Ordonnancement temps réel réparti : étude de la faisabilité

Département informatique

Frank Singhoff

C-208

singhoff@univ-brest.fr

Sommaire

Partie 1

Introduction et rappels

Ordonnancement temps réel (1)

• **Objectifs :** prendre en compte les besoins d'urgence, d'importance et de réactivité dans l'exécution des tâches d'une application temps réel.

• **Elements de taxinomie :**

– Algorithmes hors ligne/en ligne : moment où sont effectués les choix d'allocation.

– Priorités statiques/dynamiques : les priorités changent elles ? Algorithmes statiques/dynamiques.

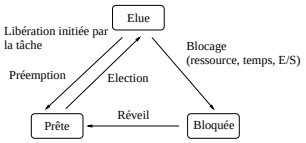
– Algorithmes préemptifs ou non : tâches interruptibles par d'autres ? non préemptif = exclusion mutuelle des ressources aisée, surcoût de l'ordonnanceur moins élevé et efficacité moindre.

Ordonnancement, définitions (2)

- Propriétés recherchées :

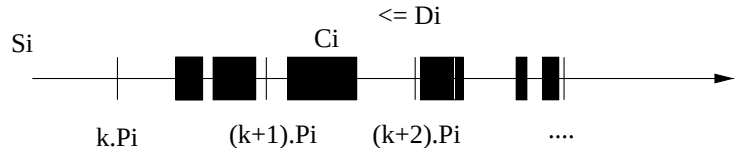
1. **Optimalité :** critère de comparaison des algorithmes (un algorithme est dit optimal s'il est capable de trouver un ordonnancement pour tout ensemble faisable de tâches).
2. **Complexité :** les tests de faisabilité sont ils polynômiaux ?, exponentiels ?, etc
3. **Facilité de mise en œuvre :** l'ordonnanceur est-il facile à implanter ?
4. **Faisabilité :** est il possible d'exhiber un test de faisabilité ?
   - Condition permettant de décider hors ligne du respect des contraintes des tâches.
   - Prédictibilité du temps de réponse des tâches.

Notion de tâche (1)



- **Processeur** = souvent l'unique ressource partageable dans un système temps réel *mono-processeur* $\Longrightarrow$ exécutif temps réel.
- **Tâche :** suite d'instructions + données + contexte d'exécution. Etats de la tâche.
- **Familles de tâches :**
  - Tâches dépendantes ou non. Tâches importantes, urgentes.
  - Tâches répétitives : activations successives (tâches périodiques ou sporadiques) $\Longrightarrow$ **tâches critiques**.
  - Tâches non répétitives/apériodiques : une seule activation $\Longrightarrow$ **tâches non critiques**.

Notion de tâche (2)



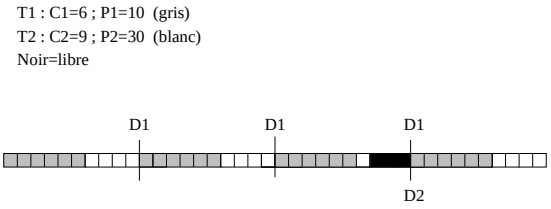
- **Paramètres définissant une tâche i :**
  - Arrivée de la tâche dans le système : S_i.
  - Borne sur le temps d'exécution d'une activation : C_i (capacité).
  - Période d'activation : P_i.
  - Délai critique : D_i (relatif à P_i).
- **Un modèle simplifié de tâches : le modèle de Lui et Layland[LIU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

Exemple : algo. à priorité fixe (1)

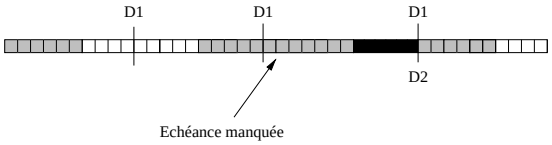
- Priorités fixes $\Longrightarrow$ analyse hors ligne $\Longrightarrow$ applications statiques et critiques.
- Hypothèse pour la suite : modèle de Lui & Layland uniquement.
- **Fonctionnement :**
  1. Affectation des priorités selon l'urgence ou l'importance. Ex : période (Rate Monotonic), délai critique (Deadline Monotonic), criticité.
  2. Phase d'élection : élection de la plus forte priorité.
- **Propriétés :**
  - Complexité faible et mise en œuvre facile.
  - Affectation Rate Monotonic : algorithme optimal dans la classe des algorithmes à priorité fixe.

Exemple : algo. à priorité fixe (2)

- Cas de l'affectation Rate Monotonic avec ordonnanceur préemptif :



- Ordonnanceur non préemptif :



Exemple : algo. à priorité fixe (3)

- Quelques 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) :

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

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

Exemple : algo. à priorité fixe (4)

- 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 :
  - Echec si $w_i^n > P_i$.
  - Réussite si $w_i^{n+1} = w_i^n$.

Exemple : algo. à priorité fixe (5)

- Exemple :
  - Tâche 1 : P1=7 ; C1=3
  - Tâche 2 : P2=12 ; C2=2
  - Tâche 3 : P3=20 ; C3=5

  - $w_1^0 = 3 \implies TR_1 = 3$
  - $w_2^0 = 2$
  - $w_2^1 = 2 + \left\lceil \frac{2}{7} \right\rceil 3 = 5$
  - $w_2^2 = 2 + \left\lceil \frac{5}{7} \right\rceil 3 = 5 \implies TR_2 = 5$
  - $w_3^0 = 5$
  - $w_3^1 = 5 + \left\lceil \frac{5}{7} \right\rceil 3 + \left\lceil \frac{5}{12} \right\rceil 2 = 10$
  - $w_3^2 = 5 + \left\lceil \frac{10}{7} \right\rceil 3 + \left\lceil \frac{10}{12} \right\rceil 2 = 13$
  - $w_3^3 = 5 + \left\lceil \frac{13}{7} \right\rceil 3 + \left\lceil \frac{13}{12} \right\rceil 2 = 15$
  - $w_3^4 = 5 + \left\lceil \frac{15}{7} \right\rceil 3 + \left\lceil \frac{15}{12} \right\rceil 2 = 18$
  - $w_3^5 = 5 + \left\lceil \frac{18}{7} \right\rceil 3 + \left\lceil \frac{18}{12} \right\rceil 2 = 18$
    $\implies TR_3 = 18$

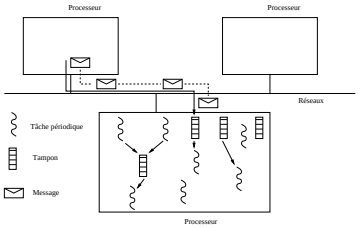
Problèmes étudiés dans ce cours (1)

- "*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. Partage des ressources (données, applications, périphériques chers). Optimisation de leur utilisation.

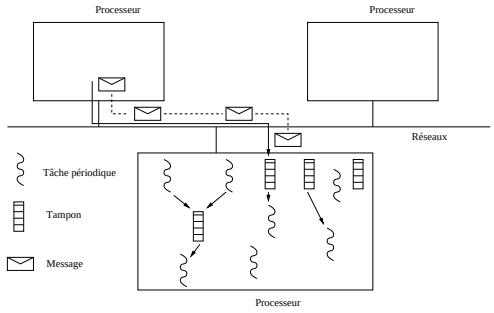
Problèmes étudiés dans ce cours (2)



- Deux types d'ordonnancement temps réel dans un système réparti/multiprocesseurs :

1. **Ordonnancement global :** choisir d'abord la tâche, puis le processeur. Un ordonnanceur global + migration de tâches. Peu utilisé (migration et hétérogénéité).
2. **Ordonnancemant par partitionnement :** placement des tâches et ordonnancement local.

Problèmes étudiés dans ce cours (3)



1. Comment placer les tâches ? (s'il le faut ! !).
2. Comment modéliser les contraintes de précédence entre émetteur et récepteur ?
3. Comment partager le réseau et calculer les temps de communication des messages ?
4. Peut on calculer un pire délai d'exécution entre émetteur et récepteur ?
5. Que dire de la taille des tampons de communications (ex : sockets) ?

$\Longrightarrow$ **Faisabilité d'une application temps réel répartie**

Faisabilité des applications temps réel réparties

1. Placement de tâches.
2. Contraintes de précédence.
3. Ordonnancement de messages.
4. Contraintes de bout en bout.
5. Dimensionnement de tampons.

Partie 2

Placement de tâches

Placement de tâches (1)

- **Heuristiques de placement :**

– Très souvent basées sur RM [OH 93] (tests de faisabilité faciles à exhiber, approche pratique).

– Principales heuristiques d'ordonnancement : Rate-Monotonic-First-Fit (RM-FF) et Rate-Monotonic-Next-Fit (RM-NF) [DHA 78] .

– Autres heuristiques de placement : RM-FFDU[OH 95], RM-ST, RM-GT et RM-GT/M [BUR 94]. RM-BF [OH 93], EDF-FFD (First Fit Decreasing) [BAR 03].

Placement de tâches (2)

- **Fonctionnement général des heuristiques de placement :**

1. Les tâches sont ordonnées selon un paramètre donné (période, charge processeur, ...).
2. Elles sont assignées au processeur courant tant que la charge processeur maximum théorique ne soit pas atteinte. Cette charge dépend du ou des algorithmes d'ordonnancement du système. Si une tâche ne peut être placée sur le processeur courant, un autre processeur est choisi.
3. On termine lorsque toutes les tâches sont attribuées aux processeurs.

Placement de tâches (3)

- **Exemple de Rate Monotonic Next Fit** :

1. Les tâches sont ordonnées dans l'ordre croissant de leur période.
2. On assigne une tâche i à un processeur j si la condition de faisabilité est respectée ; c-a-d $U \leq 0.69$.
3. Dans le cas contraire, on assigne la tâche 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.

Faisabilité des applications temps réel réparties

1. Placement de tâches.
2. Contraintes de précédence.
3. Ordonnancement de messages.
4. Contraintes de bout en bout.
5. Dimensionnement de tampons.

Partie 3

Contraintes de précédence

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

Contraintes de précédence (1)

- Problèmes soulevés :

1. Calculer un ordonnancement qui respecte les contraintes de précédence (simulation, exécution).

2. Décider de la faisabilité hors ligne lorque les tâches sont soumises à des contraintes de précédence.

Contraintes de précédence (2)

- 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 (3)

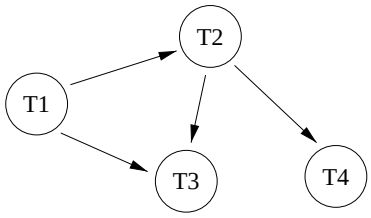
• Principe de la solution de Blazewicz [BLA 76] et de Chetto et al [CHE 90] : rendre les tâches indépendantes en modifiant leurs paramètres.

• Hypothèses : Tâches soit apériodiques, soit périodiques de même période.

• Technique :

1. Modification pour RM :
   - $\forall\, i,j \mid i \prec j \;:\; priorite_i > priorite_j$
2. Modification pour EDF :
   - $D_i^* = min(D_i, min(\forall\, j \mid i \prec j \;:\; D_j^* - C_j))$.

Contraintes de précédence (4)



	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

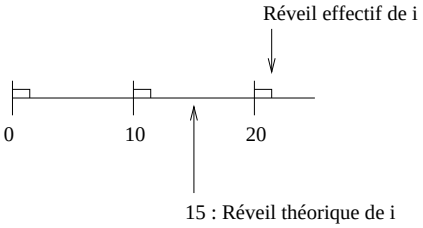
• Exemple : EDF + tâches apériodiques.

- $D_4^* = 14$;
- $D_3^* = 8$;
- $D_2^* = min(D_2, D_3^* - C_3, D_4^* - C_4) = min(10, 8-1, 14-2) = 7$;
- $D_1^* = min(D_1, D_2^* - C_2, D_3^* - C_3) = min(5, 7-2, 8-1) = 5$;

Contraintes de précédence (5)

• Utilisation du **Jitter**. Exemple historique $\Longrightarrow$ le timer d'un système est modélisé comme une tâche périodique avec $P_{timer} = 10\ ms, C_{timer} = 3\ ms$.

• On souhaite réveiller une tâche i à l'instant $t = 15\ ms$.



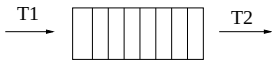
La date effective de réveil de la tâche i sera $23ms$. Sa gigue est de $J_i = 8\ ms$.

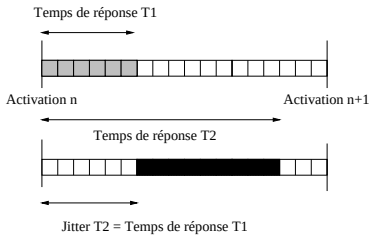
• Temps de réponse $= r_i = w_i + J_i$, avec :

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

Contraintes de précédence (6)

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 doit l'afficher à l'écran.
- T2 doit être activée sur terminaison de T1.
- Quel est le temps de réponse de T2 ?

Faisabilité des applications temps réel réparties

1. Placement de tâches.
2. Contraintes de précédence.
3. Ordonnancement de messages.
4. Contraintes de bout en bout.
5. Dimensionnement de tampons.

Partie 4

Ordonnancement de messages

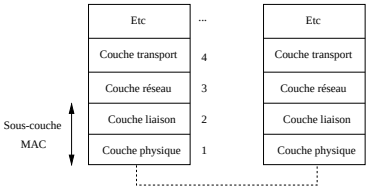
Ordonnancement de messages (1)

- Problèmes soulevés :

1. Comment déterminer le temps de communication d'un message ?
2. Comment les messages doivent ils se partager la ressource réseau tout en respectant leurs contraintes temporelles.

Ordonnancement de messages (2)

- Les contraintes de temps dans le modèle ISO :



- Transport : Expression + négociation des contraintes de temps.
- Réseau : commutation de paquet vs commutation.
- Physique/Liaison/MAC : partage du medium $\Longrightarrow$ protocole d'arbitrage.

- Bus/réseaux temps réel = couches 1/2 + couche applicative de type "transport".

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].

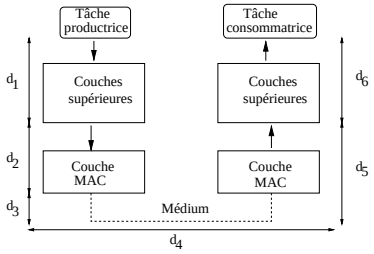
– Domotique : BatiBus[CIA 99].

– Aéronautique/spatial : STD MIL 1553 et ARINC 429 et 629. Aviation civile et militaire, la sonde PathFinder [DEC 96, HEA 96].

– Industrie automobile : bus CAN [ZEL 96].

Ordonnancement de messages (4)

- Temps de communication composé de [COT 00] :



– d_1 et d_6 = délais de traversée des couches ; d_5 = délai de réception. Calcul facile, fixe (borne).
– d_3 = délai de transmission sur le médium (variable, calcul facile $\Longrightarrow$ taille message/débit).
– d_4 = délai de propagation (variable, calcul facile $\Longrightarrow$ taille réseau/vitesse).
– d_2 = **délai d'attente pour l'accès au réseau (variable, dépend du protocole d'arbitrage).**

Ordonnancement de messages (5)

• L'arbitrage définit la méthode d'accès au médium. **Objectif principal : comportement prédictible des temps accès.**

• Eléments de taxinomie des protocoles d'arbitrage :

– Algorithme par coopération ou compétition.

– Arbitrage symétrique ou asymétrique. Arbitrage centralisé/réparti. Notion de maîtres (ou arbitres) et d'esclaves : qui prend l'initiative de la communication ?

– Synchones ou asynchones $\Longrightarrow$ y a t il une horloge globale à tous les coupleurs ?

Ordonnancement de messages (6)

• Principaux protocoles d'arbitrage[UPE 94] :

– **TDMA [KOO 95] :**
  – Une station dédiée, appelée "maître" émet cycliquement une trame de synchronisation.
  – Chaque esclave émet alors, à un instant relatif par rapport à la trame de synchronisation.
  – L'ordre d'émission des esclaves est prédéterminé et fixe $\Longrightarrow$ partage temporel du médium.

– **Protocole à jeton :**
  – Une jeton circule selon un ordre fixe entre les stations (ex : topologie en boucle).
  – Une station qui reçoit le jeton émet ses données, puis, transmet le jeton à la station suivante.

Ordonnancement de messages (7)

- **Polling :**
  - Une station dédiée, dite "maître", émet un message sur le médium, invitant un "esclave" à émettre.
  - En réponse, l'esclave émet ses données. Puis, le maître interroge l'esclave suivant.
  - Le maître dispose d'une liste d'invitations fixe qu'il parcourt de façon séquentielle.
- **CSMA/CA :**
  - 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.

Ordonnancement de messages (8)

• **Le bus CAN :**

- Créée 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-maîtres symétrique.
- Transmission par diffusion.
- Topologie en bus, paires torsadées, généralement longueur maximum 40 m pour un débit de 1MBit/s.
- Composants très fiables à faibles coût (automobile).
- Services de sûreté de fonctionnement très évolués (CRC, acquittement, diagnostique de coupleurs).

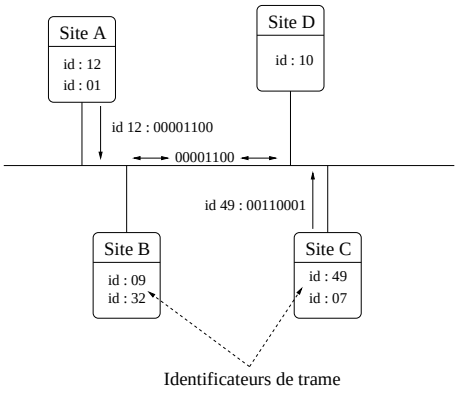
Ordonnancement de messages (9)

• Arbitrage CAN = arbitrage par compétition => CSMA non destructif (CSMA/CA).

• Identificateurs de trame = priorités. Identificateurs uniques et émis par une seule station.

- Lorsque le bus est libre, émission bit à bit de l'identificateur puis 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.
- Emission bit à bit + écoute porteuse = faible débit/taille du réseau.

Ordonnancement de messages (10)



- Les sites A, B, C et D sont des maîtres : émetteurs de trames. Chacun détient une liste d'identificateurs uniques.
- Les sites 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 (entre 0 et 8 octets).

Ordonnancement de messages (11)

- De l'ordonnancement de tâches aux messages :

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

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

Ordonnancement de messages (12)

- Comment calculer le temps de communication d'un message périodique ? $\Longrightarrow$ en appliquant les mêmes méthodes que pour le partage du processeur.

- Exemple dans le cas du bus CAN :

– L'identificateur d'un message CAN = priorité fixe d'une tâche. Affecter l'identificateur des messages selon Rate Monotonic.

– Le protocole d'arbitrage de CAN est un ordonnanceur à priorité fixe en mode *non préemptif : tests de faisabilité non préemptif.* Ex. temps de réponse :

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

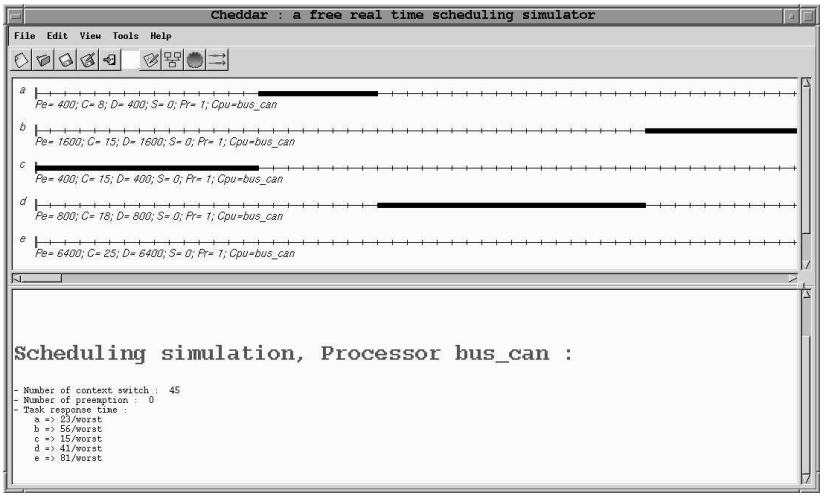
Ordonnancement de messages (13)

- 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 (14)



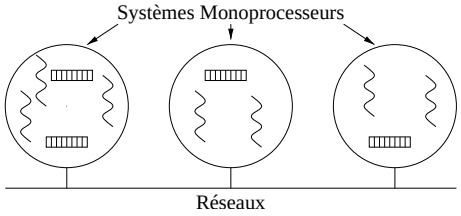
Faisabilité des applications temps réel réparties

1. Placement de tâches.
2. Contraintes de précédence.
3. Ordonnancement de messages.
4. Contraintes de bout en bout.
5. Dimensionnement de tampons.

Partie 5

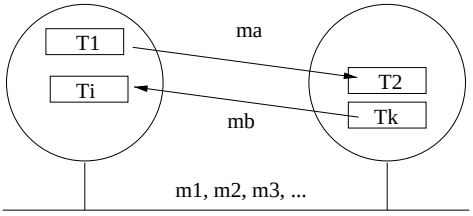
Contraintes de bout en bout

Contraintes de bout en bout



- Approches métiers : concepteur de systèmes ou équipementiers.
- Que cherche-t-on à valider ?
  - Délai/contrainte de bout en bout : ordonnancement de message puis test de la faisabilité de proche en proche (ex : [TIN 94, LEB 95, RIC 01, CHA 95]).
  - Contraintes locales. Test de la faisabilité processeur par processeur.

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    ∀i : J_i := 0, r_i := 0, r'_i := 0;
20    ∀i : Calculer_temps_de_réponse (r_i) ;
30    Tant que (∃i : r_i ≠ r'_i) {
40        ∀i : J_i := max(J_i, ∀j avec j ≺ i : r_j);
50        ∀i : r'_i := r_i;
60        ∀i : Calculer_temps_de_réponse (r_i) ;
70    }
```

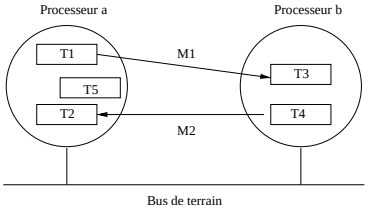
Si i est une tâche :

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

Si i est un message dont le temps de communication de tâche à tâche est borné par M_i :

$$r_i = J_i + M_i$$

Temps de réponse bout en bout (3)



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 (4)

- **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 (5)

- **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

Faisabilité des applications temps réel réparties

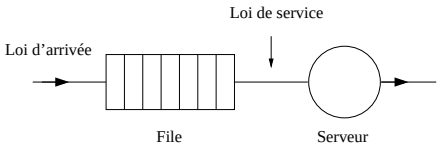
1. Placement de tâches.
2. Contraintes de précédence.
3. Ordonnancement de messages.
4. Contraintes de bout en bout.
5. Dimensionnement de tampons.

Partie 6

Dimensionner les tampons de communications

- Afin de :
  1. Déterminer la taille des tampons afin d'éviter un éventuel débordement
  2. **Déterminer le pire délai de mémorisation d'un message $\Longrightarrow$ calcul holistique.**

Dimensionnement de tampon (1)



• **Objectif :** description loi arrivée λ et loi service μ pour obtenir des critères de performance (L=nombre de message, W=délai de mémorisation d'un message). Little : $L = \lambda.W$.

• Notation de kendall : X/Y/n :
- X : loi d'arrivée des clients (M,G,D).
- Y : 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)

• **File d'attente P/P/1[LEG 03] :**
- n producteurs et 1 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 :**

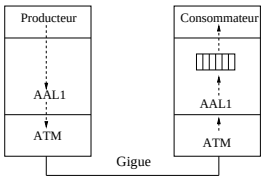
Pour un jeu de tâches harmoniques :

$$L_{max} = 2.n$$
$$W_{max} = 2.n.P_{cons}$$

Et dans le cas contraire :

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

Dimensionnement de tampon (3)



• La taille maximum des tampons régulant le trafic de la couche de transport à débit fixe d'ATM est de :

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

où $W_{max} = 2.gigue.$ est **le plus grand délai d'accumulation des messages dans le tampon et** $gigue = delai_{max} - delai_{min}$.

• Illustration de la loi de Little ($\lambda_{max} = 1/d$) :

$$L_{max} = \lambda_{max} W_{max}$$

Dimensionnement de tampon (4)

• Base de la démonstration :

1. **Délai de mémorisation :**

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

y messages déja 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 sur y. Obtention de W_{max} par application de Little.

Partie 7

Résumé

Résumé

- Difficultées supplémentaires liées à la répartition :
  - Un exécutif, n'est plus suffisant : processeur + réseaux + mémoire à gérer globalement.
  - Placement de tâches.
  - Ordonnancement de messages.
  - Dimensionnement de tampons de communications.
  - Contraintes temporelles de bout en bout.
- Maturité du domaine pas encore atteinte :
  - Peu de résultats théoriques qui se soient imposés.
  - Déterminisme/précision nécessairement moins bon à cause des communications.
  - L'expérience prime encore...

Partie 8

Acronymes

- **MAP**. Manufactoring 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.

Partie 9

Références

[BAR 03] S. K. Baruah and S. Funk. « Task assignment in heterogeneous multiprocessor platforms ». Technical Report, University of North Carolina, 2003.

[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, Noth-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.

[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.

[DEC 96] T. Decker. « Three Popular Avionics Databuses ». *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.

[DWY 91] J. Dwyer and A. Ioannou. *Les réseaux locaux industriels MAP et TOP.* Editions Masson, mars 1991.

[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 Techologies 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.

[LIU 73] C. L. Liu and J. W. Layland. « Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environnment ». *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 P roblem ». Technical Report, May 1993.

[OH 95] Y. Oh and S.H. Son. « Fixed-priority scheduling of periodic tasks on multiprocessor systems ». 1995.

[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.

[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. Upendar 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.