TD temps réel (version 6.0)

1 Mise en application de RM, EDF et LLF

Soit les tâches périodiques synchrones (prêtes à t = 0) à échéances implicites (D=T).

T1 : (C=1, T=3)
T2 : (C=1, T=4)
T3 : (C=2, T=6)

Pour chacune des politiques d'ordonnancement RM, EDF et LLF :

- Calculer U, le facteur d'utilisation du processeur, que peut-on en conclure ?
- Donner le chronogramme des tâches. Quel commentaire peut-on faire ?

Indication : $n(2^{(1/n)}-1) = 0{,}78$ pour $n = 3$.

2 Calcul du temps de réponse

Soit les tâches périodiques synchrones (prêtes à t = 0) à échéances implicites (D=T).

T1 : (C=25, T=100)
T2 : (C=50, T=200)
T3 : (C=100, T=300)

Calculer leur temps de réponse pour déterminer si elles sont ordonnançables avec RMS.
Rappel : $R_N(t) = \Sigma_{j \text{ in } hp(i)} C_j * \lceil t/T_j \rceil$ avec hp(i) contenant les tâches plus prioritaires que la tâche i

3 Serveurs de tâches apériodiques

3.1 T1 tâche périodique

Soit les tâches périodiques synchrones (prêtes à t = 0) à échéances implicites (D=T).

T1: (C=2, T=10)
T2: (C=4, T=14)
T3: (C=5, T=14)

Démontrer sans faire le chronogramme complet que le système est ordonnançable.

3.2 T1 serveur différé

On suppose désormais que T1 est un serveur différé. On introduit dans le système précédent A, une tâche apériodique caractérisée par : réveil en 28, C=6
Donner le chronogramme illustrant l'ordonnancement de T1, T2, T3 et A. Que se passe-t-il ?

3.3 T1 serveur scrutation

On suppose désormais que T1 est un serveur à scrutation. On introduit la même tâche apériodique. Donner le chronogramme et le temps de réponse de A.

3.4 T1 serveur sporadique

On suppose désormais que T1 est un serveur sporadique. On introduit la même tâche apériodique. Donner le chronogramme et le temps de réponse de A.

4 Integrated Modular Avionic

Soit les tâches périodiques synchrones (prêtes à t = 0) à échéances implicites (D=T). Elles s'exécutent sur une plate-forme partitionnée ARINC 653 pour mono-processeur.

Nom	Capacité	Période/Echéance
T1	1	3
T2	1	6
T3	2	6
T4	2	12

On souhaite appliquer un ordonnancement Rate Monotonic préemptif. En cas d'égalité de priorité, on utilise l'ordre lexicographique.

4.1 Question 1

Expliquer pourquoi le test d'ordonnançabilité ne permet pas de conclure sur l'ordonnançabilité de ce lot de tâches. Etablir le chronogramme sur l'hyper-période.

4.2 Question 2

Les tâches T1 et T3 ont même niveau de criticité A et T2 et T4 ont même niveau de criticité B. On souhaite rassembler les tâches par niveau de criticité pour former des partitions. En s'appuyant sur l'ordonnancement précédent et en supposant que l'ordonnancement des partitions reste Rate Monotonic, énumérer les fenêtres de partition (*partition windows*) que cela nécessite ainsi que le nombre de changement de partition.

4.3 Question 3

On souhaite réduire ce nombre de changement. Proposer une autre configuration de fenêtres de partition de sorte à réduire le nombre de changement de partition.

5 EDF et LLF en multi-cœurs (m cœurs)

Soit les tâches périodiques synchrones (prêtes à t = 0) à échéances implicites (D=T).

T_1 : (C=2*epsilon, T)

…

T_m : (C=2*epsilon, T)

T_{m+1} : (C=T, T+epsilon)

5.1 Utilisation

Calculer l'utilisation du système lorsque epsilon est petit et T grand.

5.2 Global EDF

Appliquer Global EDF à cet ensemble de tâches. Que constate-t-on ?

5.3 Global LLF

Appliquer Global LLF à cet ensemble de tâches. Que constate-t-on ?

5.4 Conclusions

Que conclure de ces exemples ?

6 Ordonnancements partitionnés en multi-cœurs (m cœurs)

Trouver un ordonnancement de tâches **identiques** périodiques synchrones (prêtes à t = 0) à échéances implicites (D=T) de sorte qu'elles ne soient pas ordonnançables en partitionné mais ordonnançables en global

7 Ordonnancements globaux et partitionnés sur 2 cœurs

Soit les tâches périodiques synchrones (prêtes à t = 0) à échéances implicites (D=T).

Nom	Capacité	Période/Echéance
T1	4	6
T2	7	12

T3	4	12
T4	10	24

7.1 Pfair

Est-ce que le système de tâches est ordonnançable en utilisant PFair. Produire le chronogramme.

7.2 Global-RM

Est-ce que le système de tâches est ordonnançable en utilisant Global-RM. Produire le chronogramme.

7.3 Global-RM

Est-ce que le système de tâches est ordonnançable en utilisant une architecture partitionnée, chaque partition ordonnançant ses tâches avec RMS. Produire le chronogramme.

8 Couverture de code de type MC/DC

On a le code suivant:

F3(x, y, z) : if (((x==1) && (y==2)) ll (z==3)) {F1(x, y, z) ;} ; F2(x, y, z) ;

8.1 2 tests

Selon MC/DC, est-ce que les tests F3(1, 2, 3) et F(2, 2, 3) sont redondants ou complémentaires ? Pourquoi dans les deux cas.

8.2 Table de vérité

Construire la table de vérité.

9 Bus CAN

Rappeler les principes de fonctionnement du bus CAN. Vous détaillerez le résultat de l'émission simultanée de messages sur le bus par des tâches ayant des identificateurs 4, 5, 6 et 7.

10 Ordonnancement avec blocage

On souhaite exécuter sur un mono-processeur un ensemble de tâches partageant 2 verrous S1 et S2.

Nom	C	T	Exécution
T1	2	8	L(S1) ; Exec(1) ; U(S1) ; Exec(1) ;
T2	3	10	L(S1) ; Exec(1) ; U(S1) ; Exec(1) ; L(S2) ; Exec(1) ; U(S2)
T3	3	20	Exec(1) ; L(S2) ; Exec(1) ; U(S2) ; Exec(1) ;
T4	7	40	L(S1) ; Exec(3) ; U(S1) ; Exec(1) ; L(S2) ; Exec(3) ; U(S2)

Soit les tâches périodiques synchrones (prêtes à t = 0) à échéances implicites (D=T). Elles s'exécutent sur une plate-forme partitionnée ARINC 653 pour mono-processeur. On souhaite appliquer un ordonnancement Earliest Deadline First préemptif. On étudie l'ordonnancement

dans le cas de protocoles de synchronisation PIP-EDF et SRP-EDF. Déterminer les temps de blocages. Appliquer les tests d'ordonnancabilité de systèmes en présence de blocage pour EDF.

11 Mise en application de RM, EDF et LLF

U=1/3+1/4+2/6=11/12

Le système est ordonnançable par LLF et EDF car il vérifie la condition nécessaire et suffisante U ≤ 1.

On peut rien dire pour RMS car la condition suffisante n'est pas vérfiée car

U ≥ n($2^{(1/n)}$-1) = 0,78 pour n = 3. Par contre, le chronogramme sur l'hyper-période 12 montre que le système est ordonnançable avec RMS.

12 Calcul du temps de réponse

On calcule par itération le temps de réponse donné par $R_i(n+1)\ \Sigma_{j\leq i}C_j * \lceil R_i(n)/T_j \rceil$ pour toutes les tâches j de plus haute priorité que la tâche i. Sachant que $R_i(0) = C_i$. Le temps de réponse final doit être inférieur à l'échéance.

$R_1(0) = 25$, $R_1(1) = R_1(0) = 25$ or $R_1(1) < D_1$: T1 respecte son échéance avec RMS.

$R_2(0) = 50$, $R_2(1) = 25*\lceil 50/100 \rceil + 50 = 75$, $R_2(2) = R_2(1) = 75$, $R_2(1) < D_2$: T2 respecte son échéance

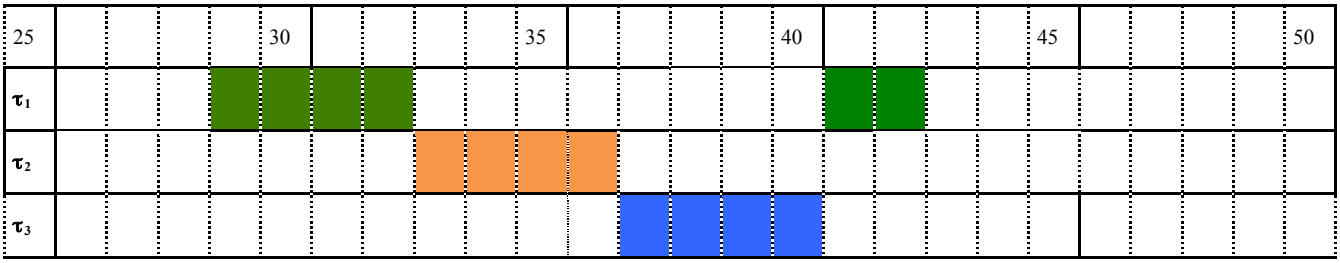
$R_3(0) = 100$, $R_3(1) = 25*\lceil 100/100 \rceil + 50*\lceil 100/200 \rceil + 100 = 175$, $R_3(1) = 25*\lceil 175/100 \rceil + 50*\lceil 175/200 \rceil + 100 = 200$, $R_3(1) = 25*\lceil 200/100 \rceil + 50*\lceil 200/200 \rceil + 100 = 200$ $R_3(3) = R_3(2) = 200$, $R_3(3) < D_3$: T3 respecte son échéance

13 Serveurs de tâches apériodiques

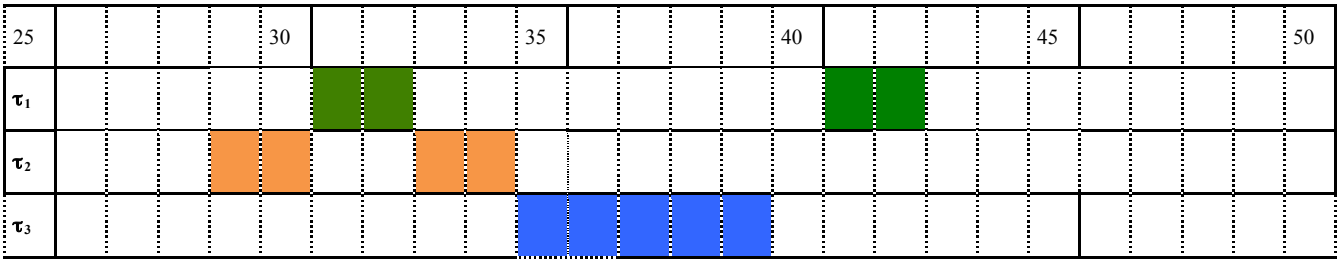
On calcule par itération les temps de réponse pour montrer que le système est ordonnançable.

13.1 T1 serveur différé

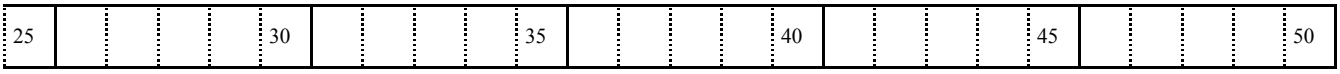
La tâche T3 rate son échéance à T=42.

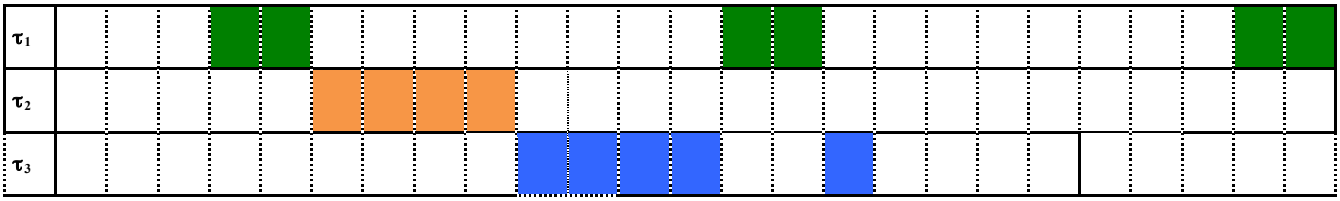


13.2 T1 serveur scrutation



13.3 T1 serveur sporadique





14 Integrated Modular Avionic

14.1 Question 1

U=1 donc avec RMS on ne peut rien déduire.

1	2	3	4	5	6	7	8	9	10	11	12
T1	**T2**	**T3**	**T1**	**T3**	**T4**	**T1**	**T2**	**T3**	**T1**	**T3**	**T4**

14.2 Question 2

Il y a 8 changements de partitions (en comptant l'hyper période comme changement de partitions obligatoire)

P1	**P2**	**P1**	**P1**	**P1**	**P2**	**P1**	**P2**	**P1**	**P1**	**P1**	**P2**
T1	**T2**	**T3**	**T1**	**T3**	**T4**	**T1**	**T2**	**T3**	**T1**	**T3**	**T4**

14.3 Question 3

On essaye de regrouper les tâches T1 et T3 puis T2 et T4. Il y a 3 changements de partitions (en comptant l'hyper période comme changement de partitions obligatoire)

P1	**P1**	**P1**	**P1**	**P2**	**P2**	**P2**	**P2**	**P1**	**P1**	**P1**	**P1**
T1	**T3**	**T3**	**T1**	**T2**	**T4**	**T4**	**T2**	**T1**	**T1**	**T3**	**T3**

15 EDF et LLF en multi-cœur (m cœurs)

15.1 Utilisation

U = (2m/P)*epsilon + P/P+epsilon donc U vaut 1 approximativement. Très inférieur à m.

15.2 Global-EDF

Avec Global-EDF, T1 à Tm démarrent sur les cœurs C1 à Cm. A t=2*epsilon, T1 à Tm se terminent et Tm+1 peut s'exécuter, se termine à 2*epsilon+P et loupe son échéance.

15.3 Global-LLF

Avec Global-LLF, Tm+1 démarre sur C1 et T1 à Tm-1 démarrent sur les autres cœurs puis lorsque T1 se termine sur C2, Tm démarre sur C2 et toutes les tâches respectent leur échéance.

15.4 Conclusions

D'une part, même si U=1 Global-EDF peut échouer en multi-cœurs ! D'autre part, Global-LLF domine Global-EDF.

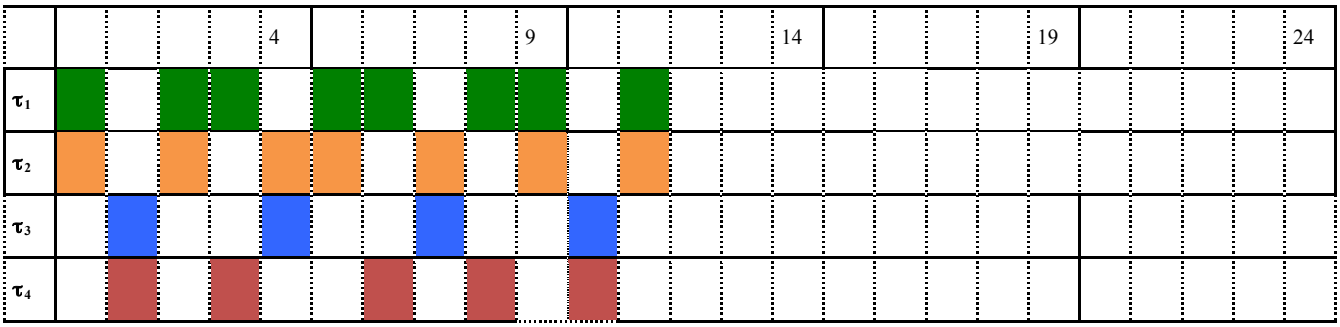
16 Ordonnancements partitionnés sur m cœurs

Il suffit de faire en sorte que l'utilisation de deux tâches (identiques) ne tienne pas sur un cœur (U > 1) et que la somme des utilisations tienne sur m cœurs. Pour ce faire, on choisit m+1 tâches de C=(T+epsilon)/2. 2U = (T+epsilon)/T > 1. Mais (m+1)(T+epsilon)/2T <= m si (T+epsilon) <= m(T-epsilon), donc dès que 2 <= m.

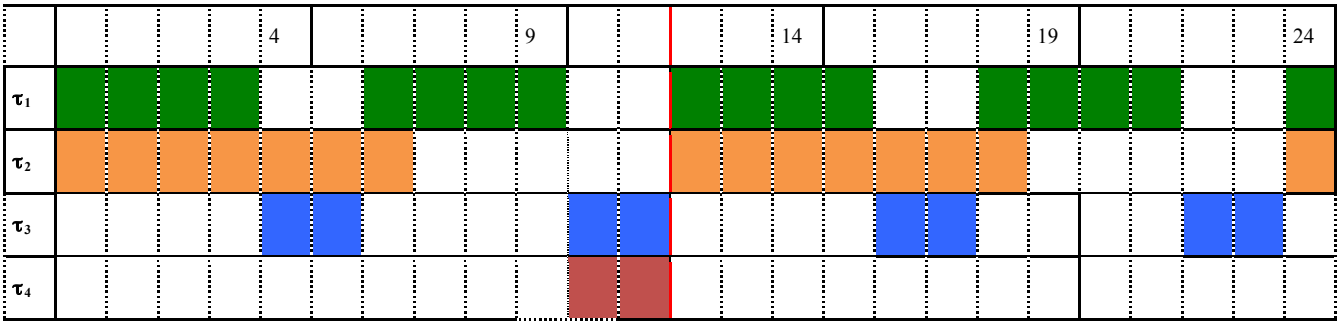
17 Ordonnancements globaux et partitionnés sur 2 cœurs

17.1 PFair

Pour chaque tâche i, on crée Ci sous-tâches (une unité j pour chaque unité de budget) telle que l'activation (resp l'échéance) soit $\lfloor (j-1)/Ui \rfloor$ (resp $\lceil j/Ui \rceil$).
Pour T1, les intervalles sont (0,2), (1,3), (3,5), (4,6), (6,8), (7,9), (9,11), (10,12) (U=4/6)
Pour T2, les échéances sont (0,2), (1,4), (3,6), (5,7), (6,9) (8,11), (10,12) (U=7/12)
Pour T3, les échéances sont (0,3), (3,6), (6,9), (9,12) (U=4/12)
Pour T4, les échéances sont (0,3), (2,5), (4,8), (7,10), (9,12), (12,15), (14,17), (16,20), (19,22) (21, 24). (U=10/24)



17.2 Global-RM



Dépassement d'échéance en t=12.

17.3 Global-RM

L'utilisation globale étant de 2 pour 2 cœurs disponibles, il faut que chaque cœur supporte l'exécution de tâches avec une utilisation de 1. La seule solution consiste à mettre T2 (7/12) avec T4 (10/24) et donc T1 (4/6) avec T3 (4/12). On calcule rapidement que le temps de réponse de T3 est 12 inférieur à son échéance et celui de T4 est 24.

18 Couverture de code de type MC/DC

18.1 Deux tests

Ils sont redondants car comme z==3 on fait varier x et y sans modifier la décision. Le test final est toujours vrai.

18.2 Table de vérité

x==1	y==2	z==3	décision
F	F	V	V
F	F	F	F
F	V	F	F
V	V	F	V

19 Bus CAN

Le bus commence par émettre l'identificateur sur le bus, bit par bit. Le bit 1 est récessif (il perd dans un conflit qui impliquerait au moins un nœud émettant un bit 0). Sur l'exemple, on détaillera bit par bit, les décisions que prennent les nœuds pour déterminer s'ils sont émetteurs ou récepteurs (4=1000, 5=1001, 6=1010, 7=1011). Au final, les nœuds de plus faibles identificateurs sont les plus prioritaires.

20 Ordonnancement avec blocage

Exemple décrit dans le cours.