
Ordonnancement des applications temps réel réparties

Frank Singhoff

Bureau C-203

Université de Brest, France

LISyC/EA 3883

singhoff@univ-brest.fr

Sommaire

1. Introduction et rappels.
2. Ordonnancement global.
3. Ordonnancement par partitionnement.
4. Résumé.
5. Acronymes.
6. Références.
7. Remerciements.

Ordonnancement, définitions (1)

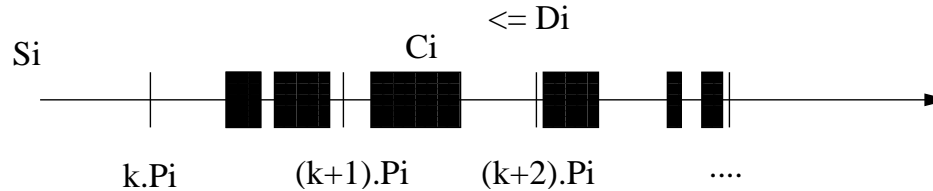
- **Objectifs** : prendre en compte les besoins d'urgence, d'importance et de réactivité dans l'exécution des tâches d'une application temps réel.
- **Taxinomie** : préemptif/non préemptif, priorités statiques/dynamiques, hors/en ligne.
- **Principales propriétés recherchées** :
 1. **Facilité de mise en œuvre** : l'ordonnanceur est-il facile à implanter ?
 2. **Faisabilité** : est il possible d'exhiber un test de faisabilité ? Condition permettant de décider hors ligne du respect des contraintes des tâches.
 3. **Efficacité** : critères de comparaison des algorithmes \implies optimalité, dominance, équivalence et comparabilité.

Ordonnancement, définitions (2)

- **Critères de comparaison des algorithmes:**

- **Optimalité** : un algorithme a est optimal s'il est capable de trouver un ordonnancement pour tout ensemble faisable de tâches.
- **Dominance** : a domine b si tous les jeux de tâche faisables par b le sont aussi par a et s'il existe des jeux de tâches faisable par a qui ne le sont pas par b .
- **Equivalence** : a et b sont équivalents si tous les jeux de tâche faisable par a le sont aussi par b , et inversement.
- **Incomparable** : a et b sont incomparables s'il existe des jeux de tâches faisables par a qui soient infaisables par b et inversement.

Ordonnancement, définitions (3)



- Paramètres définissant une tâche périodique i , tâche/fonction critique :
 - Arrivée de la tâche dans le système : S_i .
 - Borne sur le temps d'exécution d'une activation : C_i (capacité).
 - Période d'activation : P_i . Notion de travail.
 - Délai critique : D_i (relatif à P_i).
- Modèle tâches périodiques synchrones à échéances sur requêtes [LIU 73] :
 - Tâches périodiques et indépendantes.
 - avec $\forall i : S_i = 0$ (instant critique, pire cas).
 - $\forall i : P_i = D_i$.

Exemple : algo. à priorité fixe (1)

- Priorités fixes \implies analyse hors ligne \implies applications statiques et critiques.
- **Propriétés/hypothèses :**
 - Tâches périodiques synchrones à échéances sur requêtes.
 - Complexité faible et mise en œuvre aisée.
 - Affectation Rate Monotonic : algorithme optimal dans la classe des algorithmes à priorité fixe. Dominé par EDF.
- **Fonctionnement :**
 1. Affectation des priorités selon l'urgence ou l'importance. Ex : période (Rate Monotonic), délai critique (Deadline Monotonic).
 2. Phase d'élection : election de la plus forte priorité.

Exemple : algo. à priorité fixe (2)

- Tests de faisabilité (cas préemptif) :
 1. **Période d'étude** = $[0, PPCM(P_i)]$. Condition suffisante et nécessaire.
 2. **Test sur le taux d'utilisation** (Rate Monotonic seulement) :

$$U = \sum_{i=1}^n \frac{C_i}{P_i} \leq n(2^{\frac{1}{n}} - 1) \leq 0.69$$

Condition suffisante mais non nécessaire.

3. **Test sur le temps de réponse** (qq soit l'affectation) :

$$r_i = C_i + \sum_{j \in hp(i)} I_j = C_i + \sum_{j \in hp(i)} \left\lceil \frac{r_i}{P_j} \right\rceil C_j \leq D_i$$

Condition généralement suffisante et nécessaire. $hp(i)$ est l'ensemble des tâches de plus forte priorité que i . Instant critique requis.

Exemple : algo. à priorité fixe (3)

- Technique de calcul : on évalue de façon itérative w_i^n par :

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{P_j} \right\rceil C_j$$

- On démarre avec $w_i^0 = C_i$.
- Conditions d'arrêt :
 - Echech si $w_i^n > P_i$.
 - Réussite si $w_i^{n+1} = w_i^n$.

Exemple : algo. à priorité fixe (4)

- Exemple (3 tâches) : $P1=7$; $C1=3$; $P2=12$; $C2=2$; $P3=20$; $C3=5$

- $w_1^0 = 3 \implies r_1 = 3$

- $w_2^0 = 2$

- $w_2^1 = 2 + \left\lceil \frac{2}{7} \right\rceil 3 = 5$

- $w_2^2 = 2 + \left\lceil \frac{5}{7} \right\rceil 3 = 5 \implies r_2 = 5$

- $w_3^0 = 5$

- $w_3^1 = 5 + \left\lceil \frac{5}{7} \right\rceil 3 + \left\lceil \frac{5}{12} \right\rceil 2 = 10$

- $w_3^2 = 5 + \left\lceil \frac{10}{7} \right\rceil 3 + \left\lceil \frac{10}{12} \right\rceil 2 = 13$

- $w_3^3 = 5 + \left\lceil \frac{13}{7} \right\rceil 3 + \left\lceil \frac{13}{12} \right\rceil 2 = 15$

- $w_3^4 = 5 + \left\lceil \frac{15}{7} \right\rceil 3 + \left\lceil \frac{15}{12} \right\rceil 2 = 18$

- $w_3^5 = 5 + \left\lceil \frac{18}{7} \right\rceil 3 + \left\lceil \frac{18}{12} \right\rceil 2 = 18 \implies r_3 = 18$

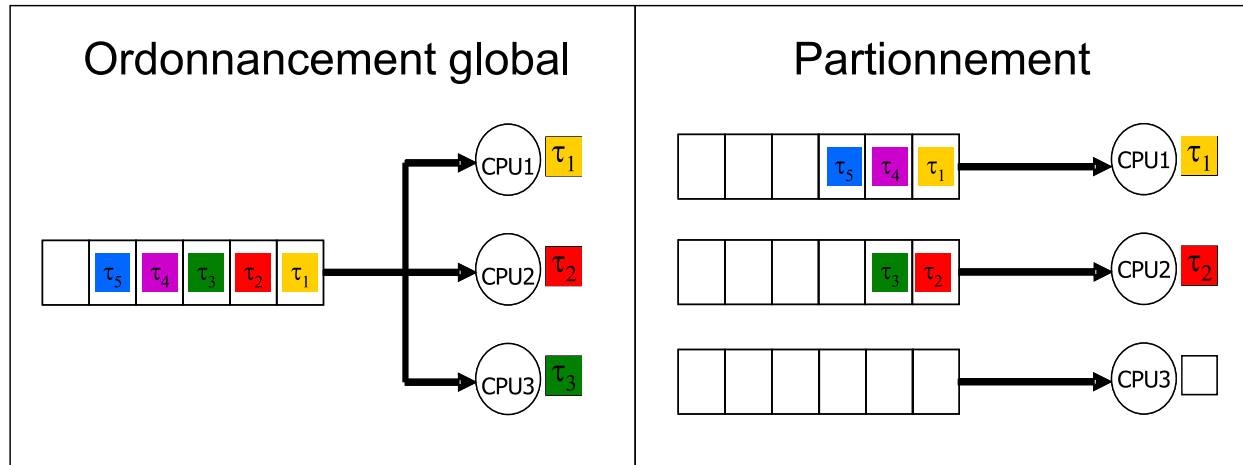
Systemes répartis ciblés (1)

- *"Un système réparti est un ensemble de machines autonomes connectées par un réseau, et équipées d'un logiciel dédié à la coordination des activités du système ainsi qu'au partage de ses ressources." Coulouris et al. [COU 94].*
- **Pourquoi un système réparti ?**
 1. Tolérance aux pannes (fiabilité, disponibilité).
 2. Contraintes physiques (ex : avionique, usines automatisées).
 3. Contraintes liées au processus industriel.
 4. Partage des ressources (données, applications, périphériques chers). Optimisation de leur utilisation.
 5. ...

Systèmes répartis ciblés (2)

- **Catégories de systèmes :**
 - Avec processeurs/systèmes d'exploitation homogènes ... ou non.
 - Fortement couplé : horloge commune et mémoire partagée.
 - Ou faiblement couplé : pas d'horloge commune, mais un réseau de communication. Pas de mémoire partagée => flot de données.
- **Deux approches :** approches par ordonnancement global ou par partitionnement et ordonnancement local.

Systèmes répartis ciblés (3)



- Ordonnancement dans un système réparti/multi-processeurs :
 1. **Ordonnancement global** : choisir d'abord la tâche, puis placer la tâche sur un des processeurs libres. Approche en-ligne.
 2. **Ordonnancement par partitionnement** : placement des tâches pour ordonnancement local, puis vérification éventuelle des délais de bout en bout. Approche hors-ligne.

Systèmes répartis ciblés (4)

1. Ordonnancement global :

- Inadéquation aux systèmes actuels : migration de tâches/travaux ? Systèmes hétérogènes et processus industriels/systèmes de systèmes.
- Théorie ordonnancement temps réel global peu mature (résultats différents ordonnancement mono-processeur).
- Meilleure optimisation des ressources : processeur occupé, moins de préemption ... mais des migrations.
- Plutôt système temps réel ouvert. Système homogènes et fortement couplés \Rightarrow architecture multi-coeurs.

2. Ordonnancement par partitionnement :

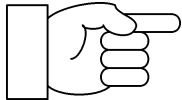
- Meilleure adéquation aux systèmes actuels.
- Théorie ordonnancement temps réel mono-processeur mature.
- Utilisation moins optimale des ressources (processeur libre).
- Retards/pannes/anomalies d'une tâche limités à un processeur.
- Vérification de contrainte de bout en bout délicate.
- Problème de Bin-packing : NP-dur donc.
- Plutôt système temps réel fermé. Adapté systèmes hétérogènes, faiblement couplés.

Systemes répartis ciblés (5)

Ordonnancement global

ou

Ordonnancement par partitionnement :



Les deux approches sont incomparables !

Sommaire

1. Introduction et rappels.
2. Ordonnancement global.
3. Ordonnancement par partitionnement.
4. Résumé.
5. Acronymes.
6. Références.
7. Remerciements.

Ordonnancement global (1)

- **Types de plate-forme :**

- **Processeurs identiques** : processeur ayant la même capacité de calcul.
- **Processeurs uniformes** : cadence d'exécution associée à chaque processeur tel que exécuter un travail de durée t sur un processeur de cadence c requiert $t \cdot c$ unités de temps.
- **Processeurs spécialisés** : cadence d'exécution définie par le triplet $(r_{i,j}, i, j)$: le travail i requiert $r_{i,j} \cdot t$ unités de temps pour une durée de travail t sur le processeur j .
- Identique \subset Uniforme \subset Spécialisé.
- Spécialisés et uniformes = processeurs hétérogènes.
- Identiques = processeurs homogènes.

Ordonnancement global (2)

- **Un ordonnanceur global traite deux problèmes [DAV 09, BER 07] :**
 - Quand et comment affecter les priorités des tâches/travaux.
 - Choisir le processeur où exécuter la tâche.

Ordonnancement global (3)

- **Instants de migration :**

- Migration impossible : la tâche ne peut pas migrer. Tous ses travaux sont affectés à un processeur unique => partitionnement.
- Migration possible entre les différents travaux d'une tâche. Un travail ne peut pas migrer.
- Migration d'un travail possible d'une unité de temps à une autre (parallélisation d'un travail généralement interdit).

- **Affectations de priorité :**

- Priorité fixe associée à la tâche (ex : RM).
- Priorité fixe associée au travail (ex : EDF).
- Priorité dynamique chaque unité de temps du travail/tâche (ex : LLF).

Ordonnancement global (4)

- Deux types d'algorithmes

1. Adaptations d'algorithmes centralisés :

- global RM, global EDF, global DM, global LLF, ...
- Choix du niveau de migration
- Appliquer globalement sur l'ensemble des processeurs une stratégie d'ordonnancement. Attribuer à chaque instant les m processeurs aux m tâches/travaux les plus prioritaires.
- Prémption d'une tâche/travail lorsque tous les processeurs sont occupés.

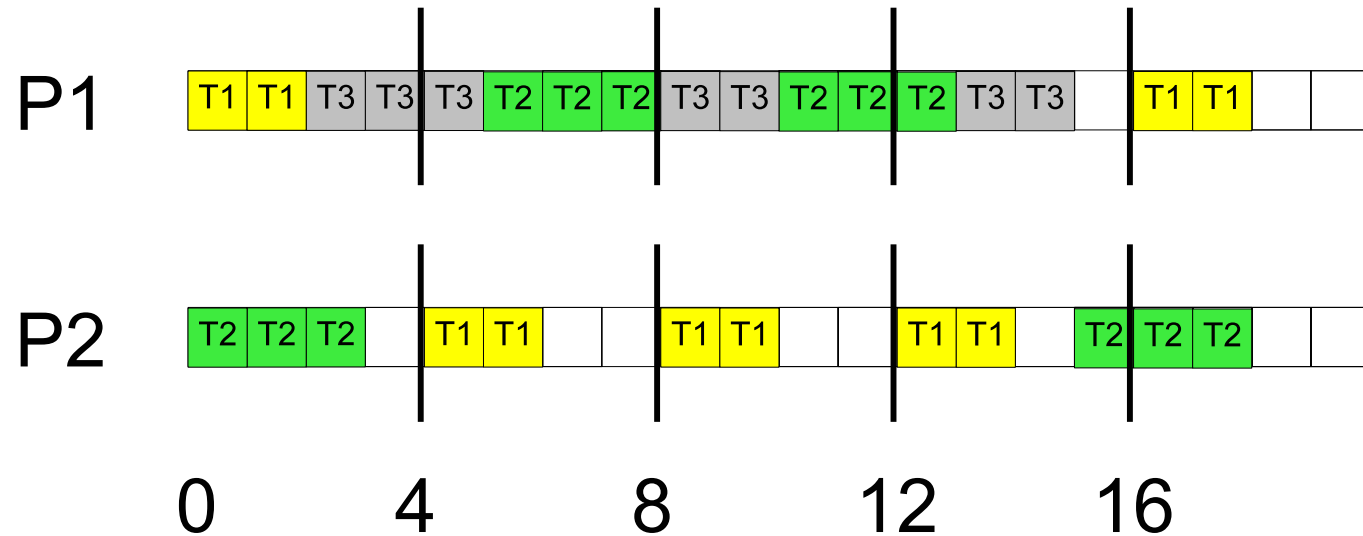
2. Algorithmes nouveaux : PFair, LLREF, EDF(k), SA, EDZL, ...

- Attention : domaine différent du mono-processeur => résultats/propriétés différents, résultats généraux peu nombreux.
- Par la suite, on suppose des processeurs identiques.

Ordonnancement global (5)

- **Exemple** : global Deadline Monotonic

	Ci	Pi	Di
T1	2	4	4
T2	3	5	5
T3	7	20	20

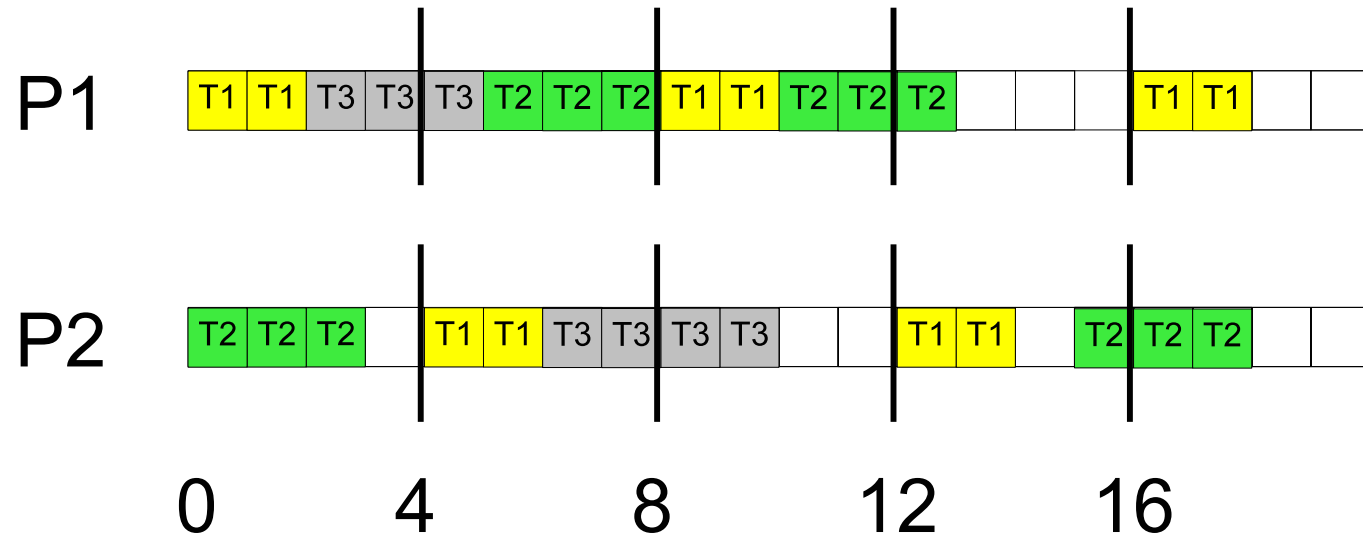


- Affectation des priorités : $T_1 > T_2 > T_3$.
- Migrations possibles des travaux.

Ordonnancement global (6)

- **Exemple** : global Deadline Monotonic

	Ci	Pi	Di
T1	2	4	4
T2	3	5	5
T3	7	20	20



- Affectation des priorités : $T_1 > T_2 > T_3$.
- Migrations possibles chaque unité de temps.

Ordonnancement global (7)

3: migration chaque unité de temps	(1,3)	(2,3)	(3,3)
2: migration travaux	(1,2)	(2,2)	(3,2)
1: migration interdite	(1,1)	(2,1)	(3,1)
	1: priorités fixes tâches	2: priorité fixes travaux	3: priorités dynamiques chaque unité de temps

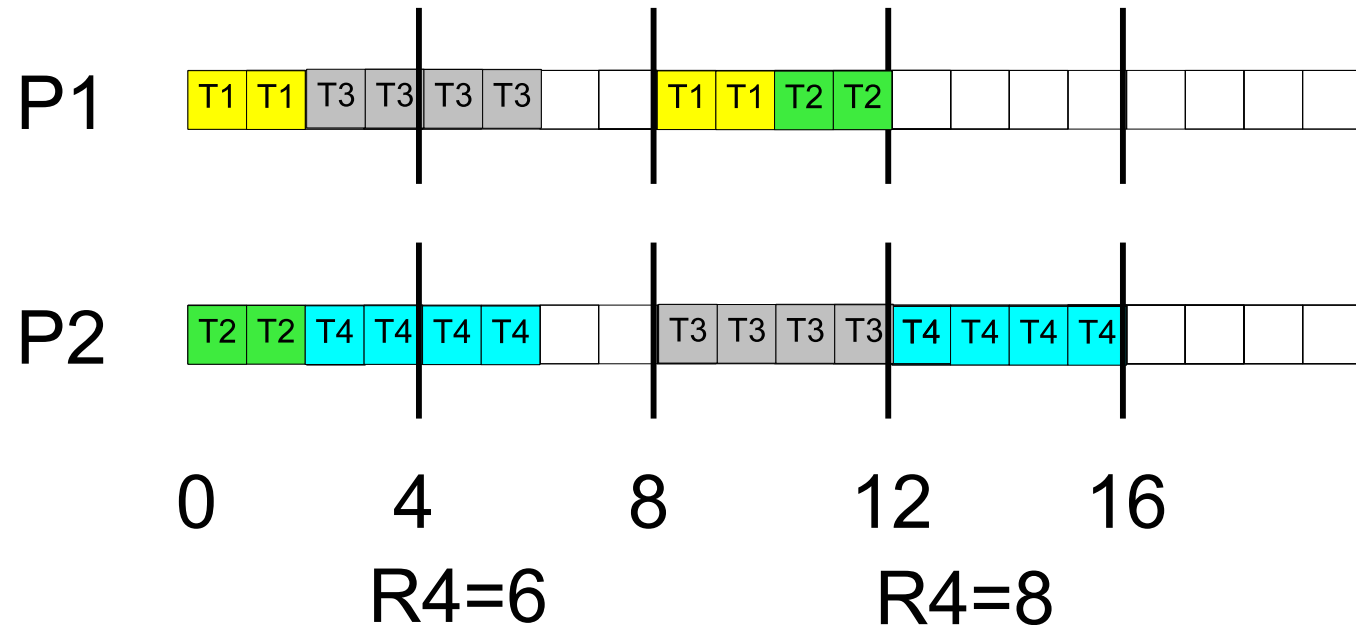
- **Comparaison des algorithmes :**

- Dominance des algorithmes à priorités dynamiques. (3,3) est la classe dominante d'algorithme. Hiérarchie différente de celle en mono-processeur : global LLF domine global EDF.
- (1,*) sont incomparables entre eux.
- (*,1) sont incomparables avec (*,2) et (*,3).
- Algorithmes Pfair : avec processeurs identiques + tâches périodiques synchrones à échéances sur requêtes = algorithme optimal.

Ordonnancement global (8)

- Instant critique :

	Ci	Di	Pi
T1	2	2	8
T2	2	4	10
T3	4	6	8
T4	4	7	8



- En mono-processeur, l'instant critique est le pire scénario pour les tâches périodiques. Hypothèse pour le calcul du pire temps de réponse.
- N'est plus vrai en multi-processeur.

Ordonnancement global (9)

- **Période d'étude :**

- En mono-processeur, la période d'étude permet de vérifier :
 1. Un jeu de tâches périodiques asynchrones/synchrones, $\forall D_i$.
 2. Avec un algorithme d'ordonnancement déterministe, sans test de faisabilité !

$$[0, LCM(\forall i : P_i) + 2 \cdot \max(\forall i : S_i)]$$

- En multi-processeurs identiques, un seul résultat connu :

$$[0, lcm(\forall i : P_i)]$$

pour des jeux de tâches synchrones à échéances sur requêtes et algorithme à priorités fixes seulement :-(

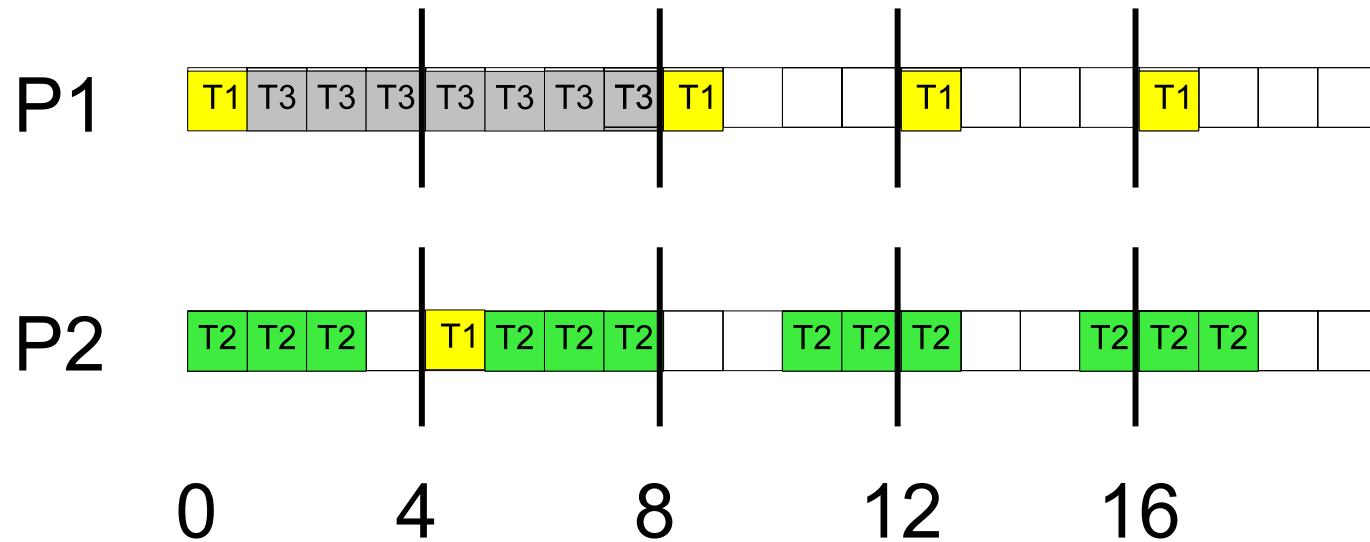
Ordonnancement global (10)

- **Anomalies d'ordonnancement :**
 - Anomalie : changement intuitivement positif de l'architecture qui conduit à un jeu de tâches non faisable.
 - Méthode de vérification utilisée en mono-processeur : vérification pire cas (ex : tâches sporadiques validées comme des tâches périodiques).
 - Paramètres concernés : $C_i, P_i \implies$ baisse du taux d'utilisation.

Ordonnancement global (11)

- Anomalies d'ordonnancement :

	Ci	Pi	Di
T1	1	4	2
T2	3	5	3
T3	7	20	8



- Migration possible chaque unité de temps.
- Jeu de tâches faisable.

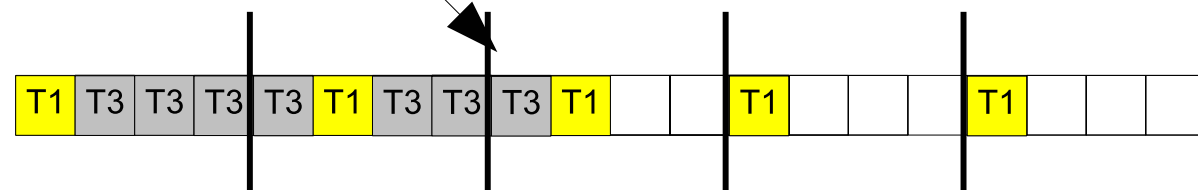
Ordonnancement global (12)

- Anomalies d'ordonnancement :

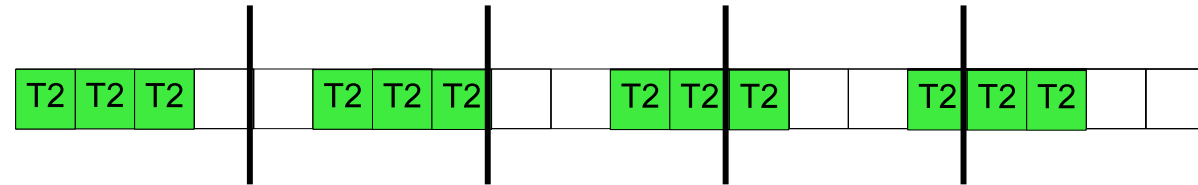
Echéance manquée de T3

	Ci	Pi	Di
T1	1	5	2
T2	3	5	3
T3	7	20	8

P1



P2



0 4 8 12 16

- Migration possible chaque unité de temps.
- Augmentation de P_1 qui conduit à un jeu de tâches **non faisable**.

Ordonnancement global (13)

- **Principe des algorithmes Pfair :**
 - On cherche un partage équitable des processeurs (Proportionate Fair [AND 05]).
 - Algorithme optimal si processeurs identiques + tâches périodiques synchrones échéances sur requêtes.
 - Orienté échéances.
 - Migration nécessaire entre chaque unité de temps.
 - Pas de parallélisme.
 - Beaucoup de commutations de contexte.

Ordonnancement global (14)

- **Principe des algorithmes Pfair :**

1. On cherche à exécuter les tâches selon un taux constant, c-à-d :

$$\forall i : WT(T_i, t) = t \cdot \frac{C_i}{P_i}$$

2. Peut être approché par :

$$retard(T_i, t) = WT(T_i, t) - \sum_{k=0}^{t-1} Sched(T_i, k)$$

où $Sched(T_i, t) = 1$ quand T_i est ordonnancée dans intervalle $[t, t + 1[$ et $Sched(T_i, t) = 0$ sinon

3. Un ordonnancement est dit Pfair si et seulement si :

$$\forall i, T_i, t : -1 \leq retard(T_i, t) \leq 1$$

4. Un ordonnancement Pfair est par construction faisable.

Ordonnancement global (15)

- **Fonctionnement des algorithmes Pfair :**

- Pour assurer que chaque tâche soit exécutée quasiment à un taux constant, on la découpe en plusieurs sous-tâches avec une capacité d'**une** unité de temps (quantum).
- Affectation d'une priorité dynamique $d_{(T_i,j)}$ et d'une date de réveil $r_{(T_i,j)}$ à chaque sous-tâche :

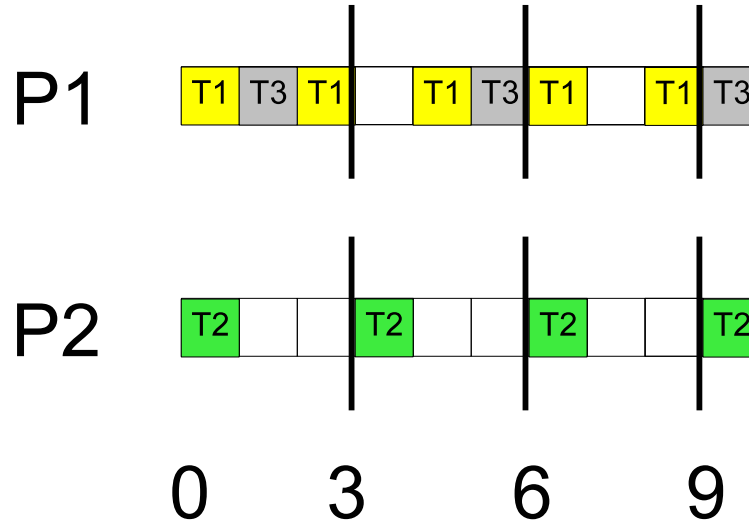
$$d_{(T_i,j)} = \left\lceil \frac{j}{C_i/P_i} \right\rceil$$

$$r_{(T_i,j)} = \left\lceil \frac{j-1}{C_i/P_i} \right\rceil$$

- Ordonnancement selon $d_{(T_i,j)}$ (échéance la plus courte).
- Variantes (optimales) si échéances identiques des sous-tâches: PF , PD , PD^2 .

Ordonnancement global (16)

	Ci	Pi
T1	1	2
T2	1	3
T3	2	9



$$\bullet d_{(T_1,1)} = \left\lceil \frac{1}{1/2} \right\rceil = 2$$

$$\bullet d_{(T_2,1)} = \left\lceil \frac{1}{1/3} \right\rceil = 3$$

$$\bullet d_{(T_3,1)} = \left\lceil \frac{1}{2/9} \right\rceil = [4, 5] = 5$$

$$\bullet d_{(T_3,2)} = \left\lceil \frac{2}{2/9} \right\rceil = 9$$

$$\bullet r_{(T_1,1)} = \left\lceil \frac{1-1}{1/2} \right\rceil = 0$$

$$\bullet r_{(T_2,1)} = \left\lceil \frac{1-1}{1/3} \right\rceil = 0$$

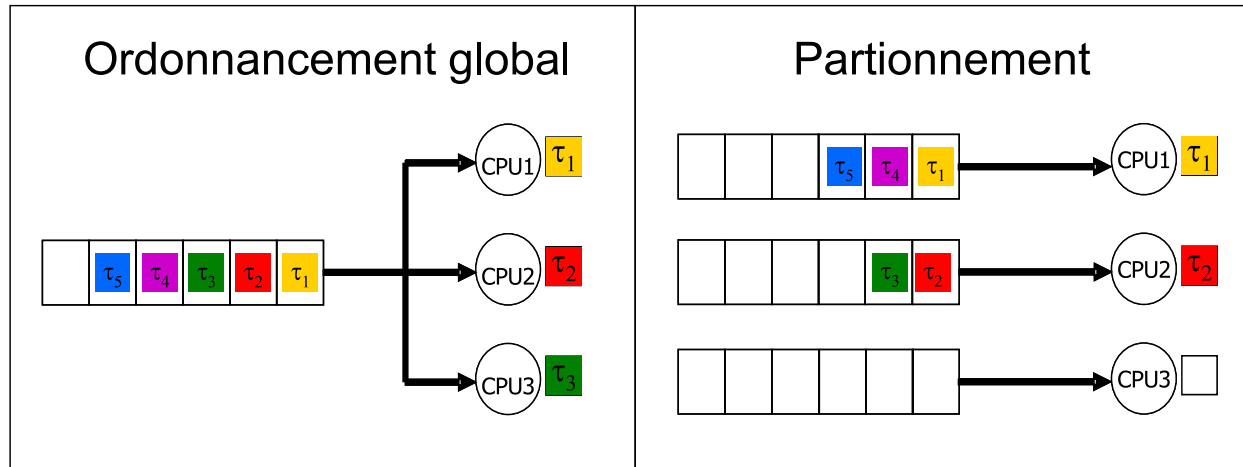
$$\bullet r_{(T_3,1)} = \left\lceil \frac{1-1}{2/9} \right\rceil = 0$$

$$\bullet r_{(T_3,2)} = \left\lceil \frac{2-1}{2/9} \right\rceil = [4, 5] = 5$$

Ordonnancement global (17)

- **L'ordonnancement global est donc un domaine différent :**
 - Propriétés différentes (dominance/optimalité des algorithmes, instant critiques, période d'étude, ...).
 - Paramètres supplémentaires de l'architecture : migration, affectation tâches/processeurs, types de processeurs, ...
- ⇒ Résultats/tests de faisabilité similaires (basés sur les pires temps de réponse ou le taux d'utilisation) mais différents, moins généraux et plus nombreux aussi !
- ⇒ Nous n'avons pas abordé les dépendances (ressources partagées, précédences).
- ⇒ Nous nous sommes limités à des processeurs identiques et à un modèle de tâches simplifié.

Systèmes répartis ciblés

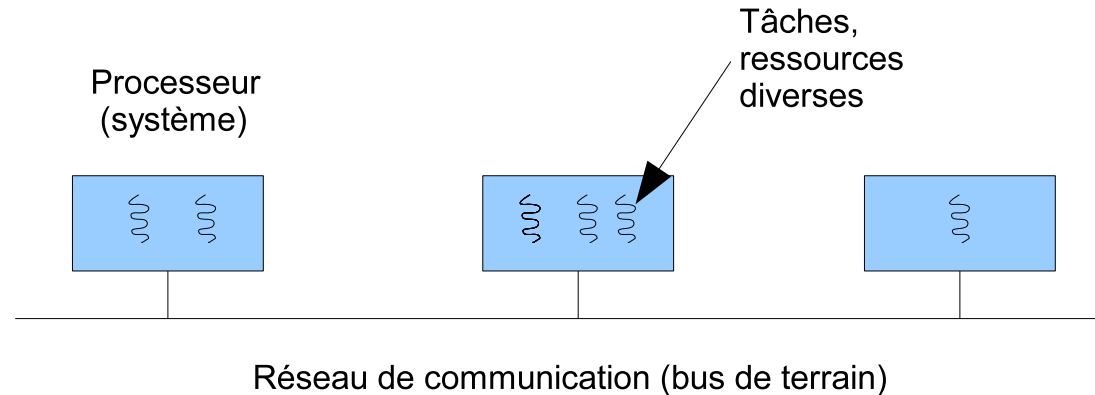


- Ordonnancement dans un système réparti/multi-processeurs :
 1. **Ordonnancement global** : choisir d'abord la tâche, puis placer la tâche sur un des processeurs libres. Approche en-ligne.
 2. **Ordonnancement par partitionnement** : placement des tâches pour ordonnancement local, puis vérification éventuelle des délais de bout en bout. Approche hors-ligne.

Sommaire

1. Introduction et rappels.
2. Ordonnancement global.
3. Ordonnancement par partitionnement.
4. Résumé.
5. Acronymes.
6. Références.
7. Remerciements.

Approche par partitionnement (1)



1. **Quel métier** : concepteur de systèmes (équipementiers) ou de systèmes de systèmes ?
2. **Conception d'un système** afin de vérifier :
 - Des contraintes locales. Test de la faisabilité processeur par processeur. Fonctions centralisés sur un processeur.
3. **Conception du système de systèmes** :
 - Configuration du réseau et placement des tâches, afin de vérifier:
 - Des délais/contraintes de bout en bout : contraintes liant plusieurs processeurs entre eux [TIN 94, LEB 95, RIC 01, CHA 95]. Fonctions réparties sur plusieurs processeurs.

Approche par partitionnement (2)

1. Placement des tâches ? Paramètres des tâches (priorités) ?
2. Modélisation des contraintes de précédence émetteur/récepteur ?
3. Calcul temps acheminement des messages (accès au réseau) ?
4. Calcul délais de bout en bout émetteur/récepteur ?
5. Calcul temps de mémorisation dans les tampons. Taille des tampons de communications (ex: sockets) ?

Sommaire

- Approche par partitionnement :
 1. Placement de tâches, paramètres des tâches.
 2. Contraintes de précédence.
 3. Ordonnancement de messages.
 4. Contraintes de bout en bout.
 5. Dimensionnement de tampons.

Placement de tâches (1)

- **Comment statiquement affecter un ensemble de tâches à un ensemble de processeurs ?**
 - Problème de "bin-packing" (emballage): trouver le rangement le plus économique possible pour un ensemble d'articles dans des boîtes.
 - NP-difficile \implies heuristiques de partitionnement généralement basées sur des tests de faisabilité.
- **Exemples de paramètre :**
 - Processeurs (identiques ou non), tâches (fixé ou non, priorités, périodes, capacités).
 - Prendre en compte les communications entre les tâches, les ressources partagées ...
- **Exemples de fonction objective:**
 - Minimiser les nombres de processeurs, les communications, les latences, ...
- **Difficultés :**
 - Comment comparer l'efficacité des heuristiques.
 - Paramètres nombreux et souvent dépendants entre eux.
 - Fonctions objectives contradictoires. Contraintes techniques.

Placement de tâches (2)

- **(Exemple de) fonctionnement général de ces heuristiques de placement :**
 1. Les tâches sont classées selon un paramètre donné (ex : période, priorité, charge processeur, ...).
 2. On traite les tâches séquentiellement, selon le classement.
 3. Pour une tâche donnée, on recherche un processeur en parcourant la liste des processeurs selon une politique (ex : Best Fit, First Fit, Next Fit, ...).
 4. Les tâches sont assignées au processeur afin que la charge processeur maximum théorique ne soit pas dépassée grâce à un test de faisabilité.
 5. On termine lorsque toutes les tâches sont attribuées aux processeurs.

Placement de tâches (3)

- **Exemples d'heuristiques de placement :**
 - Très souvent basées sur RM [OH 93] car tests de faisabilité faciles à exhiber et approche pratique/statique.
 - Principales heuristiques d'ordonnancement : Rate-Monotonic First-Fit, Next-Fit ou Best-Fit [DHA 78, OH 93].
 - Autres heuristiques de placement : RM-FFDU[OH 95], RM-ST, RM-GT et RM-GT/M [BUR 94], RM-Matching[DAV 09], EDF-FFD (First Fit Decreasing) [BAR 03], EDF-FF/EDF-BF, FFDUF[DAV 09].

Placement de tâches (4)

- **Exemple de Rate Monotonic Next Fit :**
 1. Les tâches sont classées dans l'ordre croissant de leur période. On commence par la tâche $i = 0$ et le processeur $j = 0$.
 2. On assigne la tâche i au processeur j si la condition de faisabilité est respectée (ex: $U \leq 0.69$).
 3. Dans le cas contraire, on assigne la tâche i au processeur $j + 1$.
 4. On passe à la tâche suivante.
 5. Arrêt lorsqu'il n'y a plus de tâche à ordonnancer, $j =$ nombre de processeurs nécessaires.

Placement de tâches (5)

- Problème de Next Fit = taux d'utilisation des processeurs parfois bas. Autres heuristiques similaires:
 - First Fit : pour chaque tâche i , regarder les processeurs en commençant par $j = 0$ et placer la tâche i sur le premier processeur dont le test de faisabilité est vérifié.
 - Best Fit : pour chaque tâche i , regarder tous les processeurs et placer la tâche i sur le processeur dont le test de faisabilité est vérifié et qui maximise le taux d'occupation du processeur.
 - ...

Placement de tâches (6)

- Ratio d'approximation Ra [DAV 09] :

Algorithme	Ra
RM-NF	2.67
RM-FF, RM-BF	2.33
EDF-FF, EDF-BF	1.7

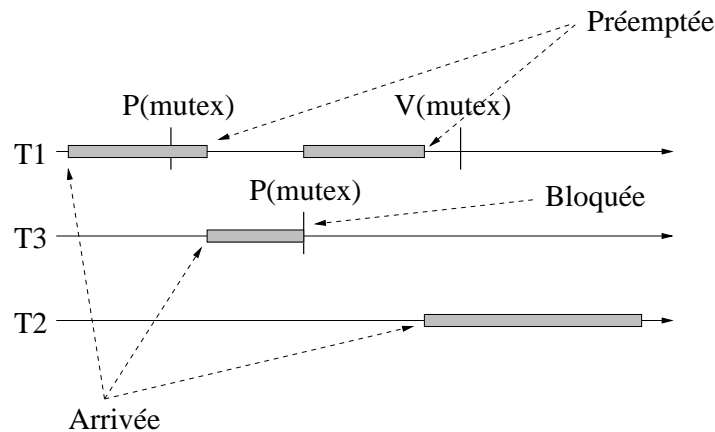
- Critère de comparaison d'algorithmes :

$$Ra = \frac{Ma}{Mo}$$

- $Ra = 1$ pour l'algorithme optimal.
- Ma nombre de processeurs pour l'algorithme a .
- Mo nombre de processeurs pour l'algorithme optimal.

Paramètres des tâches (1)

- Répartir une application implique une **affectation globale** des priorités de tâches \implies inversion de priorités, niveaux de priorité offerts sur chaque processeur (caractéristiques du système d'exploitation).

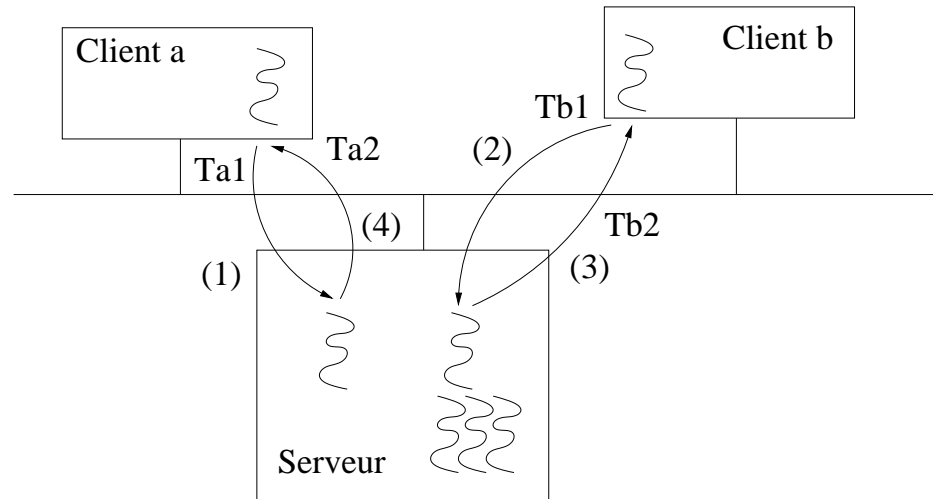


- Comment éviter les inversions de priorité ? \implies PCP, PIP ...
- Comment calculer B_i , la borne sur le temps de blocage ? Temps de réponse :

$$r_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{r_i}{P_j} \right\rceil C_j$$

Paramètres des tâches (2)

- **Exemple** : application client-serveur.



- Un serveur est une ressource partagée \implies inversion de priorité entre les invocations T_{a1}/T_{a2} et T_{b1}/T_{b2} .
- Solution : comme PCP/PIP, affectation de priorités afin de garantir l'absence d'interruption.
- Heuristique (conjointe) de partitionnement et d'affectation de priorités. Boîtes noires, composants sur étagère ?

Sommaire

- Approche par partitionnement :
 1. Placement de tâches, paramètres des tâches.
 2. Contraintes de précedence.
 3. Ordonnancement de messages.
 4. Contraintes de bout en bout.
 5. Dimensionnement de tampons.

Contraintes de précédence (1)

- Dépendances entre les tâches :
 1. Partage de ressources (ex : PIP, PCP, SRP, ...).
 2. Contraintes de précédence (ex : communications).

Contraintes de précédence (2)

- Problèmes soulevés :
 1. Calculer un ordonnancement qui respecte les contraintes de précédence (simulation, exécution).
 2. Décider de la faisabilité hors ligne lorsque les tâches sont soumises à des contraintes de précédence.

Contraintes de précédence (3)

- Principales approches :
 1. Conditions initiales (paramètre S_i). Exécution et faisabilité (avec formules spécifiques).
 2. Affectation des priorités (Chetto/Blazewicz [BLA 76, CHE 90]). Applicabilité limitée. Faisabilité et exécution.
 3. Modifications des délais critiques (Chetto/Blazewicz). Applicabilité limitée. Faisabilité et exécution.
 4. Utilisation du paramètre "Jitter" [TIN 94]. Seulement pour la faisabilité (pire cas éventuellement très grand).
 5. Heuristique d'ordonnancement (Xu et Parnas [XU 90]). Pas de faisabilité.

Contraintes de précédence (4)

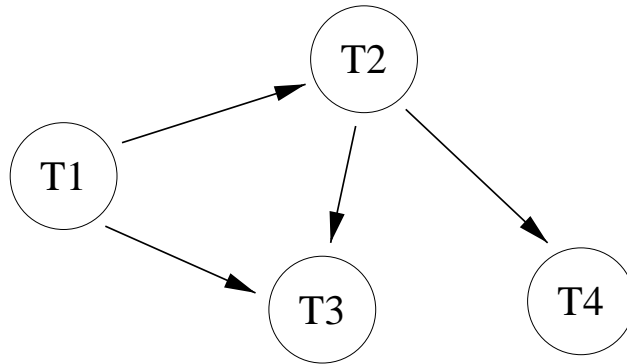
- Principe de la solution de Blazewicz [BLA 76] et de Chetto et al. [CHE 90] : rendre les tâches indépendantes en modifiant leurs paramètres.
- Hypothèses : Tâches soit apériodiques, soit périodiques de même période.
- Méthodes proposées :
 1. Modification pour RM :

$$\bullet \forall i, j \mid i \prec j : \text{priorite}_i > \text{priorite}_j$$

2. Modification pour EDF :

$$\bullet D_i^* = \min(D_i, \min(\forall j \mid i \prec j : D_j^* - C_j)).$$

Contraintes de précédence (5)



	C_i	D_i	D_i^*
T_4	2	14	14
T_3	1	8	8
T_2	2	10	7
T_1	1	5	5

- Exemple : EDF + tâches apériodiques.

- $D_4^* = 14; D_3^* = 8;$

- $D_2^* = \min(D_2, D_3^* - C_3, D_4^* - C_4) = \min(10, 8 - 1, 14 - 2) = 7;$

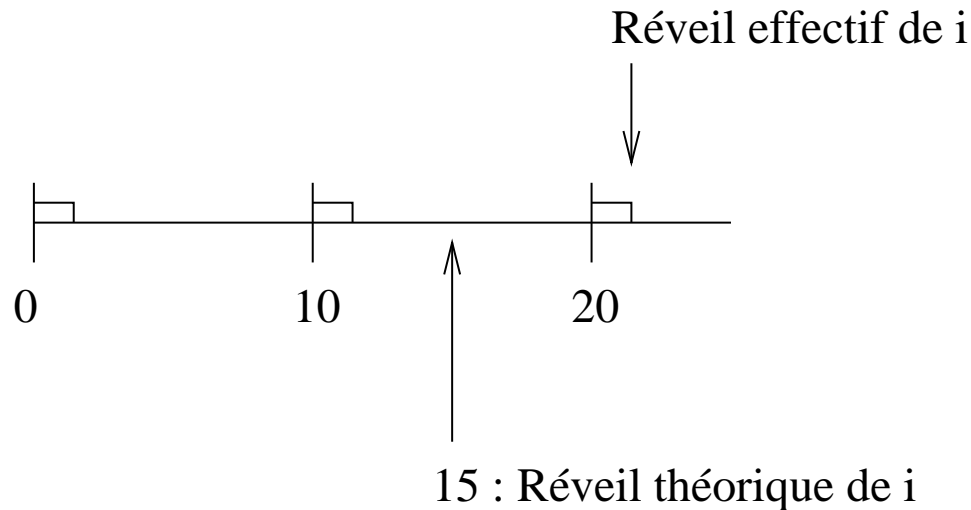
- $D_1^* = \min(D_1, D_2^* - C_2, D_3^* - C_3) = \min(5, 7 - 2, 8 - 1) = 5;$

Contraintes de précédence (6)

- Utilisation du **Jitter**. Exemple historique \implies le timer d'un système est modélisé comme une tâche périodique avec

$$P_{timer} = 10 \text{ ms}, C_{timer} = 3 \text{ ms}.$$

- On souhaite réveiller une tâche i à l'instant $t = 15 \text{ ms}$.



Date effective de réveil de la tâche $i = 23 \text{ ms}$. Gigue $J_i = 8 \text{ ms}$.

- Temps de réponse :

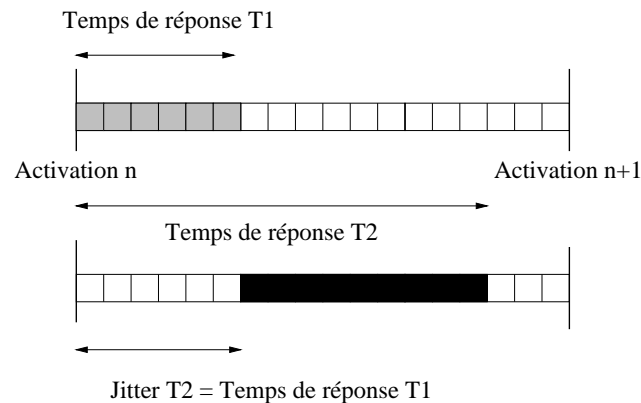
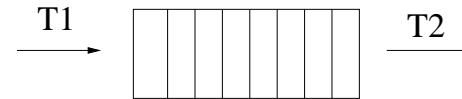
$$r_i = w_i + J_i$$

$$w_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i + J_j}{P_j} \right\rceil C_j$$

Contraintes de précédence (7)

T1 : C1=6 ; P1=18 (gris)

T2 : C2=9 ; P2=18 (noir)



- Exemple du producteur/consommateur :
 - T1 et T2 sont activées toutes les 18 unités de temps.
 - T1 lit un capteur et transmet la valeur vers T2 qui l'affiche à l'écran.
 - T2 doit être activée sur terminaison de T1.
 - Quel est le temps de réponse de T2 ?

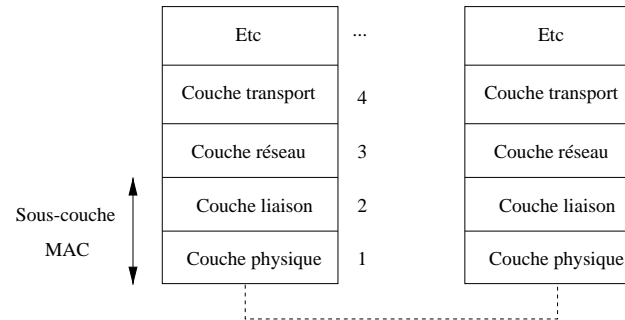
Sommaire

- Approche par partitionnement :
 1. Placement de tâches, paramètres des tâches.
 2. Contraintes de précédence.
 3. Ordonnancement de messages.
 4. Contraintes de bout en bout.
 5. Dimensionnement de tampons.

Ordonnancement de messages (1)

- Problèmes soulevés :
 1. Comment déterminer le temps de communication d'un message ? \implies réseaux **déterministes, prédictibles**.
 2. Comment les messages doivent ils se partager la ressource réseau tout en respectant leurs contraintes temporelles.

Ordonnancement de messages (2)



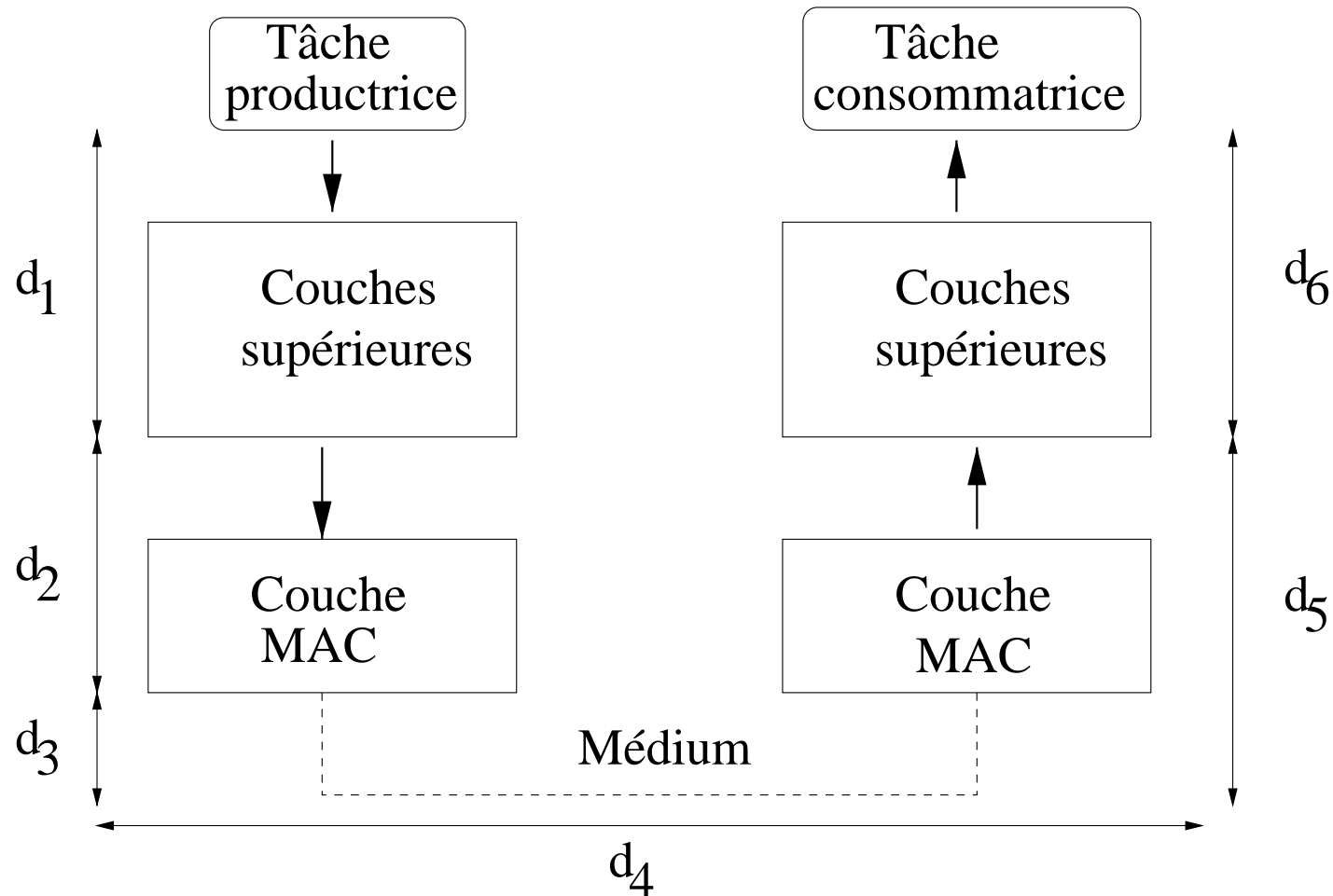
- Transport : application, expression/négociation contraintes de temps.
 - Réseau : commutation, routage.
 - Physique/Liaison/MAC : partage du medium \Rightarrow **protocole d'arbitrage.**
-
- Bus/réseaux temps réel = MAC + couche applicative de type "transport".
 - Couche applicative adaptée (événements, variables périodiques),
 - Débit faible, petite taille, contraintes environnement (température, vibrations, etc).

Ordonnancement de messages (3)

- **Exemples de réseaux déterministes :**
 - Productique et robotique : MAP [DWY 91], Token bus/IEEE 802.4 [PUJ 95], FIP [CIA 99], Profibus [STR 96].
 - Aéronautique/spatial/naval : DigiBus, STD MIL 1553, ARINC 429 et 629. Aviation civile et militaire, la sonde PathFinder, systèmes navals [DEC 96, HEA 96].
 - Industrie automobile : bus CAN [ZEL 96].
 - Domotique : BatiBus[CIA 99].

Ordonnancement de messages (4)

- Temps de communication composé de [COT 00] :



Ordonnancement de messages (5)

- Temps de communication composé de [COT 00] :
 - d_1 et d_6 = délais de traversée des couches ; d_5 = délai de réception. Calcul facile, peu variable et borné.
 - d_3 = délai de transmission sur le médium. Variable, calcul facile \implies taille message/débit.
 - d_4 = délai de propagation. Variable, calcul facile \implies taille réseau/vitesse.
 - d_2 = **délai d'attente pour l'accès au réseau. Variable, dépend du protocole d'arbitrage.**

Ordonnancement de messages (6)

- L'arbitrage définit la méthode d'accès au médium.

Objectif principal : comportement prédictible des temps accès.

- Éléments de taxinomie des protocoles d'arbitrage :
 - Algorithme par coopération ou compétition.
 - Arbitrage symétrique ou asymétrique. Arbitrage centralisé/réparti. Notion de maîtres, d'arbitres et d'esclaves : qui prend l'initiative de la communication ?
 - Synchrones ou asynchrones \implies y a t il une horloge globale à tous les coupleurs ?

Ordonnancement de messages (7)

- Principaux protocoles d'arbitrage[UPE 94] :
 - **Mono-maître** : ARINC 429, DigiBus
 - 1 émetteur et n récepteurs. Bus mono-directionnel.
 - n bus si communications bi-directionnelles.
 - **CSMA/CA** : CAN, Batibus
 - Une priorité fixe est associée à chaque station.
 - Chaque station émet quand elle le souhaite.
 - Les éventuelles collisions sont réglées par le biais de la priorité : la station de plus forte priorité obtient le médium.
 - **Protocole à jeton** : Token-Bus, Profibus
 - Une jeton circule selon un ordre fixe entre les stations (ex : topologie en boucle).
 - Une station reçoit le jeton émet ses données et transmet le jeton à la station suivante.

Ordonnancement de messages (8)

● **Polling** : FIP, MIL-STD-1553

- Une station dédiée, dite "arbitre", émet un message sur le médium, invitant un "maître" à émettre.
- En réponse, le maître émet ses données. Puis, l'arbitre interroge le maître suivant.
- L'arbitre dispose d'une liste d'invitations fixe qu'il parcourt de façon séquentielle.

● **TDMA [KOO 95]** : ARINC 629

- Une station dédiée, appelée "arbitre" émet cycliquement une trame de synchronisation (timers).
- Chaque maître émet alors, à un instant relatif par rapport à la trame de synchronisation.
- L'ordre d'émission des maîtres est prédéterminé et fixe \implies partage temporel du médium.

Ordonnancement de messages (9)

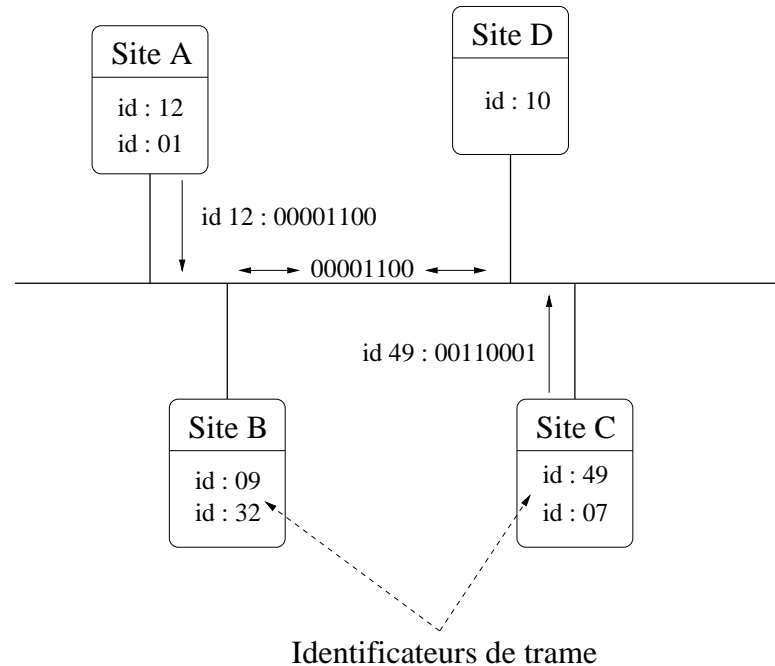
- **CSMA/CA : le bus CAN**

- Créé par Bosh et Intel pour les applications automobiles.
- Utilisé initialement par Mercedes-Benz pour la Classe S. Adopté aujourd'hui par de nombreux constructeurs automobiles.
- Leader pour les applications automobiles.
- Réseau multi-mâîtres symétriques.
- Transmission par diffusion.
- Topologie en bus, paires torsadées, généralement longueur maximum 40 m pour un débit de 1 MBit/s.
- Composants très fiables à faibles coût (automobile).
- Services de sûreté de fonctionnement très évolués (CRC, acquittement, diagnostique de coupleurs).

Ordonnancement de messages (10)

- Identificateurs de trame = priorités. Identificateurs uniques et émis par une seule station.
- Arbitrage CAN = arbitrage par compétition = CSMA non destructif (CSMA/CA) :
 - Lorsque le bus est libre, émission bit à bit de l'identificateur avec écoute de la porteuse.
 - Un bit à 1 (récessif) est masqué par un bit à 0 (dominant).
 - Tout coupleur lisant un bit différent de celui qu'il vient d'émettre passe en réception. Puis, réémet immédiatement lorsque la porteuse est de nouveau libre.
 - Émission bit à bit + écoute porteuse = faible débit/taille du réseau.

Ordonnancement de messages (11)



- A, B, C et D sont des maîtres (émetteurs). Chacun détient une liste d'identificateurs uniques.
- A et C commencent à émettre en même temps les identificateurs 12 et 49 (priorités).
- Lors de la transmission du 3 ème bit, le site C passe en réception.
- Le site A gagne et transmet son information sur le bus (de 0 et 8 octets).

Ordonnancement de messages (12)

- De l'ordonnancement de tâches aux messages :

Ordo. de tâches	Ordo. de messages
Tâches Processeur Capacité Temps d'interférence Temps de réponse	Messages Médium de communication Temps de transmission + temps de propagation + temps de traversée Temps d'accès Temps de communication de tâche à tâche

- Points communs : échéance, période, etc.
- Spécificité : caractère **non préemptif**.

Ordonnancement de messages (13)

- Comment calculer le temps de communication d'un message périodique ? En appliquant les mêmes méthodes que pour le partage du processeur.
- Analyse avec le bus CAN :
 - Identificateur d'un message CAN = priorité fixe d'une tâche. Affecter les identificateurs selon Rate Monotonic.
 - Le protocole d'arbitrage de CAN est un ordonnanceur à priorité fixe *non préemptif*. Temps de réponse dans le cas non préemptif [GEO 96]:

$$r_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{r_i}{P_j} \right\rceil C_j + \max(C_k, \forall k \in lp(i))$$

avec $lp(i)$, l'ensemble des tâches de plus faible priorité que i .

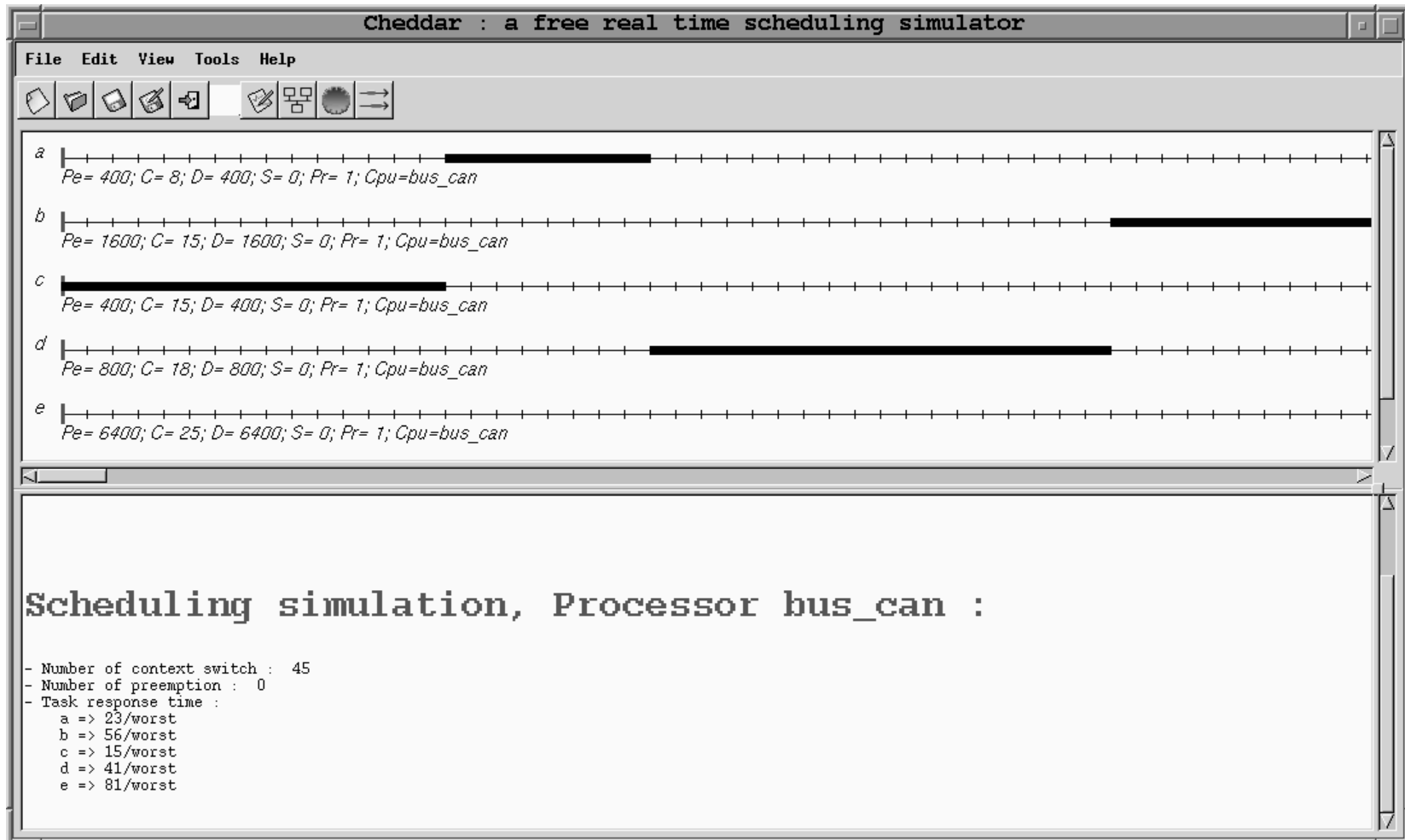
Ordonnancement de messages (14)

- Exemple :

Msg	Période (en ms)	Transmi./Propag. (en μs)	Accès (en μs)	Total (en μs)
a	4	80	150	230
b	16	150	410	560
c	4	150	0	150
d	8	180	230	410
e	64	250	560	810

- Temps de traversée des couches et de réception supposés nuls.

Ordonnancement de messages (15)



- Outil de simulation Cheddar :

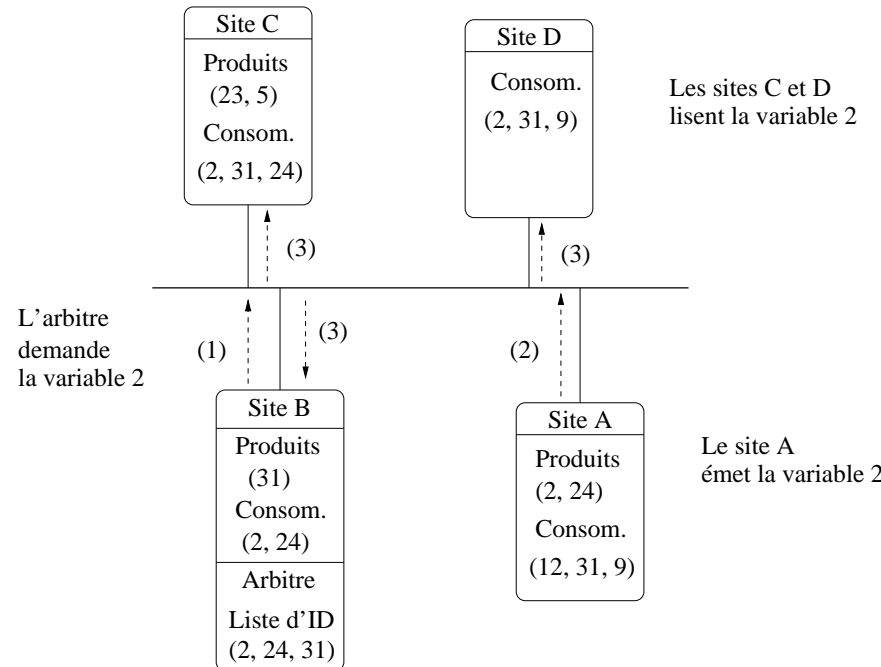
<http://beru.univ-brest.fr/~singhoff/cheddar>

Ordonnancement de messages (16)

- **Polling : le bus FIP**

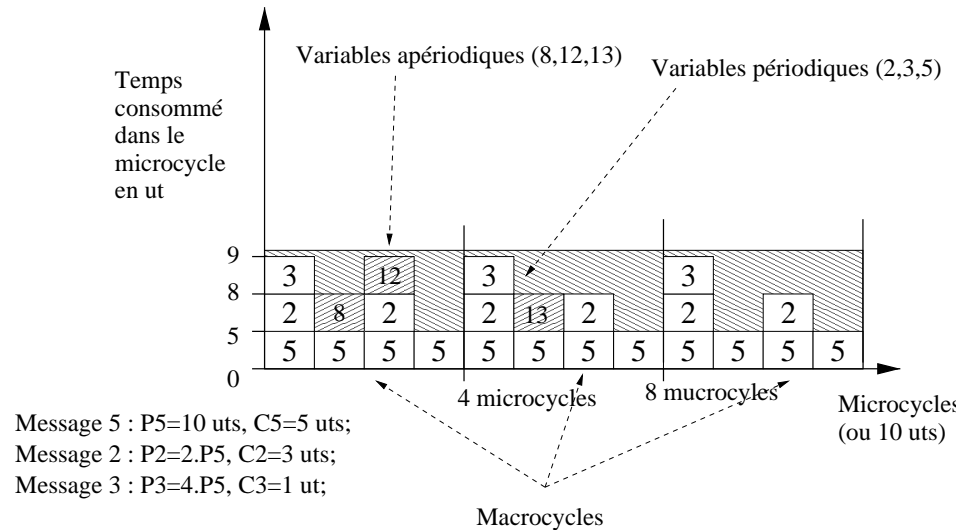
- Réseau pour la productique.
- Polling : multi-maîtres, arbitre centralisé.
- Topologie en bus ou en étoile. Paire torsadée. Longueur de 4000 m avec un débit de 1Mbit/s.
- Sûreté de fonctionnement : coupleur bi-médium.
- Orienté variables (de 1 à 128 octets). Variable = valeur de capteur.
- Protocole de type producteurs/consommateurs.
- Diffusion de variables sur le bus.

Ordonnancement de messages (17)



- L'arbitre détient une liste de variables qu'il exploite séquentiellement :
 - L'arbitre indique la variable à transmettre par un message (1).
 - Le producteur émet la valeur sur le bus (2).
 - Les consommateurs lisent le bus (3).

Ordonnancement de messages (18)

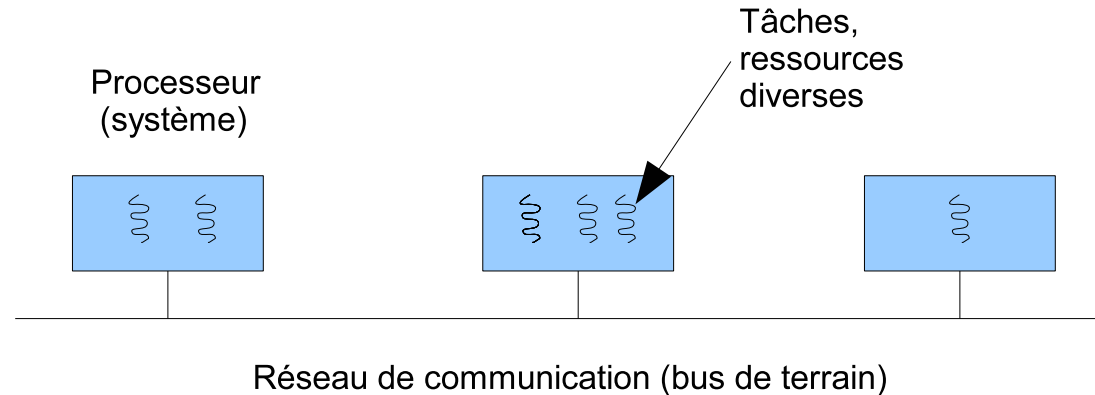


- La table d'arbitrage :
 - L'arbitre est une tâche périodique, qui se réveille à chaque début de microcycle pour la transmission des variables périodiques d'abord, puis, s'il reste du temps, des variables apériodiques.
 - Quand les variables sont elles demandées par l'arbitre : notion de microcycle (plus petite période) et de macrocycle (PPCM des périodes).

Sommaire

- Approche par partitionnement :
 1. Placement de tâches, paramètres des tâches.
 2. Contraintes de précédence.
 3. Ordonnancement de messages.
 4. Contraintes de bout en bout.
 5. Dimensionnement de tampons.

Faisabilité dans le cas réparti (1)

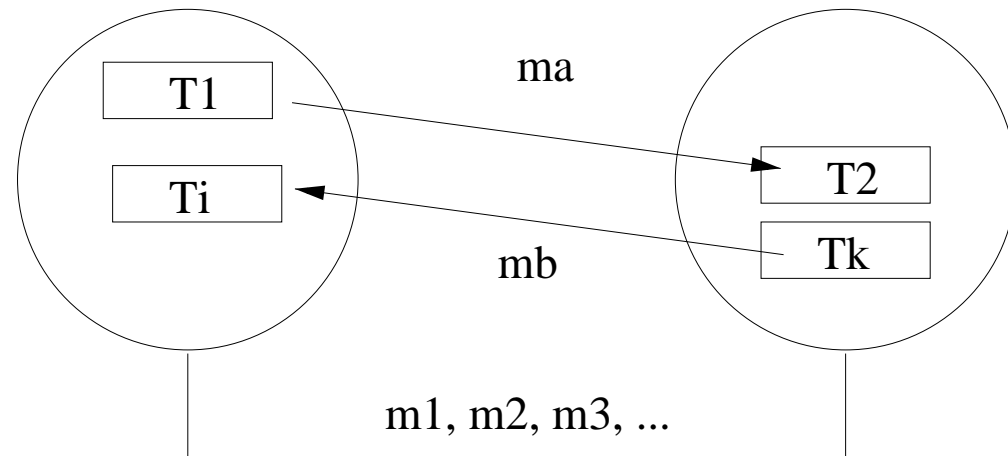


1. **Quel métier** : concepteur de systèmes (équipementiers) ou de systèmes de systèmes ?
2. **Conception d'un système** afin de vérifier :
 - Des contraintes locales. Test de la faisabilité processeur par processeur. Fonctions centralisés sur un processeur.
3. **Conception du système de systèmes** :
 - Configuration du réseau et placement des processeurs, afin de vérifier:
 - Des délais/contraintes de bout en bout : contraintes liant plusieurs processeurs entre eux. Fonctions réparties sur plusieurs processeurs.

Faisabilité dans le cas réparti (2)

- Plusieurs approches de vérification possibles :
 - Approche basée sur Chetto et Blazewicz \implies cours de Bertrand Dupouy sur les bus et réseaux de terrain.
 - Approche basée sur les offsets [BAT 97, PAL 03].
 - Approche Holistique [TIN 94, LEB 95, RIC 01, CHA 95].

Temps de réponse bout en bout (1)



- **Contrainte à vérifier :** $r_{(T1+ma+T2)} \leq D$
- **Le calcul Holistique :** injection du temps de réponse sous la forme d'une gigue[TIN 94] :
 - Soit r_{T1} , le temps de réponse de T1.
 - r_{ma} est calculé, en fonction du protocole d'arbitrage, des autres messages périodiques et tel que $J_{ma} = r_{T1}$.
 - r_{T2} est calculé avec $J_2 = r_{ma}$.
 - **Calcul itératif jusqu'à convergence.**

Temps de réponse bout en bout (2)

```
10   $\forall i : J_i := 0, r_i := 0, r'_i := 0;$   
20   $\forall i : \text{Calculer\_temps\_de\_réponse}(r_i);$   
30  Tant que  $(\exists i : r_i \neq r'_i)$  {  
40       $\forall i : J_i := \max(J_i, \forall j \text{ avec } j \prec i : r_j);$   
50       $\forall i : r'_i := r_i;$   
60       $\forall i : \text{Calculer\_temps\_de\_réponse}(r_i);$   
70  }
```

Si i est une tâche :

$$r_i = J_i + w_i \text{ avec } w_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i + J_j}{P_j} \right\rceil C_j$$

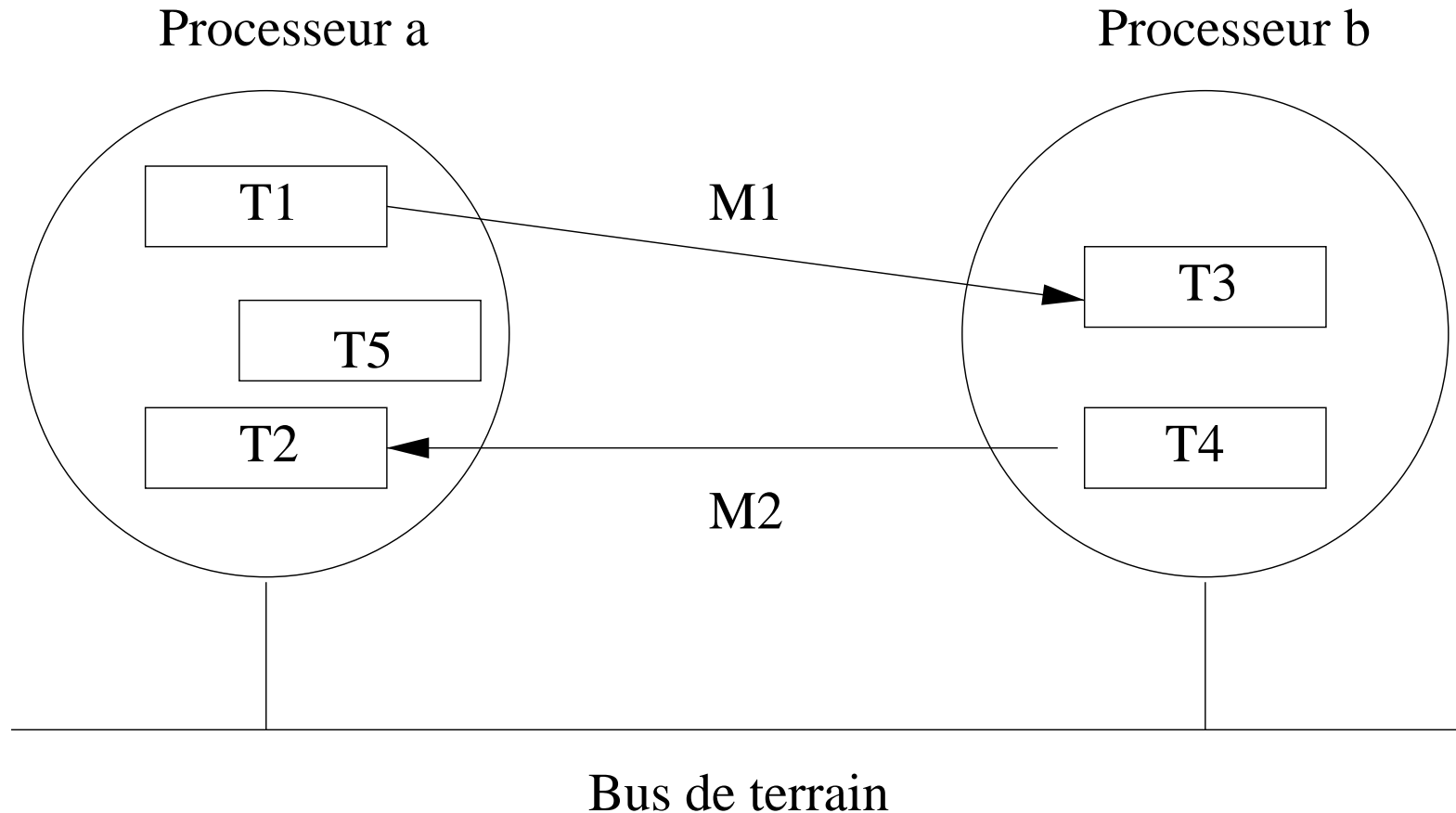
Si i est un message :

$$r_i = J_i + M_i$$

dont le temps de communication de tâche à tâche est borné par M_i .

Temps de réponse bout en bout (3)

- Soit le placement suivant :



Temps de réponse bout en bout (4)

- Et les paramètres suivants :

Tâche	Période	Capacité	Priorité	Processeur
T_1	100	4	1	a
T_2	60	5	2	a
T_3	100	3	2	b
T_4	60	2	1	b
T_5	90	3	3	a

Message	Période	Délai de communication de tâche à tâche
M_1	100	6
M_2	60	1

Temps de réponse bout en bout (5)

- Lignes 10-20 : $\forall i : J_i = 0$

Message/Tâche	M_1	M_2	T_1	T_2	T_3	T_4	T_5
J_i	0	0	0	0	0	0	0
r_i	6	1	4	9	5	2	12

- Première itération, lignes 40-60 : modification jitter + calcul temps de réponse. $J_{M_1} = r_{T_1}$, $J_{M_2} = r_{T_4}$, $J_{T_3} = r_{M_1}$ et $J_{T_2} = r_{M_2}$.

Message/Tâche	M_1	M_2	T_1	T_2	T_3	T_4	T_5
J_i	4	2	0	1	6	0	0
r_i	6+4 =10	1+2 =3	4	9+1 =10	5+6 =11	2	12

Temps de réponse bout en bout (6)

- **Deuxième itération, lignes 40-60** : modification jitter + calcul temps de réponse. $J_{M_1} = r_{T_1}$, $J_{M_2} = r_{T_4}$, $J_{T_3} = r_{M_1}$ et $J_{T_2} = r_{M_2}$.

Message/Tâche	M_1	M_2	T_1	T_2	T_3	T_4	T_5
J_i	4	2	0	3	10	0	0
r_i	6+4 =10	1+2 =3	4	9+3 =12	5+10 =15	2	12

- **Troisième itération, lignes 40-60** : modification jitter $J_{M_1} = r_{T_1}$, $J_{M_2} = r_{T_4}$, $J_{T_3} = r_{M_1}$ et $J_{T_2} = r_{M_2}$. Jitters identiques => même temps de réponse = convergence et arrêt des calculs.

Message/Tâche	M_1	M_2	T_1	T_2	T_3	T_4	T_5
J_i	4	2	0	3	10	0	0

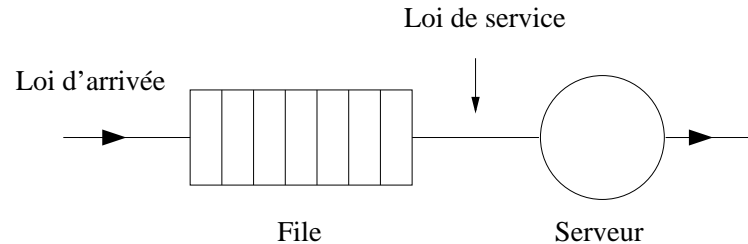
Sommaire

- Approche par partitionnement :
 1. Placement de tâches, paramètres des tâches.
 2. Contraintes de précédence.
 3. Ordonnancement de messages.
 4. Contraintes de bout en bout.
 5. Dimensionnement de tampons.

Dimensionnement des tampons

- Problèmes soulevés :
 1. Déterminer la taille des tampons afin d'éviter un éventuel débordement, et dans certain cas une famine.
 2. Déterminer le pire délai de mémorisation d'un message
⇒ calcul holistique.

Dimensionnement de tampon (1)



- **Objectif** : description loi arrivée λ et loi service μ .
- Notation de kendall : $\lambda/\mu/n$:
 - λ : loi d'arrivée des clients (M,G,D).
 - μ : loi de service des clients (M,G,D).
 - n : nombre de serveurs.
- Exemples : $M/M/1$, $M/D/1$, $M/G/1$, $P/P/1$, ...

Dimensionnement de tampon (2)

- **Critères de performance recherchés** : L =nombre moyen de message, W =délai de mémorisation moyen d'un message dans la file d'attente (W_q et L_q), ou dans le serveur (L_s et W_s). Avec :

$$W = W_q + W_s$$

$$L = L_q + L_s$$

- **Loi de Little** : relation entre délai/latence et taille/dimensionnement.

$$L = \lambda.W$$

$$W = \frac{L}{\lambda}$$

Dimensionnement de tampon (3)

- Critères de performance des principaux modèles :

	L	L_q	W	W_q
M/M/1	$\frac{\lambda W_s}{1-\rho}$	$\frac{\lambda^2 W_s^2}{1-\rho}$	$\frac{W_s}{1-\rho}$	$\frac{\lambda W_s^2}{1-\rho}$
M/G/1	$\lambda W_s + \frac{\lambda^2 (W_s^2 + \sigma^2)}{2(1-\rho)}$	$\frac{\lambda^2 (W_s^2 + \sigma^2)}{2(1-\rho)}$	$W_s + \frac{\lambda (W_s^2 + \sigma^2)}{2(1-\rho)}$	$\frac{\lambda (W_s^2 + \sigma^2)}{2(1-\rho)}$
M/D/1	$\lambda W_s + \frac{\lambda^2 W_s^2}{2(1-\rho)}$	$\frac{\lambda^2 W_s^2}{2(1-\rho)}$	$W_s + \frac{\lambda W_s^2}{2(1-\rho)}$	$\frac{\lambda W_s^2}{2(1-\rho)}$

- Ces modèles ne sont pas adaptés au modèle périodique de tâches de la théorie de l'ordonnancement temps réel : réveils périodiques, service lié à l'ordonnancement déterministe, ...

Dimensionnement de tampon (4)

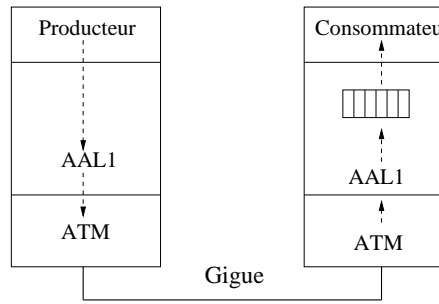
- **File d'attente P/P/1[LEG 03] :**

- "P" pour périodique.
- n tâches périodiques producteurs et 1 tâche consommateur.
- Contraintes temporelles des tâches respectées et $\forall i : D_i \leq P_i$.
- Loi de conservation du débit : $\frac{\lambda}{\mu} \leq 1$.
- 1 message produit ou consommé par activation.

- **Test de faisabilité :**

Harmonique	Non harmonique
$L_{max} = 2.n$	$L_{max} = 2.n + 1$
$W_{max} = 2.n.P_{cons}$	$W_{max} = (2.n + 1).P_{cons}$

Dimensionnement de tampon (5)



- **Résultat dans le cas ATM/AAL1 :**

1. Taille maximum des tampons régulant le trafic :

$$L_{max} = \left\lceil \frac{W_{max}}{d} \right\rceil$$

où $W_{max} = 2.gigue$. est le **délai de mémorisation d'une cellule** avec $gigue = delai_{max} - delai_{min}$.

2. C'est une illustration de la loi de Little : $L_{max} = \lambda_{max}.W_{max}$ avec $\lambda_{max} = 1/d$

Dimensionnement de tampon (6)

- Application de la méthode AAL1/Little :

1. Délai de mémorisation :

$$W_{max} = (y + 1).P_{cons} + D_{cons}$$

y messages déjà présents dans le tampon. $cons$ = tâche consommateur.

2. Borne nombre de message :

$$L_{max} = \max_{\forall y \geq 0} \left(\sum_{prod \in PROD} \left\lceil \frac{W_{max} + O_{prod}}{P_{prod}} \right\rceil - y \right)$$

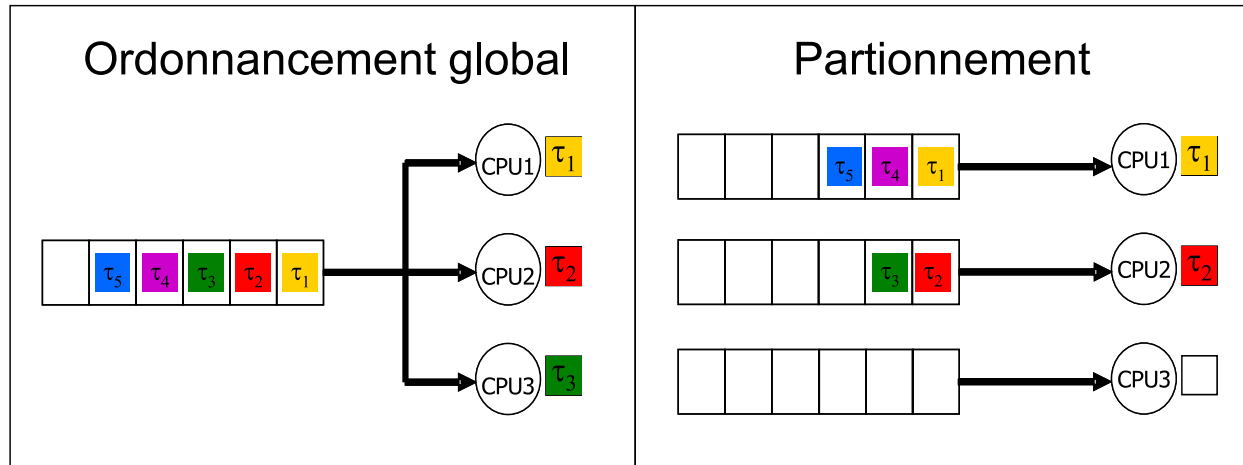
O_{prod} = désynchronisation d'un producteur vis-à-vis du consommateur.

3. **Obtention** de L_{max} par études aux limites de y . Obtention de W_{max} par application de Little.

Sommaire

1. Introduction et rappels.
2. Ordonnancement global.
3. Ordonnancement par partitionnement.
4. Résumé.
5. Acronymes.
6. Références.
7. Remerciements.

Résumé (1)



- Ordonnancement dans un système réparti/multi-processeurs :
 1. **Ordonnancement global** : choisir d'abord la tâche, puis placer la tâche sur un des processeurs libres. Approche en-ligne.
 2. **Ordonnancement par partitionnement** : placement des tâches pour ordonnancement local, puis vérification éventuelle des délais de bout en bout. Approche hors-ligne.

Résumé (2)

- Difficultés supplémentaires liées à la répartition :
 - Ressources : plusieurs processeurs + réseaux + mémoire à gérer globalement.
 - Ordonnancement global : résultats peu nombreux.
 - Partitionnement : placement des tâches, paramètres des tâches, contraintes de précedence, ordonnancement de messages, dimensionnement de tampons de communications.
 - Vérification de nouvelles contraintes temporelles : contraintes de bout en bout.
- Pas de maturité du domaine :
 - Peu de résultats théoriques généraux qui se soient imposés.
 - Beaucoup de solutions ad-hoc.
 - Déterminisme/précision nécessairement moins bon à cause des communications.
 - Un peu de pratique ? la majorité des méthodes analytiques présentées dans ce cours ont été implantées dans Cheddar (cf. page 44).

Sommaire

1. Introduction et rappels.
2. Ordonnancement global.
3. Ordonnancement par partitionnement.
4. Résumé.
5. Acronymes.
6. Références.
7. Remerciements.

Acronymes

- **MAP.** Manufacturing Automation Protocol.
- **CAN.** Controller Area Network.
- **FIP.** Factory Instrumentation Protocol.
- **FIFO.** First In First Out.
- **MAC.** Medium Acces Control.
- **PROFIBUS.** Process Field Bus.
- **CSMA/CA.** Carrier Sence Multiple Access and Collision Detection Avoidance.
- **TDMA.** Time Division Multiple Access.

Références

- [AND 05] James Anderson, Philip Holman, and Anand Srinivasan. « Fair Scheduling of Real-time Tasks on Multiprocessors ». In *Handbook of Scheduling*, 2005.
- [BAR 03] S. K. Baruah and S. Funk. « Task assignment in heterogeneous multiprocessor platforms ». Technical Report, University of North Carolina, 2003.
- [BAT 97] I. Bate and A. Burns. « Schedulability Analysis of Fixed priority Real-Time Systems with Offsets », 1997.
- [BER 07] M. Bertogna. « Real-Time scheduling analysis for multiprocessor platforms ». In *Phd Thesis, Scuola Seprrore Sant Anna, Pisa*, 2007.
- [BLA 76] J. Blazewicz. « Scheduling Dependant Tasks with Different Arrival Times to Meet Deadlines ». In. Gelende. H. Beilner (eds), *Modeling and Performance Evaluation of Computer Systems*, Amsterdam, North-Holland, 1976.
- [BUR 94] A. Burchard, J. Liebeherr, Y. Oh, and S. H. Son. « Assigning real-time tasks to homogeneous multiprocessor systems ». January 1994.
- [CHA 95] S. Chatterjee and J. Strosnider. « Distributed pipeline scheduling : end-to-end Analysis of heterogeneous, multi-resource real-time systems ». 1995.

Références

- [CHE 90] H. Chetto, M. Silly, and T. Bouchentouf. « Dynamic Scheduling of Real-time Tasks Under Precedence Constraints ». *Real Time Systems, The International Journal of Time-Critical Computing Systems*, 2(3):181–194, September 1990.
- [CIA 99] CIAME. *Réseaux de terrain*. Edition Hermès, 1999.
- [COT 00] F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri. *Ordonnancement temps réel*. Hermès, 2000.
- [COU 94] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems—Concepts and Design, 2nd Ed.* Addison-Wesley Publishers Ltd., 1994.
- [DAV 09] R.I. Davis and A. Burns. « A Survey of Hard Real-Time Scheduling Algorithms and Schedulability Analysis Techniques for Multiprocessor Systems ». In *University of York Technical Report, YCS-2009-443*, 2009.
- [DEC 96] T. Decker. « Three Popular Avionics Databases ». *Real Time Magazine*, (2):29–34, April 1996.
- [DHA 78] S. Dhall and C. Liu. « On a real time scheduling problem ». *Operations Research*, 26:127–140, 1978.

Références

- [DWY 91] J. Dwyer and A. Ioannou. *Les réseaux locaux industriels MAP et TOP*. Editions Masson, mars 1991.
- [GEO 96] L. George, N. Rivierre, and M. Spuri. « Preemptive and Non-Preemptive Real-time Uni-processor Scheduling ». INRIA Technical report number 2966, 1996.
- [HEA 96] D. Head. « MIL-STD-1553B ». *Real Time Magazine*, (2):25–28, April 1996.
- [KOO 95] P. J. Koopman. « Time Division Multiple Access Without a Bus Master ». Technical Report RR-9500470, United Technologies Research Center, June 1995.
- [LEB 95] L. Leboucher and J. B. Stefani. « Admission Control end-to-end Distributed Bindings ». pages 192–208. COST 231, Lectures Notes in Computer Science, Vol 1052, November 1995.
- [LEG 03] J. Legrand, F. Singhoff, L. Nana, L. Marcé, F. Dupont, and H. Hafidi. « About Bounds of Buffers Shared by Periodic Tasks : the IRMA project ». In the 15th Euromicro International Conference of Real Time Systems (WIP Session), Porto, July 2003.

Références

- [LIU 73] C. L. Liu and J. W. Layland. « Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment ». *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.
- [OH 93] Y. Oh and S. H. Son. « Tight Performance Bounds of Heuristics for a Real-Time Scheduling Problem ». Technical Report, May 1993.
- [OH 95] Y. Oh and S.H. Son. « Fixed-priority scheduling of periodic tasks on multiprocessor systems ». 1995.
- [PAL 03] C. Palencia and M. Gonzalez Harbour. « Offset-Based Response Time Analysis of Distributed Systems Scheduled under EDF ». In *15th Euromicro Conference on Real-Time Systems (ECRTS'03)*, 2003.
- [PUJ 95] G. Pujolle. *Les réseaux*. Editions Eyrolles, décembre 1995.
- [RIC 01] P. Richard, F. Cottet, and M. Richard. « On line Scheduling of Real Time Distributed Computers With Complex Communication Constraints ». 7th Int. Conf. on Engineering of Complex Computer Systems, Skovde (Sweden), June 2001.
- [STR 96] H. Strass. « Factory Floor Networks PROFIBUS : the natural choice ». *Real Time Magazine*, (2):6–8, April 1996.

Références

- [TIN 94] K. W. Tindell and J. Clark. « Holistic schedulability analysis for distributed hard real-time systems ». *Microprocessing and Microprogramming*, 40(2-3):117–134, April 1994.
- [UPE 94] P. Uendar and P. J. Koopman. « Communication Protocols for Embedded Systems ». *Embedded Systems Programming*, 7(11):46–58, November 1994.
- [XU 90] J. Xu and D. Parnas. « Scheduling Processes with Release Times, Deadlines, Precedence, and Exclusion Relations ». *IEEE Transactions on Software Engineering*, 16(3):360–369, March 1990.
- [ZEL 96] H. Zeltwanger. « CAN in industrial Applications ». *Real Time Magazine*, (2):20–24, April 1996.

Remerciements

- Certains de ces transparents sont extraits ou inspirés des cours de C. Pagetti et de J. Goossens. Merci à ces auteurs !