Rapport Interne No: 0003-2000

Analyse Holistique des Systèmes Temps Réels Distribués

M. Richard {richardm@ensma.fr} P. Richard {richardp@ensma.fr}
F. Cottet {cottet@ensma.fr}

Laboratoire d'Informatique Scientifique et Industrielle
École Nationale Supérieure de Mécanique et d'Aérotechnique
Futuroscope

22 novembre 2000

Table des matières

Chapitre 1

Ordonnancement des systèmes distribués temps réel

1.1 Problématique

La conception d'un système informatique temps réel nécessite une validation temporelle de son comportement afin de vérifier que les tâches respectent leurs échéances. Les algorithmes d'ordonnancement des systèmes d'exploitation temps réel sont généralement en-ligne, et fondée sur des priorités fixes (Rate Monotonic, Deadline Monotonic, ...) ou dynamiques selon l'échéance la plus proche (Earliest Deadline). A chaque instant la tâche prête la plus prioritaire s'exécute sur le processeur considéré. Les activations des tâches sont périodiques, et la durée de vie du système est a priori infinie.

La validation mathématique d'une configuration de tâches dépend de l'application visée : système embarqué, réseau, multimédia... Deux modèles de base permettent de représenter les problèmes canoniques. Dans le premier, le système de tâche est dirigé par le temps. Soit C_i la durée d'exécution d'une tâche τ_i dans le pire cas, T_i sa période, alors les dates d'activations sont $k.T_i$, avec k entier [But97, SSRB98]. Dans le second modèle, l'activation des tâches suit les événements périodiques externes, comme l'acquisition de signaux, ou interne comme la réception de message. Ce modèle, plus général que le premier, est notamment utilisé dans l'analyse RMA (Rate Monotonic Analysis) [KRP$^+$93]. Le temps de réponse d'une tâche est la différence entre la fin de son exécution et sa date d'activation. Dans les deux cas, un ordonnancement sera valide si le temps de réponse de toutes les instances de toutes les tâches est inférieur à l'échéance D_i de la tâches τ_i, relative à son activation. La faisabilité d'un ordonnancement est un problème NP-Difficile, même lorsque les tâches sont indépendantes et s'exécutent sur un processeur unique [LW82, BRH90].

Lorsque le système est distribué, un réseau informatique est l'unique moyen de communication entre les processeurs et constitue ainsi une ressource partagée par les tâches. Plusieurs niveaux de problèmes sont à considérer en fonction des caractéristiques du système distribué. Par exemple le placement des tâches sur le processeur, les éventuelles migrations des tâches durant leurs exécutions d'un processeur vers un autre, les protocoles de synchronisation des tâches [Sun97], les protocoles d'accès aux ressources [But97], l'ordonnancement des tâches sur les processeurs et des messages sur le réseau sont autant de paramètres bouleversant les performances du système. La validation d'un système temps réel distribué sera donc un compromis entre le temps de calcul à dépenser et la qualité de service pouvant être restituée par le système. Dans la suite nous considérons uniquement les problèmes liés à l'ordonnancement des tâches et des messages. Nous supposons que le placement des tâches est connu dès la conception du système, ce qui est en général le cas puisque de nombreux processeurs sont dédiés, interdisant tout autre placement ainsi que la migration des tâches. Le lecteur intéressé par ces problèmes pourra se reporter à [CDKM00].

1.2 Ordonnancement conjoint des tâches et des messages

En mono-processeur, lorsque les tâches contiennent des sections critiques, une tâche prioritaire peut être bloquée par une tâche moins prioritaire détenant une ressource. La transmission des messages sur un réseau, en environnement distribué, va provoquer le même phénomène d'inversion de priorité puisque la transmission d'un message quelconque ne peut pas être interrompue par l'émission d'un message prioritaire. De plus, seuls les pires temps d'exécution des tâches sont connus, et non leurs durées exactes. L'exécution plus rapide d'une tâche peut lui permettre d'émettre plus tôt son message, bloquant le message d'une tâche prioritaire. Ce problème est illustré figure 1.1, où la tâche τ_1 émet un message m_1 pour la tâche τ_3, et τ_2 émet un message m_2 pour la tâche τ_4. Dans le cas (a), les tâches s'exécutent avec leurs pires temps d'exécution, et chaque tâche respecte son échéance. Dans le cas (b), la tâche τ_2 s'exécute plus vite entraînant l'émission de m_2 avant m_1 sur le réseau. La tâche τ_3 manque alors son échéance (représentée par une flèche descendante).

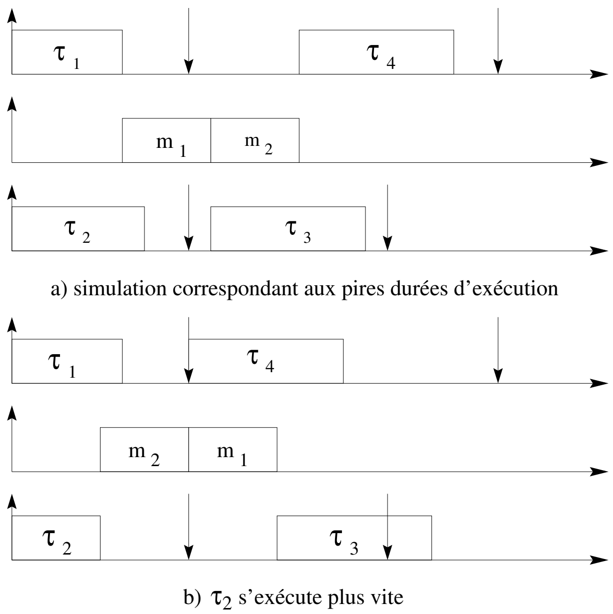

FIG. 1.1 – Simulation et validation d'une configuration de tâches

L'analyse d'ordonnançabilité d'une configuration de tâches consiste à vérifier que les échéances sont toutes respectées sachant que le protocole de synchronisation des tâches, les protocoles d'accès aux ressources critiques, et la politique d'ordonnancement des processeurs sont des paramètres fixés a priori. En raison du phénomène d'inversion de priorité et de la connaissance d'une borne supérieure de la durée d'exécution des tâches, seule une analyse du pire cas peut être faite. Ceci conduit à une condition suffisante d'ordonnançabilité. C'est-à-dire, si le résultat de l'analyse indique que le système est non ordonnançable, cela ne veut pas dire que celui-ci ne le sera pas durant son exécution sur la plate-forme réelle. Par contre, si l'analyse indique que le système est ordonnançable, il le sera quel que soit son comportement à l'exécution (i.e. sans faire d'hypothèse sur les durées d'exécution des tâches).

Nous considérons dans la suite un système distribué constitué d'un ensemble de stations reliées par un réseau informatique. Chaque station est dotée d'un processeur unique exécutant un système d'exploitation temps réel. Nous établissons des hypothèses simplificatrices suivantes :
- existence d'une horloge globale (i.e. les horloges des stations sont parfaitement synchronisées)
- le réseau est fiable, les communications sont de durées bornées et sans erreur de transmission.
- aucune ressource n'est partagée entre les processeurs autre que le réseau. La préemption est autorisée sur chaque processeur,

- les tâches ne sont pas ré-entrantes, et les instances des tâches restent ordonnées en suivant leurs dates d'activation.
- les tâches sont à départ simultané au démarrage du système.
- les tâches, et leurs priorités fixes éventuelles, sont affectées aux processeurs à la conception du système.
- les tâches ne peuvent pas migrer dans le système.
- les messages sont lus au début de l'exécution des tâches.
- les messages sont émis à la fin d'exécution des tâches.

Dans de nombreux cas certaines de ces hypothèses peuvent être supprimées, mais la méthode d'analyse devient alors plus complexe. Nous les avons conservées dans un souci de clarté. Nous présentons dans la suite les concepts de base sur lesquels reposent l'analyse holistique, analyse permettant de valider analytiquement un système distribué. Nous la décrivons dans les principes et donnons en annexe les algorithmes détaillés nécessaires à la mise en œuvre complète de l'analyse holistique, ce que nous n'avons jamais trouvés dans la littérature présentée en bibliographie. La section 3.4 décrit un exemple montrant l'intérêt pratique et la souplesse de la méthode. Nous donnons ci-après les notations utilisées dans la suite de l'article.

Notations :

t_i identificateur de tâche

C_i pire temps de réponse de τ_i pour chaque activation

D_i échéance de τ_i, mesurée relativement à la date d'activation de la tâche

T_i période d'activation de τ_i

J_i le pire temps de gigue sur l'activation de τ_i (différence entre l'activation et le réveil de la tâche)

$Prio_i$ priorité de la tâche τ_i

B_i facteur de blocage de τ_i, durée maximum d'attente avant d'obtenir une ressource

TR_i pire temps de réponse de τ_i, durée entre l'activation d'une tâche et sa fin d'exécution

Chapitre 2

Détermination et analyse du pire cas

La construction d'un ordonnancement ne permet pas de valider les applications temps réel distribuées. La solution est de caractériser le ou les pire(s) scénario(s) possible(s). L'activité du processeur lié à ces scénarios est étudiée afin de calculer le temps de réponse des tâches. L'esprit est de minimiser la période d'étude.

2.1 Concepts de base

La notion de *période d'activité* (Busy Period) permet de caractériser les ordonnancements à priorités fixes et dynamiques en identifiant les périodes d'activités du processeur. L'idée principale de cette méthode est de déterminer la période d'activité engendrant le pire temps de réponse d'une tâche.

Une période d'activité est un intervalle de temps sans *temps creux* durant lequel le processeur est donc toujours occupé. Tout au long de l'exécution d'un système, il existe une infinité de période d'activité. En introduisant ci-dessous la notion d'*instant critique* une période d'activité particulière peut alors être définie.

Définition 1 *[LL73] Un instant critique est défini par une date t à laquelle toutes les tâches sont activées simultanément.*

Ainsi la première période d'activité obtenue lors d'un réveil synchrone des tâches à la date $t = 0$ (i.e. la date $t = 0$ est un instant critique) porte le nom de période d'activité synchrone et possède des propriétés intéressantes.

Définition 2 *[JP86, Leh90, KLS93] Une période d'activité synchrone (Synchronous Busy Period) est définie comme un intervalle de temps $[0, L)$ délimité par un instant critique à la date $t = 0$ et un temps creux à la date L. L est appelé la longueur de la période d'activité synchrone.*

L'étude et la détermination de la longueur de cette période d'activité synchrone constitue la base de la méthode de calcul des pires temps de réponse. Nous donnons ci-dessous la définition de la *charge* pour une configuration τ de n tâches. Précisément, la charge induite par les n tâches de la configuration à l'instant t est définie par :

Définition 3 *[BMR90] Soit un instant critique ayant lieu à la date $t = 0$, la charge, notée $W(t)$, est la durée d'exécution des n tâches sur l'intervalle $[0, t[$:*

$$W(t) = \sum_{i=0}^{n-1} \left\lceil \frac{t}{T_i} \right\rceil C_i \tag{2.1}$$

La longueur L d'une période d'activité synchrone peut-être calculée à l'aide d'une suite récurrente. Considérant un intervalle $[0, t[$, l'idée est de comparée la charge induite $W(t)$ avec la longueur t de l'intervalle. Si la charge est supérieure à la longueur de l'intervalle, la charge calculée à la date t est donc une borne inférieure de la longueur L de la période d'activité synchrone. Le calcul est alors appliqué récursivement en considérant les intervalles $[0, W(t)[$, $[0, W(W(t))[$, ... aux charges induites $W(W(t))$, $W(W(W(t)))$, ... et ce jusqu'à ce que la nouvelle valeur calculée soit égale à la précédente. Cette dernière valeur est la longueur L de la période d'activité synchrone. De manière plus formelle, L peut-être définie comme le point fixe du système récurrent suivant :

$$\left\{ \begin{array}{rcl} L^{(0)} & = & \sum_{i=0}^{n-1} C_i \\ L^{(m+1)} & = & W(L^{(m)}) \end{array} \right. \tag{2.2}$$

Le calcul est arrêté lorsque deux valeurs consécutives sont égales : $L = L^{(m)} = L^{(m+1)}$.
La définition suivante, introduisant le facteur d'utilisation processeur, permet d'énoncer la propriété 1 prouvant la convergence de l'équation (2.2).

Définition 4 *[LL73] Le facteur d'utilisation processeur U est la fraction de temps d'utilisation du processeur nécessaire à l'exécution de la configuration de tâche :*

$$U = \sum_{i=0}^{n-1} \frac{C_i}{T_i} \tag{2.3}$$

Une condition nécessaire d'ordonnançabilité d'une configuration de tâches découlant de cette définition est : $U \leq 1$

Propriété 1 *si $U \leq 1$, l'équation 2.2 converge en un nombre fini de pas.*

La propriété 2 ne dépend pas de la politique d'ordonnancement utilisée, mais uniquement de la loi d'arrivée des tâches. La seule hypothèse faite est que la politique d'ordonnancement soit une politique *au-plus-tôt* (i.e. conservatif).

Propriété 2 *La plus longue période d'activité est la période d'activité synchrone de longueur L.*

Considérons l'exemple d'une configuration τ de quatre tâches possédant les paramètres temporelles présentés dans le tableau 2.1. La figure 2.1 représente la courbe de la fonction charge $W(t)$ définie

	C_i	T_i	U_i
τ_1	5	20	0.25
τ_2	2	8	0.25
τ_3	1	7	0.14
τ_4	3	10	0.3
			$U = 0.94$

TAB. 2.1 – Configuration τ

précédemment. La droite $y = x$ symbolise la capacité maximale d'exécution du processeur. Les deux parties hachurées représentent deux périodes occupées, séparées par un temps creux, dont la période d'activité synchrone (notons que de manière cohérente avec la propriété 2 cette dernière est plus longue que la suivante). Cette représentation de la fonction $W(t)$ permet également d'illustrer la propriété 1.

2.2 Définition du temps de réponse

Nous étudions dans cette section le *pire temps de réponse* d'une tâche. Il existe deux modèles différents permettant de représenter une configuration : le premier est un modèle dit *périodique* inspiré du modèle de Liu et Layland [LL73], tandis que le second est un modèle plus général dit *événementiel* plus particulièrement destiné aux applications de *"contrôle-commande"*.

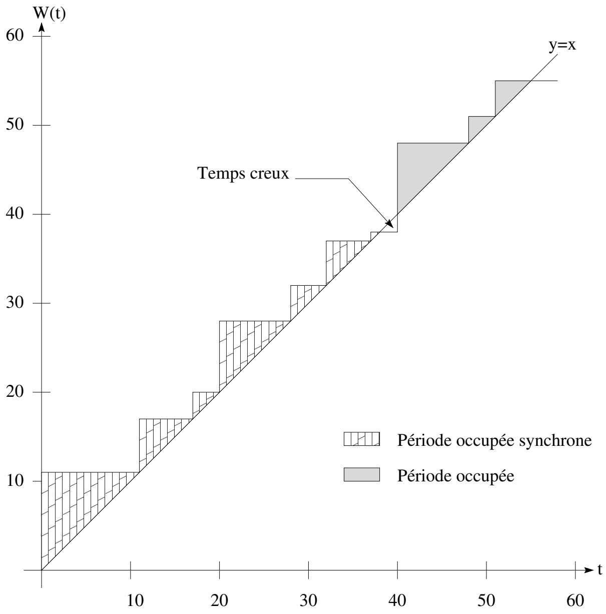

FIG. 2.1 – Charge et période d'activité

2.2.1 Modèle périodique

Le modèle périodique est basé sur la notion de tâche. Une tâche est représentée par un triplet (C_i, D_i, T_i). Une tâche périodique est composée d'instance. Le temps de réponse d'une instance de la tâche τ_i est alors défini ainsi :

Définition 5 *Le temps de réponse d'une instance de la tâche τ_i est défini comme la durée entre la date d'activation de l'instance et sa date de fin d'exécution (Cf. figure 2.2).*

Le *pire temps de réponse* (Worst Case Response Time) d'une tâche τ_i est alors défini ainsi :

Définition 6 *Le pire temps de réponse d'une tâche τ_i, noté TR_i est le plus grand temps de réponse parmi toutes les instances de τ_i.*

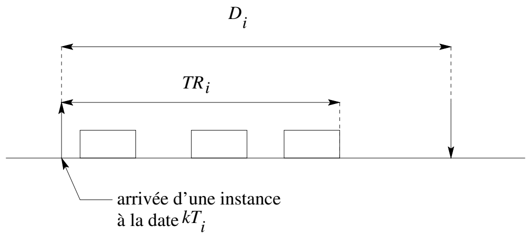

FIG. 2.2 – Temps de réponse dans le modèle périodique

2.2.2 modèle événementiel

Ce modèle, basée sur les évènements, est celui utilisé dans la méthode RMA [KRP+93]. Dans ce modèle les tâches sont déclenchées en réponse à des évènements. Il existe plusieurs types d'évènements (caractérisés par leur loi d'arrivée) et les tâches peuvent être définie de manière assez fine. Le lecteur intéressé pourra se reporter à [KRP+93] pour une description détaillée.

Le modèle périodique vu précèdement est un cas particulier du modèle événementiel. En effet, dans le premier, le caractère périodique des tâches implique une connaissance exacte de la date d'activation des différentes instances de cette tâche. A contrario, dans le modèle événementiel, la date d'arrivée de l'évènement n'est pas obligatoirement connue. La figure 2.3 permet de définir le temps de réponse dans ce modèle événementiel.

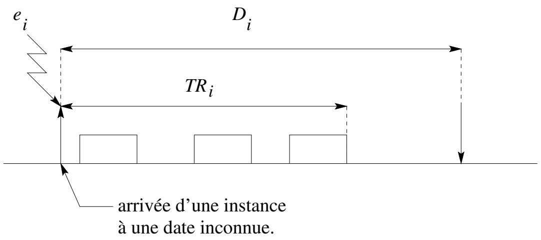

FIG. 2.3 – Temps de réponse du modèle événementiel

Bien que les résultats connus sur le modèle périodique tiennent, cette généralisation du modèle rend le calcul du pire temps de réponse un peu plus délicat. Il faut en effet calculer une borne inférieure de la date d'arrivée de l'évènement déclencheur en plus du calcul du temps de réponse qui reste lui identique à celui effectuée dans le modèle périodique [GGH97]. Dans un souci de clarté, nous utiliserons dans toute la suite le dernier modèle périodique.

Afin de calculer ces pires temps de réponse, nous devons mettre en exergue et analyser les scénarios les produisant.

2.3 Scénario produisant le pire temps de réponse

Lorsque les priorités des tâches sont fixes, le pire temps de réponse d'une tâche survient dans une période d'activité synchrone [Leh90]. Mais, lorsque les tâches sont ordonnancées par exemple selon une politique EDF ou FIFO, le pire temps de réponse ne survient pas nécessairement dans une période d'activité synchrone [Spu96a, GM97a]. Toutefois, ces auteurs donnent une caractérisation des périodes d'activité à étudier afin de calculer le pire temps de réponse d'une tâche. Cette caractérisation tient pour tous les algorithmes d'ordonnancement au plus tôt (conservatif).

Ainsi, pour une tâche τ_i, les seuls scénarios à étudier sont ceux où toutes les autres tâches sont synchrones. La propriété 2 prend alors toute son importance. En effet, la longueur de tous scénarios à tester dans le but d'obtenir le pire temps de réponse d'une tâche τ_i est borner par L. Le résultat suivant permet de diminuer le nombre de scénarios à envisager :

Propriété 3 *Le pire temps de réponse d'une tâche τ_i, $i \in [0, n-1]$ est obtenu lorsque toutes les tâches autre que τ_i arrivent simultanément.*

Ainsi, pour calculer le pire temps de réponse d'une tâche τ_i il suffit de considérer les scénarios où :
- toutes les autres tâches que τ_i sont activées simultanément,
- τ_i est activée à une date $t \in [0, T_i]$.

L'analyse de τ_i permettant d'obtenir son pire temps de réponse TR_i nécessite donc d'examiner *au plus* T_i scénarios, c'est-à-dire T_i périodes d'activité différentes.

Dans le chapitre 3 nous présentons une analyse plus fine permettant une meilleur approximation du pire temps de réponse en considérant diverses propriétés des tâches (échéance et période reliées ou non) et des ordonnanceurs (priorités fixes, priorités variables ...).
Tous les travaux effectués sur ce sujet sont situés dans un contexte mono-processeur. La technique présentée dans le chapitre 2.4 va permettre d'étendre ces travaux aux systèmes distribués. L'idée est de se ramener dans un contexte mono-processeur en modélisant toutes communications et précédences grâce à la notion de gigue temporelle sur l'activation des tâches.

2.4 Analyse Holistique

Le terme d'analyse holistique a été introduit par Kurt Tindell [Tin94, TC94], et symbolise la prise en compte de la dépendance entre l'ordonnancement des tâches et des messages dans les systèmes temps réel distribués. Précisément, la date d'émission d'un message dépend du temps de réponse de la tâche émettrice, et la date de réveil d'une tâche réceptrice dépend du temps de réponse du message à recevoir. La figure 2.4 présente l'ordonnancement de deux tâches τ_i et τ_j communiquant par message sur le réseau. L'émission du message est décalée par l'attente de la fin d'exécution de τ_i , et de même le début de τ_j est lié à l'arrivée du message sur son site. La gigue du message est notée J_m, et celle de la tâche réceptrice J_j.

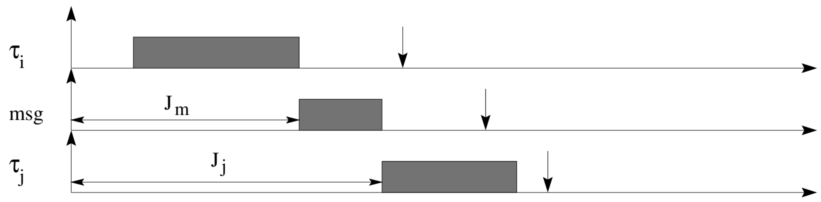

FIG. 2.4 – La gigue modélise l'attente due à la fin de la tâche précédente.

Les gigues sont les variables du problème modélisant la dépendance de l'ordonnancement conjoint des tâches et des messages. L'analyse holistique va permettre de les calculer par la résolution de systèmes d'équations récurrentes estimant le temps de réponse des tâches et des messages, et déterminant ainsi les valeurs des gigues. Nous obtenons ainsi les équations suivantes :

$$\begin{array}{ll} \text{Pour les tâches :} & J_i = \max\limits_{m \,\in\, prec(i)} (R^\star_m) \\ \text{Pour les messages :} & J_i = \max\limits_{j \,\in\, prec(i)} \left(R^\star_j\right) \end{array} \tag{2.4}$$

Toute modification des gigues va entraîner des modifications des temps de réponse des tâches dépendantes ou ceux des tâches ordonnancées sur le même processeur. Le principe de l'analyse holistique est de recommencer les calculs jusqu'à ce qu'un point fixe soit atteint. En pratique, les messages sont considérés comme des tâches et le réseau comme un processeur supplémentaire. Les contraintes de communication sont alors considérées comme des relations de précédence entre tâches. Nous définissons les fonctions *ResponseTime* qui calculent le pire temps de réponse d'une tâche τ_i en fonction des gigues des tâches et des messages et du processeur où τ_i est affectée. Soit n le nombre de tâches et de messages, le système d'équations à résoudre est :

$$1 \leq i \leq n \qquad \left\{ \begin{array}{lcl} R_i^{(0)} & = & C_i \\ J_i^{(0)} & = & 0 \\ R_i^{(k)} & = & ResponseTime\left(J_i^{(k-1)}\right) \\ J_i^{(k)} & = & \max\limits_{j \,\in\, prec(i)} \left(R_j^{(k)}\right) \end{array} \right. \tag{2.5}$$

Le point fixe est atteint lorsque pour un entier k on vérifie :

$$R_i^\star = R_i^{(k)} = R_i^{(k-1)}, \quad 1 \leq i \leq n \tag{2.6}$$

La convergence de l'algorithme repose sur une propriété unique de la fonction temps de réponse. Elle doit être non décroissante en fonction des gigues des tâches et des messages. Cette propriété assurant la convergence de la méthode fait que l'analyse holistique n'est qu'une condition suffisante d'ordonnançabilité.

Théorème 1 *L'analyse holistique converge si les fonctions calculant les pires temps de réponse des tâches et des messages sont non décroissantes en fonction des gigues des tâches et des messages.*

[SH98b] propose des heuristiques permettant d'accélérer de façon substantielle la convergence de l'algorithme, lorsque des tâches à priorité fixe sont considérées. Ces optimisations reposent sur l'ordre de résolution des équations récurrentes et sur l'amélioration de la borne inférieure qui les initialisent. L'algorithme principal de l'analyse holistique est présenté en annexe.

Chapitre 3

Calcul des pires temps de réponse des tâches et des messages

3.1 Pire temps de réponse des tâches à priorité fixe

Le calcul du pire temps de réponse d'une tâche, avec ordonnancement par priorité fixe, a été étudié par de nombreux auteurs. Les travaux originels dans ce cas sont [JP86, Leh90, Tin94]. Dans tout ce paragraphe nous supposons que les tâches sont triées par ordre de priorité : $i < j \Leftarrow Prio(\tau_i) > Prio(\tau_j)$.

Les premiers travaux dans le contexte monoprocesseur et concernant une configuration de n tâches périodiques, (mais toujours avec une période supérieure à l'échéance) ou sporadiques sont dus à [JP86]. Dans ce contexte, le théorème suivant donne le scénario aboutissant au pire temps de réponse d'une tâche τ_i.

Théorème 2 *[LL73, JP86] Le pire temps de réponse d'un tâche τ_i d'un ensemble de tâches périodiques ou sporadiques, est obtenu dans le scénario dans lequel **toutes** les tâches sont à leur rythme maximal et sont activées de manière synchrone à un instant critique $t = 0$.*

[LL73] pour des tâches périodiques à échéance sur requête, puis [JP86] dans un cadre plus général (i.e. $D_i \leq T_i$) ont prouvé que le pire temps de réponse d'une tâche τ_i était associé à la première instance de cette tâche à l'intérieur de la période d'activité synchrone. Ainsi, seule la première instance doit être testée pour pouvoir déterminer si la tâche est ordonnançable ou non. Or, dans le cas ou l'on n'impose pas de relation entre la période T_i et l'échéance D_i, [Leh90] montre que le théorème 2 n'est plus valable. En effet, il donne un exemple de configuration de tâche, avec des périodes inférieures aux échéances, pour laquelle le pire temps de réponse d'une des tâches est obtenu lors de la troisième instance de celle-ci.
Pour analyser le comportement d'une tâche, Lehoczky définit le concept de *période d'activité de niveau i* où le processeur n'exécute que des tâches de priorités supérieures ou égales à $Prio_i$ (i-level busy period). Cette nouvelle notion de période d'activité définit la durée d'étude nécessaire pour calculer le pire temps de réponse d'une tâche.

Définition 7 *[Leh90] Une période d'activité de niveau i est l'intervalle $[a, b]$ dans lequel seulement les tâches de priorité $Prio_i$ ou supérieure sont exécutées et aucune tâche de priorité $Prio_i$ ou plus n'est exécutée dans les intervalles $(a - \epsilon, a)$ et $(b, b + \epsilon)$, avec ϵ suffisamment petit.*

Lehoczky montre alors qu'une période d'activité de niveau i est initiée par un instant critique. Ce résultat est donc une extension du résultat sur l'instant critique de Liu et Layland [LL73], qui survient lorsqu'une tâche débute son exécution au même instant que toutes les tâches plus prioritaires qu'elle.

Théorème 3 *[Leh90] Le plus grand temps de réponse d'une tâche τ_i survient dans une période d'activité de niveau i où toutes les tâches débutent par un instant critique.*

La figure 3.1 donne l'exemple d'une période d'activité. Comme le montre cette figure, la fin de la tâche de priorité $Prio_i$ ne termine pas la période d'activité car une tâche de priorité supérieure à $Prio_i$, ou elle-même, peut interférer sur la prochaine exécution de τ_i. La période d'activité peut donc comporter plusieurs exécutions des tâches. La figure 3.1 considère une tâche τ_i et les tâches plus prioritaires qu'elle. Nous supposons que les échéances sont supérieures aux périodes. Pour les trois premières instances de τ_i (numérotées 1,2,3), la tâche se termine après l'activation de l'instance suivante, prolongeant ainsi la période d'activité.

Nous décrivons maintenant le calcul du temps de réponse qui repose dans la période d'activité. L'interférence liée aux tâches plus prioritaires que τ_i, dans une période d'activité de longueur t, est :

$$\sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \tag{3.1}$$

A cette interférence, il faut ajouter la charge processeur liée à τ_i : si $(q+1)$ requêtes sont activées dans la période $[0, t[$, la charge processeur induite par τ_i est :

$$(q+1)C_i \tag{3.2}$$

Le temps de réponse R_i de τ_i est alors donné par :

$$R_i = \max_{q=0,1,2,\ldots,Q} (t - qT_i) \tag{3.3}$$

qT_i est la date de la $(q+1)^{\text{ième}}$ requête de τ_i. Deux paramètres restent à calculer :

- le nombre Q de requêtes de τ_i contenues dans la période d'activité,
- la longueur t de la période d'activité.

Ces deux paramètres sont liés, mais il est important de remarquer que la période d'activité s'arrête lorsque le temps de réponse de la tâche τ_i est inférieur à sa période.

Théorème 4 *La période d'activité de niveau i se termine lorsque le temps de réponse de τ_i est inférieur à sa période. Plus formellement :*

$$t \leq (q+1)T_i \qquad q \in N \tag{3.4}$$

Le calcul des deux paramètres revient à résoudre le système d'équations :

$$\begin{array}{l} Min\ t \\ \text{Soumis à} \\ \qquad \begin{array}{rcl} t & = & (q+1)C_i + \displaystyle\sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \\ t & \leq & (q+1)T_i \end{array} \end{array} \tag{3.5}$$

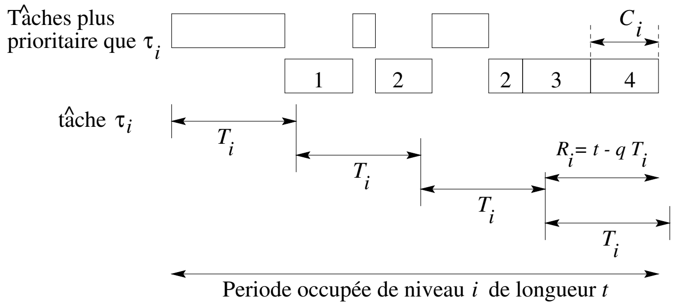

FIG. 3.1 – Une période d'activité de niveau i

La dernière équation de ce système est le test d'arrêt, qui revient à tester que le temps de réponse de l'instance courante est inférieur à la prochaine requête de la tâche. Le point fixe $t(k) = t(k+1)$ est recherché par une méthode itérative. Dans une période d'activité, la charge minimale est définie par l'exécution d'une instance de chaque tâche. Ceci définit la valeur initiale $t(0)$ de la suite. Précisément, la suite récurrente est :

$$\begin{cases} t^{(0)} & = & \sum_{j=1}^{i-1} C_i \\ t^{(k+1)} & = & (q+1)C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t^{(k)}}{T_j} \right\rceil C_j \end{cases} \tag{3.6}$$

Sous cette forme, q est un paramètre. Le principe de l'algorithme va consister à incrémenter q tant que le test d'arrêt n'est pas vérifié. Chaque incrémentation de q revient à explorer une période supplémentaire de la tâche τ_i, appartenant à la période d'activité.

Nous étendons maintenant les formules afin de prendre en compte la durée maximum de blocage B_i associée à la prise d'une ressource par la tâche τ_i. L'inversion de priorité ne peut survenir qu'une fois, et seulement au début de la période d'activité. En conséquence le pire temps d'attente Bi est simplement ajouté au temps de réponse de la tâche, obtenu sans prendre en compte les ressources (la suite reste initialisée comme précédemment) :

$$t^{(k+1)} = (q+1)C_i + B_i + \sum_{j=1}^{i-1} \left\lceil \frac{t^{(k)}}{T_j} \right\rceil C_j \tag{3.7}$$

La méthode précédente suppose que la date d'activation (release time) de la tâche est connue a priori. Ceci n'est pas le cas si, par exemple, une tâche est bloquée dans l'attente d'un message. Dans la littérature, ce problème est appelé le problème de la gigue sur l'activation des tâches (the release jitter problem). La différence entre deux arrivées successives d'une tâche peut être plus grande que la différence entre deux réveils successifs. La gigue d'une tâche τ_j, notée J_j, doit être prise en compte dans le calcul du temps de réponse d'une tâche. Cette extension élégante des formules précédentes a été effectuée dans [Tin94].

L'instant critique survient lorsque toutes les tâches reçoivent leurs messages au même instant, comme indiqué figure 3.2. Ainsi les dates d'activation des tâches dans la période d'activité sont données par :

$$\max(qT_i - J_i; 0) \qquad \forall\ q = 0...Q \tag{3.8}$$

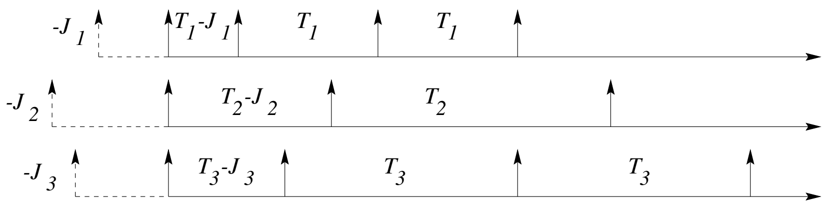

FIG. 3.2 – Prise en compte des gigues sur l'activation des tâches

La période d'activité se termine lorsque le temps de réponse R_i est inférieur à la période T_i. Le théorème 3 reste donc valide en présence de gigue sur l'activation des tâches. La preuve de [Tin94] a été montrée incomplète dans [SL96]. Mais un complément de preuve est donné dans [GGH97]. L'interférence des tâches plus prioritaires est calculée par le théorème 5.

Théorème 5 *L'interférence dans le pire cas engendrée par les tâches plus prioritaires que τ_i, sur des tâches de priorité inférieure dans une période d'activité de longueur t, est donnée par :*

$$\sum_{j=1}^{i-1} \left\lceil \frac{J_j + t}{T_j} \right\rceil C_j \tag{3.9}$$

Tindell montre que l'accroissement d'une gigue ne peut faire décroître le temps de réponse d'une tâche quelconque. Ainsi le calcul d'une borne du pire temps de réponse d'une tâche τ_i en présence de gigue se fait par :

$$\begin{aligned} R_i &= \max_{q=0,1,2,\ldots Q} (t + J_i - qT_i) \\ t^{(k+1)} &= (q+1)C_i + B_i + \sum_{j=1}^{i-1} \left\lceil \frac{J_j + t^{(k)}}{T_j} \right\rceil C_j \end{aligned} \tag{3.10}$$

[SH98b] montre que chaque suite récurrente peut être initialisée par une borne inférieure quelconque du temps de réponse de la tâche considérée, sans risque de rater le point fixe. Pour une valeur q fixée, une borne inférieure s'obtient facilement depuis l'équation récurrente de recherche du pire temps de réponse d'une tâche. Ainsi la suite récurrente est initialisée avec la borne inférieure du temps de réponse suivante :

$$t^{(0)} = \frac{(q+1)C_i + B_i + \sum_{j=1}^{i-1} \frac{J_j}{T_j} C_j}{1 - \sum_{j=1}^{i-1} \frac{C_j}{T_j}} \tag{3.11}$$

L'algorithme *ResponseTime_FixedPriority*, présenté en annexe, définit la résolution de ces équations récurrentes. Les gigues sont des paramètres en entrée de l'algorithme, ainsi que la durée maximum de blocage. L'initialisation des suites récurrentes est faite par la formule précédente.

3.2 Pire temps de réponse des tâches à priorité dynamique

Lorsque l'ordonnancement des tâches est fait selon EDF (Earliest Deadline First), le concept de période d'activité va permettre de borner l'intervalle d'étude de l'analyse d'ordonnançabilité. Les priorités étant dynamiques, il n'est pas possible de déterminer a priori quel sous-ensemble de tâches va préempter une tâche τ_i donnée.

Liu et Layland [LL73] montrent qu'il est possible de limiter l'intervalle d'étude pour valider les tâches à l'intervalle défini par la plus longue période d'activité du processeur, c'est-à-dire sans temps creux (i.e. unité de temps où le processeur est libre).

Théorème 6 *S'il y a une surcharge pour une certaine arrivée des tâches, alors il existe une surcharge sans temps creux avant celle-ci, initiée par un instant critique.*

La charge cumulée du processeur sur l'intervalle $[0, t[$ est donnée par $W(t)$, et la charge sur l'intervalle $[0, t]$ est donnée par $\overline{W}(t)$:

$$W(t) = \sum_{i=0}^{n-1} \left\lceil \frac{t}{T_i} \right\rceil C_i \qquad \overline{W}(t) = \sum_{i=0}^{n-1} \left(1 + \left\lfloor \frac{t}{T_i} \right\rfloor\right) C_i \tag{3.12}$$

La longueur L de la période d'activité peut être calculée de façon itérative, comme dans le cas des tâches à priorité fixe. C'est-à-dire :

$$\left\{ \begin{array}{rcl} L^{(0)} &=& \sum_{i=0}^{n-1} C_i \\ L^{(n+1)} &=& W\left(L^{(n)}\right) \end{array} \right. \tag{3.13}$$

La longueur L est donnée par : $L = L^{(n+1)} = L^{(n)}$, c'est-à-dire que le point fixe a été atteint. Ce système admet un point fixe si, et seulement si, la configuration des tâches (à échéance sur requête et à départ simultané au démarrage) est ordonnançable. Cette dernière condition correspond au test d'ordonnançabilité pour EDF [LL73] :

$$\sum_{i=0}^{n-1} \frac{C_i}{T_i} \leq 1 \tag{3.14}$$

Remarquons que si le facteur d'utilisation processeur est égale à 1, alors la période d'activité est infinie. Or il est connu que la période d'analyse d'une configuration de tâche synchrone au démarrage est bornée par le plus petit commun multiple des périodes [LM80]. L'algorithme *BusyPeriod*, donné en annexe, calcule la longueur L de la période d'activité.

Nous étudions tout d'abord le calcul du temps de réponse d'une tâche en considérant que les gigues sont nulles. Ces résultats seront ensuite étendus pour traiter les gigues dues à l'attente des messages et les blocages liés à la prise des ressources. Contrairement au cas des priorités fixes, le pire temps de réponse d'une tâche n'est pas nécessairement obtenu lorsque toutes les tâches sont réveillées à l'instant critique, débutant une période d'activité. [Spu96a] donne une condition permettant de calculer le temps de réponse.

Théorème 7 *Le pire temps de réponse d'une tâche τ_i est trouvé dans une période d'activité où les tâches autres que τ_i sont réveillées de façon synchrone et aussitôt que possible.[Spu96a]*

D'après ce résultat, pour évaluer le pire temps de réponse de τ_i, noté R_i, il est nécessaire d'examiner plusieurs dates d'activation pour la tâche τ_i, alors que les autres tâches sont toutes activées à l'instant où débute la période d'activité (i.e. la date 0). Si une instance de τ_i débute à la date a, la première instance dans la période d'activité débute à la date $s_i(a)$, calculée comme :

$$S_i(a) = a - \left\lfloor \frac{a}{T_i} \right\rfloor T_i \tag{3.15}$$

Pour évaluer la période d'activité, par rapport aux dates d'arrivée des instances de τ_i, il est utile de définir le concept de période d'activité associée à l'échéance $d = a + Di$. Cette notion permet de clarifier la présentation des calculs.

Définition 8 *La période d'activité associée à une échéance d (deadline d-busy period) est définie par l'exécution des tâches dont les échéances sont inférieures ou égales à d.[GRS96]*

La période d'activité associée à $d = a + D_i$ repose sur la charge des tâches plus prioritaires, c'est-à-dire selon EDF, les tâches ayant une échéance inférieure ou égale à d. Cette charge va donc dépendre de la date t considérée, et de la date d'activation a de l'instance de τ_i considérée. La charge des tâches autres que τ_i varie selon que $t < a$ (l'intervalle considéré est alors $[0, t[$) ou que $t > a$ (l'intervalle considéré est $[0, d - D_j]$) :

$$\sum_{\substack{j \neq i \\ D_j \leq a + D_i}} \left(\min \left(\left\lceil \frac{t}{T_j} \right\rceil, \ 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor \right) \right) C_j \tag{3.16}$$

La charge processeur associée aux instances de τ_i doit tenir compte des cas suivants :
- si $t \leq s_i(a)$, alors la charge liée à τ_i est nulle.
- si $t > s_i(a)$, alors dans l'intervalle $[s_i(a), t[$ la charge de τ_i est :
  - si $a > t$

$$\left\lceil \frac{t - s_i(a)}{T_i} \right\rceil C_i$$

  - sinon nous devons considérer les instances jusqu'à la date a incluse (intervalle $[0, a]$) :

$$\left(1 + \left\lfloor \frac{a}{T_i} \right\rfloor \right) C_i$$

Nous pouvons donc définir la charge processeur $\delta_i(a,t)$ des instances de τ_i dans la période d'activité par une formule unique, regroupant les cas précédents :

$$\delta_i(a,t) = \min\left(\left\lceil \frac{t - s_i(a)}{T_i} \right\rceil, 1 + \left\lfloor \frac{a}{T_i} \right\rfloor\right) C_i \qquad \text{si } t > s_i(a),\ 0 \text{ sinon} \tag{3.17}$$

La charge processeur $W_i(a,t)$ dans la période d'activité associée à $d = a + D_i$ est :

$$W_i(a,t) = \sum_{\substack{j \neq i \\ D_j \leq a+D_i}} \left(\min\left(\left\lceil \frac{t}{T_j} \right\rceil,\ 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right) C_j\right) + \delta_i(a,t) \tag{3.18}$$

Le calcul de la longueur de la période d'activité associée à d peut être fait de façon itérative. Pour cela il faut définir une borne inférieure de cette période pour initialisée la suite. La période d'activité comportera au minimum une exécution de chaque tâche autre que τ_i, et une de τ_i si $s_i(a) = 0$. On obtient donc le système suivant :

$$\begin{array}{l} I_i = 1 \quad \text{si} \quad s_i(a) = 0 \quad , \quad 0 \text{ sinon} \\ \left\{ \begin{array}{rcl} L_i^{(0)}(a) & = & \displaystyle\sum_{\substack{i \neq j \\ D_j \leq a+D_i}} C_j + I_i C_i \\ L_i^{(m+1)}(a) & = & W_i\left(a, L_i^{(m)}(a)\right) \end{array} \right. \end{array} \tag{3.19}$$

[Spu96a] montre qu'une condition nécessaire et suffisante de convergence est que la charge processeur soit inférieure à 1. La longueur de la période d'activité associée à $d = a + D_i$ est :

$$L_i(a) = L_i^{(m+1)}(a) = L_i^{(m)}(a) \tag{3.20}$$

Connaissant la longueur de la période d'activité associée à une échéance d de τ_i, nous pouvons déterminer le pire temps de réponse de τ_i (en considérant qu'une instance de τ_i débute à la date a).

$$R_i(a) = \max\left(C_i, L_i(a) - a\right) \tag{3.21}$$

Le dernier problème à résoudre est de choisir les dates a à considérer. Ces dates doivent appartenir à la période d'activité, donc à l'intervalle $[0, L_i(a) - C_i[$. D'une part, on vérifie que $L_i(a) \leq L$, où L est la longueur de la période d'activité débutant par un instant critique. D'autre part, [Spu96a] montre que les maxima locaux de la fonction $L_i(a)$ reviennent à choisir l'échéance $d = a + D_i$ d'une instance de τ_i égale à une échéance d'une autre tâche dans la période d'activité associée à d, ou bien à considérer les dates a telles que $s_i(a) = 0$ (c'est-à-dire que la période d'activité débute par un instant critique).
Le premier ensemble de dates est défini par les solutions de :

$$a + D_j = kT_j + D_j \qquad k \in N \ , \quad j \neq i \ , \ kT_j + D_j \leq L - C_i \tag{3.22}$$

Le deuxième ensemble de dates correspondant à l'ensemble des solutions de l'équation appartenant à la période d'activité associée à $d = a + D_i$:

$$s_i(a) = a - \left\lfloor \frac{a}{T_i} \right\rfloor T_i = 0 \qquad \rightarrow \qquad a = kT_i \ , \ k \in N \tag{3.23}$$

Ces deux cas peuvent être regrouper en considérant l'ensemble A des dates d'activation de τ_i dans la plus longue période d'activité, c'est-à-dire $[0, L - C_i[$:

$$A = \{a | a = kT_j + D_j - D_i \ , \ k \ \in N \ , \ kT_j + D_j \leq L - C_i\} \tag{3.24}$$

Cette remarque permet donc de limiter le nombre de dates d'activation possible pour τ_i dans la période d'activité.

La prise en compte des gigues repose sur le constat que le pire cas survient si tous les messages arrivent à la même date, définissant ainsi un instant critique (figure 4). Précisément, le théorème 6 reste valide en prenant en compte les gigues [Spu96a]. L'ensemble des formules doit simplement être adapté pour déterminer les dates d'activation des tâches, en prenant en compte les gigues. La première activation de τ_i a lieu à la date $-J_i$, et les suivantes aux dates $kT_i - J_i$, où k est un entier naturel.
Tout d'abord, la charge processeur doit tenir compte des gigues J_i des tâches. Le calcul de la longueur de la période d'activité repose sur la formule suivante :

$$W(t) = \sum_{i=0} n - 1 \left\lceil \frac{t + J_i}{T_i} \right\rceil C_i \tag{3.25}$$

Pour une tâche τ_i, le nombre d'instances réveillées avant, ou à la date a est :

$$\left(1 + \left\lfloor \frac{a + J_i - D_i}{T_i} \right\rfloor\right) C_i \tag{3.26}$$

Si une date d'activation de τ_i est a alors la première date d'activation $s_i(a)$ est :

$$s_i(a) = a + J_i - \left\lfloor \frac{a + J_i}{T_i} \right\rfloor T_i \tag{3.27}$$

Nous pouvons donc définir la charge processeur $\delta_i(a,t)$ des instances de τ_i par la formule unique, regroupant les cas précédents :

$$\delta_i(a,t) = \min\left(\left\lceil \frac{t - s_i(a) + J_i}{T_i} \right\rceil,\ 1 + \left\lfloor \frac{a + J_i}{T_i} \right\rfloor\right) C_i \qquad \text{si } t > s_i(a), \quad 0 \text{ sinon} \tag{3.28}$$

La charge processeur $W_i(a,t)$ dans la période d'activité associée à $d = a + D_i$ est donc :

$$W_i(a,t) = \sum_{\substack{i \neq j \\ D_j \leq a + J_j + D_i}} \left(\min\left(\left\lceil \frac{t + J_j}{T_j} \right\rceil,\ 1 + \left\lfloor \frac{a + D_i + J_j - D_j}{T_j} \right\rfloor\right) C_j\right) + \delta_i(a,t) \tag{3.29}$$

Comme précédemment, la longueur de la période d'activité associée à d revient à chercher le point fixe du système suivant :

$$\begin{array}{l} I_i = 1 \quad \text{si} \quad s_i(a) = 0 \quad , \quad 0 \text{ sinon} \\ \left\{ \begin{array}{rcl} L_i^{(0)}(a) & = & \displaystyle\sum_{\substack{i \neq j \\ D_j \leq a + D_i + J_j}} C_j + I_i C_i \\ L_i^{(m+1)}(a) & = & W_i\left(a, L_i^{(m)}(a)\right) \end{array} \right. \end{array} \tag{3.30}$$

La longueur de la période d'activité associée à $d = a + D_i$ est le point fixe du système. Connaissant la longueur de la période d'activité associée à une échéance d de τ_i, nous pouvons déterminer le pire temps de réponse de τ_i (en considérant qu'une instance de τ_i débute à la date a) :

$$R_i(a) = \max(C_i + J_i, L_i(a) - a) \tag{3.31}$$

Le dernier problème à résoudre est de choisir les dates a à considérer. Les dates a doivent appartenir à la période d'activité, donc à l'intervalle $[-J_i, L_i(a) - J_i - C_i[$. Comme précédemment les maxima locaux de la fonction $L_i(a)$ reviennent à choisir l'échéance $d = a + D_i$ d'une instance de τ_i égale à une échéance d'une autre tâche dans la période d'activité associée à d, ou bien à considérer les dates a telles que $s_i(a) = 0$ (c'est-à-dire que la période d'activité débute par un instant critique). Le premier ensemble de dates est défini par les solutions de :

$$a + D_i = kT_j + D_j - J_j \qquad k \in N\ ,\ j \neq i\ ,\ kT_j + D_j - J_j \leq L - J_i - C_i \tag{3.32}$$

Le deuxième ensemble de dates correspond à l'ensemble des solutions de l'équation appartenant à la période d'activité associée à $d = a + D_i$:

$$s_i(a) = a + J_i - \left\lfloor \frac{a + J_i}{T_i} \right\rfloor T_i = 0 \quad \Rightarrow \quad a = kT_i - J_i \ , \ k \in N \tag{3.33}$$

Ces deux cas peuvent être regrouper en considérant l'ensemble A des dates d'activation possible pour τ_i :

$$A = \{a | a = kT_j + D - j - J_j - D_i \ , \ k \in N \ , \ kT_j + D_j - J_j \leq L - C_i - J_i\} \tag{3.34}$$

Jusqu'à maintenant les tâches ne partagent pas de ressource. Comme dans le cas à priorité fixe, nous étendons les résultats précédents avec la prise en compte du pire temps de blocage B_i des tâches, aussi appelé le facteur de blocage. Nous rappelons que cette quantité est fonction du protocole qui régit l'accès aux ressources [But97]. Bien que la charge processeur ne change pas en présence de ressource, le phénomène d'inversion de priorité va modifier l'ordonnancement EDF. En conséquence le facteur de blocage doit être pris en compte dans le calcul du pire temps de réponse des tâches.

[Spu96a] montre que parmi les tâches plus prioritaires que τ_i, nous devons considérer le facteur de blocage de la tâche τ_k dont la valeur $D_k - J_k$ est la plus grande. On définit une fonction $k(d)$, où d est une date, permettant de déterminer cette tâche :

$$k(d) = \max \{k | D_k - J_k \leq d\} \tag{3.35}$$

Le facteur va être pris en compte dans le calcul de la période d'activité associée à une échéance $d = a + D_i$. On obtient :

$$L_i^{(m+1)} = W_i \left(a, L_i^{(m)}(a) \right) + B_{k(a+D_i)} \tag{3.36}$$

De même, le temps de réponse d'une tâche, comme dans le cas à priorité fixe, intègre directement le facteur de blocage dans le calcul.

$$R_i(a) = \max (C_i + J_i + B_i; L_i(a) - a) \tag{3.37}$$

Les activations de la tâche τ_i à considérer appartiennent à l'intervalle $[-J_i, L - J_i - C_i - B_i[$. Le choix des valeurs de a, maximisant localement les fonctions $L_i(a)$ repose sur le même principe. En conclusion, le pire temps de réponse de la tâche τ_i est obtenu comme :

$$R_i = \max_{a \in A} (R_i(a)) \tag{3.38}$$

Les algorithmes nécessaires au calcul du temps de réponse d'une tâche sont donnés en annexe. D'un point de vue complexité, [Spu96a] montre que la recherche d'un point fixe pour calculer la longueur de la période d'activité est :

$$O \left(n \sum_{i=0}^{n-1} C_i \right) \tag{3.39}$$

Notons enfin que l'existence d'un algorithme fortement polynomial est un problème ouvert [Spu96b].

3.3 Pire temps de réponse des messages

[TBW95a] étudie deux protocoles de communication : le jeton temporisé et le bus à priorité. [Spu96b] étudie un réseau avec une topologie en anneau avec jeton pour accéder au médium de communication. [ZB95] étudie le cas du réseau FDDI. [PB97] étudie le cas du réseau FIP avec un trafic de messages sporadiques. [SH98a, EHS98] étudient le réseau ATM, et enfin [AG97] étudie le réseau ARINC 629 (domaine de l'aéronautique). Nous développons le calcul du pire temps de réponse des messages à travers l'exemple du réseau CAN (bus à priorité) [THW94, TBW95b], et montrons le lien avec l'ordonnancement de tâches non préemptives.

3.3.1 Réseau CAN

Le temps de propagation d'une trame de n octets ($n \leq 8$) dans le réseau CAN, où τ_{bit} est le temps de propagation d'un bit dans le réseau, est donné par la formule suivante :

$$C_m = \left(\left\lfloor \frac{34 + 8n}{5} \right\rfloor + 47 + 8n \right) \tau_{bit} \tag{3.40}$$

Le réseau CAN est une ressource non partageable et chaque message possède une priorité fixe, arbitrant l'accès au canal de communication. En conséquence, l'ordonnancement du réseau s'apparente alors à l'ordonnancement monoprocesseur non préemptif de tâches. Dans la suite nous ne ferons plus de différence entre les termes messages et tâches.

Conceptuellement, il n'existe pas de différence entre l'ordonnancement préemptif et non préemptif, dans le sens où les temps creux laissés par ces deux modes d'ordonnancement sont les mêmes. Ceci résulte directement que la charge processeur engendrée par une configuration est la même en préemptif qu'en non préemptif. Nous illustrons ce propos figure 3.3, où la configuration de tâches considérées est ordonnancée dans les deux cas par l'algorithme *Rate Monotonic* (plus petite période d'abord).

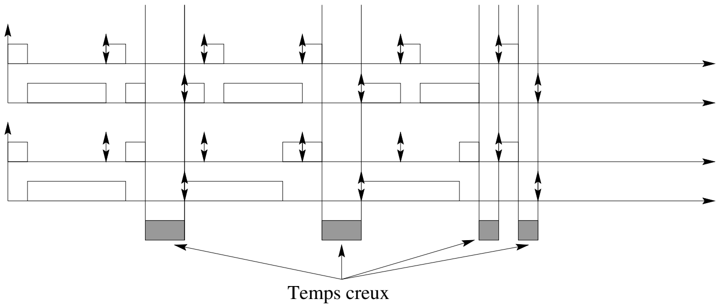

FIG. 3.3 – Ordonnancement préemptif et non-préemptif d'une configuration de tâche sur un processeur.

Il découle alors que les résultats conçus pour l'ordonnancement préemptif peuvent être aisément adaptés à l'ordonnancement non préemptif de tâches. L'unique différence est liée au décalage pouvant être engendré par une tâche moins prioritaire, utilisant le processeur à l'instant critique. Les tâches ne pouvant pas être interrompues, l'instant critique va être décalé d'au maximum la durée de la plus longue tâche, parmi les tâches moins prioritaires. Le pire cas survient lorsque cette tâche débute une unité de temps avant l'instant critique, c'est-à-dire l'instant -1 en imposant que l'instant critique survienne à une date zéro (changement de l'origine des temps).

Théorème 8 *[GRS96] En ordonnancement non préemptif à priorité fixe, le pire temps de réponse d'une tâche τ_i est trouvé dans la période d'activité de niveau i et survient lorsque toutes les tâches plus prioritaires sont réveillées à un instant critique, et en réveillant la tâche la plus lente parmi les moins prioritaires (s'il en existe) à la date -1.*

Dans un réseau informatique, la durée de transmission n'est pas nécessairement une grandeur discrète. En conséquence, la tâche la moins prioritaire débutera donc à la date 0 et non à une date discrète -1. Cette légère modification constitue une borne supérieure du résultat présenté dans [GRS96]. Nous incluons immédiatement la présence d'une gigue sur le message, qui modélisera le temps de réponse de la tâche émettrice. On obtient donc des formules très proches de celles établies dans le cas préemptif à priorité fixe (nous supposons que les messages sont triés par ordre de priorité croissante) :

$$\left\{ \begin{array}{rcl} R_i & = & \max\limits_{q=0,1,2,\ldots,Q} (t + J_i - qT_i) \\ t & = & (q+1)C_i + \sum\limits_{j=0}^{i-1} \left(\left\lfloor \frac{J_j + t}{T_j} \right\rfloor + 1 \right) C_j + \max\limits_{i<k\leq n} (C_k) \end{array} \right. \tag{3.41}$$

L'algorithme de résolution est similaire à de celui présenté pour calculer le temps de réponse de tâches préemptives à priorité fixe. Pour cette raison nous ne le détaillons pas.

3.4 Exemples d'applications

L'intérêt de l'approche est de pouvoir étudier des systèmes hétérogènes, où les processeurs n'utilisent pas les mêmes politiques d'ordonnancements, d'accès aux ressources et peuvent être reliés par un ensemble de réseaux de topologie quelconque. Nous présentons un cas d'école afin de fournir quelques résultats numériques de l'implémentation des algorithmes fournis en annexe.

Nous considérons 2 processeurs reliés par un réseau CAN. Le premier processeur (P_1) ordonnance des tâches à priorité fixe, tandis que le second (P_2)les exécutent selon EDF. Les caractéristiques des tâches sont indiquées dans le tableau 3.1. Comme nous l'avons vu dans les paragraphes précédents, le réseau est un processeur supplémentaire et les messages sont des tâches non préemptives. La structure logicielle et matérielle de l'application est donnée figure 3.4. Sans perte de généralité, nous imposons des temps de propagation de 1 aux messages, pour minimiser le volume des calculs. Nous supposons τ_1 moins prioritaire que τ_2 sur le processeur P_1 et m_1 plus prioritaire que m_2 dans le réseau CAN.

Nom	Processeur	C_i	D_i	T_i
τ_1	P_1	52	160	100
τ_2	P_1	52	180	160
τ_3	P_2	10	60	40
τ_4	P_2	20	80	60
τ_5	P_2	20	124	160
τ_6	P_2	20	188	100
m_1	CAN	1		
m_2	CAN	1		

TAB. 3.1 – Caractéristiques des tâches

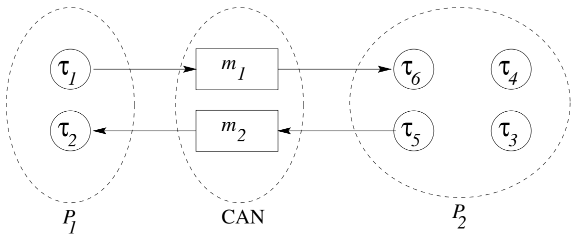

FIG. 3.4 – Graphe de précédence des tâches et des messages

La résolution de ce problème avec l'analyse holistique présentée se fait en seulement deux itérations dont les résultats respectifs sont présentés tableau 2. Pour chaque itération sont reportés le temps de réponse R_i et la gigue J_i.

Nous détaillons maintenant les calculs du temps de réponse de la tâche τ_1 durant la dernière itération de l'analyse holistique. La valeur maximum après les quatre itérations du tableau 3.3 est $R_1 = 156$.

Nous terminons cet exemple avec l'étude du processeur P_2 : nous illustrons le calcul de la période d'activité pour la dernière itération de l'analyse holistique, conduisant à une longueur $L = 480$ au

Tâches	Itération 1		Itération 2	
	R_i	J_i	R_i	J_i
τ_1	104	0	156	0
τ_2	115	63	1356	83
τ_3	30	0	50	0
τ_4	50	0	70	0
τ_5	60	0	80	0
τ_6	158	106	178	158
m_1	106	104	158	156
m_2	63	60	83	80

TAB. 3.2 – Trace des deux itérations de l'analyse holistique

bout de 12 itérations (Cf. ci-dessous).

$$\begin{array}{lllll}
L^{(0)} & = 70 & & & \\
L^{(1)} & = W\left(L^{(0)}\right) & = & \left\lceil\frac{70+0}{40}\right\rceil \times 10+\left\lceil\frac{70+0}{60}\right\rceil \times 20+\left\lceil\frac{70+0}{160}\right\rceil \times 20+\left\lceil\frac{70+158}{100}\right\rceil & = 140 \\
L^{(2)} & = W\left(L^{(1)}\right) & = & \left\lceil\frac{140+0}{40}\right\rceil \times 10+\left\lceil\frac{140+0}{60}\right\rceil \times 20+\left\lceil\frac{140+0}{160}\right\rceil \times 20+\left\lceil\frac{140+158}{100}\right\rceil & = 180 \\
L^{(3)} & = W\left(L^{(2)}\right) & = & \left\lceil\frac{180+0}{40}\right\rceil \times 10+\left\lceil\frac{180+0}{60}\right\rceil \times 20+\left\lceil\frac{180+0}{160}\right\rceil \times 20+\left\lceil\frac{180+158}{100}\right\rceil & = 230 \\
\ldots & & & & \\
L^{(10)} & = W\left(L^{(9)}\right) & = & \left\lceil\frac{430+0}{40}\right\rceil \times 10+\left\lceil\frac{430+0}{60}\right\rceil \times 20+\left\lceil\frac{430+0}{160}\right\rceil \times 20+\left\lceil\frac{430+158}{100}\right\rceil & = 450 \\
L^{(11)} & = W\left(L^{(10)}\right) & = & \left\lceil\frac{450+0}{40}\right\rceil \times 10+\left\lceil\frac{450+0}{60}\right\rceil \times 20+\left\lceil\frac{450+0}{160}\right\rceil \times 20+\left\lceil\frac{450+158}{100}\right\rceil & = 480 \\
L^{(12)} & = W\left(L^{(11)}\right) & = & \left\lceil\frac{480+0}{40}\right\rceil \times 10+\left\lceil\frac{480+0}{60}\right\rceil \times 20+\left\lceil\frac{480+0}{160}\right\rceil \times 20+\left\lceil\frac{480+158}{100}\right\rceil & = 480
\end{array}$$

La méthode holistique permet d'étudier des systèmes complexes, composés de processeurs ordonnancés individuellement, partageant un ou plusieurs réseaux. Cette caractéristique se retrouve dans de nombreux systèmes réels où au niveau capteur se trouvent des processeurs ordonnançant les tâches en suivant des priorités fixes. Ces processeurs, couplés au procédé contrôlé, sont usuellement reliés par un réseau de terrain. Tandis qu'au niveau supervision, les processeurs communiquent via un réseau haut débit comme FDDI ou ATM. L'analyse holistique est à notre connaissance l'unique méthode permettant de valider de telles architectures distribuées.
De plus les temps de calculs sont généralement très courts puisqu'ils dépendent essentiellement de la charge des processeurs, et non de leurs nombres.

$q = 0$ $t^{(1)} = 52+\left\lceil\frac{104+83}{160}\right\rceil 52 = 156$ $t^{(2)} = 52+\left\lceil\frac{156+83}{160}\right\rceil 52 = 156$ $R_1 = 156-0\times 100 = 156$	$q = 1$ $t^{(1)} = 2\times 52+\left\lceil\frac{156+83}{160}\right\rceil 52 = 208$ $t^{(2)} = 2\times 52+\left\lceil\frac{208+83}{160}\right\rceil 52 = 208$ $R_1 = 208-1\times 100 = 108$
$q = 2$ $t^{(1)} = 3\times 52+\left\lceil\frac{208+83}{160}\right\rceil 52 = 260$ $t^{(2)} = 3\times 52+\left\lceil\frac{260+83}{160}\right\rceil 52 = 312$ $t^{(3)} = 3\times 52+\left\lceil\frac{312+83}{160}\right\rceil 52 = 312$ $R_1 = 312-2\times 100 = 112$	$q = 0$ $t^{(1)} = 4\times 52+\left\lceil\frac{312+83}{160}\right\rceil 52 = 364$ $t^{(2)} = 4\times 52+\left\lceil\frac{364+83}{160}\right\rceil 52 = 364$ $R_1 = 364-3\times 100 = 64$

TAB. 3.3 – Calcul du temps de réponse de τ_1 sur le processeur P_1

3.5 Notes Bibliographiques

L'analyse holistique définit un cadre général d'étude qui peut être facilement étendu pour traîter des cas plus complexes. Par exemple si les tâches en relation de communication sont synchronisées temporellement et non sur la réception d'un événement (i.e. les activations des tâches sont décalées de façon à ce que les messages soient arrivés à destination), les formules présentées doivent être adaptées. Une telle étude à été menée pour les tâches à priorité fixe dans [Tin94, Sun97, PH98] et dans [DG99, Goo99] pour l'ordonnancement EDF.
Lorsque l'activation de tâches n'est pas dirigée par le temps mais par les événements, le calcul du temps de réponse à un événement repose, en plus du pire temps de réponse, sur le calcul du meilleur temps de réponse d'une tâche [GGH98].

Notons également que l'analyse holistique repose sur la majoration de la fonction de la charge processeur (dates d'arrivées des tâches). Des résultats sont énnoncés dans le cas préemptif et non préemptif avec présence de surcharge processeur dans [LBJMC92] et [LBCJM95]. Une étude générale des fonctions de majoration de la charge processeur est menée dans [MJM98] et [MJM99] pour diverses politiques d'ordonnancement, dont notamment à priorité fixe et de type *round-robin* multi-niveaux.

Le calcul du temps de réponse d'une tâche dans un ordonnancement FIFO (i.e. first in, first out) est étudié dans [GM97b]. Dans [ABR+96] est présenté le calcul du temps de réponse d'une tâche périodique ou apériodique ordonnancer périodiquement (i.e. l'ordonnanceur est invoqué par une interuption périodique)(*Tick Scheduling*).

l'analyse holistique peut être étendue à différents types de tâche. Dans [BBG97, Tin94] on trouve une extension de cette méthode permettant la prise en compte de groupes de tâches apériodiques dans un système distribué.
Dans [ABR+96, Spu96a] les auteurs montrent qu'il est possible de calculer le pire temps de réponse de tâches *sporadiquement périodique* (i.e. une tâche peut s'exécuter de manière périodique un certain nombre de fois, puis ne plus être activé pendant une longue période et puis reprendre une activation périodique etc.).
L'analyse holistique permet également d'étudier les tâches dites *multiframes* [MJMN99].

L'analyse holistique permet également de borner de manière fine le temps de réponse d'un message pour un grand nombre de protocoles d'accès. [TBW95b] propose une étude très fine du réseau CAN et propose une borne simple (i.e. fonctionnement idéale) et une borne beaucoup plus fine prenant en compte les détections et corrections d'erreurs de transmission. On trouve également dans [TBW95c] une étude de deux protocoles différents, TDMA et 802.5 respectivement étudiés pour deux approches différentes, *sur demande* et *serveur périodique.*

Chapitre 4

conclusion

Nous avons considéré dans cet article l'ordonnançabilité d'une configuration de tâches à priorité fixe et dynamique, devant s'exécuter sur un ensemble de sites reliés par un réseau. La validation d'un système temps réel distribué par une simulation directe du système est impossible si des ordonnanceurs en ligne sont utilisés sur les sites. Car la transmission des messages peut engendrer des inversions de priorité pour les tâches. L'analyse holistique est une approche élégante permettant de résoudre ce problème. Elle repose sur le calcul du pire temps de réponse des tâches et des messages dans un intervalle de temps restreint appelé la *période active* et sur la recherche de points fixes dans des systèmes d'équations récurrentes.

Bibliographie

[ABR⁺96] N. C. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. Technical report, Departement of Computer Science, University of York, 96.

[AG97] N. C. Audsley and A. Grigg. Timing analysis of the arinc 629 databus for real-time application. *Microprocessors & Microsystems*, 21(1), 1997.

[BBG97] P. Bizzarri, A. Bondavalli, and F. Di Giandomenico. A scheduling algorithm for aperiodic groups of tasks in distributed real-time systems and its holistic analysis. *Procedings of the 6th IEEE Workshops on Future Trends of Distributed Computing Systems (FTDCS'97)*, 1997.

[BMR90] S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard real-time sporadic tasks on one processors. *Proc. of the 11th Real-Time Systems Symposium*, pages 182–190, 1990.

[BRH90] S. K. Baruah, L. E. Rosier, and R. R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2 :301–324, 1990.

[But97] G. C. Buttazzo. *Hard Real-Time Computing Systems : Predicatable Scheduling Algorithms and Applications*. Kluwer Academic Publisher, 1997.

[CDKM00] F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri. *Ordonnancement Temps Réel*. Hermès Editions, 2000.

[DG99] R. Devillers and J. Goossens. General response time computation for the deadline driven scheduling of periodic tasks. *Fundamentae Informaticae*, 40((2-3)) :199–219, nov-dec 1999.

[EHS98] A. Ermedahl, H. Hansson, and M. Sjödin. Response-time guarantees in atm networks. *Procedings of RTSS'98*, 1998.

[GGH97] J. C. Palencia Gutiérrez, J. J. Gutiérrez Garcia, and M. Gonzales Harbour. On the schedulability analysis for distributed hard real-time systems. *Euromicro RTS'97*, 1997.

[GGH98] J. C. Palencia Gutiérrez, J. J. Gutiérrez Garcia, and M. Gonzales Harbour. Best-case analysis for improving the worst case schedulability test for distributed hard real-time systems. *Euromicro, RTS'98*, 1998.

[GM97a] L. George and P. Minet. A fifo worst case analysis for a hard real-time distributed problem with consistency constraints. *In procedings of IEEE. Int. Conf. on Distributed Computing Systems*, pages 441–448, 1997.

[GM97b] L. George and P. Minet. A fifo worst case analysis for a hard real-time distributed problem with consistency constraints. *Procedings of the IEEE Int. Conference on Distributed Computing Systems*, 1997.

[Goo99] J. Goossens. *Scheduling of Hard Real-Time Periodic Systems with Various Kind of Deadline and Offset Constraints*. PhD thesis, Université Libre de Bruxelles, Belgium, 1999.

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

[JP86] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5) :390–395, 1986.

[KLS93] D. I. Katcher, J. P. Lehoczky, and J. K. Strosnider. Scheduling models of dynamic priority schedulers. Technical report, Research Report CMU CDS-93-4, Carnegie Mellon University, Pittsburgh, April 1993.

[KRP$^+$93] M. Klein, T. Ralya, B. Pollak, R. Obenza, and M. G. Harbour. *A practitioner's handbook for real-time systems analysis.* Kluwer Academic Publisher, 1993.

[LBCJM95] S. Lefebvre-Barbaroux, C. Chaouiya, and A. Jean-Marie. Real-time scheduling of periodic tasks. *in : Scheduling Theory and its Applications, P. Chretienne, Z. Liu, Wiley*, pages 167–191, 1995.

[LBJMC92] S. Lefebvre-Barbaroux, A. Jean-Marie, and C. Chaouiya. Problèmes d'ordonnancement d'une superposition de flux périodiques sous contraintes temps-réel. Technical Report 1576, INRIA Research Report, 1992.

[Leh90] J. P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. *Procedings of the 11^{th} IEEE Real-Time Systems Symposium*, December 90.

[LL73] 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, 1973.

[LM80] J.Y.-T. Leung and M. L. Merrill. A note on preemptive scheduling of periodic real-time tasks. *Information Processing Letters*, 11(3) :115–118, November 1980.

[LW82] J. Y.T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, 2 :237–250, 1982.

[MJM98] J. M. Migge and A. Jean-Marie. Timing analysis of scheduling policies : a trajectory based model. Technical Report 3561, INRIA Research Report, nov. 1998.

[MJM99] J. M. Migge and A. Jean-Marie. Real-time scheduling : Non-preemption, critical sections and round-robin. Technical Report 3678, INRIA Research Report, April 1999.

[MJMN99] J. Migge, A. Jean-Marie, and N. Navet. Timing analysis of compound scheduling policies : fixed priorities and round robin. Technical report, soumis à Special Issue of the IEEE Procedings Software Engineering on Real-Time Systems, 1999.

[PB97] P. Pedro and A. Burns. Worst-case response time analysis of real-time sporadic traffic in fip networks. *Proc. 9^{th} Euromicro Workshop of Real-Time Systems*, 11-13,june 1997.

[PH98] J. C. Palencia and M. Gonzales Harbour. Scheduling analysis for tasks with static and dynamic offsets. *Proc. IEEE Real-Time System Symposium*, 1998.

[SH98a] M. Sjödin and H. Hansson. Analysing multimedia traffic in real-time atm networks. *Procedings of the 5^{th} IEEE Real-Time Technology and Application Symposium (RTAS'98)*, 1998.

[SH98b] M. Sjödin and H. Hansson. Improved response-time analysis calculations. *RTSS'98*, 1998.

[SL96] J. Sun and J. Liu. Bounding the end-to-end response times of tasks in a distributed real-time system using the direct synchronization protocol. Technical Report RR-96-1949, University Urbana Champaign, 1996.

[Spu96a] M. Spuri. Analysis of deadline scheduled real-time systems. Technical Report 2772, Rapport INRIA, Janvier 1996.

[Spu96b] M. Spuri. Holistic analysis for deadline scheduled real-time distributed systems. Technical Report 2873, Rapport INRIA, Avril 1996.

[SSRB98] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo. *Deadline Scheduling for Real-Time Systems : EDF and related Algorithms*. Kluwer Academic Publishers, 1998.

[Sun97] J. Sun. *Fixed Priority End-To-End Scheduling in Distributed Real-Time Systems*. PhD thesis, University of Illinois at Urbana-Champain, 1997.

[TBW95a] K. W. Tindell, A. Burns, and J. Wellings. Analysis of hard real-time communications. *Real Time Systems*, 9(2) :287–300, 1995.

[TBW95b] K. W. Tindell, A. Burns, and J. Wellings. Calculating controller area network (can) message response time. *Control Engineering Practice*, 3(8) :1163–1169, 1995.

[TBW95c] K. W. Tindell, A. Burns, and J. Wellings. Guaranteeing hard real time end-to-end communications deadlines. Technical report, University of York, 1995.

[TC94] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessors and Microprogramming*, 40, 1994.

[THW94] K. W. Tindell, H. Hansson, and A. J. Wellings. Analysing real-time communications : Controller area network (can). *IEEE*, 1994.

[Tin94] K. W. Tindell. *Fixed Priority Scheduling of Hard Real-Time Systems*. PhD thesis, Univeristy of York, 1994.

[ZB95] S. Zhang and A. Burns. Guaranteeing synchronous message sets in fddi networks. *Proc. 13^{th} IFAC Workshop on Distributed Computer Control Systems*, pages 107–112, Sept. 1995.

Chapitre 5

Annexes

5.1 Algorithme de l'analyse holistique

Initialement les gigues sont toutes nulles. Pour simplifier la présentation de l'algorithme nous considérons que les messages sont des tâches, en relation de précédence. On considère au total n tâches. La convergence de l'algorithme sera plus rapide si les tâches sont parcourues selon l'ordre topologique des relations de précédence. Pour cela, nous supposons que le tableau rang, qui associe à chaque tâche son ordre topologique, a été préalablement calculé. Précisément, soit $\tau_i = rang(k)$ et $\tau_j = rang(l)$, on vérifie alors $\tau_i \prec \tau_j \Rightarrow k < l$.

```
ALGORITHME AH d'Analyse Holistique
/*résolution système d'équation (2.5)*/
Entrée: tableau rang : 1..n       /*ordre topologique des sommets du graphes*/
        matrice mat :1..n,1..n    /*contrainte de précédence*/
Sortie: tableau entier R: 1..n
Var   : tableau entier J: 1..n
Début
  J_i = 0; R_i = 0           1 ≤ i ≤ n
  nb_itération=0;
  Faire
    fini = Vrai
    pour k = 1 à n faire
        i=rang[k]
        R'_i = R_i
        R_i=ResponseTime(i)
        Si R'_i ≠ R^i Alors fini = Faux    /*équation 2.6*/
        Pour j = 1 à n faire
            Si mat[i,j] = 1 Alors          /*j est un successeur de i*/
               J_j = max (J_j, R_i)
            Fin Si
        Fin Pour
        nb_itération=nb_itération + 1
  Tant Que Non fini
Fin
```

5.2 Pire temps de réponse d'une tâche à priorité fixe

```
ALGORITHME ResponseTime_FixedPriority
/*résolution système d'équation 3.10*/
Entrées: B_i, C_i, D_i, T_i, J_i,  1 ≤ i ≤ n
Sortie : R*_i i
Var : r, t, t'
Début
  R*_i = 0; q = 0, a = 0, b = 0
  /*calcul d'une borne inférieure du temps de réponse: équation (3.11)*/
  Pour j = 1 à i − 1 faire
      a = a + C_j/T_j
      b = b + J_j * C_j/T_j
  Fin Pour
  Faire
      t = ((q + 1) * C_i + B_i + b)/a
      Faire
          t' = t
          t = q + 1) * C_i + B_i
          Pour j = 1 à i − 1 faire
              t = t+Ceil((J_i + t')/T_j) * C_(j)
          Fin Pour
      Jusqu'à t = t'
      r = t + J_i − qT_i
      R'_i =max(R'_i; r)
      q = q + 1
  Jusqu'à t > (q + 1)T_i
Fin
```

5.3 Pire temps de réponse d'une tâche à priorité dynamique (EDF)

Nous décrivons maintenant les algorithmes de calcul du temps de réponse d'une tâche, correspondant aux formules présentées. L'implémentation repose sur cinq fonctions calculant respectivement la charge processeur dans la période occupée à l'échéance $d = a + D_i$ (fonction Workload), la longueur de la période occupée associée à la date a (fonction aBusyPeriod), les dates d'activation possibles des instances de τ_i dans la période occupée (fonction TestValues) et enfin le calcul du temps de réponse de τ_i (fonction ResponseTime). Le calcul de la plus grande période occupée L repose sur l'algorithme BusyPeriod. Les paramètres des tâches $(J_i, B_i, C_i, D_i, T_i)$ sont supposés visibles dans ces fonctions, et ne sont pas en conséquence considérés comme des entrées des algorithmes.

```
ALGORITHME BusyPeriod
/résolution du système (3.13)*/
Entrées : C_j;T_j,  1 ≤  j ≤  n
Sortie  : L            /*Longueur de la période d'activité*/
Var : L'
Début
   L' = C_1 + C_2 + C_3 + ... + C_n
  Faire
      L = L'
      L' = 0
      Pour i = 1 à n faire
          L' = L' + ⌈ L/T_i⌉  * C_i
      Fin Pour
  Tant Que (L ≠ L')
Fin


FONCTION TestValues
/*calcul de l'ensemble A: équation (3.34)*/
Entrées : entiers i, L
Sortie  : A
Var : entiers p, q, d
      booléen fini
Début
  q = 0     /*q désigne le numéro de l'instance considérée*/
  fini=Faux
  Tant Que Non fini
          fini=Vrai
          Pour j = 1 à n faire
              d = q * C_j + D_j − D_i − J_i
              Si (d > (−J_i)) Et d < L − C_i − J_i − B_i) Alors
                 inserer(A, d)
                 Fini=Faux
              Fin Si
          Fin Pour
          q = q + 1
  Fin Tant Que
  A =SupprimerDoublons(A)
Fin           /*il n'est pas nécessaire de trier l'ensemble A*/
```

```
FONCTION Workload
/*résolution su système d'équations (3.18)*/
Entrées : entier a,i,t
Sortie  : W
Var : entiers d,δ,s
Début
  d = a + D_i
  W = 0
  Pour j = 0 à n faire
      Si (i ≠ j Et D_j ≤ (d + J_j)) Alors
         W = W+min(⌈(t + J_j)/T_j⌉, 1 + ⌊(d + J_j − D_j)/T_j⌋) ∗ C_j
      Fin Si
  Fin Pour
  s_ia = a + J_i − ⌊(a + J_i)/T_i⌋ ∗ T_i
  Si (t > s )
    Alors
      δ = C_i∗min(⌈(t + J_i − s)/T_i⌉, 1 + ⌊(a + J_i)/T_i)⌋
    Sinon
      δ = 0
      W = W + δ
Fin
```

```
FONCTION Ressource_k
/*résolution du système d'équation (3.30)*/
Entrée : entier d
Sortie : entier k
Var : v
Début
  v = 0
  Pour j = 1 à n faire
      Si (v > D_j − J_j Et d > D_j − J_j) alors
        v = D_j − J_j
        k = j
      Fin Si
  Fin Pour
Fin
```

```
FONCTION aBusyPeriod
/*résolution du système d'équation (3.30)*/
Entrées : entier a, i
Sortie  :  L_ia
Var : entier d, k, s_ia
Début
  d = a + D_i
  k =Ressource_k(d)
  s_ia = a + J_i − ⌊(a + J_i)/T_i⌋ ∗ T_i
  Si s_ia = 0
    Alors
          Lprim = C_i
    Sinon
          Lprim = 0
  Pour j = 1 à n faire
      Si (j ≠ i Et D_j ≤ (d_ik + J_j)) Alors
        Lprim = Lprim + C_j
  Fin Pour
  Faire
        L_ia = Lprim
Lprim =Workload(i, a, L_i)
  Tant Que (L_ia ≠ Lprim)
Fin
```

```
FONCTION ResponseTime
/*Calcul de l'équation (3.38)*/
Entrée : entier i
Sortie : entier R
Var : entier a, L, L_ia
      Liste A
Début
  R = C_i
  L =BusyPeriod()                           /*résoudre (3.13)*/
  A =TestValues(i, L)                       /*calculer (3.34)*/
  Tant Que Non ListeVide(A) faire
      a =ExtraireElement(A)
      L_ia =aBusyPeriod(i, a)               /*calculer (3.30)*/
      R =max(R,max(C_i + B_i + Ji, L_ia − a))   /*calculer (3.31,3.38)*/
  Fin Tant Que
Fin
```