Ordonnancement temps réel

-Tâches Périodiques-

Maria Zrikem

GI3 - GS3 ENSA de Marrakech

2024 - 2025

Ordonnancement

C’est lors de décisions d’ordonnancement que l’on choisit la tâche qui devient la tâche courante. Dans les systèmes TR, on espère collecter suffisamment d’informations à priori pour pouvoir prédire le comportement de l’application et garantir son fonctionnement par la suite.

Ordonnaçabilité d’un ensemble de tâches

Lorsqu’il existe au moins un algorithme faisant en sorte que toutes les tâches respectent leur contraintes.

Ordonnancement optimal

Un ordonnanceur A est optimal si $\forall \Gamma = \{P_1, ..., P_n\}$ un ensemble de tâches faisables par un autre ordonnanceur B, alors l’ordonnanceur A peut également produire un ordonnancement valide pour Γ.

Complexité du problème d’ordonnancement : NP-complet en général

Ordonnancement

On distingue :

- ✓**Non-préemptifs :** n'interviennent que si la tâche courant se finit ou demande à être mise en attente..
- ✓ **Préemptifs :** sont capables de retirer l'UC à une tâche pour la donner à une autre.

On distingue également les ordonnancements :

- ✓**off-line :** la séquence d'ordonnancement est déterminée à l'avance,
- ✓**on-line :** la séquence d'ordonnancement est décidée durant l'exécution du système.

On parle enfin d'ordonnancements :

- ✓ **statiques :** les paramètres des tâches guidant l'ordonnancement ne peuvent pas évoluer,
- ✓ **dynamiques :** les paramètres des tâches guidant l'ordonnancement peuvent évoluer pendant l'exécution du système. Une même tâche peut donc être ordonnancée différemment à deux instants différents.

Ordonnancement temps réel

Types d'ordonnancement :

✓ Statique : ordonnancement fixe

✓ Statique préemptif basé sur les priorités : cas de l'analyse *rate monotonic.*

✓ Dynamique basé sur une planification à l'exécution : on a suffisamment d'informations pour prendre une décision intelligente (EDF).

✓ Dynamique basé sur la notion du meilleur effort : le système fait de son mieux pour répartir les ressources.

On s'intéresse aux techniques d'ordonnancement pour lesquelles il existe des tests permettant de savoir si un ensemble de tâches (configuration) est ordonnaçable.

Types de tâches

✓ Périodique : la tâche est activée à des intervalles réguliers. Elle doit se terminer avant son prochain démarrage. On présence d'autres tâches plus prioritaires, l'instant de démarrage peut être un peu retarder.

✓ Sporadique : on ne sait pas quand elle se déclenche, mais il y a un temps minimum entre deux déclenchements.

✓ Apériodique : on ne peut rien dire.

Attributs généraux :

- Arrivée — Ai (ou Oi Offset)
- Temps de calcul maximum sans préemption — Ci
- Echéance (deadline) : relative ou absolue — Di
- Date de début (start time) — Si
- Date de fin (finish time) — Fi
- Retard (lateness) — Li = (Fi - Di)
- Laxité (slack time) — xi(t) = Di - (t + Ci - ci(t))

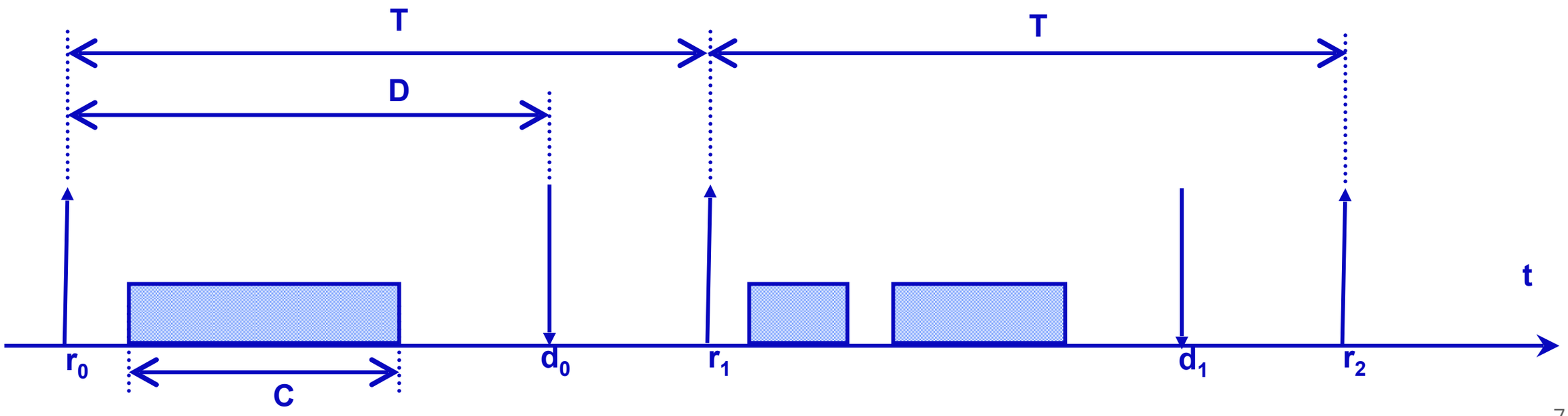
Xi = xi(Ai) = (Di - Ai - Ci)

Attributs usuels

✓ T : Période de la tâche. La tâche devrait démarrer aux temps 0, T, 2T, 3T,

✓ D : Échéance (deadline). Temps limite pour la fin de la tâche ; à la période n, la tâche doit s'exécuter entre le temps nT et nT + D.

✓ C : Capacité. Le temps d'exécution ; la capacité est forcement inférieure à l'échéance, sinon même la tâche seule elle ne serait ordonnançable.



Attributs usuels

✓ T : Période de la tâche. La tâche devrait démarrer aux temps 0, T, 2T, 3T,

✓ D : Échéance (deadline). Temps limite pour la fin de la tâche ; à la période n, la tâche doit s'exécuter entre le temps nT et nT + D.

✓ C : Capacité. Le temps d'exécution ; la capacité est forcement inférieure à l'échéance, sinon même la tâche seule elle ne serait ordonnançable.

✓ Facteur d'utilisation du processeur pour *n* tâches périodiques : $\sum_{i=1}^{i=n} \frac{C_i}{P_i}$

✓ Facteur de charge du processeur pour *n* tâches périodiques : $\sum_{i=1}^{n} \frac{C_i}{D_i}$

Hyperpériode (période d'étude)

Durée = [*Début*, *Fin*]

Début = Min(*Ai*) si *Ai* est la date de première activation de toute tâche *Ti*

Fin = Max(*Ai*) + PPCM(*Pi*) si toutes les tâches sont périodiques

OU

$$Fin = Max((Ai)_{périodiques}, (Ai+Di)_{apériodiques}) + 2 * PPCM(Pi)_{périodiques}$$

Théorème J. Y. T. Leung, M. L. Merrill, 1980 :

Si la séquence est valide sur l'intervalle [Début, Fin], alors elle est valide sur un temps infini.

Arrivées des tâches

- **Périodiques** : arrivée à intervalles réguliers (Pi)
  - Date d'activation initiale, offset Ai
  - Si pour tout i, j Ai=Aj, tâches synchrones
  - Si Di = Pi, tâche à échéance sur requête
- **Sporadiques** : on connaît une borne minimale sur l'intervalle entre deux arrivées
- **Apériodiques** : tout ce qui ne rentre pas dans les deux catégories précédentes

Ordonnancement temps réel : RM

Ordonnancement Rate Monotonic (RM) (monotone par taux)

Méthode d'affectation (hors ligne) de priorités statiques à un ensemble de tâches. On suppose :

✓ travailler avec un ordonnanceur préemptif,

✓ Les tâches sont périodiques, indépendantes et leurs échéances sont égales aux périodes

✓ si T_i désigne la période de la tâche i alors sa priorité est égale à la fréquence $1/T_i$

✓ l'ordonnancement s'effectue avec la politique HPF (sélection de la tâche de plus forte priorité)

Ordonnancement temps réel : Rate Monotonic (RM)

Faisabilité d'ordonnancement

L'ensemble des tâches dont on connaît la capacité Ci et la période Ti sont ordonnançables si le taux d'occupation du processeur ne dépasse pas une certaine limite dépendant du nombre n de processus.

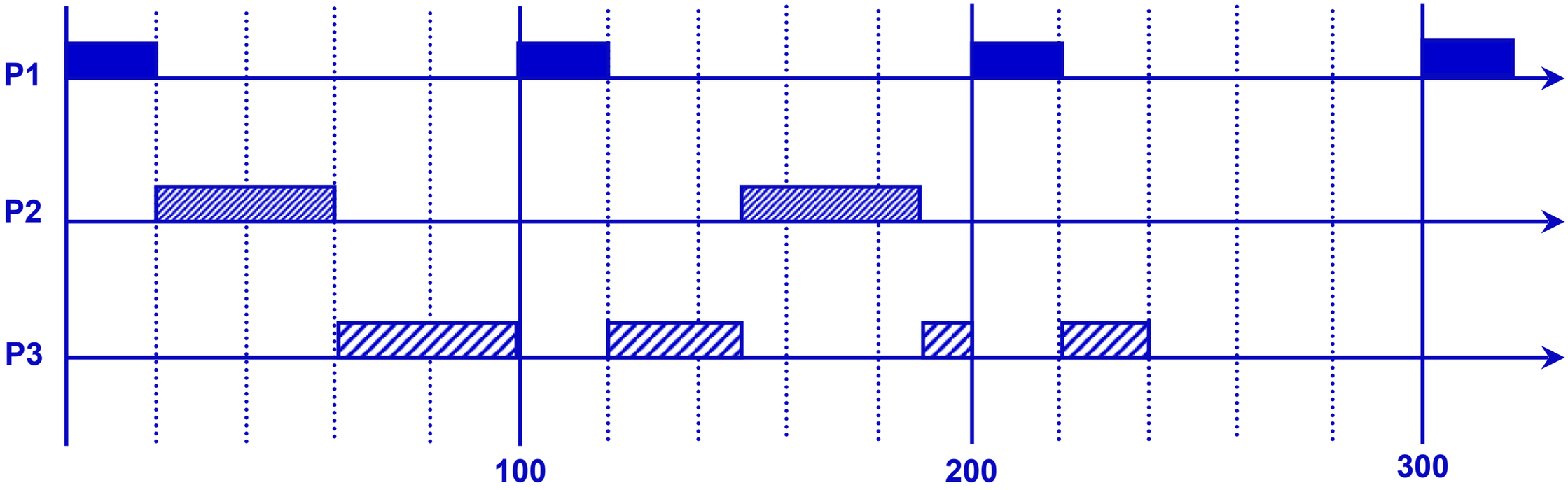
$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$$

$\Rightarrow$ Cette condition est suffisante mais pas nécessaire

Ordonnancement temps réel : Rate Monotonic (RM)

Exemple :

3 tâches P1, P2, P3.
T1=100, T2=150, T3=350 C1=20, C2=40, C3=100
Ui=Ci/Ti U1=0.2, U2=0.267, U3=0.287
$\sum Ui=0.753 < 3*(2^{1/3}-1)=0.779$



Faisabilité de l'Ordonnancement RM

Théorème de la zone critique

Si toutes les tâches arrivent initialement dans le système simultanément et si elles respectent leur première échéance, alors toutes les échéances seront respectées par la suite

⇒ C'est une condition nécessaire et suffisante si toutes les tâches sont initialement prêtes au même instant.

⇒ C'est uniquement une condition suffisante dans le cas contraire

⇒ Il ne peut y avoir de temps libre tant que la tâche la moins prioritaire n'a pas fini son premier travail

Théorème de la zone critique : mise en oeuvre

La fonction suivante calcule pour un temps t (< première échéance de la tâche la moins prioritaire) le temps auquel se terminera peut être la tâche i. C'est le temps mis par toutes les tâches les plus prioritaires pour terminer leur travail plus la capacité de la tâche i.

$$W_i(t) = \sum_{j=0}^{i-1} C_j \left\lceil \frac{t}{T_j} \right\rceil + C_i$$

⇒ La tâche i respecte son échéance s'il existe $t < D_i$ tel que $W_i(t) = t$

Faisabilité de l'Ordonnancement RM

Théorème de la zone critique : mise en oeuvre

- On commence par ordonner les tâches dans l'ordre décroissant de leurs priorités : P_1 est la tâche de priorité la plus forte, P_2 la tâche de priorité juste inférieure à P_1, etc. jusqu'à P_n la tâche de priorité la plus faible.

- Soit $\sum_{j=1}^{i} C_j \left\lceil \frac{t}{T_j} \right\rceil$

- $\left\lceil \frac{t}{T_j} \right\rceil$ représente le nombre de fois que la tâche Pj est activée dans l'intervalle [0, t]
- $C_j \left\lceil \frac{t}{T_j} \right\rceil$ représente la demande en consommation de la tâche Pj dans l'intervalle [0, t]
- W_i (t) représente la demande cumulée en UC de toutes les tâches de priorités plus fortes que celle de P_i dans [0,t].

Faisabilité de l'Ordonnancement RM

Théorème de la zone critique : mise en oeuvre

$$W_i(t) = \sum_{j=0}^{i-1} C_j \left\lceil \frac{t}{T_j} \right\rceil + C_i$$

- Autrement dit : $\sum_{j=1}^{i} C_j \left\lceil \frac{t}{T_j} \right\rceil$ représente la consommation supplémentaire (le retard) imposée à P_i par les tâches de priorités supérieures.
- Si il existe t tel que $W_i(t) = t$, alors le temps processeur a pu être alloué à P_i au bout de t.
- Pour étudier l'ordonnançabilité, il suffit donc de trouver pour toute tâche P_i un t tel que $W_i(t) = t$ et que évidemment $t \leq D_i$.

Faisabilité de l'Ordonnancement RM

Exemple d'utilisation du théorème de la zone critique :

3 tâches P1, P2, P3. T1=100, T2=150, T3=350 C1=40, C2=40, C3=100

Ui=Ci/Ti U1=0.4, U2=0.267, U3=0.287

$\sum Ui=0.953$ <? $3*(2^{1/3}-1) = 0.779$

⇒ Mais répond au théorème de la zone critique

- Pour i =1 (tâche P1) :
  - $W_1(0) = C_1 = 40 < 100 = D_1$
  - $W_1(40) = C_1 = 40 < 100 = D_1$ OK

- Pour i =2 (tâche P2) :
  - $W_2(40) = C_1 * \lceil 40/ T_1=100 \rceil + C_2 = C_1 + C_2 = 40 + 40 = 80 < 150 = D_2$
  - $W_2(80) = C_1 * \lceil 80/ T_1=100 \rceil + C_2 = C_1 + C_2 = 40 + 40 = 80 < 150 = D_2$

  OK

Faisabilité de l'Ordonnancement RM

Exemple d'utilisation du théorème de la zone critique :

3 tâches P1, P2, P3. T1=100, T2=150, T3=350 C1=40, C2=40, C3=100

Ui=Ci/Ti U1=0.4, U2=0.267, U3=0.287

$\sum Ui=0.953$ <? $3*(2^{1/3}-1) = 0.779$

⇒ Mais répond au théorème de la zone critique

♦Pour i =3 (tâche P3) :

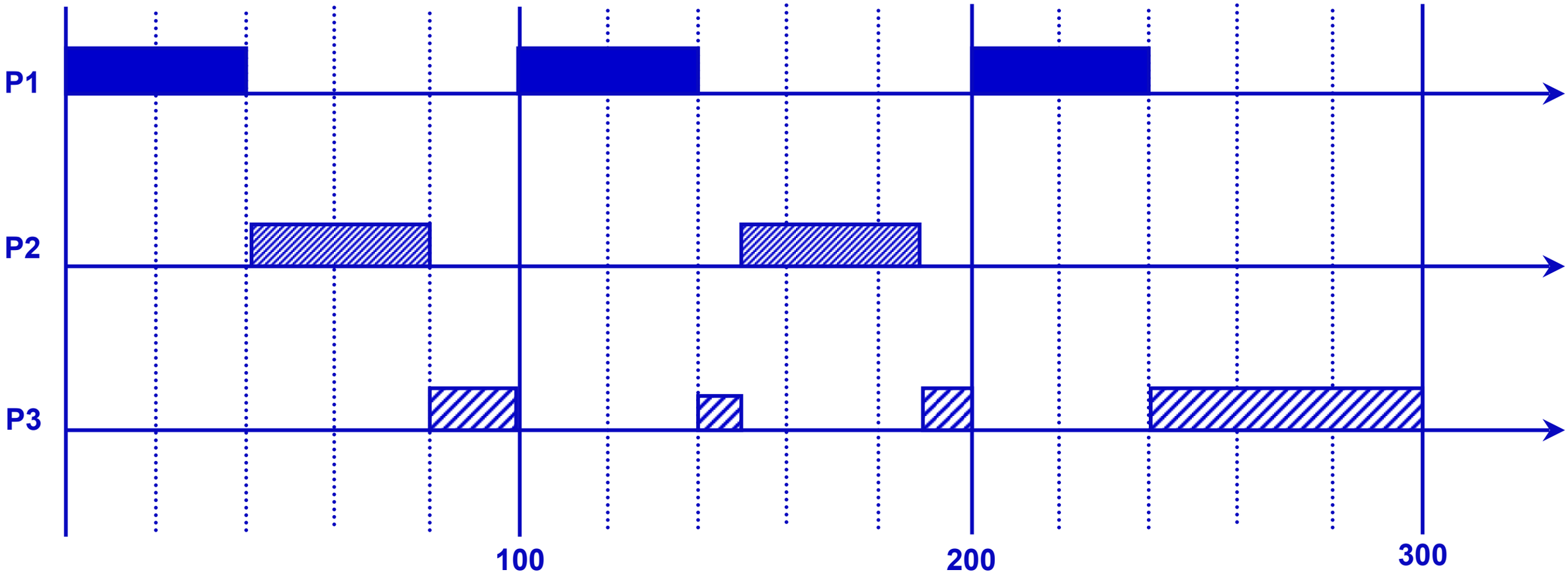
- $W_3(80) = C_1 *\lceil 80/ T_1=100\rceil + C_2 *\lceil 80/ T_2=150\rceil + C_3 = C_1 + C_2 + C_3 = 180 < 350 = D_3$
- $W_3(180) = C_1 *\lceil 180/ T_1=100\rceil + C_2 *\lceil 180/ T_2=150\rceil + C_3 = 2C_1 + 2C_2 + C_3 = 260 < 350 = D_3$
- $W_3(260) = C_1 *\lceil 260/ T_1=100\rceil + C_2 *\lceil 260/ T_2=150\rceil + C_3 = 3C_1 + 2C_2 + C_3 = 300 < 350 = D_3$
- $W_3(300) = C_1 *\lceil 300/ T_1=100\rceil + C_2 *\lceil 300/ T_2=150\rceil + C_3 = 3C_1 + 2C_2 + C_3 = 300 < 350 = D_3$

OK

Faisabilité de l'Ordonnancement RM

Exemple d'utilisation du théorème de la zone critique :

3 tâches P1, P2, P3. T1=100, T2=150, T3=350 C1=40, C2=40, C3=100



Faisabilité de l'Ordonnancement RM

Théorème de la zone critique : cas ou l'échéance est inférieure à la période

⇒ On applique l'analyse RM en utilisant des priorités basées sur les périodes Ti

⇒ il faut rajouter un facteur de blocage Ei = Ti - Di

⇒ le test s'écrit : existe-t-il un t tel que :

$$t = W_i(t) = \sum_{j=1}^{i} C_j \left\lceil \frac{t}{T_j} \right\rceil + E_i$$

Avec la condition que le temps t trouvé soit inférieur à Ti

Analyse Deadline Monotonic (DM)

C'est une dérivée de l'analyse RM avec une affectation de priorités légèrement différente : en prenant les mêmes critères pour les tâches, mais dans l'hypothèse ou D < T, on attribue les priorités aux tâches dans l'ordre inverse de leurs échéances.

✓ affectation toujours statique des priorités en utilisant les échéances. On favorise les tâches de plus court deadline.

✓ la formule de test d'ordonnancement ne change pas (en remplace T par D dans la formule RM)

✓ on peut également utiliser le théorème de la zone critique

Ordonnancements dynamiques :

Les trois principaux ordonnanceurs donnent la main à la tâche :

✓ qui a la plus grande priorité HPF : *Highest Priority First*.

✓ qui est la plus proche de son échéance EDF : *Earliest Deadline First*, il a besoin de connaître l'échéance.

✓ qui a la moins de marge de fonctionnement LLF : *Least Laxity First*, il a besoin de connaître l'échéance et la capacité.

Ces algorithmes sont dits dynamiques car l'affectation de priorités aux tâches se fait lorsque le système est en fonctionnement (en ligne). Selon l'approche de son échéance ou de la marge de temps restant, on augmente dynamiquement la priorité des tâches.

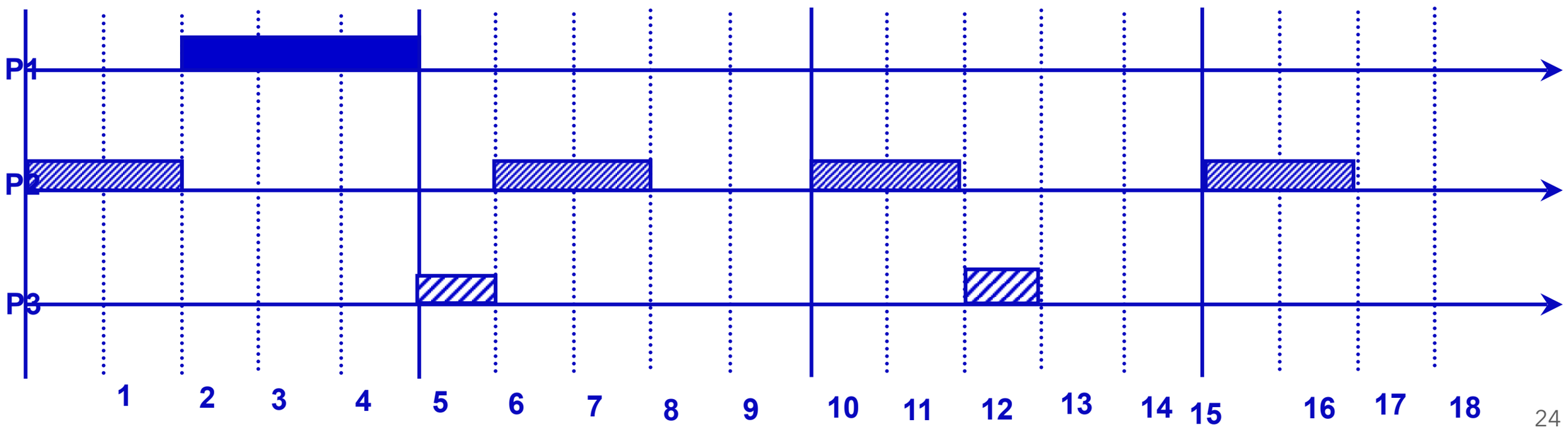
Ordonnancements dynamiques :

EDF : Earliest Deadline First

- Algorithme on-line dynamique
- Principe : ordonnancement par priorité inversement proportionnelle à

$D_i(t) = D_i - (t - r_i)$, le temps restant avant la prochaine échéance. En cas d'égalité, EDF laisse la tâche courante s'exécuter.

tâche	Capacité	Échéance	Période
P1	3	7	20
P2	2	4	5
P3	1	8	10



Ordonnancements dynamiques :

EDF : Earliest Deadline First

- Il produit moins de changement de contexte
- Il est optimal (si un ordonnancement peut ordonnancer une configuration alors EDF peut aussi)
- En cas de surcharge, il est possible de découper chaque tâche en une partie obligatoire et une partie optionnelle qu'on peut laisser de côté

LLF : Least Laxity First

L'ordonnancement LLF attribue à tout instant la plus haute priorité à la tâche ayant la plus faible laxité dynamique.

La laxité dynamique Li(t) représente le temps maximum pendant lequel l'exécution de la Tâche Ti peut être retardée sans que celle-ci manque son échéance.

$$Li(t) = Di - (t + Ci(t))$$

où : di est l'échéance de Ti,

t l'instant courant,

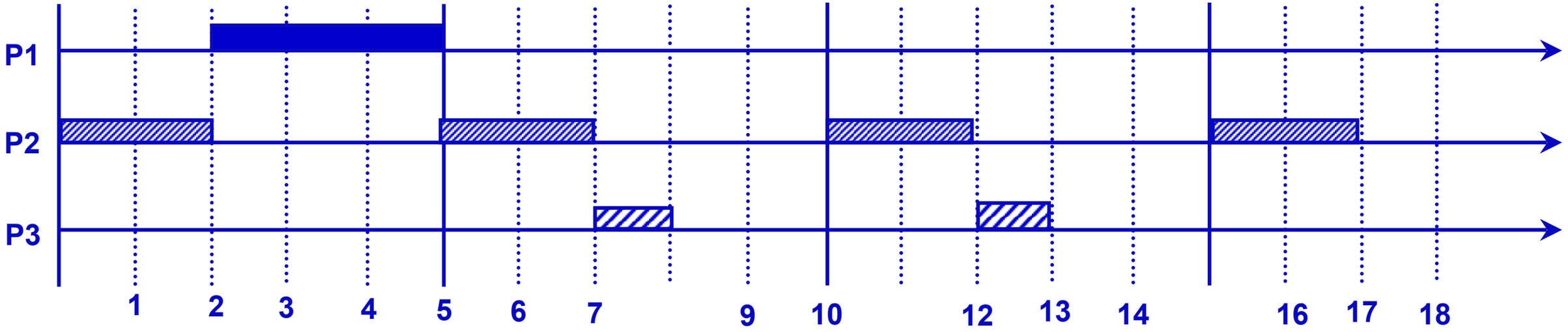
Ci(t) le temps d'exécution restant pour Ti à l'instant t.

C'est donc un algorithme à priorités dynamiques car la priorité de chaque tâche non exécutée augmente avec le temps.

LLF : Least Laxity First

On exécute en priorité la tâche à laquelle reste le moins de marge possible entre la fin de son calcul et son échéance. Comme EDF, en cas d'égalité on laisse la tâche courante s'exécuter.

tâche	Capacité	Échéance	Période
P1	3	7	20
P2	2	4	5
P3	1	8	10



Remarque : à l'instant t=5, on peut élire P2 ou P3 (laxité dynamique = 2)

- Il prend en compte la durée du travail
- Il détecte les dépassements d'échéance avant qu'ils ne se produisent.

Critère d'ordonnancabilité EDF et LLF

Si l'on applique les algorithmes EDF et LLF à des tâches indépendantes périodiques comme c'est le cas dans RM, on a un critère simple de faisabilité.

- Si $d_i = t_i$

  Condition nécessaire et suffisante : Un en semble de n taches est ordonnançable par EDF ou LLF ssi le taux d utilisation du processeur est inferieur a 100 %.

$$U = \sum_{i=1}^{n} \frac{c_i}{t_i} \leq 1$$

  Difficile de faire mieux sans déborder.

- Si $d_i < t_i$

  Condition suffisante mais pas nécessaire

  Analyse du temps plus difficile qu'en RM.

$$U = \sum_{i=1}^{n} \frac{c_i}{d_i} \leq 1$$

EDF VS LLF

Contrairement à un algorithme à priorités fixes, EDF, LLF nécessite de mettre à jour continuellement les priorités des tâches.

Pour EDF cette mise à jour n'est faite qu'au réveil du job. Puisqu'on calcule la laxité à chaque réveil ou terminaison de job, LLF entraine plus de préemptions et donc plus de changements de contexte que EDF. On peut en déduire que EDF est plus efficace et plus facilement implémentable que LLF.

EDF et LLF VS RM

Ceci ne condamne en rien l'analyse RM car EDF et LLF peuvent être instables en cas de surcharge du processeur (i. e. on ne peut pas assurer que toutes les tâches respectent toutes leurs échéances), Si dans tous les cas de figures on peut garantir qu'il y aura pas de surcharge, alors EDF et LLF sont meilleurs que RM

Ordonnancement cyclique

⇒ Ordonnancement de tâches périodiques et ordonnanceur non préemptif

⇒ Cycle majeur : l'intervalle de temps sur lequel l'ordonnancement est précalculée. Il se répète périodiquement. Sa durée est le PPCM des périodes des tâches.

⇒ le cycle majeur est décomposé en cycles mineurs : une interruption est reçue au début de chaque cycle mineur. On peut prendre la plus petite période comme cycle mineur.

⇒ il n y a pas d'algorithme efficace pour trouver l'ordonnancement quand il y a beaucoup de tâches.

Ordonnancement cyclique exemple

le cycle majeur : 100

le cycle mineur : 25

tâche	période	échéance	capacité
A	25	25	10
B	25	25	8
C	50	50	5
D	50	50	4
E	100	100	2