Ordonnancement

Charles André - UNSA

Ordonnancement

$\mathcal{P}$: ensemble de processeurs

$\mathcal{T}$: ensemble de taches

$\mathcal{S}$: ordonnancement (scheduling)

$$\mathcal{S} : \mathcal{P} \times temps \rightarrow \mathcal{T}$$

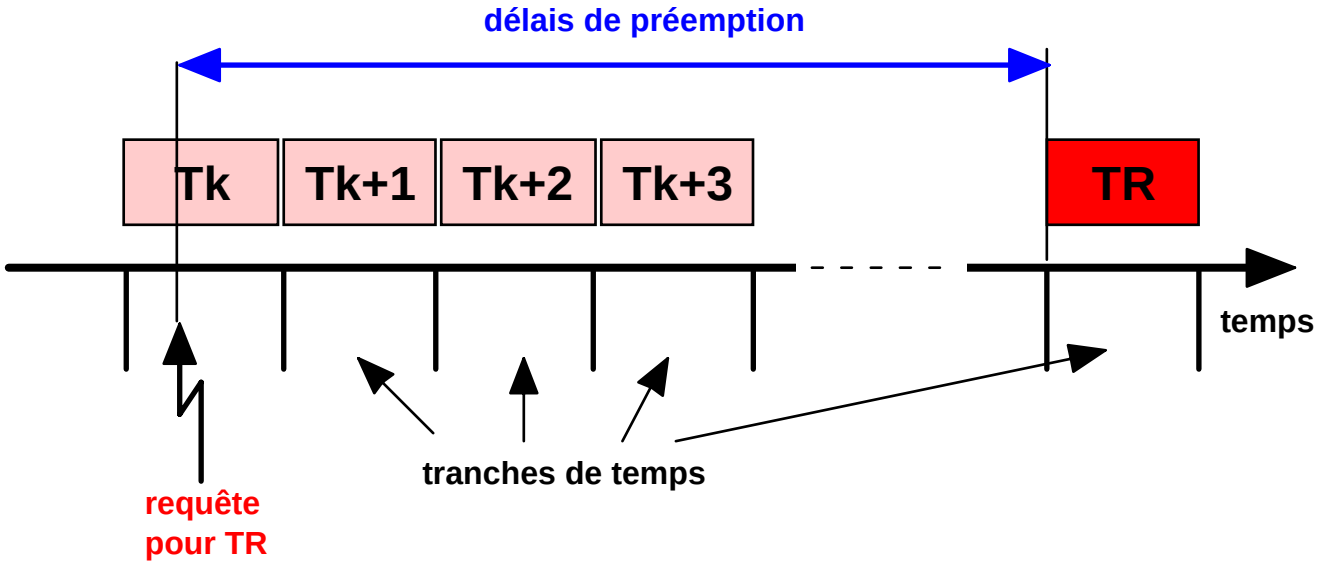
$\mathcal{P}$, $\mathcal{T}$ et un algorithme d'ordonnancement $\mathcal{A}$ étant donnés, $\mathcal{S}$ est un ordonnancement faisable si toutes les tâches respectent leur échéance.

Types d'ordonnancement

- Ordinaire / Temps réel
- En-ligne / Hors-ligne
- Non préemptif / préemptif
- Statique / Dynamique

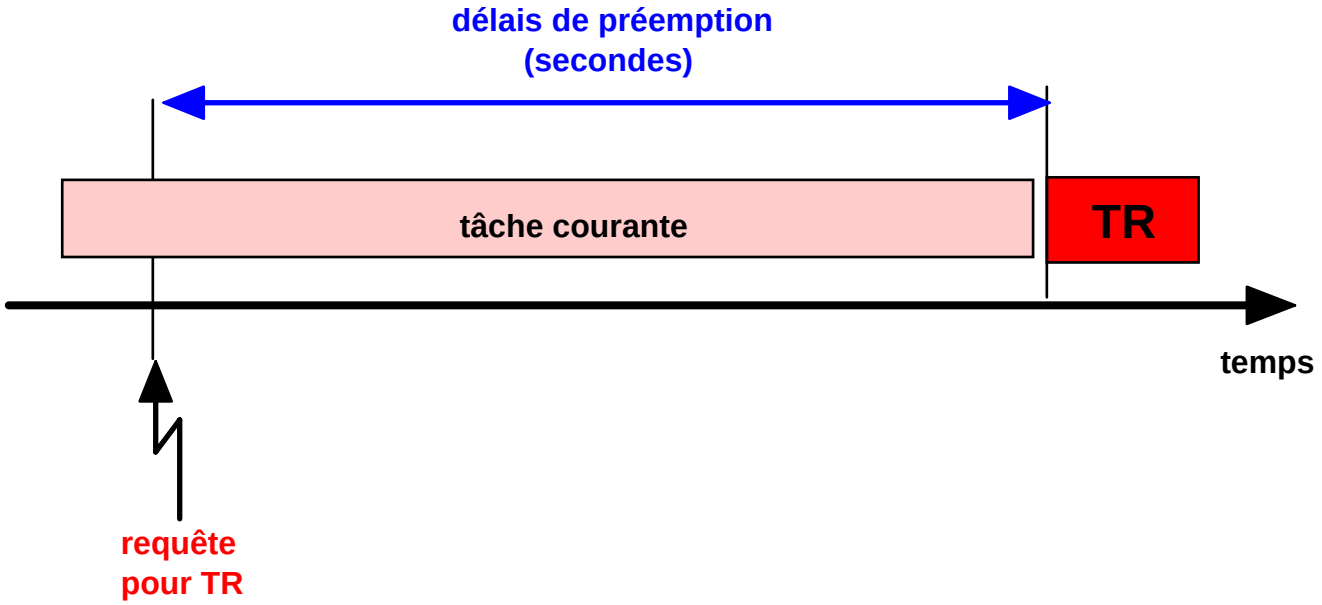
- Contraintes autres que temporelles:
  - Précédence
  - Partage de ressource
- Indépendance : ni précédence, ni partage

Tourniquet



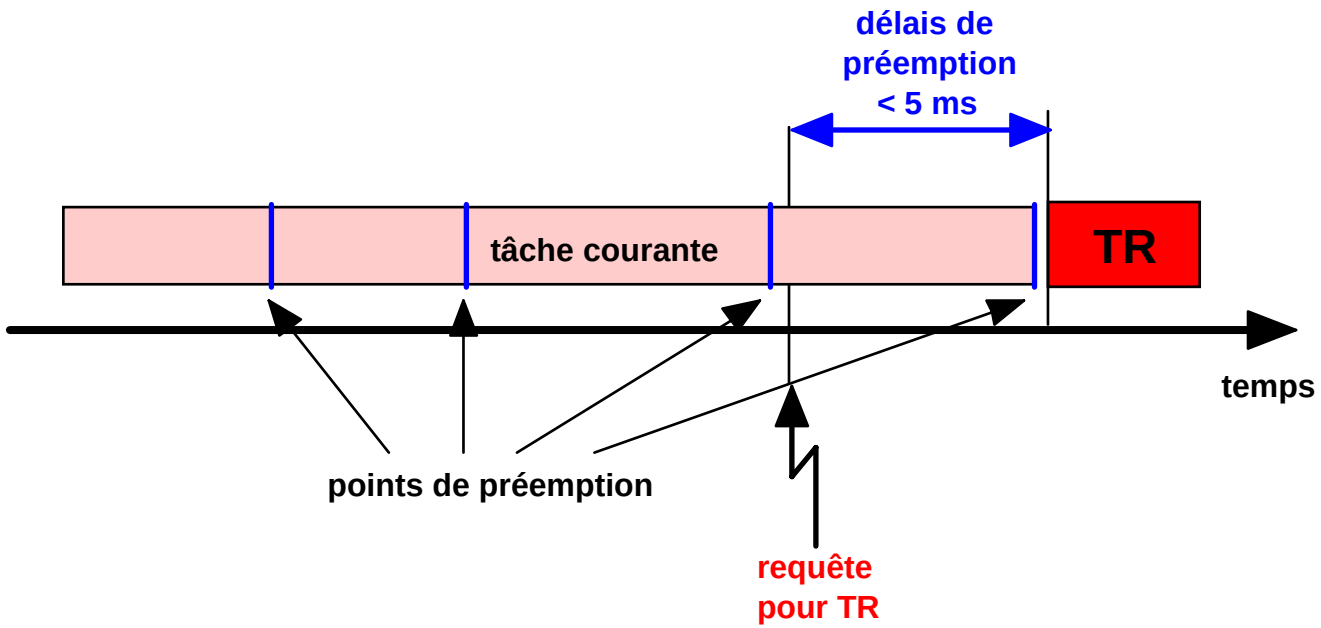
Absolument pas temps réel

Sans préemption



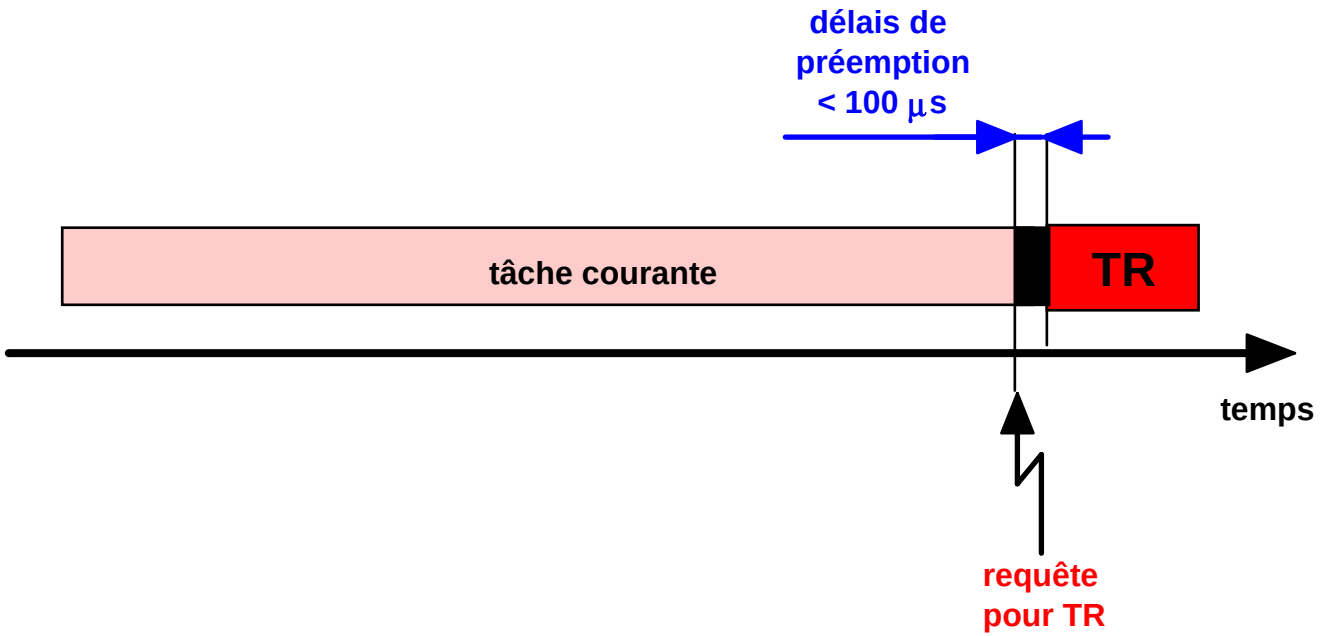
Mauvais pour temps réel

Points de préemption



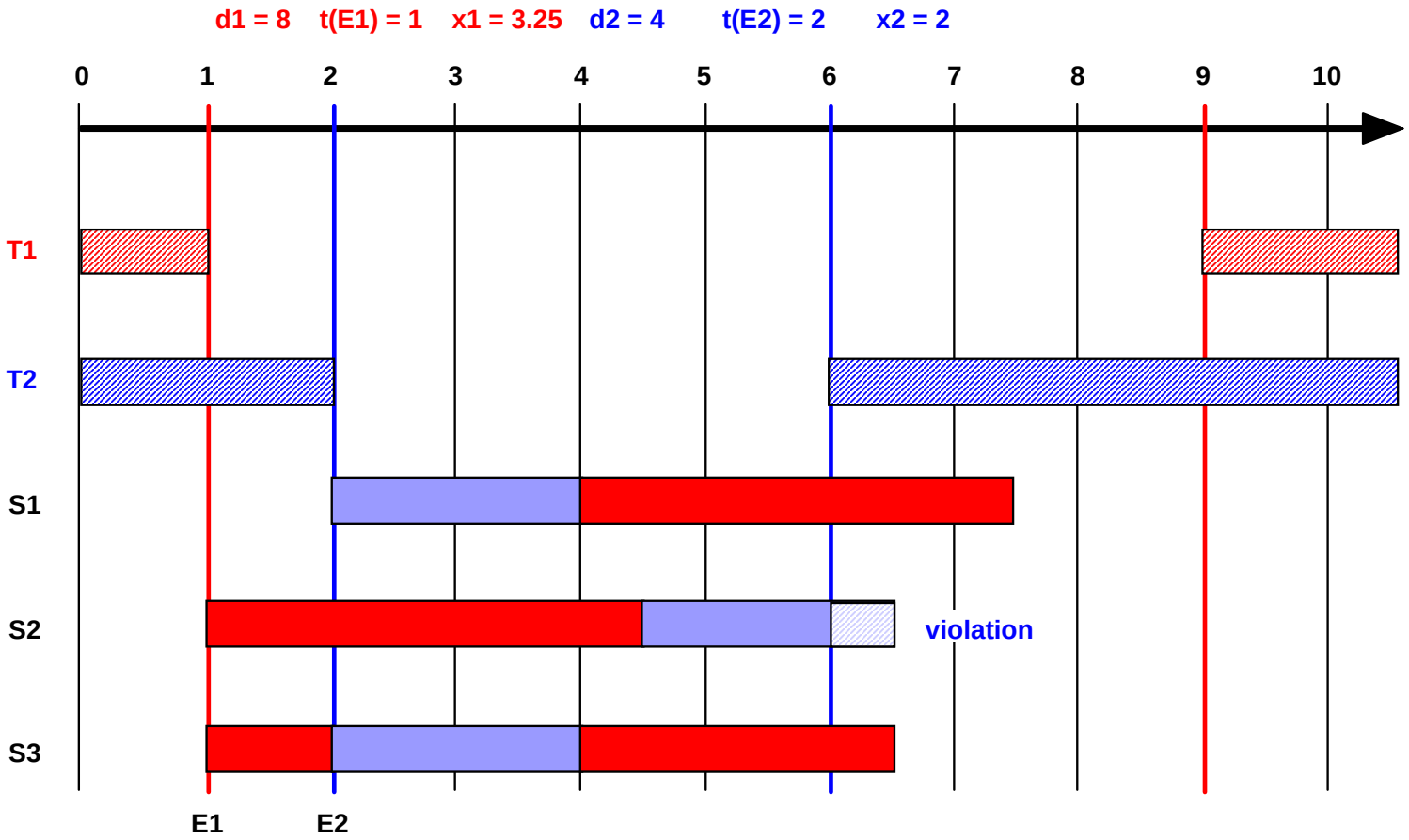
Peut être acceptable si 1 seule tâche temps réel et durées non-préemptibles compatibles avec les contraintes temporelles

Pleinement préemptif



Solution très souvent adoptée

Ordonnançabilité

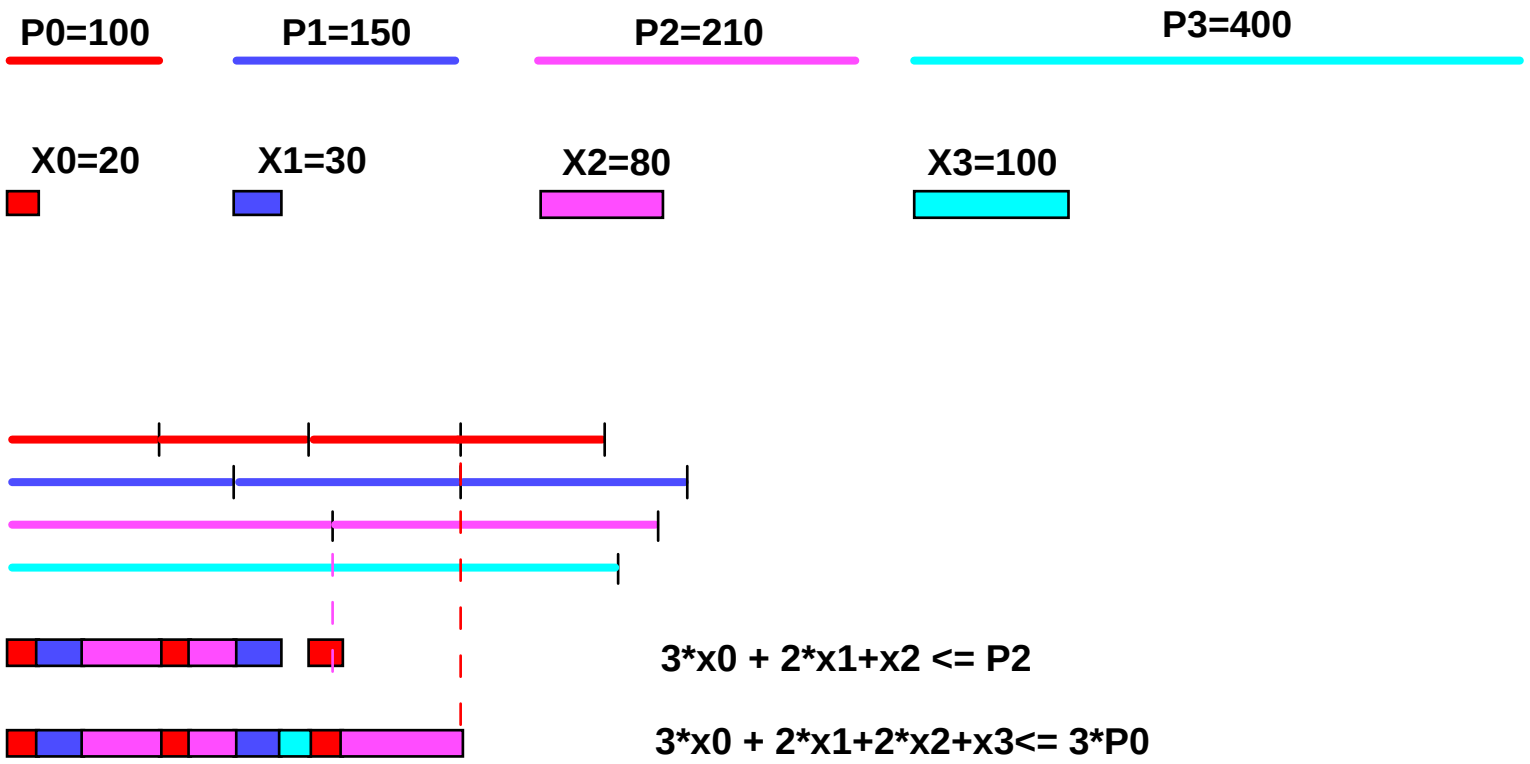


Rate Monotonic

Hypothèses

- H1 : Les tâches sont pleinement préemptibles
- H2 : Seuls les traitements relatifs aux tâches prennent du temps
- H3 : Toutes les tâches sont indépendantes
- H4 : Toutes les tâches sont périodiques
- H5 : L'échéance relative de chaque tâche est égale à la période.

Rate Monotonic



Rate Monotonic (2)

$P_m < P_n \Rightarrow$ priorité$_m >$ priorité$_n$

Durée d'exécution de la tâche T_i (pire des cas)

$$W_i(t) = \sum_{j=1}^{i} x_j * \left\lceil \frac{t}{P_j} \right\rceil \leq t$$

$$\text{pour } t < P_i : \; W_i(t) = x_i + \sum_{j=1}^{i-1} x_j * \left\lceil \frac{t}{P_j} \right\rceil \leq t$$

Condition suffisante de RM-Ordonnançabilité

$$U = \sum_{i=1}^{i=n} \frac{x_i}{P_i} \leq n * \left(2^{\frac{1}{n}} - 1 \right)$$

Rate Monotonic (3)

Condition nécessaire et suffisante

T_i (et tous les T_i j<i) **est RM-ordonnançable ssi**

$$\exists t : W_i(t) \leq t$$

En pratique : calculer W_i pour

$$t \in \tau_i = \left\{ l \times P_j \,\middle|\, j = 0 \ldots i; l = 1 \ldots \left\lfloor \frac{P_i}{P_j} \right\rfloor \right\}$$

Temps de réponse

$$R_k = x_k + \sum_{j \in pp(k)} \left\lceil \frac{R_k}{P_j} \right\rceil \times x_j$$

pp(k) = ensemble des tâches plus prioritaires que Tk

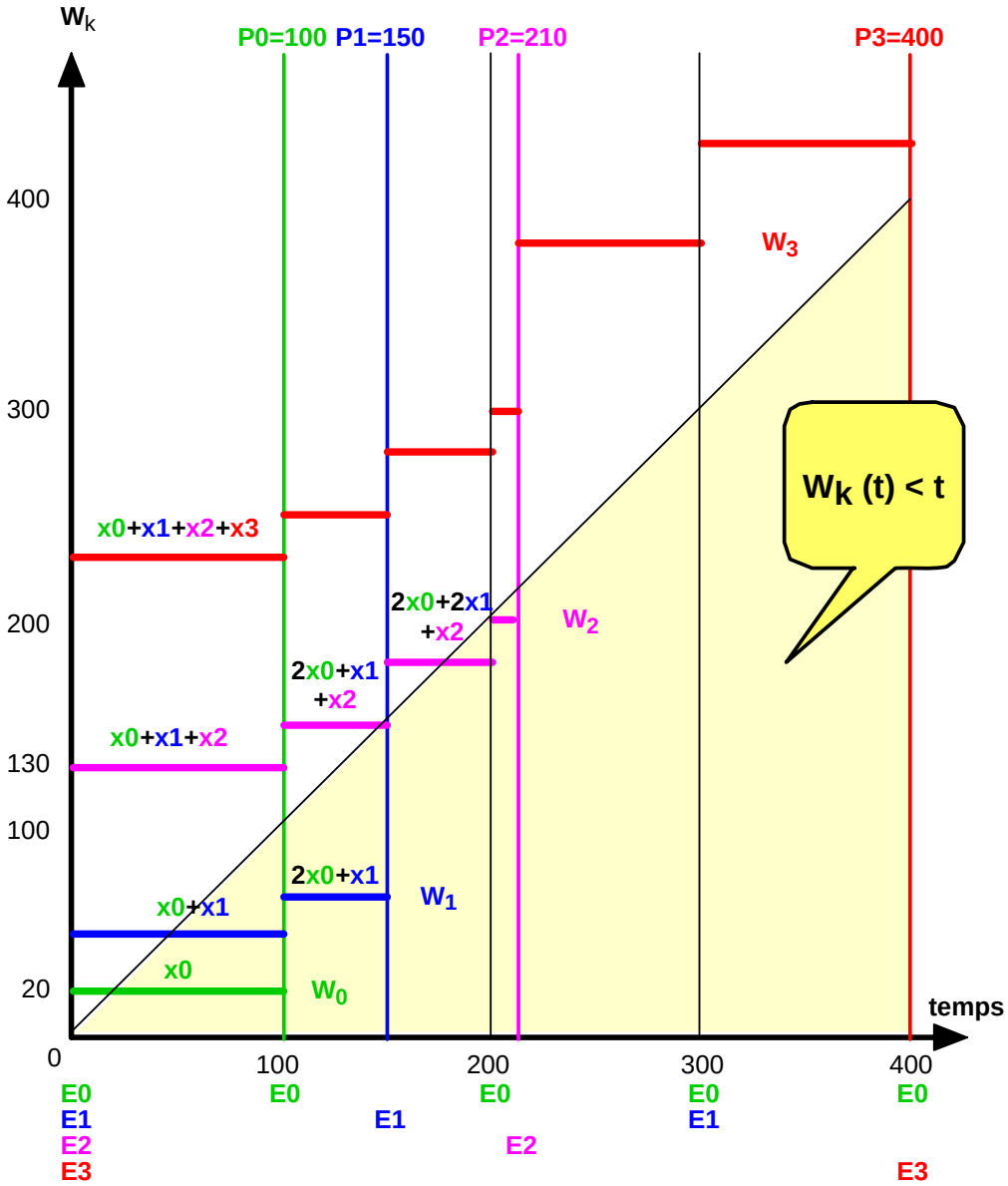
Calcul de point fixe :

```
Pour tout k dans 0..n-1 faire
        Rk ← xk
        itérer
                R ← f(Rk)
                si R = Rk alors
                        exit // Rk est solution
                sinon
                        si R > Pk alors
                                exit //Tk n'est pas ordonnançable
                        sinon
                                Rk ← R
                        fin si
                fin si
        fin itérer
fait
```

Rate Monotonic (4)

Tâche	0	1	2	3
X	20	30	80	100
P	100	150	210	400

Rate Monotonic (5)



Rate Monotonic (6)

Tâche **A** : $P_A = 20$, $x_A = 10$ Tâche **B** : $P_B = 50$, $x_B = 25$

$P_A < P_B \Rightarrow$ priorité$_A$ > priorité$_B$

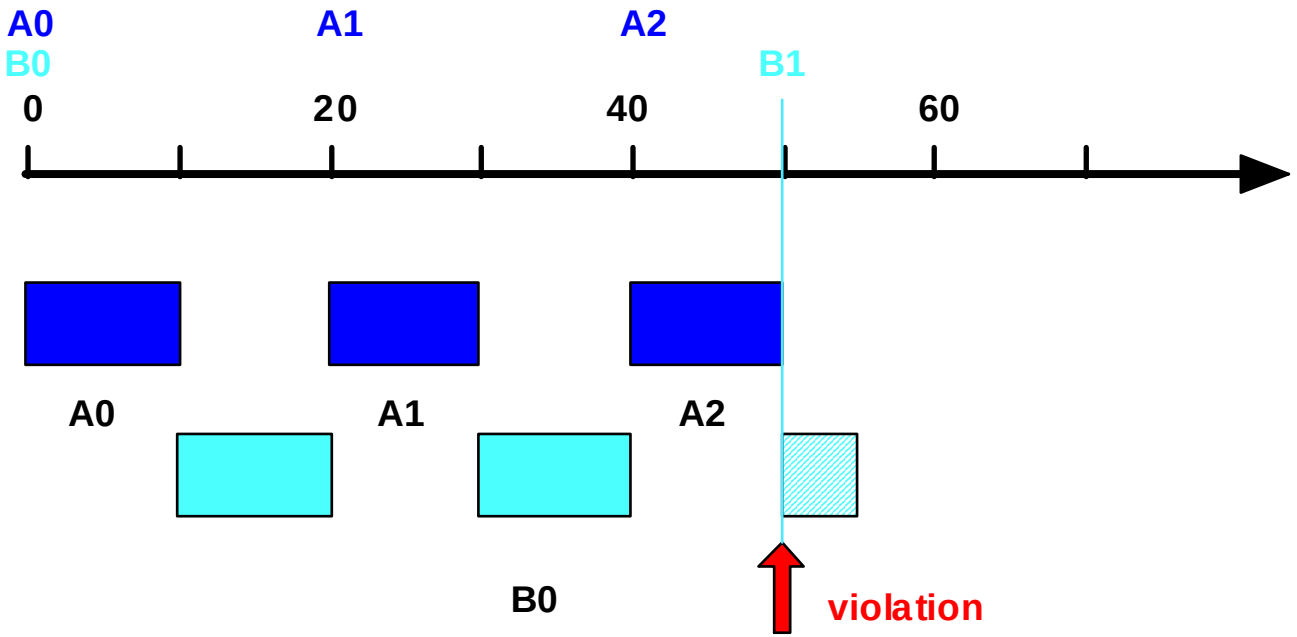
U = 10/20 + 25/50 = 1 > 2 *(2 ½ -1) = 0,828

Donc pas condition suffisante de RM-ordonnançabilité

Date	Evénement	Restant A	Restant B
0	**A0,B0**	**10**	**25**
10	↓A0	0	**25**
20	**A1**	**10**	15
30	↓A1	0	**15**
40	**A2**	**10**	5
50	**B1**	0	**5**

VIOLATION

Rate Monotonic (7)

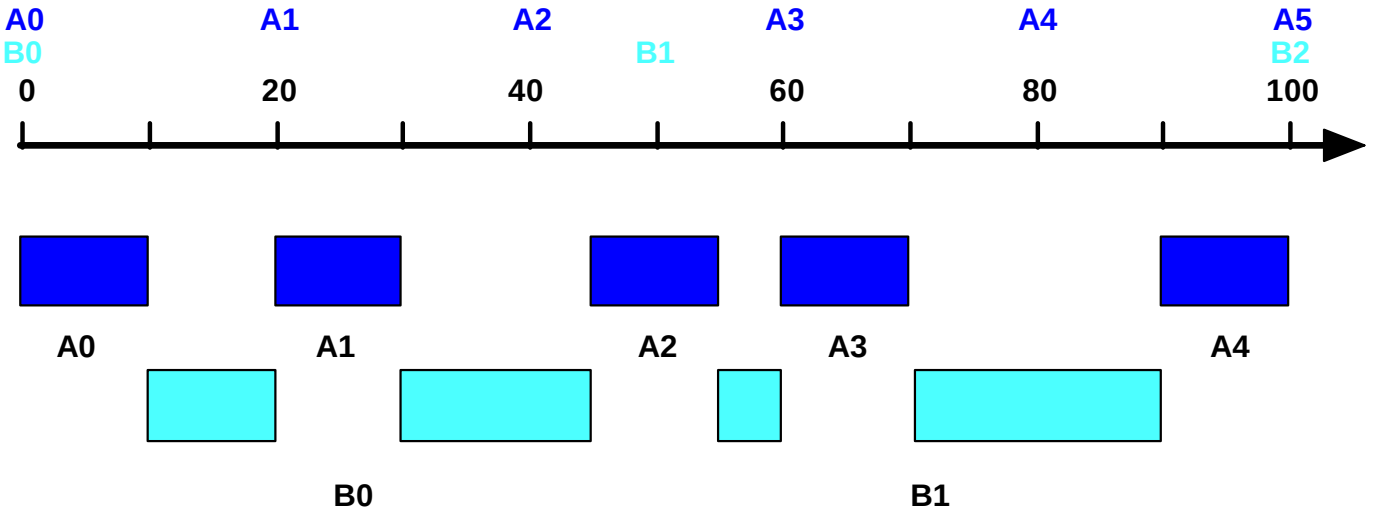


Earliest Deadline First

Tâche **A** : $P_A = 20$, $x_A = 10$ Tâche **B** : $P_B = 50$, $x_B = 25$

$P_A < P_B \Rightarrow$ priorité$_A$ > priorité$_B$ pour RM
Priorité à déterminer à chaque fois pour EDF

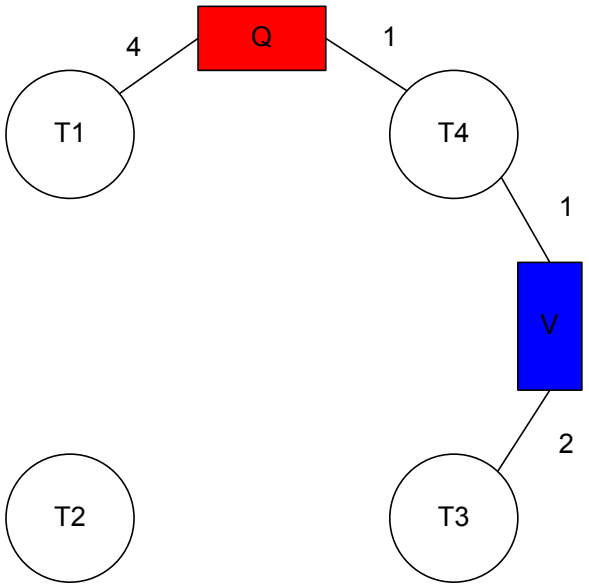
U = 10/20 + 25/50 = 1

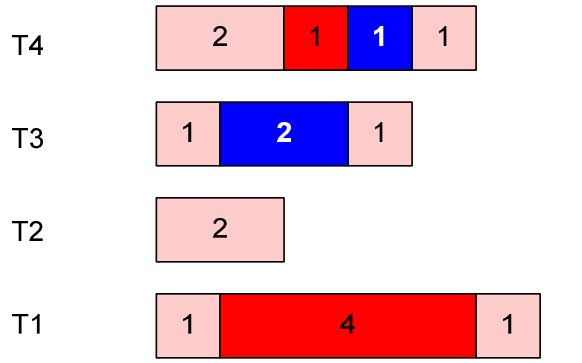


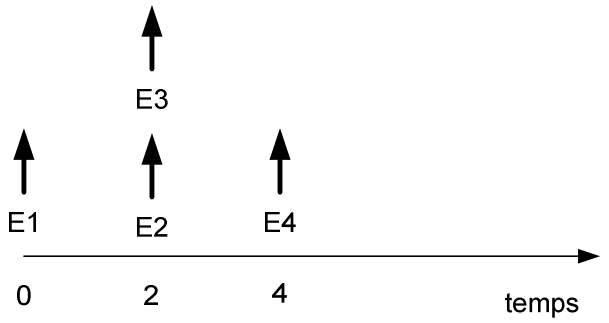
Date	Evénement	Echéance A	Restant A	Echéance B	Restant B
0	**A0**,**B0**	**20**	10	50	25
10	↓A0	-	0	**40**	25
20	**A1**	**20**	10	30	15
30	↓A1	-	0	**20**	15
40	**A2**	20	10	**10**	5
45	↓B0	**15**	10	-	0
50	**B1**	**10**	5	50	25
55	↓A2	-	0	**45**	25
60	**A3**	**20**	10	40	20
70	↓A3	-	0	**30**	20
80	**A4**	**20**	10	**20**	10
90	↓B1	**10**	10	-	0
100	↓A4,**A5**,**B2**	**20**	10	50	25

Charles André - UNSA

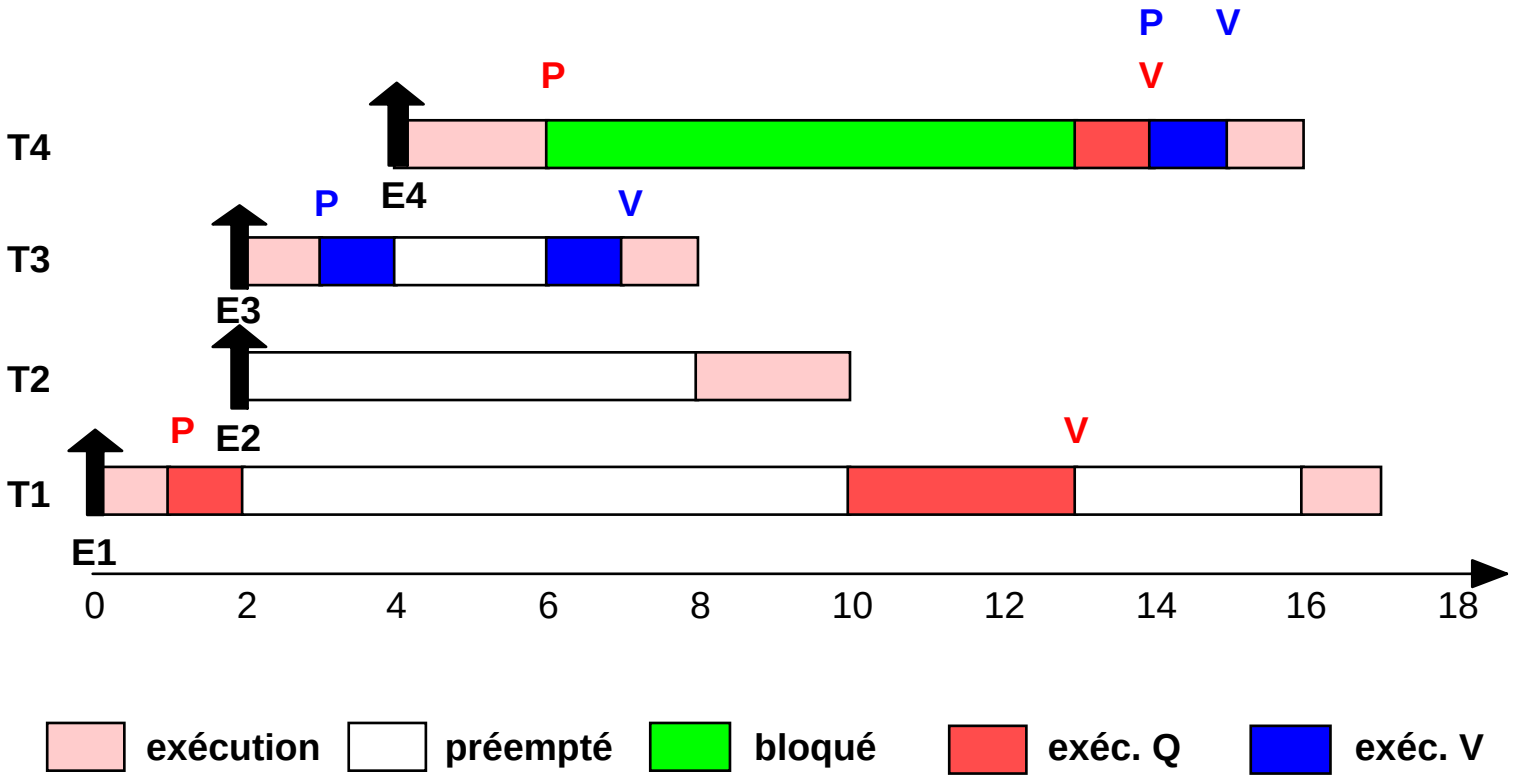
Partage de ressource



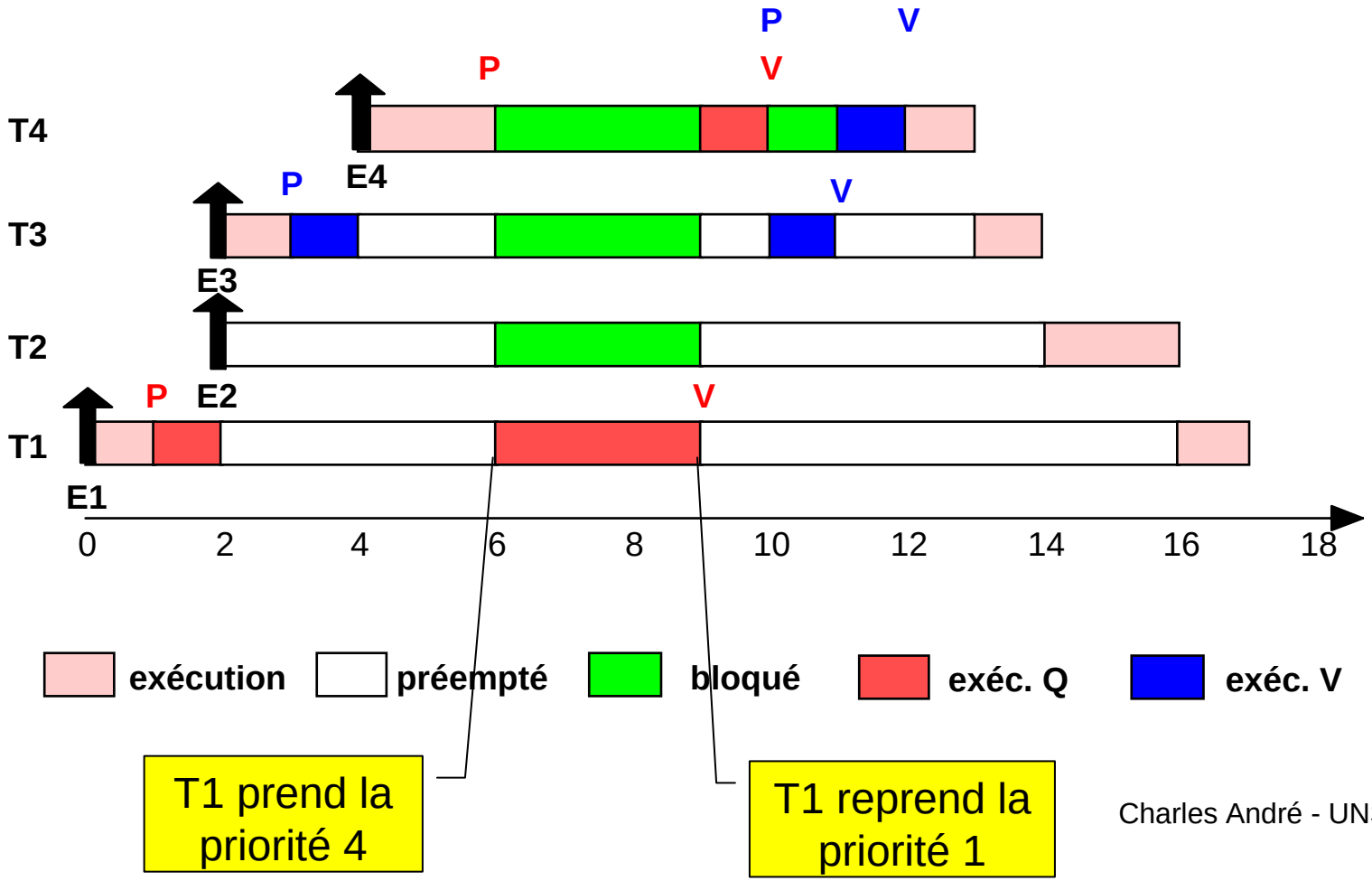




Inversion de priorité



Héritage de priorité



Héritage de priorité

Propriétés

Si une tâche contient m sections critiques elle peut être bloquée au plus m fois par des tâches de priorité inférieure à la sienne.

Dans le pire des cas le temps de blocage pour la tâche T_k est exprimé par

$$B_k = \sum SC(l, j) \text{ pour } T_l \in pp(T_k), j \in 1..m$$

où $pp(T_k)$ est l'ensemble des tâches ayant une priorité plus petite que T_k, SC(l,j) est la durée d'exécution, hors blocages et préemptions, de la tâche T_l, dans la section critique j.

Le temps de réponse pour la tâche T_k devient alors

$$R_k = x_k + B_k + \sum_{T_j \in pp(T_k)} \left\lceil \frac{R_k}{P_j} \right\rceil \times x_j$$

Le gros problème avec cette technique est qu'elle peut conduire à des deadlocks.

Priorité plafond

(Priority Ceiling Protocol)

Chaque tâche a une priorité statique par défaut,

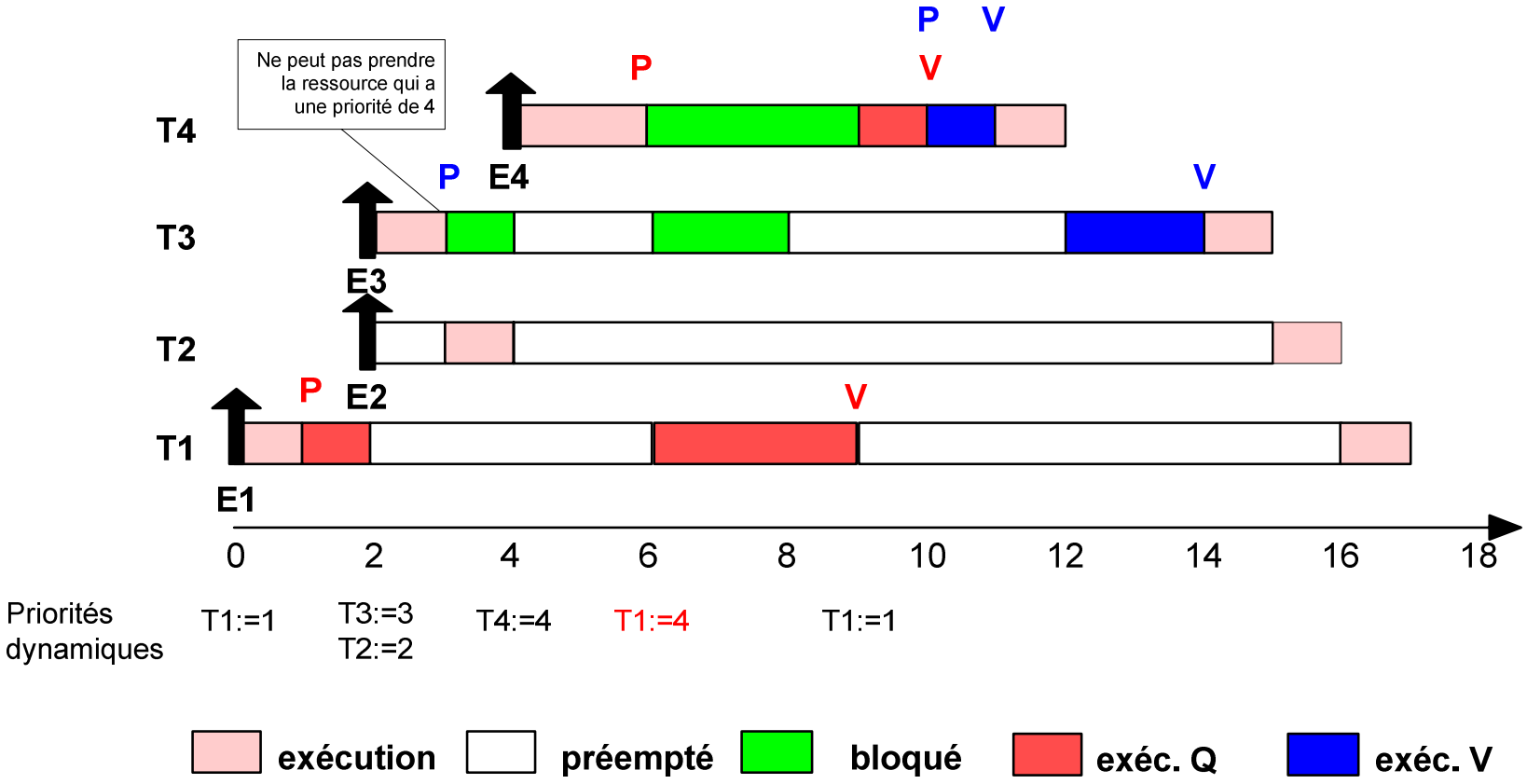
Chaque ressource (ou section critique) a une priorité statique plafond égale à la priorité maximale des tâches qui y accèdent,

Une tâche a une priorité dynamique qui est le maximum de sa priorité statique et des priorités héritées des tâches plus prioritaires qu'elle bloque,

Une tâche ne peut prendre (lock) une ressource que si sa priorité dynamique est supérieure à la priorité plafond des ressources déjà occupées (sans tenir compte de celle qu'elle occupe elle-même).

Priorité plafond

Priorité plafond des ressources: Q:4; V:4



Priorité plafond

Propriétés

Une tâche prioritaire ne peut être bloquée que par au plus une tâche moins prioritaire,

L'absence de deadlocks,

Les ressources sont accédées de façon exclusive (propriété assurée par le protocole lui-même).

Priorité plafond immédiate

(Immediate Ceiling Priority Inheritance)

Il s'agit d'une variante du précédent qui adapte une discipline plus simple : la priorité de la tâche est montée dès qu'elle prend la ressource (et non quand elle bloque effectivement une tâche plus prioritaire).

Les règles deviennent :

- Chaque tâche a une priorité statique par défaut,
- Chaque ressource (ou section critique) a une priorité statique plafond égale à la priorité maximale des tâches qui y accèdent,
- Une tâche a une priorité dynamique qui est le maximum de sa priorité statique et des priorités plafond de toutes les ressources qu'elle occupe.

Priorité plafond immédiate