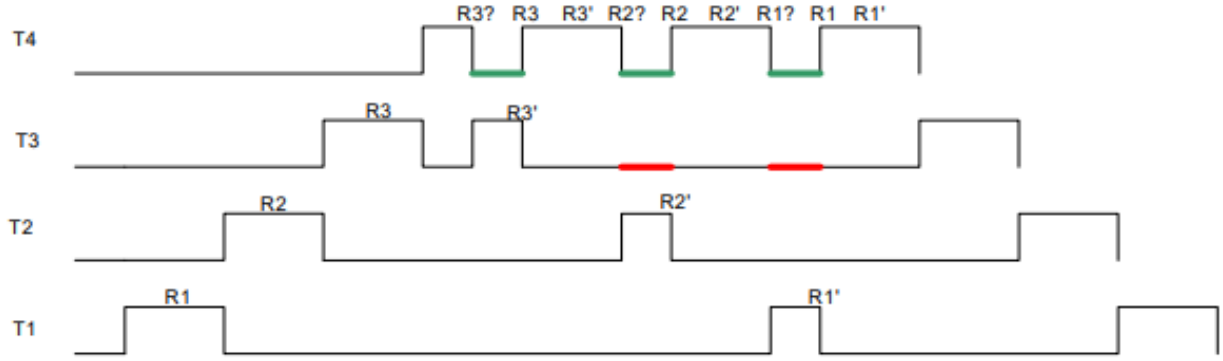
Q1

Soit 4 tâches T1 à T4 en ordre de priorité croissante (T4 étant donc la plus prioritaire), montrez graphiquement un exemple de pire cas de blocage sur T4 en considérant le protocole héritage de priorité. Illustrez et expliquez clairement où se trouve ce pire cas sur votre schéma. Utilisez la convention utilisée en classe: Ri? indique que la tâche en exécution veut accéder à Ri; Ri indique que la tâche en exécution entre dans Ri et Ri‘ indique que la tâche en exécution sort de Ri .

Réponse :



Q2

Un des avantages du protocole ICPP (*Immediate Ceiling Priority Protocol*) par rapport à l'héritage de priorité est d'éviter les interblocages. Soient deux tâches A et B où A est plus prioritaire que B et qui à l'aide des mutex m1 et m2 utilisent les sections critiques S1 et S2 dans la séquence suivante :

Séquence de la tâche A	Séquence de la tâche B
OSMutexPend (m1); … OSMutexPend (m2); … OSMutexPost(m2); … OSMutexPost(m1);	OSMutexPend (m2); … OSMutexPend (m1); … OSMutexPost (m2); … OSMutexPost (m1);

Sachant que la tâche B est la première à démarrer mais que la tâche A démarre avant que B verrouille m1, démontrez comment avec l'utilisation du protocole d'héritage de priorité, cette séquence peut mener à une situation d'interblocage.

En deux lignes, expliquez comment le protocole ICPP évite cet interblocage.

Réponse

- *B démarre et verrouille m2*
- *Avant que m1 soit verrouillé par B, A démarre et par préemption bloque B*
- *A verrouille m1*
- *Puis quand A arrive pour verrouillé m2, il donne sa priorité par héritage afin que ce dernier quitte m2. Cependant B ne peut quitter car il a besoin de verrouiller m1 lui-même déjà verrouillé par A.*

N.B. Dans ICPP, A n'aurait pas verrouillé m1 car sa priorité n'aurait pas été strictement plus grande que le ceiling de m2. Donc B aurait complètement terminé puis A aurait été exécuté. On peut donc dire que ICPP est deadlock free.

Q3

Soit les 5 processus périodiques suivants utilisant un ordonnancement RMA (Rate Monotonic Assignement) combiné au protocole ICPP :

Tâche (T_i)	Priorité (P_i)*	Numéro de la période de départ	Nombre de périodes d'exécution du thread	Séquence d'exécution
T1	1	6	4	EVQE
T2	2	4	3	EQE
T3	3	3	3	EEE
T4	4	y	10	EQQQQQQQQE
T5	5	x	8	EVVVVVVE

* Plus la priorité est élevée, plus le processus est prioritaire

Table 3.1

a) Soit x = 0 et y = 2, complétez sur la figure 3.1 la trace d'exécution de ces 5 tâches et montrez à l'aide de votre résultat, que le temps d'exécution de T5 dans la section critique du mutex V (6 cycles) représente une borne supérieure du temps de blocage maximal de T3.

b) Soit x = 2 et y = 0, complétez sur la figure 3.2 la trace d'exécution de ces 5 tâches et montrez à l'aide de votre résultat, que le temps d'exécution de T4 dans la section critique du mutex Q (8 cycles) représente une bonne approximation du temps de blocage maximal de T3.

c) De manière générale, montrez que le temps de blocage maximal de T3 peut être estimé par le temps d'exécution de T4 dans la section critique du mutex Q, peu importe la valeur de x et y.

Réponse page suivante :

a)

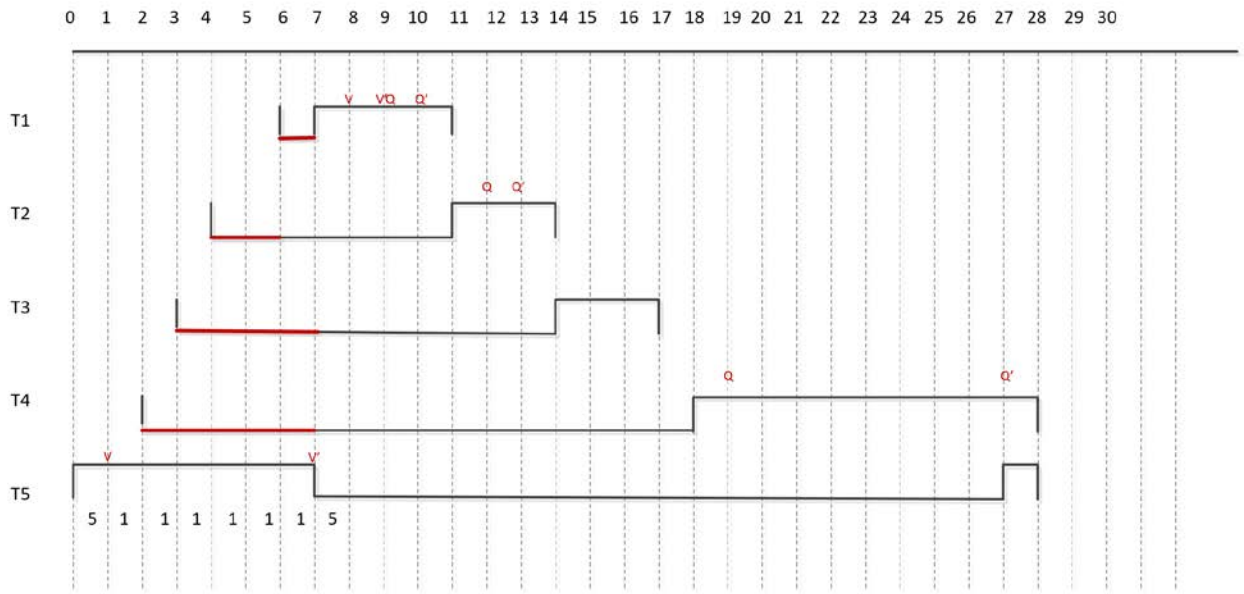


Figure 3.1

4 cycles de blocage pour T3, donc on est inférieur à 6

b)

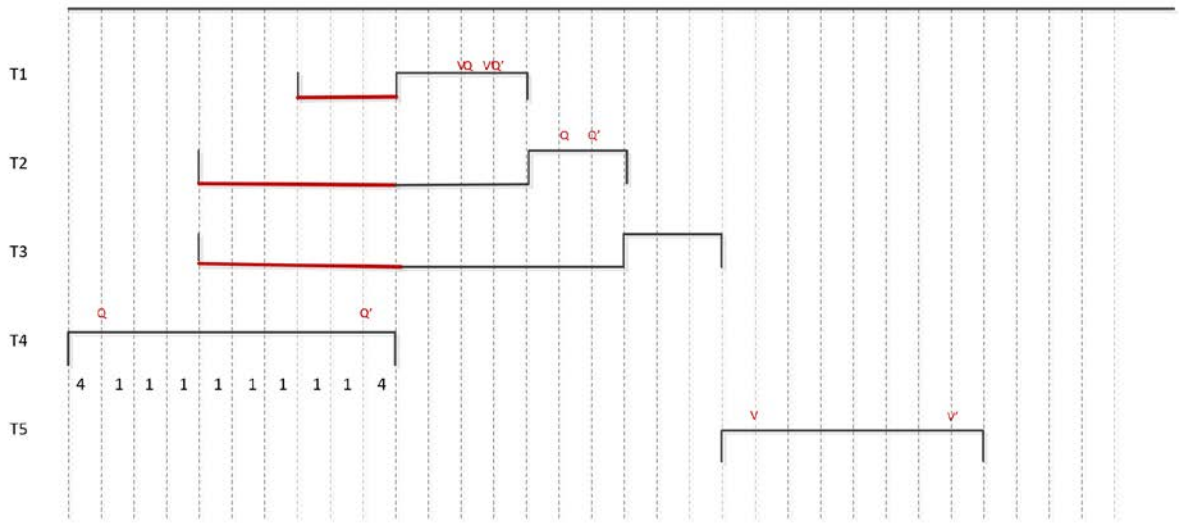


Figure 3.2

6 cycles de blocage pour T3, donc on est inférieur à 8

Selon la formule on prend max (6, 8) = 8 (blocage maximal dans T4). Par conséquent on peut garantir que T3 ne bloquera jamais plus que 8 (il s'agit d'une borne supérieure). L'avantage ici est vous pouvez garantir que T3 ne bloquera jamais plus que 8 sans avoir fait aucune exécution de code…

Q4. Soit l'ensemble de tâches périodiques suivant qui roule sur un processeur P1:

Tâche (T_i)	Temps d'exécution (C_i)	Période (T_i)	Deadline (D_i)
T_1	2	10	10
T_2	1	6	6
T_3	9	20	20

Ensemble de tâches d'un système

a) Existe-t-il pour ces tâches un ordonnancement statique c'est-à-dire avec priorité fixe?

On assigne selon RMA

$T_2 \rightarrow 6$ (+ prioritaire) $T_1 \rightarrow 10$ $T_3 \rightarrow 20$ (- prioritaire)

et on fait d'abord le test de Liu and Layland :

2/10 + 1/6 + 9/20 ≅ .82 < .78 donc on ne peut rien dire

b) En utilisant les tests et/ou les méthodes de calcul pour borner R_i, montrez l'existence ou non d'un tel ordonnancement.

Toujours en assignant selon RMA :

$T_2 \rightarrow 6$ (+ prioritaire) $T_1 \rightarrow 10$ $T_3 \rightarrow 20$ (- prioritaire)

L'équation de R dont on a besoin ici est la suivante :

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

$w_{2,0} = C_2 = 1 < 6 \rightarrow$ ok

$w_{1,0} = C_1 = 2$

$$w_{1,1} = 2 + \left\lceil \frac{2}{6} \right\rceil * (1) = 3$$

$$w_{1,2} = 2 + \left\lceil \frac{3}{6} \right\rceil * (1) = 3 \leq T1 = 10 \rightarrow OK$$

$w_{3,0} = C_3 = 9$

$$w_{3,1} = 9 + \left\lceil \frac{9}{6} \right\rceil * (1) + \left\lceil \frac{9}{10} \right\rceil * (2) = 13$$

$$w_{3,2} = 9 + \left\lceil \frac{13}{6} \right\rceil * (1) + \left\lceil \frac{13}{10} \right\rceil * (2) = 16$$

$$w_{3,3} = 9 + \left\lceil \frac{16}{6} \right\rceil * (1) + \left\lceil \frac{16}{10} \right\rceil * (2) = 16 \prec 20 \rightarrow OK$$

c) Que se passse-t-il si on a le blocage suivant pour une utilisation de mutex:

Tâche (T_i)	Temps d'exécution (C_i)	Période (T_i)	Deadline (D_i)	Blocage (Bi)
T_1	2	10	10	3
T_2	1	6	6	2
T_3	9	20	20	1