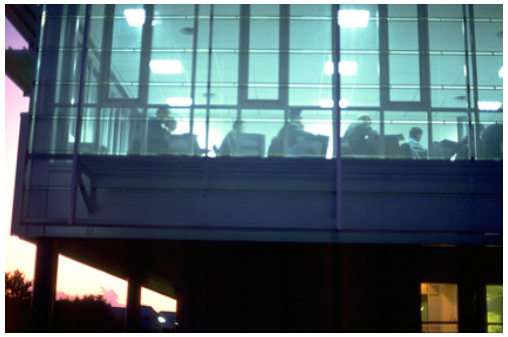


Cours A7 : Temps Réel

Pierre.Paradinas / @ / cnam.fr
Cnam/Cedric
Systèmes Enfouis et Embarqués (SEE)

Motivations

- Du jour :
  - Test de faisabilité d'ordonnancement
- Comprendre les mécanismes systèmes mis en oeuvre dans un système temps réel

Monotone par Taux

- Le “Rate monotonic” a été introduit par Liu & Layland en 1973.
- Rate monotonic :
  - Basé sur les priorités
  - Les tâches sont périodiques :
    - pas de communication,
    - temps de commutation négligeable
  - Période = échéance

Priorité

- Les tâches sont ordonnancées sur la priorité
  - La tâche de priorité maximale a la fréquence la plus haute
  - La tâche de priorité minimale a la fréquence la plus faible
  - Les tâches ne s'arrêtent pas elles mêmes
  - Les tâches sont préemptables
- Si on respecte cette contrainte on dispose de tests à priori sur les tâches pour assurer la faisabilité.

Condition suffisante

Définitions :

– Plafond $\lceil n \rceil$: *entier* qui suit *n*, exemple $\left\lceil \frac{4}{3} \right\rceil = 2$

– Planché $\lfloor n \rfloor$: entier qui précède *n*,exemple $\left\lfloor \frac{4}{3} \right\rfloor = 1$

Taux Utilisation Processeur $= \sum_{1}^{n} \frac{durée_i}{période_i}$

Si on a n tâches, elles respectent l'échéance si :

$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq n(2^{\frac{1}{n}} - 1)$

Pour :

$n = 1$, U = 1 (100%)

$n = 2$, U = 0.78 (78%)

...

$n = 10$, U = 0.69 (69%)

Démonstration

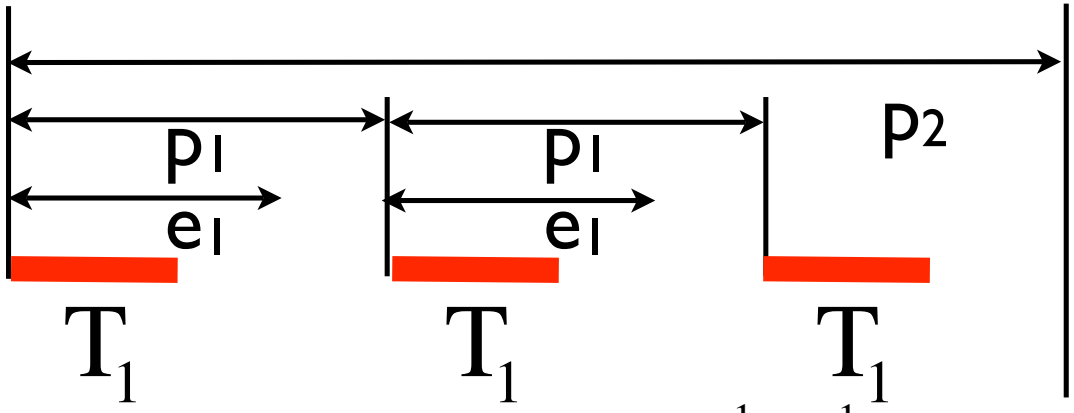
Cas simplifié

Tâches $T_i(e_i, p_i, c_i)$, où les éléments du triplet représentent respectivement l'échéance, la période et la capacité (temps d'exécution au pire) de T_i.

On se donne T1 et T2 : $T_1(e_1, p_1, c_1)$ et $T_2(e_2, p_2, c_2)$.

Objectifs:

* On cherche à minimiser U de manière indépendante de T_i et de c_i.



Les priorités de T_1 et de T_2 sont : $\frac{1}{p_1}$ et $\frac{1}{p_2}$ respectivement.

On a : $p_1 < p_2$ et $\frac{1}{p_1} > \frac{1}{p_2}$

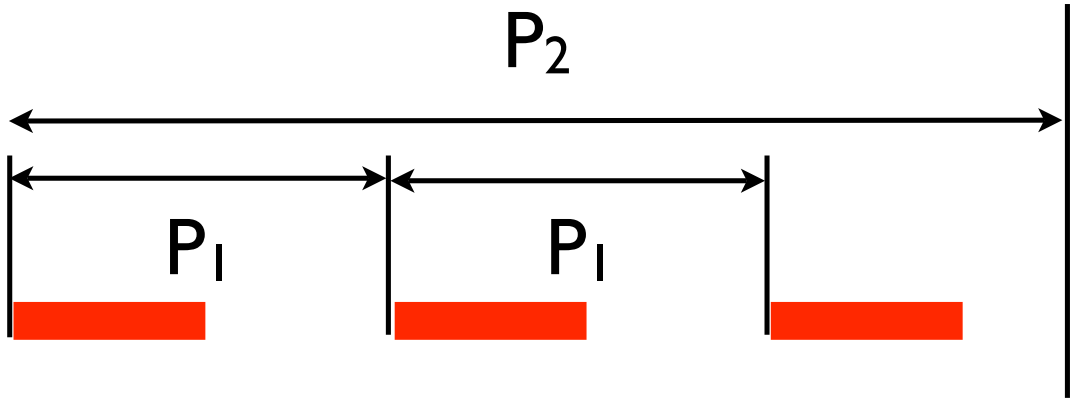
T_1 sera activé $\left\lceil \frac{p_2}{p_1} \right\rceil$ fois dans $[0, p_2]$

Démonstration (suite)

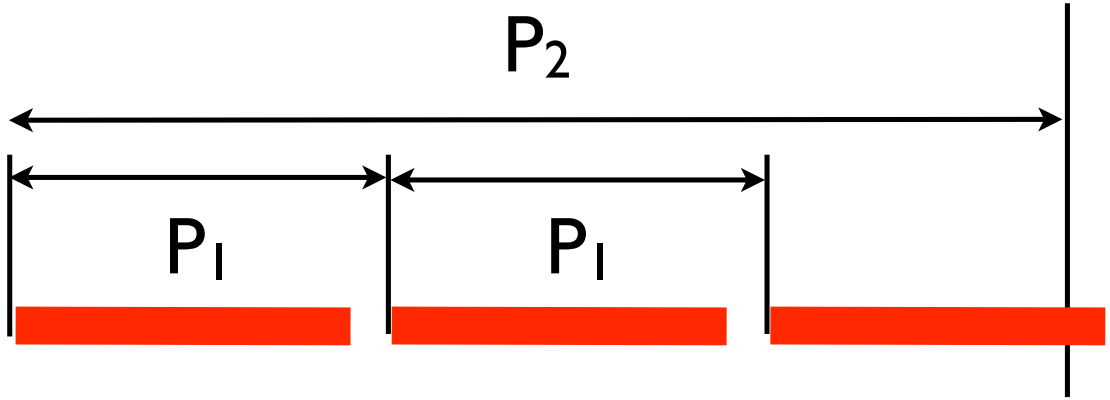
$$U = \frac{c_1}{p_1} + \frac{c_2}{p_2} \Rightarrow U = \frac{c_1}{p_1} + \frac{f(c_1, p_1, p_2)}{p_2}$$

2 cas :

- T_1 termine son exécution avant l'activation suivante de p_2



- T_1 ne termine pas son exécution avant l'activation suivante de p_2



Démonstration (suite)

1er Cas :

$$c_1 + p_1\left\lfloor\frac{p_2}{p_1}\right\rfloor \le p_2 \Rightarrow c_1 \le p_2 - p_1\left\lfloor\frac{p_2}{p_1}\right\rfloor$$

Maximum de c_2 quand $c_2 + c_1\left\lfloor\frac{p_2}{p_1}\right\rfloor = p_2 \Rightarrow c_2 = p_2 - c_1\left\lfloor\frac{p_2}{p_1}\right\rfloor$

$$U = \frac{c_1}{p_1} + \frac{p_2 - c_1\left\lfloor\frac{p_2}{p_1}\right\rfloor}{p_2} = \frac{c_1}{p_1} + 1 - \frac{c_1}{p_2}\left\lfloor\frac{p_2}{p_1}\right\rfloor = 1 + c_1\left(\frac{1}{p_1} - \frac{1}{p_2}\left\lfloor\frac{p_2}{p_1}\right\rfloor\right)$$

comme $\left\lceil\frac{p_2}{p_1}\right\rceil \ge \frac{p_2}{p_1} \Rightarrow \left(\frac{1}{p_1} - \frac{1}{p_2}\left\lceil\frac{p_2}{p_1}\right\rceil\right) \le 0$

Si c_1 croit alors U décroit.

Démonstration (suite)

2eme Cas :

$$c_1 + p_1\left\lfloor\frac{p_2}{p_1}\right\rfloor \geq p_2 \Rightarrow c_1 \geq p_2 - p_1\left\lfloor\frac{p_2}{p_1}\right\rfloor$$

Maximum de c_2 quand $c_2 + c_1\left\lfloor\frac{p_2}{p_1}\right\rfloor = p_1\left\lfloor\frac{p_2}{p_1}\right\rfloor \Rightarrow c_2 = p_1\left\lfloor\frac{p_2}{p_1}\right\rfloor - c_1\left\lfloor\frac{p_2}{p_1}\right\rfloor$

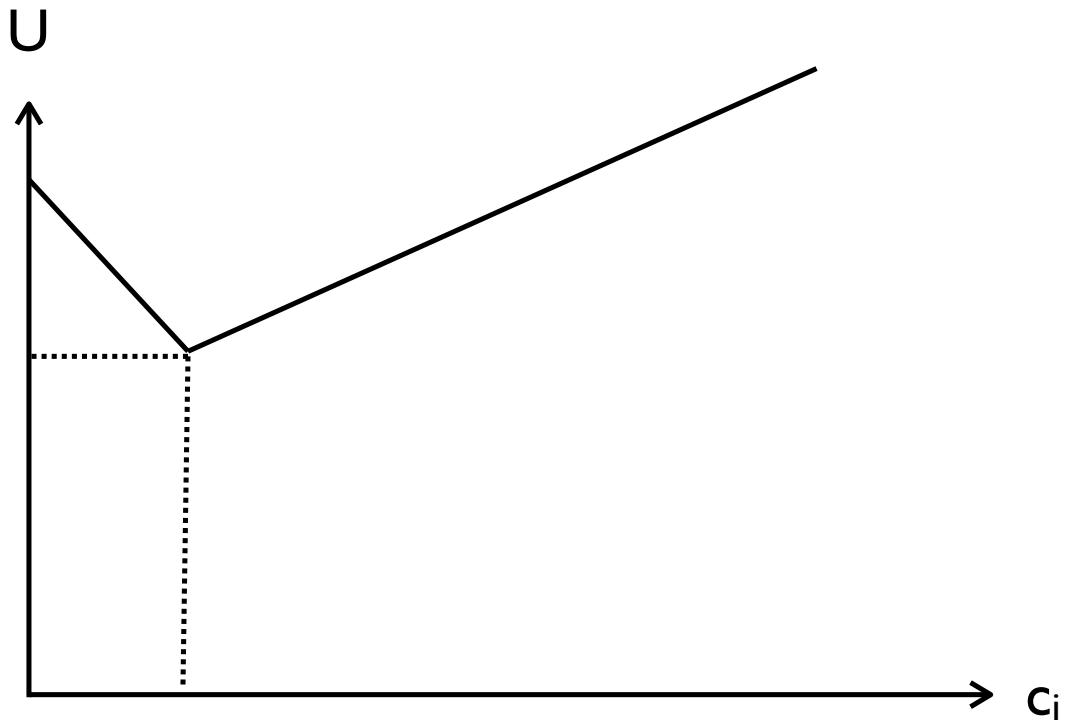
$$U = \frac{c_1}{p_1} + \frac{p_1\left\lfloor\frac{p_2}{p_1}\right\rfloor - c_1\left\lfloor\frac{p_2}{p_1}\right\rfloor}{p_2} = \frac{c_1}{p_1} + \frac{p_1}{p_2}\left\lfloor\frac{p_2}{p_1}\right\rfloor - \frac{c_1}{p_2}\left\lfloor\frac{p_2}{p_1}\right\rfloor =$$

$$U = \frac{p_1}{p_2}\left\lfloor\frac{p_2}{p_1}\right\rfloor + c_1\left(\frac{1}{p_1} - \frac{1}{p_2}\left\lfloor\frac{p_2}{p_1}\right\rfloor\right)$$

qui est positif car $\left(\left\lfloor\frac{p_2}{p_1}\right\rfloor \leq \frac{p_2}{p_1}\right) \Rightarrow \left(\frac{1}{p_1} - \frac{1}{p_2}\left\lfloor\frac{p_2}{p_1}\right\rfloor\right) \geq 0$

Si c_1 croit alors U croit.

Démonstration (suite)



Démonstration (suite)

U est minimal avec $c_1 = p_2 - p_1 \left\lfloor \frac{p_2}{p_1} \right\rfloor$

$$U = 1 + (p_2 - p_1 \left\lfloor \frac{p_2}{p_1} \right\rfloor)(\frac{1}{p_1} - \frac{1}{p_2} \left\lceil \frac{p_2}{p_1} \right\rceil) = 1 + (\frac{p_2}{p_1} - \left\lceil \frac{p_2}{p_1} \right\rceil - \left\lfloor \frac{p_2}{p_1} \right\rfloor + \frac{p_1}{p_2} \left\lfloor \frac{p_2}{p_1} \right\rfloor \left\lceil \frac{p_2}{p_1} \right\rceil)$$

$$= 1 - \frac{p_1}{p_2}(\left\lceil \frac{p_2}{p_1} \right\rceil - \frac{p_2}{p_1})(\frac{p_2}{p_1} - \left\lfloor \frac{p_2}{p_1} \right\rfloor)$$

Soit $f = \frac{p_2}{p_1} - \left\lfloor \frac{p_2}{p_1} \right\rfloor$ (% partie fractionnaire de $\frac{p_2}{p_1}$)

$$I = \left\lfloor \frac{p_2}{p_1} \right\rfloor (\%\ p_2 \geq p_1\ I_{\text{minimum}} = 1)$$

$$U = 1 - \frac{p_2}{p_1}((\left\lfloor \frac{p_2}{p_1} \right\rfloor + 1 - \frac{p_2}{p_1})(\frac{p_2}{p_1} - \left\lfloor \frac{p_2}{p_1} \right\rfloor) = 1 - \frac{p_2}{p_1}(1-f)f = 1 - \frac{(1-f)f}{I+f}$$

$$U_{\text{minimum}} = 1 - \frac{(1-f)f}{1+f}$$

Démonstration

$$U_{\text{minimum}} = 1 - \frac{(1-f)f}{1+f}$$

$$U'(f) = \frac{2f(1+f) - (1+f^2)}{(1+f)^2} \text{ nulle si f } = \sqrt{2} - 1$$

$$U(\sqrt{2}-1) = 2(\sqrt{2}-1)$$

Remarque

- U = 0.69 (=Ln(2))
- La condition est stricte.
- Dans la réalité on peut relâcher et utiliser une contrainte à 85 -90%.
  - Avec des tâches choisies aléatoirement cela fonctionne encore.

Exemple

- Calculer :
  - U_1, U_2, U_3, U_4

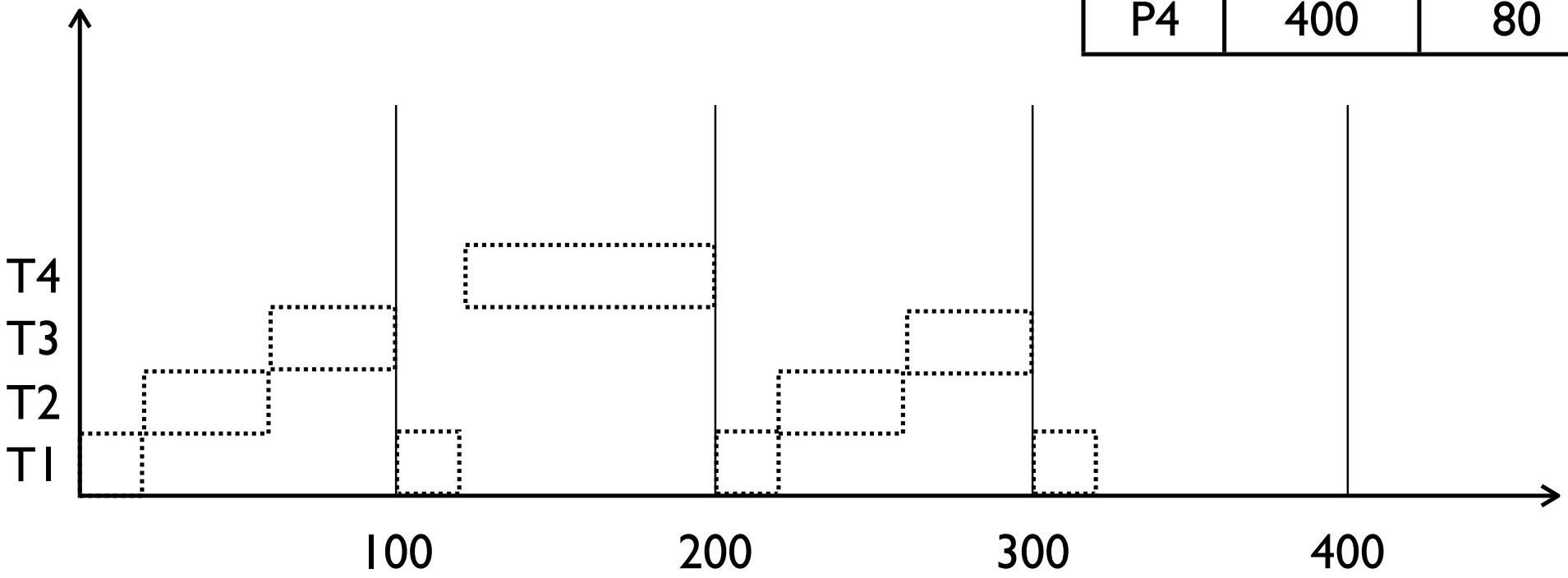
	p	t
T1	100	20
T2	200	40
T3	200	40
T4	400	80

U_1	U_2	U_3	U_4
0.2	0.2	0.2	0.2

  - $\sum U_i = 0.8 < 4*(2^{1/4}-1)=$
  - Représenter les tâches sur un diagramme temps

Diagramme des tâches

	p	t
P1	100	20
P2	200	40
P3	200	40
P4	400	80



Relâcher... avec le théorème de la zone critique

- Hypothèse
  - Si toutes les tâches arrivent initialement simultanément et si elles respectent leurs premières échéances alors elles respecteront les autres qque soient l'ordre d'arrivée des tâches.
  - Si les tâches ne sont pas toutes prêtes au départ la condition est suffisante.

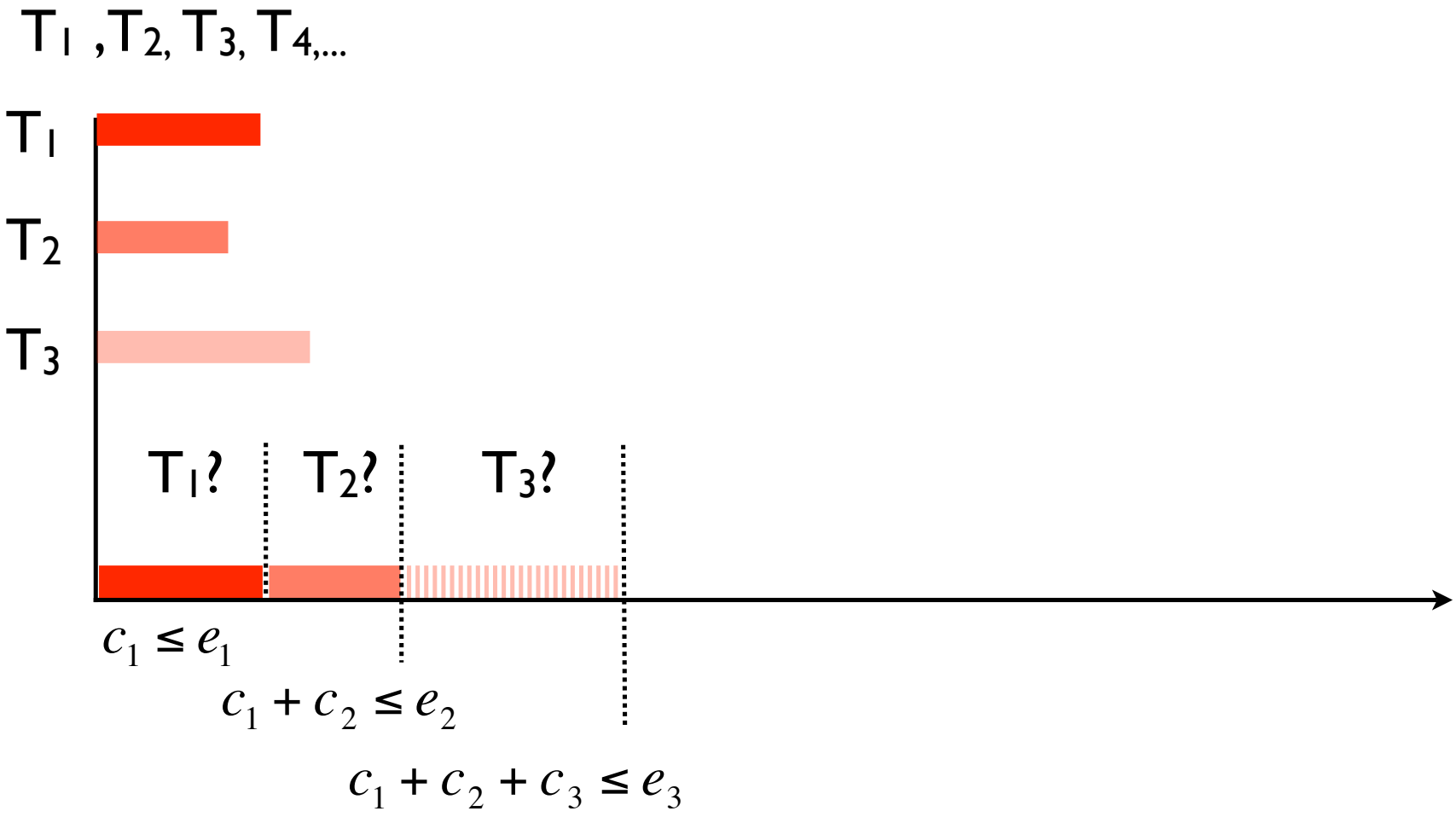
Le test de terminaison

$$\forall i, 1 \le i \le n \quad \underset{0 \le t \le e_i}{Min} \sum_{j=1}^{i} \frac{c_i}{t} \left\lceil \frac{t}{p_i} \right\rceil \le 1$$

- Toutes les tâches sont prêtes => les tâches de fortes priorités sont activées au début, si malgré cela les tâches de plus faibles priorités "passent" avant leurs échéances respectives, ça pourra pas être pire!

L'idée...

- Tâches : $T_1, T_2, T_3, \ldots, T_n$, T_i plus forte priorité que T_{i+1}, etc...



Méthode d'obtention : itération

On cherche t tq

$\forall i \ 1 \le i \le n, \ \exists t \le e_i, \ t = \sum_{j=1}^{i} c_i \left\lceil \frac{t}{p_i} \right\rceil$

Soit $W_i(t) = \sum_{j=1}^{i} c_i \left\lceil \frac{t}{p_i} \right\rceil$, $\left\lceil \frac{t}{p_i} \right\rceil$ nombre de fois que T_i est activée dans $[0,t]$

et $c_i \left\lceil \frac{t}{p_i} \right\rceil$ le temps consommé par dans l'intervalle $[0,t]$

$W_i(t)$ représente la demande cumul de temps des tâches 1 à i dans $[0,t]$

$W_i(t) = \sum_{j=1}^{i-1} c_i \left\lceil \frac{t}{p_i} \right\rceil + c_i \Rightarrow$ est le retard imposé par les tâches plus prioritaire que T_i

On cherche t, tq $W_i(t) = t$

L'algotrithme d'itération :

- $t_0 = \sum_{j=1}^{i-1} c_j$ chaque tâche s'exécutant une fois. Si l'instant t est tq $W_i(t) \neq t$

alors on itère avec la valeur t obtenue. (Voir exemple).

Exemple 1

- $\sum Ui = 0.953 < 3(2^{1/3}-1) = 0.779$ ne répond pas au théorème de la zone critique.

	p	t	U
P1	100	40	0.4
P2	150	40	0.267
P3	350	100	0.287

i	Wi
1	40
2	80
3	180

W3	Wi(t) = t
180	2t1+2t2+t3 = 260
260	3t1+2t2+t3 = 300
300	3t1+2t2+t3 = 300

- Wi = demande cumulée de temps CPU de toutes les tâches i dans l'intervalle [0,t].
- On cherche Wi(t)=t.

Diagramme temps de l'exemple

Exemple 2

- $\sum Ui = 0.8 < 4(2^{1/4}-1)$ ne répond pas au théorème de la zone critique.

	p	t
P1	100	20
P2	200	40
P3	200	40
P4	400	80

i	Wi
1	20
2	60
3	100
4	180

W4	Wi(t) = t
180	$2t_1+2t_2+2t_3+t_4= 280$
280	$3t_1+2t_2+2t_3+t_4= 300$
300	$4t_1+2t_2+2t_3+t_4= 320$
320	$4t_1+2t_2+2t_3+t_4= 320$

Monotone par taux et échéance

- Monotone par taux mais priorité fonction de l'échéance :
  - échéance proche => forte priorité,
  - ordonnancement reste statique
- Condition et test d'ordonnancement :
  - d <= p : même que monotone par taux
  - d > p (une tâche finit son exécution en dehors de sa période sans dépasser son échéance et "passer aussi dans la période suivante