SE206: Modélisation, génération de code et vérification

Analyse de temps de réponse

Etienne Borde
etienne.borde@telecom-paristech.fr

Objectifs du cours

- Comprendre les objectifs et hypothèses d'une analyse de temps de réponse, ainsi qu'un type de modèle associé
- Savoir calculer le temps de réponse d'une tâche selon la technique d'analyse dite « RTA », pour une classe d'algorithmes d'ordonnancement très classiques (FPS, RMS)
- Connaître les limitations de cette technique

Remarques sur ce cours

- On ne verra qu'un petit sous-ensemble des techniques d'analyse temporelle des systèmes embarqué.
  - ⌘ Une seul classe d'algorithme d'ordonnancement
  - ⌘ Un seul test d'ordonnancement
  - ⌘ Une seule politique de protection des données

- Des UEs de 3ème année (notamment SE301) permettent d'aller plus loin.

- Le but est d'introduire UNE technique d'analyse avant de l'utiliser sur des modèles d'architecture…

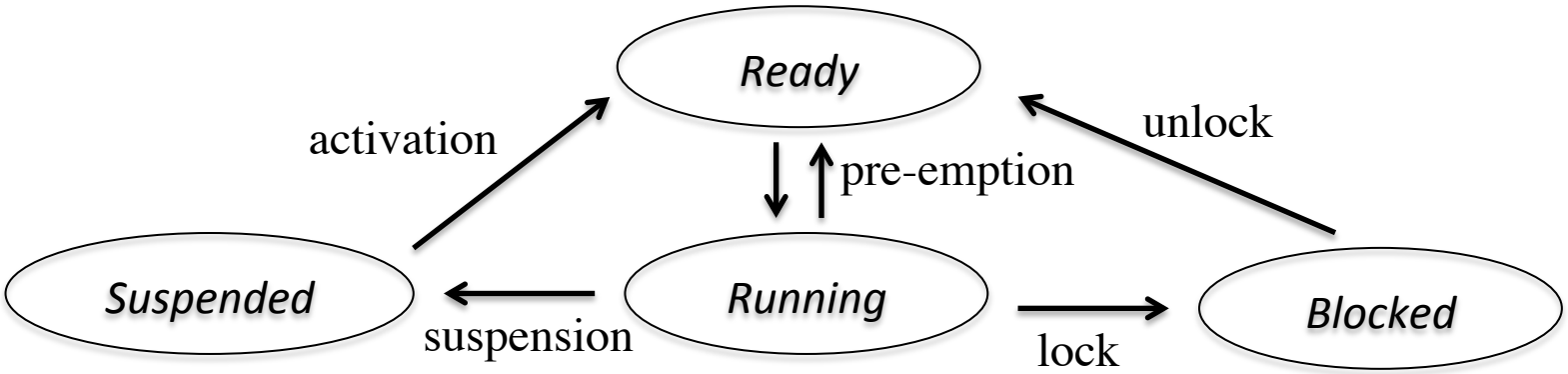
Plan du cours

1. Présentation du modèle de tâches
2. Formule de calcul du pire temps de réponse
3. Prise en compte du temps de blocage (verrous pour données partagée)

Modèle de tâche et d'ordonnancement

Les états d'une tâches

- On considère un système dans lequel un seul processus est constitué de N tâches (ou *threads*)
- Une tâche a quatre états:



Remarques:

- les état « Bloqué » et « Suspendue » sont très similaires, et auraient pu être confondus. On distingue cependant ces deux états pour identifier
  - le cas où la tâche a fini son exécution et attend la prochaine activation (*suspended*)
  - le cas où la tâche est bloquée sur une entrée/sortie (*blocked*) mais n'a pas fini son exécution.
- Le seul cas de blocage considéré dans ce cours correspond à l'usage de verrous
- Les tâches *ready* sont répertoriés dans une *« ready queue »:* une ou plusieurs files d'attente.

Modèle de tâches

- Une tâche τ est:
  - Périodique: τ est activée avec un délais ***constant*** entre deux activation. Cela permet de traiter l'activation de tâches récurrentes, comme les tâches de contrôle/commande (exemple du segway vu en introduction).
  - Sporadique: τ est activée avec un délais ***minimal*** entre deux activation. Cela permet de traiter des évènements rares tout en partageant l'accès à la ressource de calcul (exemple, détection d'obstacles).

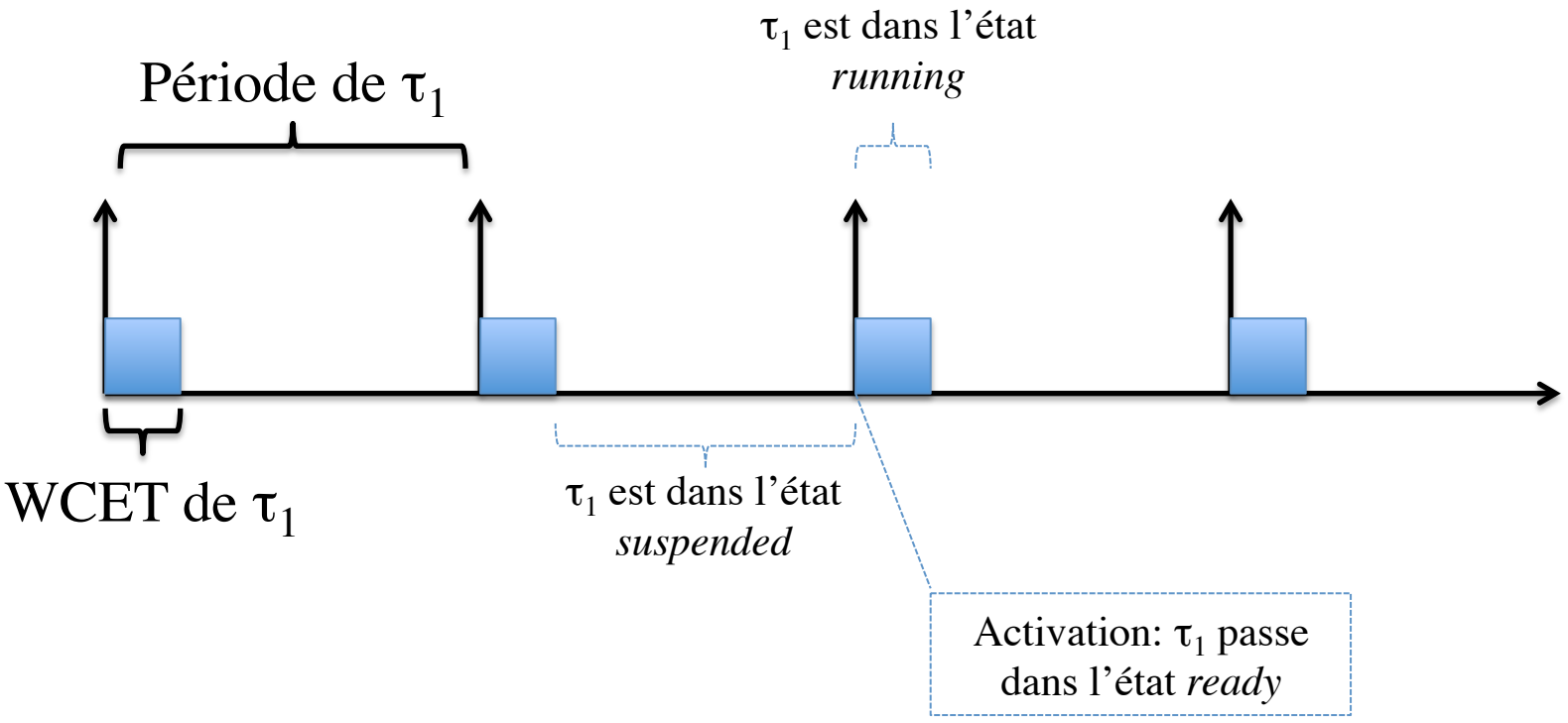
- Un système temps réel est composé d'un ensemble de N tâches $\{\tau_i\}_{0<i\leq N}$, caractérisés par:
  - T_i : période, ou intervalle de temps entre deux réveils de τ_i
  - C_i : pire temps d'exécution (WCET) de τ_i isolée des autres tâche, et ininterrompue.
  - D_i : échéance de τ_i

- On suppose que toutes le tâches commencent leur exécution à la même date théorique t=0 unité de temps

- On néglige le temps de changement de contexte

Exemple (très simple!) de représentation classique d'un modèle de tâche

Tâche	Période	WCET	Deadline
τ_1	10 ms	2 ms	6 ms



Ordonnancement

- Ordonnanceur = composant logiciel qui choisit, à certains moments, quelle tâche est exécutée par le processeur

- Algorithme d'ordonnancement = algorithme qu'utilise l'ordonnanceur pour le choix.

- Test d'ordonnancement = formule qui, pour un algorithme d'ordonnancement et un ensemble de tâches, donne une condition nécessaire et/ou suffisante pour garantir que les contraintes temporelles seront satisfaites.

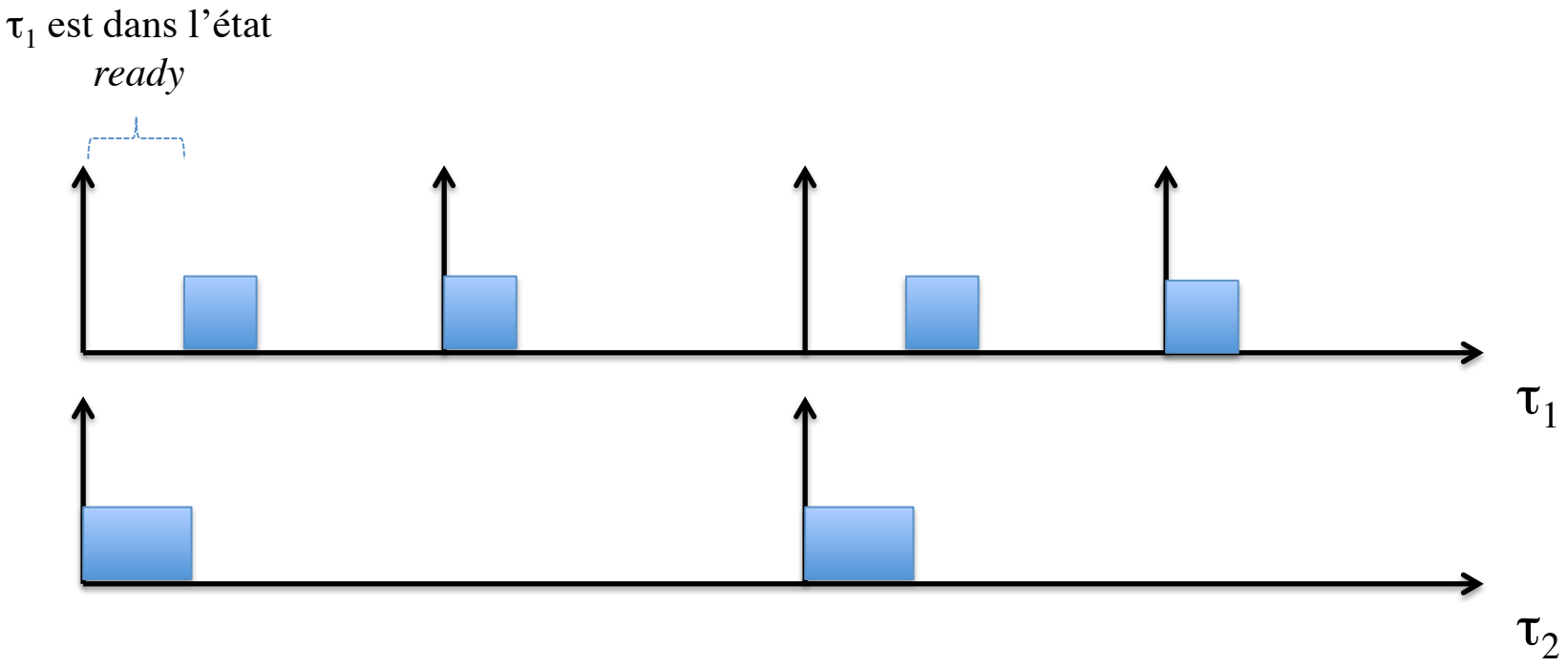
Algorithmes d'ordonnancement

- Dans ce cours, nous nous limitons à deux algorithmes d'ordonnancement, préemptifs, à priorité fixe:
  - FPS (*Fixed Priority Scheduling*): en plus des caractéristiques mentionnées ci-dessus, une tâche τ_i se voit attribué un niveau de priorité P_i (représenté par un entier en général).
  - RMS (*Rate Monotonic Scheduling*): dérivé de FPS, les priorité sont fixés en fonction de la période des tâches. Plus la période est petite, plus la priorité est élevée. Plus formellement, $T_i < T_j \Leftrightarrow P_i > P_j$

- Effet de la pré-emption:
  - lorsqu'une tâche passe de l'état *blocked/suspended* à *ready*, si cette tâche est la plus prioritaire elle passe dans l'état *running*. Si cela se produit alors qu'une autre tâche s'exécutait, cette dernière passe dans l'état *ready* et attend d'être choisie par l'ordonnanceur.
  - Lorsqu'une tâche passe de l'état *running* à *blocked/suspended*, l'ordonnanceur consulte la ready queue, et si elle n'est pas vide il choisit la tâche la plus prioritaire. Cette dernière passe dans l'état running et s'exécute

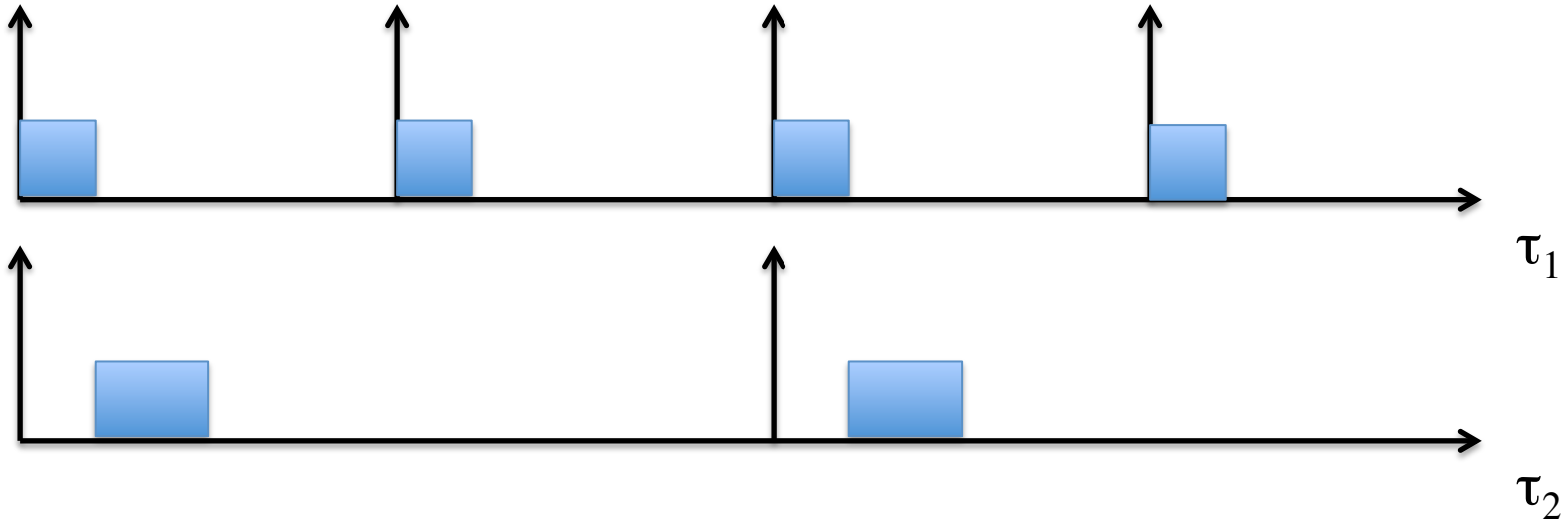
Exemple FPS

Tâche	Période	WCET	Deadline	Priorité
τ_1	10 ms	2 ms	6 ms	5
τ_2	20 ms	3 ms	10 ms	10



Exemple RMS

Tâche	Période	WCET	Deadline
τ_1	10 ms	2 ms	6 ms
τ_2	20 ms	3 ms	10 ms



Formule du calcul de temps de réponse (tâches indépendentes)

Temps de réponse avec FPS

- Le temps de réponse R_i d'une tâche τ_i, se calcule en considérant:
  - C_i, le pire temps d'exécution de τ_i
  - I_i, le temps correspondant aux interférences des tâches de plus forte priorité

$$R_i = C_i + I_i$$

- Il suffit alors de vérifier que $R_i \leq D_i$ pour toutes les tâches: toutes les tâche finissent leur exécution avant leur échéance.

- Toute la difficulté est de calculer I_i bien sûr…

Interférences: nombre de pré-emptions

- Pour prendre en compte les interférences, on doit calculer combien de fois tâche τ_j (plus prioritaire que τ_i) préempte τ_i.
- En supposant que toutes les tâche démarrent en même temps, à l'instant t=0, le nombre maximum de préemption de τ_i par τ_j est:

$$\left\lceil \frac{R_i}{T_j} \right\rceil$$

- Cela revient à compter le nombre de fois que τ_j est activée dans l'intervalle $[0,R_i]$.

Interférences: temps associé

- Pour chaque préemption, le pire temps d'interférence est C_j

- Par conséquent, le temps pendant lequel τ_i subit les interférences de τ_i est:

$$\left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Temps de réponse

- Considérons une tâche τ_i, et hp_i l'ensemble des tâches ayant une priorité supérieure à celle de τ_i

- Nous pouvons alors calculer le pire temps de réponse de τ_i comme suit:

$$R_i = C_i + \sum_{j \in hp_i} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- Calcul par récurrence, initialisé avec $R_i^0 = C_i$, qui s'arrête lorsque
  - $R_i^N > D_i$: le test d'ordonnancement échoue (*i.e.* le système peut ne pas respecter ses contraintes temporelles).
  - $R_i^{N+1} = R_i^N$: le test d'ordonnancement réussi et on a obtenu le pire temps de réponse de τ_i.

Exemple (énoncé)

- En considérant le lot de tâches suivant, ordonnancé avec RMS:

Tâche	Période	WCET	Deadline
τ_1	8 ms	4 ms	6 ms
τ_2	16 ms	3 ms	16 ms
τ_3	4 ms	1 ms	2 ms

- Calculez le pire temps de réponse de τ_3, τ_2, et τ_1
- Le système est-il ordonnançable?

Exemple (corrigé)

- $R_3^0=C_3=1$; τ_3 est la tâche la plus prioritaire, donc pas d'interférences

$$R_3=1$$

- $R_2^0=C_2=3$; τ_2 peut subir des interférences de τ_3 et τ_1
- $R_2^1=ceil(R_2^0/T_3)*C_3+ceil(R_2^0/T_1)*C_1+C_2=ceil(3/4)*1+ceil(3/8)*4+3=8ms$
- $R_2^2=ceil(R_2^1/T_3)*C_3+ceil(R_2^1/T_1)*C_1+C_2=ceil(8/4)*1+ceil(8/8)*4+3=9$ ms
- $R_2^3=ceil(R_2^2/T_3)*C_3+ceil(R_2^2/T_1)*C_1+C_2=ceil(9/4)*1+ceil(9/8)*4+3=14$ ms
- $R_2^4=ceil(R_2^3/T_3)*C_3+ceil(R_2^3/T_1)*C_1+C_2=ceil(14/4)*1+ceil(14/8)*4+3=15$ ms
- $R_2^5=ceil(R_2^4/T_3)*C_3+ceil(R_2^4/T_1)*C_1+C_2=ceil(15/4)*1+ceil(15/8)*4+3=15$ ms

$$R_2=15 \text{ ms}$$

Exemple (corrigé)

- $R_1^0 = C_1 = 4$; τ_1 peut subir des interférences de τ_3
- $R_1^1 = \text{ceil}(R_1^0/T_3)*C_3 + C_1 = \text{ceil}(4/4)*1+4 = 5$ ms
- $R_1^2 = \text{ceil}(R_1^1/T_3)*C_3 + C_1 = \text{ceil}(5/4)*1+4 = 6$ ms
- $R_1^3 = \text{ceil}(R_1^2/T_3)*C_3 + C_1 = \text{ceil}(6/4)*1+4 = 6$ ms

$$R_1 = 6 \text{ ms}$$

- Le lot de tâches est-il ordonnançable?

Formule du calcul de temps de réponse (tâches avec variables partagées)

Problèmes posées par les variables partagées

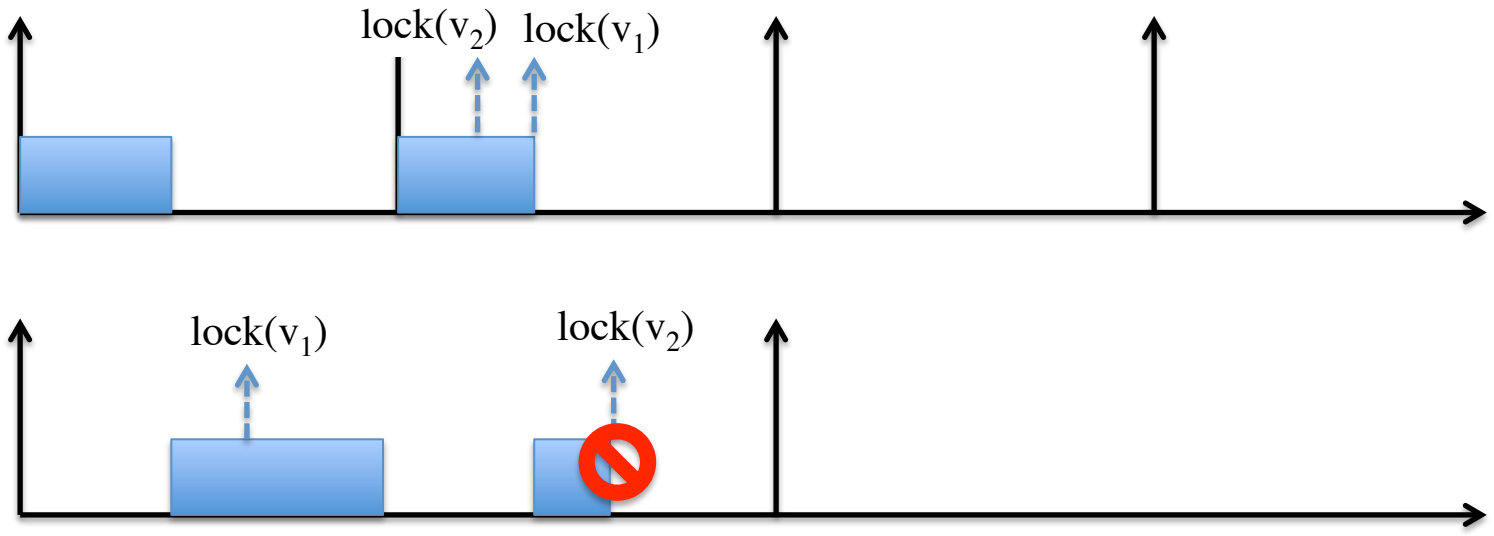
- Pour garantir leur cohérence, il faut (parfois) utiliser des verrous.
  - Risques de deadlock.
  - Risques d'inversion de priorité.

- Rappel: un verrou permet d'accéder à une variable partagée, dans une section critique, en **exclusion mutuelle:**
  - Fonction lock : entrer dans la section critique en exclusion mutuelle.
  - Fonction unlock : sortir de la section critique.

Risque de deadlock (interblocage)

- Un deadlock peut se produire lorsqu'un ensemble de tâche attendent **cycliquement** que la tâche qui les précèdent (dans le cycle) libère une ressource (un verrou dans notre cas).
- Exemple simple: ordo RMS, 2 tâches (τ_1 et τ_2) et deux verrous (v_1 et v_2).

Tâche	Période	WCET	Deadline
τ_1	10 ms	4 ms	10 ms
τ_2	20 ms	7 ms	20 ms

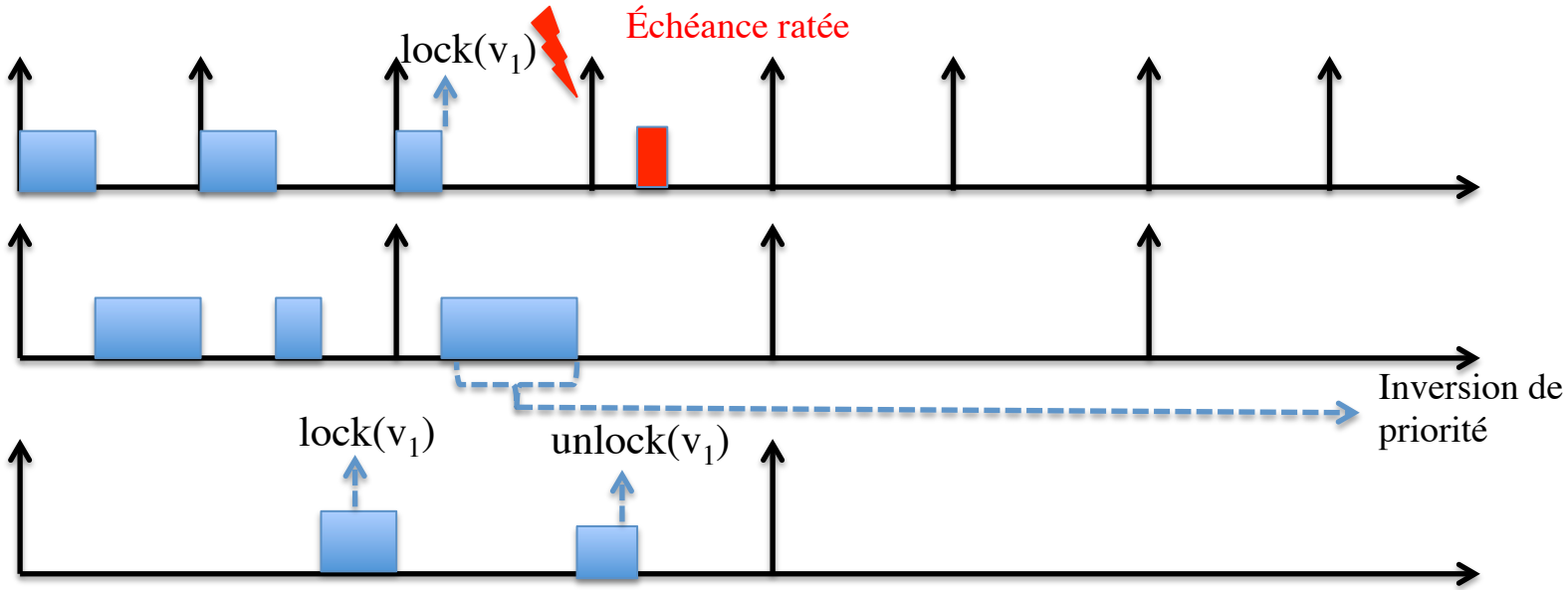


Deadlock!

Risque d'inversion de priorité

- Une inversion de priorité peut se produire lorsqu'une tâche de faible priorité s'exécute avant une tâche de plus forte priorité car celle-ci est bloqué par un verrou.
- Exemple simple: ordo RMS, 3 tâches (τ_1, τ_2, et τ_3) et un verrou (v_1).

Tâche	Période	WCET	Deadline
T_1	5 ms	2 ms	5 ms
τ_2	10 ms	4 ms	10 ms
τ_3	20 ms	7 ms	20 ms



Solution: politique d'accès dédiée

- Différentes politiques ont été proposées pour remédier à ce problème, nous n'en présentons qu'une: *ICPP, Immediate Ceiling Priority Protocol.*
  - Lorsqu'une tâche entre dans la section critique, elle prend **immédiatement une priorité supérieur** à la plus forte priorité de l'ensemble des tâches susceptible d'entrer dans la section critique.

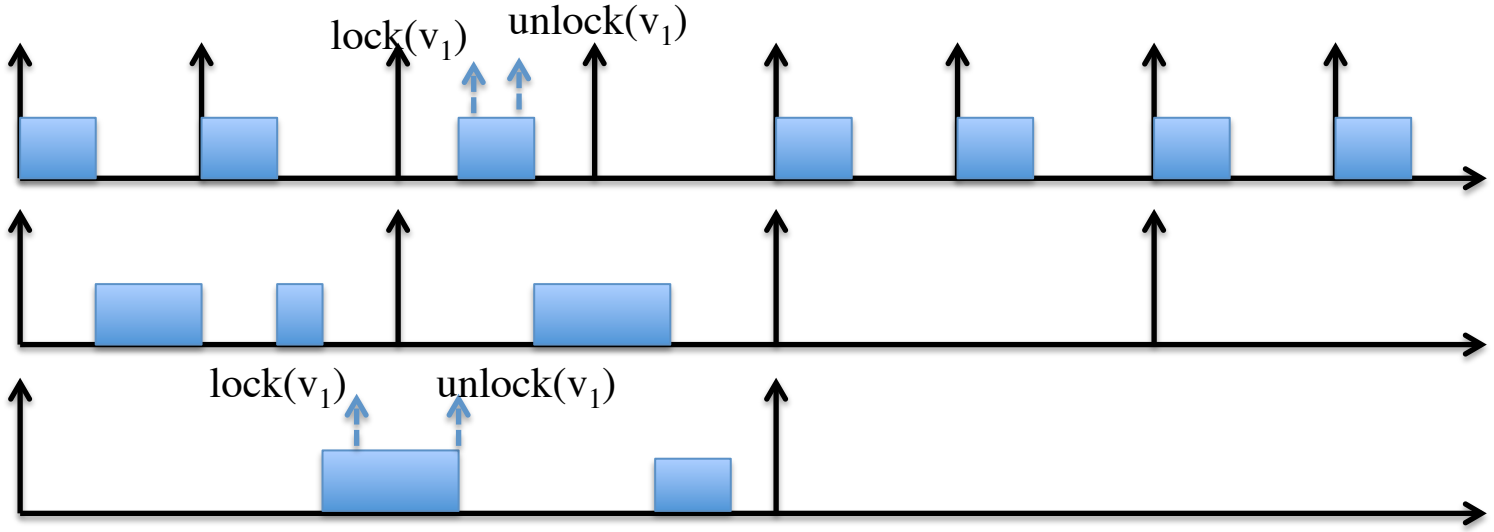
- Illustration (page suivante)

ICPP, illustration

- Exemple simple: ordo RMS, 3 tâches (τ_1, τ_2, et τ_3) et un verrou (v_1).

Tâche	Période	WCET	Deadline	Priorité
τ_1	5 ms	2 ms	5 ms	3
τ_2	10 ms	4 ms	10 ms	2
τ_3	20 ms	7 ms	20 ms	1

- τ_1 et τ_3 utilise v_1. En utilisant la politique ICPP, la priorité de ces tâches dans la section critique et supérieure à 3.



Impact sur le temps de réponse

- L'utilisation de verrous a chamboulé l'ordonnancement: une tâche moins prioritaire peut bloquer une tâche plus prioritaire.
  - Il faut donc mettre à jour la formule du temps de réponse pour une tâche τ_i:

$$R_i = C_i + I_i + B_i$$

- On considère un nombre V de verrous pour l'ensemble des tâches.
  - Pour un verrou v et une tâche τ_i, on définit la fonction *usage(v,i)* qui revoie:
    - 1 si v est utilisé par au moins une tâche de priorité strictement inférieure à τ_i, et par au moins une tâche de priorité supérieure ou égale à τ_i.
    - 0 sinon.
  - Pour un verrou vet une tâche τ_i, le pire temps passé dans la section critique est donné par $C_{v,i}$
- On a alors:

$$B_i = \max_{v=0}^{V} (usage(v,i) \cdot C_{v,i})$$

Pire temps de réponse avec temps de blocage

- Le calcul du temps de réponse revient à résoudre, par récurrence, l'équation suivante:

$$R_i = C_i + \sum_{j \in hp_i} \left\lceil \frac{R_i}{T_j} \right\rceil C_j + \max_{v=0}^{V} (usage(v,i) \cdot C_{v,i})$$

$$C_{v,i} = \max_{\tau_j \in Ts.t.Pj<P_i} (t_{unlock}^{\tau_j} - t_{lock}^{\tau_j})$$

- A propos d'ICPP
  - Ce protocole résout les problèmes d'interblocage et d'inversion de priorité. Par ailleurs, il:
    - minimise le temps de blocage (voir définition plus loin)
    - introduit des perturbations pour les tâches de niveau de priorité intermédiaire

Exercice (énoncé)

- En considérant le lot de tâches suivant, ordonnancé avec RMS, ainsi qu'une ressource R partagée par τ_1 and τ_2 :

Tâche	Période	WCET	Deadline	Resource usage
τ_1	8 ms	4 ms	6 ms	Uses R for 1 ms
τ_2	16 ms	3 ms	16 ms	Uses R for 2 ms
τ_3	4 ms	1 ms	2 ms	Does not use R

- R est protégée en utilisant la politique ICPP
- Calculez le pire temps de réponse de τ_3, τ_2, et τ_1
- Le lot de tâches est-il ordonnançable?

Exercice (corrigé)

- $R_3^0=C_3=1$; τ_3 est la tâche la plus prioritaire, donc pas d'interférences

$$R_3=1$$

- $R_2^0=C_2=3$; τ_2 peut subir des interférences de τ_3 et τ_1; usage(R,2) = 0; donc R2 n'est pas impacté par l'usage de la ressource R.
- $R_2^1=ceil(R_2^0/T_3)*C_3+ceil(R_2^0/T_1)*C_1+C_2=ceil(3/4)*1+ceil(3/8)*4+3=8ms$
- $R_2^2=ceil(R_2^1/T_3)*C_3+ceil(R_2^1/T_1)*C_1+C_2=ceil(8/4)*1+ceil(8/8)*4+3=9$ ms
- $R_2^3=ceil(R_2^2/T_3)*C_3+ceil(R_2^2/T_1)*C_1+C_2=ceil(9/4)*1+ceil(9/8)*4+3=14$ ms
- $R_2^4=ceil(R_2^3/T_3)*C_3+ceil(R_2^3/T_1)*C_1+C_2=ceil(14/4)*1+ceil(14/8)*4+3=15$ ms
- $R_2^5=ceil(R_2^4/T_3)*C_3+ceil(R_2^4/T_1)*C_1+C_2=ceil(15/4)*1+ceil(15/8)*4+3=15$ ms

$$R_2=15 \text{ ms}$$

Exemple (corrigé)

- $R_1^0=C_1=4$; τ_1 peut subir des interférences de τ_3; τ_1 peut être bloquée par τ_2. $B_1(R)=2ms$
- $R_1^1=ceil(R_1^0/T_3)*C_3+C_1+B_1(R)=ceil(4/4)*1+4+2=7$ ms
- $R_1^2=ceil(R_1^1/T_3)*C_3+C_1=ceil(7/4)*1+4+2=8$ ms
- $R_1^3=ceil(R_1^2/T_3)*C_3+C_1=ceil(8/4)*1+4+2=8$ ms

$$R_1 = 8 \text{ ms}$$

- Le lot de tâches considéré n'est pas ordonançable

A propos des analyses d'architectures de systèmes temps-réel

Analyse structurelle d'un modèle

- Nous avons vu dans ce cours que pour appliquer une analyse de temps de réponse, il faut respecter certaines contraintes « structurelle »:
  - Tâches périodiques ou sporadiques seulement
  - Toutes les tâches ont une période et un WCET
  - L'algorithme d'ordonnancement est FPS ou RMS
  - Si l'algorithme d'ordonnancement est RMS, la priorité des tâches doit être conforme à cet algorithme
  - Le seul mécanisme de protection des ressources est le mécanisme de verrou, configuré avec ICPP
- La vérification de ces contraintes peut se faire sur
  - Du code (profile Ada Ravenscar)
  - Des modèles d'architecture

Limitations de l'analyse de temps de réponse

TELECOM ParisTech

Concernant l'analyse présentée dans ce cours

- Suppose le WCET connu, ce qui n'est pas si facile (analyse statique de code, tests exhaustif!)
  - ⌘ Plus d'infos dans SE301
- Très pessimiste, donc mauvaise utilisation des ressources de calcul
  - ⌘ Toutes les tâches s'exécutent selon leur WCET.
  - ⌘ Toutes les sections critiques s'exécutent selon leur WCET.
- Très pessimiste, mais nécessaire à la certification et au dimensionnement de certains systèmes (dits temps-réel critiques).
- Considère une architecture ***mono-cœur***! Se complexifie fortement sur multi-cœur (placement, interconnexion…)
- Les lois de contrôle sont souvent robustes à K dépassements d'échéance parmi N activations successives
  - ⌘ La méthode d'analyse est donc à adapter.

Besoin d'un langage support

- Un langage de description d'architecture pour les systèmes temps réel est nécessaire pour:
  - Vérifier l'applicabilité de l'analyse de temps de réponse
  - Automatiser le calcul (outils Cheddar, MAST)
- D'autres caractéristiques techniques devront être évaluées:
  - Temps de transmission de messages sur un réseaux,
  - Temps de réponse de bout en bout (tâches+messages),
  - Fiabilité du système,
  - Sécurité du système,
  - …
- D'autres modèles de tâches plus sophistiqués existent dans la littérature.
- Nous allons pour cela présenter un grand standard de modélisation pour le logiciel embarqué : AADL