Affectation des Priorités dans les Systèmes Temps Réel Distribués

Michael Richard, Pascal Richard, Francis Cottet

Affectation des Priorités dans les Systèmes Temps Réel Distribués

Michaël RICHARD, Pascal RICHARD et Francis COTTET

Laboratoire d'Informatique Scientifique et Industrielle
Ecole Nationale de Mécanique et d'Aérotechnique
Téléport 2 - BP 109 F-86960 Futuroscope, France
{richardm,richardp,cottet}@ensma.fr,

MOVEP'2k, Nantes, 19-23 juin 2000

Superviseur(s) Pascal RICHARD, Francis COTTET
Keywords : Temps réel, systèmes distribués, Analyse Holistique, Affectation de priorités

Résumé Toute tâche ordonnancée dans un système informatique temps réel doit respecter ses contraintes temporelles (échéances, périodicité). Dans un système distribué, les ordonnancements des tâches et messages échangés sur le réseau sont fortement dépendants. Nous étudions une méthode arborescente pour affecter les priorités aux tâches et aux messages afin d'obtenir un système ordonnançable. Le test d'ordonnançabilité repose sur le calcul des pires temps de réponses des tâches et des messages (analyse holistique).

1 Ordonnancement des systèmes temps réel distribués

Nous étudions des systèmes temps réel distribués, composés de calculateurs communiquant à l'aide de messages via un réseau, sans mémoire partagée. Les applications temps réel possèdent des contraintes temporelles strictes. La conception d'un tel système nécessite donc une validation temporelle de son comportement, afin de vérifier le respect des échéances. Chaque tâche τ_i est représentée par un triplet (C_i, D_i, T_i) où C_i est le pire temps d'exécution, D_i l'échéance relative au réveil de la tâche et T_i la période des activations de τ_i.

Nous supposons que l'affectation des tâches sur les processeurs est connue initialement et qu'elle n'évolue pas dans le temps. Cette hypothèse n'est pas restrictive puisque les tâches s'exécutent généralement sur des processeurs dédiés. Le réseau est vu comme une ressource critique partagée par toutes les tâches. La transmission des messages peut alors engendrer des problèmes *"d'inversion de priorité"* [But97]. En conséquence, seule une analyse du *"pire cas"* peut être effectuée pour valider l'application. Elle fournit une condition suffisante d'ordonnançabilité.

Afin d'illustrer le type d'application à valider, nous décrivons figure 1 l'architecture informatique embarquée pour la gestion d'une automobile. Le système

regroupe un ensemble d'ECU – Electronic Component Units – qui assurent les différentes fonctions. Ces unités ECU sont liées par deux réseaux : le réseau CAN, dédié aux fonctions critiques de l'automobile, telle que l'ABS et le contrôle moteur et le réseau VAN utilisé pour relier les ECU assurant des fonctions non-critiques (possédant des contraintes temporelles plus lâches) comme l'affichage sur la console et la gestion de la climatisation. L'ECU 6 de ce réseau est une passerelle entre le réseau CAN et le réseau VAN. Cette application est composée

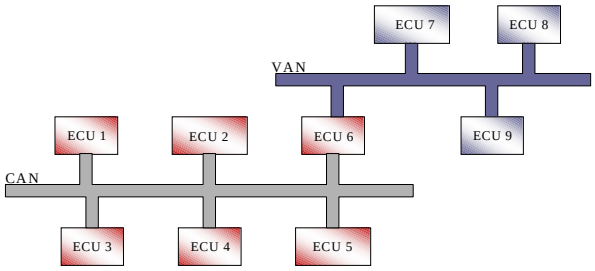

FIG. 1 – Structure d'un système embarqué dans l'automobile

de 44 tâches s'exécutant sur les différents ECU et 19 messages circulant sur les deux réseaux. Le choix des priorités des tâches et des messages est prépondérant pour ne pas surdimensionner le système. Des méthodes heuristiques ont été proposées dans la littérature [GH95,TBW92]. Dans la suite nous proposons une nouvelle approche pour résoudre ce problème. Notre objectif est d'utiliser le plus efficacement possible les ressources du système.

Nous présentons dans la section 2 une méthode de validation qui considère les priorités des tâches et des messages comme des paramètres. Nous décrivons enfin les principes d'une méthode d'affectation des priorités.

2 L'Analyse Holistique des systèmes à priorités fixes

L'analyse holistique permet de calculer le pire temps de réponse des tâches et des messages [Tin94]. L'intérêt de cette méthode réside dans la prise en compte de la dépendance entre les ordonnancements des tâches et des messages. Ceci permet d'élaborer une analyse plus réaliste et plus fine du pire cas pouvant survenir dans la vie du système. Avec cette approche, les messages sont considérés comme des tâches s'exécutant sur un processeur fictif : le réseau. Les contraintes de communication sont alors considérées comme des relations de précédence entre tâches.

Les décalages sur l'activation des tâches, dus à l'attente des messages sont modélisés grâce à la notion de gigue temporelle sur l'activation des tâches (notée J_i). L'analyse holistique définit les temps de réponse des tâches et des messages

en fonction de ces gigues. Au cours des itérations de la méthode, elles sont affinées jusqu'à l'obtention d'un point fixe. Soient *ResponseTime* la fonction calculant le pire temps de réponse d'une tâche τ_i et n le nombre de tâches (messages inclus), le système d'équations à résoudre est le suivant :

$$1 \leq i \leq n \quad \begin{cases} R_i^{(0)} = C_i \\ J_i^{(0)} = 0 \\ R_i^{(k)} = ResponseTime\left(J_i^{k-1)}\right) \\ J_i^{(k)} = \max\limits_{j \in prec(i)} \left(R_i^{(k)}\right) \end{cases} \tag{1}$$

Le point fixe est atteint lorsque pour k $\in \mathbb{N}$ on vérifie :

$$R_i^* = R_i^{(k)} = R_i^{(k-1)}, 1 \leq \ i \leq n \tag{2}$$

Le calcul du temps de réponse d'une tâche et d'un message repose sur des résultats classiques de la littérature [LL73,GRS96,Tin94]. Plus précisément, le temps de réponse d'une tâche τ_i est basée sur la plus grande période de temps où le processeur est pleinement occupé par des tâches de priorité supérieure ou égale à celle de τ_i (*"i-level Busy Period"*)[Leh90]. La charge processeur d'une tâche τ_i est le rapport $\frac{C_i}{T_i}$. La longueur de cette période occupée est alors la charge processeur engendrée par les tâches lorsque celles-ci sont activées simultanément lors de la réception de leurs premiers messages et les exécutions suivantes sont exécutées sans attente [Tin94]. Ainsi une période occupée de longueur t comportant q exécutions de τ_i, est terminée si et seulement si : $t < (q+1)T_i$ (i.e. il n'existe plus de tâche à exécuter de priorité supérieure ou égale à i). Calculer le plus grand temps de réponse R_i revient à examiner les Q exécutions de τ_i dans la période occupée de longueur t. Le temps de réponse R_i de la tâche τ_i est alors donné par :

$$R_i = \max_{q=1,2,\ldots,Q} (t + J_i - qT_i) \tag{3}$$

La longueur de la période occupée se définit comme le point fixe de l'équation (4). Le point fixe est obtenu lorsque $t = t^{(k+1)} = t^{(k)}$.

$$t^{(k+1)} = (q+1)C_i + \sum_{j=1}^{i-1} \left\lceil \frac{J_j + t^{(k)}}{T_j} \right\rceil C_j \tag{4}$$

Nous avons expérimenté cette méthode sur l'application présentée dans l'introduction. Bien que l'analyse holistique soit exponentielle en temps, la validation ne nécessite pas plus d'une seconde de traitement sur un micro-ordinateur. On pourra se reporter à [RRSC00] pour les détails de cette expérimentation.

3 Travaux en cours

La méthode holistique permet de valider l'application en supposant que les priorités des tâches et des messages sont connues. Comme nous l'avons précisé dans l'introduction, le choix des priorités est déterminant pour utiliser au mieux les ressources du système. L'objectif est donc de déterminer s'il existe une affectation des priorités des tâches et de messages pour laquelle l'application est ordonnançable.

Les méthodes présentées dans [GH95,TBW92] ne parcourent pas toutes les priorités possibles, mais seulement un sous-ensemble. Nous proposons une méthode optimale, vis à vis de l'analyse holistique. Précisément, s'il existe un ensemble de priorités tel que l'application soit validée par une analyse holistique, alors notre méthode doit le trouver.

L'énumération de toutes les priorités possibles est mémorisée dans un arbre de recherche. Chaque nœud de l'arbre représente l'affectation d'une priorité à une tâche. Afin d'éviter les inévitables branches redondantes, l'affectation est réalisée processeur par processeur (comme le propose la méthode de [BFR75] pour un problème de placement et d'ordonnancement d'atelier de production). La figure 2 présente la structure générale de l'arbre de recherche. Chaque branche complète de cet arbre représente un ensemble de priorités pour les tâches et les messages. L'analyse holistique permet alors de valider cet ensemble. Dans le cas contraire, la recherche continue avec un retour arrière (*backtracking*), pour explorer une autre partie de l'espace des solutions. Notons qu'au même titre que les tâches sur les différents sites, les messages sont énumérés réseau par réseau, ces derniers étant vu comme des processeurs (c.f. analyse holistique).

Pour améliorer l'efficacité de la recherche, c'est-à-dire pour augmenter les chances de trouver une solution rapidement, nous pensons que les paramètres importants sont :

- l'ordre de parcours des sites :
  Les sites sont classés selon leur coefficient de charge. Deux cas peuvent alors être envisagés suivant que la charge des sites est croissante ou non. Dans le premier cas, le site ayant le coefficient de charge le plus grand se trouvera en bas de l'arbre. Or comme l'on parcours plus souvent le site de niveau (i) que celui de niveau $(i-k)$, la chance de trouver une solution rapidement est donc plus grande. Dans le second cas, le site ayant le coefficient de charge le plus grand se trouvera en première position dans l'arbre. Puisque sa charge est plus grande, ce site sera plus "difficile à ordonnancer" et la chance de trouver une bonne séquence sera plus faible. De nombreuses coupes pourront alors être faites, diminuant ainsi fortement l'espace des solutions à parcourir.
- ordre d'affectation des priorités aux tâches et messages :
  De la même manière, une relation d'ordre est mise en place à l'intérieur de chaque site (processeurs et réseaux). Les tâches de chaque site sont, dans un premier temps, classées selon la politique d'ordonnancement "Deadline

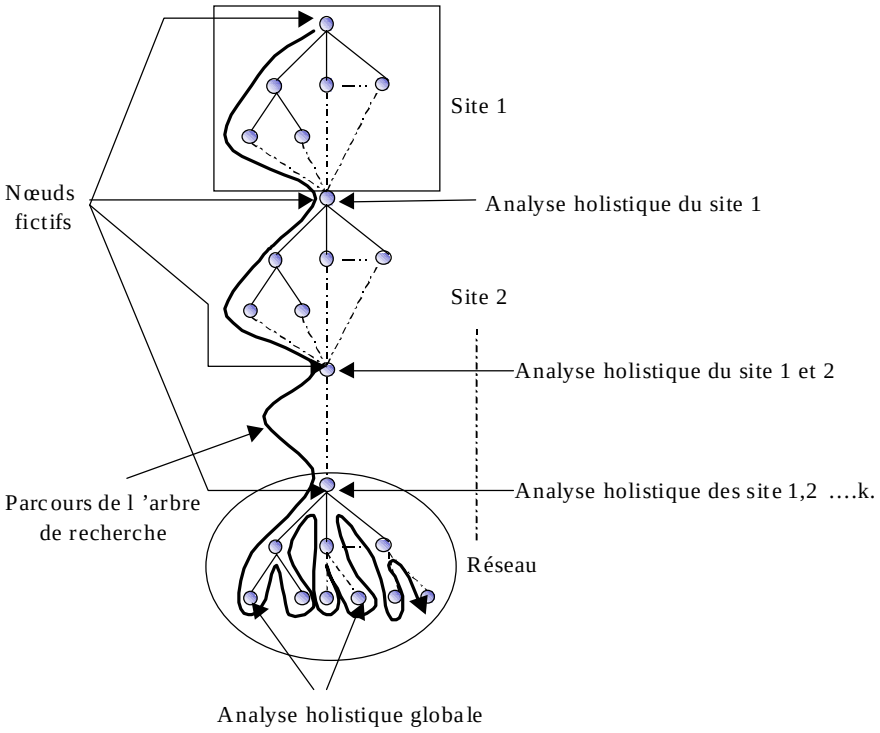

FIG. 2 – Structure de l'arbre de recherche

Monotonic" qui donne la plus forte priorité à la tâche possédant la plus petite échéance relative ($D_i < D_j \Rightarrow Prio_i > Prio_j$).

À chaque création d'un nœud, il faut pouvoir décider si la tâche correspondante est ordonnançable avec le niveau de priorité assigné. Pour cela, le système d'équations (3,4) est résolu. Deux cas se présentent :

- Si la tâche est ordonnançable, on continue la progression dans la branche de l'arbre,
- sinon, on effectue un retour arrière dans l'arbre de recherche.

Lorsque toutes les tâches d'un site possèdent une priorité, on effectue une analyse holistique sur l'ensemble des sites déjà traités. En fonction du résultat de cette analyse, on poursuit la recherche ou bien on effectue un retour arrière. Enfin lorsque toutes les tâches et tous les messages possèdent une priorité, une analyse holistique globale est réalisée.

Cette procédure de recherche optimale des priorités des tâches et des messages peut également être enrichie pour traiter des problèmes plus généraux. Voici quelques idées que nous comptons développer dans le futur :

- comment prendre en compte la *tolérance aux fautes* matérielles,

– optimiser des critères de décisions comme la *qualité de service*,
– comment prendre en comptes des tâches apériodiques.

Cette méthode est en cours de mise en œuvre actuellement.

4 Conclusion

Nous avons présenté une méthode d'affectation des priorités pour les systèmes temps réel distribués. L'intérêt de ce travail est d'améliorer le taux d'utilisation des ressources du système. Ceci est notamment très important pour les systèmes embarqués comme l'illustre le problème présenté dans l'introduction.
Nous pensons de plus mener une expérimentation numérique comparative de notre méthode avec l'heuristique présentée dans [GH95]. Ceci permettra de fixer les limites de ces deux approches en fonction des types de configuration de tâches.

Références

[BFR75] P. Bratley, M. Florian, and P. Robillard. Scheduling with earliest start and due date constraints on multiple machines. *Naval Research Logistic Quaterly*, 22(1) :165–173, 1975.

[But97] G.C. Buttazzo. *Hard real-time computing systems : predictable algorithms and applications*. Kluwer Academic Publishers, 1997.

[GH95] J.J.G. García and M.G. Harbour. Optimized priority assignment for tasks and messages in distributed hard real-time systems. *Proc 3^d Workshop on Parallel and Distributed Hard Real-Time Systems, Santa Barbara*, pages 124–132, 1995.

[GRS96] L. George, N. Rivierre, and M. Spuri. Preemptive and non-preemptive real-time uniprocessor scheduling. Technical Report 2966, INRIA Research Report, 1996.

[Leh90] J.P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. *proc : 11^{th} IEEE Real-Time Systems Symposium*, pages 201–209, 1990.

[LL73] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1), 1973.

[RRSC00] M. Richard, P. Richard, Y. Song, and F. Cottet. Validation of multiple field-bus architecture in automotive systems based on holistic analysis. *Soumis à IEEE WFCS'2000, Porto*, 2000.

[TBW92] K.W. Tindell, A. Burns, and A.J. Wellings. Allocating real-time tasks. an np-hard problem made easy. *Real-Time Systems Journal*, 4(2), 1992.

[Tin94] K.W. Tindell. *Fixed Priority scheduling in Distributed Real-Time Systems*. PhD thesis, University of York, 1994.