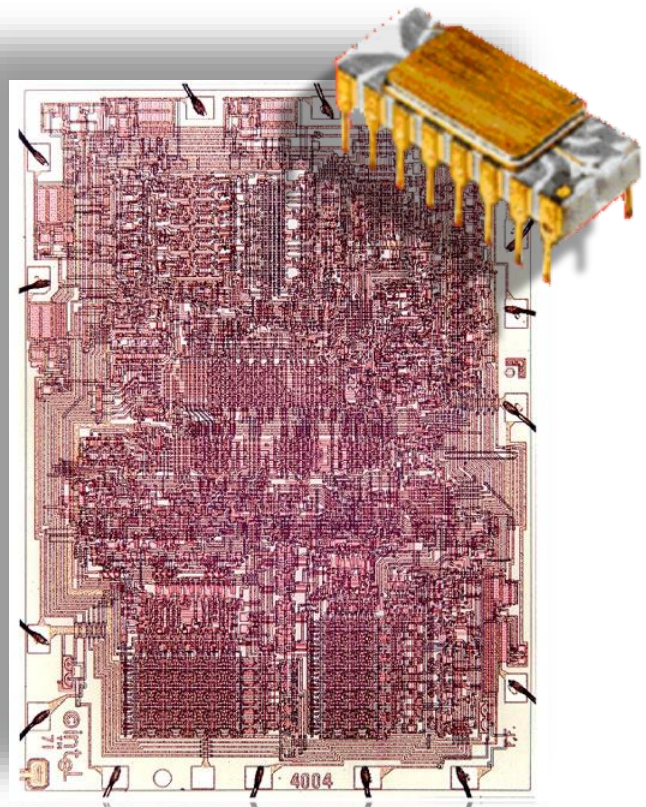
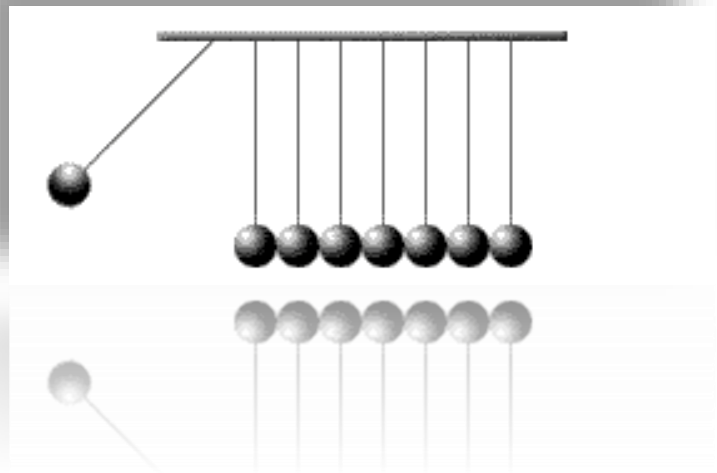
Multitâches

Fork, pthreads, etc..

Rappel…

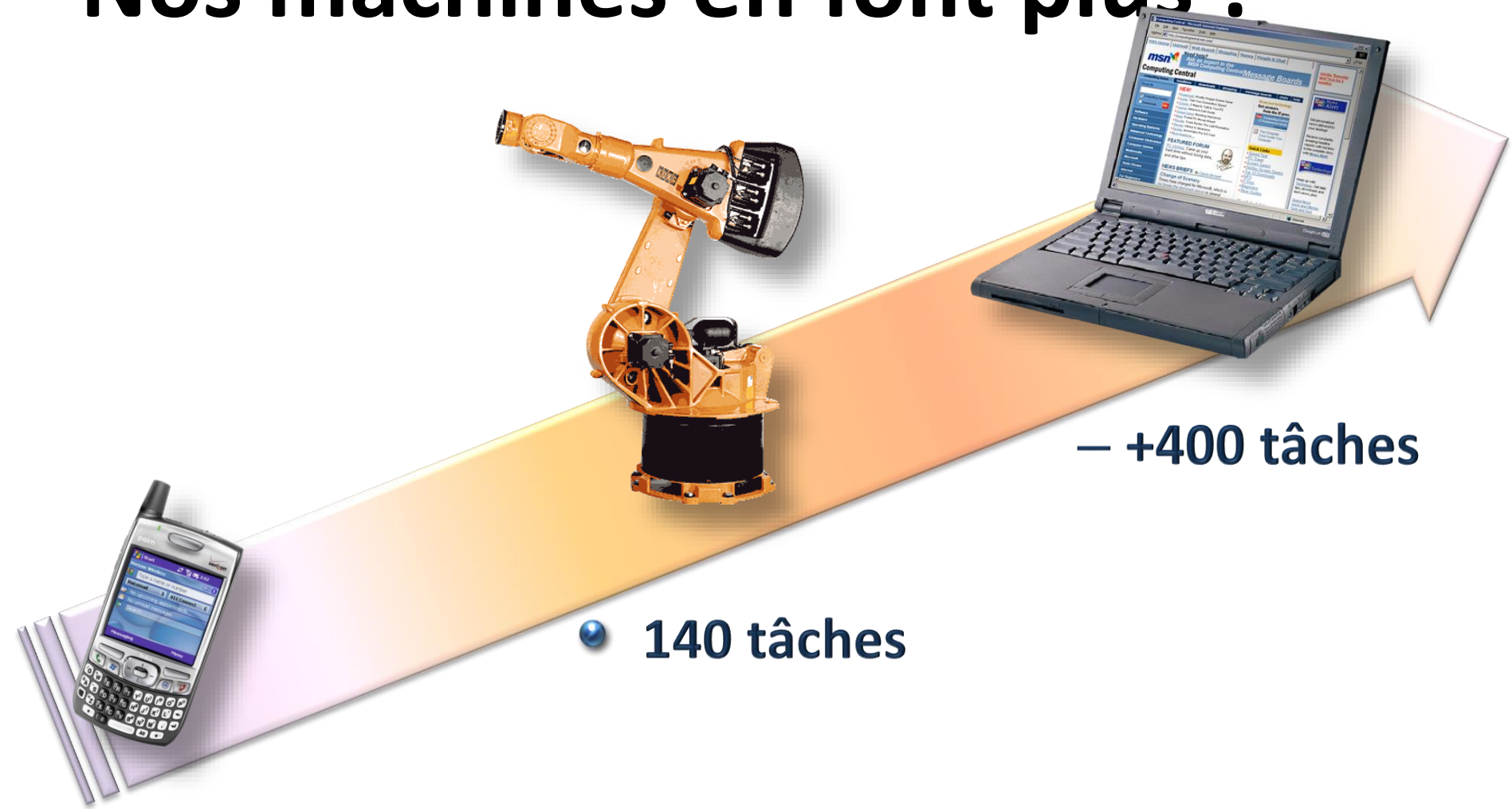


=



L’UC ne fait qu’une chose à la fois !

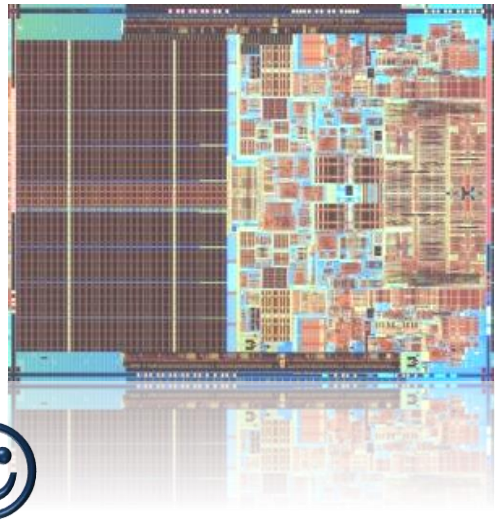
Nos machines en font plus !



Deux approches

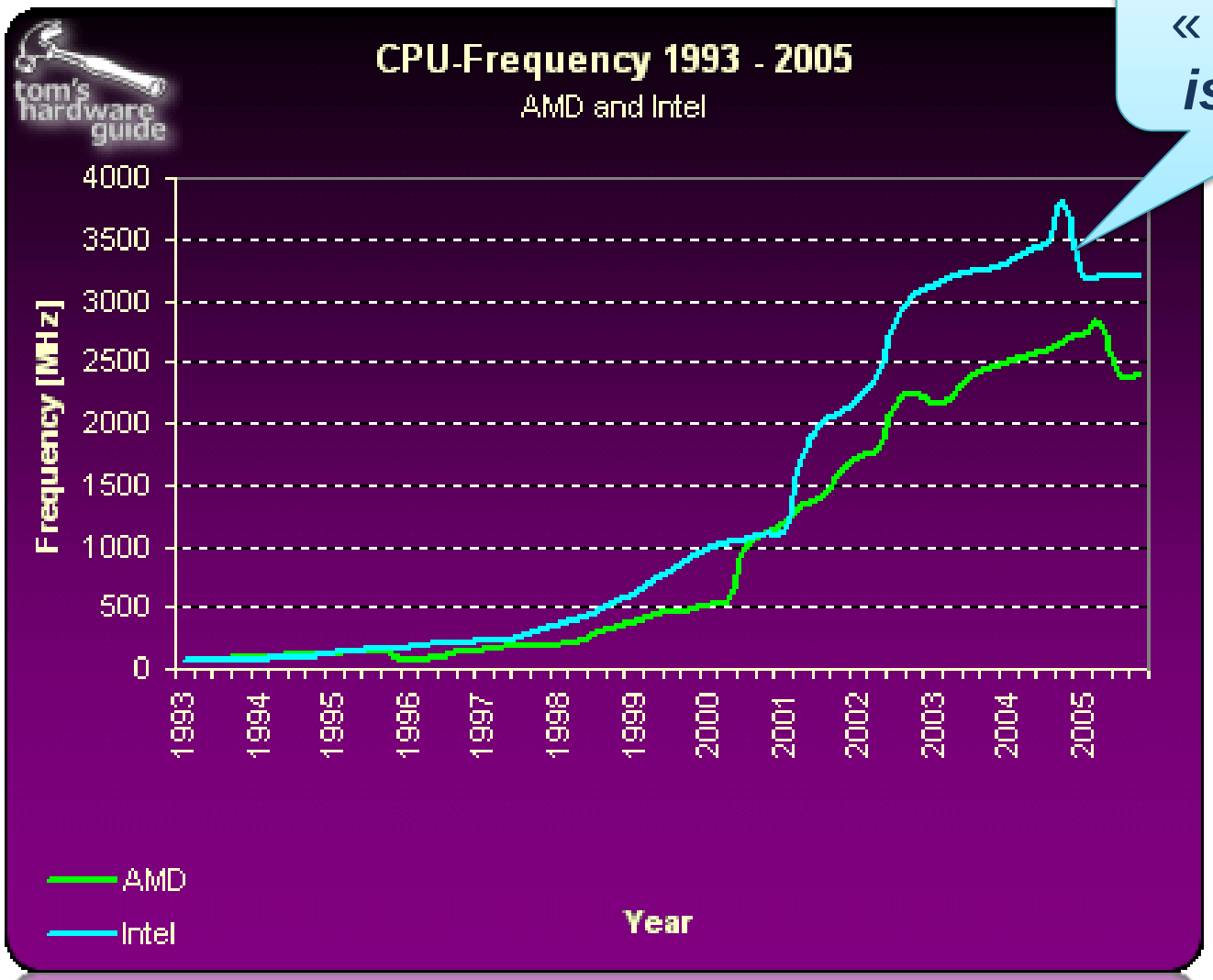
- Multi-Tâches
  - Création de CPU virtuelles
  - Economique en matériel
  - Nécessite une couche de logiciel

- Multi-Cœur
  - L'UC fait « plusieurs choses »
  - Justifié par la loi de MOORE
  - Le logiciel doit s'adapter…



Multi-tâches + Multi-Cœur = ☺

Fréquences CPU



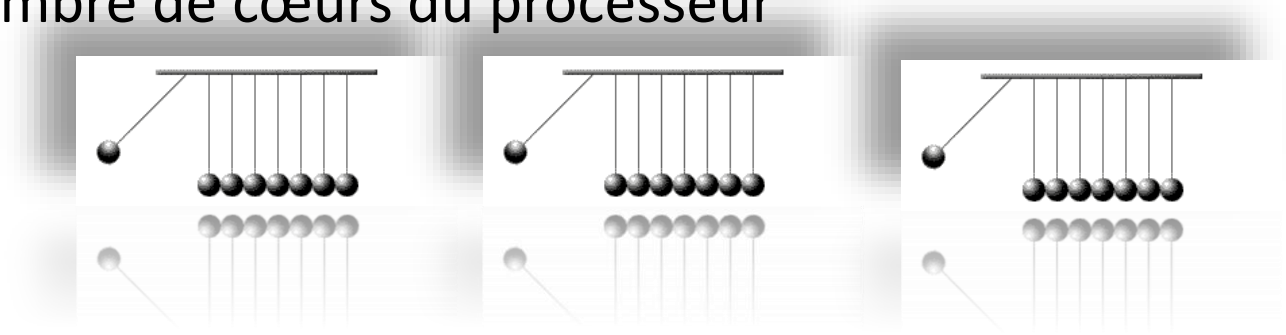
Oups!
« ***Free lunch is over*** » ☹

Evolution des CPU

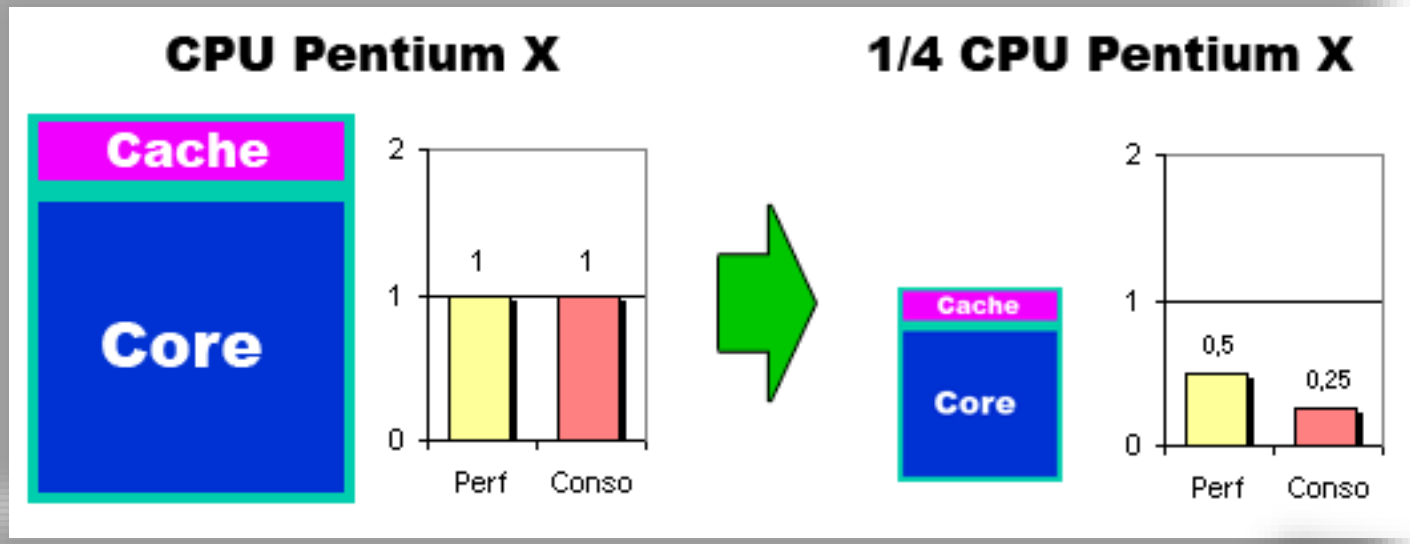
- Pour pallier au problème de consommation, on introduit l'exécution simultanée de programmes

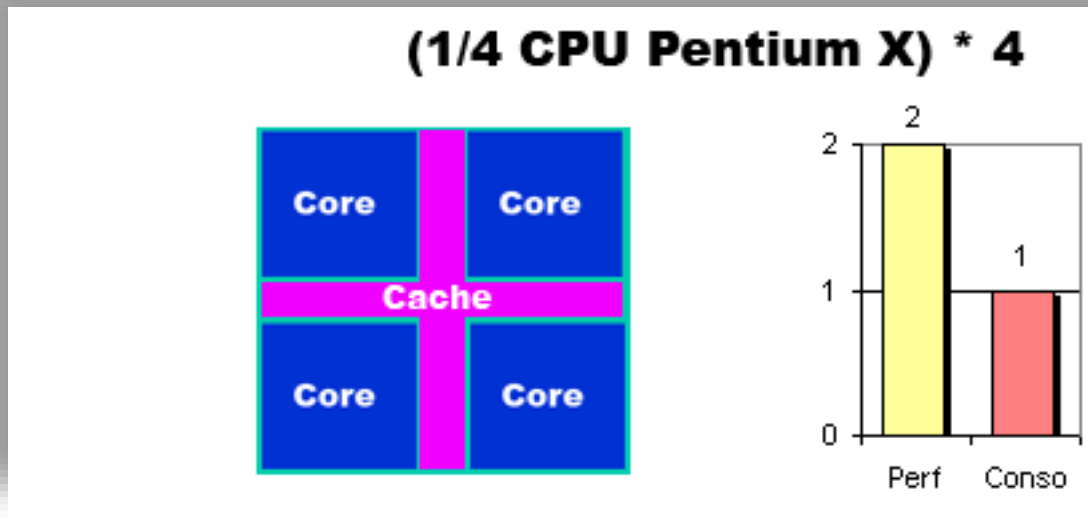
- Augmenter le Parallélisme d'exécution
  - SMT (Hyper-threading): 2002 (Pentium 4)
  - CMP (Chip Multi Processing): 2005/2006 (Athlon X2, Pentium D, Core Duo)

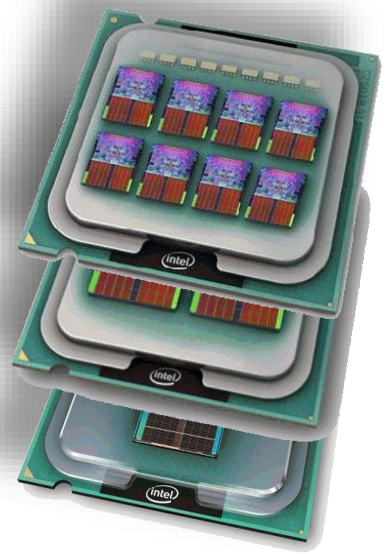
- Une des voies d'avenir des processeurs est de faire monter le nombre de cœurs du processeur



Une Evolution favorable



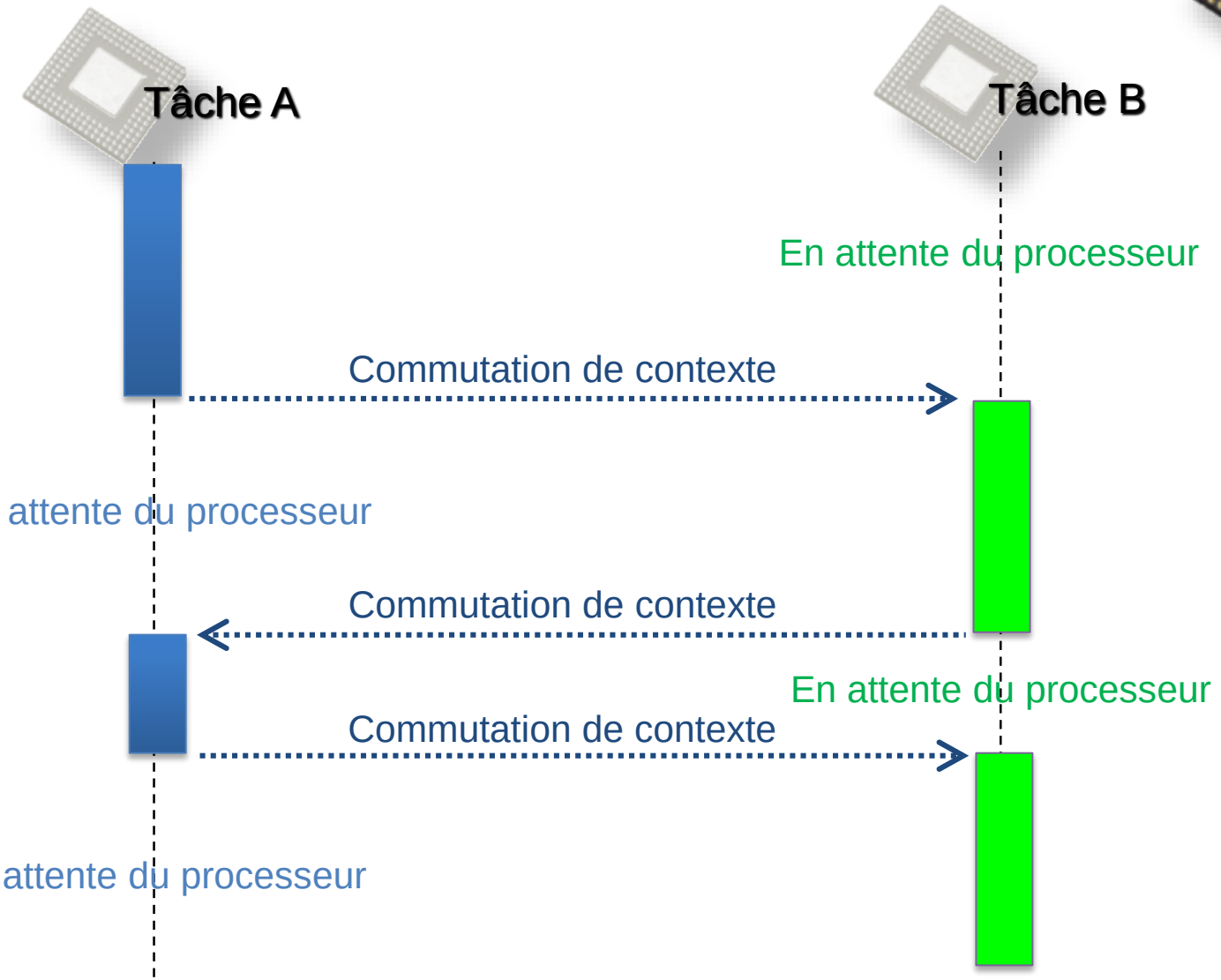




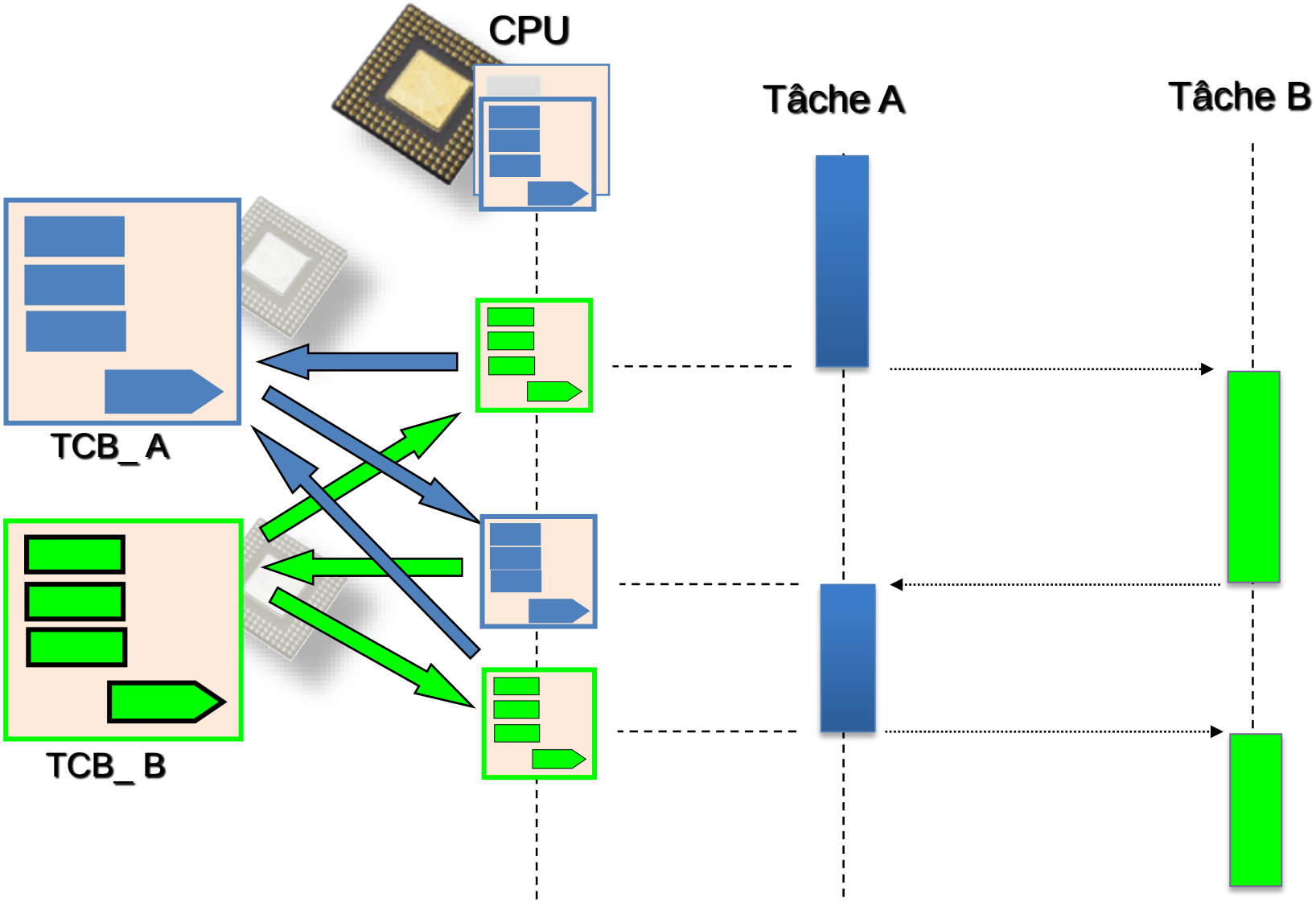
3. Multitâches

```
MOV EAX, a
CMP EAX, b
NEW CPU
MOV EAX, 1
JMP end
```

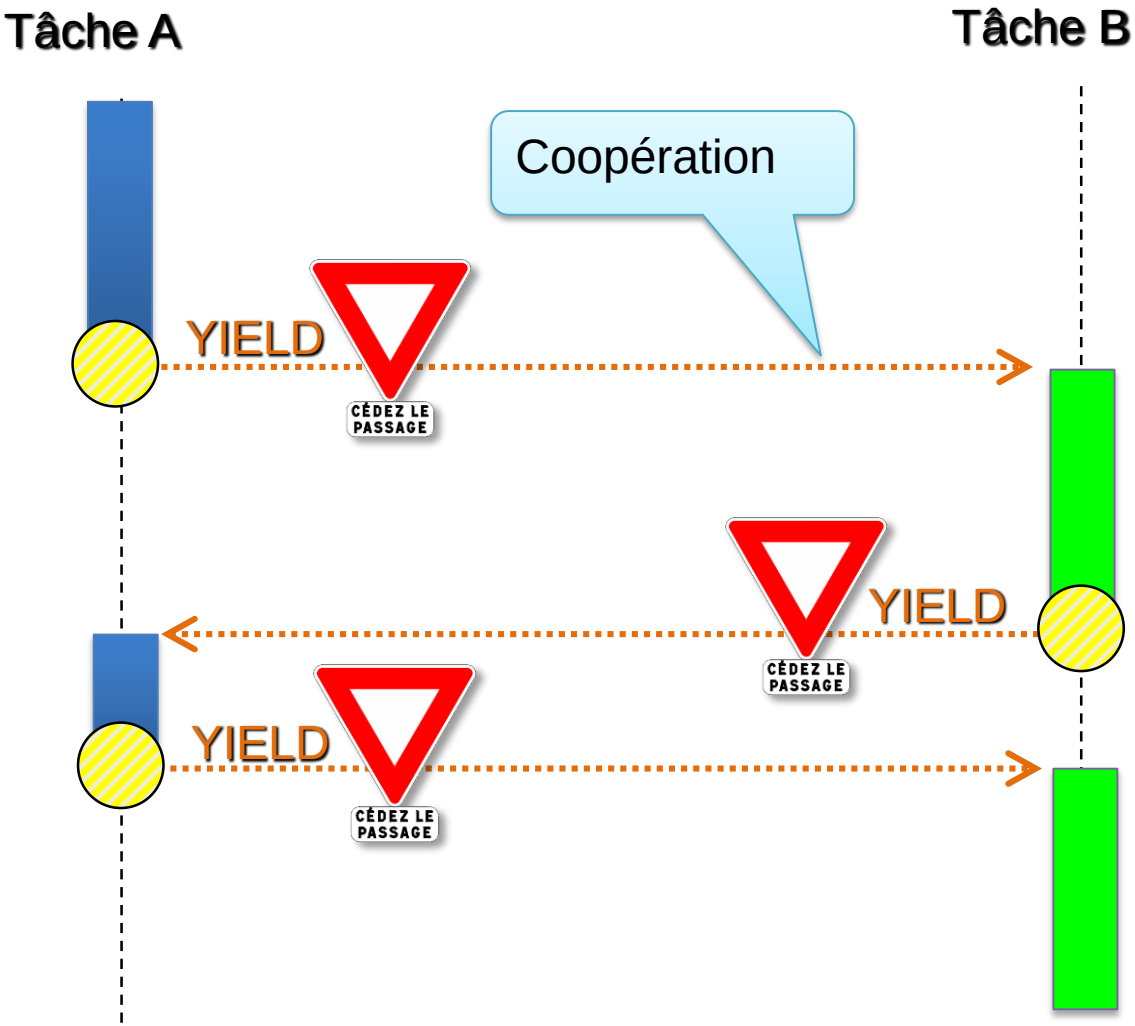
Gestion du Processeur



Contexte de Tâche (TCB)



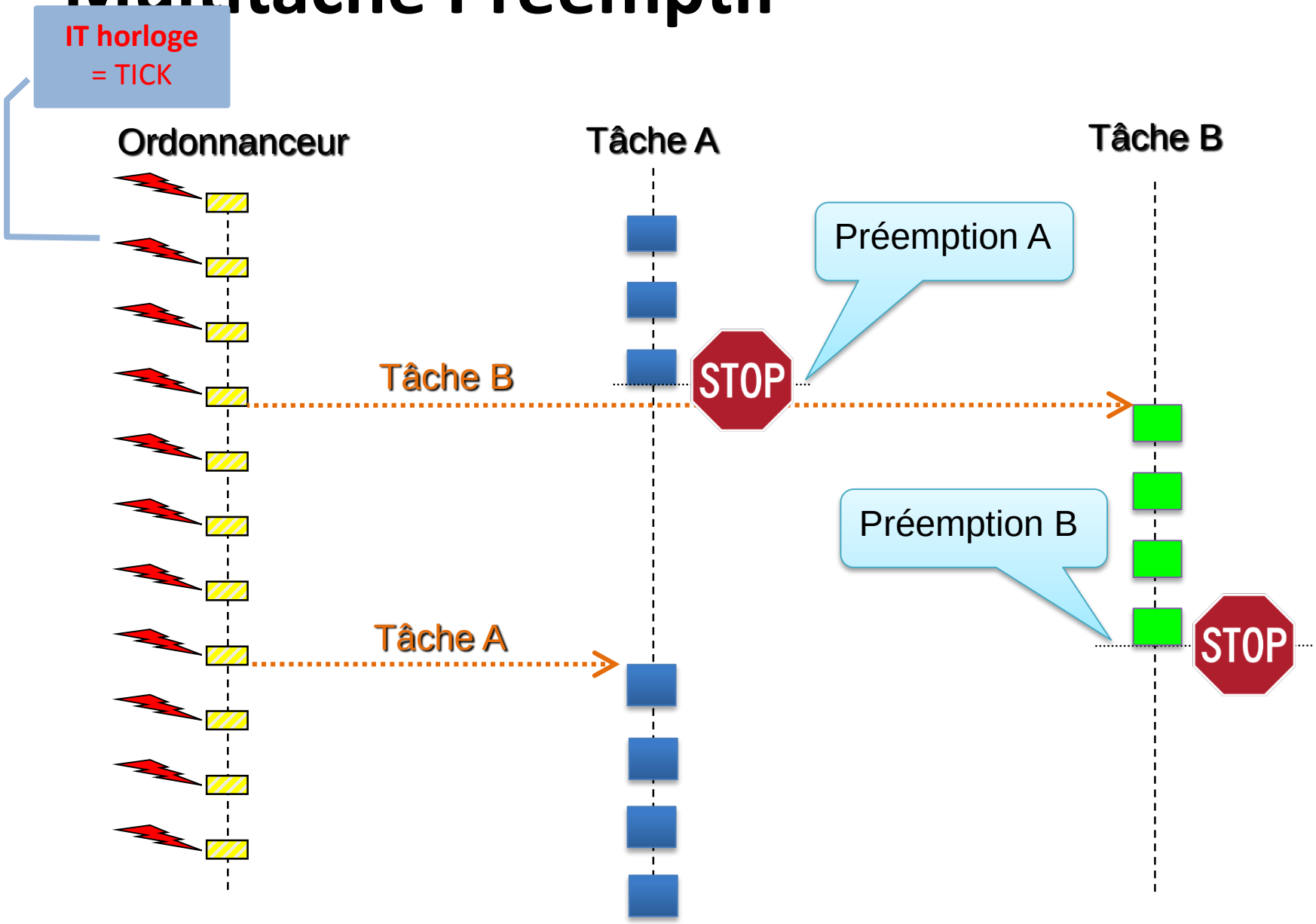
Multitâche Coopératif



Multitâche Cooopératif

- La commutation de contexte n'intervient que suite à un appel explicite dans la tâche en cours
  - La tâche prend le risque de se faire « arrêter »
- Le code de commutation de tâche s'exécute dans le contexte de la tâche qui fait l'appel
  - Le code de changement de contexte peut être « linké »
  - Tout est parfaitement séquentiel
- Le séquencement est entièrement écrit par les développeurs des applications

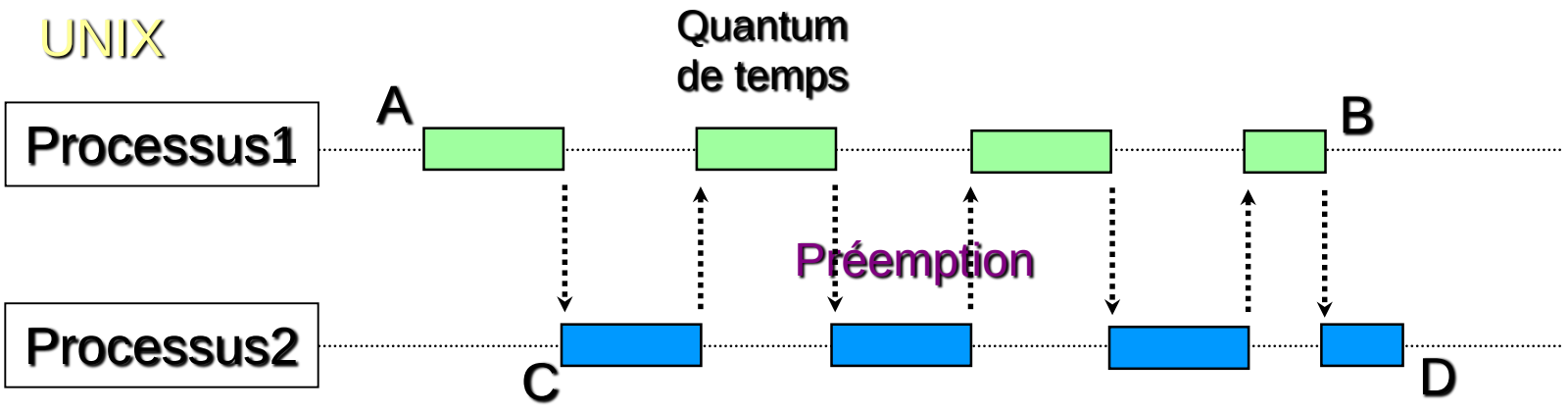
Multitâche Préemptif



Multitâche Préemptif

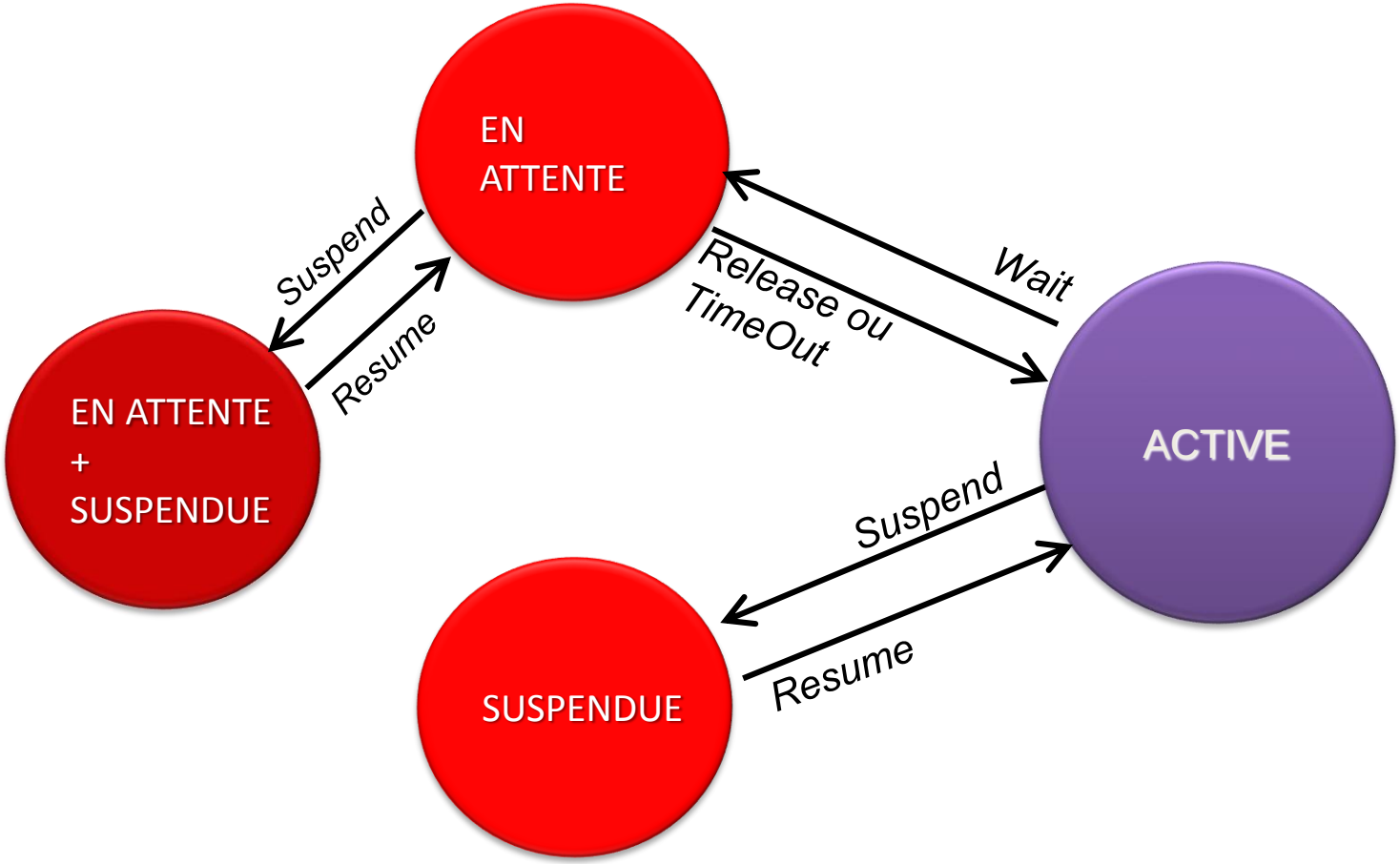
- La commutation de contexte intervient sur un critère extérieur au flux d'exécution
  - La tâche préemptée « n'a rien demandé… »
- Le code de commutation de tâche s'exécute dans un contexte « hors tâches »
  - Il est nécessaire de faire intervenir une interruption
  - Le code de commutation est critique
  - Notion de TICK système (grain temporel)
- Il est possible d'effectuer les changements de contexte sans faire intervenir les développeurs

Exemple préemptif

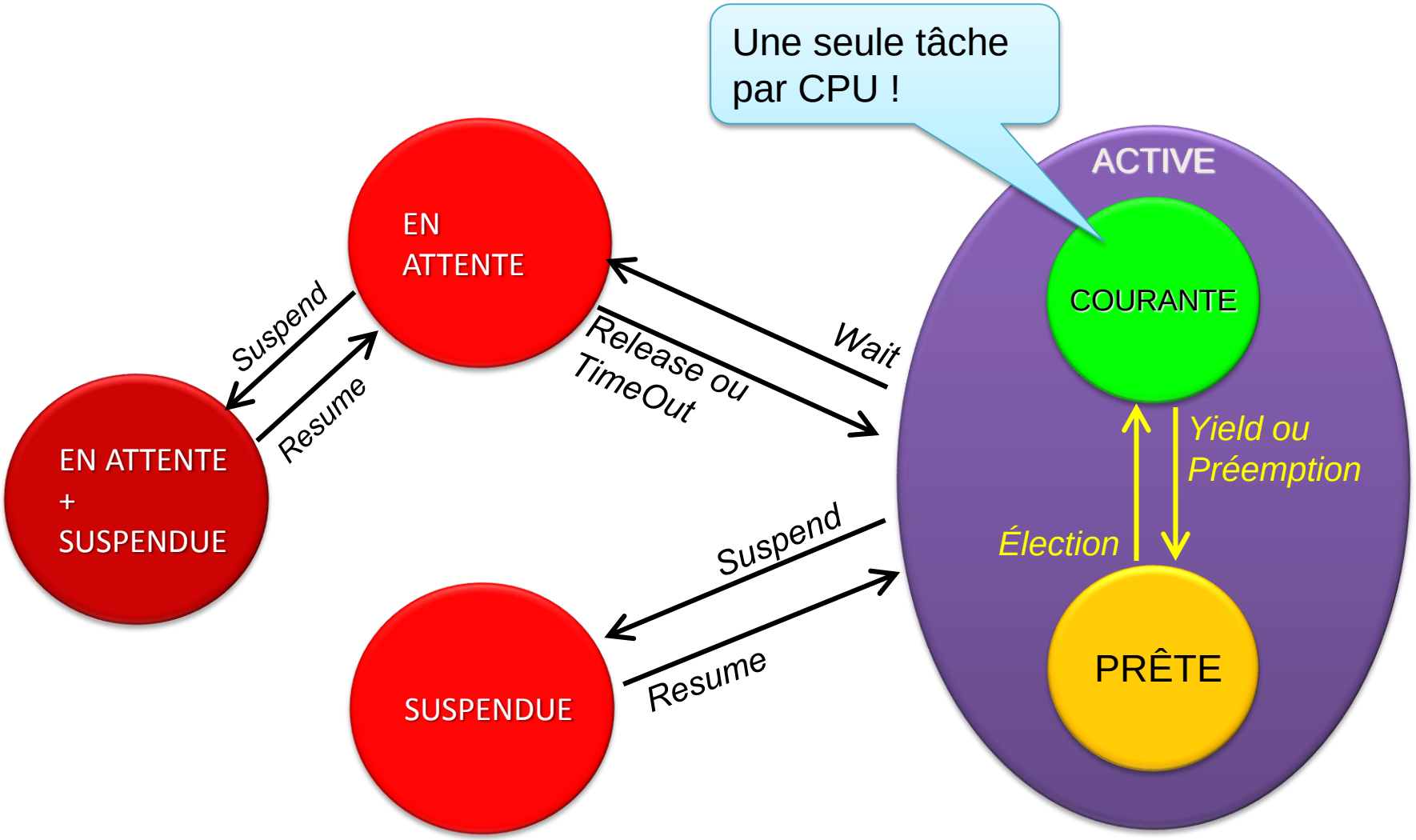


- Partage du système entre plusieurs utilisateurs
- Mécanisme de « temps partagé » avec quantum
- Le passage d'une tâche à l'autre est également appelé « Round Robin »

États d'une tâche – vue du développeur



États d'une tâche – vue du scheduler



La planification des tâches

- Le changement de contexte doit être précédé d'un travail de planification qui permet de décider quelle tâche doit tourner.
  - Notion de SCHEDULER *(Ordonnanceur)*
- Les choix d'implémentation au niveau de la planification sont fortement influencées par l'usage qui sera fait de l'OS :
  - Confort d'utilisation → Planification sophistiquée et très intrusive (UNIX, NT), l'objectif est un équilibrage global en réduisant les risques de blocage.
  - Déterminisme → Respect rigoureux des règles définies par le développeur (QNX, VxWorks, CE), le développeur doit assumer les risques.

Ordonnanceurs à priorités fixes

- On associe à chaque tâche un numéro d'ordre qui permet de déterminer laquelle s'exécute en priorité
- A un niveau de priorité donné on ne sera perturbé que par les traitements de niveau supérieur
  - Tâches
  - Interruptions
- Une tâche typique passe son temps à attendre un événement ou des données, puis à les traiter séquentiellement

Autres ordonnanceurs

- On associe à chaque tâche (i) un certain nombre de paramètres temporels correspondant à des échéances (Ti, ti, Ri) et des temps de traitement (Ci)
  - RMS = tâches périodiques Ti, Ci
  - EDF = tâches sporadiques ti, Ci, Ri

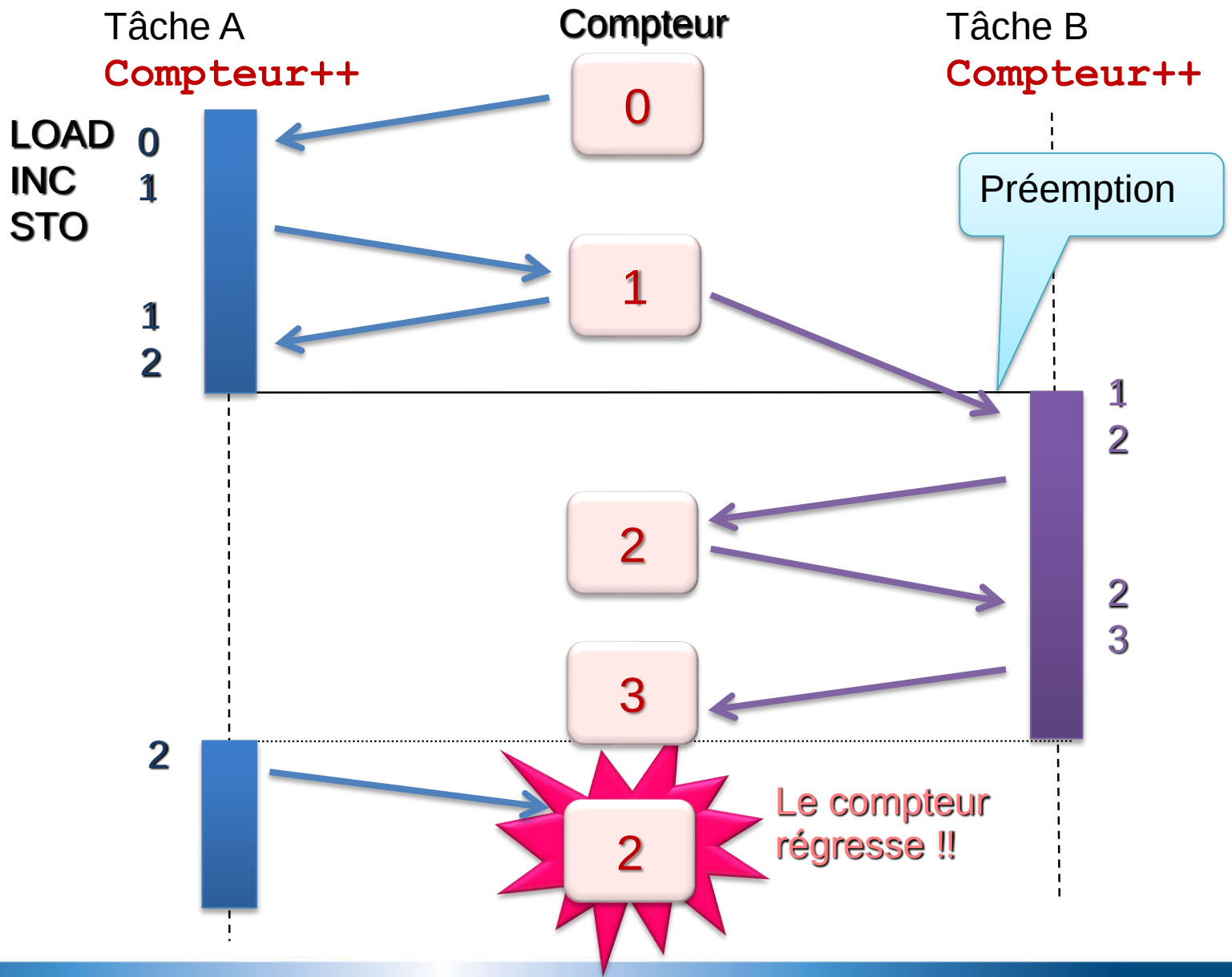
- A tout instant (TICK) l'ordonnanceur doit déterminer la tâche ayant la plus forte contrainte
  - La faisabilité peut être calculée par avance
  - Le concepteur doit connaitre le comportement de son système

Effets de bord…

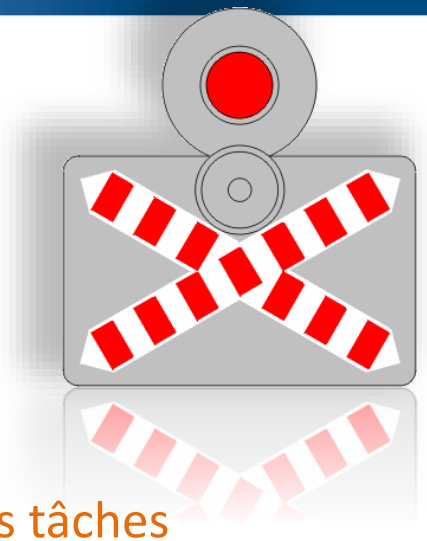
- Multitâche coopératif ou préemptif FIFO
  → FAMINE … une tâche ne tourne jamais

- Multitâche préemptif
  → ATOMICITE … une donnée est corrompue

Problème d'atomicité…



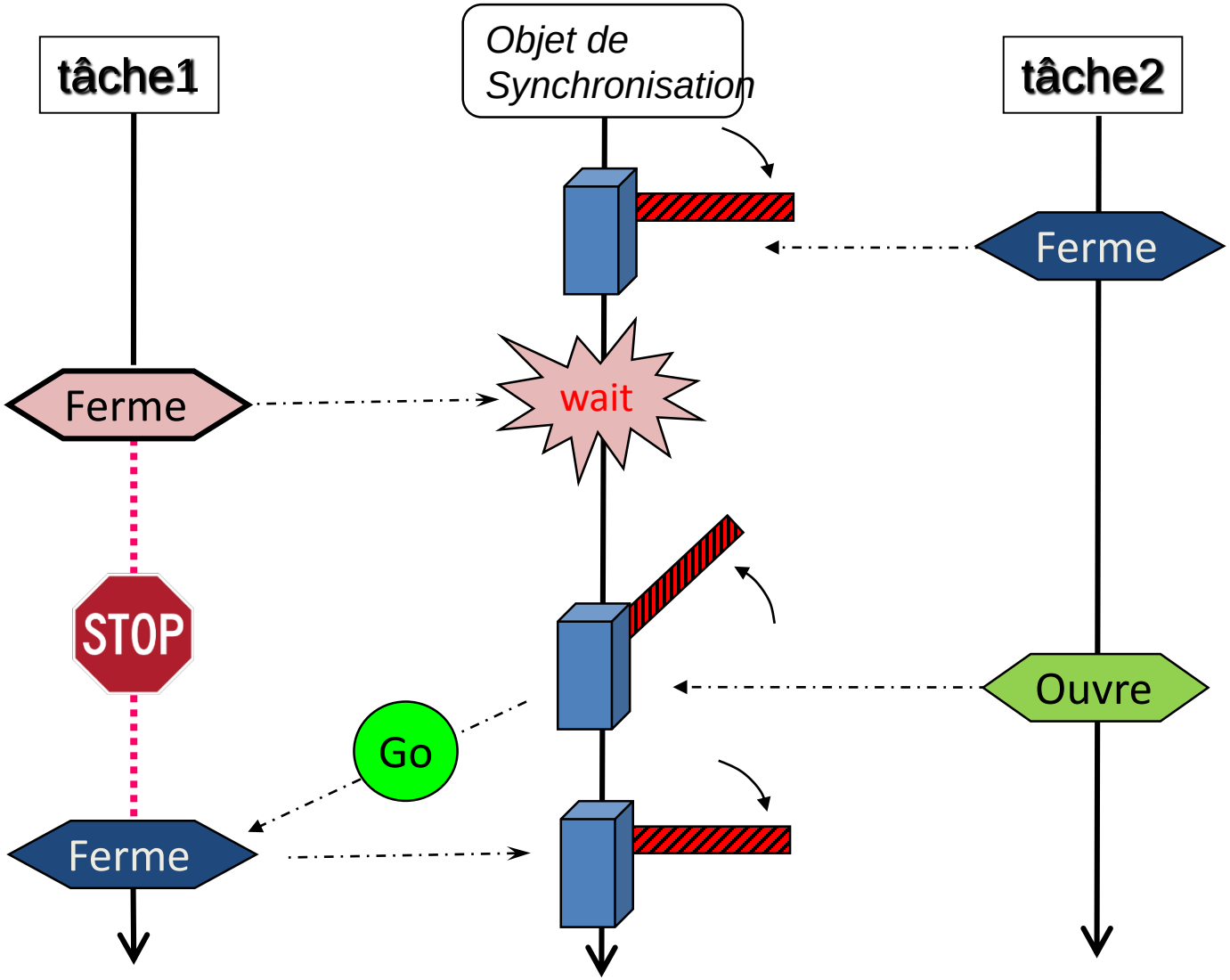
Objets de synchronisation



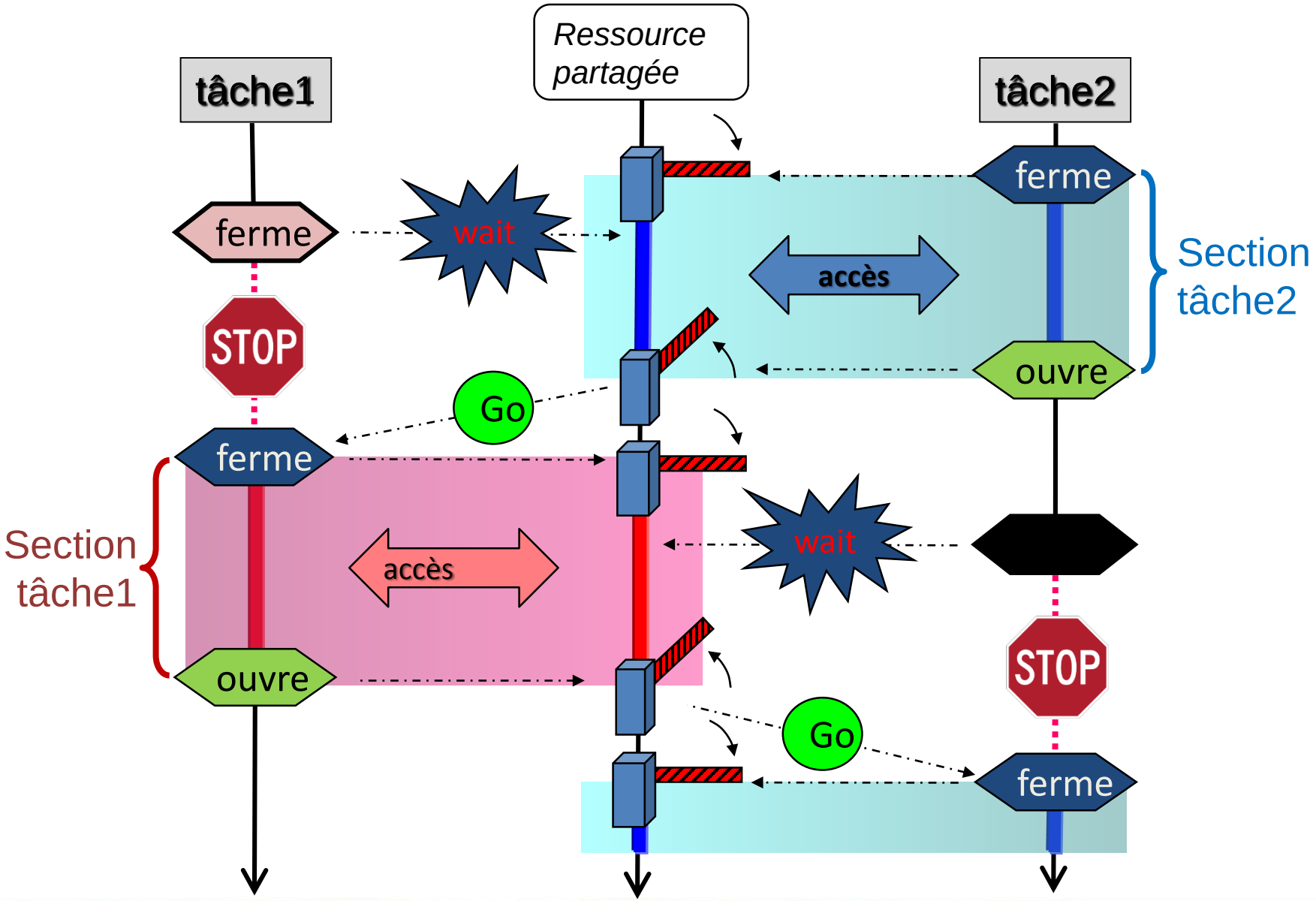
- Les objets de synchronisation codifient l'interaction entre les tâches
  - Ils permettent à une tâche de libérer le CPU lorsqu'elle n'a pas d'événements à traiter
  - Ils permettent de présenter à une tâche un accès exclusif à des données ou ressources pourtant partagées

- L'OS ayant une vue globale de l'ensemble des tâches, il est en mesure de mettre en oeuvre des mécanismes optimisés inaccessibles à une tâche isolée
  - Par exemple : l'attente active pour détecter un événement présente l'énorme inconvénient de monopoliser le CPU

Signalisation



Protection par section critique



Le mutex

- **Notion de propriétaire**

  ⇒ une seule tâche possède le mutex et peut accéder à une ressource protégée par ce mutex. Seule cette tâche peut libérer le mutex et redonner accès à cette ressource

- **Utilisation :**
  - Get(mutex)
  - Lecture, exploitation, écriture de la ressource
  - Release(mutex)

☞ Dans certains OS le mutex correspond à l'objet « sémaphore binaire »

Le sémaphore

- un compteur, initialisé à une valeur >= 0 associé à une file d'attente de tâches

- Opérations
  - Prendre → P
    - compteur = compteur - 1
    - si compteur = 0, alors mettre la tâche dans l'état EN ATTENTE et la placer dans la file d'attente
  - Libérer → V
    - compteur = compteur + 1
    - si la file d'attente est non vide, sortir une tâche de la file et l'activer

L’événement

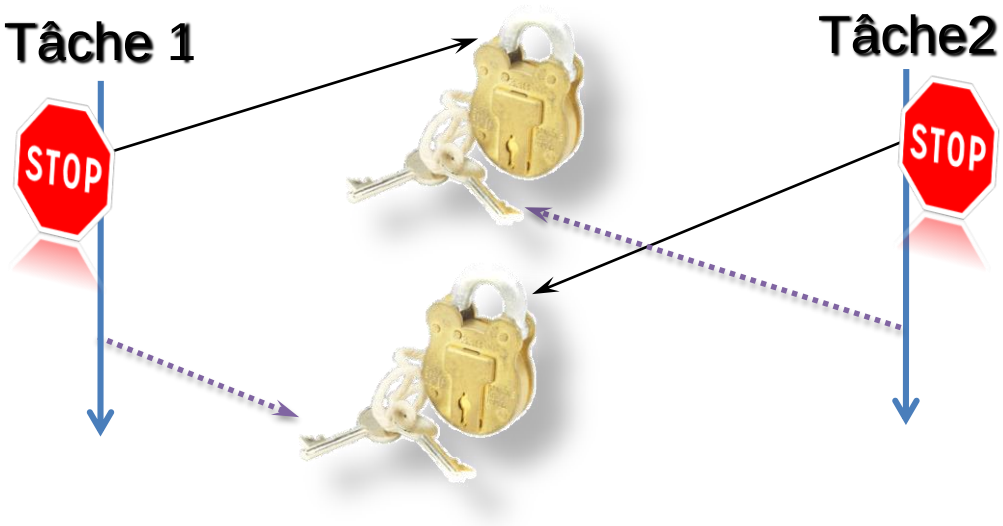
- **Opérations**
  - Signaler : Autoriser une ou plusieurs tâches à s’exécuter
  - Attendre : Blocage de la tâche par l’ordonnanceur

- **L’évènement est qualitatif et non quantitatif (contrairement au Mutex et au Sémaphore)**

- **En fonction des OS l’usage des événements peut varier :**
  - Evénements locaux à une tâche ou globaux au système
  - Evénements dont la remise à zéro est automatique ou nécessite une action spécifique (reset)

Effets de bord…

- Multitâche coopératif ou préemptif
  → DEADLOCK … attente réciproque

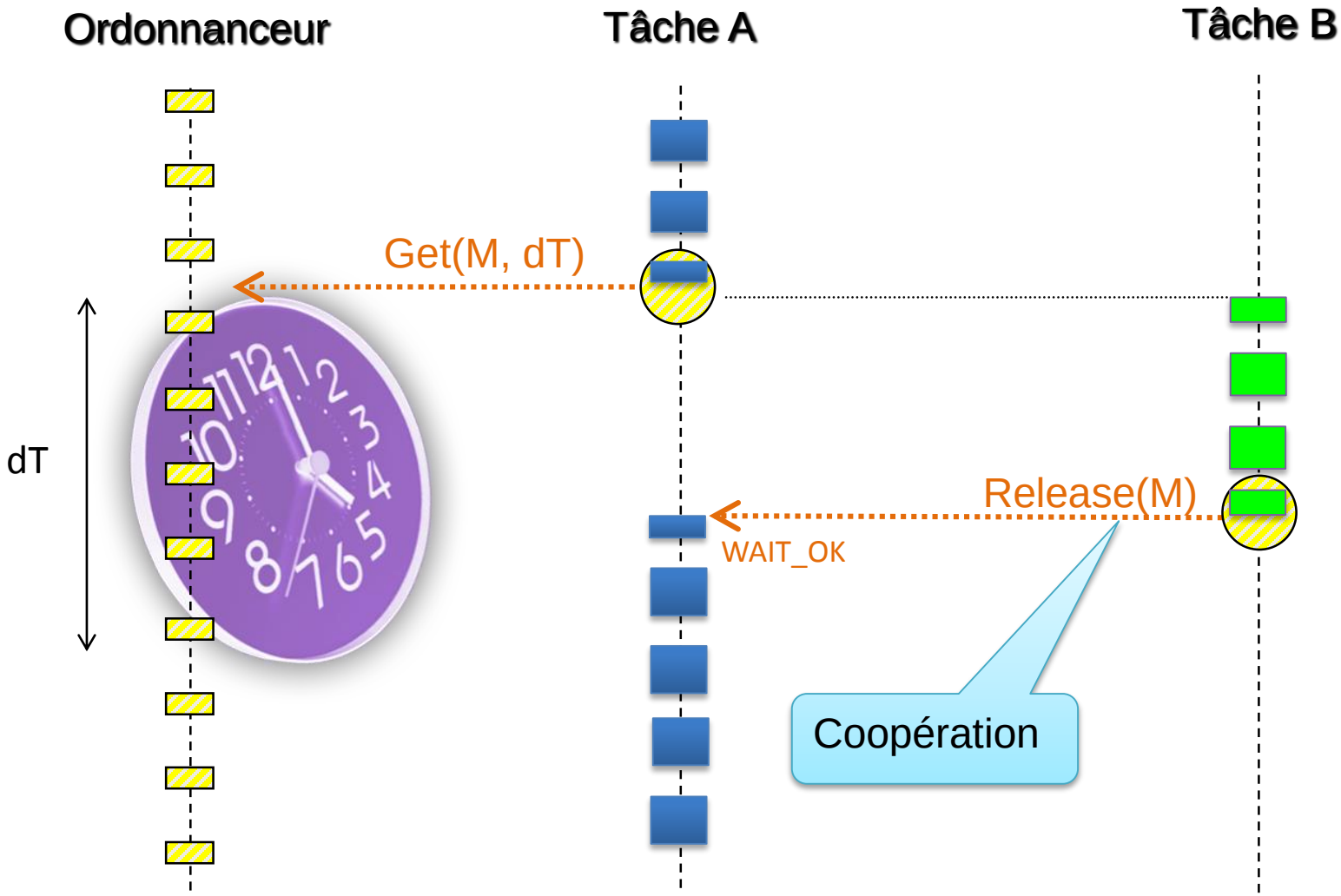


☞ **Une situation que le système ne gère pas !**

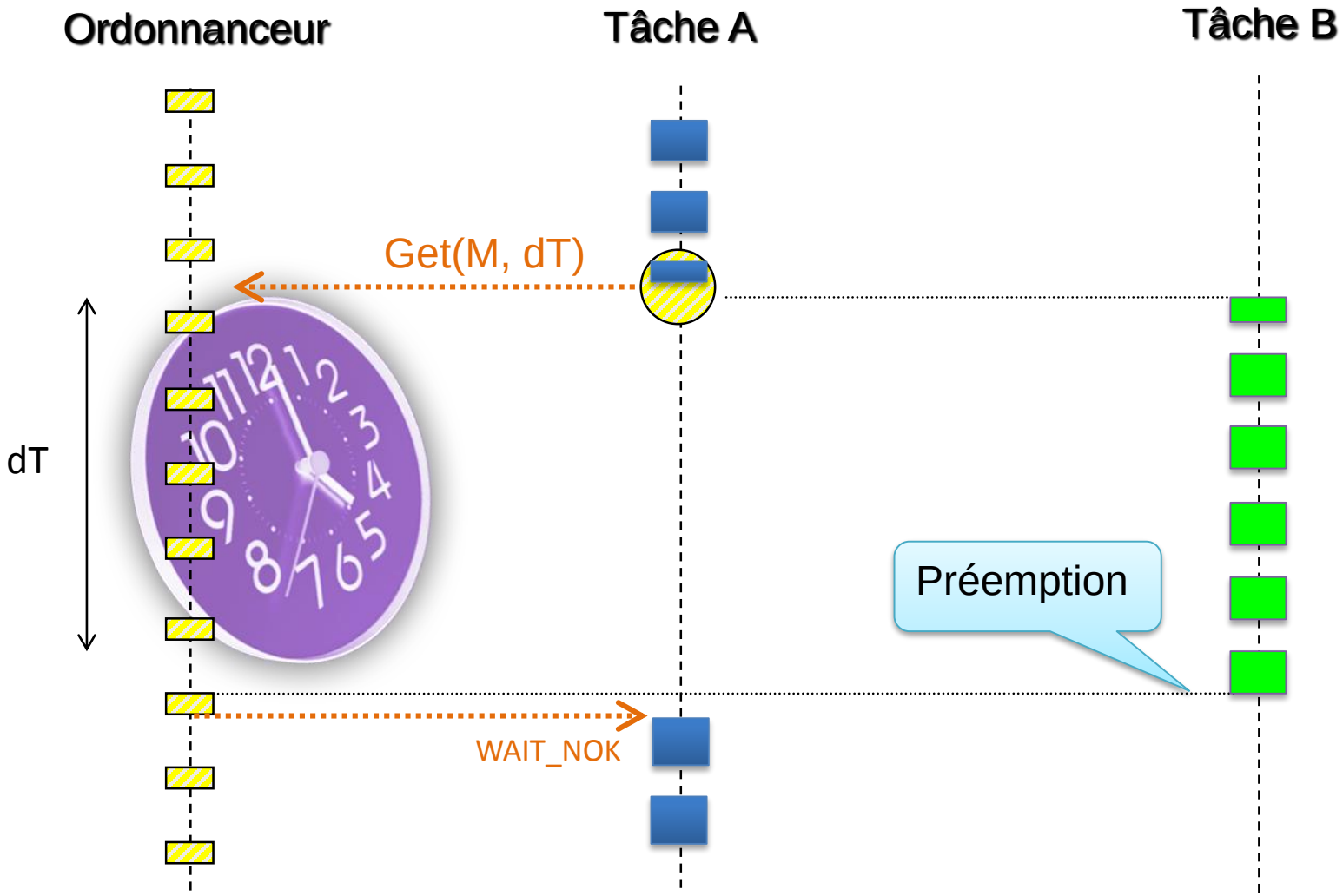
Mécanismes de déblocage

- **Watchdog Matériel**
  - Doit être prévu à la conception de la carte
  - Reset du système si un compteur n'est pas mis à jour
  - Mécanisme brutal qui n'est pas nécessairement compatible avec les fonctions d'entrée/sortie (fichiers)
- **Attente avec TimeOut**
  - Un appel coopératif bloquant avec un borne temporelle
  - Cette technique permet d'éliminer les Deadlocks mais sans éliminer leur causes!!

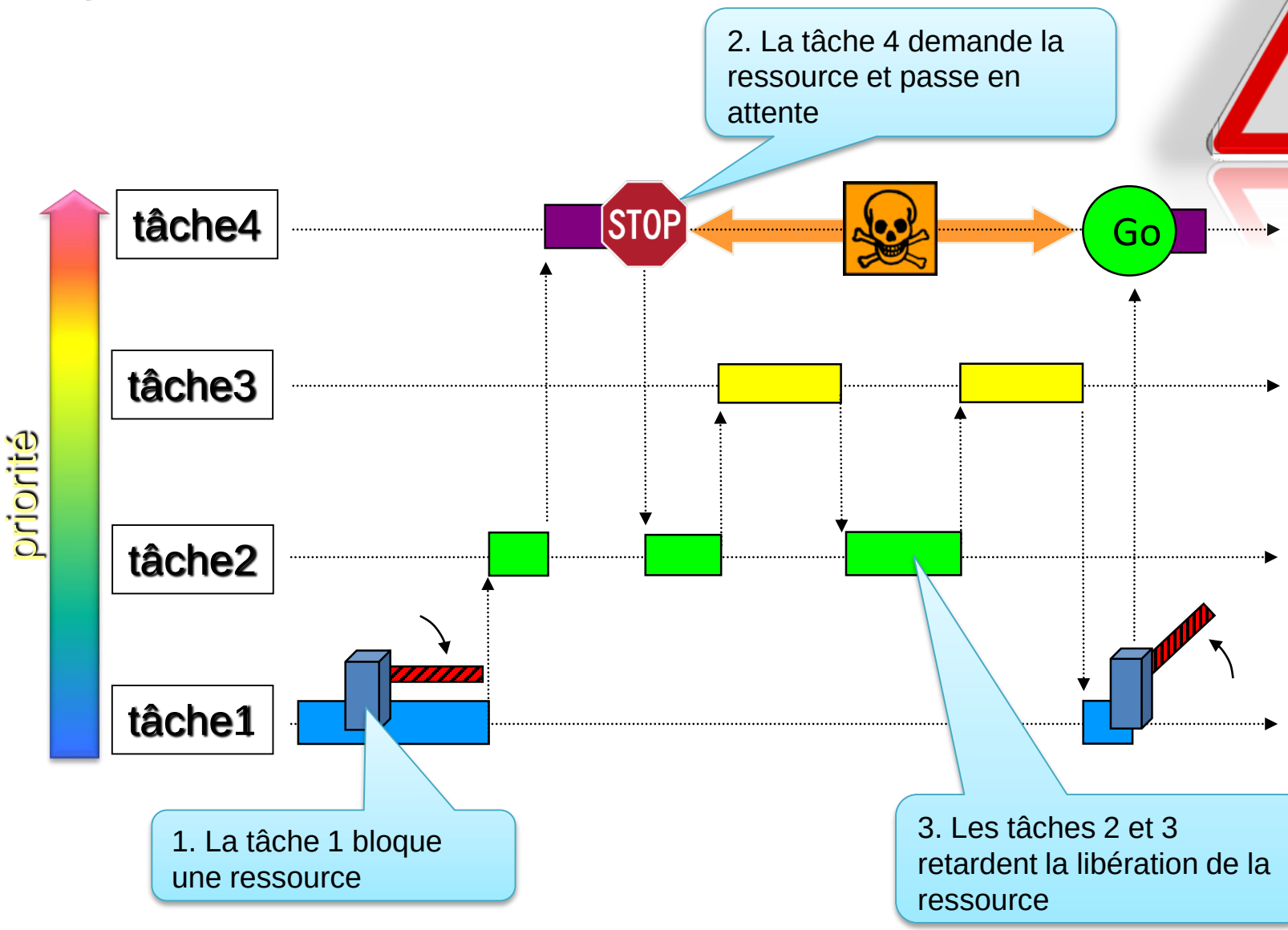
Synchronisation réussie



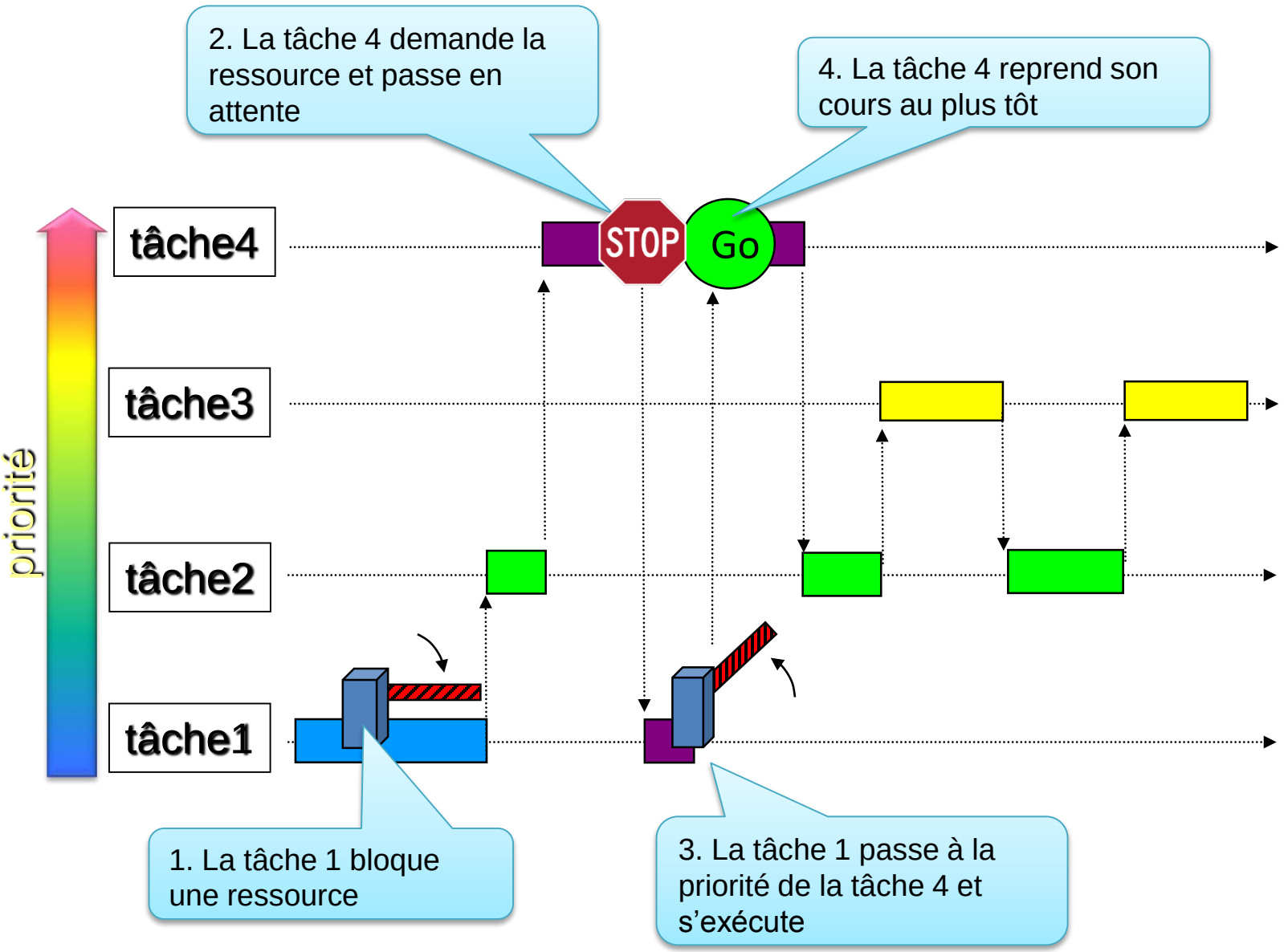
Synchronisation en Timeout



Synchronisation & Famine



Inversion de priorité ☺



Synthèse multitâches