ENSEEIHT

Département Sciences du Numérique (SN)

Compte Rendu TP1

Réaliser par :
Yassine Zitouni

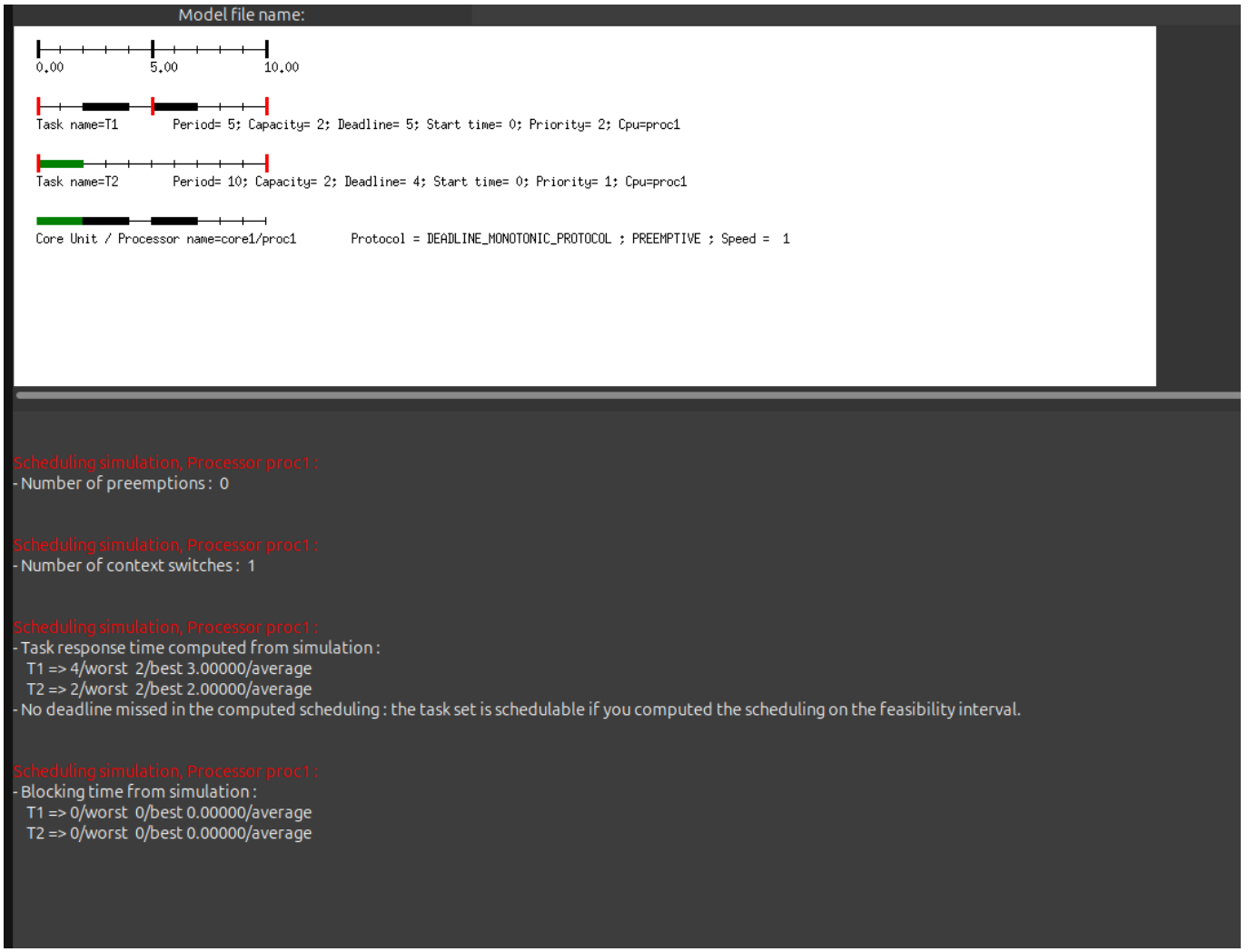
année universitaire : 2024 - 2025

Contents

INTRODUCTION

1 Exercice 1

1.1 Résultats de simulation



1.2 Ordonnabilité des tâches

En appliquant la formule suffisante mais non nécessaire :

$$\sum_{i=1}^{n} \frac{C_i}{D_i} < n\left(2^{1/n} - 1\right)$$

Pour notre cas, cela donne :

$$0.9 < 0.77$$

Ainsi, la condition suffisante n'est pas vérifiée. Cela signifie que, bien que cette formule ne confirme pas l'ordonnabilité du système, elle n'est pas

nécessairement une preuve d'ordonnabilité échouée. Le résultat de simulation montre qu'aucune tâche n'a dépassé sa deadline dans l'intervalle de faisabilité analysé. Cela signifie que l'ensemble des tâches est ordonnable et respecte les contraintes temporelles définies par leurs deadlines respectives.

1.3 Calcul du WCRT pour T_2

T_2 a la plus haute priorité (puisque sa deadline est plus petite). Il n'est donc pas affecté par T_1.

Le WCRT de T_2 est simplement son WCET :

$$R_2 = C_2 = 2$$

1.4 Calcul du WCRT pour T_1

T_1 est affecté par les interférences de T_2, donc nous utilisons l'équation de récursion pour calculer son WCRT.

1.4.0.1 Initialisation :

$$R_1^{(0)} = C_1 = 2$$

1.4.0.2 Première itération :

$$R_1^{(1)} = C_1 + \left\lceil \frac{R_1^{(0)}}{P_2} \right\rceil C_2 = 2 + \left\lceil \frac{2}{10} \right\rceil \cdot 2 = 2 + 1 \cdot 2 = 4$$

1.4.0.3 Deuxième itération (convergence) :

$$R_1^{(2)} = C_1 + \left\lceil \frac{R_1^{(1)}}{P_2} \right\rceil C_2 = 2 + \left\lceil \frac{4}{10} \right\rceil \cdot 2 = 2 + 1 \cdot 2 = 4$$

On constate que $R_1^{(1)} = R_1^{(2)}$, donc le WCRT de T_1 est 4.

1.5 Ordonnabilité avec un ordre de priorité fixe

Comme la configuration est déjà ordonnable avec *Deadline Monotonic* (qui est un algorithme de priorité fixe), la réponse est oui, elle est ordonnable avec un autre ordre de priorité fixe.

2 Exercice 2

2.1 question 1

Avec un algorithme de priorité fixe, dans ce cas *Deadline Monotonic*, la simulation montre que le système est ordonnable.

2.2 question 2

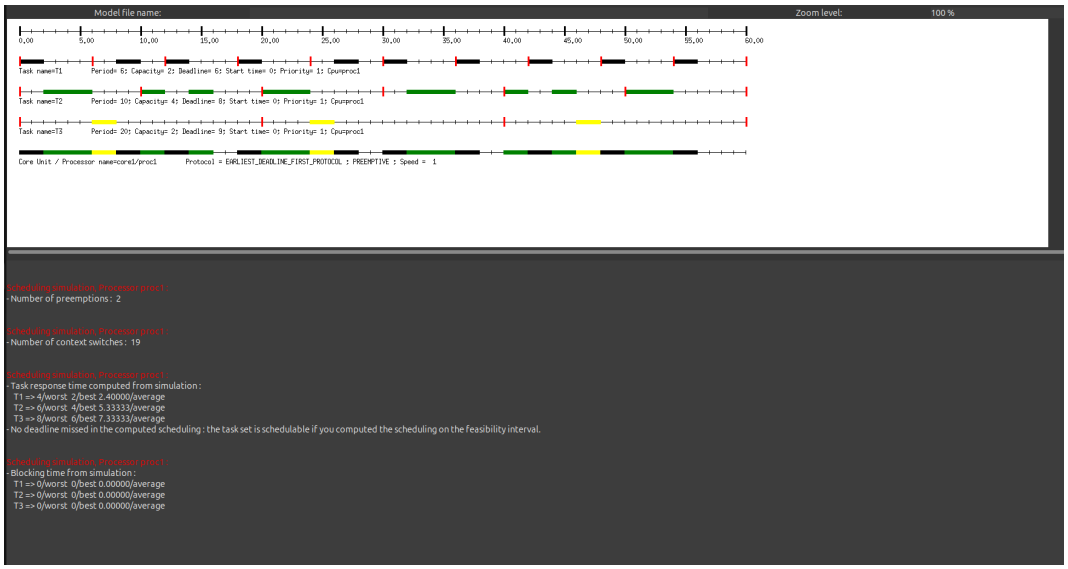
Avec un algorithme de priorité fixe, dans ce cas *Deadline Monotonic*, la simulation montre que le système est ordonnable.

2.3 question 3

2.3.1 algorithm de priotité fixe

Avec un algorithme de priorité fixe, dans ce cas *Deadline Monotonic*, la simulation montre que le système est n'est pas ordonnable.

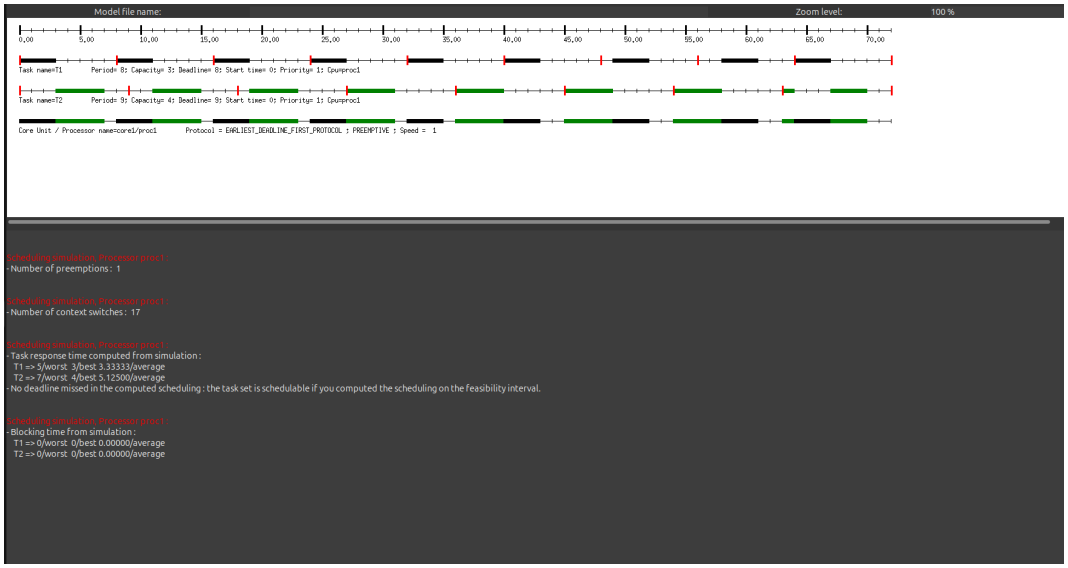
2.3.2 algorithm EDF



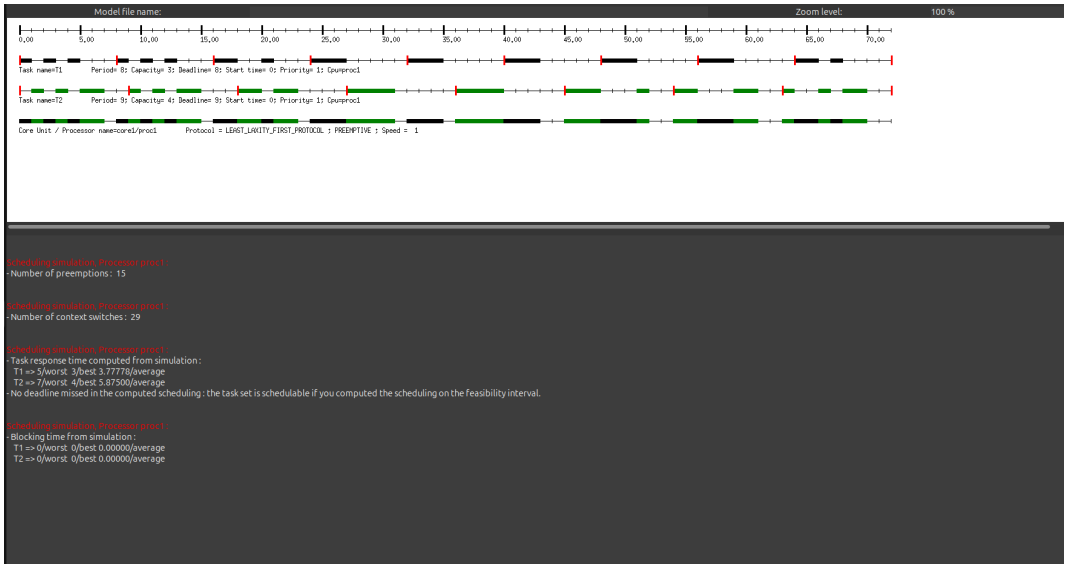
Avec un algorithme EDF la simulation montre que le système est ordonnable.

3 Exercice 3

3.1 scheduling sequences obtained by Ealiest Deadline First



3.2 scheduling sequences obtained byLeast Laxity First



3.3 Analyse comparative

Préemptions et changements de contexte : LLF montre un nombre significativement plus élevé de préemptions et de changements de contexte par rapport à EDF (15 préemptions et 29 changements de contexte contre 1 et 17 respectivement). Cela indique que LLF entraîne plus de basculements entre les tâches, ce qui peut ajouter une surcharge au système en termes de gestion des tâches.

Temps de réponse : Les deux algorithmes réussissent à respecter les échéances, mais LLF a un temps de réponse moyen légèrement plus élevé pour T2 (5.8750 contre 5.1250 pour EDF), ce qui peut être dû à la nature réactive et dynamique de l'ajustement de priorité dans LLF.

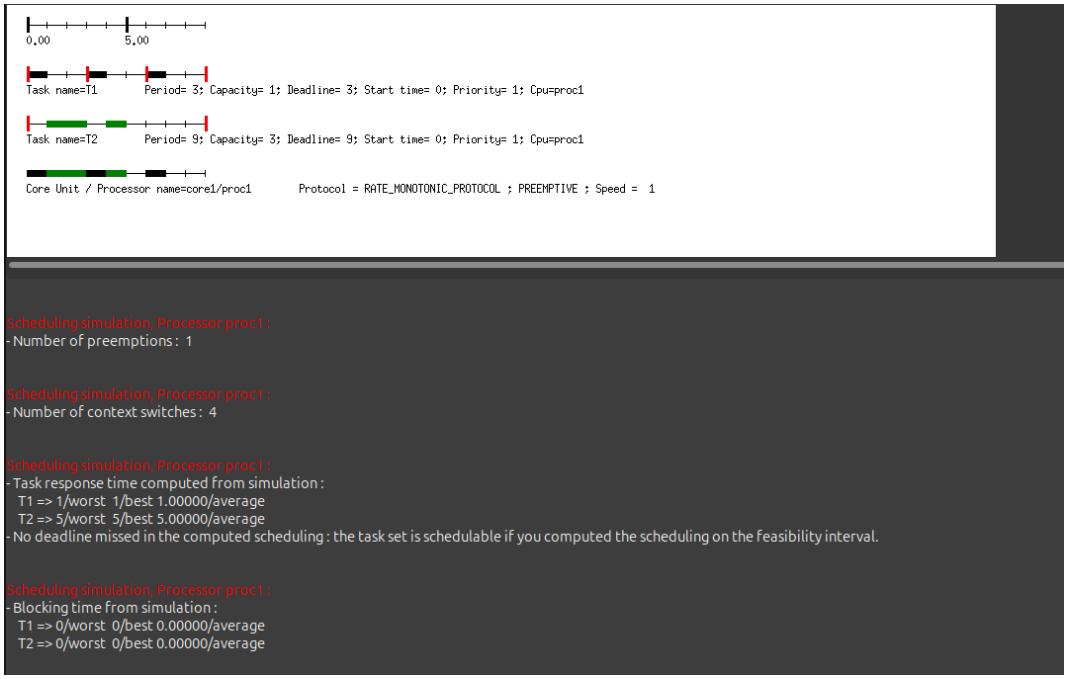
Planifiabilité : Les deux algorithmes garantissent que toutes les échéances sont respectées, mais EDF semble plus efficace en termes de stabilité avec moins de perturbations du planificateur, tandis que LLF, bien qu'il respecte aussi les délais, le fait au prix d'une complexité accrue.

Conclusion : EDF semble être l'option plus stable et plus simple, avec moins de préemptions et de changements de contexte, tandis que LLF offre une réactivité plus granulaire mais au prix d'une gestion plus complexe, avec plus de préemptions et de basculements fréquents entre les tâches.

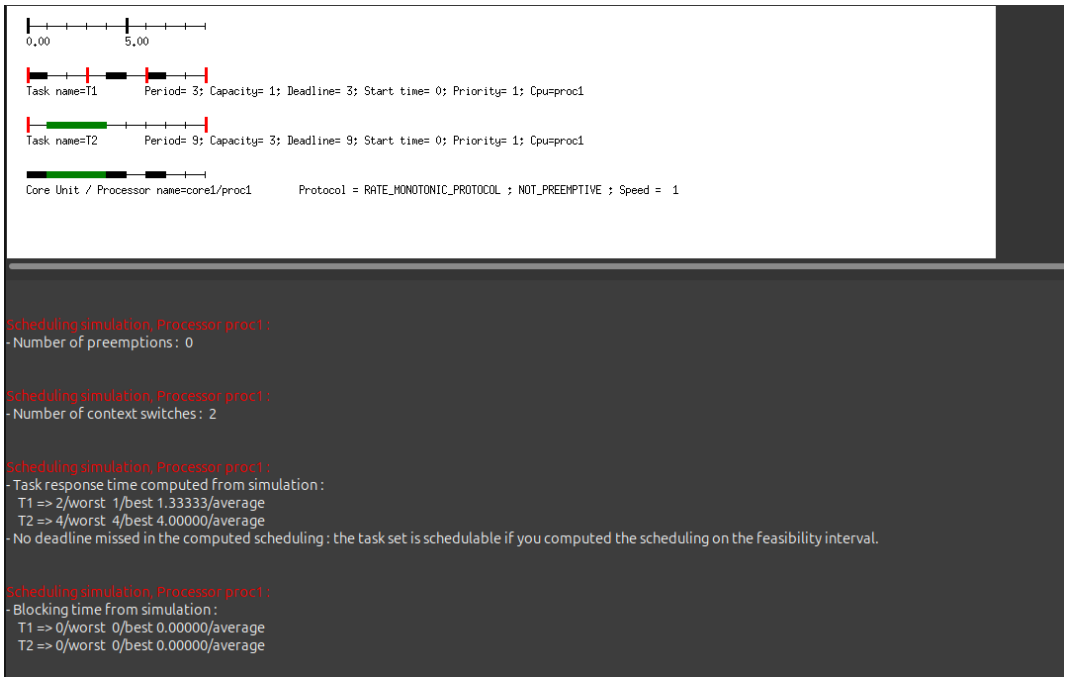
4 Exercice 4

4.1 Exemple 1

4.1.1 Preemptive Rate Monotonic



4.1.2 Non Premptive Rate Monotonic

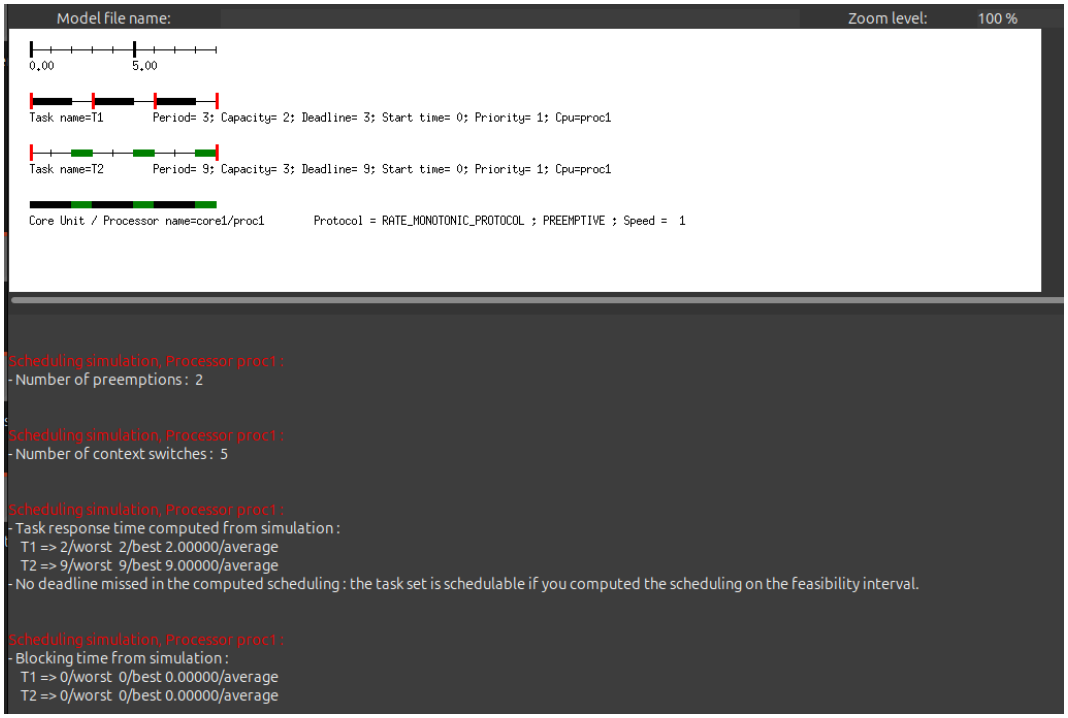


Dans cette simulation, nous avons comparé les résultats d'une planification préemptive et non-préemptive en utilisant le Rate Monotonic Protocol.

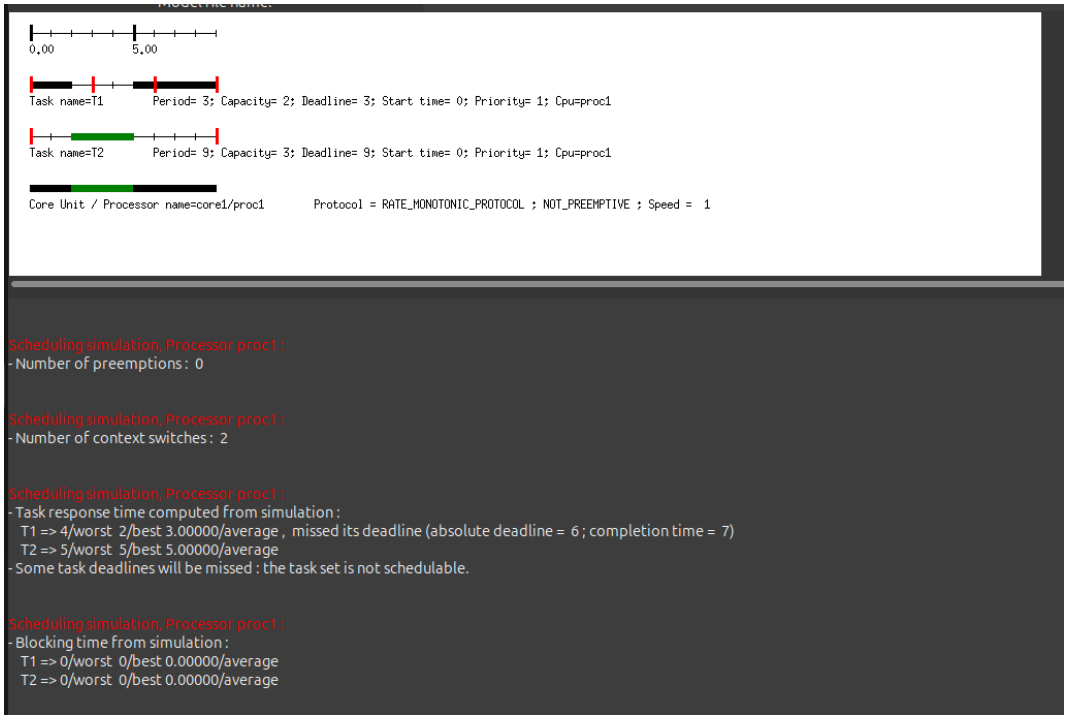
En mode préemptif, nous observons une réactivité accrue, notamment pour les tâches à période courte comme T1, au prix d'un nombre plus élevé de préemptions (1) et de changements de contexte (4). En revanche, en mode non-préemptif, bien qu'il n'y ait aucune préemption et que les changements de contexte soient réduits (2), les temps de réponse peuvent être légèrement allongés pour les tâches urgentes. Cela illustre l'inévitable compromis entre réactivité et simplicité de gestion dans les systèmes temps réel.

4.2 Exemple 2

4.2.1 Premptive Rate Monotonic



4.2.2 Non Premptive Rate Monotonic

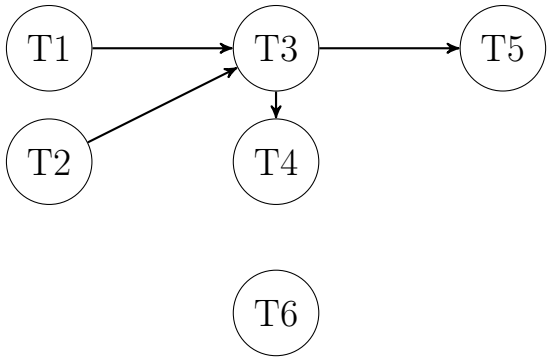


Dans cette situation, le mode préemptif permet de respecter toutes les échéances, malgré un plus grand nombre de préemptions et de changements de contexte. Le mode non-préemptif, en revanche, échoue à respecter l'échéance de la tâche T1, car celle-ci est retardée par la tâche T2 qui prend plus de temps. Cela démontre une fois de plus que le mode préemptif offre une meilleure capacité à répondre aux contraintes strictes de temps réel, surtout lorsque les tâches ont des périodes et priorités différentes, mais cela se fait au prix d'une gestion plus complexe des interruptions et des changements de contexte. Le mode non-préemptif, bien qu'il réduise ces coûts de gestion, peut entraîner des dépassements d'échéances dans des scénarios de concurrence pour le processeur.

5 Exercice 5

5.1 Algorithme de Rate Monotonic

La graphe suivante représente le graphe de contraintes de précédence. On constate que $T6$ a la période la plus élevée. Selon l'algorithme de Rate Monotonic, $T6$ est donc la tâche la moins prioritaire. En effet, plus la période d'une tâche est courte, plus elle a une haute priorité, et vice versa.



Un calcul basé sur les contraintes de précédence calculées précédemment et sur les formules suivantes pour le calcul du premier relâchement et de la deadline optimale :

$$r_j^* = \max(r_j, r_i^*) \quad \text{pour chaque } T_i \to T_j$$

$$D_j^* = D_j - (r_j^* - r_j)$$

Tâche	First release	First release optimale	WCET	D	D optimale	P
T_1	0	0	1	10	10	10
T_2	2	2	1	8	8	10
T_3	0	2	2	10	8	10
T_4	0	2	1	10	8	10
T_5	0	2	1	10	8	10
T_6	0	0	8	15	15	20

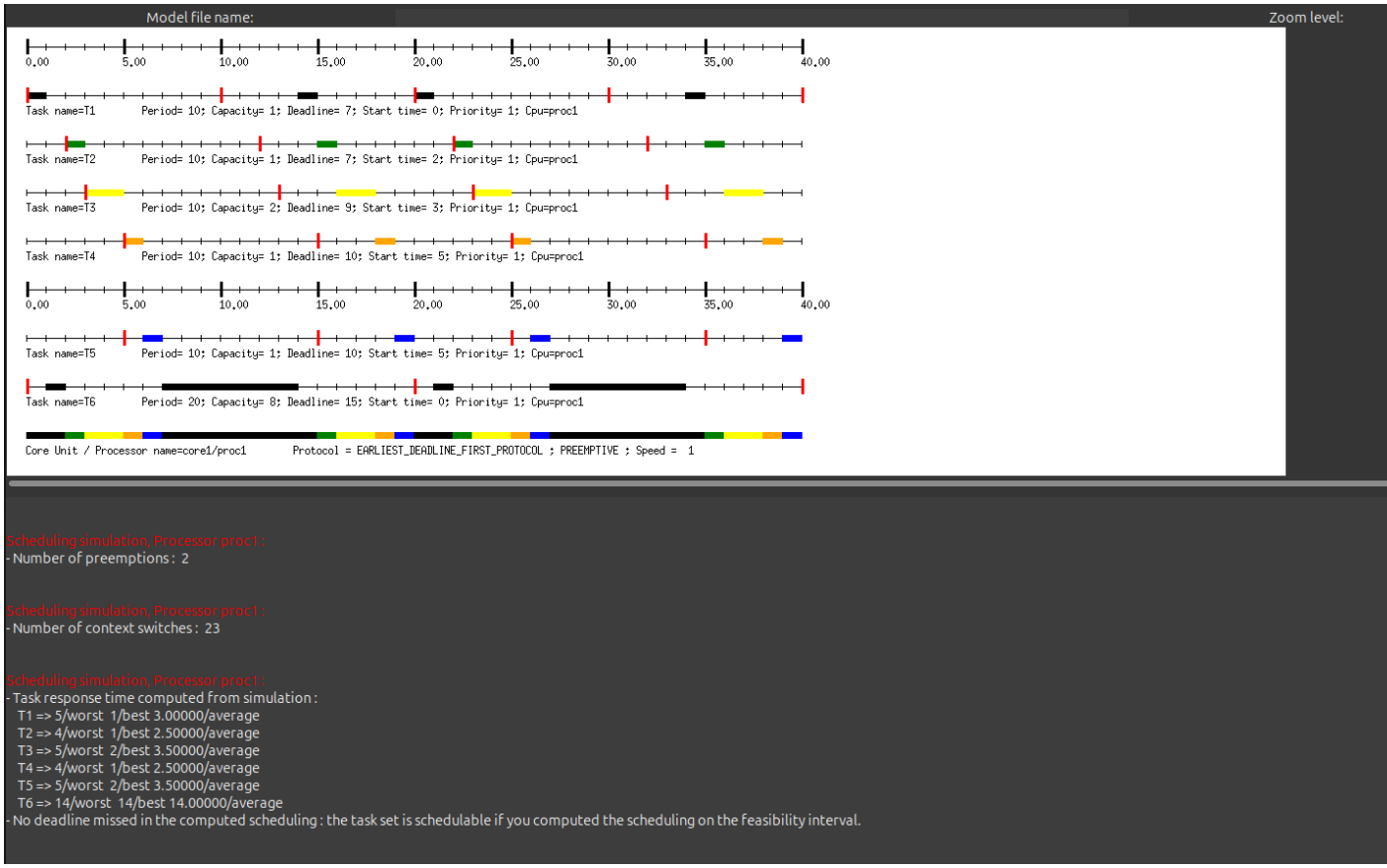
5.2 Algorithme de EDF

Pour l'algorithme EDF avec les contraintes de précédence, nous appliquons les formules suivantes pour déterminer les valeurs optimales du premier relâchement et de la deadline :

$$r_j^* = \max(r_j, \max(r_i^* + C_i)) \quad \text{pour tout } T_i \to T_j$$

$$d_j^* = \min(d_j, \min(d_k^* - C_k)) \quad \text{pour tout } T_j \to T_k$$

Tâche	First release	First release optimale	WCET	D	D optimale	P
T_1	0	0	1	10	7	10
T_2	2	2	1	8	7	10
T_3	0	3	2	10	9	10
T_4	0	5	1	10	10	10
T_5	0	5	1	10	10	10
T_6	0	0	8	15	15	20



Résultats d'ordonnancement :

Le simulateur a enregistré 2 préemptions durant l'exécution, ce qui signifie que deux fois, une tâche en cours d'exécution a été interrompue pour laisser place à une autre tâche avec une date limite plus proche. Cela reflète le comportement attendu dans un algorithme de type EDF (Earliest Deadline First), où les tâches sont priorisées en fonction de leur deadline.

Conclusion de l'ordonnancement :

Aucune deadline n'a été manquée lors de la simulation, ce qui montre que l'ensemble des tâches est ordonnançable. Cela implique que toutes les tâches ont pu être exécutées avant leurs échéances respectives, validant ainsi la faisabilité de l'ordonnancement avec l'algorithme EDF dans l'intervalle de temps observé.