servidor recupera su tiempo de cómputo máximo ($C_s = 5$). Dado que el DS se ejecuta con prioridad intermedia, las tareas periódicas más prioritarias no se ven afectadas.

17.2.3. Paso 3: Planificación de las tareas aperiódicas

Las tareas aperiódicas (J_1, J_2, J_3) se ejecutarán utilizando el tiempo asignado al DS. Asumimos los tiempos de llegada:

- J_1 : Llega en $t = 0$.
- J_2 : Llega en $t = 8$.
- J_3 : Llega en $t = 18$.

17.2.4. Paso 4: Cronograma de ejecución

Tiempo (ms)	0-5	5-8	8-10	10-15	15-18	18-20
τ_1	X	-	X	-	X	-
τ_2	-	X	-	X	-	X
DS	J_1	-	J_1	J_2	J_2	J_3

Cuadro 31: Cronograma de tareas periódicas y aperiódicas utilizando el DS

Explicación del cronograma:

- En $t = 0$, J_1 comienza a ejecutarse utilizando $C_s = 5$.
- En $t = 8$, J_2 llega y utiliza el tiempo restante del DS en $T_s = 10$.
- En $t = 18$, J_3 utiliza el siguiente período del DS.

17.3. Conclusión

El Servidor Diferido ($U_s = 0,55$) permite planificar las tareas aperiódicas sin afectar la planificabilidad de las tareas periódicas. Aunque el DS garantiza una latencia predecible, puede ser menos eficiente que el Servidor Esporádico en términos de reutilización del tiempo no consumido.

18 Ejercicio 108: Planificación con Servidor Esporádico (SS)**18.1. Enunciado**

Utilizar un **Servidor Esporádico (SS)** con capacidad $C_s = 2$ y período $T_s = 6$ para planificar las siguientes tareas:

Tarea Periódica	Ci	Ti
τ_1	1	4
τ_2	2	6

Cuadro 32: Conjunto de tareas periódicas

Tarea Aperiódica	Ci	ai
J_1	2	2
J_2	5	1
J_3	10	2

Cuadro 33: Conjunto de tareas aperiódicas

18.2. Solución

18.2.1. Paso 1: Verificación de la planificabilidad de las tareas periódicas

La utilización total del procesador por las tareas periódicas es:

$$U_{\text{periodicas}} = \sum_{i=1}^n \frac{C_i}{T_i} = \frac{1}{4} + \frac{2}{6} = 0,25 + 0,333 = 0,583$$

Con el Servidor Esporádico (U_s), la utilización total del procesador es:

$$U_{\text{total}} = U_{\text{periodicas}} + U_s$$

La utilización del SS se calcula como:

$$U_s = \frac{C_s}{T_s} = \frac{2}{6} = 0,333$$

Entonces:

$$U_{\text{total}} = 0,583 + 0,333 = 0,916$$

Dado que $U_{\text{total}} < 1$, el sistema es planificable.

18.2.2. Paso 2: Funcionamiento del Servidor Esporádico (SS)

El SS asigna su capacidad ($C_s = 2$) únicamente cuando hay tareas aperiódicas en espera. Una vez consumido su tiempo, el servidor regenera su capacidad en el próximo período ($T_s = 6$).

18.2.3. Paso 3: Planificación de las tareas aperiódicas

Las tareas aperiódicas (J_1, J_2, J_3) se ejecutan en orden de llegada, utilizando el tiempo asignado al SS. Asumimos los tiempos de llegada (a_i):

- J_1 : Llega en $t = 2$.
- J_2 : Llega en $t = 1$.
- J_3 : Llega en $t = 2$.

18.2.4. Paso 4: Cronograma de ejecución

Tiempo (ms)	0-1	1-2	2-4	4-6	6-8	8-10	10-12
τ_1	X	-	X	-	X	-	X
τ_2	-	X	-	X	-	X	-
SS	-	J_2	J_1	J_3	J_3	-	-

Cuadro 34: Cronograma de tareas periódicas y aperiódicas utilizando el SS

Explicación del cronograma:

- En $t = 1$, J_2 comienza a ejecutarse con el tiempo del SS ($C_s = 2$).
- En $t = 2$, J_1 y J_3 llegan. El SS prioriza J_1 por orden de llegada.
- En $t = 6$, el SS regenera su capacidad y continúa ejecutando J_3 .

18.3. Conclusión

El Servidor Esporádico ($C_s = 2$, $T_s = 6$) permite planificar las tareas aperiódicas sin afectar la planificabilidad de las tareas periódicas. Su capacidad de regenerar tiempo no consumido optimiza el uso del procesador.