Conditions de faisabilité pour l'ordonnancement temps réel préemptif et non préemptif

Laurent George
Ecole Centrale d'Electronique (ECE)
lgeorge@ece.fr

ETR'05 - Nancy

Conditions de faisabilité pour l'ordonnancement temps réel préemptif et non préemptif

1. Introduction
2. Caractérisation d'un problème d'ordonnancement
   - Modèles de tâches
   - Modèles de contraintes temporelles
   - Modèles d'ordonnancement
3. Algorithmes d'ordonnancement / optimalité
   - Priorités fixes, plus haute priorité en premier (FP)
   - Priorités dynamique (DP)
   - FP/DP
4. Périodes actives et scenarii pires cas
5. Conditions de faisabilité

1. Introduction

On considère un Système Temps Réel (STR) monoprocesseur que l'on souhaite dimensionner.

Approche pire cas:

- Pas une approche « en moyenne »
- Pas basée sur des simulations ou des tests d'un système réel
- Objectifs: identifier les pires scénarii de fonctionnement d'un STR
- Vérifier que sur ces scénarii, le STR reste conforme à ses spécifications

Un STR conforme en pire cas est conforme pour tout scénario d'activation possible

Deux stratégies pour le dimensionnement temps réel:

- L'approche synchrone: un scénario particulier d'activation est imposé et reproduit à l'identique pendant toute la vie du système
- L'approche asynchrone: on ne fait pas d'hypothèse particulière sur les instants où peuvent se produire les évènements

1. Introduction

- **Le Déterminisme**:
  - Pouvoir garantir que le STR respectera ses spécifications pendant toute sa durée de vie.

- **Méthode de résolution d'un STR:**
  - Approche méthodologique:
    - ✓ Identifier à partir des spécifications la classe de problèmes à résoudre
    - ✓ Déterminer les scénarii pire cas pour ce problème
    - ✓ Exprimer les conditions de faisabilité (CF) associées
    - ✓ Donner les valeurs numériques des paramètres des CF
    - ✓ Vérifier que les CF sont validées

2. Caractérisation d'un problème d'ordonnancement

Un STR monoprocesseur est défini par:

- Un modèle de tâche
- Un modèle de contraintes temporelles
- Un modèle d'ordonnancement

2.1 Modèles de tâches

- Tâches concrètes: on impose un scénario d'activation particulier des tâches.
- Tâches non concrètes: on ne connaît pas à priori les instants d'activation (ou de première activation) des tâches.

2. Caractérisation d'un problème d'ordonnancement

2.1 Modèles de tâches (suite)

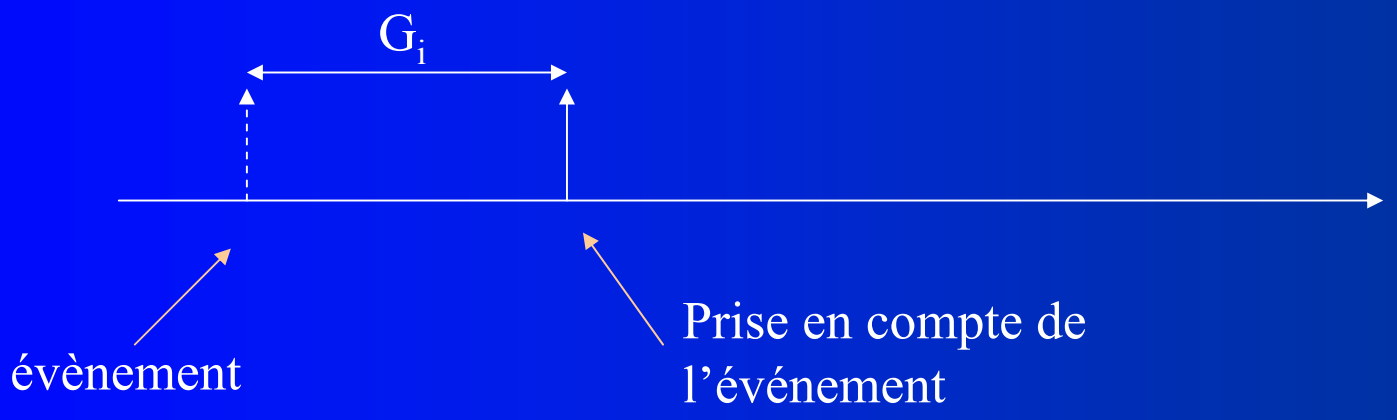
- Durée pire cas d'une tâche τ_i (Worst Case Execution Time) : C_i
  - Durée d'exécution pire cas de la tâche seule sans aucune interruption de l'OS
  - Nécessite une analyse précise du code machine de la tâche et de l'architecture matérielle sur laquelle s'exécute la tâche ou une mesure par benchmark
- Pire temps de réponse d'une tâche τ_i : r_i
  - Temps maximum qui s'écoule entre une demande d'activation d'une tâche et la fin d'exécution de cette tâche. Prise en compte du retard introduit par les autres tâches dans l'exécution de τ_i
  - $$r_i \geq C_i$$

2. Caractérisation d'un problème d'ordonnancement

2.1 Modèles de tâches (suite)

La gigue d'une tâche:

- La gigue est un phénomène qui introduit un délai entre un évènement et la prise en compte de cet évènement par le système
- On parle de *gigue d'activation* lorsque cet évènement correspond à une demande d'activation d'une tâche
- La gigue d'une tâche τ_i est notée: G_i



2. Caractérisation d'un problème d'ordonnancement

2.1 Modèles de tâches (suite)

- **Lois d'arrivée d'une tâche τ_i :**

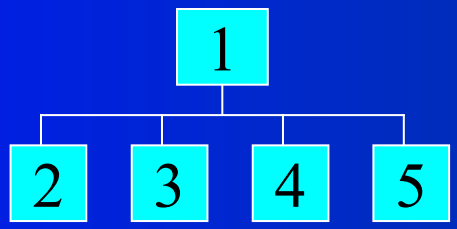
  - **Périodique**: les demandes d'activation de la tâche sont périodiques, de période T_i
  - **Sporadique**: les demandes d'activation de la tâche ont une inter-arrivée $\geq T_i$
  - **Apériodique**: 1 seule demande d'activation de la tâche pendant la durée du système.
  - **Fenêtrée**: Au plus n_i arrivées sur une fenêtre glissante W_i.

  Les interruptions peuvent être traitées à l'aide du modèle fenêtré si l'on dispose d'un mécanisme de masquage d'IT.

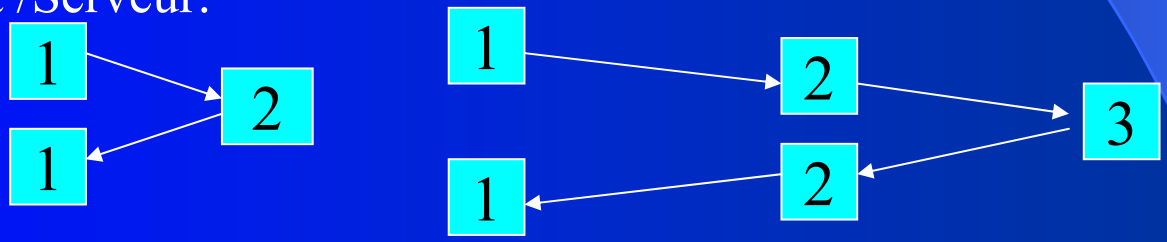
2. Caractérisation d'un problème d'ordonnancement

2.1 Modèles de tâches (suite)

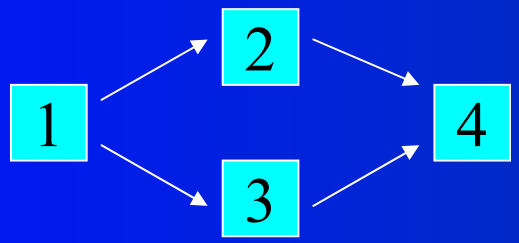
- **Structures des tâches:**
  - La séquence: tâche élémentaire constituée d'un code séquentiel
  - L'arbre:



  - Client /Serveur:



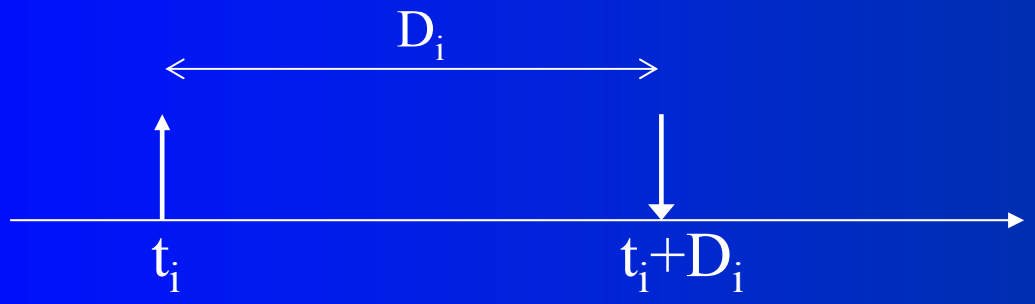
  - Le graphe:



2. Caractérisation d'un problème d'ordonnancement

2.2 Modèles de contraintes temporelles

- L'échéance de terminaison au plus tard
  - Toute tâche τ_i activée à l'instant t_i doit impérativement être terminée à l'instant $t + D_i$
  - D_i est l'échéance relative de la tâche
  - $t_i + D_i$ est l'échéance absolue de la tâche



2. Caractérisation d'un problème d'ordonnancement

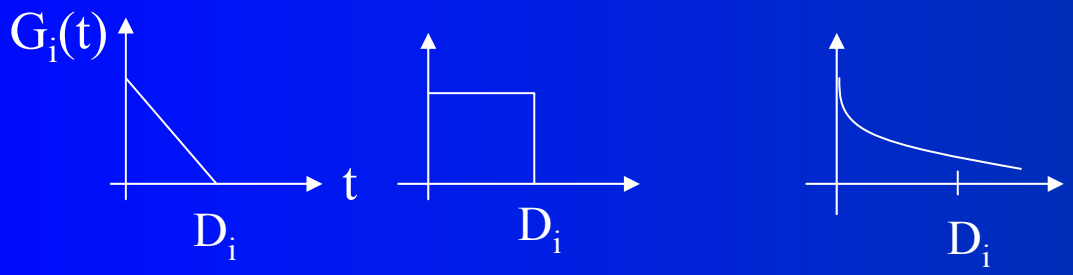
2.2 Modèles de contraintes temporelles (suite)

- L'échéance de démarrage au plus tard
  - Toute tâche τ_i activée à l'instant t_i doit impérativement être démarrée à l'instant $t_i + D_i$
  - D_i est l'échéance relative de démarrage de la tâche
  - $t_i + D_i$ est l'échéance absolue de démarrage de la tâche

2. Caractérisation d'un problème d'ordonnancement

2.2 Modèles de contraintes temporelles (suite)

- La fonction de valeur (Time Value Function: TVF)
  - A toute tâche τ_i est associée une fonction de gain $G_i(t)$ obtenu en cas d'exécution de la tâche à l'instant t.
  - L'objectif est de maximiser le gain total pour l'ensemble des tâches.
  - ***Exemples de fonctions de gain:***



- Nous étudions dans la suite l'échéance de terminaison au plus tard.

2. Caractérisation d'un problème d'ordonnancement

2.2 Modèles de contraintes temporelles (suite)

Comment garantir l'échéance d'une tâche ?

- En établissant des conditions de faisabilité qui si elles sont vérifiées garantissent que toutes les tâches respectent leur échéance.
- Approche pire de temps de réponse
  - Une CNS de faisabilité en découle:
    - ∀i=1..n, $r_i \leq D_i$
    - Que la charge soumise au processeur $U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$
- Approche basée sur un ensemble un ensemble d'inéquations (demande processeur h(t), charge U, …)
- Approche basée sur une approximation des conditions de faisabilité classiques pour obtenir une condition suffisante polynomiale

2. Caractérisation d'un problème d'ordonnancement

Pour une tâche de type séquence sans gigue, nous avons donc:

- La tâche périodique τ_i (C_i, T_i, D_i)
- La tâche sporadique τ_i (C_i, T_i, D_i)
- La tâche apériodique τ_i (C_i, D_i)
- La tâche fenêtrée τ_i (n_i, W_i, D_i)

Dans la suite, on ne considérera que des tâches sporadiques

2. Caractérisation d'un problème d'ordonnancement

2.3 Le modèle d'ordonnancement

- **Event Driven**: L'ordonnanceur est invoqué sur réception d'un évènement.
- **Time Driven**: L'ordonnanceur détermine ses instants d'invocation (en général: invocation périodique de période T_{tick}
  - La valeur de la période a plusieurs impacts:
    - Un retard sur la prise en compte d'une tâche se traduit par une gigue d'activation
    - Un coût supplémentaire lié à l'exécution de l'Ordonnanceur: Overhead
    - L'Overhead limite la valeur minimale de T_{tick}

Nous considérons dans la suite le modèle Event Driven

2. Caractérisation d'un problème d'ordonnancement

2.3 Le modèle d'ordonnancement (suite)

- Ordonnancement oisif ou non oisif:
  - Un ordonnanceur est oisif si il n'exécute pas toujours les tâches en attente, autrement il est non-oisif .
- Ordonnanceur oisif:
  - Complexité générale: NP-Difficile
  - Principe du régulateur (Shaping)
  - On distingue deux types de régulateurs:
    - Régulateurs de débit (pour éviter les surcharges)
    - Régulateurs de gigue (pour réduire la gigue des tâches)

Nous considérons dans la suite un ordonnancement non oisif qui fournit des conditions de faisabilité polynomiales ou pseudo polynomiale, à l'exception du contexte non préemptif concret où le problème est NP-Complet.

2. Caractérisation d'un problème d'ordonnancement

2.3 Le modèle d'ordonnancement (suite)

Ordonnancement par priorités:

- Fixe (Fixed Priority: FP): la priorité P_i d'une tâche est identique pour toutes ses activations. Ex: RM, DM
- Dynamique (Dynamic Priority): la priorité d'une tâche est définie pour chaque activation de la tâche. Ex EDF, LLF, (FIFO)
- Mixte: FP/DP. Ex: FP/FIFO, FP/EDF. On définit alors une priorité généralisée PGi(t_i) pour une tâche τ_i activitée en t_i: FP pour des tâches de priorité différent de τ_i et DP sinon.

Propriétés 1: (vérifiée par FP, EDF, FIFO, FP/FIFO, FP/EDF):

- La priorité d'une tâche τ_i activée en t_i est constante pour tout instant $t \geq t_i$,
- la priorité de τ_i ne décroît pas si t_i décroıt.

2. Caractérisation d'un problème d'ordonnancement

2.3 Le modèle d'ordonnancement (suite)

Optimalité d'un algorithme d'ordonnancement:

- Un algorithme d'ordonnancement est optimal si il trouve toujours un solution pour respecter les contraintes temporelles des tâches lorsqu'il en existe une.

Contraposée de cette définition:

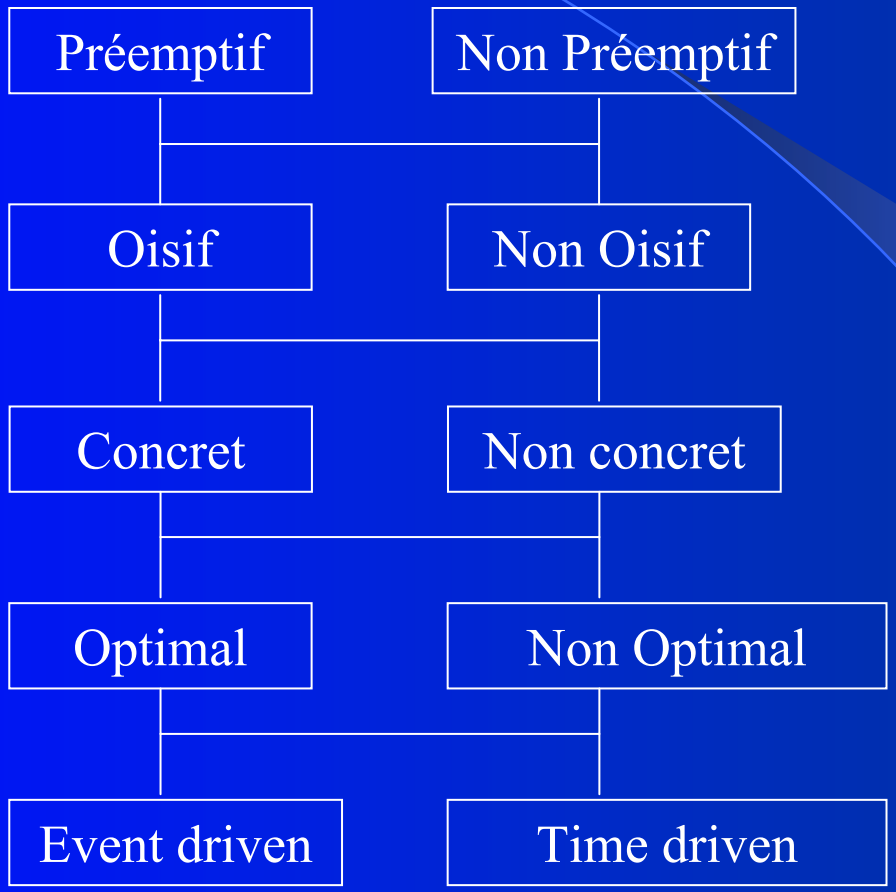
- Si un algorithme optimal ne trouve pas de solution pour respecter les contraintes temporelles des tâches alors il n'existe pas de solution au problème d'ordonnancement

L'optimalité d'une politique FP n'est pas générale:

- On peut trouver une solution faisable avec un algorithme d'ordonnancement DP alors qu'il n'existe pas de solution avec FP.

2. Caractérisation d'un problème d'ordonnancement

- **Décomposition d'une problème d'ordonnancement**



3. Algorithmes d'ordonnancement / optimalité

3.1 Algorithmes d'ordonnancement FP

Rate Monotonic (RM)

- Priorité fonction de la période(T_i)
- Plus la période est petite, plus la tâche est prioritaire
- Résultat d'optimalité:
  - ✓ Pour des tâches sporadiques ou périodiques concrètes ou non concrètes
  - ✓ Avec un ordonnancement préemptif
  - ✓ Pour toutes les tâches τ_i , $i=1\ldots n$, $T_i = D_i$

  => RM est optimal dans ce contexte

RM n'est plus optimal si $\forall i=1\ldots n$, $D_i \leq T_i$
RM n'est plus optimal en contexte non préemptif.

3. Algorithmes d'ordonnancement / optimalité

3.1 Algorithmes d'ordonnancement FP (suite)

Deadline Monotonic (DM)

- Priorité fonction de l'échéance relative(D_i)
- Plus l'échéance relative est petite, plus la tâche est prioritaire.
- Résultat d'optimalité:
  - Pour des tâches sporadiques ou périodiques concrètes ou non concrètes
  - Avec un ordonnancement préemptif
  - Pour toutes les tâches τ_i , i=1…n, $D_i \leq P_i$

    => DM est optimal dans ce contexte

- DM n'est plus optimal en non-préemptif en général
- DM est optimal en non-préemptif si $\forall(i,j)\ C_i \leq C_j \Rightarrow D_i \leq D_j$

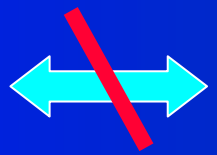
3. Algorithmes d'ordonnancement / optimalité

3.1 Algorithmes d'ordonnancement FP (suite)

Fixed Priority, highest priority first (FP)

- Attribution arbitraire des priorités
  - hp_i : Ensemble des tâches plus prioritaires que τ_i sauf τ_i
  - sp_i : Ensemble des tâches de même priorité que τ_i sauf τ_i
  - $\overline{hp}_i$: Ensemble des tâches moins prioritaires que τ_i
- Il existe une procédure d'attribution optimale des priorités (Audsley) valable en préemptif et en non préemptif
- Un ordonnancement FP non préemptif n'est pas plus dur que l'ordonnancement FP préemptif.

Non ordonnançable en préemptif



Non ordonnançable en non préemptif

3. Algorithmes d'ordonnancement / optimalité

3.1 Algorithmes d'ordonnancement FP (suite)

Procédure d'attribution optimale des priorités (Audsley):

- Basée sur le calcul du pire temps de réponse
- On attribue les priorités de la tâche la moins prioritaire vers la tâche la plus prioritaire

- Si pour un niveau de priorité donné on trouve une tâche qui respecte son échéance, on lui attribue la priorité. Puis on itère jusqu'à la tâche la plus prioritaire

- Si pour une niveau de priorité, aucune tâche sans priorité ne respecte son échéance

  $\Rightarrow$ ordonnancement non faisable

3. Algorithmes d'ordonnancement / optimalité

3.2 Algorithmes d'ordonnancement dynamiques (DP)

First In First Out: FIFO

- Les tâches sont traitées dans l'ordre de leur demande d'activation.
- Minimise le temps de réponse de l'ensemble des tâches.
- Toutes les tâches ont le même temps de réponse maximum.
- Pas optimal pour des tâches qui ont des échéances différentes. Toutes les tâches ont la même importance pour FIFO.

3. Algorithmes d'ordonnancement / optimalité

3.2 Algorithmes d'ordonnancement dynamiques (suite)

Round Robin (RR)

- Le processeur est attribué à tour de rôle aux tâches actives.
- Équitable.
- Pas optimal pour des tâches temps réel. Mais on peut montrer que RR peut trouver un ordonnancement faisable alors que le problème n'est pas faisable avec FP.

3. Algorithmes d'ordonnancement / optimalité

3.2 Algorithmes d'ordonnancement dynamiques (suite)

Variante: Weighted Fair Queuing (WFQ) en contexte réseau

- Utilisé pour le partage du débit nominal réseau B.
- Chaque tache dispose d'un poids Φ_i.
- Le débit B_i de la tâche τi est:

$$B_i = \frac{\varphi_i}{\sum_{j \in \text{ tâches actives}} \varphi_j} \bullet B$$

- Pas adapté pour des tâches de faible échéance et de faible débit
- Le temps de réponse est inversement proportionnel à B_i.

3. Algorithmes d'ordonnancement / optimalité

3.2 Algorithmes d'ordonnancement dynamiques (suite)

Earliest Deadline First (EDF)

- Si t_i est l'instant de demande d'activation d'une tâche τ_i, la priorité est: $\mathbf{t_i + D_i}$
- Algorithme optimal en préemptif concret et non-concret.
- Algorithme optimal en contexte non-préemptif non-concret mais pas en contexte concret
- Algorithme très performant, il permet d'atteindre 100% sous certaines conditions.
- Instable en cas de surcharge (effet avalanche)
  - Technique: basculer vers un autre ordonnancement en cas de surcharge (D_Over), éliminer des tâches ou dégrader le service (m,k)-firm.

3. Algorithmes d'ordonnancement / optimalité

3.2 Algorithmes d'ordonnancement dynamiques (suite)

Shortest Slack Time (SST), ou Least Laxity Time (LLF)

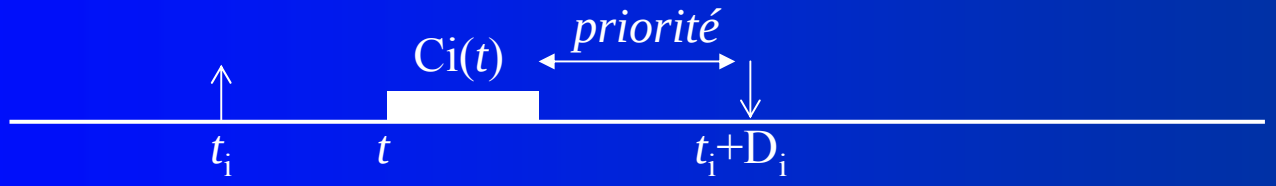
- La priorité à un instant *t* est:

$$t_i + D_i - (t + C_i(t))$$

Avec:

t_i: instant de demande d'activation de τ_i

$C_i(t)$: temps d'exécution restant pour τ_i



- Algorithme optimal en préemptif concret et non concret
- Algorithme non optimal en non préemptif
- Problème: grand nombre de changements de contexte en préemptif.

3. Algorithmes d'ordonnancement / optimalité

3.3 Algorithmes d'ordonnancement FP/DP

- FP/FIFO : Une implémentation classique de FP avec un ordonnancement FIFO sur un niveau de priorité fixe.
- FP/RR : RR pour les tâches de même priorité fixe
- FP/FIFO et FP/RR sont proposés par Posix 1003.1.b.
- FP/LLF : Dénommé également MUF (Maximum Urgency First)
- FP/EDF. FP/EDF domine FP/FIFO sous certaines hypothèses.

On définit pour un ordonnanceur DP ou FP/DP qui respecte la propriété 1, les ensembles:

- $hp_i(t_i)$: ensemble des tâches τ_j sauf τ_j qui ont au moins une activation plus prioritaire que τ_j activée en t_i dans une période active
- $\overline{hp}_i(t_i)$: ensemble des tâches qui n'ont pas d'activation plus prioritaire que τ_j cette une période active

4. Périodes actives et scenarii pires cas

4.1 Pour l'ordonnancement FP

Instant oisif de niveau P_i:

- Un instant oisif de niveau de priorité *Pi* est un instant *t* tel qu'il n'y a plus de tâches dans $hp_i \cup sp_i$ activées avant *t* et non terminées à l'instant *t*.

Période active de niveau P_i:

- Une période active de niveau de priorité P_i est un intervalle de temps [*a*, *b*[tel que *a* et *b* sont deux instants oisifs de niveau de priorité P_i et qu'il n'y a pas d'instant oisif de niveau de priorité P_i dans]*a*, *b*[.

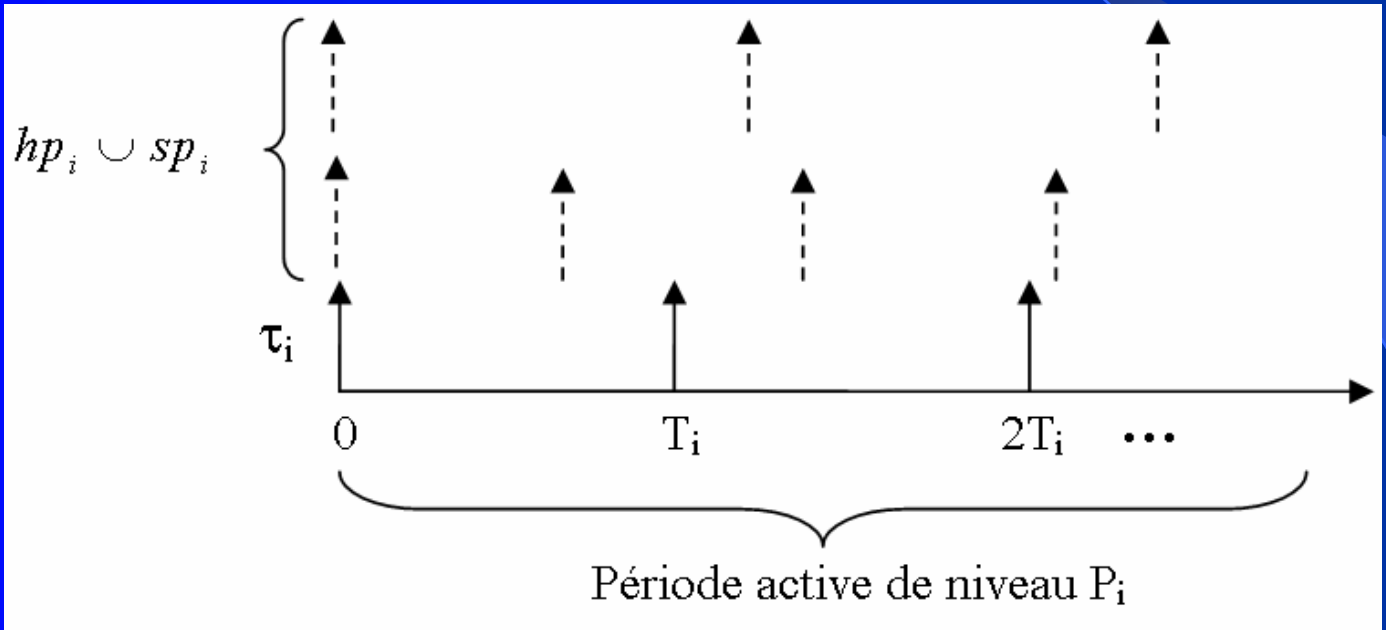
- ✓ Le pire temps de réponse d'une tâche τ_i est obtenu dans une période active de niveau P_i.
- ✓ Mais pour quel scénario d'activation et combien d'activations de τ_i considérer ?

4. Périodes actives et scenarii pires cas

4.1 Pour l'ordonnancement FP (suite)

Scénario pire cas : FP préemptif

- Le scénario synchrone



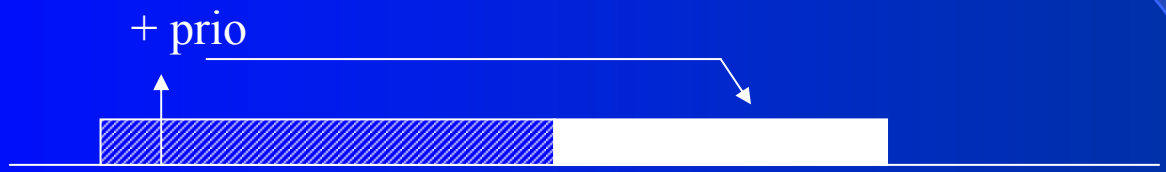
- Tester les activations de τ_i en 0, T_i, 2Ti, …KT_i, où K est tel que la tâche τ_i activée en KT_i se termine avant la prochaine activation de τ_i en (K+1) T_i

4. Périodes actives et scenarii pires cas

4.1 Pour l'ordonnancement FP (suite)

Scénario pire cas: FP non préemptif:

- Une tâche moins prioritaire peut retarder une tâche plus prioritaire si elle a débuté son exécution.

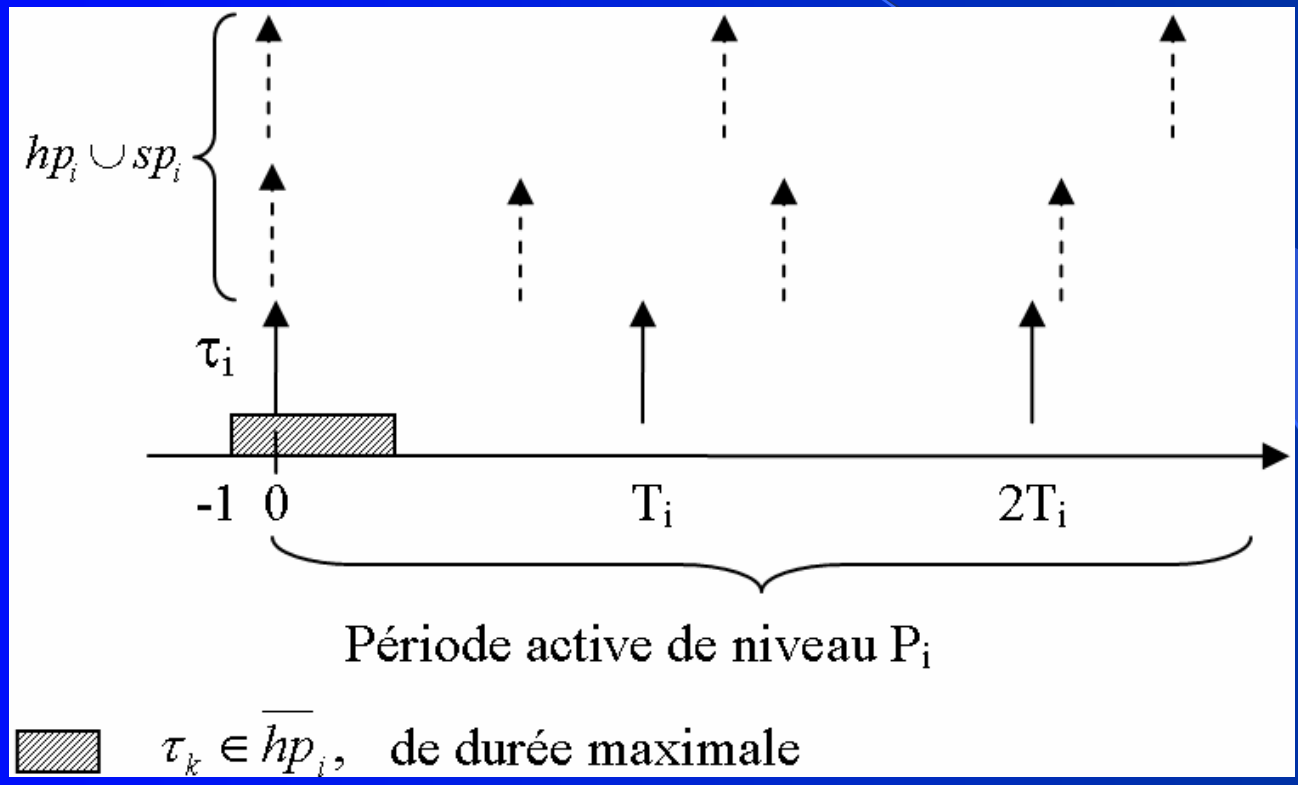


- On dénote cela « l'effet non préemptif »
- L'effet non préemptif ne peut se produire qu'une seule fois pour une tâche de priorité donnée, avant son démarrage

4. Périodes actives et scenarii pires cas

4.1 Pour l'ordonnancement FP (suite)

Scénario pire cas: cas FP non préemptif



- Là aussi, il faut tester les activations de τ_i en 0, T_i, 2Ti, …KT_i, où K est tel que la tâche τ_i activée en KT_i se termine avant la prochaine activation de τ_i en (K+1) T_i

4. Périodes actives et scenarii pires cas

4.2 Pour l'ordonnancement FP/DP et DP

Soit un ordonnanceur FP/DP ou DP qui respecte la propriété 1.

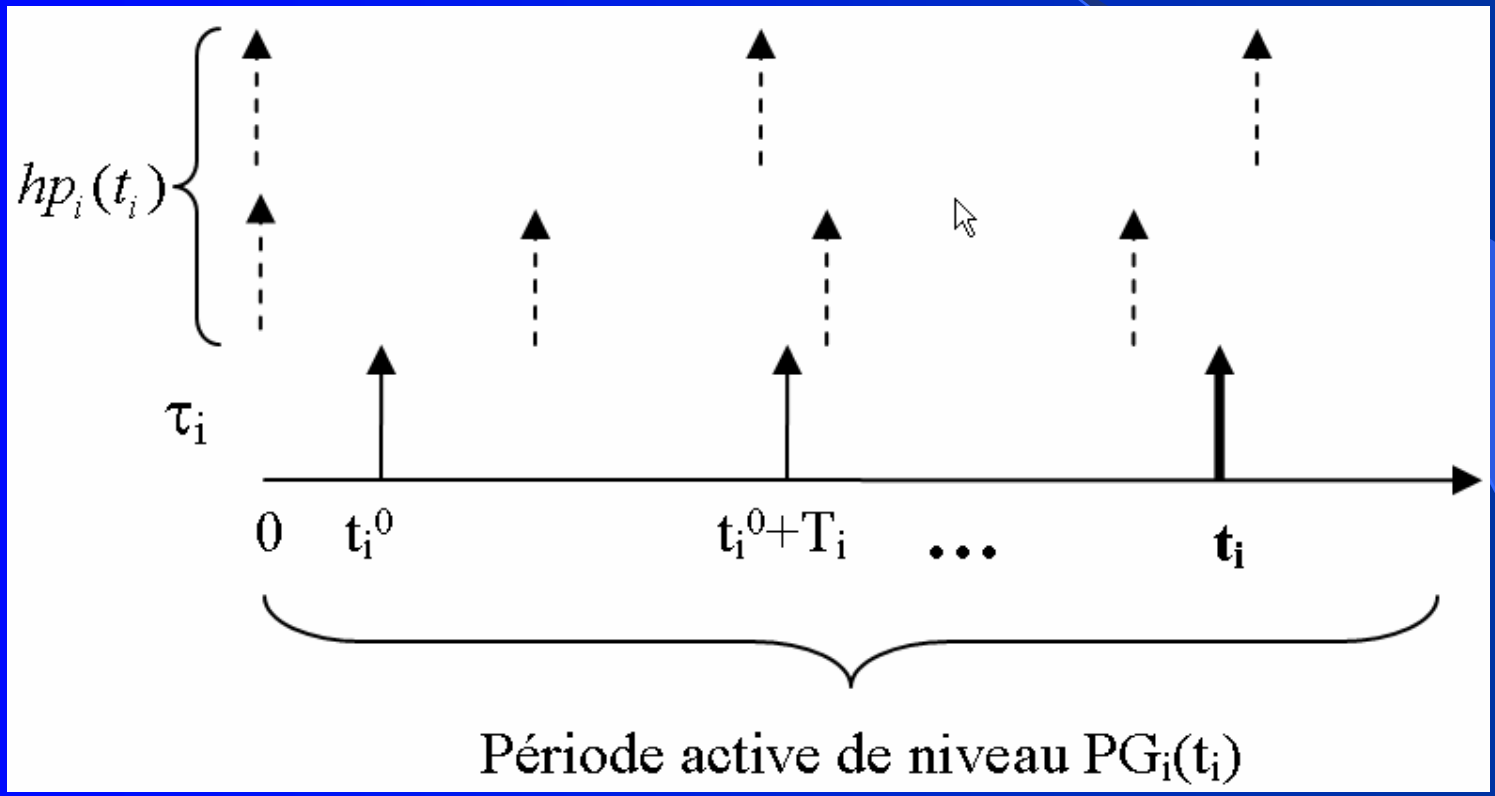
Soit τ_i une tâche activée en t_i

- Un **instant oisif t de niveau de priorité $PG_i(t_i)$** est un instant tel que toutes les tâches ayant une priorité supérieure ou égale à $PGi(t_i)$ et activées avant t sont terminées en t.
- Une **période active de niveau $PGi(t_i)$** est un intervalle de temps [a, b) tel que a et b sont deux instants oisifs de niveau $PG_i(t_i)$ et qu'il n'existe pas d'instant oisif de niveau de priorité $PG_i(t_i)$ dans l'intervalle]a, b[.

Le pire temps de réponse d'une tâche τ_i est obtenu dans une période active de niveau $PGi(t_i)$.

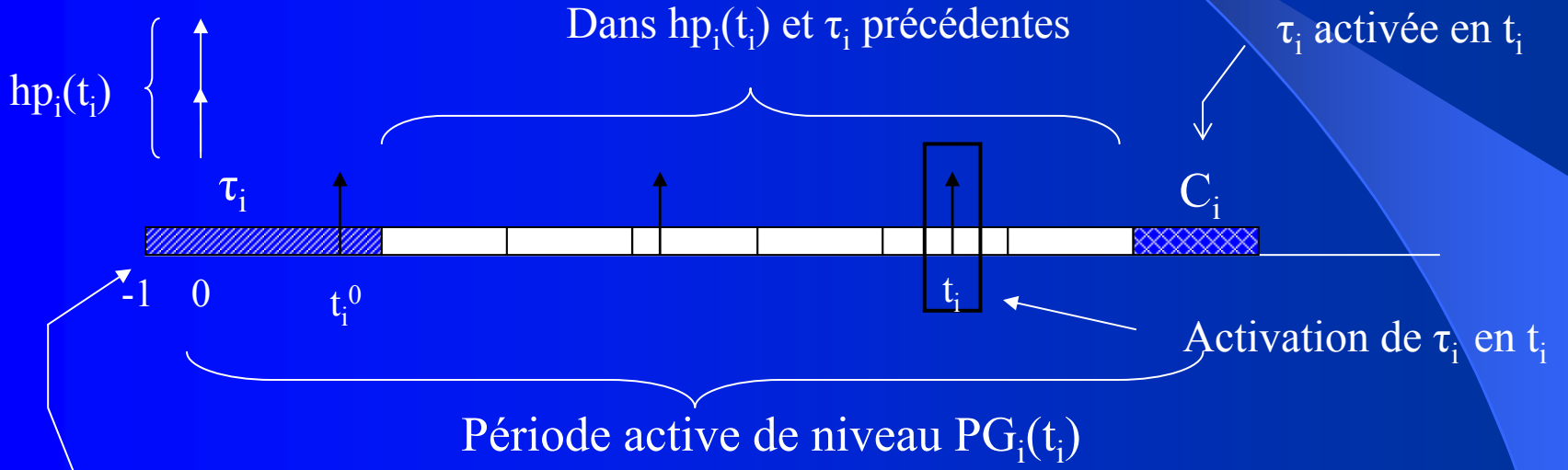
4. Périodes actives et scenarii pires cas

Scénario pire cas: cas DP préemptif



4. Périodes actives et scenarii pires cas

Scénario pire cas: *cas DP* non préemptif:



Une tâche moins prioritaire que τ_i activée en t_i de durée maximale.

4. Périodes actives et scenarii pires cas

- Bilan des activations de τ_i à tester pour DP ou FP/DP pour obtenir le pire temps de réponse de τ_i:

- Potentiellement toutes les activations de τ_i en t_i dans une période active de niveau $PG_i(t_i)$:

$$t_i = t_i^0 + kT_i, k \in \mathrm{N}, t_i^0 \in [0, T_i[$$

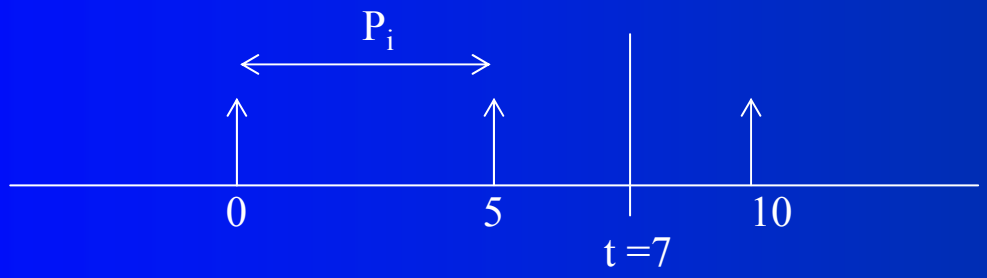
- Pour un t_i^0, on peut limiter ces instants par $B_i(t_i^0) \leq L$ la plus longue période d'activité du processeur obtenue dans le scénario totalement synchrone (toutes les tâches démarrent en 0)

5. Conditions de faisabilité

- Condition de charge (nécessaire)

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

- Quantité de travail demandée au processeur dans [0, *t*[



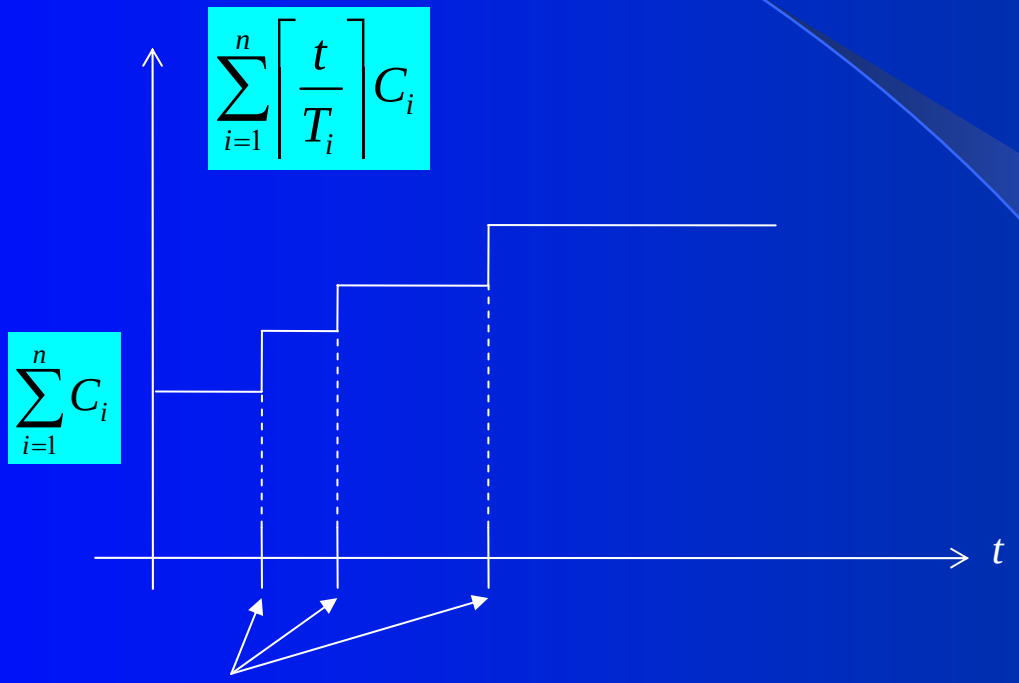
τ_i (C_i =3,P_i =5)

A t = 7, τ_i demande au processeur: 2 x 3 unités de temps

- Quantité de travail totale demandée au processeur:

$$\sum_{i=1}^{n} \left\lceil \frac{t}{T_i} \right\rceil C_i, \qquad \lceil x \rceil = \text{ partie entière supérieure de x}$$

5. Conditions de faisabilité



Demande d'activation d'une tâche

5. Conditions de faisabilité

- Quantité de travail demandée dans [0, t]:

$$\sum_{i=1}^{n}\left(1+\left\lfloor\frac{t}{T_i}\right\rfloor\right)C_i, \qquad \lfloor x \rfloor \text{ est la partie entiere de x}$$

- La plus longue période d'activité du processeur est la première période d'activité du scénario totalement synchrone =L
- L est solution de : $\sum_{i=1}^{n}\left\lceil\frac{L}{T_i}\right\rceil C_i = L$

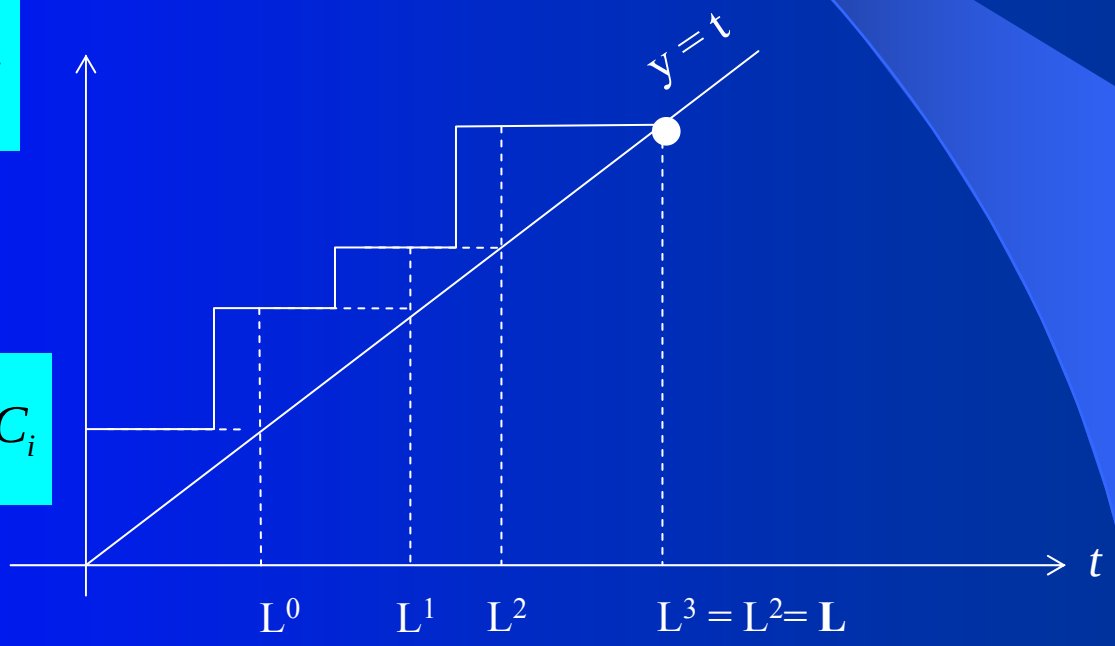
5. Conditions de faisabilité

- Pour calculer L: on utilise la suite:

$$\begin{cases} L^{m+1} = W\left(L^m\right) \\ L^0 = \sum_{i=1}^{n} C_i \end{cases}$$

$$\sum_{i=1}^{n} \left\lceil \frac{t}{T_i} \right\rceil C_i$$

$$L^0 = \sum_{i=1}^{n} C_i$$



5. Conditions de faisabilité

5.1 Ordonnancement FP

- Si $\forall i, D_i = P_i$, sans gigue
  - Si on utilise RM (optimal dans ce cas)
  - CS de faisabilité:

$$U \leq n(2^{1/n}-1)$$

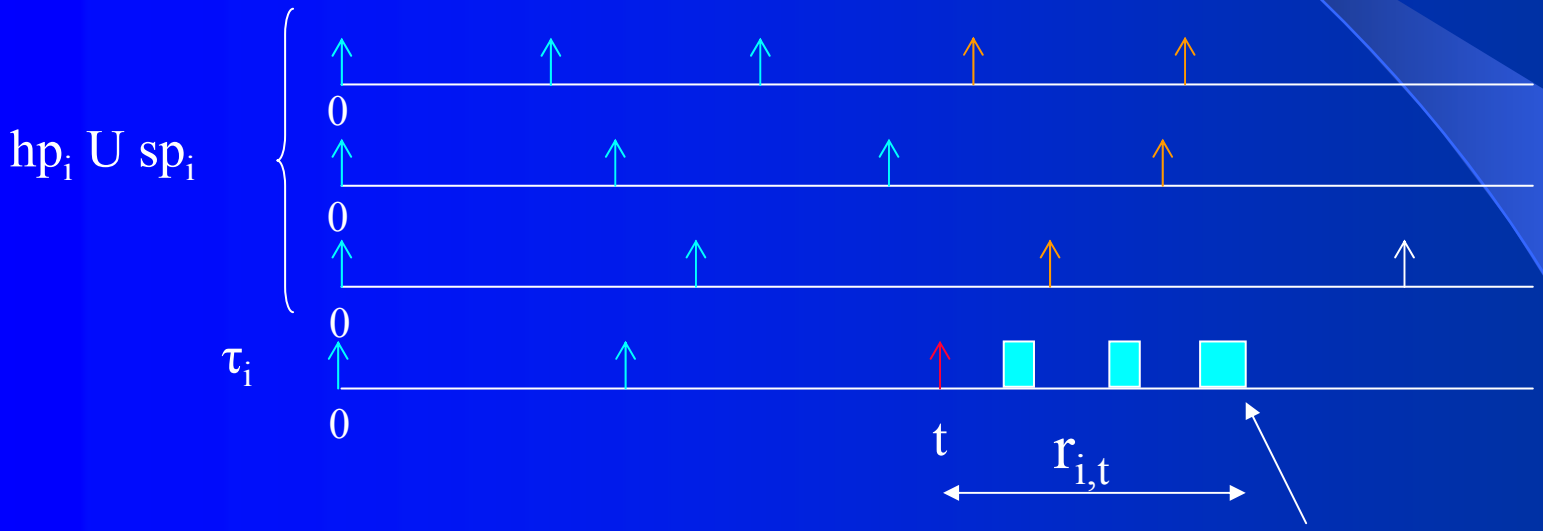
  - $\text{Lim}_{n->+\infty} [n(2^{1/n}-1)] = \ln(2) = 0{,}73$

    => tant que $U \leq 0{,}73$, RM fonctionne

5. Conditions de faisabilité

5.1 Ordonnancement FP (suite)

- Cas préemptif



- Un tâche plus prioritaire peut retarder τ_i jusqu'à la fin de son exécution

$W_i(t)$: fin d'exécution de τ_i

C_i

5. Conditions de faisabilité

5.1 Ordonnancement FP (suite)

Calcul du temps de réponse pire cas FP préemptif d'une tâche τ_i

- Cas D_i, T_i quelconque:

$$r_i = \max_{t \in S}(W_i(t) - t)$$

avec $W_i(t)$ solution de:

$$W_i(t) = (1 + \left\lfloor \frac{t}{T_i} \right\rfloor) C_i + \sum_{\tau_j \in hp_i \cup sp_i} \left\lceil \frac{W_i(t)}{T_j} \right\rceil C_j$$

et

$$S = \{kT_i, k = 0 \ldots K, k \in \mathrm{N}\}$$

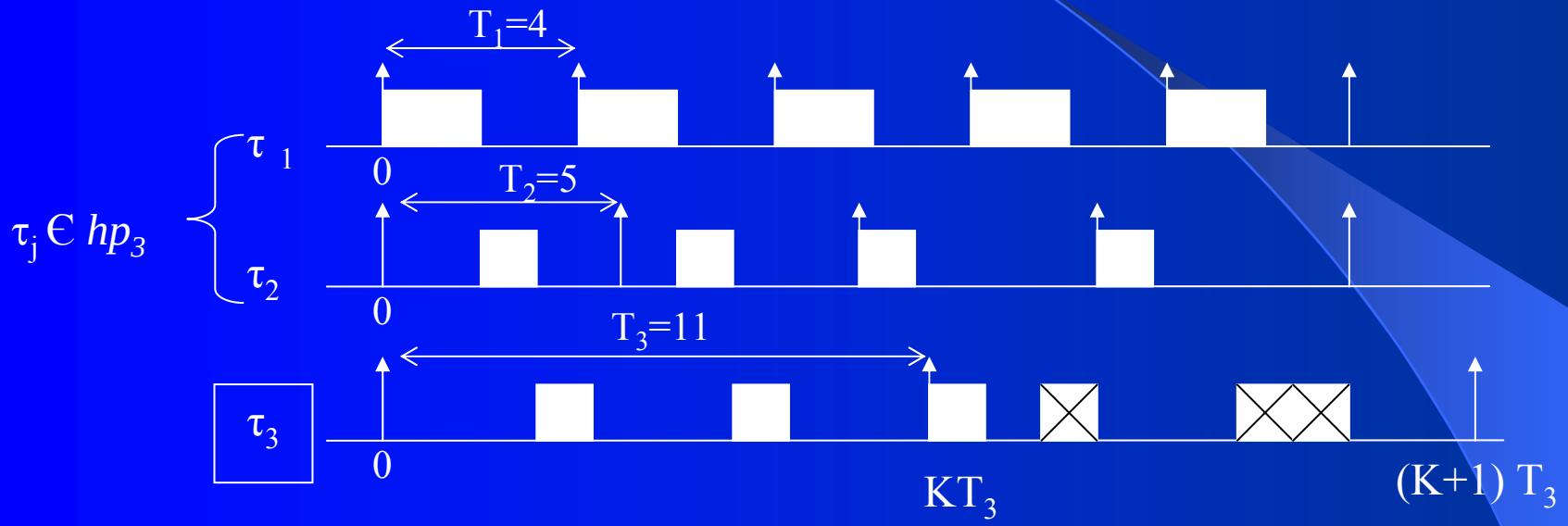
Remarques:

- K est tel que $W_i(KT_i) \leq (K+1)T_i$
- Wi(t) est la date de fin d'exécution de τ_i activée en t.
- $W_i(KT_i)$ est la longueur de la première période active de niveau i
- Si $D_i \leq T_i$, alors $W_i(KT_i) \leq T_i$ (K=0).

5. Conditions de faisabilité

5.1 Ordonnancement FP (suite)

Exemple (ordonnancement DM préemptif pour τ_3):



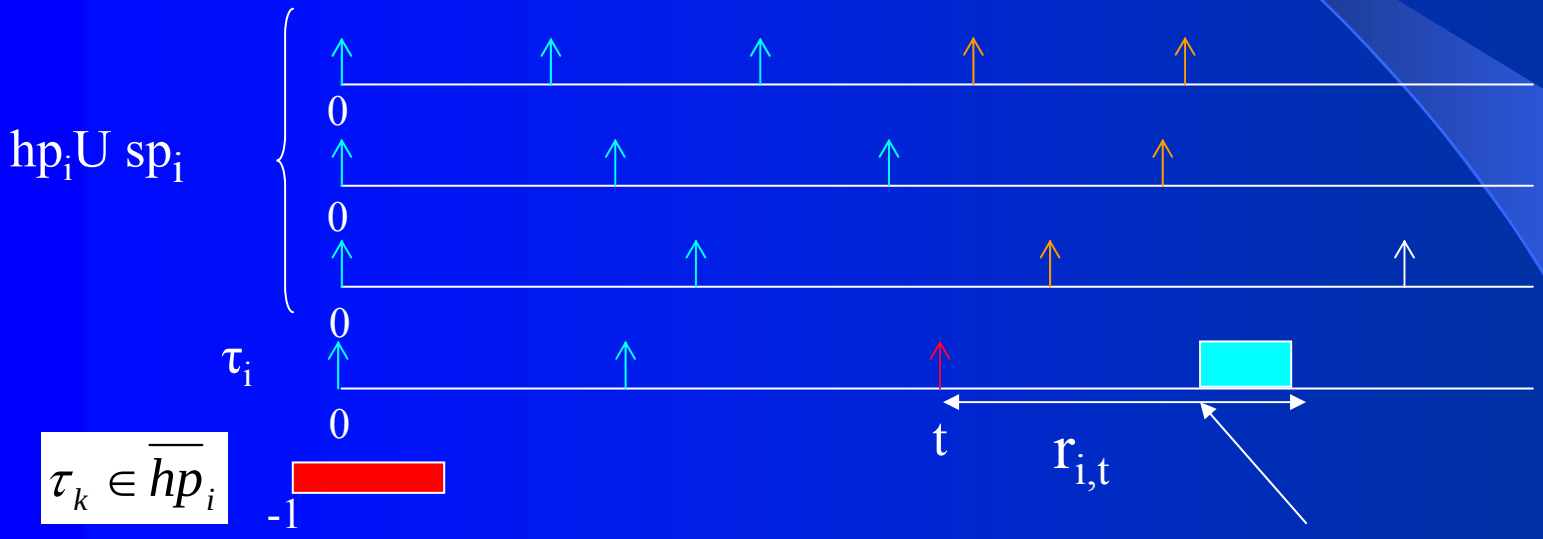
- K = 1
- 3 tâches:

	C_i	T_i	D_i
τ_1	2	4	3
τ_2	1	5	5
τ_3	3	11	12

5. Conditions de faisabilité

5.1 Ordonnancement FP (suite)

- Cas non préemptif



- Un tâche plus prioritaire ne peut plus retarder τ_i après son début d'exécution

$\overline{W_i}(t)$: début d'exécution de τ_i

C_i

5. Conditions de faisabilité

5.1 Ordonnancement FP (suite)

Calcul du temps de réponse pire cas FP non préemptif d'une tâche τ_i

- Contrairement au cas préemptif, on s'intéresse aux instants de démarrage et non de fin d'exécution des tâches

avec:

$$r_i = \max_{t \in S} (\overline{W}_i(t) + C_i - t)$$

$$\overline{W}_i(t) = \left\lfloor \frac{t}{T_i} \right\rfloor C_i + \sum_{\tau_j \in hp_i} (1 + \left\lfloor \frac{\overline{W}_i(t)}{T_j} \right\rfloor) C_j + \max_{\tau_k \in \overline{hp}_i} (C_k - 1)$$

$$S = \{kT_i, k = 0 \ldots K, k \in \mathrm{N}\}$$

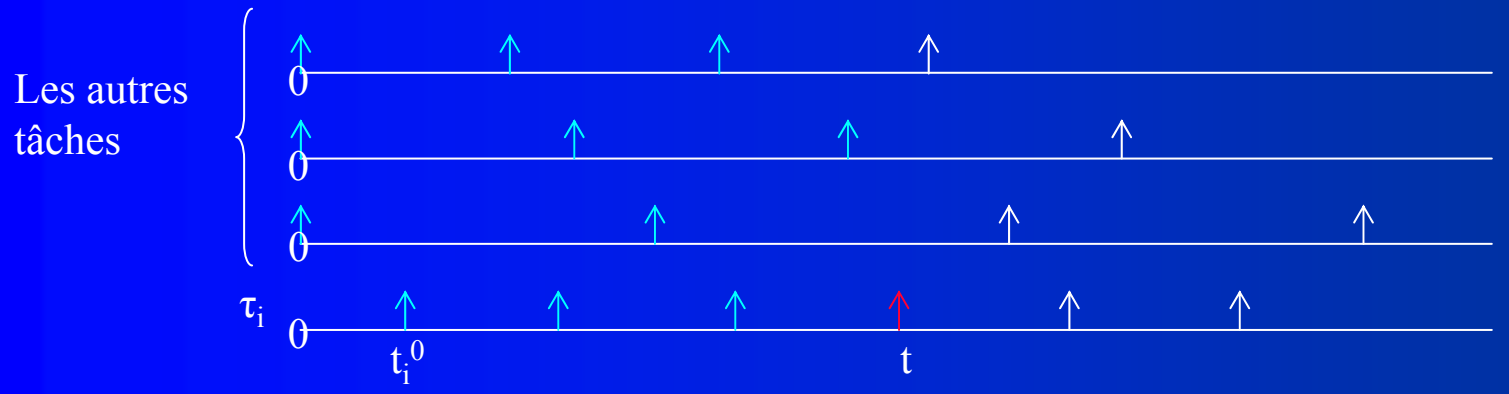
Et K tel que $W_i(KT_i) \leq (K+1)T_i$

5. Conditions de faisabilité

5.2 Ordonnancement DP

Condition de faisabilité FIFO:

- Le scénario pire cas pour une tâche est de maximiser le nombre de tâches avant elle.
- Une tâche τ_i est plus prioritaire qu'une tâche τ_j si elle est activée avant τ_j .
- Le scénario pire cas est donc pour des tâches sans gigue le synchrone pour toutes les tâches sauf τ_i en $t_i^0 \in [0,T_i[$



5. Conditions de faisabilité

- La tâche τ_i activée en t a en temps de réponse $r_{i,t}$:

$$r_{i,t} = \sum_{j \neq i} \left(1 + \left\lfloor \frac{t}{T_j} \right\rfloor\right) C_j + \left(1 + \left\lfloor \frac{t - t_i^0}{T_i} \right\rfloor\right) C_i - t$$

$$\text{avec}: t = t_i^0 + \left\lfloor \frac{t - t_i^0}{T_i} \right\rfloor T_i \text{ on trouve}:$$

$$r_{i,t} = \sum_{j=1}^{n} \left(1 + \left\lfloor \frac{t}{T_j} \right\rfloor\right) C_j - t$$

- Il faut calculer $r_{i,t}$ pour t Є S:

$$S = \bigcup_{t_i^0=0}^{T_i - 1} S_j(t_i^0) \quad , \quad S_j(t_i^0) = \left\{ t_i^0 + kT_j, k \in N \right\} \cap \left[0, B_i(t_i^0)\right[, t_i^0 \in [0, T_i[$$

$$B_i(t_i^0) = \sum_{j \neq i} \left\lceil \frac{B_i(t_i^0)}{T_j} \right\rceil Cj + \left\lceil \frac{B_i(t_i^0) - t_i^0}{T_i} \right\rceil C_i \leq L$$

5. Conditions de faisabilité

Le pire temps de réponse FIFO est donc:

$$r_i = \max_{t \in S}(r_{i,t})$$

On montre que pour des tâches sans gigue,

$$\max_{t \in S}(r_{i,t}) = \sum_{j=1}^{n} C_j$$

$$\Rightarrow \forall i = 1...n, r_i = \sum_{j=1}^{n} C_j$$

Avec FIFO, toutes les tâches ont le même temps de réponse maximum

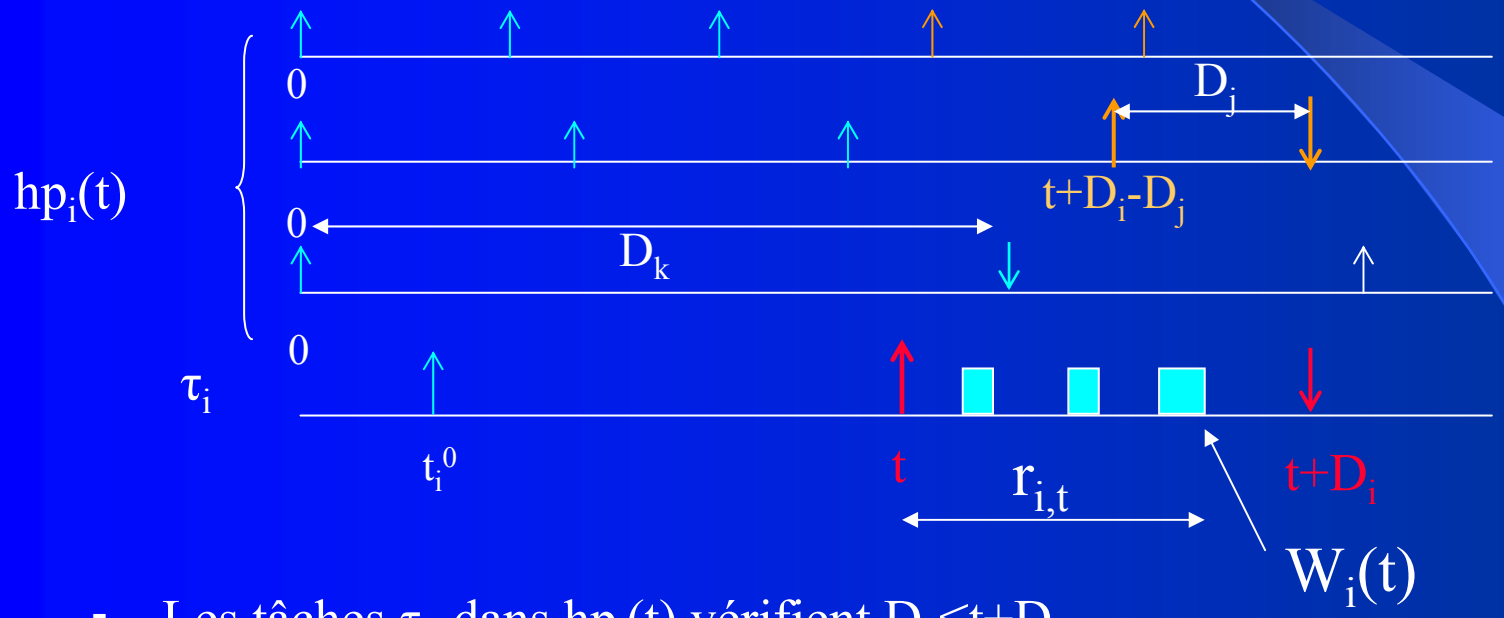
Une CNS pour FIFO est donc:

$$\forall i = 1..n, r_i \leq \min_{j=1..n}(D_j)$$

5. Conditions de faisabilité

5.2 Ordonnancement DP

Cas EDF préemptif:



- Les tâches τ_k dans $hp_i(t)$ vérifient $D_k \leq t+D_i$
- La date maximale d'activation d'une tâche τ_j dans $hp_i(t)$ pour retarder τ_i est $t+D_i-D_j$

5. Conditions de faisabilité

5.2 Ordonnancement DP (suite)

Calcul du temps de réponse maximum EDF préemptif:

$$r_i = \max_{t \in S}(W_i(t) - t)$$

Avec:

$$W_i(t) = (1 + \left\lfloor \frac{t}{T_i} \right\rfloor) C_i + \sum_{\tau_j \in hp_i(t)} \min\left(\left\lceil \frac{W_i(t)}{T_j} \right\rceil j, 1 + \left\lfloor \frac{t + D_i - D_j}{T} \right\rfloor \right) C_j$$

$$S = \bigcup_{t_i^0 = 0}^{T_i - 1} S_j(t_i^0) \quad , \quad S_j(t_i^0) = \left\{ t_i^0 + kT_j, k \in N \right\} \bigcap \left[0, B_i(t_i^0) \right[, t_i^0 \in [0, T_i[$$

$$B_i(t_i^0) = \sum_{j \neq i} \left\lceil \frac{B_i(t_i^0)}{T_j} \right\rceil Cj + \left\lceil \frac{B_i(t_i^0) - t_i^0}{T_i} \right\rceil C_i \leq L$$

5. Conditions de faisabilité

5.3 Ordonnancement FP/FIFO

- Cas FP préemptif:

$$r_i = \max_{t \in S}(W_i(t) - t)$$

$$W_i(t) = \sum_{\tau_j \in sp_i \cup \tau_i} (1 + \left\lfloor \frac{t}{T_j} \right\rfloor) C_j + \sum_{\tau_j \in hp_i} \left\lceil \frac{W_i(t)}{T_j} \right\rceil C_j$$

Avec:

$$S = \bigcup_{t_i^0 = 0}^{T_i - 1} S_j(t_i^0) \quad , \quad S_j(t_i^0) = \{t_i^0 + kT_j, k \in N\} \cap [0, B_i(t_i^0)[, t_i^0 \in [0, T_i[$$

$$B_i(t_i^0) = \sum_{j \neq i} \left\lceil \frac{B_i(t_i^0)}{T_j} \right\rceil Cj + \left\lceil \frac{B_i(t_i^0) - t_i^0}{T_i} \right\rceil C_i \leq L$$

5. Conditions de faisabilité

5.3 Ordonnancement FP/FIFO (suite)

Cas FP non préemptif:

$$r_i = \max_{t \in S} (\overline{W}_i(t) + C_i - t)$$

$$\overline{W}_i(t) = \sum_{\tau_j \in sp_i} \left(1 + \left\lfloor \frac{t}{T_j} \right\rfloor\right) C_j + \left\lfloor \frac{t}{T_i} \right\rfloor C_i + \sum_{\tau_j \in hp_i} \left\lceil \frac{\overline{W}_i(t)}{T_j} \right\rceil C_j + \max_{\tau_k \in \overline{hp_i}} (C_k - 1)$$

Avec:

$$S = \bigcup_{t_i^0 = 0}^{T_i - 1} S_j(t_i^0) \quad , \quad S_j(t_i^0) = \left\{ t_i^0 + kT_j, k \in N \right\} \cap \left[0, B_i(t_i^0)\right[, t_i^0 \in [0, T_i[$$

$$B_i(t_i^0) = \sum_{j \neq i} \left\lceil \frac{B_i(t_i^0)}{T_j} \right\rceil Cj + \left\lceil \frac{B_i(t_i^0) - t_i^0}{T_i} \right\rceil C_i \leq L$$

5. Conditions de faisabilité

5.2 Ordonnancement FP/FIFO (suite)

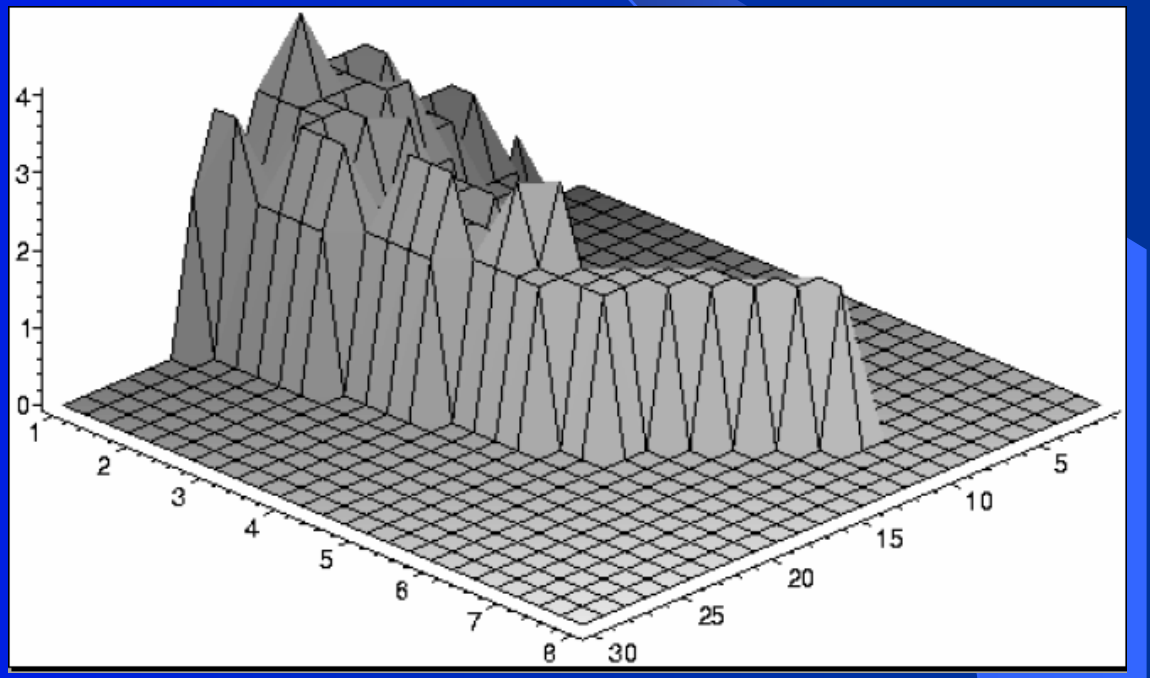
Exemple d'amélioration de la région d'ordonnançabilité avec FP/FIFO par rapport aux test FP (différence entre les deux régions)

On fait varier C_1, C_2 et C_3 jusqu'à la limite de faisabilité

3 tâches τ_1, τ_2, τ_3 de paramètres :

- T_1=50, D_1=100, P_1=1
- T_2=10, D_2=25, P_2=2
- T_3=10, D_3=35, P_3=2

FP non préemptif



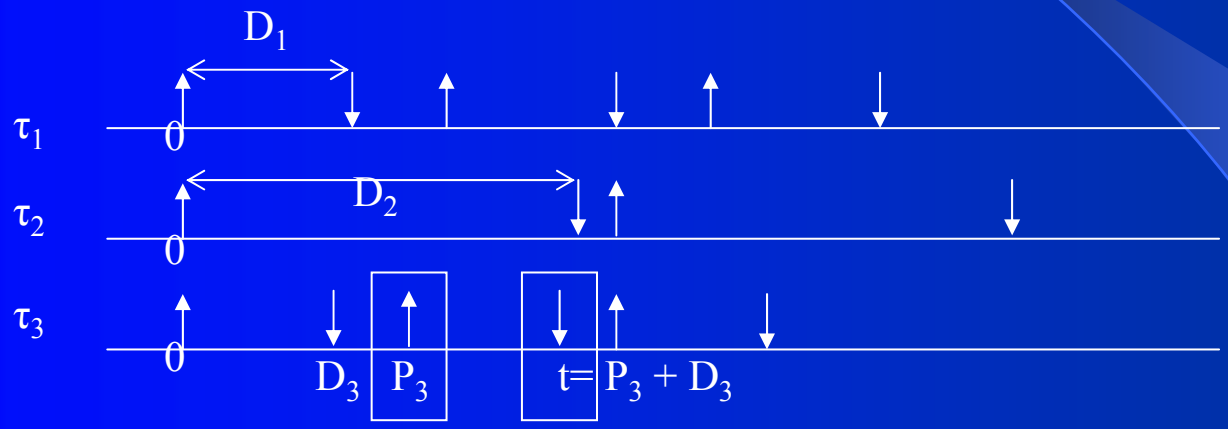
Conclusion

Quelques problèmes récents:

- Robustesse temporelle (prise en compte de l'imperfection des OS)
- Java et temps réel (problème du WCET)
- Condition de faisabilité approchées polynomiales
- Prise en compte du coût de l'ordonnanceur dans le contexte event driven
- Multiprocesseur
- Algorithmes qui optimisent plusieurs critères (échéance, consommation d'énergie, gigue maximale,…)
- Respect de contraintes temporelles dans un graphe distribué

Scénario pire cas EDF préemptif pour la tâche τ_3 d'échéance en t:

- Le scénario pire cas dépend de la tâche et de son instant d'activation



- Pour τ_3 activée en P_i, à l'instant *t*, τ_1 retarde τ_3 mais pas τ_2
- Scénario pire cas:
  - Placer toutes les tâches qui vérifient $D_j \leq t$ en 0

Condition de faisabilité EDF:

- Scénario pire cas pour la faisabilité: synchrone
- On détermine à un instant t dans le scénario synchrone la nombre d'activations de la tâche τ_i qui ont atteint leur échéance dans [0,t]

$$\max\left(0,1+\left\lfloor\frac{t-D_i}{P_i}\right\rfloor\right)$$

- Une CNS pour que EDF doit faisable est que la quantité de travail qui résulte des tâches ayant atteint leur échéance soit inférieure à la quantité de travail exécutée par le processeur.

$$\forall t>0, h(t)=\sum_{i=1}^{n}\max\left(0,1+\left\lfloor\frac{t-D_i}{P_i}\right\rfloor\right)C_i \le t$$

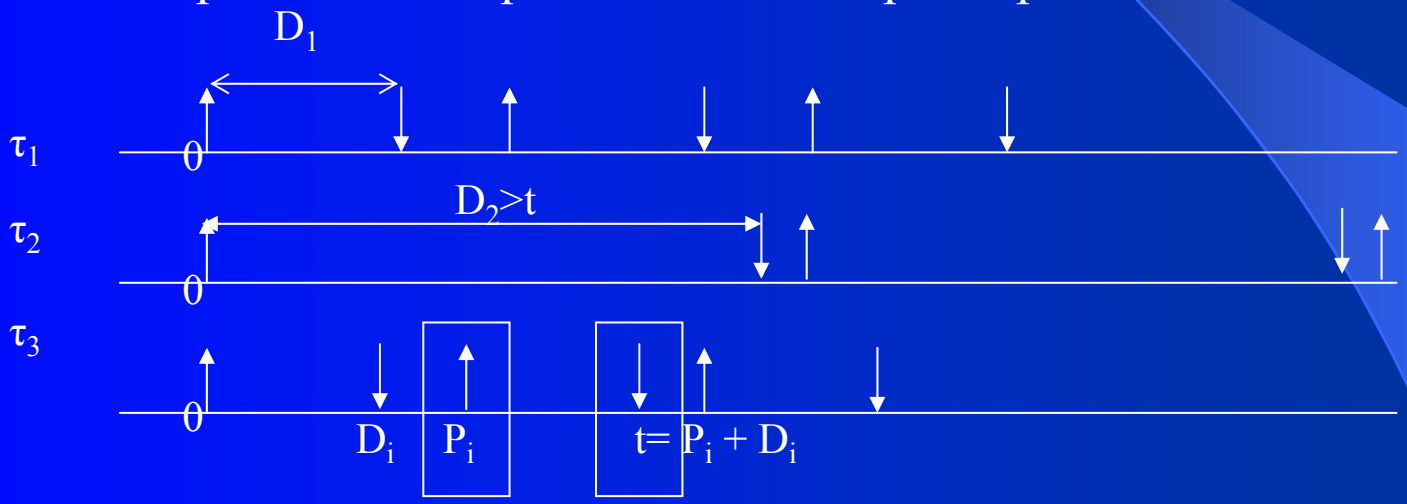
- On peut ramener cette CNS à:

$$\forall t \in S, h(t) \leq t$$

$$\text{Avec } S = \bigcup_{i=1}^{n} \{kP_i + D_i, k \in N\} \cap [0, L[$$

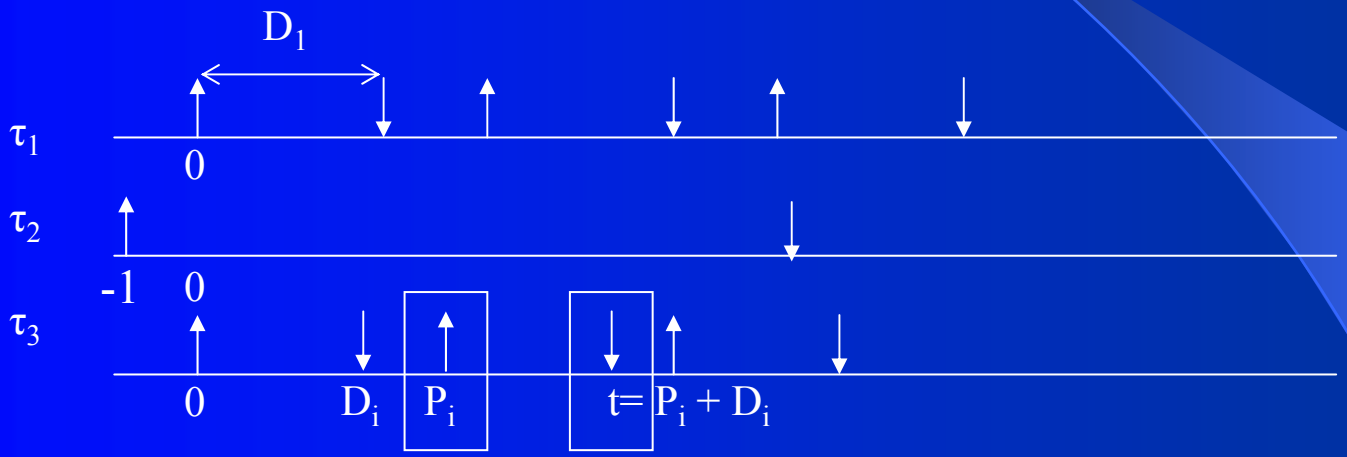
Scénario pire cas EDF préemptif pour τ_3 ayant une échéance en t insuffisant:

- Il faut en plus tenir compte de l'effet non-préemptif



- Pour τ_3 d'échéance $t=P_3+D_3$, activée en P_3, il faut placer τ_2 en -1 pour qu'elle gène τ_3

Scénario pire cas EDF non préemptif pour τ_3 ayant une échéance en t :



- Scénario pire cas:
  - Placer toutes les tâches qui vérifient $D_j \leq t$ en 0
  - Placer une tâche τ_k du durée max en -1 telle que $D_k > t$

Condition de faisabilité EDF non préemptif:

- $h(t) \leq t$ est nécessaire mais plus suffisant

CNS EDF Finale:

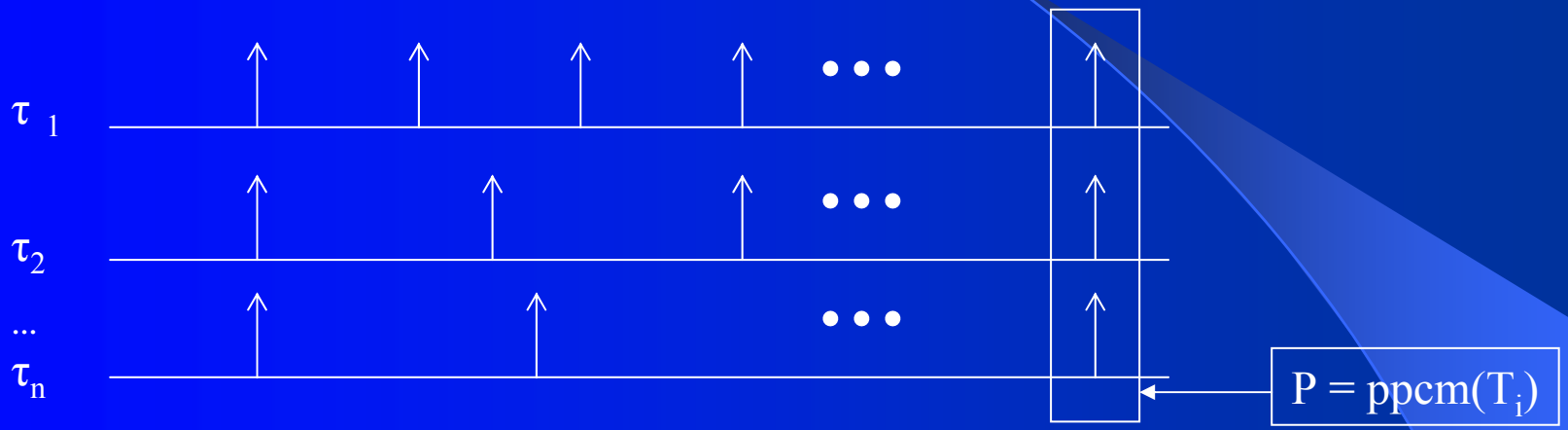
$$\forall t \in S, \max_{D_k > t}(C_{k-1}) + h(t) \leq t$$

$$\text{Avec}: \quad S = \bigcup_{i=1}^{n} \{kP_i + D_i, k \in N\} \cap \{0, L\}$$

- On ne teste que les échéances des tâches dans le scénario synchrone, dans [0, L[

5. Conditions de faisabilité

Cas non-concret: Durée de test du scénario synchrone:



Le scénario synchrone se reproduit au bout de P=ppcm(T_i)

Condition nécessaire et suffisante de faisabilité pour tout algorithme en préemptif:

- $U \leq 1$
- Tester si les tâches respectent leurs échéances pour le scénario synchrone dans [0, P]

5. Conditions de faisabilité

Le test peut être long

Ex: $\forall i, P_i \geq 2$ et les P_i sont premières entre elles:

$\Rightarrow P \geq 2^n$ pour n tâches

- Amélioration: il suffit de tester l'algorithme dans [0, L]
  - A l'instant L il n'y a plus de tâches activée avant L
  - En L, le scénario qui redémarre est:
    - ✓ Soit ≠ synchrone => moins dur que celui testé
    - ✓ Soit = synchrone => déjà testé

5. Conditions de faisabilité

Test général pour tout algorithme en contexte préemptif non concret:

- Tester que l'algorithme respecte les échéances des tâches pour le scénario synchrone dans [0 , L]
- Vérifier que $U \leq 1$

=> C'est une CNS !