Ordonnancement centralisé de tâches temps réel

Samia Bouzefrane

Maître de Conférences

CEDRIC –CNAM

samia.bouzefrane@cnam.fr
http://cedric.cnam.fr/~bouzefra

Sommaire

1. Caractéristiques de l'ordonnancement temps réel
2. Ordonnancement de tâches périodiques temps réel
3. Ordonnancement de tâches apériodiques
4. Ordonnancement avec contraintes de ressources
5. Prise en compte de la précédence des tâches
6. Conclusion

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Rappels

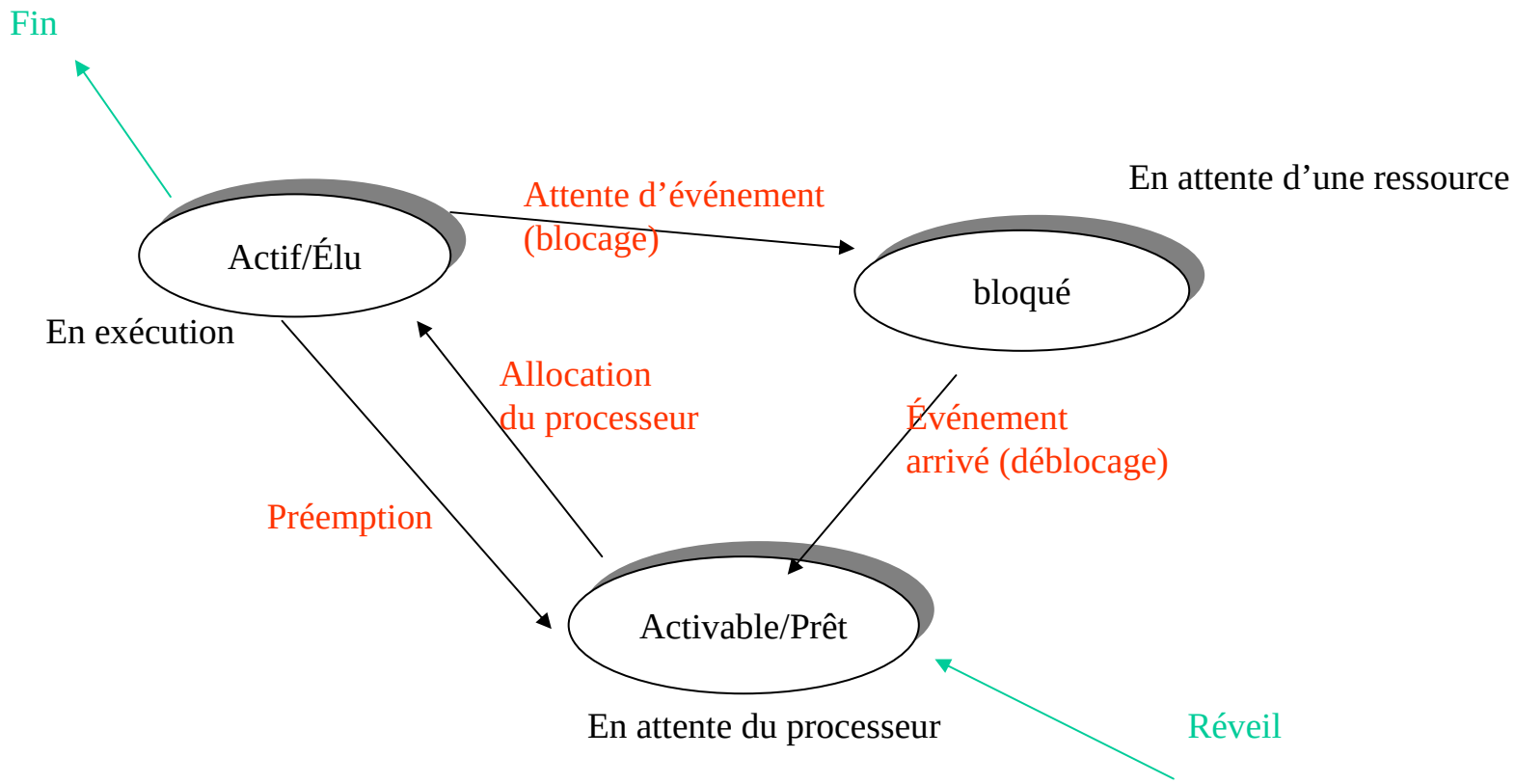
- ***Ordonnancement*** : planification de l'exécution des tâches. Détermine l'ordre d'allocation du processeur.

- ***Tâche*** : entité d'exécution. Instance dynamique d'un programme exécutable. (processus Unix, thread Unix/Java, task Ada)

- ***Ordonnancement préemptif ou non préemptif*** : Un ordonnancement préemptif autorise l'opération de réquisition du processeur au profit d'une tâche plus prioritaire.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

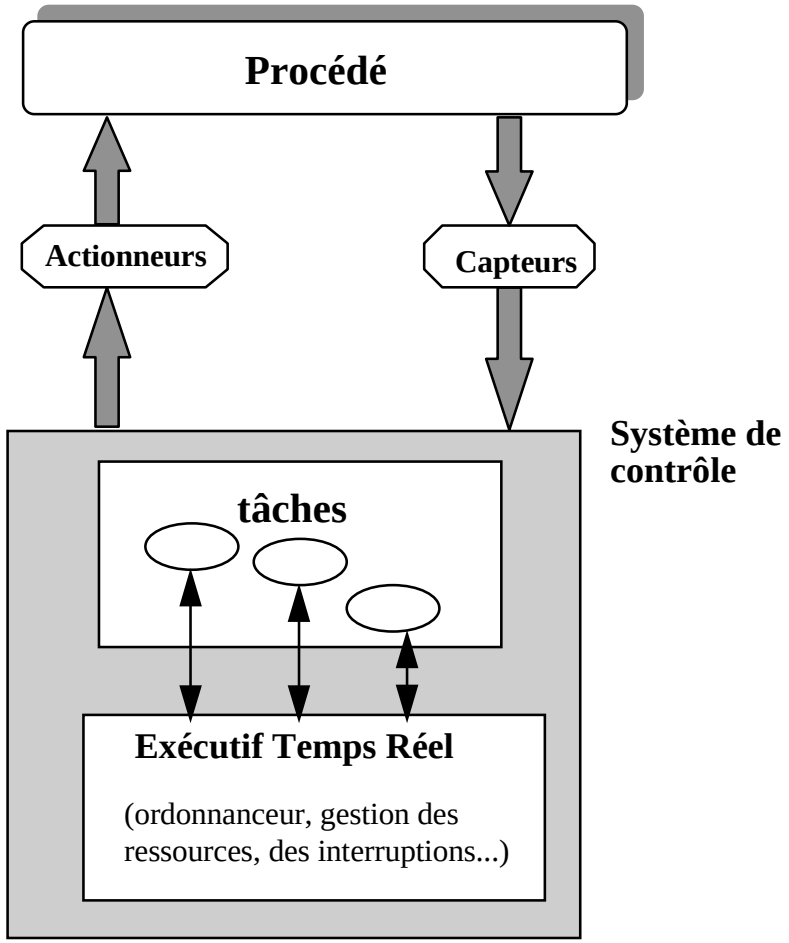
Comportement d'une tâche



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Contexte applicatif

Applications de contrôle
- **Multitâches** avec
- **Concurrence**
- **Communication**
- **Contraintes de temps**
- **Contraintes de précédence** et
- **Partage de ressources**



Structure d'un noyau temps réel

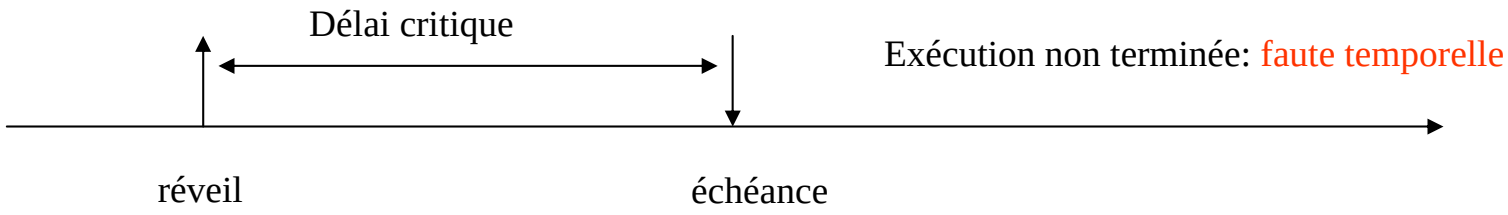
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Ordonnancement temps réel

•***But principal de l'ordonnancement*** : permettre le respect des contraintes temporelles associées à l'application et aux tâches.

•Chaque Tâche possède ***un délai critique***: temps maximal pour s'exécuter depuis sa date de réveil. La date butoir résultante est appelée ***échéance***.

• Le dépassement d'une échéance est appelée ***faute temporelle***.



Certification

•***Applications embarqués et critiques*** : nécessité de certifier l'ordonnancement réalisé, c-à-d de vérifier avant le lancement de l'application (hors ligne) le respect des contraintes temporelles.

•Cette certification s'effectue à l'aide de ***tests d'acceptabilité*** qui prennent en compte les paramètres temporels des tâches (temps d'exécution des tâches).

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Certification

- **Faisabilité** : est-il possible d'exhiber un test de faisabilité ?
  - Condition permettant de décider hors ligne du respect des contraintes des tâches.
  - Prédictibilité du temps de réponse des tâches.

- **Optimalité** : critère de comparaison des algorithmes (un algorithme est dit optimal s'il est capable de trouver un ordonnancement pour tout ensemble faisable de tâches).

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Test d'acceptabilité

- ***Tests d'acceptabilité*** : utilisent les temps d'exécution des tâches
  - ✓Il faut pouvoir déterminer ces temps
  - ✓L'exécutif doit être **déterministe**.

- ***Un exécutif déterministe*** est un exécutif pour lequel les temps de certaines opérations système et matérielles élémentaires peuvent être bornés : temps de commutation, temps de prise en compte des interruptions, etc.

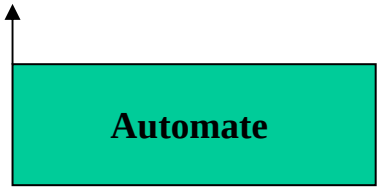
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Ordonnancement hors ligne

•Un ***ordonnancement hors ligne*** établit avant le lancement de l'application une séquence fixe d'exécution des tâches à partir de tous les paramètres de celles-ci. Cette séquence est rangée dans une table et exécutée en ligne par un automate (séquenceur).

t=0	t=5	t=8	t=15	t=30	t=32
tache1	tache3	tache1	tache3	tache5	tache4

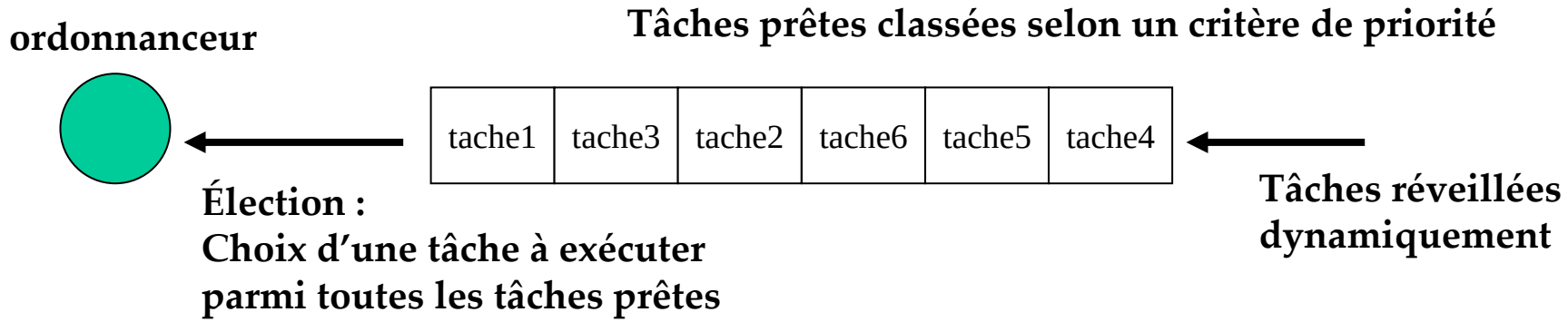
Séquence construite hors ligne



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Ordonnancement en ligne

• ***Ordonnancement en ligne*** : la séquence d'exécution des tâches est établie dynamiquement par l'ordonnanceur au cours de la vie de l'application en fonction des événements qui surviennent. L'ordonnanceur choisit la prochaine tâche à élire en fonction d'un critère de priorité.



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Modélisation de l'application

•***Tâches périodiques*** : elles correspondent aux mesures sur le procédé; elles se réveillent régulièrement (toutes les P unités de temps).

- ✓***Périodiques strictes*** : contraintes temporelles dures à respecter absolument.
- ✓***Périodiques relatives*** : contraintes temporelles molles qui peuvent être ou ou non respectées de temps à autre (sans échéance).
- ✓***Périodiques à échéance sur requête*** : délai critique égal à la période.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Modèle de tâches périodiques strictes

$T(r_0, C, R, P)$

Avec $0 <= C <= R <= P$

r_0 : date de réveil de la tâche

C: durée d 'exécution maximale

R : délai critique

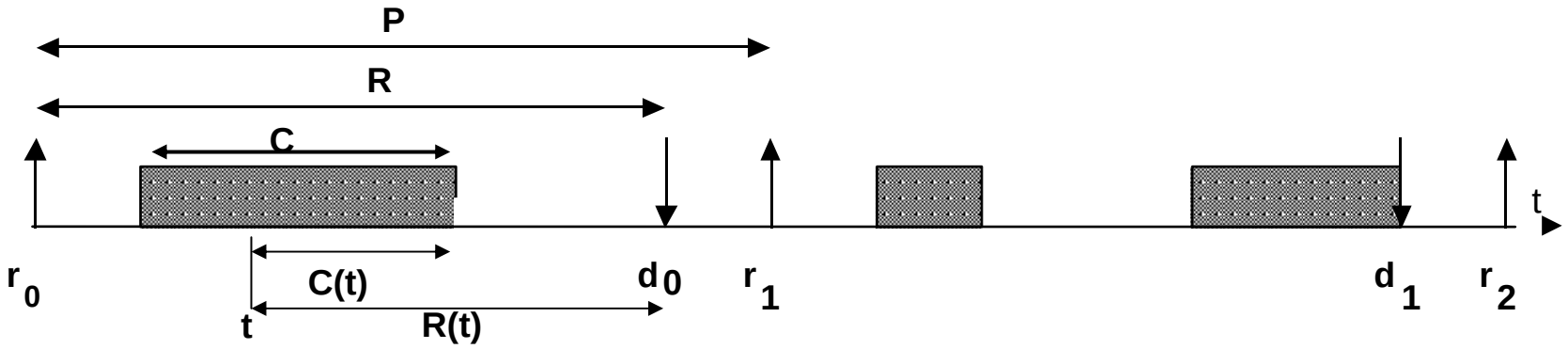
P : période d 'exécution

r_k: date de réveil de la kième instance de la tâche $r_k = r_0 + kP$

d_k: échéance $d_k = r_k + R$

C(t): temps d'exécution restant à t
R(t): délai critique dynamique (temps restant à t jusqu'à d)
Quand R=P : tâche à échéance sur requête

Diagramme temporel d 'exécution



Modèle canonique d'une tâche périodique temps réel

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Modélisation de l'application

•***Tâches apériodiques*** : elles correspondent aux événements; elles se réveillent de manière aléatoire.

- ✓***Apériodiques strictes*** : contraintes temporelles dures à respecter absolument.
- ✓***Apériodiques relatives*** : contraintes temporelles molles qui peuvent être ou ou non respectées de temps à autre (sans échéance).

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Modèle de tâches apériodiques strictes

T_{ap}(r, C, R)

T_{ap} (t, C(t), R(t))

r: date aléatoire de réveil de la tâche

C: durée d 'exécution maximale

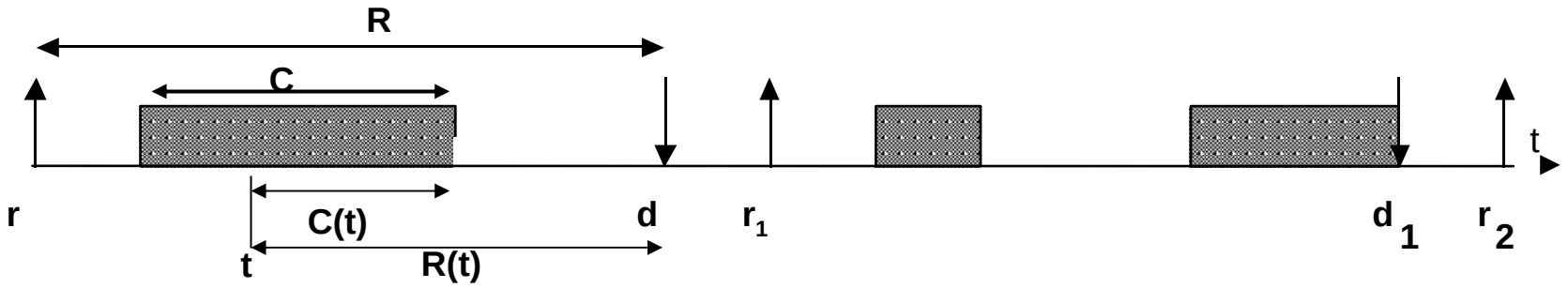
R : délai critique

d_k : échéance = r_k + R

C(t): temps d'exécution restant à t

R(t): délai critique dynamique (temps restant à t jusqu'à d)

Diagramme temporel d 'exécution



Modèle canonique d'une tâche apériodique temps réel

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Algorithmes d'ordonnancement pour les tâches périodiques

- ***Algorithmes en ligne et préemptifs avec un test d'acceptabilité évaluable hors ligne***

  ✓Nous ordonnançons un ensemble de tâches périodiques (configuration).
  Les priorités affectées aux tâches sont soit **constantes** (évaluées hors ligne et fixes par la suite), soit **dynamiques** (elles changent durant la vie de la tâche).

  ✓L'ordonnancement d'un ensemble de tâches périodiques est cyclique et la séquence se répète de manière similaire sur ce que l'on appelle la période d'étude.

  ✓Pour un ensemble de tâches à départ simultané (t=0), la période d'étude est : $[0, PPCM(P_i)]$.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Période d'étude : cas général

Durée de validation

Durée = {*Début*, *Fin*}

Début = Min(r_i) si r_i est la date de première activation de toute tâche T_i

Fin = Max(r_i) + PPCM(P_i) si toutes les tâches sont périodiques

ou

Fin = Max((r_i +R_i)$_{apériodiques}$, (r_i)$_{périodiques}$) + 2 PPCM(P_i)$_{périodiques}$ s'il y a des tâches apériodiques [LM 80]

Si la séquence est valide sur l'intervalle [*Début*, *Fin*], alors elle est valide sur un temps infini [LM 80].

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Algorithme d'ordonnancement Rate Monotonic

- *Algorithme Rate Monotonic (RM)*

  ✓Priorité de la tâche est fonction de sa période. Priorité constante.

  ✓La tâche de plus petite période est la tâche la plus prioritaire

  ✓Pour un ensemble de *n* tâches périodiques à échéance sur requête T_{pi} (r_0, C_i, R_i, P_i), un test d'acceptabilité est:

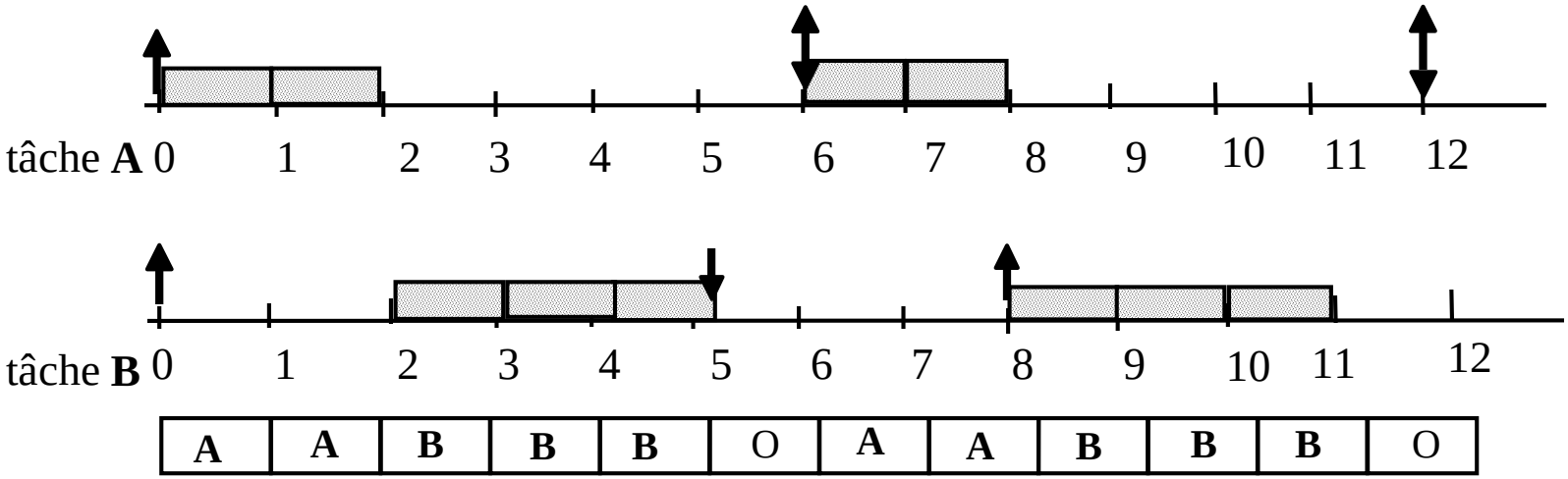
Si condition vérifiée : $\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{1/n} - 1)$ alors la configuration est ordonnançable

Condition suffisante

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Exemple avec Rate Monotonic

Tâche A (r_0=0, C=2, R=6, P=6)
Tâche B (r_0=0, C=3, R=5, P=8)



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Algorithme d'ordonnancement Inverse Deadline

- *Algorithme Inverse Deadline (ou Deadline Monotonic (DM)*)

  ✓Priorité de la tâche est fonction de son délai critique. Priorité constante.

  ✓La tâche de plus petit délai critique est la tâche la plus prioritaire

  ✓Pour un ensemble de *n* tâches périodiques à échéance sur requête T_{pi} (r_0, C_i, R_i, P_i), un test d'acceptabilité est:

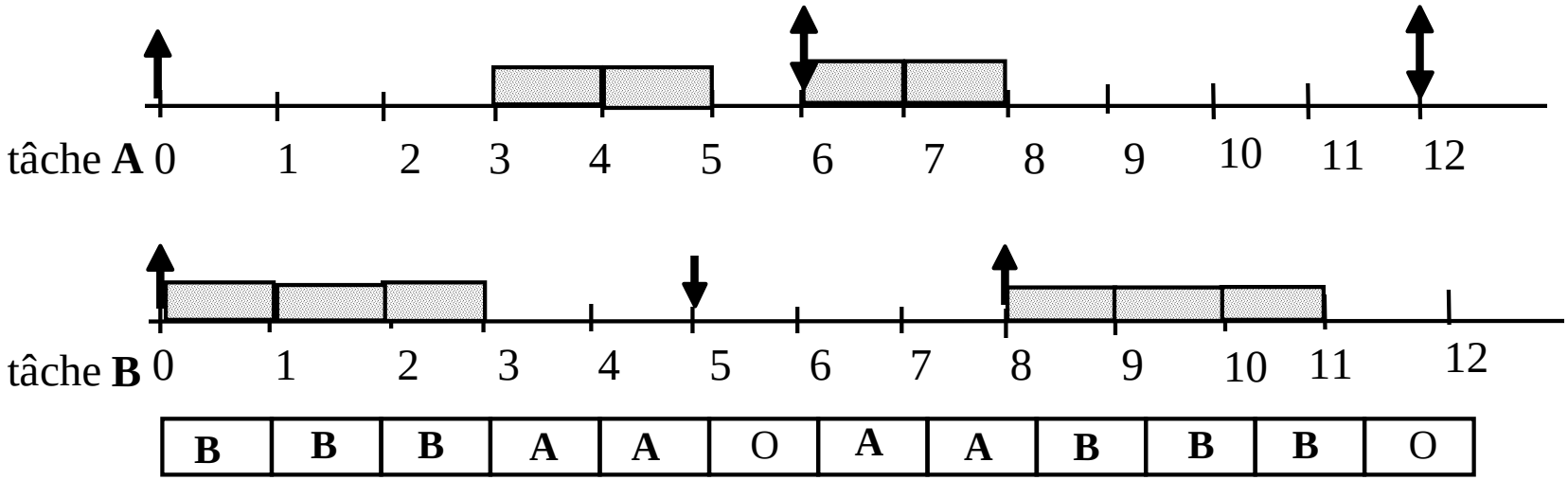
Si condition vérifiée : $\sum_{i=1}^{n} \frac{C_i}{R_i} \leq n(2^{1/n} - 1)$ alors la configuration est ordonnançable

Condition suffisante

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Exemple avec Inverse Deadline

Tâche A (r_0=0, C=2, R=6, P=6)
Tâche B (r_0=0, C=3, R=5, P=8)



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Algorithme d'ordonnancement Earliest Deadline/1

- *Algorithme Earliest Deadline (ou Earliest Deadline First (EDF)*)

  ✓Priorité de la tâche est fonction de son délai critique dynamique. Priorité dynamique.

  ✓A l'instant t, la tâche de plus petit délai critique dynamique (de plus proche échéance) est la tâche la plus prioritaire

  ✓Pour un ensemble de *n* tâches périodiques à échéance sur requête T_{pi} (r_i, C_i, R_i, P_i), un test d'acceptabilité est:

La configuration est ordonnançable **ssi** $\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$

Condition nécessaire et suffisante

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Algorithme d'ordonnancement Earliest Deadline/2

- *Algorithme Earliest Deadline (ou Earliest Deadline First (EDF)*)

✓Pour un ensemble de *n* tâches périodiques quelconques T_{pi} (r_i, C_i, R_i, P_i), un test d'acceptabilité est:

$$\sum_{i=1}^{n} \frac{C_i}{R_i} \leq 1$$ => la configuration est ordonnançable - **Condition suffisante**

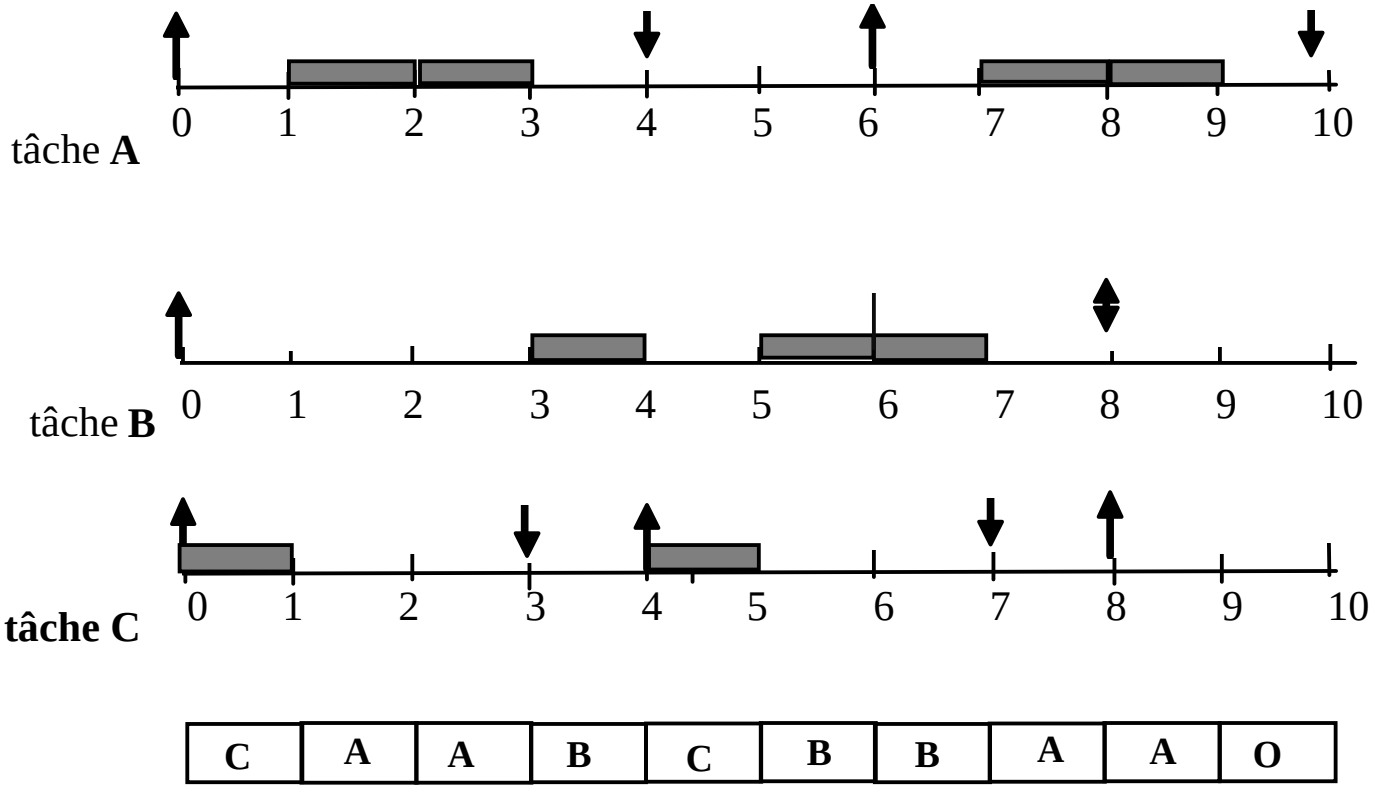
✓Pour un ensemble de *n* tâches périodiques quelconques T_{pi} (r_i, C_i, R_i, P_i), un test d'acceptabilité est:

Si la configuration est ordonnançable **alors** $$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$$ **- Condition nécessaire**

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Exemple avec Earliest Deadline

Tâche A (r_0=0, C=2, R=4, P=6)
Tâche B (r_0=0, C=3, R=8, P=8)
Tâche C (r_0=0, C=1, R=3, P=4)

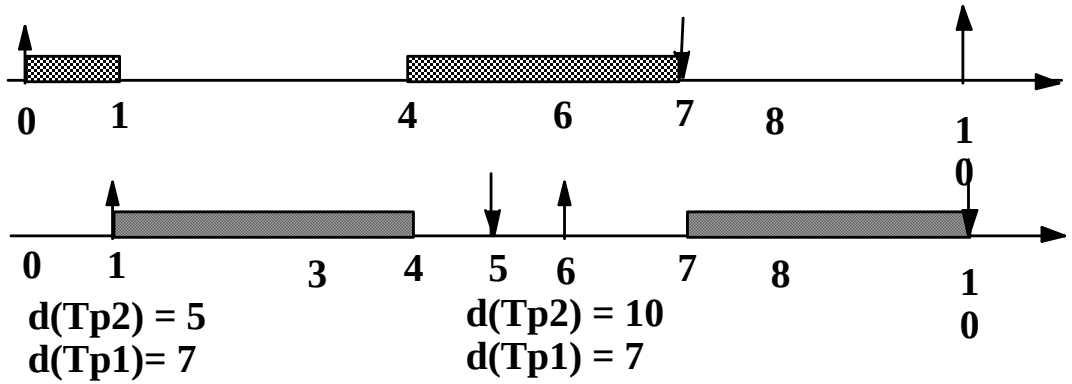


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Autre exemple avec Earliest Deadline

Tp1(r0=0, C=4, R = 7, P=10)

Tp2(r0=1, C=3, R= 4, P=5)



Réveil Echéance

ED est un algorithme optimal

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Ordonnancement de tâches apériodiques

Ordonnancement de tâches apériodiques

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Contexte du problème

- *Application de contrôle multi-tâche*
  - ✓Tâches périodiques T_{pi} (r_i, C_i, R_i, P_i)
  - ✓Tâches apériodiques T_{api} (r_i, C_i, R_i)
- *Risque de surcharge*
- *Prise en compte des tâches apériodiques au sein d'une configuration périodique ordonnançable*

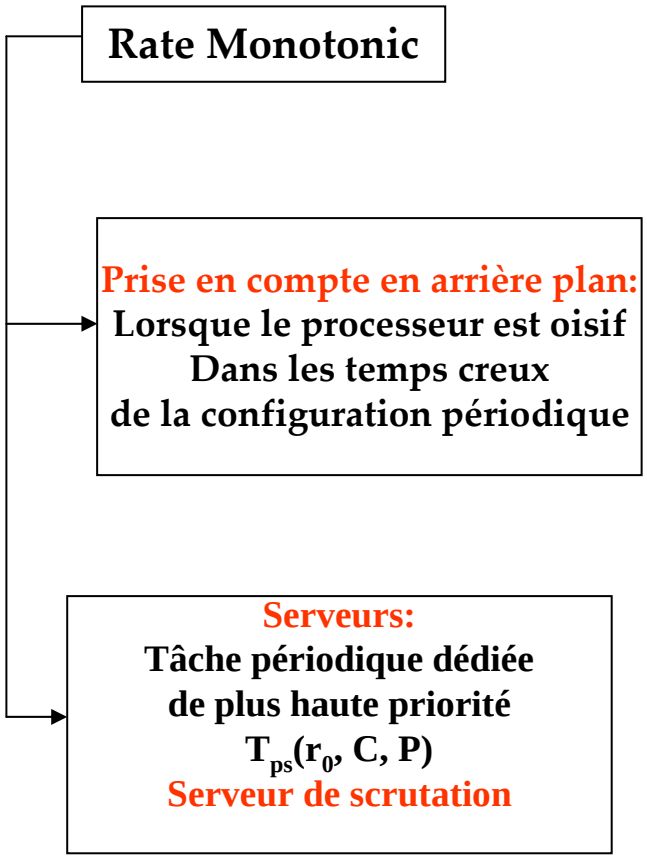
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Critères de classification des méthodes

- *Algorithme d'ordonnancement des tâches périodiques*
- *Criticité des tâches apériodiques*
- *Niveau de prise en compte des tâches apériodiques / tâches périodiques*

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Tâches apériodiques à contraintes relatives



- **Premier Groupe**
- ✓Tâches périodiques : Rate Monotonic
- ✓Tâches apériodiques à contraintes relatives
- ✓Minimiser les temps de réponse des tâches apériodiques

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Traitement d'arrière plan

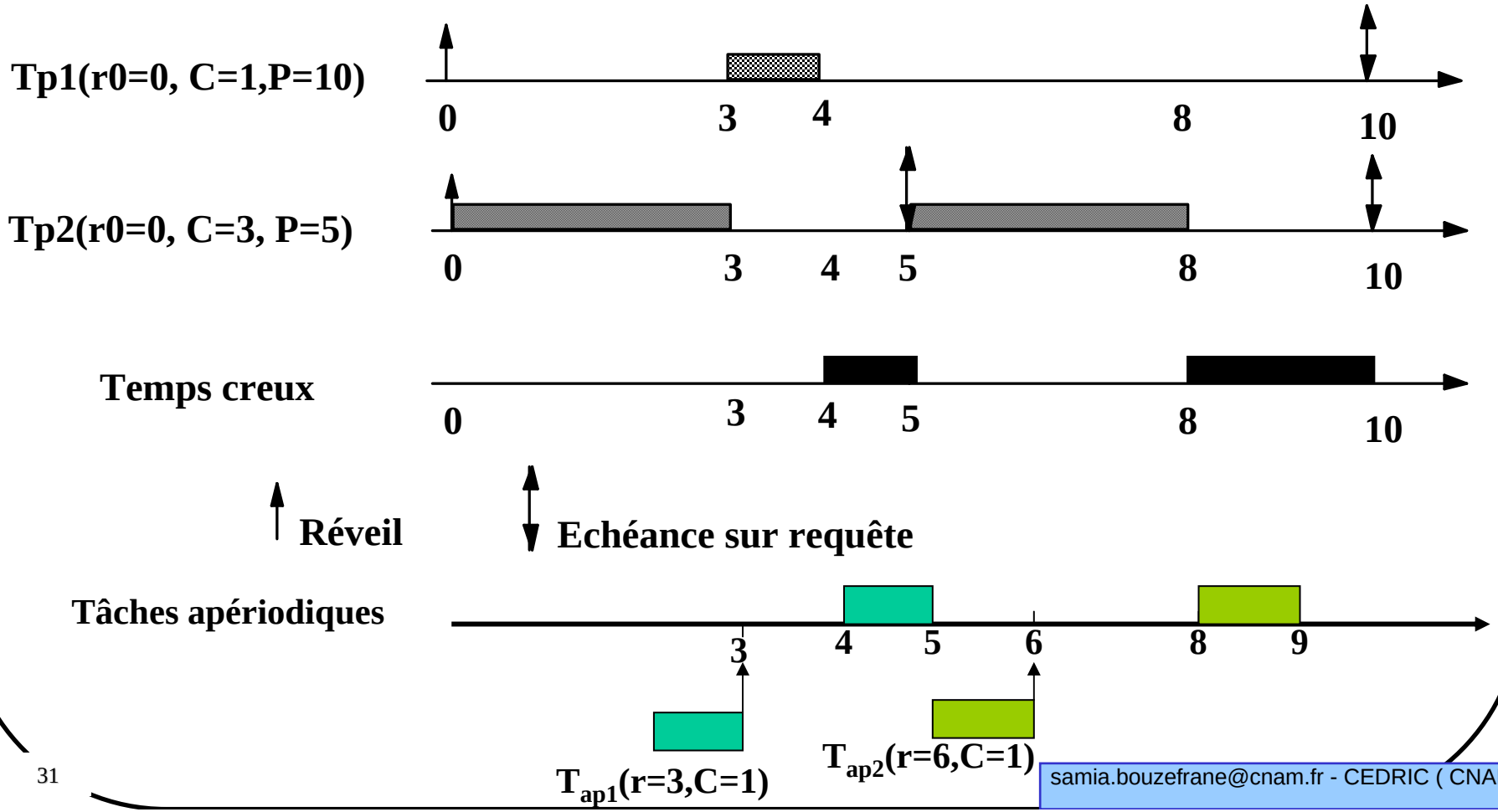
- ***Les tâches apériodiques sont ordonnancées selon un ordre FIFO lorsque le processeur est oisif***

✓Simplicité, faible coût

✓Le temps de réponse des apériodiques croit avec la charge périodique

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Exemple avec un traitement en arrière plan



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Serveur de tâches apériodiques

- ***Une tâche périodique de haute priorité est dédiée au service des tâches apériodiques en attente: T_{ps} (r_0, C, P)***

- ***Serveur de scrutation :***

  ✓A chacun de ses réveils, le serveur sert les tâches apériodiques en attente depuis son réveil précédent

  - ***jusqu'à épuisement des tâches apériodiques***
  - ***jusqu'à épuisement de sa capacité***

  ✓L'ordre de service des tâches apériodiques est quelconque

  ✓La capacité non utilisée est perdue

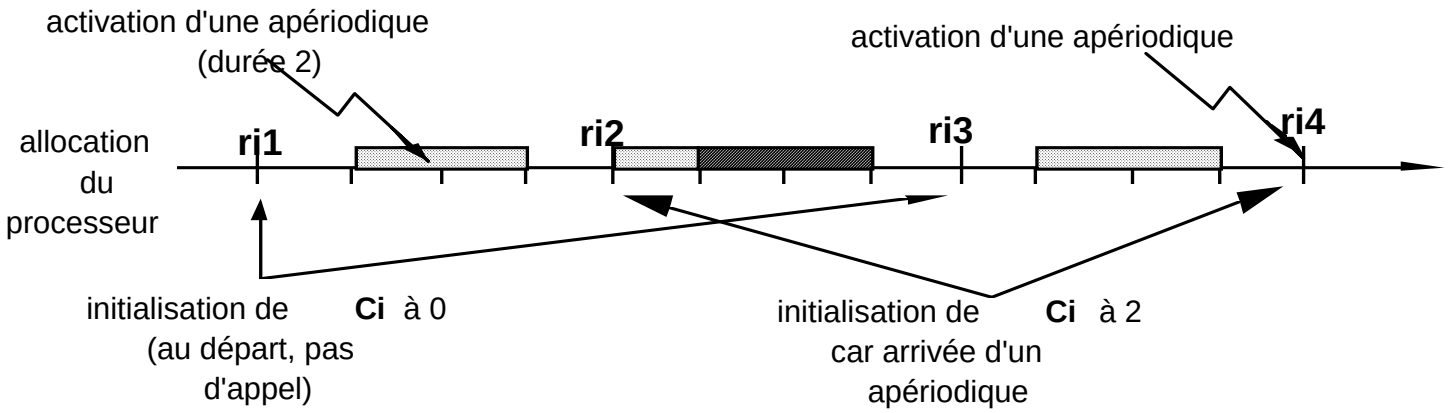
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Serveur à scrutation

Processeur utilisé pour d'autres tâches

Processeur utilisé par le serveur, **Ci** affecté à un processus aléatoire

rik : dates d'activation du serveur
Ci : capacité du serveur (égal à 2)



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

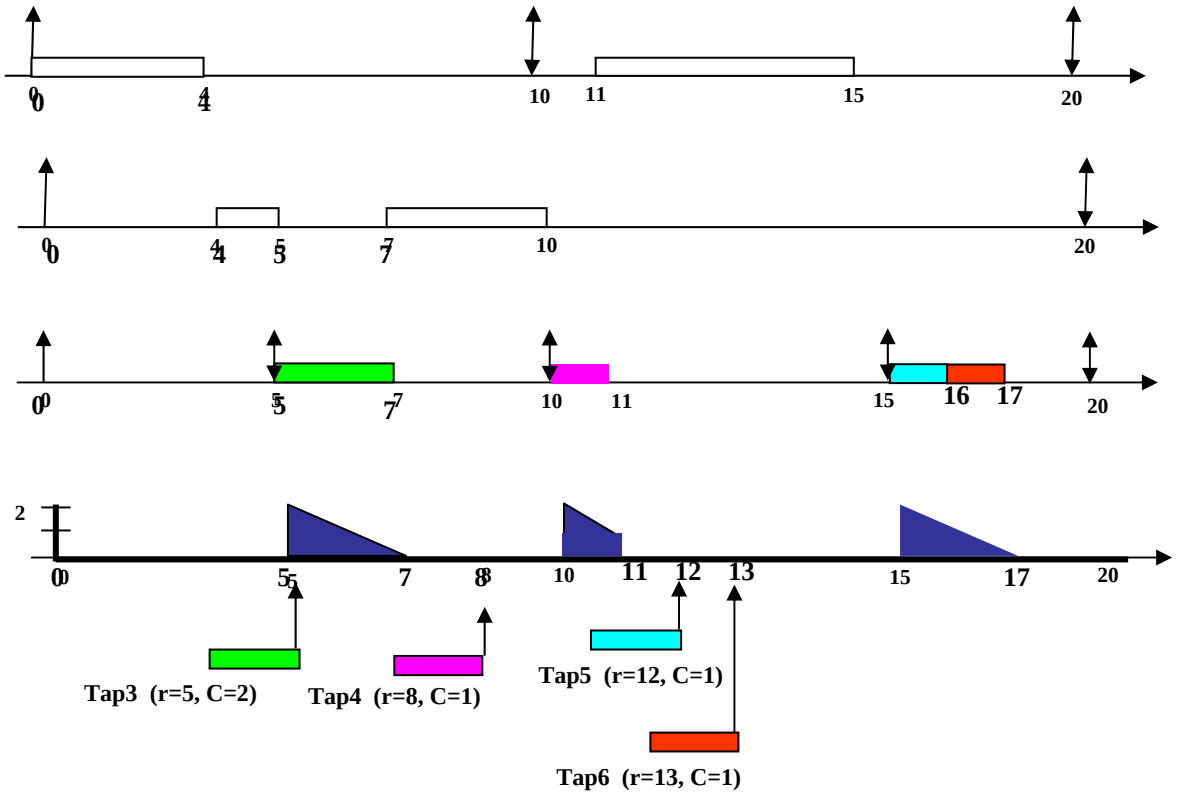
Exemple avec un serveur de scrutation

T_{P1} (r_0=0, C=4, P=10)

T_{P2} (r_0=0, C=4, P=20)

Tâches apériodiques
T_{ps} (r_0=0, C=2, P=5)
(Service FIFO)

Capacité du serveur



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Inconvénients du serveur de scrutation

- *Inadéquation entre le rythme périodique du serveur T_{ps} et le caractère aléatoire des réveils apériodiques*

- *Modifier la gestion de la capacité du serveur tel que :*

  ✓Le serveur dispose au plus vite de capacité pour servir une nouvelle tâche apériodique

  - serveur sporadique

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

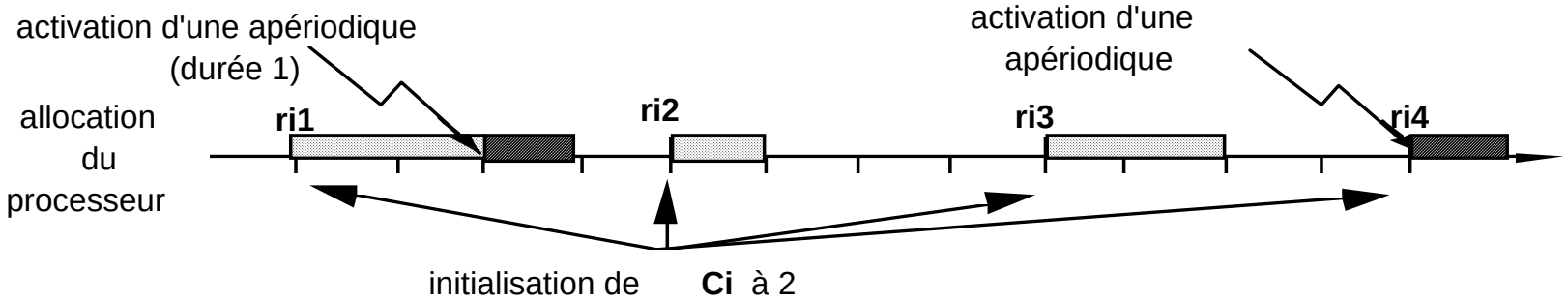
Serveur sporadique

Processeur utilisé pour d'autres tâches

Processeur utilisé par le serveur, **Ci** affecté à un processus aléatoire

rik : dates d'activation du serveur

Ci : capacité du serveur (égal à 2)



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

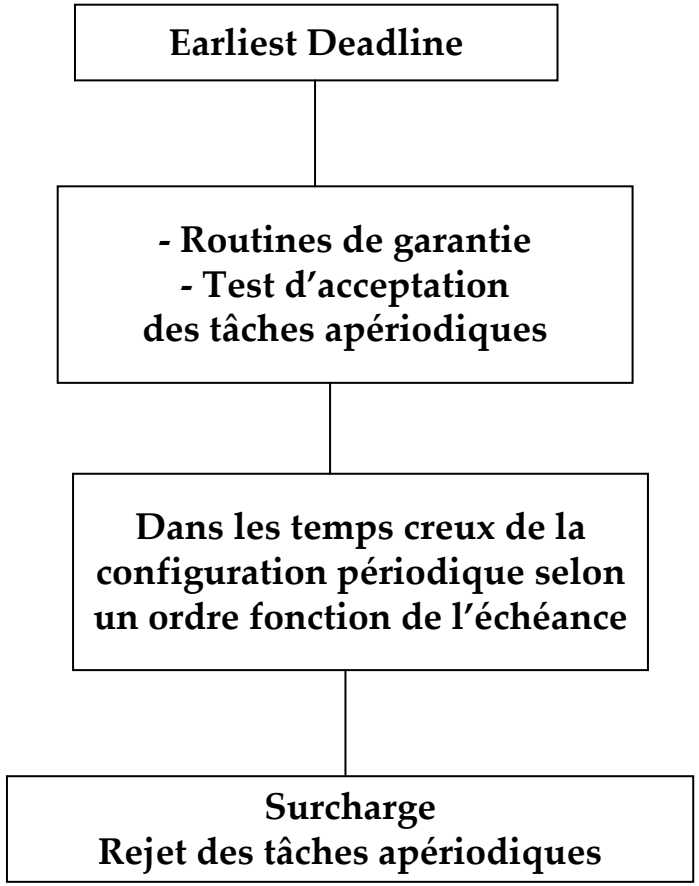
Tâches apériodiques à contraintes strictes

- **Deuxième groupe**

✓Tâches périodiques :
Earliest Deadline

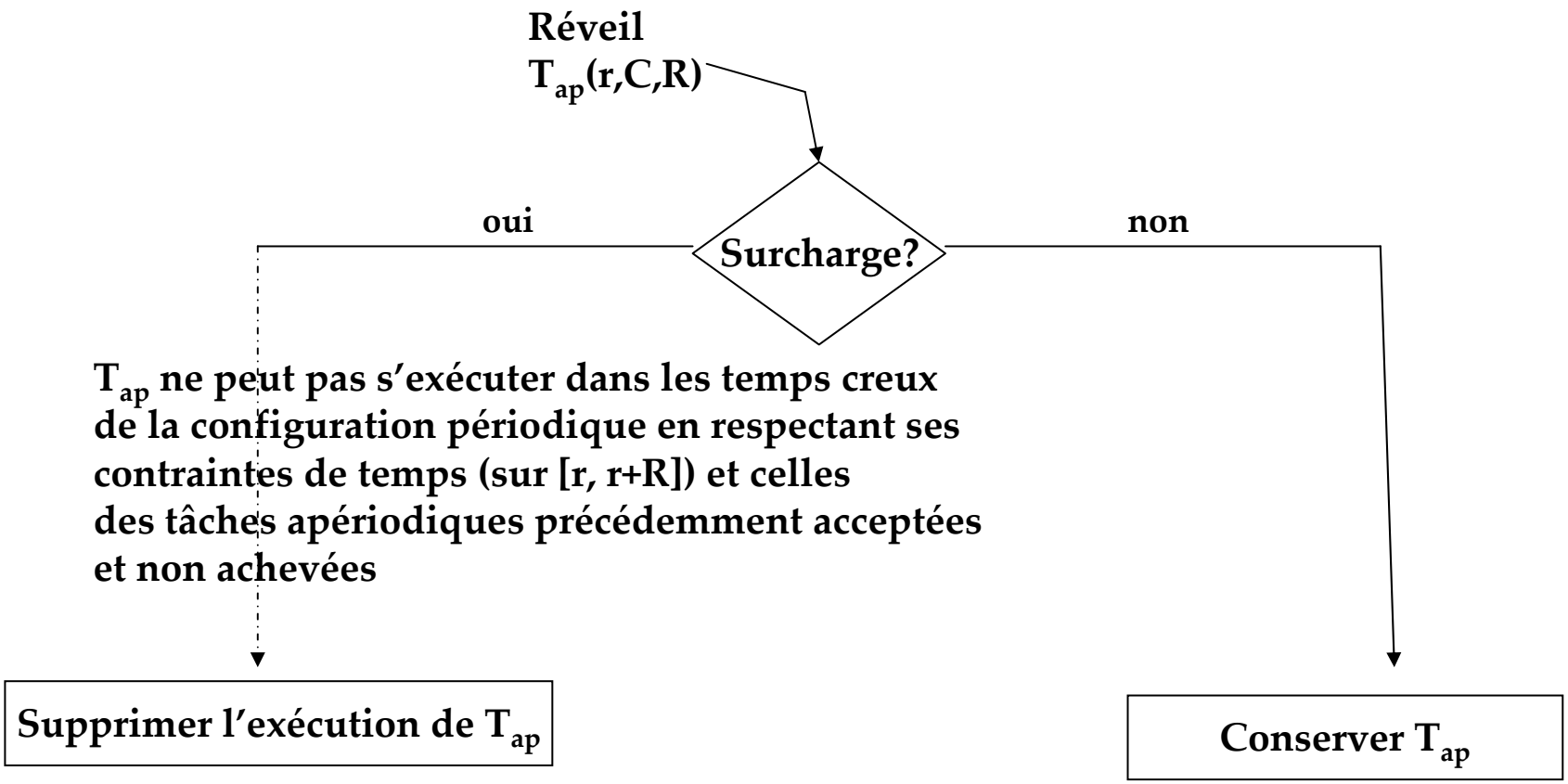
✓Tâches apériodiques:
à contraintes strictes

✓Maximiser le nombre de tâches apériodiques s'exécutant dans le respect de leurs contraintes temporelles



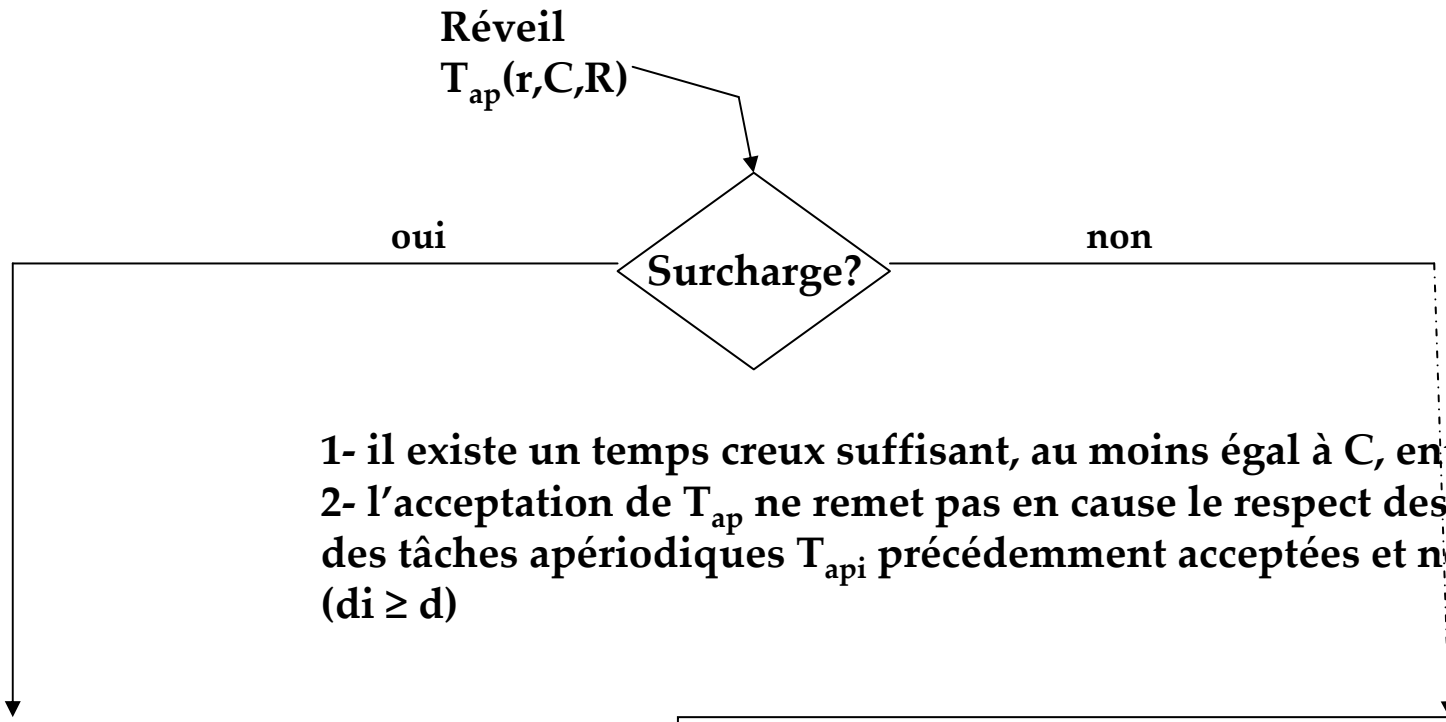
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Routines de garantie: méthode des temps creux



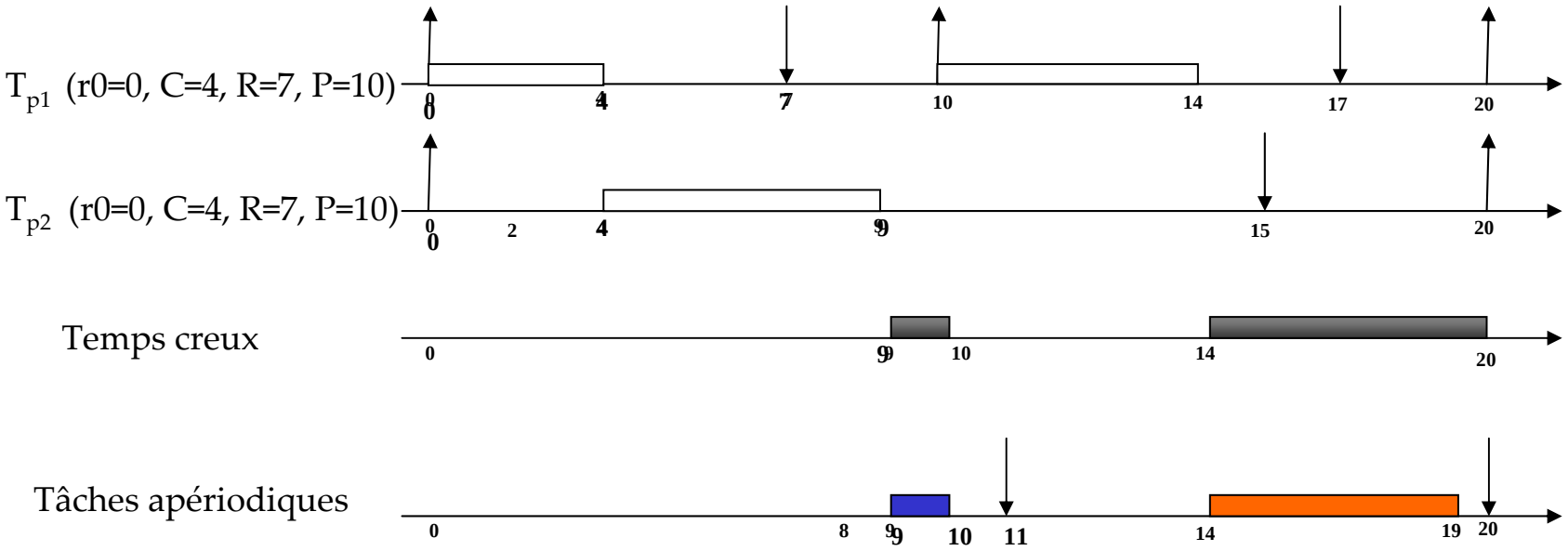
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Routines de garantie: méthode des temps creux



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Méthode des temps creux : exemple

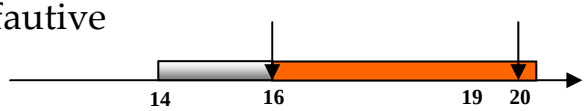


Tap3 (r=5, C=2, R=3) **refusée** : pas de temps creux entre les instants [5,8]

Tap4 (r=8, C=1, R=3) **acceptée** : temps creux suffisant entre les instants [8,11]

Tap5 (r=12, C=5, R=8) **acceptée** : temps creux suffisant entre les instants [14,20]

Tap6 (r=13, C=2, R=3) **refusée** : temps creux suffisant entre les instants [13,16], mais Tap5 fautive



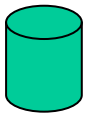
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Ordonnancement avec contraintes de ressources

Ordonnancement avec contraintes de ressources

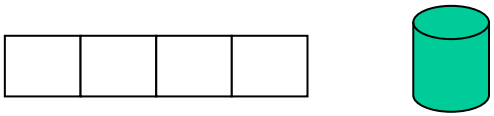
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Modèle de ressources



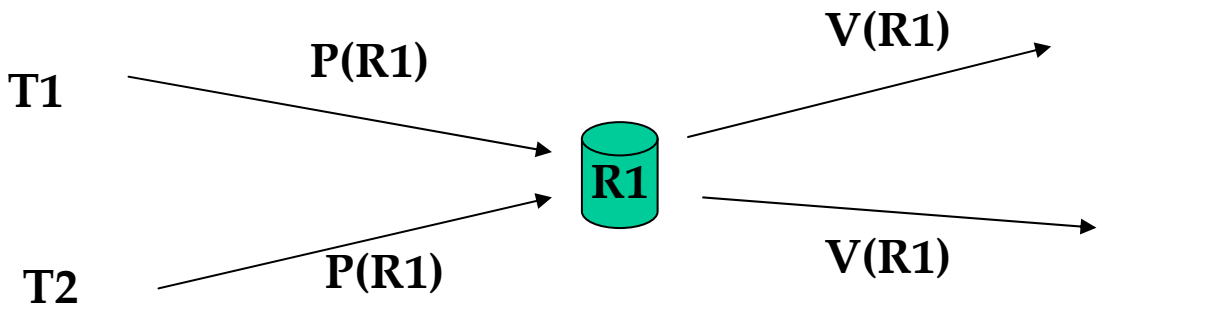
Ressource partageable à capacité d'entrée N
N=1 =>ressource critique

Mise en attente, état bloqué



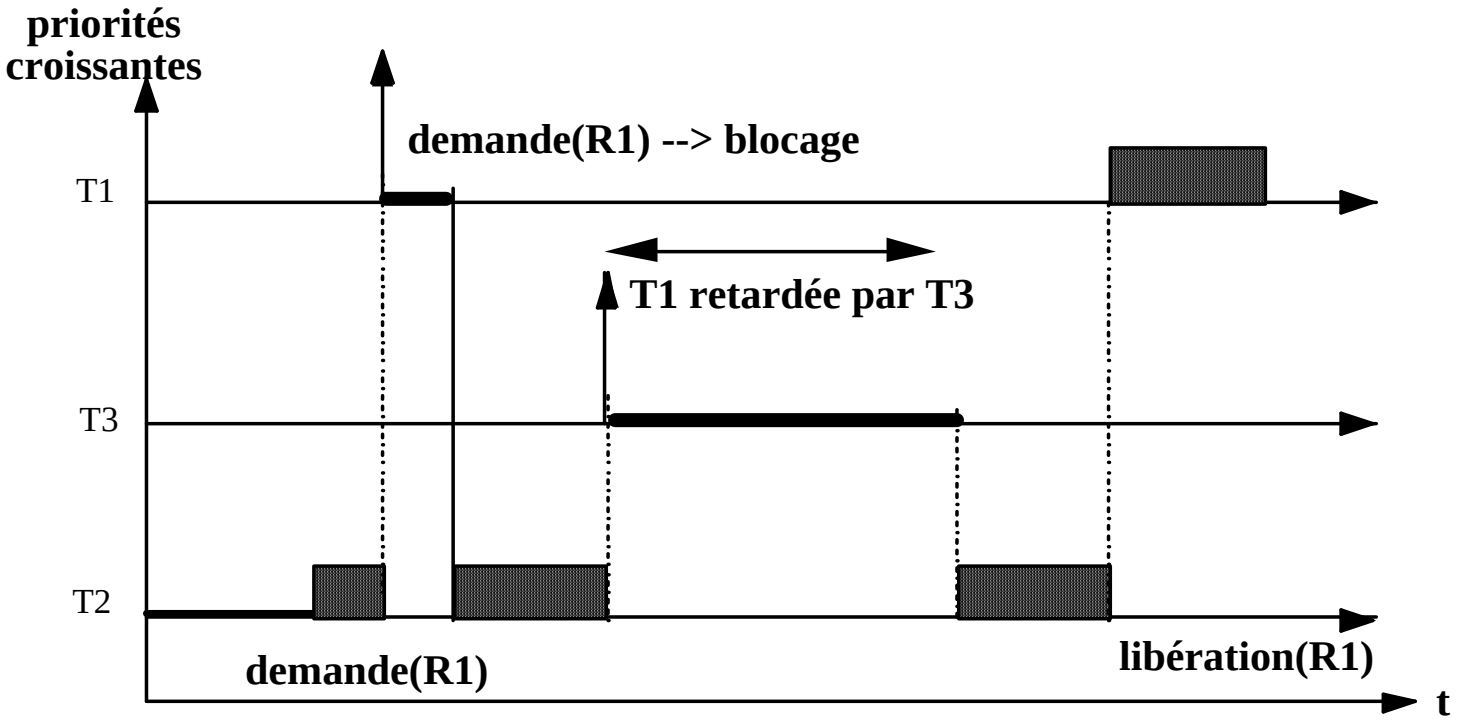
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Problème de l'inversion de priorité



Priorité(T1) > Priorité(T3) > Priorité(T2)

Problème de l'inversion de priorité: exemple



T3 se termine avant T1, bien que T1 soit plus prioritaire
T1 est retardée par toutes les tâches de priorité intermédiaire

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Problème de l'inversion de priorité

- ***La tâche T3 de priorité intermédiaire qui n'utilise pas la ressource R1 retarde la tâche T2 et donc la tâche T1***

  ✓Il peut y avoir une infinité de tâches T3

  ✓L'attente de T1 ne peut pas être bornée

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Problème de l'inversion de priorité

- *Pouvoir borner l'attente de T1 et inclure cette borne au test d'acceptabilité de la configuration à ordonnancer*

✓Test d'acceptabilité de la configuration de tâches périodiques

Test classique + facteur de blocage B

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Problème de l'interblocage

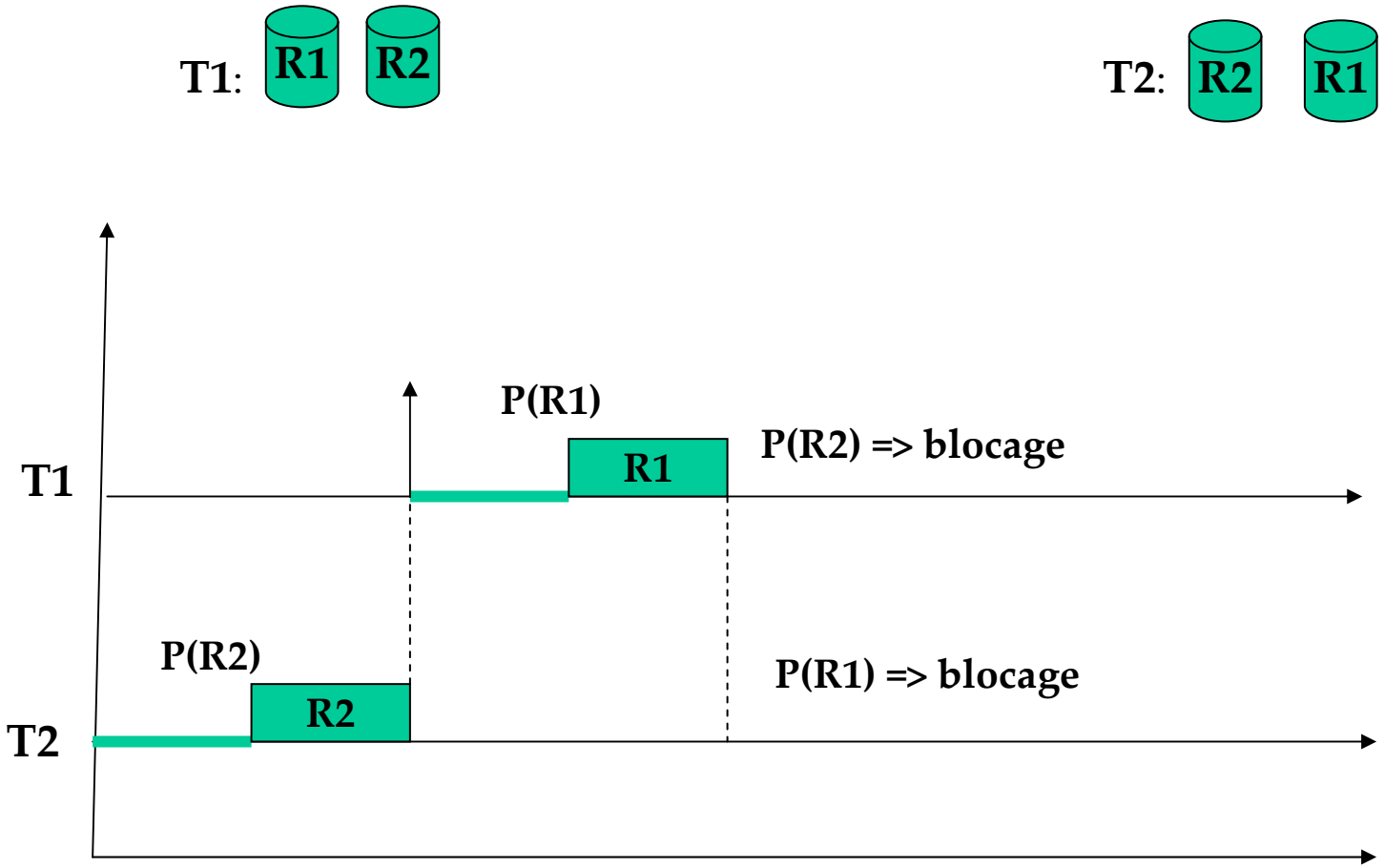
- ***Interblocage***

  Ensemble de *n* processus attendant chacun un événement ne pouvant être produit que par un autre processus de l'ensemble

  ✓Attente infinie

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Problème de l'interblocage: exemple



Priorité(T1)> Priorité(T2)

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Problème de l'inversion de priorité

- *Pouvoir borner l'attente de T1 et inclure cette borne au test d'acceptabilité de la configuration à ordonnancer*

- *Éviter les situations d'interblocage*

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

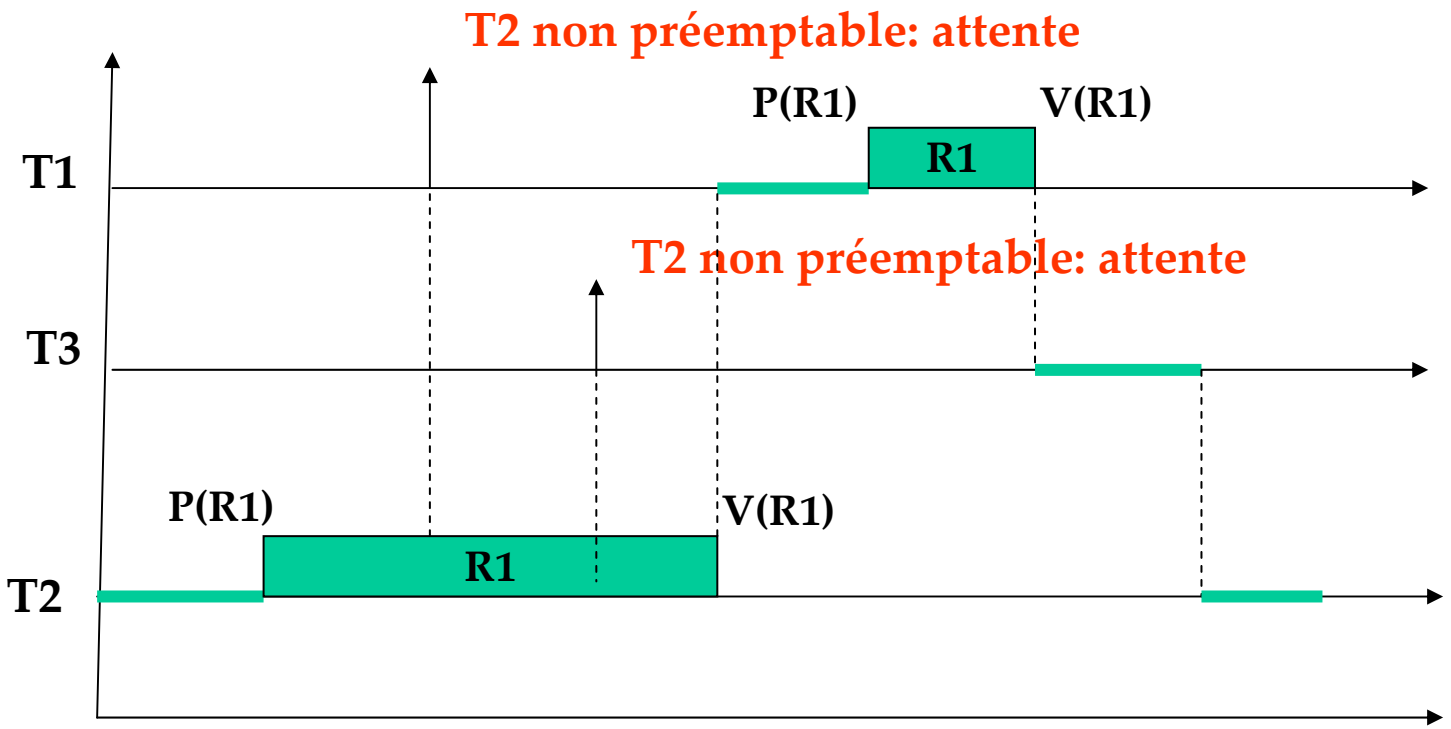
Solutions à l'inversion de priorité

- ***Triviale :*** rendre une section critique non interruptible
- ***Protocole de l'héritage de priorité*** [Sha 1990]
- ***Protocole de la priorité plafond*** [Sha 1990]
- ***Protocole à pile*** [Bak 91]: ne sera pas traité ici, décrit dans thèse [Thèse_SB 1998]

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Section critique non interruptible

Priorité(T1)> Priorité(T3)>Priorité(T2)



Une tâche en section critique ne peut être préemptée par une autre tâche

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Section critique non interruptible

- *Section critique terminée au plus tôt, T1 retardée au plus de la durée de la section critique*

- *MAIS*

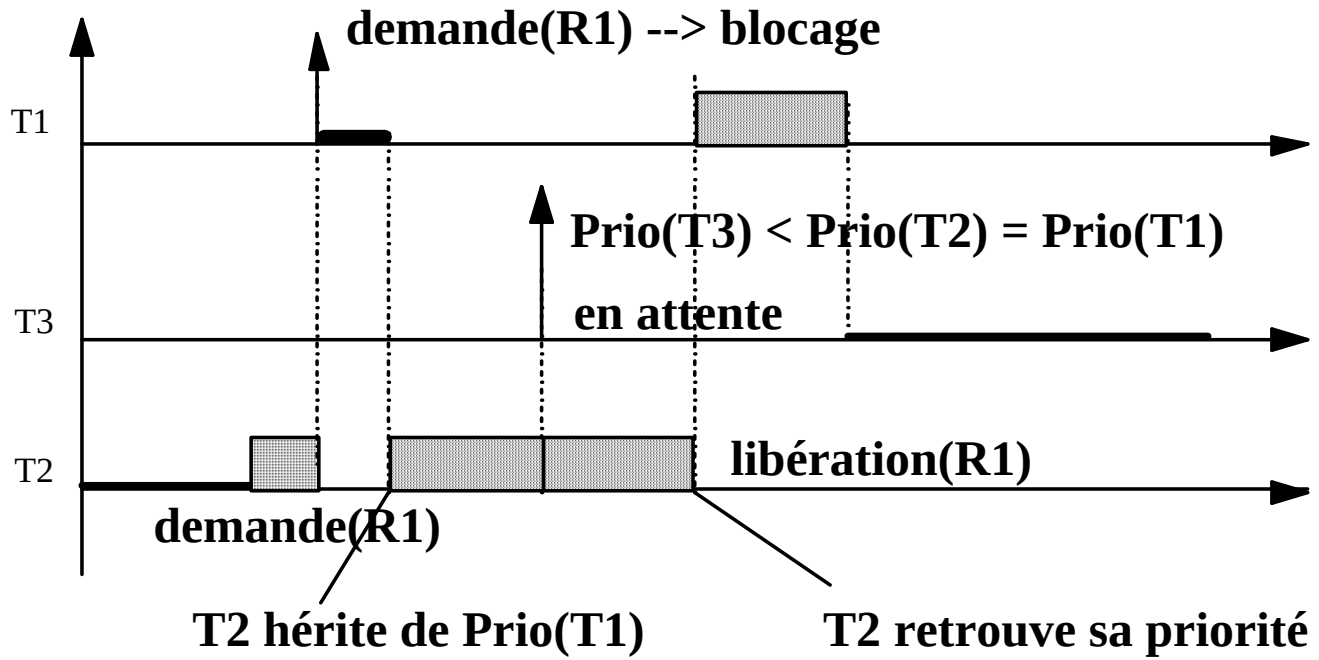
  -Difficile dans le cas d'une section critique longue

  -Blocage des processus de haute priorité n'utilisant pas la section critique

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Protocole de l'héritage de priorité

- *Une tâche en section critique hérite de la priorité de la plus haute tâche en attente sur la section critique*



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Protocole de l'héritage de priorité

• On montre que la borne ***B*** (temps de blocage maximal pour une tâche sur l'attente d'une ressource) est égale au plus à la somme des durées des sections critiques partagées avec des tâches de plus faible priorité.

Sous Rate Monotonic, le test d'acceptabilité de n tâches périodiques devient :

$$\sum_{k=1}^{i} \frac{C_k}{P_k} + \frac{B_i}{P_i} \leq i(2^{1/i} - 1)$$ Pour tout i allant de 1 à n

• ***MAIS***

- ne prévient pas les interblocages

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Protocole de la priorité plafond

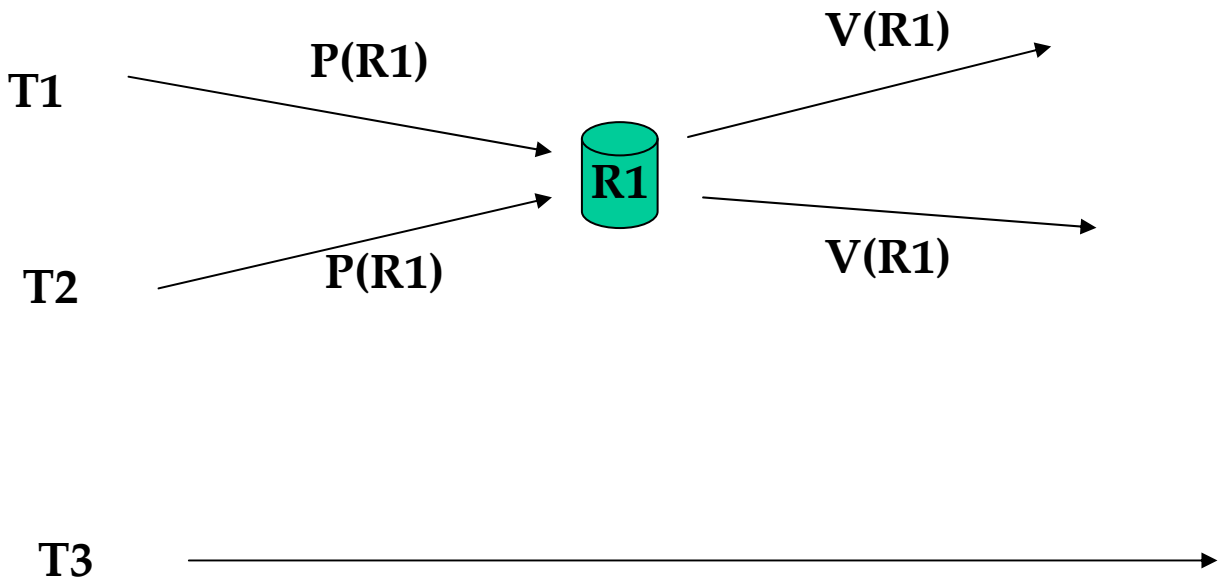
- *Une ressource critique S reçoit une priorité telle que :*
  Priorité(S)=Max(Priorité(T_i), T_i accède à S)

- *Une tâche qui accède à S prend la priorité de S si une tâche plus prioritaire qu'elle est mise en attente.*

- *Une tâche ne peut accéder à S que si sa priorité est supérieure à toutes les priorités des ressources S' détenues par les autres tâches.*

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Protocole de la priorité plafond



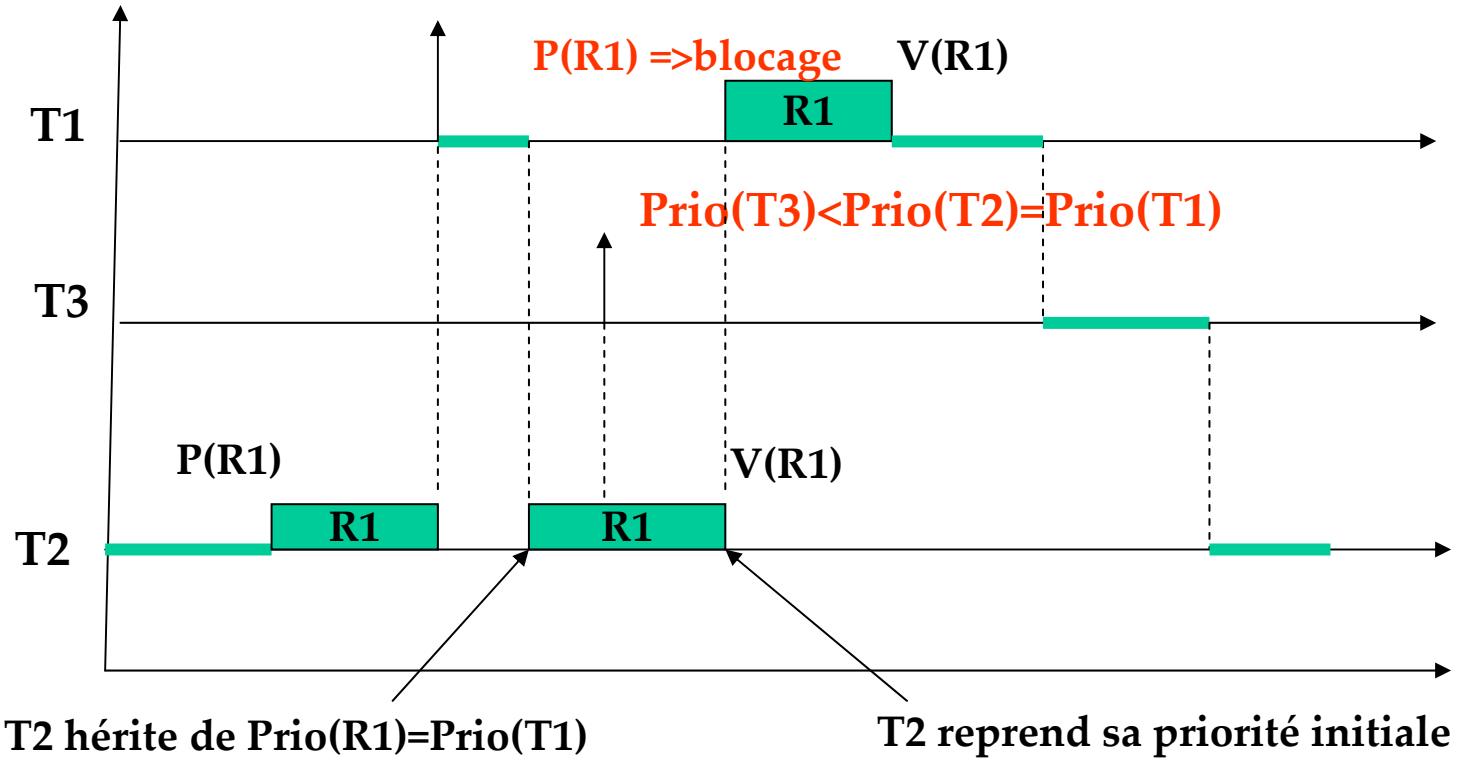
Priorité(T1) > Priorité(T3) > Priorité(T2)

Priorité(R1)=Priorité(T1)

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Protocole à priorité plafond

Priorité(T1)> Priorité(T3)>Priorité(T2)
Priorité(R1)=Priorité(T1)



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Protocole à priorité plafond

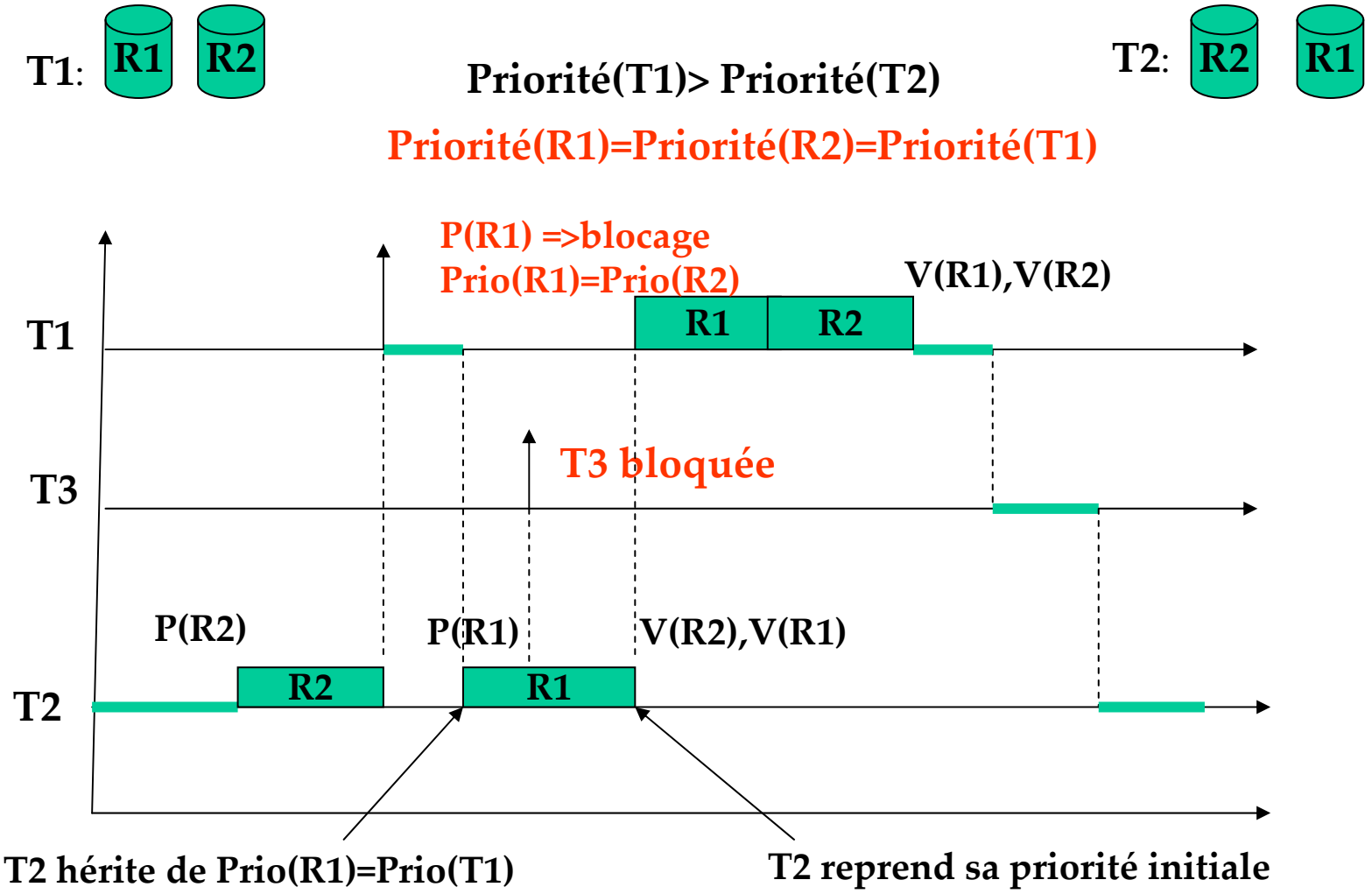
- *Une tâche ne peut être bloquée que pendant la durée d'une section critique d'une tâche de plus faible priorité :*

  Durée_blocage= Max(durée_SC_i) telle que $SC_i \in T_i$ et Priorité(Ti) est minimale

- *Prévention des interblocages.*

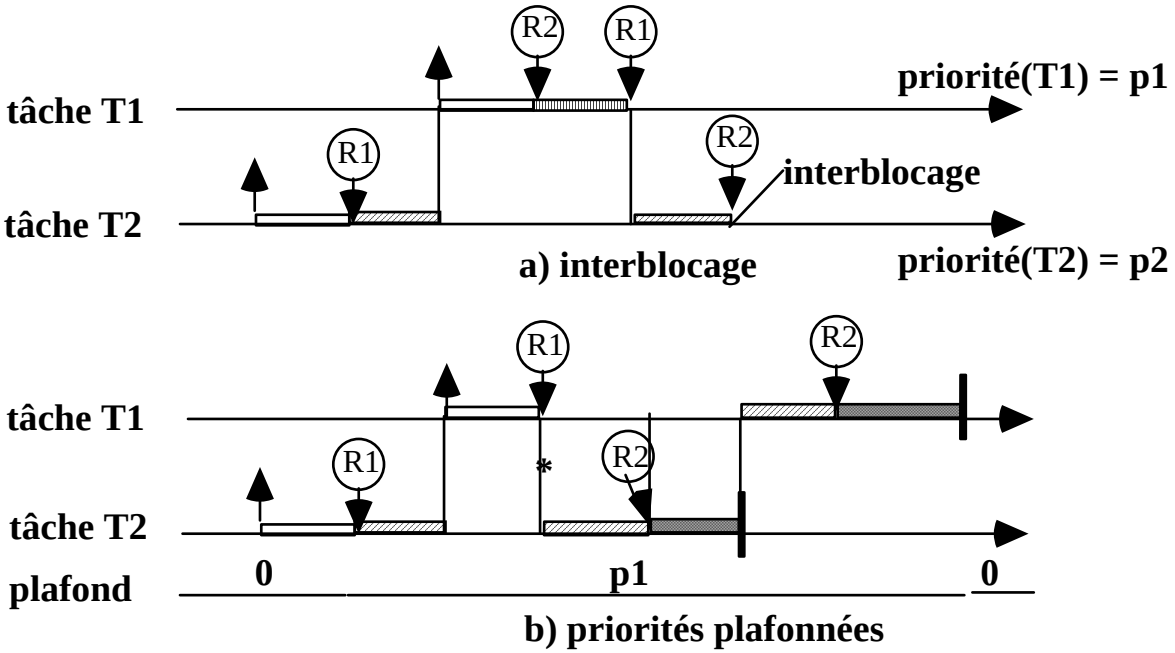
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Priorité plafonnée: exemple1



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Priorité plafonnée: exemple 2



Légende

- tâche élue
- tâche élue utilisant R1
- tâche élue utilisant R2
- tâche élue utilisant R1 et R2
- tâche terminée

p1 > p2

priorité(R1) = priorité(R2) = p1

*** T1 est bloquée car :**
priorité(T1) = p1 = plafond
donc T2 hérite de p1

Solution de l'interblocage et de l'inversion de priorité : priorité plafonnée

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Problème de l'inversion de priorité

Protocole	Ordonnancement	Propriétés
Héritage de priorité	RM	-test d'ordonnançabilité -pas de prévention d'interblocage
Priorité plafonnée	RM	-test d'ordonnançabilité -prévention d'interblocage
Priorité plafonnée dynamique	EDF	-test d'ordonnançabilité -prévention d'interblocage
Protocole à pile	RM EDF	-test d'ordonnançabilité -prévention d'interblocage

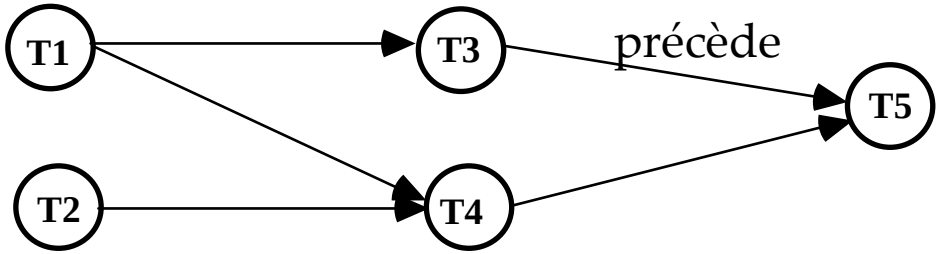
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Ordonnancement avec contraintes de précédence

Ordonnancement avec contraintes de précédence

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Tâches avec contraintes de précédence



- Graphe de précédence entre tâches d'une application
- Prise en compte des contraintes de précédence
- Modifications des paramètres temporels et affectation des priorités

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Prise en compte des contraintes de précédence

- Transformer la configuration de tâches en une configuration de tâches indépendantes

- Travaux de [Chetto 1990] en se basant sur l'algorithme EDF

- Généralisation aux algorithmes RM et DM par [Babau 1996]

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Prise en compte des contraintes de précédence

Règles de précédence [Babau 1996]

A précède $B \Rightarrow$

- **$r_B \geq r_A$**
- **Si une tâche est périodique, l'autre l'est obligatoirement et $P_A = P_B$**
- **Priorité(A)>Priorité(B)**

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Prise en compte des contraintes de précédence

Si algorithme RM:

r_i*=Max{r_i, ($r^*_{prédécesseur}$)} pour $T_{prédécesseur}$ tâche précédant T_i

<u>Si</u> *A* précède *B* avec $P_A=P_B$ <u>alors</u>
Priorité(*A*)>Priorité(*B*) selon RM

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Prise en compte des contraintes de précédence

Si algorithme DM:

r_i^*=Max{r_i, ($r^*_{prédécesseur}$)} avec $T_{prédécesseur}$ tâche précédant T_i

D_i^*=Min{D_i, ($D^*_{successeur}$)} pour $T_{successeur}$ tâche suivant T_i

Si A précède B avec P_A=P_B alors
Priorité(A)>Priorité(B) selon DM

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Prise en compte des contraintes de précédence

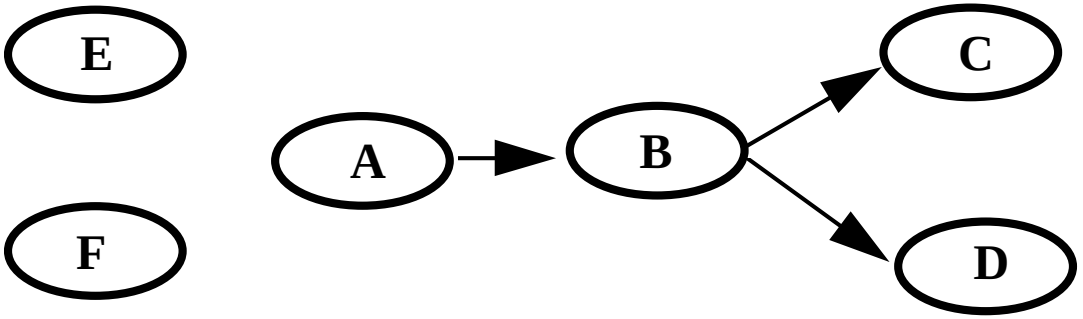
Si algorithme EDF:

$$r_i^* = \text{Max}\{r_i, (r^*_{prédécesseur} + C_{prédécesseur})\}$$

$$d_i^* = \text{Min}\{d_i, (d^*_{successeur} - C_{successeur})\}$$

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Exemple de graphe de précédences



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Caractéristiques temporelles des tâches

Nom Tâche	r_i *date de départ*	C_i *temps d'exécution*	D_i *délai critique*	$d_i=r_i+D_i$ *échéance*	P_i *période*
A	0	1	12	12	12
B	0	2	11	11	12
C	0	1	11	11	12
D	0	2	9	9	12
E	0	1	8	8	8
F	5	4	5	10	-

*Exemples de contraintes temporelles de tâches **avant** transformation*

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Prise en compte de la précédence avec EDF

Nom Tâche	r_i* *date de départ*	C_i *temps d'exécution*	d_i* *échéance*	P_i *période*
A	0	1	5	12
B	1	2	7	12
C	3	1	11	12
D	3	2	9	12
E	0	1	8	8
F	5	4	10	-

*Exemples de contraintes temporelles de tâches **après** transformation*

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Conclusion

- **Les systèmes ou applications temps réel sont**

– ordonnancés à l'aide d'algorithmes spécifiques pour la prise en compte des contraintes de temps

- **On a distingué :**

– L'ordonnancement de tâches périodiques temps réel

– L'ordonnancement de tâches apériodiques

– L'ordonnancement avec contraintes de ressources

– et la prise en compte de la précédence des tâches

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Références

Joëlle Delacroix, « Cours de Systèmes temps réel » dispensés en DESS – DLS, CNAM, 2001-2002.

[Sha 1990]: L. Sha, R. Rajkumar, J. P. Lehoczhy, « Priority inheritance protocols: an approach to real-time synchronization », IEEE Transactions on Computers, Vol. n°39, n°9, sept. 1990, pp.1175-1185.

Claude Kaiser, « Cours de Systèmes temps réel » dispensés en DEA - SAR, Paris 6, 2001-2002.

[Bak 91]: T. P. Baker, "Stack-Based Scheduling of Realtime Processes", J. Real-Time Systems, Vol. 3, N° 1, pp. 67-99, March 1991.

[Thèse_SB 1998]: Samia Bouzefrane, « Étude temporelle des Applications Temps Réel Distribuées à Contraintes Strictes basée sur une Analyse d'Ordonnançabilité », Thèse de Doctorat, LISI, ENSMA, 1998. téléchargeable sur http://cedric.cnam.fr/~bouzefra.

[Chetto 1990]: H. Chetto, « l'ordonnancement dans les systèmes de contrôle temps réel à contraintes strictes », Thèse de Doctorat d'état, université de Nantes, ENSM, déc. 1990, 192 pages.

[Babau 1996]: J. P. Babau, "Etude du comportement temporel des applications temps réel à contraintes strictes basée sur une analyse d'ordonnançabilité", thèse de doctorat, ENSMA, 1996.

[LM 80]: : J. Y. T. Leung, M. L. Merrill, « *A note on Preemptive Scheduling of Periodic Real-Time Tasks* », Information Processing Letters, vol. 11 n°3, pp. 115-118, 1980.

Francis Cottet, Joëlle Delacroix, Claude Kaiser & Zoubir Mammeri, « Scheduling in real-time systems », Ed. Wiley, 2002.