Formation Automatique et Informatique Industrielle

Master 1 S2

Matière : Systèmes Embarqués et Systèmes Temps Réel SE-STR

Par : ATOUI Hamza

Plan du cours

- Ordonnancement ? Objectifs ?
- Dépend de quoi ?
- Tâche et présentation d'état.
- Introduction sur l'ordonnancement :
  - **Modèle de tâche.**
  - **Facteur d'utilisation.**
  - **Facteur de charge.**
  - **Test de faisabilité.**
  - **Séquence valide.**

Ordonnancement ? Objectifs ?

- Ordonnancement (**SCHEDULING**) : est l'opération réalisée par le service le plus important d'un OS/RTOS appelé l'ordonnanceur (**SCHEDULER**). Ce dernier fait la recherche/chargement de la tâche **de la plus haute priorité prête à exécuter** au CPU; dont les objectifs à atteindre sont :

Ordonnancement ? Objectifs ?

- **La faisabilité/ordonnancement** : Il doit être possible d'exécuter toutes les tâches dans les délais spécifiés (satisfaction des contraintes temporelles).
- **Optimalité** : Le taux de charge du processeur doit rester le plus bas possible avec les changements de contexte entre tâches sont minimaux.
- **Complexité raisonnable** : facilité de mise en œuvre (implémentation).

Dépend de quoi ?

- Les politiques d'ordonnancement dépend d'une façon générale de :
  1. **Le nombre de CPU à utiliser.**
  2. **Classe de système (Hard ou Soft).**
  3. **Catégories de tâche (périodique, apériodique, dépendante,…).**
  4. **Modèle et hypothèses sur tâche (Arrival, Start, Deadline dates,…).**
  5. **Type de priorité (statique ou dynamique).**
  6. **Le temps d'allocation du CPU au tâche choisi (Préemptif ou Coopératif).**
  7. **Temps de charge du noyau : (négligeable ou non)**

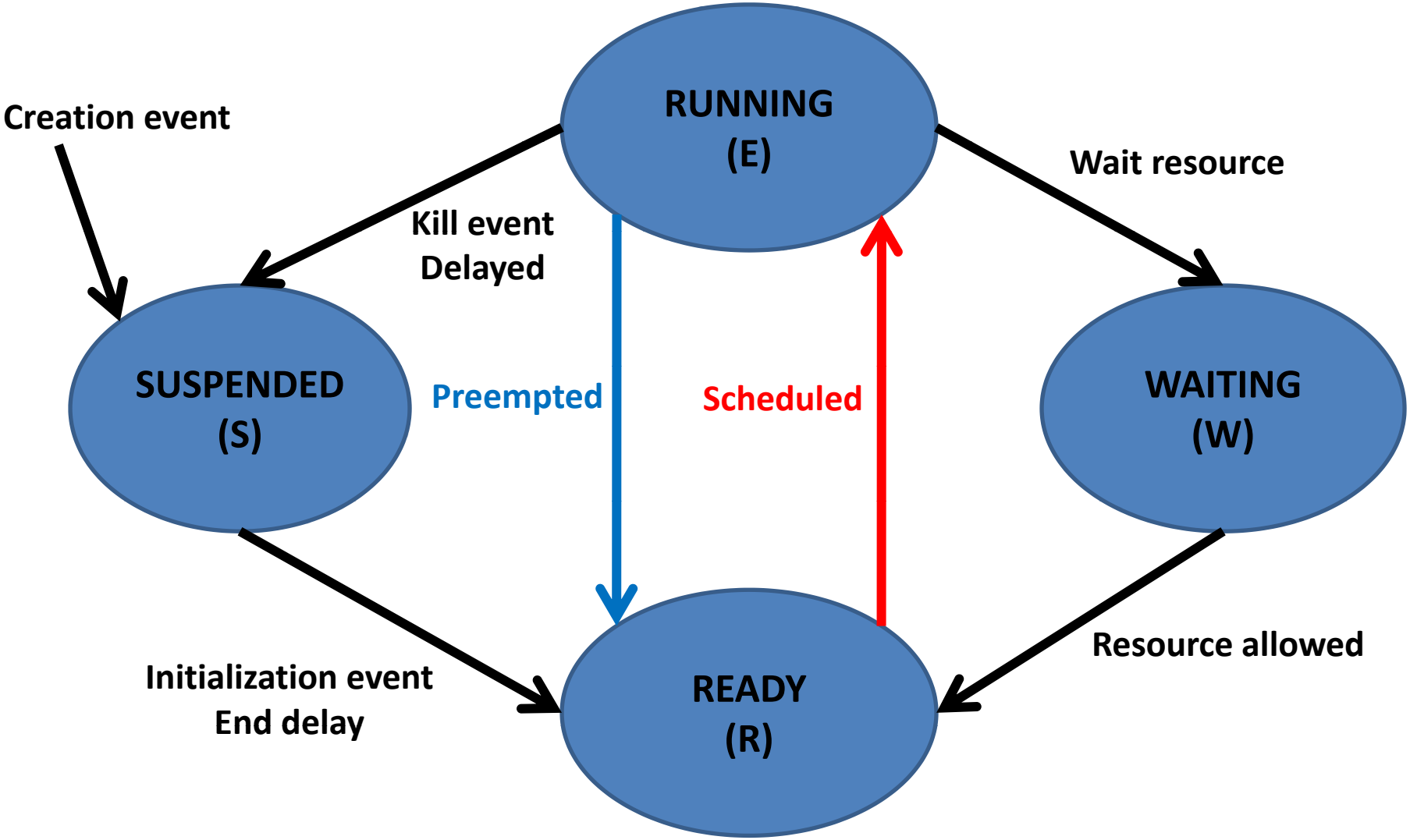
Dépend de quoi ?

- Exemple : l'algorithme **RM** qu'on va étudier prend en compte les contraintes suivantes :
  1. Nombre de CPU = 1.
  2. Classe de système = Hard/Soft avec échéance sur requête Deadline Date est à la fin de la période .
  3. Catégorie de tâches : périodique indépendante.
  4. Modèle et hypothèses sur tâche : Task(Arrival Date = 0, Burst Time, Period).
  5. Type de priorité : statique.
  6. Le temps d'allocation du CPU : préemptif.
  7. Temps de charge du noyau : négligeable.

Tâche et présentation d'état

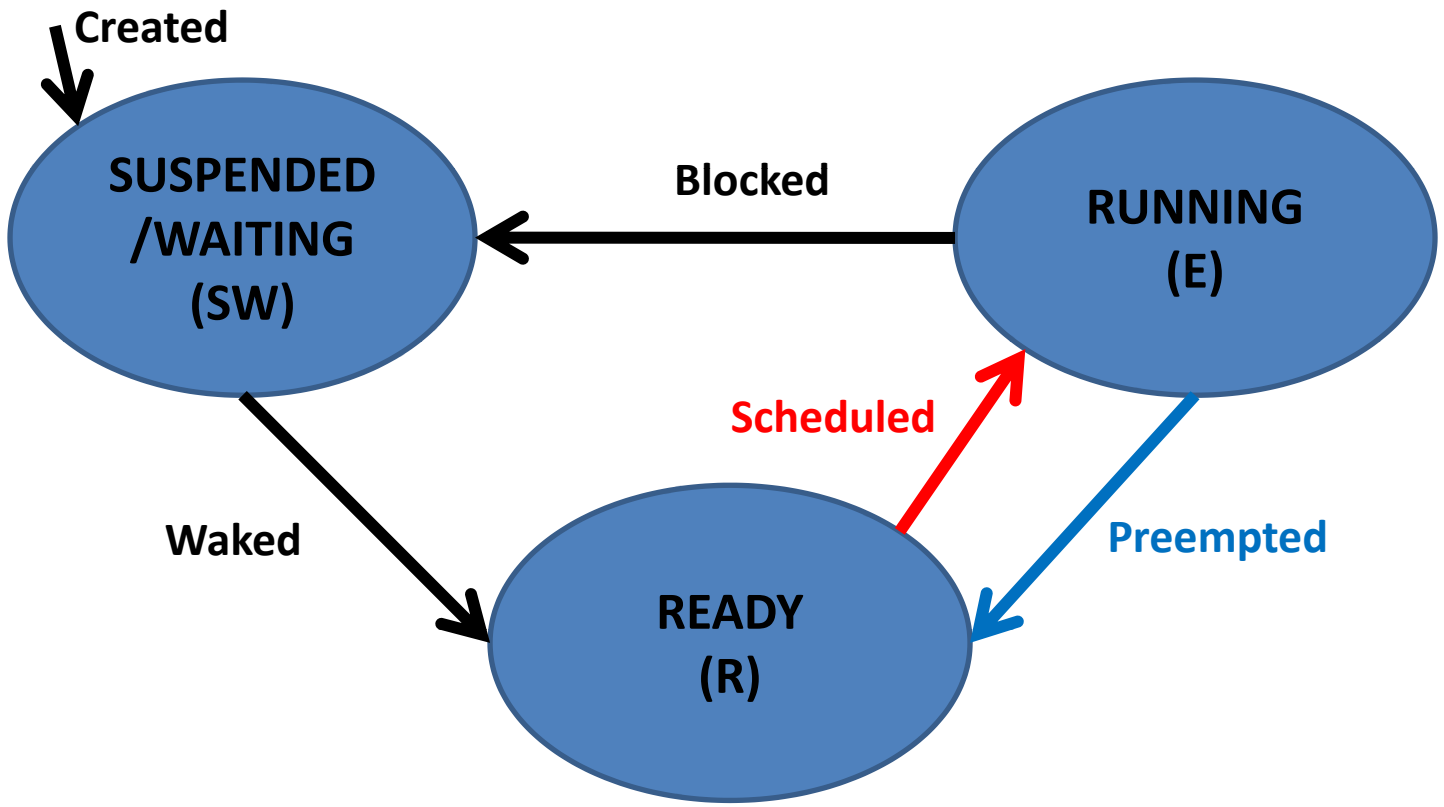
- Tâche : est l'entité de base d'un système qui prend un ensemble d'états de sa date de naissance jusqu'à la date de terminaison (Etat : **SUSPENDED**, **READY**, **RUNNING** et **WAITING**).
- Le graphe d'état suivant présente les états et transitions d'une tâche dans un contexte temps réel.

Tâche et présentation d'état



Tâche et présentation d'état

- Pour des raisons de simplification et pour bien comprendre les politiques d'ordonnancement, on va fusionner l'état SUSPENDED avec WAITING et le graphe d'état devient le suivant:

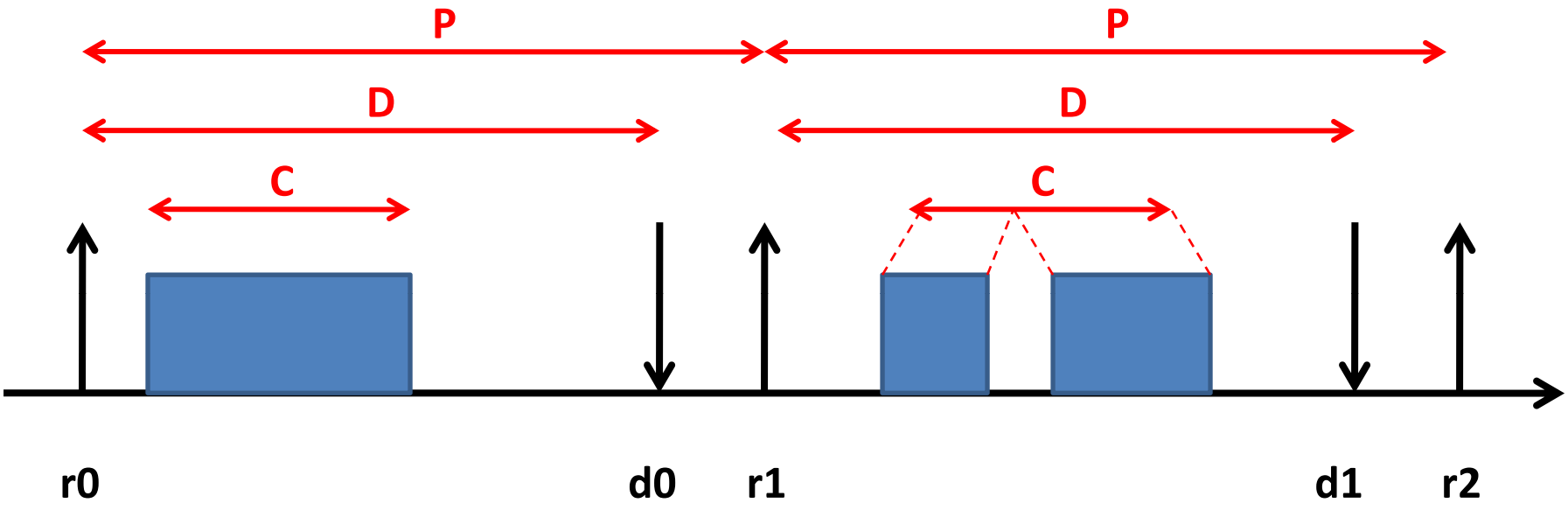


Politiques (algorithme) d'ordonnancement

- Avant d'étudier les politiques d'ordonnancement, on va prendre en compte le suivant :
- Nombre de CPU : 1.
- Le temps d'allocation du CPU : préemptif.
- Modèle de tâche : Task(r0, C, D, P).
  - r0 : Arrival Date.
  - C : Burst Time (Capacity).
  - D : Critical Delay.
  - P : Period.
- La figure suivante présente le chronogramme d'une tâche par les paramètres (r0, C, D, P) :

Politiques (algorithme) d'ordonnancement

- Chronogramme d'une tâche par les paramètres (r0, C, D, P):



d0 : première échéance = r0 + D

d1 : deuxième échéance = r1 + D

Politiques (algorithme) d'ordonnancement

- D'après les paramètres d'une tâche, on distingue les deux facteurs suivants :
- Facteur d'utilisation d'une tâche i : **Ui = Ci/Pi**.
- Facteur de charge d'une tâche i : **CHi = Ci/Di**.
- Pour un système de N tâches, les deux facteurs sont définis comme suit:

$$U = \sum_{i=1}^{N} C_i / P_i$$

$$CH = \sum_{i=1}^{N} C_i / D_i$$

Quelle est l'utilité du calcul de U et CH ???

Politiques (algorithme) d'ordonnancement

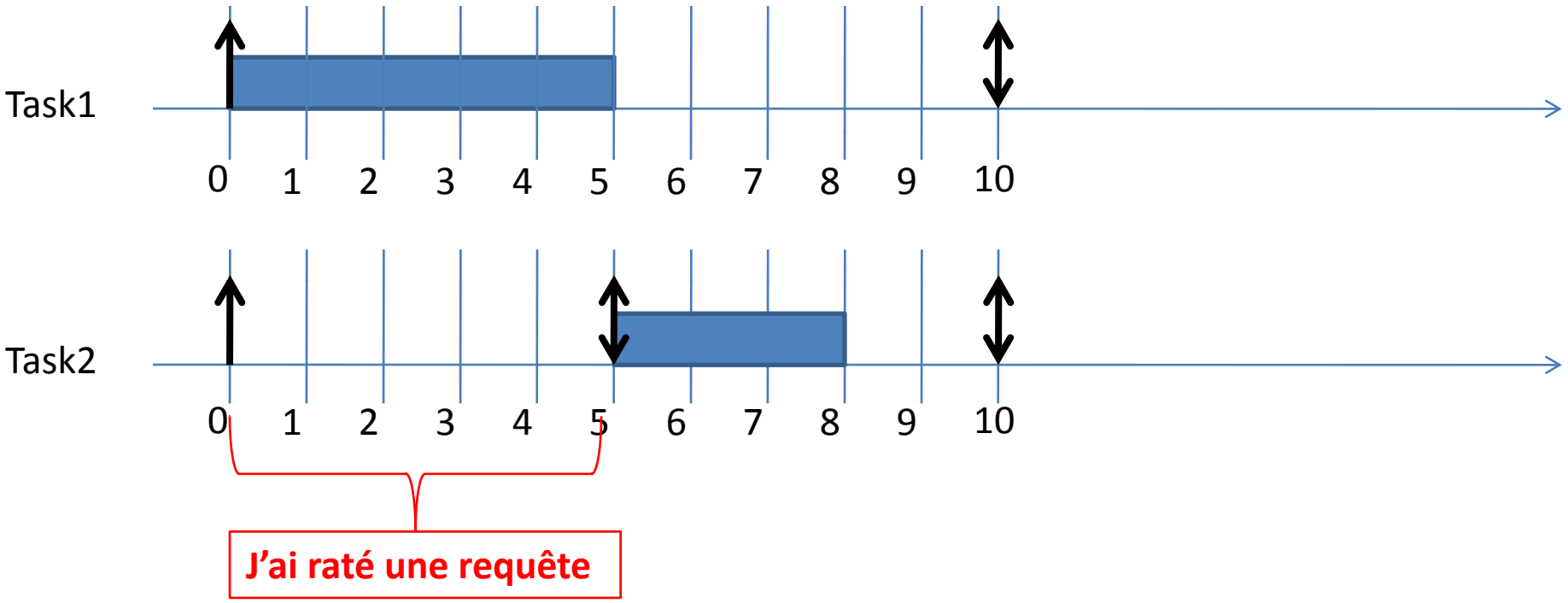
- Avant de rependre à cette question, on essaye d'ordonnancer les deux tâches suivantes :
- Task1(r0 = 0, C = 5, D = 10, P = 10).
- Task2(r0 = 0, C = 3, D = 5, P = 5).
- La première étape est le calcul de l'intervalle d'étude = [0, HyperPeriod = LCM(P1,P2)].
- LCM : Least Comman Multiple.
- HyperPeriod = LCM(10,5) = 10.

Politiques (algorithme) d'ordonnancement

- J'ai deux tâches donc, deux scénarios:
  - **Scénario 1 : Task1 plus prioritaire que Task2.**
  - **Scénario 2 : Task2 plus prioritaire que Task1.**

Politiques (algorithme) d'ordonnancement

- Scénario 1 : Task1 puis Task2
- Task1(r0 = 0, C = 5, D = 10, P = 10).
- Task2(r0 = 0, C = 3, D = 5, P = 5).

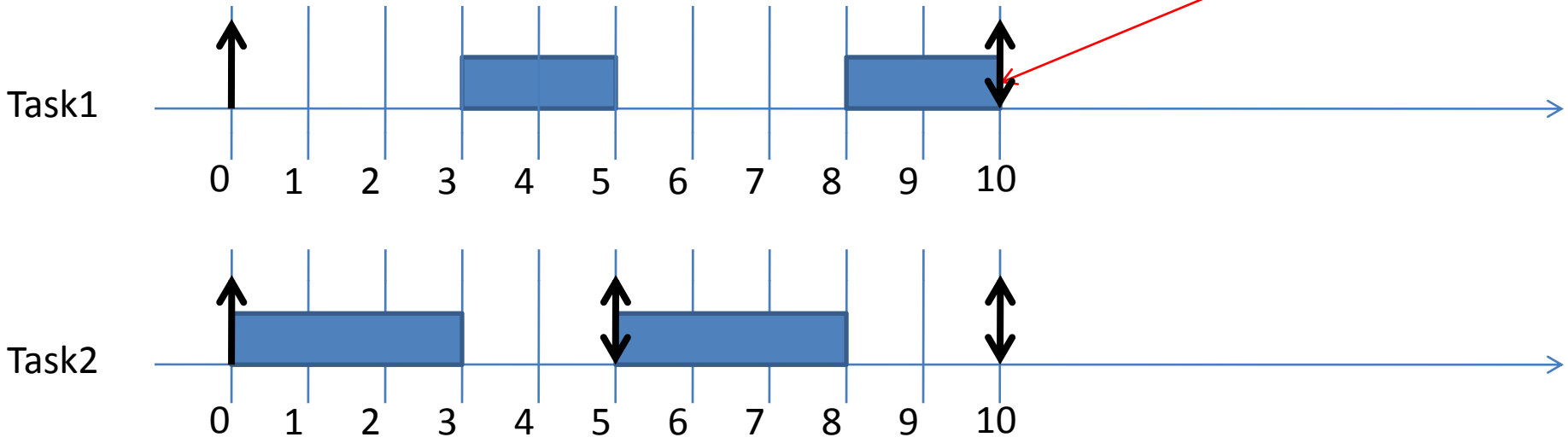


Donc, le scénario 1 n'est pas faisable

Politiques (algorithme) d'ordonnancement

- Scénario 2 : Task2 puis Task1
- Task1(r0 = 0, C = 5, D = 10, P = 10).
- Task2(r0 = 0, C = 3, D = 5, P = 5).

Une nouvel requête de Task1 arrive main, on n'a pas encore terminé l'exécution de la première requête



le scénario 2 n'est pas aussi faisable !!!!!

Donc, j'ai besoin de faire un test de faisabilité

Politiques (algorithme) d'ordonnancement

- **Test de faisabilité** : est un test sur la possibilité/capacité de prendre on charge un ensemble de N tâches sur une plateforme (Nombre de CPU – **NCPU**-).
- Dans notre cas le nombre de CPU est 1 (**NCPU=1**)donc :
- Si les tâches du système à échéance sur requête, le test de faisabilité est :

$$U = \sum_{i=1}^{N} C_i / P_i \leq NCPU = 1$$

- Si les tâches du système à échéance inferieur à la période, le test de faisabilité est :

$$CH = \sum_{i=1}^{N} C_i / D_i \leq NCPU = 1$$

Politiques (algorithme) d'ordonnancement

- Pour notre exemple :
- Task1(r0 = 0, C = 5, D = 10, P = 10).
- Task2(r0 = 0, C = 3, D = 5, P = 5).
- U = 5/10 + 3/5 = 1.1 = 110% > 100%
- Donc, il n'y pas faisable de prendre en charge sur une plateforme de nombre de CPU est égal à 1.

Politiques (algorithme) d'ordonnancement

- **Exemple 2 :**
- Task1(r0 = 0, C = 5, D = 10, P = 10).
- Task2(r0 = 0, C = 2, D = 5, P = 5).

- Est-ce qu'il est faisable de prendre en charge ?
- Essayer d'ordonnancer ces deux tâches.

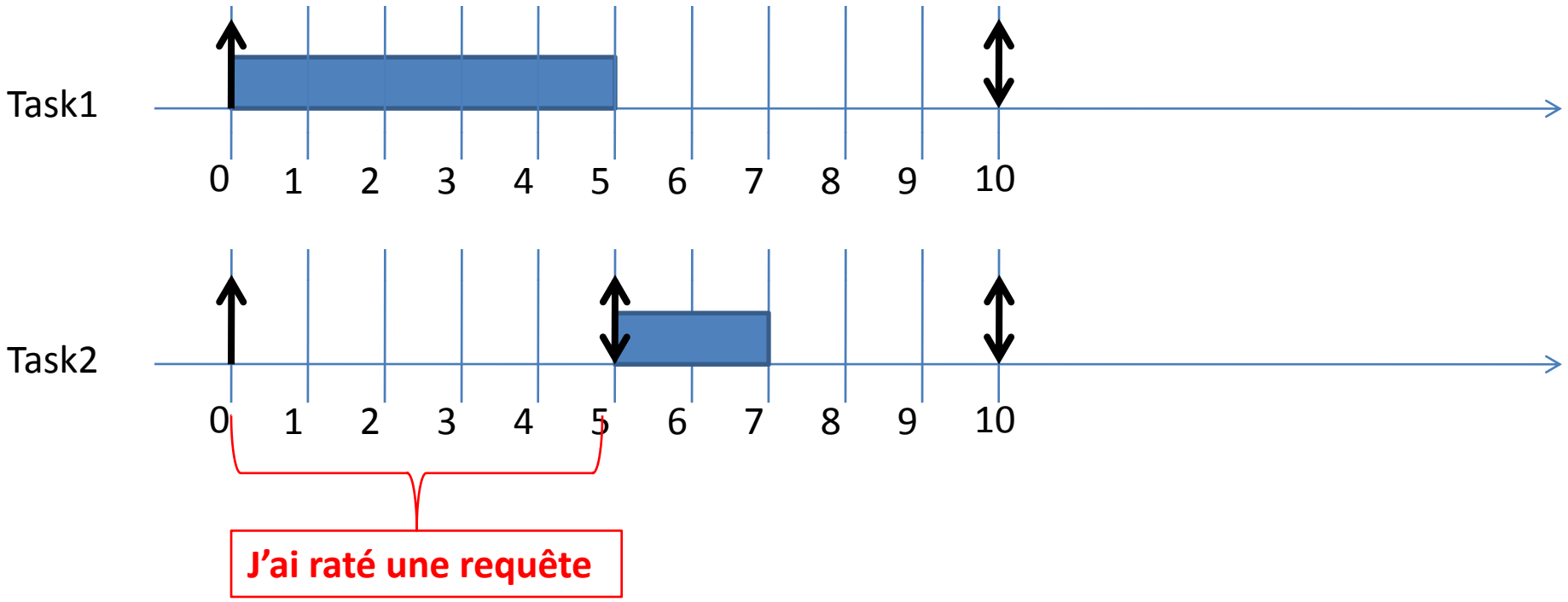
Politiques (algorithme) d'ordonnancement

- Test de faisabilité : les deux tâches sont à échéance sur requête, donc :
- U = 5/10 + 2/5 = 0.90 = 90% < 100% (faisable).

- Scénario 1 : Task1 puis Task2.
- Scénario 2 : Task2 puis Task1.

Politiques (algorithme) d'ordonnancement

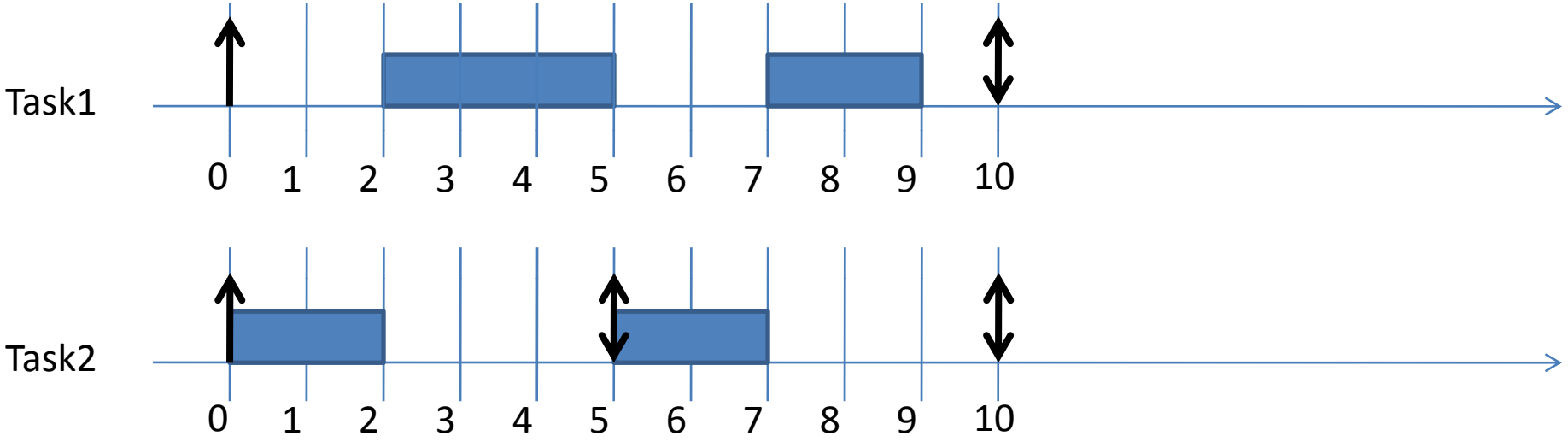
- Scénario 1 : Task1 puis Task2
- Task1(r0 = 0, C = 5, D = 10, P = 10).
- Task2(r0 = 0, C = 2, D = 5, P = 5).



Donc, le scénario 1 n'est pas une séquence valide

Politiques (algorithme) d'ordonnancement

- Scénario 2 : Task2 puis Task1
- Task1(r0 = 0, C = 5, D = 10, P = 10).
- Task2(r0 = 0, C = 2, D = 5, P = 5).



le scénario 2 est une séquence valide

Politiques (algorithme) d'ordonnancement

- Un système à N tâches est ordonnancé, si au moins une ou plusieurs **séquences valides** parmi l'ensemble des séquences possibles.

- **Une séquence valide** : est une séquence qui **satisfait** les **contraintes temporelles** du système.

Pour les meilleurs

- Redessinez le chronogramme de deuxième scénario de l'exemple en spécifiant à chaque instant l'état de la tâche (R, E ou SW).