Temps réel, déterminisme et ordonnancement : trois défis majeurs des systèmes embarqués

Joël Goossens

26 mai 2017

ULB

1. Introduction

Quidos

Objectifs de l'exposé

- Qualité des données et **temps**
- Introduction à l'informatique **temps réel** et aux systèmes embarqués
- Système d'exploitation temps réel
- Ordonnancement
- Questions de recherche

Qualité des données

- Au sens large *"Qualité d'un système d'information désigne son adéquation relative aux objectifs"* — Isabelle Boydens, STICB510, ULB.
- Souvent il n'existe pas de référentiel formel et absolu.
- Ici l'objectif principal, la validité/qualité des données/résultats est fonction de leur **"ponctualité"**.

L'INFORMATIQUE de A à Z

Temps

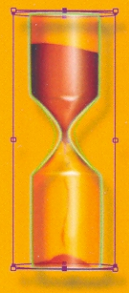
"Patientez s'il vous plaît" : voilà une petite phrase bien agaçante sur l'écran de votre ordinateur. Mais dans certaines situations, qualifiées de "temps réel", une telle attente pourrait conduire à la catastrophe !

Heureusement, quand un pilote d'avion lance la commande de sortie du train d'atterrissage, il sait que son ordinateur de bord va respecter les délais imposés.

Mais ce n'est pas tout ! En plus du temps maximal de réponse, on a besoin de faire respecter d'autres contraintes, comme l'ordre des opérations : l'ordinateur doit s'assurer que l'ouverture du clapet de protection est terminée avant de commencer à déployer les roues.

Pour que les contraintes du temps réel soient satisfaites, les informaticiens mettent au point des systèmes d'exploitation et des langages de programmation spécifiques.

Le respect des contraintes de temps est parfois vital, comme pour les systèmes de sécurité des centrales nucléaires. Dans d'autres cas, les contraintes temporelles ne sont qu'un facteur de qualité, par exemple l'affichage des pixels d'une télévision numérique.



Les systèmes temps réel sont souvent des systèmes embarqués, c'est-à-dire qu'ils sont présents dans des objets dont la vocation première n'est pas l'informatique : voitures, satellites, équipements chirurgicaux...

Pour en **savoir plus** : consultez le site Interstices : **http://interstices.info**

Centre de Vulgarisation de la Connaissance

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

Design : sophie.auvin@wanadoo.fr

Les systèmes temps réel

Définition (en temps réel)

Ce sont des systèmes dont le bon fonctionnement dépend non seulement des résultats des calculs, mais également des **instants** auxquels ces résultats sont produits.

- Typiquement chaque calcul doit être réalisé en respectant une **échéance**.
  - Système temps réel $\neq$ Système rapide ou performant
  - Système temps réel c'est un système qui est **"à l'heure"**
  - Système temps réel est **déterministe** (ou prévisible) et "démontrable"

Échéance stricte/non-stricte

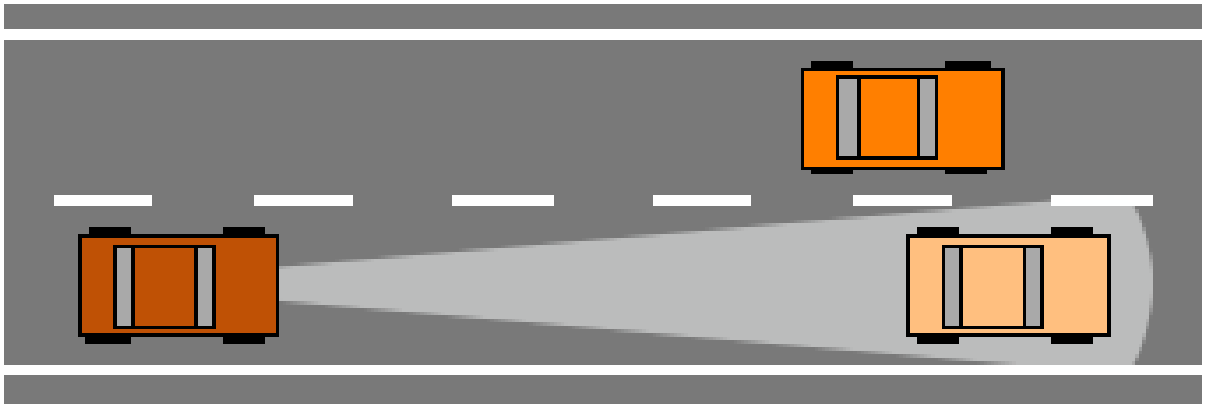
- On distingue le temps réel strict et le temps réel non-strict suivant l'importance accordée aux contraintes temporelles.
- Le temps réel strict ne tolère aucun dépassement de ces contraintes, ce qui est souvent le cas lorsque de tels dépassements peuvent conduire à des situations critiques, voire **catastrophiques** sur un plan écologique, économique ou humain

Exemple 1 : prédiction des phénomènes météorologiques

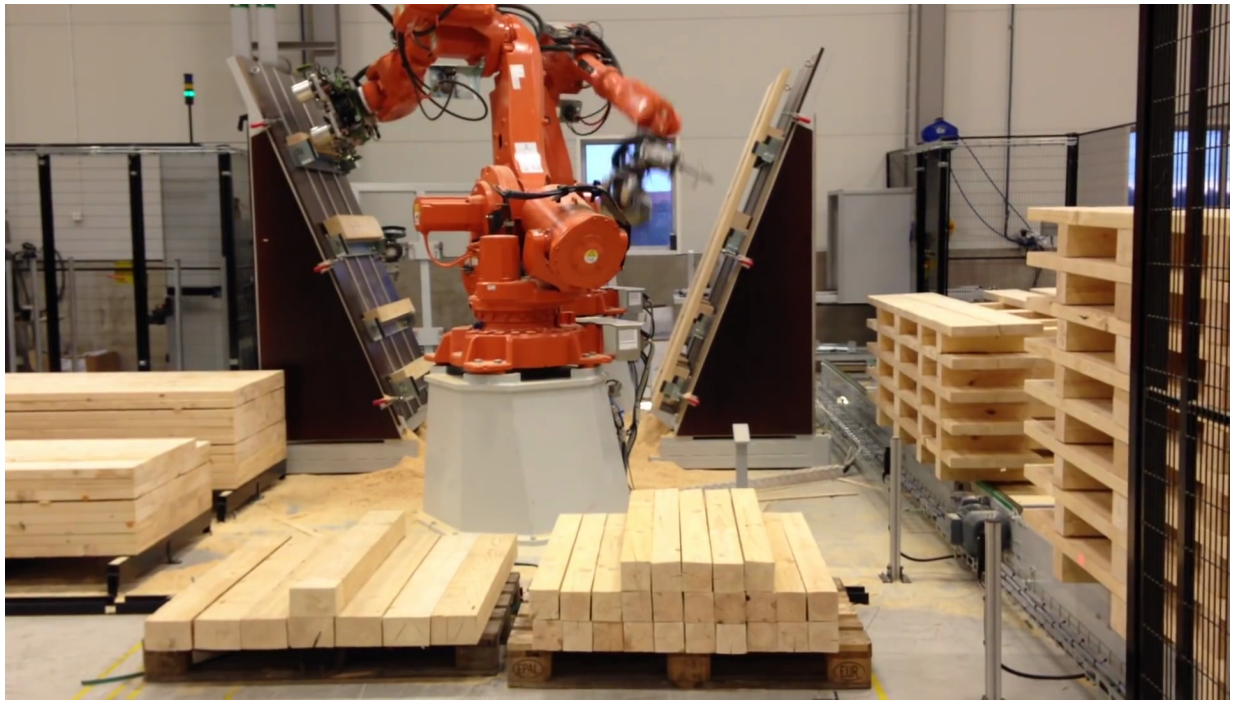
- Temps réel signifie que le résultat du calcul doit arriver à temps.
- L'exemple le plus célèbre est celui de la prédiction des phénomènes météorologiques où il y a quelques années déjà le temps du lendemain matin était prédit avec précision... mais en trois jours de calcul !

Exemple 2 : Radar de régulation de distance

- Ce système régule la vitesse de la voiture afin de garantir un distance de sécurité minimale, afin d'éviter des collisions (*Adaptive Cruise Control*).
- Donc périodiquement il faut adapter la vitesse en fonction de la distance actuelle avec le véhicule de devant.



Exemple 3 : Robotique



Systèmes temps réel, embarqués ou cyber-physiques ?

"...Cyber-Physical Systems (CPS) has emerged as a unifying name for systems where the cyber parts, i.e., the computing and communication parts, and the physical parts are tightly integrated, both at the design time and during operation ..."

"...CPS covers a very wide range of application domains from extremely light-weighted medical devices to nation-scaled power grids ..."

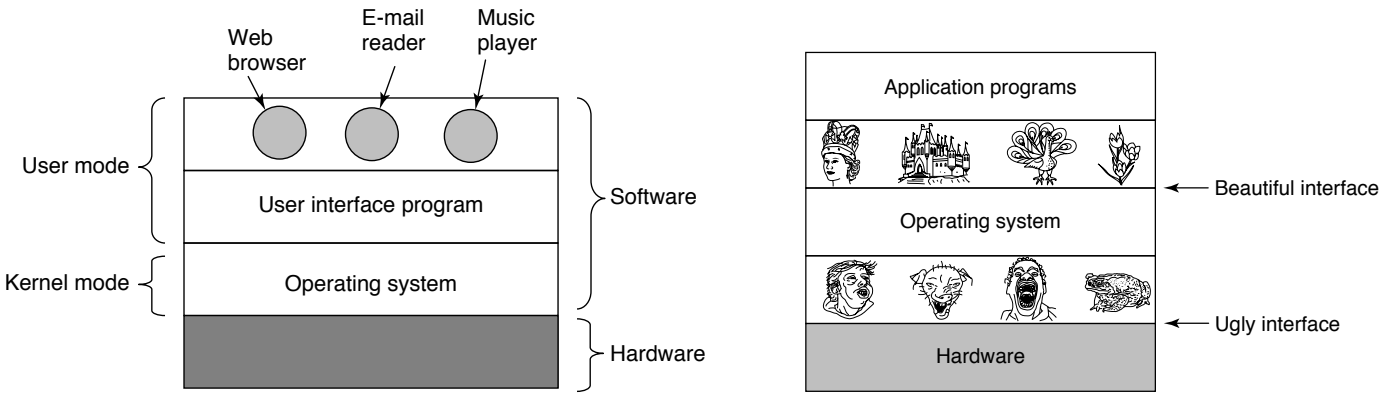
*"...CPS often involves multiple disciplines, such as **real-time systems**, sensor networks, **embedded systems**, control, information processing, and signal processing..."*

— Tei-Wei Kuo
ACM Transactions on Cyber-Physical Systems
Inaugural Issue, February 2017.

2. Système d'exploitation

Système d'exploitation – OS

- Le matériel est de plus en plus complexe, il n'est pas possible d'exiger des programmeurs de maîtriser tous les détails techniques. Les programmeurs ont besoin d'une **abstraction** :



A. Tanenbaum, Modern Operating Systems.

- Les ressources matérielles, souvent limitées, doivent être gérées efficacement. L'OS est aussi un **gestionnaire de ressources**.

GPOS vs. RTOS

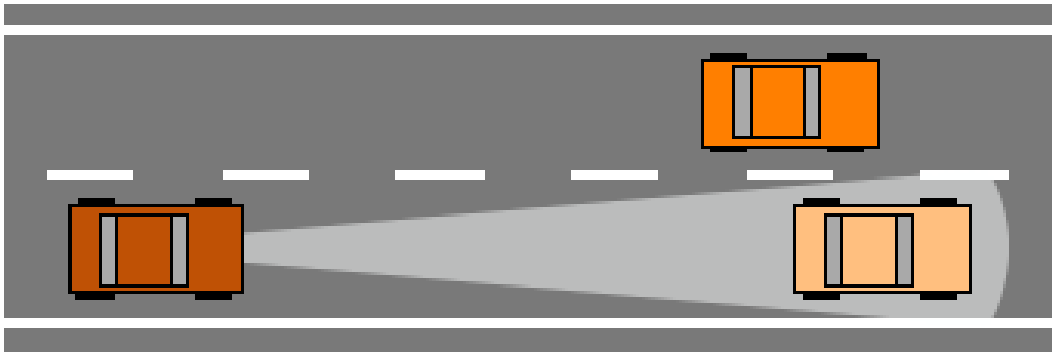
- Les GPOS (*General Purpose OS*) comme MS Windows, Mac OS X, Linux ne satisfont pas les exigences temps réel et de déterministe. Ils ne sont pas adaptés aux situations critiques. Par exemple dans le secteur avionique la probabilité d'une erreur fatale doit être 10^{-9} par heure de vol (fréquence $\approx 100\ 000$ ans), i.e., "extrêmement improbable".
- Les RTOS (*Real-Time OS*) sont conçus afin d'être déterministes et fiables.

GPOS	RTOS
optimize the **average case**	optimize the **worst case**
maximize throughput	minimize latenzy
schedule fairly	schedule **timely**
best effort	real-time & predictable
dynamic programs	specific programs
open environement	closed environment

Algorithmes d'ordonnancement

- Les RTOS gèrent les ressources et les programmes en fonction de leur urgence afin de satisfaire les exigences temps réel et de déterminisme.
- La clef du problème est d'exécuter les programmes dans le bon **ordre**.
- Des algorithmes d'ordonnancement spécifiques sont nécessaires pour "démontrer" que le système sera "toujours" à l'heure.

Adaptive Cruise Control (ACC)



- Afficher sur le tableau de bord la distance du véhicule de devant ;
- Périodiquement nous devons déterminer la vitesse et la distance de la voiture de devant et ajuster notre vitesse ;
- Il faut aussi “sonder” régulièrement la pédale de frein qui doit désactiver l’ensemble du système.

Modélisation sous forme de tâches temps réel

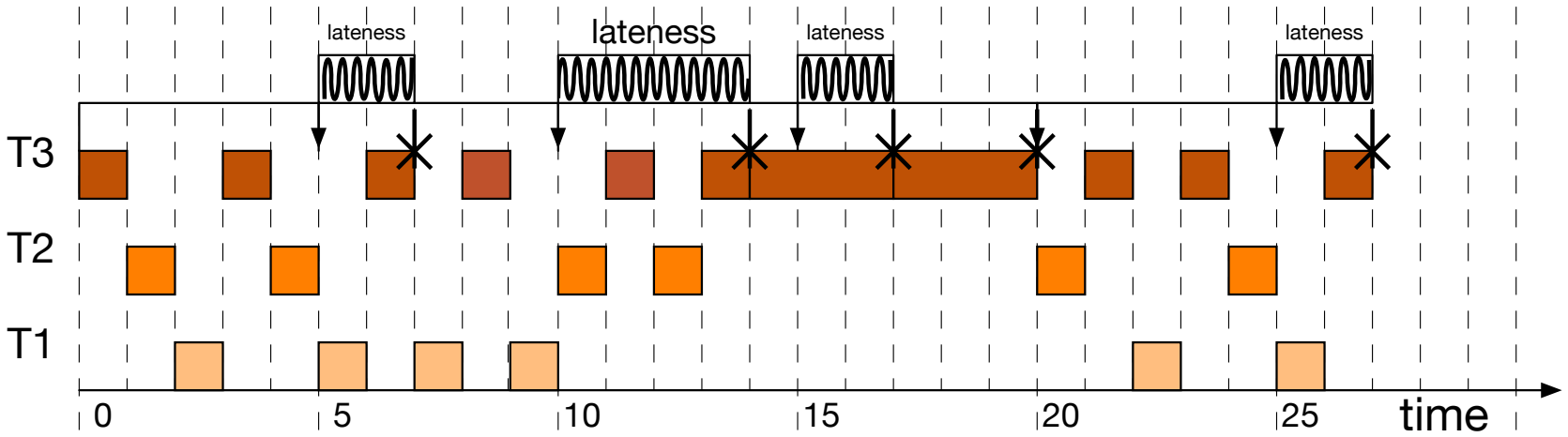
Tâche	Durée (ms)	Période/échéance (ms)	Charge
T1-displaying	4	20	20%
T2-ajust speed	2	10	20%
T3-disabling	3	5	60%
			100%

Le tourniquet : ordonnanceur non temps réel I

- Un ordonnanceur populaire dans la plupart des GPOS est d'implémenter un "tourniquet" (*Round Robin*)
- L'objectif est d'être **équitable**
- On attribue des *"quanta"* équitablement
- Ici, le tourniquet donne une unité (ms) du processeur à chaque tâche toutes les 3 unités (tant qu'il y a 3 tâches en attentes) :

Le tourniquet : ordonnanceur non temps réel II

Tâche	Durée (ms)	Période/échéance (ms)	Charge
T1-displaying	4	20	20%
T2-ajust speed	2	10	20%
T3-disabling	3	5	60%
			100%



Le tourniquet : ordonnanceur non temps réel III

- Le tourniquet ne prend pas en compte les contraintes temps réel.
- Sur notre exemple, le tourniquet produit un ordonnancement catastrophique pour la tâche T3 même si l'on donne les unités rouges (cas le plus favorable).

Rate Monotonic (RM) : ordonnanceur temps réel I

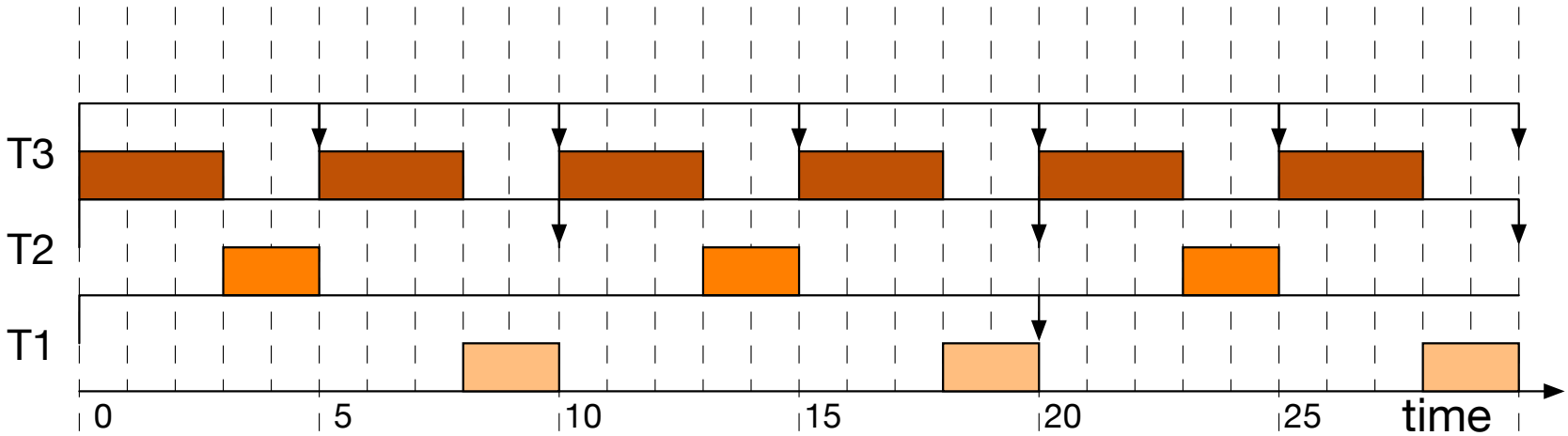
Rate Monotonic (RM) : ordonnanceur temps réel II

- Il existe un ordonnanceur simple et sans échec : *"Rate Monotonic" (NASA/JPL 1969)*, un ordonnanceur à priorité fixe.
- Chaque tâche va recevoir une **priorité** en fonction de la durée de cycle : grand cycle $\Rightarrow$ petite priorité

Rate Monotonic (RM) : ordonnanceur temps réel III

- Dans notre cas les priorités **RM** sont T3 > T2 > T1

Tâche	Durée (ms)	Période/échéance (ms)	Charge
T1-displaying	4	20	20%
T2-ajust speed	2	10	20%
T3-disabling	3	5	60%
			100%

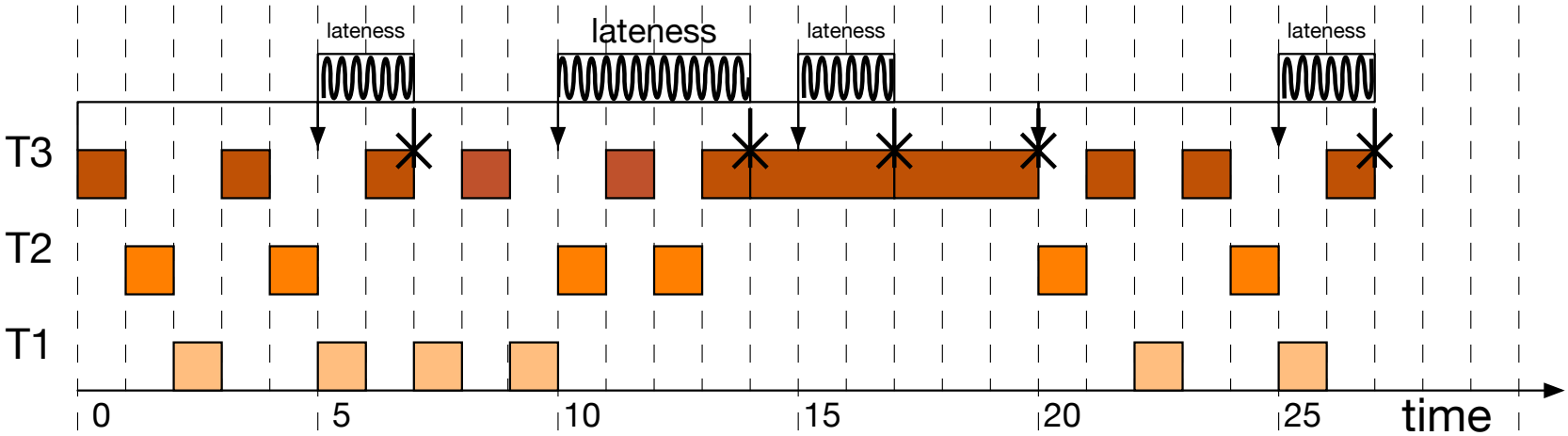


Rate Monotonic (RM) : ordonnanceur temps réel IV

- Pour le type de durée de cycle considéré, **RM** a une **propriété remarquable**

> **RM** est sans échec si le système n'est pas surchargé :
>
> $$\text{Charge} \leq 100\ \%$$

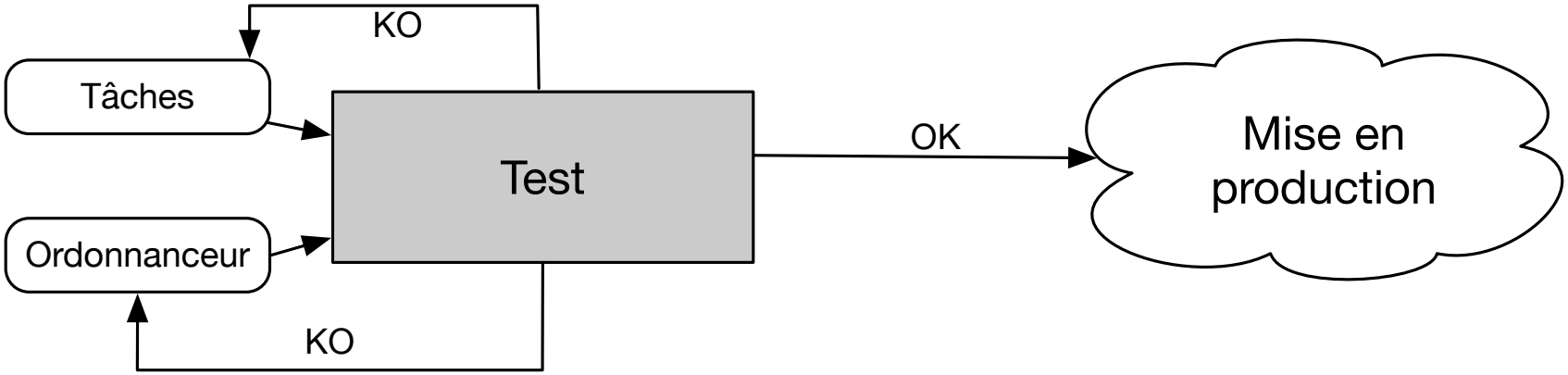
Rapidité vs. priorité I



- Exercice : Round Robin a besoin d'un processeur plus rapide, mais combien de fois plus rapide 2 fois, 3 fois, 4 fois ?
- Le vrai challenge est donc de trouver le bon ordre, i.e., les bonnes priorités.

Validation

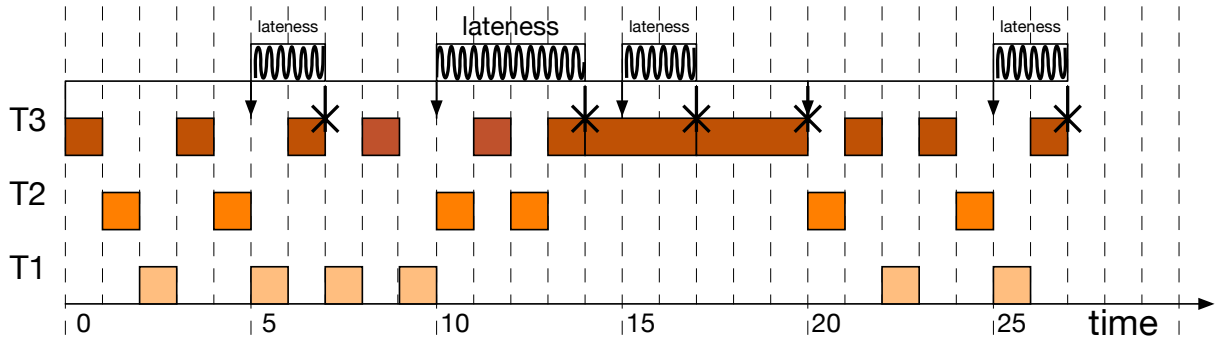
- Utilisation de tests/preuves lors de la conception du système



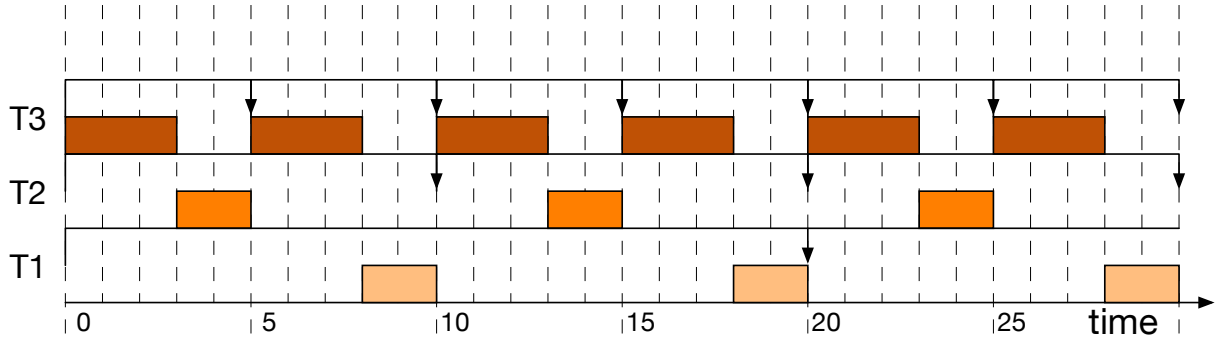
- Mise en production si l'on a "prouvé" que le système sera *"toujours"* à l'heure

Tests d'ordonnançabilité

- Conditions (mathématiques) à vérifier comme la Charge ≤ 100 %
- Simulations, i.e., construire l'ordonnancement
  - soit on rate une échéance (réponse KO)



  - soit l'ordonnancement est sans échec et se **répète** (réponse OK)



3. Questions actuelles

Énergie & température I

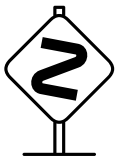
- Implant médical, pacemakers, régulateurs utilisent l'énergie des muscles
- Prototype de rétines artificielles utilisant la lumière incidente

Énergie & température II

- Aérospatial. e.g., Mars Odyssey spacecraft
  moins de batterie et de plus petits panneaux solaires

Consommation des processeurs

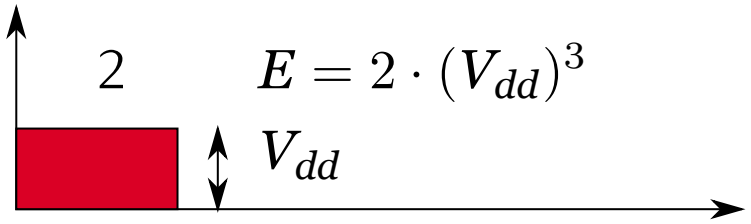
- V_{dd} sera la **tension** électrique du processeur (volt)
- Vitesse est proportionnelle à V_{dd}
  - Puissance consommée est proportionnelle à $(V_{dd})^3$
  - C'est-à-dire proportionnelle au **cube** de la tension

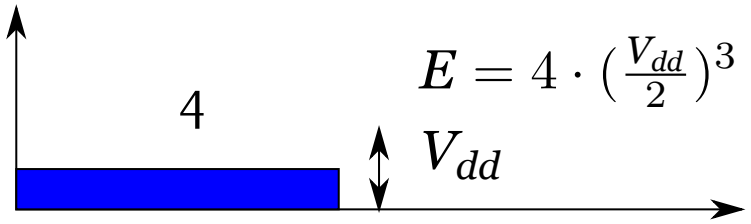


Comment réduire l'énergie I

- Si $P(t)$ est la puissance consommée à l'instant t alors
- l'énergie consommée $E = \int_0^t P(t)\mathrm{d}t$
- Réduire la tension à $\frac{V_{dd}}{2}$ correspond à un gain en énergie d'un facteur 4 :

Comment réduire l'énergie II





Techniques DVFS

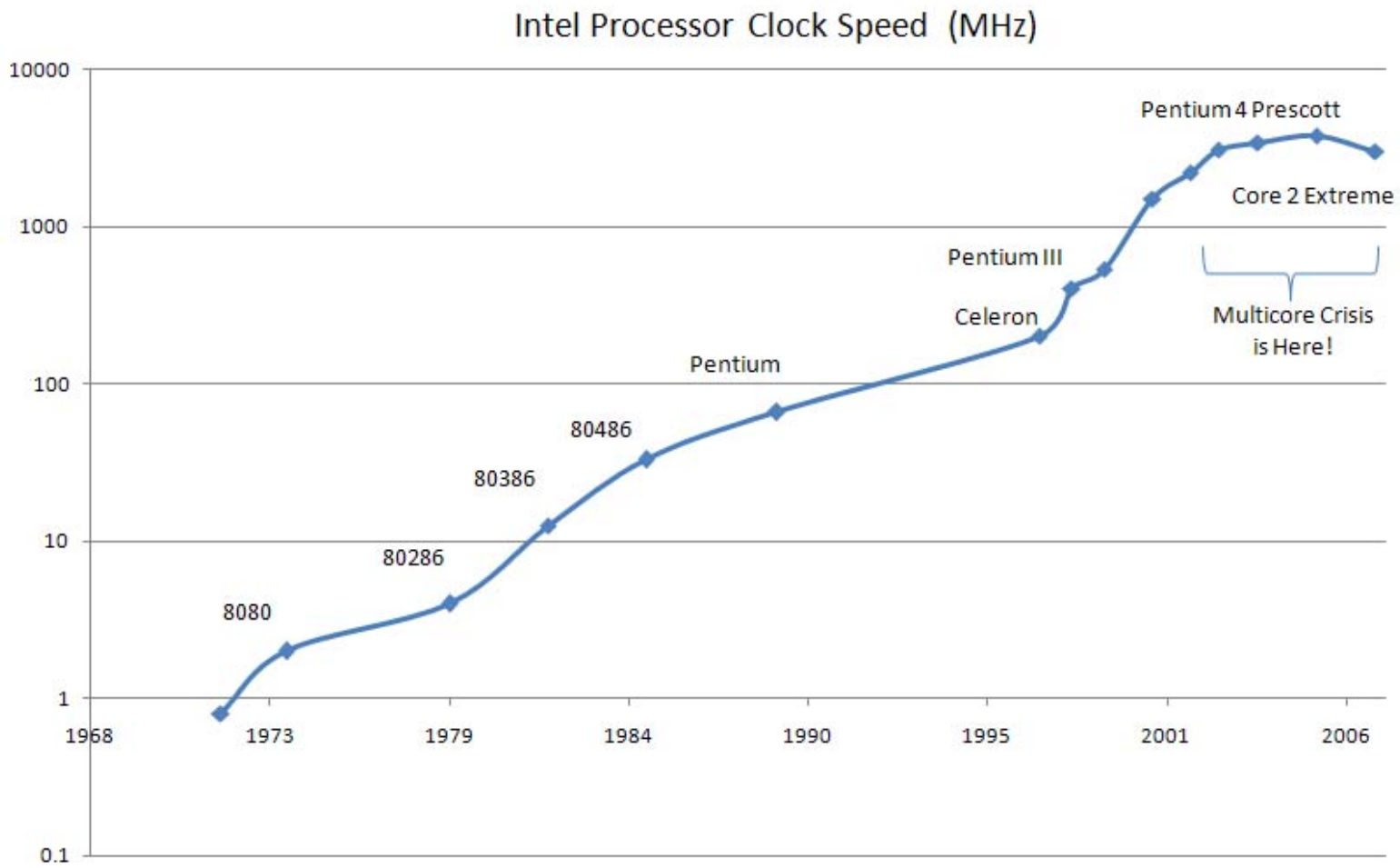
- *"Dynamic Voltage and Frequency Scaling"*, i.e., ajustement dynamique de la tension et de la fréquence des processeurs.
- Il s'agit d'exécuter les tâches plus lentement. Le challenge ici sera de continuer à respecter les contraintes temporelles.

Multicœur I

La Loi de Moore que Wikipedia résume ainsi :

> *"Constatant que la complexité des semiconducteurs proposés en entrée de gamme doublait tous les ans à coût constant depuis 1959, date de leur invention, il postulait la poursuite de cette croissance. Cette* ***augmentation exponentielle*** *fut rapidement nommée Loi de Moore."*

Multicœur II



Multicœur III

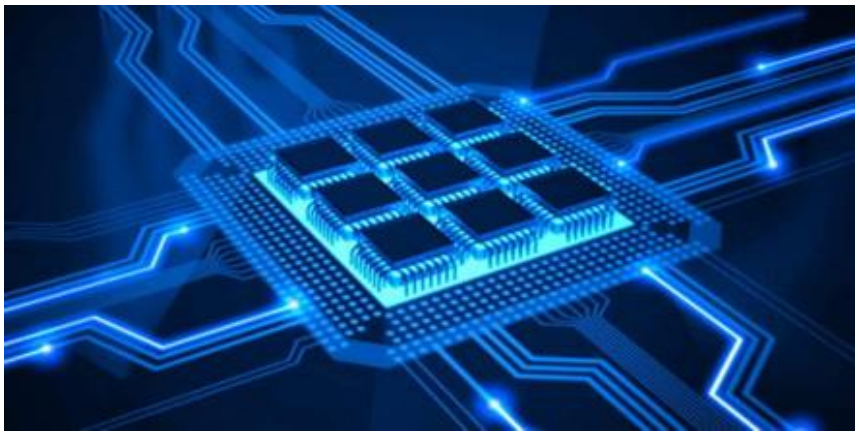
"The chips are down for Moore's law. The semiconductor industry will soon abandon its pursuit of Moore's law. Now things could get a lot more interesting."

— M. Mitchell Waldrop
Nature **530**, 144–147 (11 February 2016).

- En effet nous avons atteint les limites de cette loi :
  - Pour des raisons physiques : nous arrivons à la limite de la matière
  - Pour des raisons économiques : la production de circuits intégrés à haute densité est extrêmement coûteuse

Multicœur IV

- Pour bénéficier d'une grande puissance de calcul les architectures **parallèles multicœurs** sont incontournables.



1 cœur $\rightarrow$ m cœurs : difficultés I

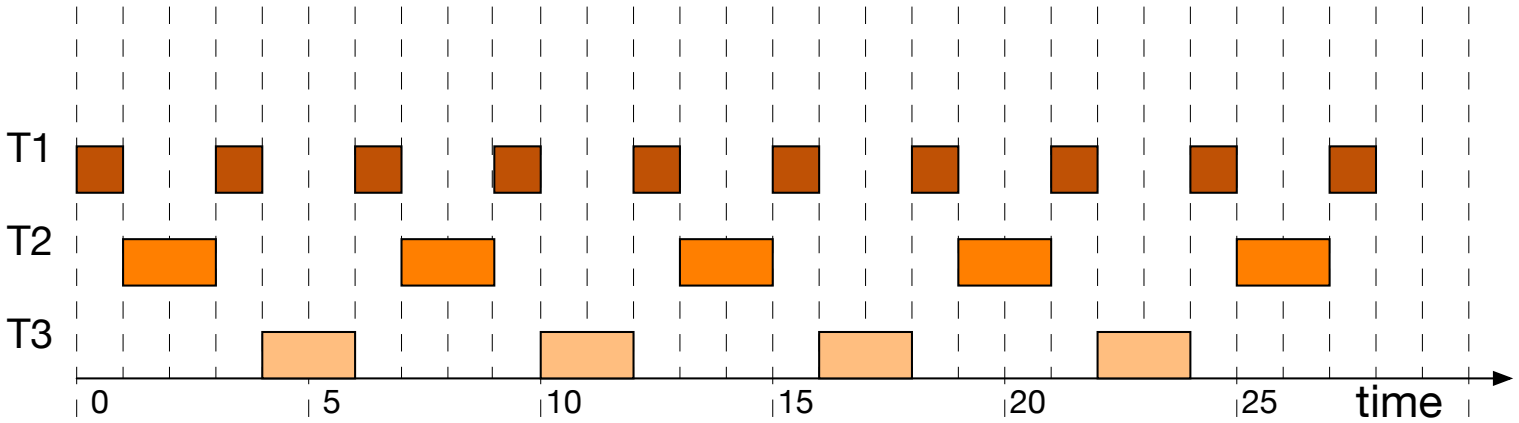
Tâche	Durée (ms)	Période/échéance (ms)	Charge
T1	2	3	66%
T2	4	6	66%
T3	4	6	66%
			200%

- Ici on a une solution simple si l'on peut concevoir un cœur 2 fois plus rapide ce qui revient à diviser par 2 les durées de calcul et ordonnancer ce système :

Tâche	Durée (ms)	Période/échéance (ms)	Charge
T1	1	3	33%
T2	2	6	33%
T3	2	6	33%
			100%

1 cœur $\rightarrow$ m cœurs : difficultés II

- Par la propriété remarquable de **RM** (en mono-cœur) (Charge ≤ 100) nous concluons que le système est ordonnançable :



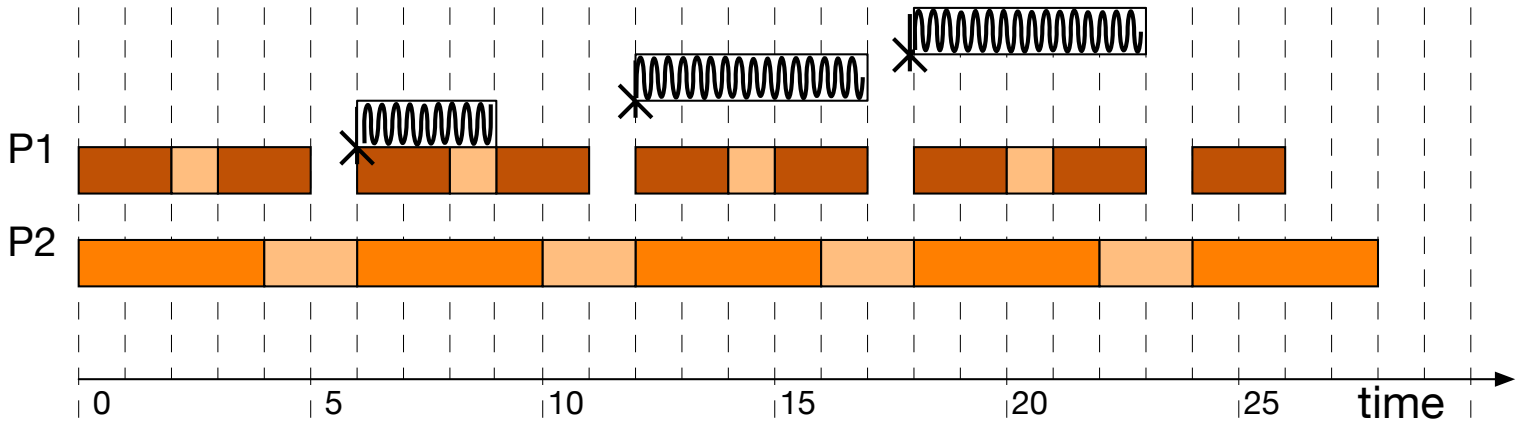
- Si l'on ne peut pas envisager un cœur si rapide nous devons trouver une solution avec au minimum deux cœurs.

Pas de solution sans migration

- Puisque Charge(T1) + Charge(T2) > 1 et Charge(T1) + Charge(T3) > 1 et Charge(T2) + Charge(T3) > 1
- il n'y a pas de solution où les tâches "restent" toujours sur le même cœur.
- nous avons besoin de faire **migrer** au moins une tâche.

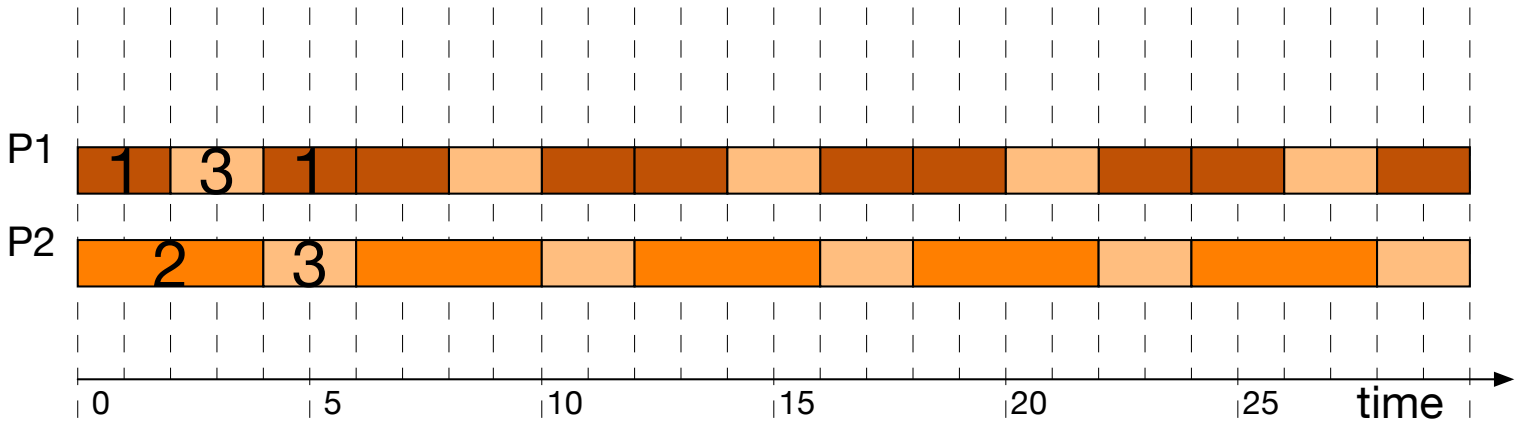
Pas de solution avec des priorités fixes

- Illustrons cela avec **RM** (T1>T2>T3)



- T3 n'est pas à l'heure.
- Pendant les "quanta" 5,11,17,23 seul un cœur travaille !

Une solution avec des priorités dynamiques et migrations



- T3 migre régulièrement !
- La priorité de T3 doit régulièrement être plus grande que celle de T1.

1 cœur $\rightarrow$ m cœurs : nouveaux problèmes

- Ce n'est pas une simple généralisation de 40+ ans de recherche.
- Il s'agit de nouveaux problèmes fondamentalement différents.

Challenges

- 1 cœur $\rightarrow$ m cœurs, systèmes contraintes par le temps, l'énergie ou la t°.
- Du point de vue de la recherche fondamentale, appliquée, des solutions et des produits il s'agit en réalité d'une révolution.
- Il y a de nombreux défis à relever pour les RTOS de demain.

4. Conclusions

Conclusions

- Qualité des données et **temps**
- Informatique **temps réel** et les systèmes embarqués
- Importance de la couche logicielle RTOS
- Ordonnancement
- Questions de recherche du domaine

Questions

Joël Goossens

Joël Goossens est professeur a l'Université libre de Bruxelles (ULB) depuis Octobre 2006.

Il a fondé et co-dirige l'unité de recherche "PARTS" (Parallel Architectures for Real-Time Systems, Faculté des Sciences/École Polytechnique de Bruxelles).

Ses recherches portent sur la théorie de l'ordonnancement pour les systemes embarqués et temps réel et la conception des systèmes d'exploitation temps réel.

Joël **Goossens** enseigne dans deux facultés : des matieres pointues de sa discipline de recherche en Faculté des Sciences et démystifie l'outil informatique en Faculté de Lettres (LTC).

Joël **Goossens** est également co-fondateur de la spinoff ULB "HIPPEROS" (High Performance Parallel Embedded Real-time Operating Systems). Il y occupe la fonction de "Chief Scientific Evangelist", il est responsable des projets de R&D et de la diffusion de l'innovation.

Liens & références

- Joël Goossens on linkedin
- Research Center PARTS (ULB)
- “Real-Time Systems”, J.W.S. Liu, Prentice Hall, 2000.
- “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”, C.L. Liu and J.W. Layland, Journal of the ACM, Volume 20 Issue 1, Jan. 1973.
- “Multiprocessor scheduling for real-time systems”, Sanjoy Baruah, Marko Bertogna, and Giorgio Buttazzo. Springer, 2014.