Noyau d'un système d'exploitation
INF2610

Chapitre 9 (suite) : Ordonnancement temps réel

Département de génie informatique et génie logiciel

POLYTECHNIQUE MONTRÉAL
AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

Automne 2018

Chapitre 9 – Ordonnancement temps réel (partie 2)

- **Qu'est qu'un système d'exploitation temps réel**
- **Ordonnancement de tâches temps réel**
  - **Tâches périodiques indépendantes**
  - **Tâches périodiques dépendantes**

Qu'est ce qu'un système d'exploitation temps réel ?

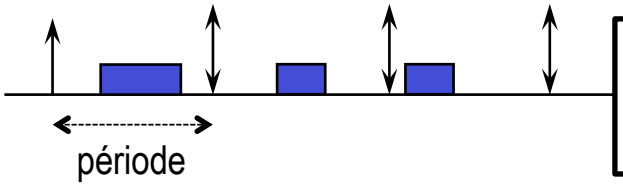
- "Real time in operating systems (RTOS):
  The ability of the operating system to provide a required level of service in a bounded response time." POSIX Standard 1003.1
- Le noyau d'un RTOS assure les fonctions de base d'un système d'exploitation (gestion des tâches, gestion des interruptions; gestion du temps (minuteries); gestion de la mémoire, etc.).
- Il doit, en plus,
  - **garantir des temps de réponses bornés et acceptables aux demandes de services, et**
  - **permettre de développer et mettre en œuvre des applications temps réel :**

  { tâches concurrentes, communicantes avec éventuellement des **contraintes temporelles** }.

Qu'est ce qu'un système d'exploitation temps réel ? (2)

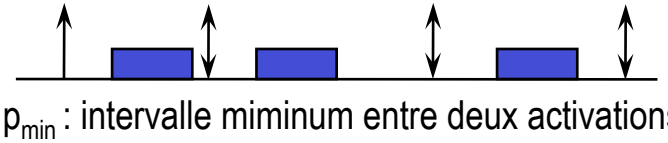
Tâches temps réel : Contraintes temporelles

- Tâche périodique

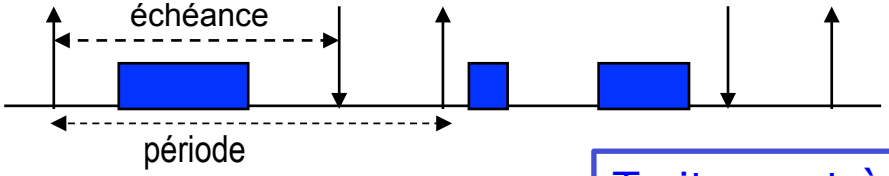


Traitement à réaliser avant la prochaine activation de la tâche.

- Tâche apériodique (sporadique)

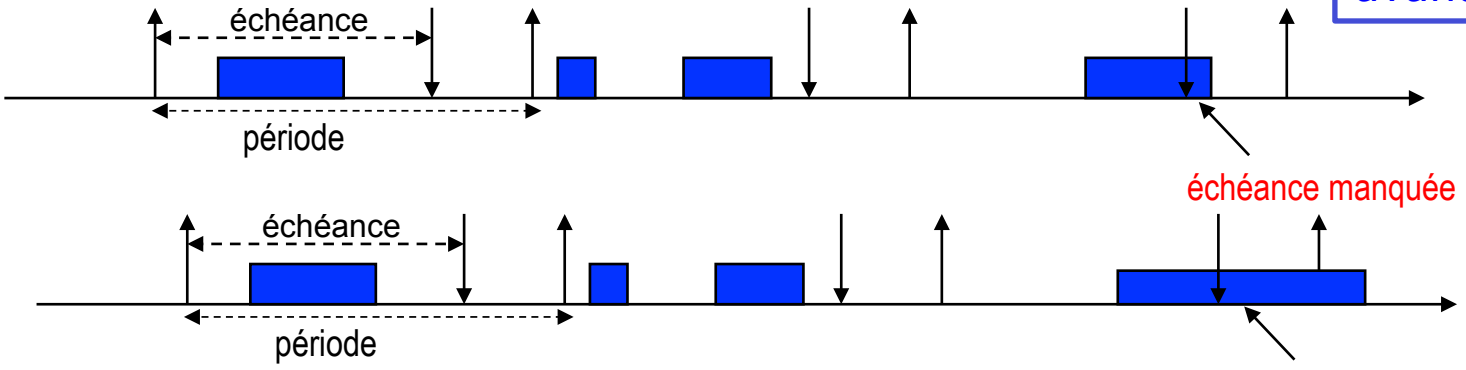


- Tâche périodique avec échéance



➔ Problème d'échéance manquée

Traitement à réaliser avant l'échéance.



Qu'est ce qu'un système d'exploitation temps réel ? (3)

Tâches temps réel : Contraintes temporelles

- Période d'activation : Pi
- Temps d'exécution de chaque tâche (au pire cas) à chaque activation (traitement périodique) : Ci
- Échéance (deadline) de chaque tâche : Di
- Tâches qui peuvent démarrer durant toute la période ou doivent démarrer durant les x premiers cycles de la période.
- Date d'activation initiale : Oi
- Tâche à échéance sur requête *:* Di = Pi
- Tâches synchrones Oi = Oj, pour tout (i, j)

Qu'est ce qu'un système d'exploitation temps réel ? (4)

- Comment garantir des temps de réponse bornés et satisfaire les contraintes temporelles ?

  ➔ Interdire tous les appels bloquants dans les ISR (Interrupt Service Routine) ou dans les tâches associées aux interruptions.

  ➔ Réduire et borner les durées de masquage des interruptions, le temps nécessaire pour déterminer la prochaine tâche à élire, etc.

  ➔ Exécuter les tâches temps réel dans l'espace noyau.

  ➔ noyau préemptif.

  ➔ Ordonnancement préemptif à base de priorités qui permet de tenir compte des contraintes temporelles (activation périodique d'une tâche, date de première activation, etc.).

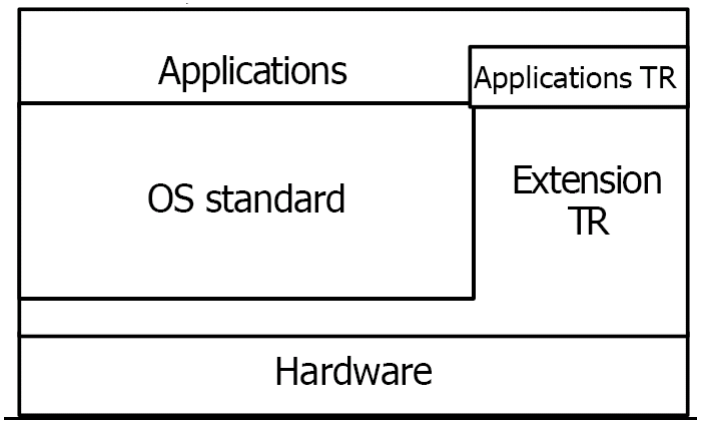
- Le bon fonctionnement d'une application temps réel est, notamment, conditionné par le respect de leurs contraintes temporelles.

Qu'est ce qu'un système d'exploitation temps réel ? (5)

- **Linux** n'est pas un système d'exploitation Temps Réel (dur) car le noyau Linux possède de longues sections de code où tous les événements extérieurs sont masqués (non interruptibles).

- Systèmes d'exploitation temps réel dédiés :
  - VxWorks (INF3610),
  - MicroC (INF3610),
  - QNX,
  - LynuxWorks, etc.

- Les extensions existantes de cohabitation du noyau Linux avec un micronoyau :
  - **RT-Linux**,
  - RTAI, etc.



Ordonnancement de tâches temps réel
RT-Linux (http://www.mnis.fr/ocera_support/rtos/c1450.html)

- Les tâches RT-Linux s'exécutent dans l'espace noyau au même titre que l'ordonnanceur temps réel.

- La structure « rt_task_struct » (ou RT_TASK) des tâches temps réel dans RT-Linux contient notamment les champs suivants :
  - État (inactive, ready, active, delayed, zombie);
  - Priorité;
  - Période;
  - Date de la prochaine activation.

Ordonnancement de tâches temps réel RT-Linux (2)

- Deux types de tâches sont gérées dans RT-Linux :

- Tâches périodiques :
  - exécutent périodiquement un traitement.
  - sont activées périodiquement par le système.

- Tâches apériodiques :
  - exécutent à des intervalles de temps irréguliers un traitement.
  - sont généralement activées par d'autres tâches (relation de précédence ou arrivée d'un événement).

Ordonnancement de tâches temps réel RT-Linux (3)

- Création d'une tâche (état inactive) :

  int rt_task_init (RT_TASK *task, void (fn)(int data), int data,

  int stack_size, int priority);

- Activation d'une tâche (état ready) :

  int rt_task_wakeup (RT_TASK *task);

- Rendre une tâche inactive (état inactive) :

  int rt_task_suspend (RT_TASK *task);

Ordonnancement de tâches temps réel RT-Linux (4)

- Transformation d'une tâche créée en une tâche périodique :

  int rt_task_make_periodic(RT_TASK **task*, RTIME *start_time*, RTIME *period*

Cette fonction doit être appelée après sa création (pas après son activation).

- Suspension d'une tâche périodique jusqu'à sa prochaine date de réveil (état delayed) :

  int rt_task_wait (void);

- Suppression d'une tâche RT-Linux (état zombie) :

  int rt_task_delete (RT_TASK *task);

Ordonnancement de tâches temps réel
RT-Linux (5)

- Ordonnancement (par défaut) préemptif à priorités statiques

  → Il sélectionne la tâche prête la plus prioritaire.

- Ordonnancement Rate Monotonic priority assignment (RMA)

- Ordonnancement Earliest Deadline First (EDF)

Rate monotonic priority assignment (RMA) [Liu & Layland, 1973]

- La priorité d'une tâche est inversement proportionnelle à sa période. Plus la période est petite, plus la priorité est grande => Priorité statique.
- Un ensemble de n tâches périodiques indépendantes où Di=Pi, est ordonnançable si (**condition suffisante** de *Liu et Layland*) :

(https://www.di.ens.fr/~pouzet/cours/systeme/bib/liu73.pdf)

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{1/n} - 1)$$

n	Utilization Bound
1	100.0%
2	82.8%
3	78.0%
4	75.7%
5	74.3%
10	71.8%

n → ∞ => borne = 69.3 %

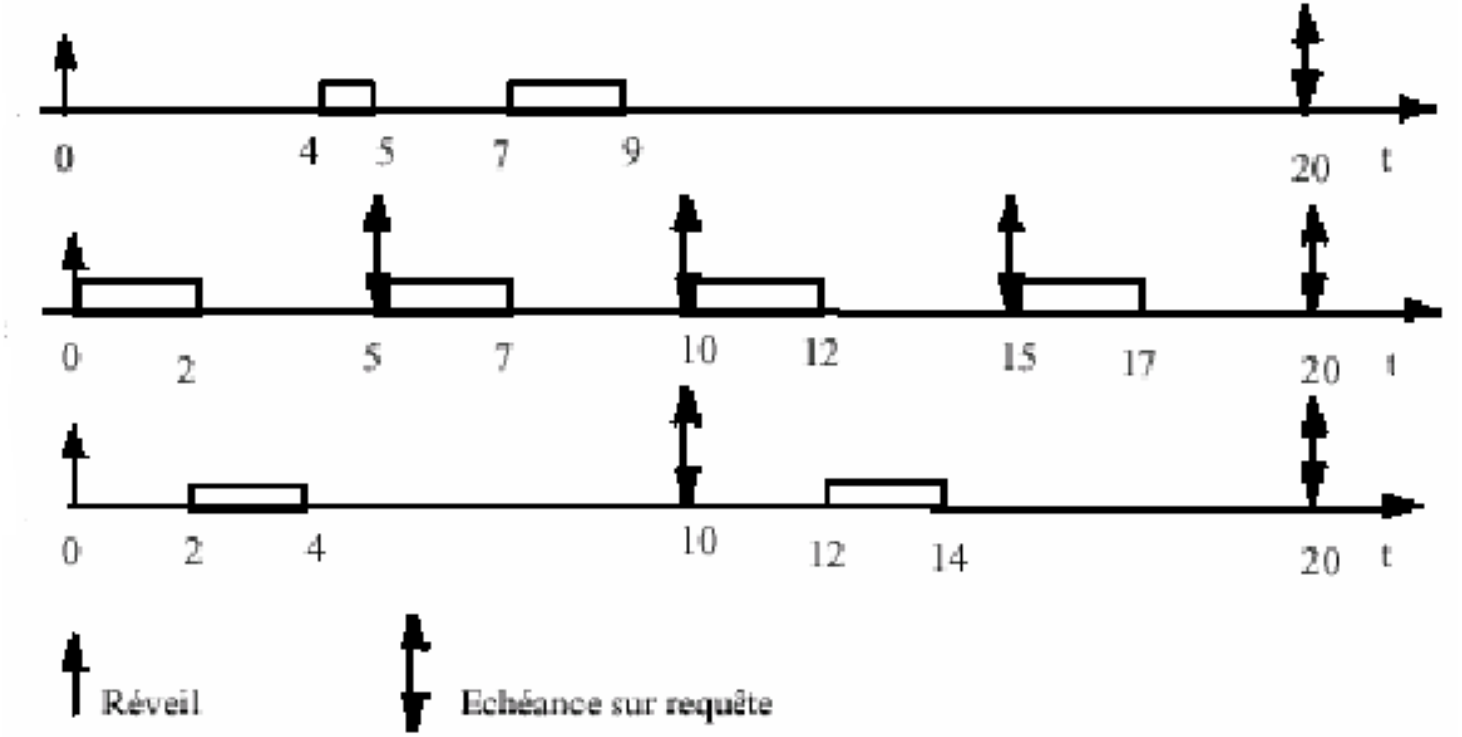
Ui = Ci / Pi Taux d'occupation processeur (ou charge) par tâche.

Rate monotonic priority assignment (RMA) (2)
Exemple 1

T1 : C1=3 P1=20 T2 : C2=2 P2 = 5
T3 : C3=2 P3 = 10 prio(T1) < prio(T3) < prio(T2)

3/20 + 2/5 + 2/10 = 75 % < 78 % => ordonnançable

Simulation
Intervalle d'étude :
[0, PPCM(P1,P2,P3)]



Deadline monotonic priority assignment (DMA) [Leung & Whitehead, 1985]

- La priorité d'une tâche est inversement proportionnelle à son *deadline (*plus le *deadline* est petit, plus la priorité est grande) → *Priorité statique*.

- Le conflit est résolu de façon arbitraire.

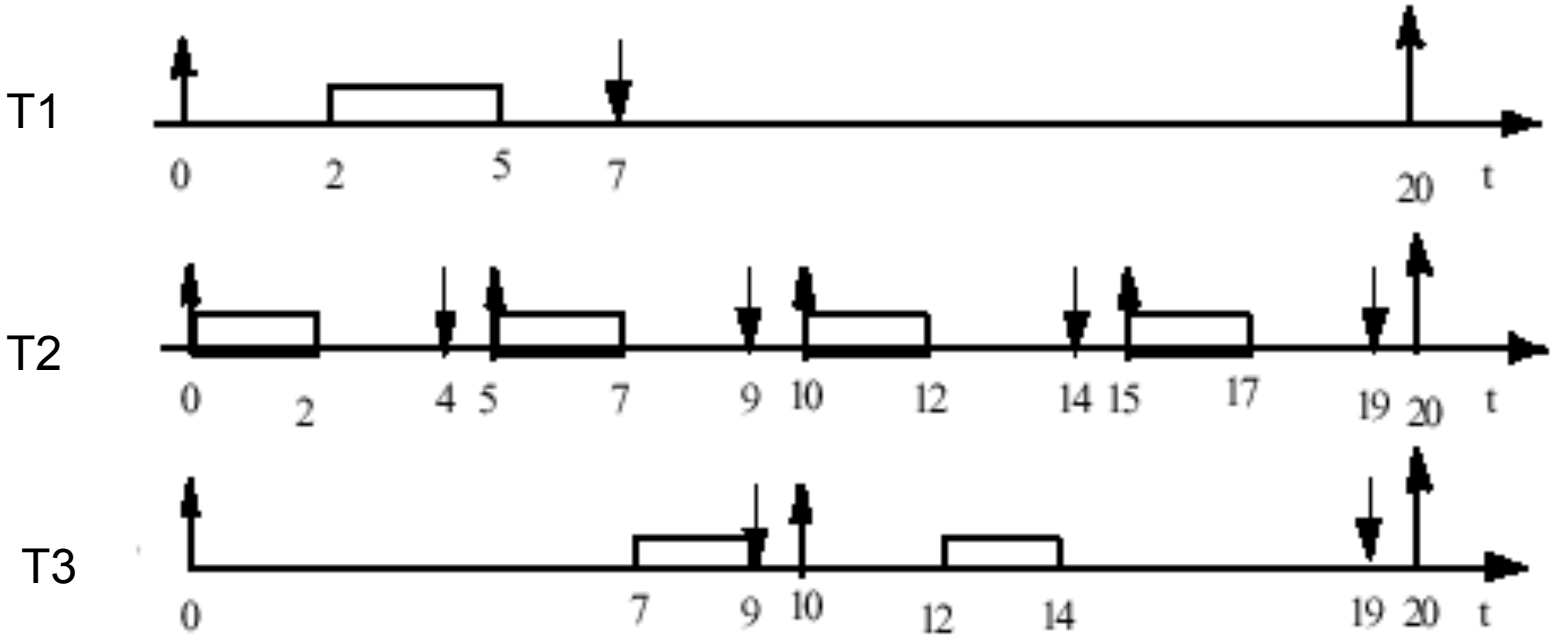
- Un ensemble de n tâches périodiques indépendantes est ordonnançable *si :*

$$\sum_{i=1}^{n} \frac{Ci}{Di} \leq n(2^{\frac{1}{n}} - 1)$$

Deadline monotonic priority assignment (DMA) (2) Exemple 2

T1 : C1=3 D1=7 P1=20 T2 : C2=2 D2=4 P2 = 5
T3 : C3=2 D3=9 P3 = 10 prio(T3) < prio(T1) < prio(T2)

3/7 + 2/4 + 2/9 = 115 % non < 78 % on ne peut pas conclure



➔ Ordonnançable

Earliest Deadline First (EDF) [Liu & Layland, 1973]

- La tâche la plus prioritaire (parmi les tâches prêtes) est celle dont l'échéance absolue est la plus proche ➔ priorités dynamiques.

- Il est applicable aussi bien pour des tâches périodiques qu'apériodiques.

- Ordonnançable :

  Pour les tâches à échéance sur requête (Pi=Di pour toute tâche Ti),

  CNS : $\sum_{i=1}^{n} \frac{Ci}{Pi} \leq 1$

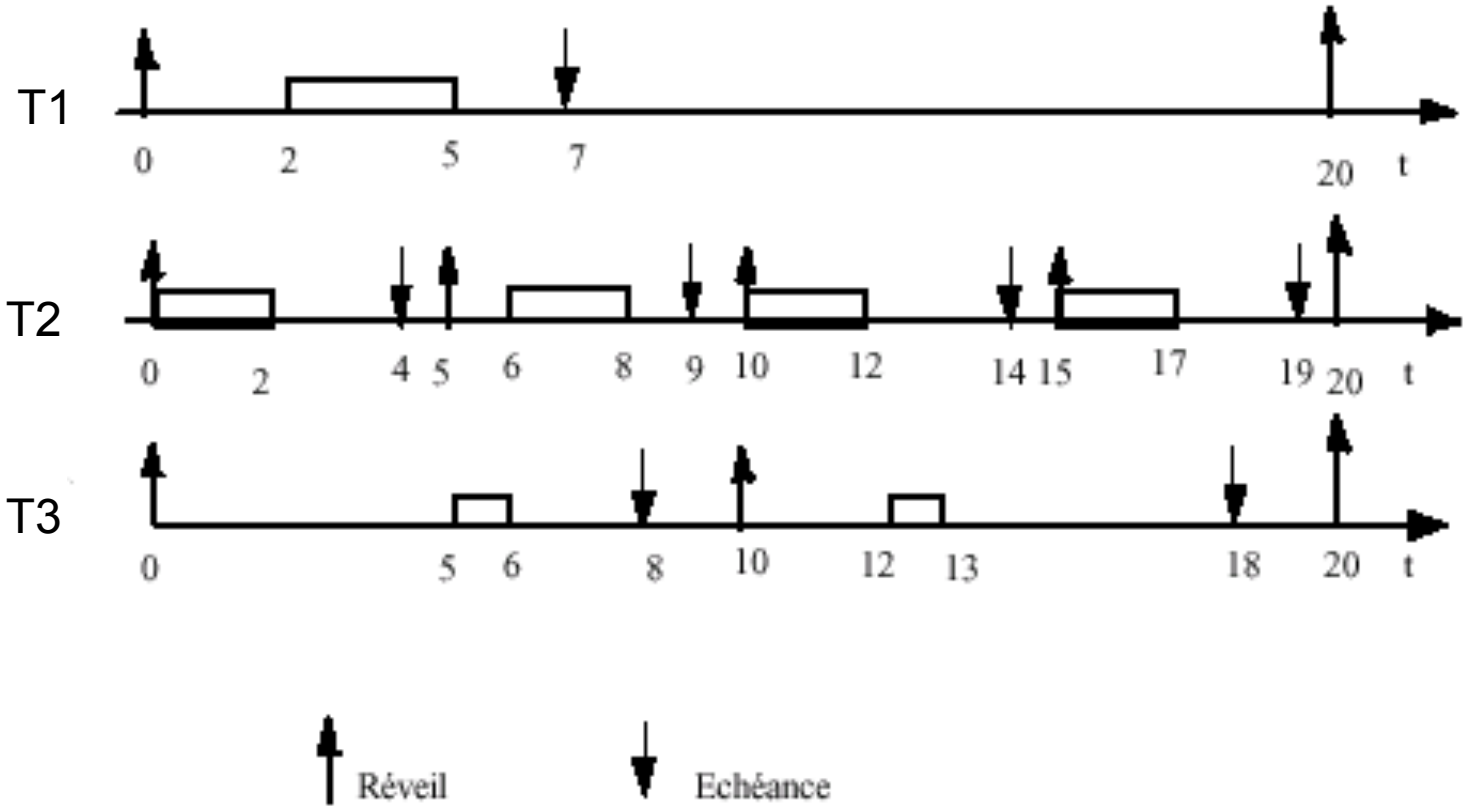
  Pour les autres cas :

  CN : $\sum_{i=1}^{n} \frac{Ci}{Pi} \leq 1$

  CS : $\sum_{i=1}^{n} \frac{Ci}{Di} \leq 1$

Earliest Deadline First (EDF) (2)
Exemple 3

T1 : C1=3 D1=7 P1=20 T2 : C2=2 D2=4 P2 = 5
T3 : C3=1 D3=8 P3 = 10
3/7 + 2/4 + 1/8 = 105 % non ≤ 1 on ne peut pas conclure



Ordonnancement en présence de ressources partagées

Problèmes de partages de ressources

- Les tâches partagent des ressources à accès exclusif.
- Lorsqu'une tâche Tx demande une ressource déjà allouée à une autre tâche Ty, Tx se met en attente de la ressource.
- Ordonnancement à priorité

⇒ Inversion de priorités possible : Une tâche de basse priorité empêche une tâche plus prioritaire d'accéder à sa section critique (bloque une tâche plus prioritaire).

⇒ Interblocage

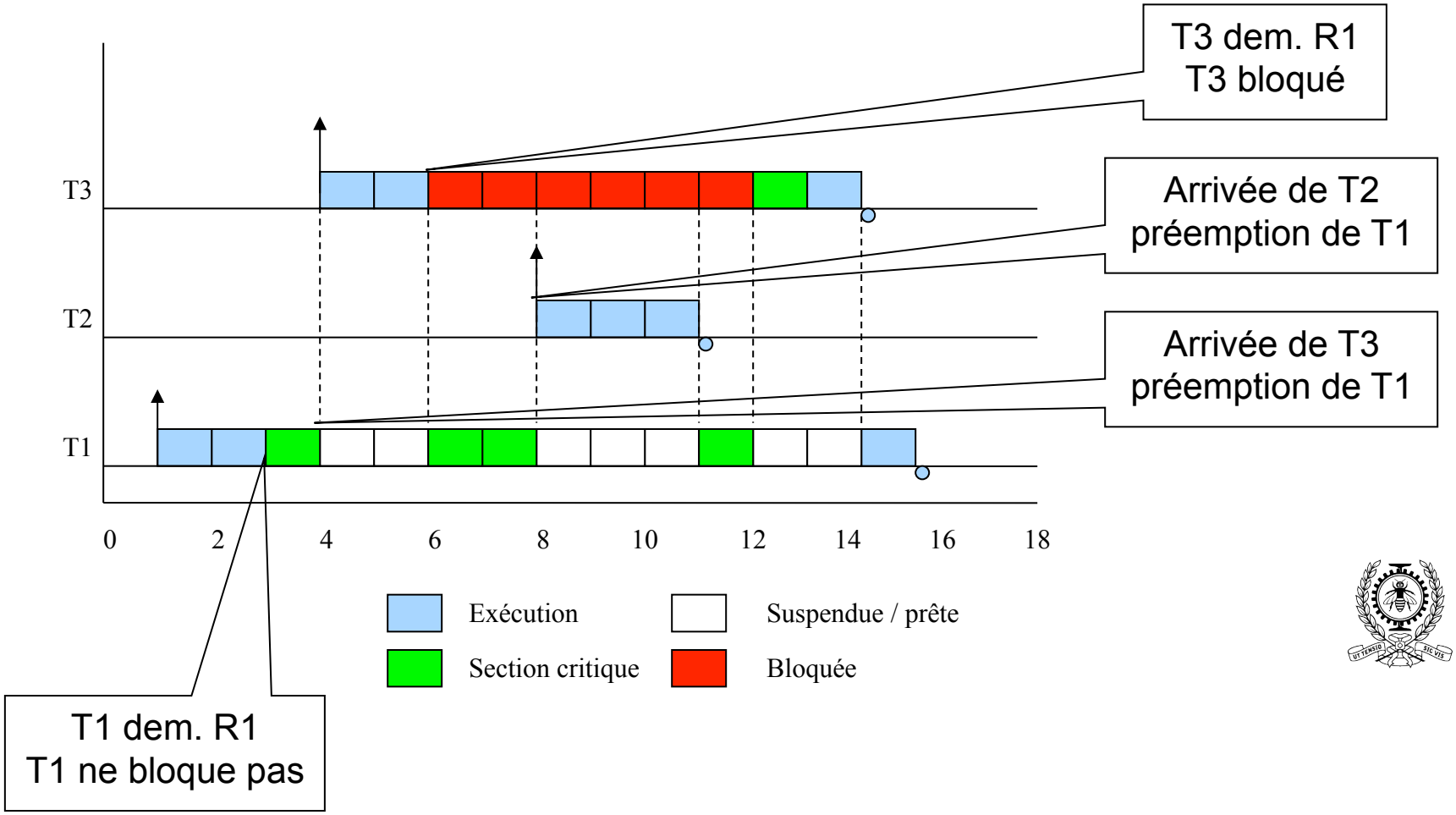
Problèmes de partages de ressources
Inversion de priorités

Exemple 4 :

Tâche	Date de départ	Priorité	Séquence d'exécution
T3	4	3	EER1E
T2	8	2	EEE
T1	1	1	EER1R1R1R1E

Problèmes de partages de ressources
Inversion de priorités (2)

Exemple 4 (suite) : Trois tâches : T1, T2, T3 → prio(T1)<prio(T2)< prio(T3)



Problèmes de partages de ressources
Inversion de priorités (3)

Traitement du problème d'inversion de priorités :

- Protocole d'héritage de priorités [Kaiser, 1982] [Sha, 1990]
- OCPP (Original Ceiling Priority Protocol) [Chen, 1990] [Sha, 1990]
- ICPP (Immediate Ceiling Priority protocol)

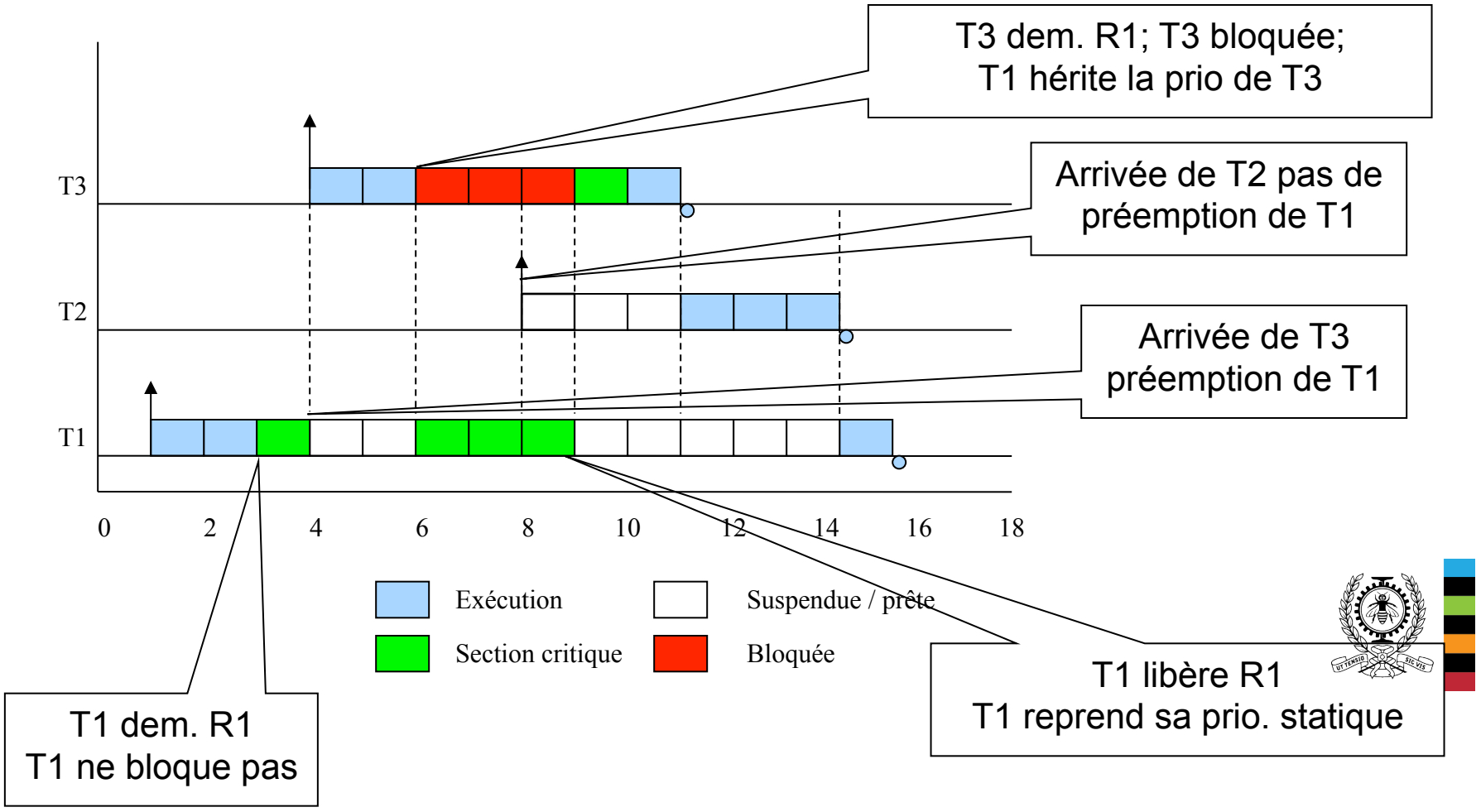
Protocole d'héritage de priorités (PIP = Priority Inheritance Protocol)

- Lorsqu'une tâche Tx bloque suite à une demande d'accès à une ressource R, la tâche bloquante Ty (celle qui détient R) hérite la priorité courante de Tx.

- La tâche Ty s'exécutera avec la priorité héritée jusqu'à la libération de la ressource.

- Elle retrouve ensuite sa priorité.

Protocole d'héritage de priorités (2) (PIP = Priority Inheritance Protocol)



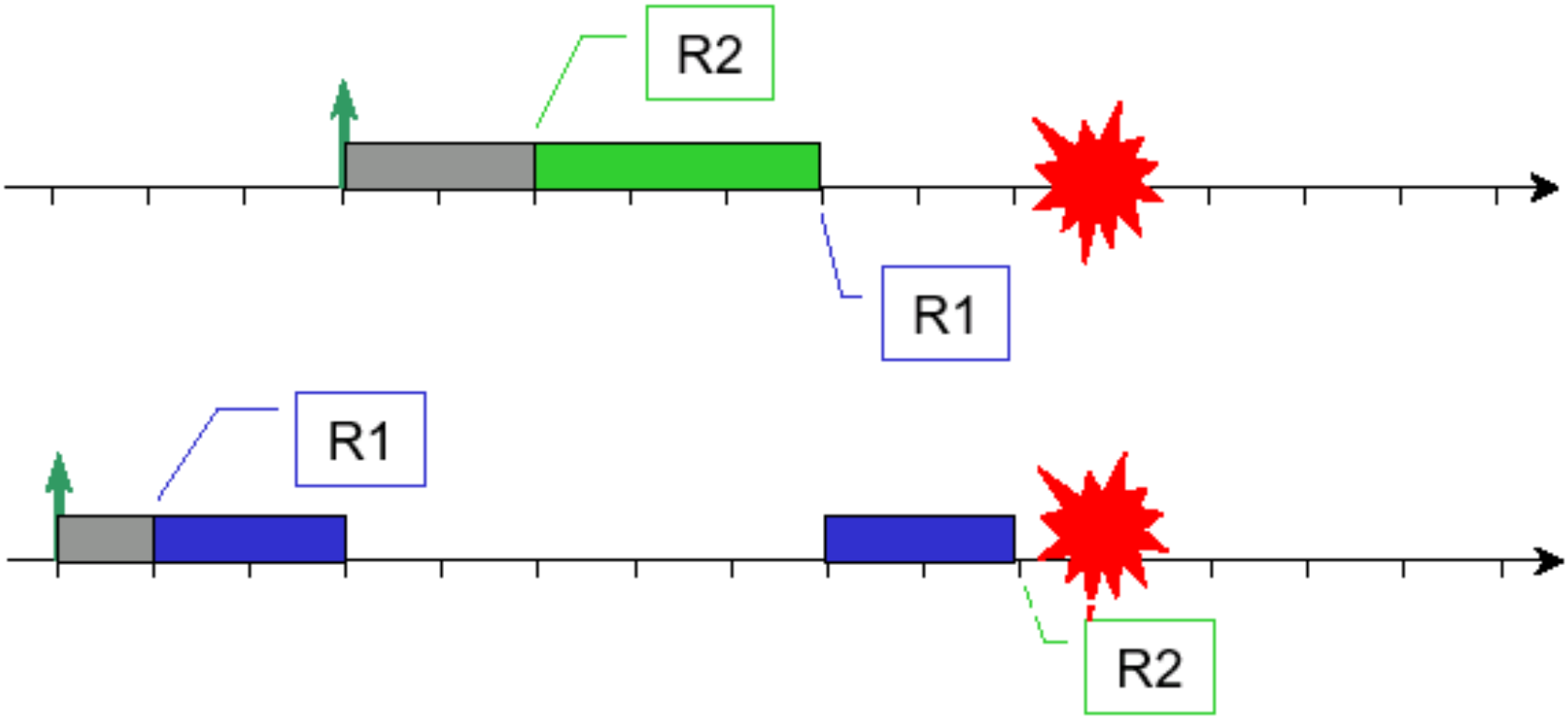
Protocole d'héritage de priorités (3) (PIP = Priority Inheritance Protocol)

- Ce protocole peut mener à un Interblocage : Deux tâches T1 et T2 utilisant deux ressources R1 et R2; prio(T1) < prio(T2); T1 est lancée avant T2.

Tâche T1	Tâche T2
Demander(R1);	Demander(R2);
/* utiliser R1	/* utiliser R2
Demander(R2) ;	Demander(R1) ;
/* utiliser R1 et R2	/* utiliser R1 et R2
Liberer(R2);	Liberer(R1);
Liberer(R1);	Liberer(R2) ;

- Temps de blocage est borné (sauf en cas d'interblocage) → PIP permet d'évaluer cette borne Bi (au pire cas) pour chaque tâche Ti.
- PIP n'élimine pas l'inversion de priorité mais il permet de la réduire.

Protocole d'héritage de priorités (4)
Problème d'interblocage



Exercice (RMA + PIP)

Tâche	Date d'arrivée	Temps d'exécution Ci	Période Pi
T3	3	3: ER1E	5
T2	4	2: EE	10
T1	1	5 : ER1R1R1E	20

- Les tâches T1, T2 et T3 sont-elles ordonnançables, selon RMA ?
- A-t-on un problème d'inversion de priorité ? Si oui, donnez le diagramme de Gantt correspondant au cas où ce problème est traité en utilisant le protocole PIP (Priority Inheritance Protocol). A-t-on des échéances manquées, dans ce cas ?
- Mission Mars Pathfinder lancée par la NASA (décembre 1996)
  http://www.fil.univ-lille1.fr/~nebut/portail/svl/fichiers/tp/tpPathfinder/tpPathfinder.pdf

  Tâche météo (la moins prioritaire)
  Tâche gestion du bus (la plus prioritaire)
  Tâche de communication (priorité intermédiaire)
  (http://degeeter.pagesperso-orange.fr/inv_fr.htm)

Suite : Cours INF3610