Institut National Polytechnique de Lorraine

Département de formation doctorale en informatique — École doctorale IAE + M

Évaluation de performances temporelles et optimisation de l'ordonnancement de tâches et messages

THÈSE

présentée et soutenue publiquement le 10 Novembre 1999

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

(spécialité informatique)

par

Nicolas NAVET

Composition du jury

Président :	Mr. Yves SOREL
Rapporteurs :	Mr. Francis COTTET Mr. Guy JUANOLE Mr. Michael RUSINOWITCH
Examinateurs :	Mr. Ye-Qiong SONG Mr. Jean-Pierre THOMESSE

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Remerciements

Je remercie Madame Françoise Simonot-Lion, responsable de l'équipe Temps Réel et Interopérabilité (TRIO) du Loria, pour l'accueil dans son équipe et pour avoir toujours veillée à ce que nous puissons travailler dans les meilleures conditions.

Mes travaux depuis le stage de DEA ont été dirigés par Messieurs Ye-Qiong Song et Jean-Pierre Thomesse. Je remercie Monsieur J.-P. Thomesse pour ses conseils et son enthousiasme communicatif. J'exprime toute ma reconnaissance à Monsieur Y.-Q. Song pour sa disponibilité, son écoute et ses conseils qui m'ont toujours été profitables.

Pendant cette thèse, et en dehors de mes encadrants, j'ai eu la chance de pouvoir collaborer avec Messieurs François Simonot, Jörn Migge et Bruno Gaujal. Je remercie particulièrement Jörn Migge et Bruno Gaujal pour le travail que nous avons fait en commun et pour nos discussions sur les sujets les plus variés.

Je tiens à exprimer mes vifs remerciements à :

- Monsieur Francis Cottet, Professeur à l'ENSMA de Poitiers,
- Monsieur Guy Juanole, Professeur à l'Université Paul Sabatier de Toulouse,
- Monsieur Michael Rusinowitch, Directeur de Recherche à l'INRIA,
- Monsieur Yves Sorel, Directeur de Recherche à l'INRIA.

pour m'avoir fait l'honneur de participer à mon jury de Thèse.

Merci à Miguel Leon, mon collègue de bureau, pour ces trois années de cohabitation chaleureuse (et enfumée). Je remercie mes autres collègues et amis, Laurent, Raphael, Paolo, Stéphan, Olivier, Gérald, Fabrice, Gladiz, Luis, Madalina, Joël... pour la bonne ambiance qui a toujours régné et les parties de football endiablées.

À Laurence,
pour sa compréhension
et son soutien.

Table des matières

Introduction générale

Un système informatique dont le bon fonctionnement ne dépend pas uniquement de la correction logique des résultats mais aussi des instants de production de ces résultats est désigné sous le terme générique de *système temps réel* [Stankovic, 1988]. Notons que si l'on s'accorde généralement sur la véracité de cette définition quelque peu minimaliste, la littérature contient une pléthore d'autres définitions[1], qui tout en restant en accord avec Stankovic, insistent plus particulièrement sur certaines caractéristiques du temps réel comme la nécessaire *"prévisibilité"* du système ou l'aspect *réactif* de l'application vis-à-vis de son environnement.
Classiquement, on décompose une application temps réel en deux entités : une *partie opérative* qui est le processus physique effectivement contrôlé et une *partie commande* qui est le système informatique assurant le contrôle de la partie opérative. La partie commande prend régulièrement connaissance de l'état du processus physique par le biais de valeurs provenant de capteurs, calcule la prochaine action à effectuer sur la base des mesures antérieures et du modèle de comportement attendu et répercute ses consignes sur le système contrôlé par le biais d'actionneurs. La partie commande de l'application peut être répartie sur différents sites, par exemple pour être capable de répondre simultanément à plusieurs sollicitations de l'environnement, pour dupliquer des fonctions et ainsi pouvoir pallier d'éventuelles défaillances ou tout simplement parce que la topologie du processus contrôlé l'impose. La coopération est alors assurée par l'échange de données sur un réseau de communication. A chaque instant, les valeurs de ces données constituent une image de l'état du processus commandé ainsi que de l'application elle même.
Les contraintes de temps pesant sur les activités d'une application temps réel sont issues de la dynamique du processus physique contrôlé et ces contraintes temporelles devront être prises en compte tout au long du cycle de vie de l'application. Classiquement le cycle de vie d'une application se décompose en six étapes qui sont la rédaction du cahier des charges, la spécification, la conception, l'implantation, l'intégration et l'exploitation (cf. figure 1). Les travaux présentés dans cette thèse portent sur les phases de conception et d'exploitation du système.

1. Plus d'une dizaine de définitions différentes sont répertoriées dans [Vega, 1996].

1 Prise en compte du temps dans le cycle de vie de l'application

Les contraintes de temps sont présentes sous une forme plus ou moins formelle dès la phase d'analyse des besoins, c'est à dire lors de l'élaboration du cahier des charges. Par exemple, si l'on considère un système de contrôle aérien, une contrainte plausible est que *"la position des avions sur l'écran radar devra être mise à jour toutes les 100ms"*.
La vérification des propriétés temporelles commence effectivement lors de la phase de spécification dont l'objectif est la définition d'une *architecture fonctionnelle* validée, c'est à dire sans ambiguïtés, sans incohérences et complète [Bayart and Simonot-Lion, 1995]. L'Architecture Fonctionnelle (ou AF) décrit formellement la structure d'une application et son comportement attendu, c'est à dire l'ensemble des services et traitements respectivement à fournir et effectuer par les fonctions ainsi que les échanges de données inter-fonctions. L'AF ne tient compte ni de la répartition des fonctions sur les sites ni de l'Architecture Matérielle (AM) support définie comme l'ensemble des composants qui assurent les ressources matérielles et logicielles nécessaires à l'exécution de l'application (e.g. machines, OS, réseaux, protocoles, ...). L'indépendance de l'AF vis-à-vis du support exécutif permet d'une part de réduire la complexité du problème général et d'autre part de faciliter la réutilisation d'AFs existantes. Néanmoins, dans cette étape du cycle de vie, une validation ne peut être menée qu'en faisant des hypothèses fortes sur les durées de traitement et donc implicitement sur les performances. Les vérifications sur l'AF portent dont essentiellement sur la correction "logique" du système, comme par exemple, l'absence d'interblocage, la terminaison des traitements ou l'équité dans l'accès à une ressource.

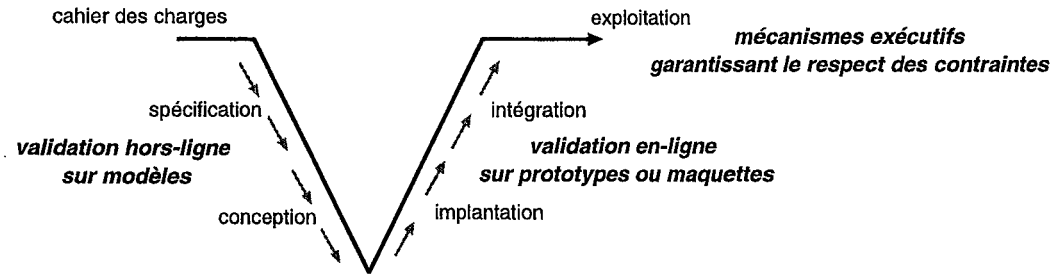


FIG. 1 – *Situation de nos travaux dans le cycle de vie d'une application temps réel.*

Une Architecture Opérationnelle (AO) validée (cf. figure 2) est l'objectif attendu de la phase de conception du système. L'AO est le résultat de la projection (*mapping*) de l'AF sur l'AM support, les composants de cette dernière étant choisis relativement à un ensemble de critères issus du cahier des charges parmi lesquels les performances évidemment, le coût, qui pour les productions de masse revêt une importance toute particulière, mais aussi par exemple, des

garanties sur la pérennité de la production. La projection comprend le placement des fonctions sur les différents sites mais aussi la fixation des paramètres des mécanismes exécutifs associés à l'implantation, comme les ordonnanceurs de tâches (*scheduler*) et les protocoles de communication. Une fois réparties sur les différents sites, les fonctions pourront être subdivisées en différentes tâches auxquelles il faudra vraisemblablement allouer une priorité et éventuellement une politique d'ordonnancement si l'OS support permet un choix[2]. De façon similaire, selon le protocole de transmission de données, il peut être nécessaire de fixer la priorité des trames échangées, priorités qui contrairement à celles des tâches, seront globales sur l'ensemble du système sous l'hypothèse d'un médium de communication partagé. Il s'agit ensuite de s'assurer que l'AO proposée vérifie les contraintes de l'application, c'est la phase cruciale de validation de l'AO. Il est évident que la remise en cause éventuelle de choix effectués en termes d'AM, de répartition ou de fixation de paramètres sera plus facile et coûteuse lors de l'étape de conception que plus en aval dans le cycle de vie de l'application et particulièrement pendant l'exploitation du système.
Une fois la conception terminée commence la phase d'implantation qui consiste d'une part à programmer les fonctions ou tâches et d'autre part à câbler physiquement les connexions entre sous-systèmes. Vient ensuite la phase d'intégration sur site puis naturellement l'exploitation du système assortie épisodiquement d'activités de maintenance.

2 Le contexte

Il existe une grande variété de systèmes temps réel; cela va de l'application centralisée mono-processeur pour laquelle les caractéristiques des tâches (durée d'exécution, instants d'activation et d'utilisation des ressources partagées ...) et les contraintes associées sont parfaitement identifiées jusqu'au système distribué avec trafic (tâches, messages) dynamique à caractéristiques et contraintes variables. Dans cette étude, nous nous situons dans un contexte intermédiaire dans lequel nous distinguons deux classes de trafic :

- du trafic dont les caractéristiques et contraintes sont parfaitement identifiées dès la phase de conception du système pour lequel nous nous efforcerons de donner des garanties fermes et/ou probabilistes sur le respect des contraintes,
- du trafic dynamique pour lequel nous envisagerons des solutions d'ordonnancement de type "best-effort". Nos connaissances sur ce trafic étant généralement parcellaires lors de la phase de conception du système, il nous faudra proposer des solutions qui fassent peu d'hypothèses sur ses caractéristiques.

2. Ainsi le standard temps réel Posix1003.1b [(ISO/IEC), 1996], étudié dans les chapitres 3 et 4, spécifie trois politiques d'ordonnancement de tâches.

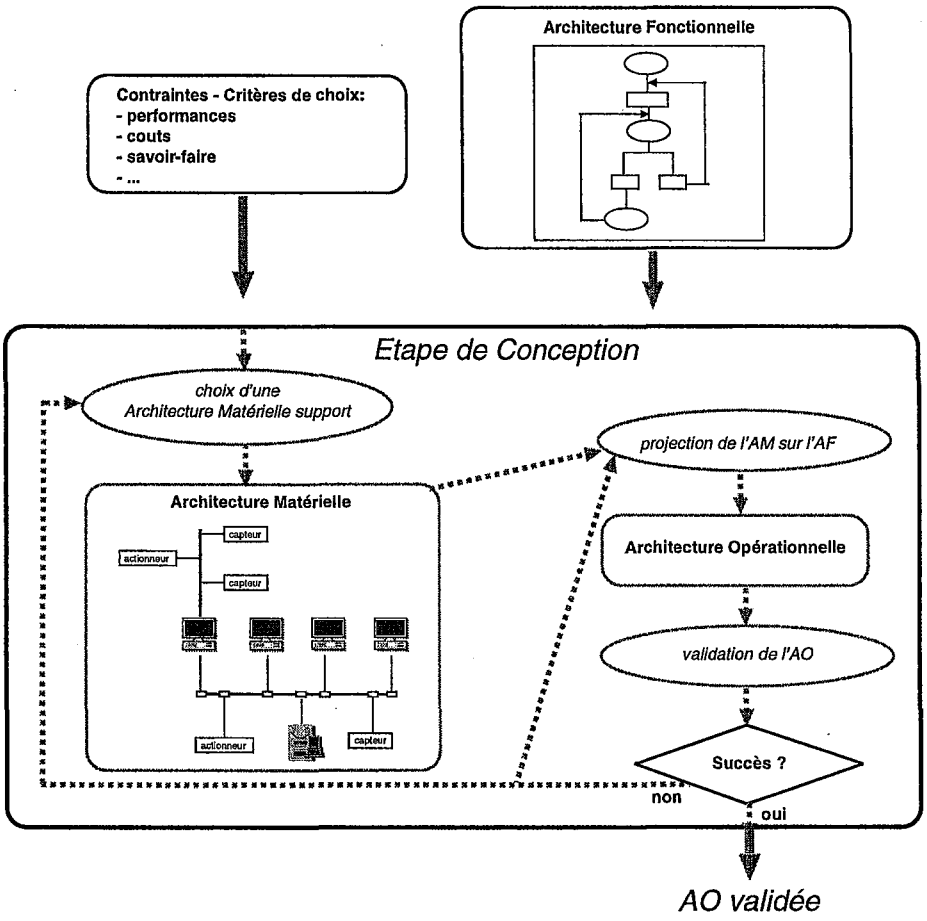


FIG. 2 – *Étape de conception.*

Comme cela a été déjà précèdemment évoqué, les contraintes de temps pesant sur les applications temps réel sont issues du processus physique sous-jacent. Plaçons nous par exemple dans le cadre de l'automobile en considérant les deux organes majeurs que sont une Boite de Vitesse (BV) et le Contrôle Moteur (CM) reliés par un réseau de communication. Après un changement de vitesse, initié par le conducteur mais détecté sur le site BV, le contrôle moteur doit effectuer un certain nombre d'ajustements comme par exemple procéder à une réduction de couple. Le délai entre l'occurrence de l'évenement "changement de vitesse" et la réduction effective du couple doit toujours rester inférieur à une certaine valeur (fixée par les concepteurs du véhicule en fonction du grand nombre d'impératifs dont évidemment des contraintes mécaniques). Il s'agit en l'occurrence d'une contrainte stricte de date au plus tard que nous qualifierons de contraintes sur un *temps de réaction* ($\tau_{reaction}$, cf. figure 3), qui se décompose en le *temps de réponse de bout-en-bout du système informatique* (R_I) plus le *temps*

de réponse de la partie opérative (R_Ψ), c'est à dire le délai entre l'application des consignes (e.g. fermeture d'un clapet) sur le site CM et la baisse effective de couple au niveau requis. Nous nous intéresserons plus particulièrement au délai R_I, R_Ψ se situant objectivement en dehors de notre domaine de compétences. Néanmoins pour vérifier le respect de la contrainte associée à $\tau_{reaction}$, il faudra nécessairement avoir la connaissance d'une borne sur R_Ψ qui nous sera fournie par des spécialistes de la partie opérative.

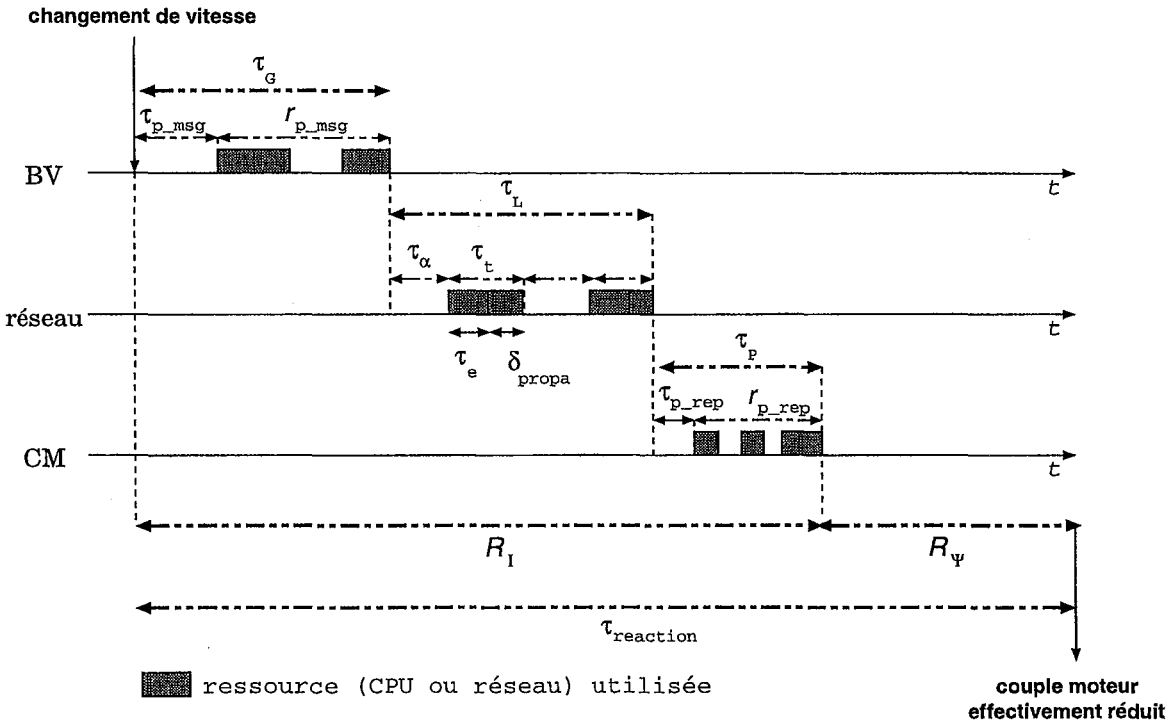


FIG. 3 – *Décomposition du temps de réaction.*

D'un point de vue implémentation, imaginons qu'après un changement de vitesse une tâche devienne active sur le site BV; elle aura, entre autres charges, d'émettre un message à la fin de son exécution à destination du CM pour l'informer de la situation courante. Sur le site CM, après réception du message, la tâche qui devra éventuellement agir sur les actionneurs, se déclenche. On peut maintenant décomposer cette contrainte de temps de réponse de bout-en-bout du système informatique (cf. figure 3) en :

- le délai de production du message τ_G : c'est le délai entre le changement de vitesse et la fin d'exécution de la tâche productrice du message destiné au CM, il se décompose lui même en :
  1. le temps de prise en compte de l'événement, noté τ_{p_msg}. D'un point de vue implémentation, deux stratégies sont possibles : (1) la tâche est périodique et τ_{p_msg}

est alors égal à la différence entre la prochaine activation de la tâche et la date du changement de vitesse, (2) la tâche est déclenchée sur l'occurrence d'un événement, par exemple une interruption matérielle (signalant le changement de vitesse), et τ_{p_msg} sera donc le temps de prise en compte de cet événement.

2. le temps de réponse de la tâche productrice, noté r_{p_msg} fonction essentiellement de la durée d'exécution de la tâche, des autres tâches du site et de la, ou des, politique(s) d'ordonnancement(s).

- le temps de latence du message, noté τ_L qui est le délai entre la fin d'éxécution de la tâche productrice et la réception complète du message par le site consommateur, en l'occurrence le site CM. Le message sera segmenté en n trame(s), où n est fonction du protocole de communication et de la taille des données. Le temps de latence du message est dépendant du protocole de communication utilisé mais si l'on considère par exemple un bus à priorité tel que CAN [ISO, 1994b], quasi standard de fait pour l'industrie automobile, chaque trame i sera reçue par le site CM après un délai égal à la somme du temps requis pour gagner l'accès au bus (τ_α^i) et du temps de transmission effectif de la trame (τ_t^i). Ce temps de transmission étant classiquement composé du temps d'émission des bits composant la trame (τ_e^i) et du délai de propagation δ_{propa}. On a donc pour un bus à priorité $\tau_L = \sum_{i=1}^{n} \tau_\alpha^i + \tau_t^i$ avec $\tau_t^i = \tau_e^i + \delta_{propa}$.
- le délai de production de la réponse, noté τ_P, qui est le délai entre l'arrivée complète du message et la fin d'exécution de la tâche consommatrice du message qui appliquera les consignes sur le moteur. Comme pour la tâche productrice, outre le temps de réponse de la tâche (r_{p_rep}), on considérera un délai (τ_{p_rep}) entre la fin d'arrivée du message et l'activation effective de cette tâche.

On identifie naturellement d'autres contraintes temporelles que celle de date au plus tard. En particulier, on peut citer les contraintes de date au plus tôt et de simultanéité sur des occurrences d'événements (le lecteur pourra se référer à [Toussaint, 1997] pour une liste plus exhaustive). Néanmoins, la contrainte de date au plus tard est au coeur de la problématique du temps réel puisqu'elle apparaît même explicitement dans de nombreuses définitions des systèmes temps réel sous le terme d'échéances (*"deadlines"*). Cette contrainte de date au plus tard sera au centre de nos préoccupations dans la suite de ce document.

Selon les conséquences du non-respect d'une contrainte, celle-ci peut être qualifiée de contrainte stricte (*hard real-time*) ou souple (*soft real-time*). Notre préoccupation sera généralement de minimiser, autant que faire ce peut, le temps de réponse du trafic à contraintes souples tout en garantissant le respect des contraintes strictes.

3 Les objectifs

Nos travaux se situent en aval de la phase de spécification du système; les fonctions, les flux de données entre fonctions ainsi que les contraintes temporelles associées sont supposés clairement identifiés pour la partie temps réel stricte du système. L'objectif de cette thèse est double, il s'agit d'une part pour nous de proposer des méthodes pertinentes d'un point de vue scientifique et néanmoins pragmatiques, c'est à dire directement applicables dans un contexte industriel, pour concevoir et valider une AO. Notre deuxième objectif est de mettre au point des mécanismes exécutifs assurant le respect des contraintes de temps pendant l'exécution de l'application ou optimisant le comportement du système relativement à des métriques de performances choisies. Ces deux objectifs sont étroitement liés. En effet, l'étape de validation de l'AO permet de mettre en évidence les insuffisances des solutions existantes en identifiant certains "goulots d'étranglements" qui amoindrissent les performances du système, ce qui nous amène à envisager de nouvelles solutions.

3.1 Conception et validation d'AO

Au cours de l'étape de conception, les contraintes de coûts et de temps, nous imposent généralement de travailler sur des modèles du système réel dès que celui-ci est un tant soit peu complexe. En recoupant les définitions de [Minsky, 1965] et de [Popper, 1973], on peut définir *un modèle comme une idéalisation du système réel nous permettant d'apprendre quelque chose d'utile sur son fonctionnement.* En l'occurrence, nous nous attacherons à vérifier, en exploitant le modèle, que l'application vérifie bien ses contraintes et le cas échéant modifierons l'AO envisagée. Ce processus itératif de modélisation - exploitation du modèle - modification de l'AO se poursuit itérativement jusqu'à ce que l'on obtienne une AO validée ou, dans le cas défavorable, jusqu'à ce que l'on dresse un constat d'échec qui nous obligera à revenir en amont dans le cycle de vie, à la phase de spécification ou même au cahier des charges pour éventuellement relaxer certaines contraintes.
Les formalismes de modélisation sont extrêmement nombreux mais en règle générale on peut dire que plus leur pouvoir d'expressivité est grand, plus leur exploitation sera problématique; il suffit pour s'en convaincre de penser au langage naturel. Dans cette thèse, nous distinguerons les modèles de simulation, c'est à dire les modèles qui se "résolvent" par simulation, et les modèles analytiques dont la résolution est purement calculatoire.
L'utilisation de la simulation dans l'objectif d'évaluer les performances d'une AO, et donc sa capacité à respecter les contraintes de l'application, est une pratique extrêmement répandue qui s'impose généralement faute d'alternatives :

- des mesures sur un système réel nécessite sa réalisation et son appareillage ce qui est

long et coûteux,

- les modèles analytiques ne sont, en règle générale, pas capable de refléter toute la complexité d'une AO : ils ne peuvent modéliser que des systèmes relativement peu complexes ou des sous-ensembles de systèmes complexes et sont donc généralement intrinsèquement insuffisants.

Si la simulation est d'un emploi plus aisé que la collecte de mesures sur un système existant ou la modélisation analytique, nous avons identifié un certain nombre de difficultés qui ont fait l'objet de nos études dans le cadre du DEA [Navet, 1995] et les premiers mois de cette thèse :

- À partir d'un certain niveau de complexité du système à modéliser, il est nécessaire d'envisager une segmentation en sous-modèles. Par exemple, il est à nos yeux difficilement envisageable de modéliser un profil de communication à 7 couches, la machine support ainsi que l'application d'un même bloc. Une fois la nécessité de modularité admise, il nous a fallu nous interroger sur la meilleure façon de rendre "inter-opérable" les sous-modèles. Outre une meilleure maîtrise de la complexité, cette segmentation en sous-constituants présente des avantages parmi lesquels une validation[3] plus aisée et plus rigoureuse de chacun des constituants et la possibilité de réutiliser les composants.
- L'exploitation de modèles de simulation par un utilisateur n'ayant pas la connaissance des techniques de modélisation ni du langage de simulation est problématique, c'est pourquoi nous avons développé dans le cadre du stage de DEA un logiciel proposant une interface conviviale qui permet la construction graphique de l'AO à partir d'une bibliothèque de modèles élémentaires et la génération du code de simulation correspondant. Ce logiciel appelé "Arcome Edit" pour "éditeur d'architectures de communication" a été utilisé dans l'étude du profil de communication MMS/TCP/Ethernet [Navet *et al.*, 1995; Song *et al.*, 1996] dont l'emploi dans des applications temps réel était, à l'époque de ces travaux, considéré comme une solution envisageable [Lefebvre *et al.*, 1995]. Une approche similaire a été développée plus tard dans l'équipe dans le cadre du stage de DEA de Pierre Bélissent [Bélissent, 1996]; la spécification de l'AO étant cette fois donnée sous forme textuelle.

La nécessaire modularité et donc la définition de techniques d'interconnexion de modèles est une problématique de recherche qui avait été étudiée préalablement à nos travaux en particulier dans [Nachef, 1992] et qui était parallèlement étudié dans le cadre d'une thèse se déroulant dans notre équipe [Lecuivre, 1996].

Ces travaux sur la modélisation, nous ont confortés dans l'idée que généralement la simulation

3. Appliquée à un modèle de simulation, la validation consiste en la vérification de la correction de l'implémentation et de la pertinence du modèle.

seule n'est pas, pour les applications temps réel, une technique de vérification suffisante, et ce à plusieurs titres :

1. Il pèse toujours une incertitude forte sur la correction de l'implantation d'un modèle et sur sa pertinence et donc sur la validité même des résultats de simulation.
2. La simulation d'une AO fait nécessairement des hypothèses simplificatrices sur le système physique contrôlé. Ainsi, le modèle de l'AO du système de contrôle-commande embarqué dans un véhicule intègre un modèle, généralement simpliste, du comportement des différents organes du véhicule.
3. Les résultats dans le pire des cas (e.g. temps de réponse maximum) n'ont qu'une valeur indicative puisque simuler consiste à dérouler un certain nombre de trajectoires qui dans le cas général, est infiniment inférieur au nombre de trajectoires possibles.

Dans une optique temps réel, le dernier argument en défaveur de la simulation est le plus fort puisque, avant de penser aux performances dans le cas probable, c'est d'abord le pire cas, même si il ne survient qu'exceptionnellement, qui nous intéresse. Il nous est donc apparu naturel de "coupler" les modèles de simulation avec des modèles analytiques donnant des bornes sur les métriques de performance considérées. Cette approche a été implantée pour la première fois au sein de notre équipe dans l'atelier de construction et de validation des systèmes embarqués dans l'automobile VACANS (VAlidation of CAN based Systems) qui a fait l'objet d'une diffusion commerciale par la société DELTA PARTNERS et a été utilisé dans un contrat industriel [Navet and Song, 1996] portant sur la validation d'une application embarquée dans un véhicule prototype de la société PSA (Peugeot-Citroën Automobiles). Initialement [Bélissent, 1996], cet outil se constituait d'un modèle de simulation de profils de communication CAN couplé à un calcul analytique de bornes sur le temps de latence des messages développé à l'Université de York [Tindell and Burns, 1994b; Tindell *et al.*, 1995]. Nous l'avons enrichi en développant deux modèles analytiques visant à évaluer la "robustesse" d'une application dans un environnement soumis à des perturbations électromagnétiques pouvant engendrer des erreurs de transmissions. Le premier modèle permet le calcul de la probabilité de non-respect des échéances des messages dans le pire cas, le second, le temps moyen d'atteinte de l'état "bus-off" (c'est à dire la déconnexion d'une station du bus, cf. section 1.2.5). Ces deux informations n'auraient pu être obtenues par simulation; la première est une borne supérieure donc non évaluable par simulation, l'atteinte de l'état bus-off quant à lui, sous des hypothèses d'erreurs réalistes, est un évenement trop rare pour que des simulations de durées raisonnables donnent des résultats statistiquement valables.

Les résultats que l'on peut obtenir en résolvant des modèles, qu'ils soient analytiques ou de simulation, ne sont valables qu'à la condition que ses modèles soient pertinents et que les hypothèses faites sur le système soient respectées. Par l'observation d'un prototype ou d'une

maquette du système, il est possible, dans une certaine mesure, de valider les modèles, de fixer les valeurs de certains paramètres et surtout de vérifier que certaines hypothèses sont respectées. Ainsi, dans le cadre des applications embarquées dans les véhicules, un observateur réseau est indispensable ne serait-ce que pour s'assurer que les différents organes électroniques du véhicule, qui sont majoritairement conçus et fabriqués par des équipementiers, respectent leur spécification en termes de trafic réseau généré. D'autre part, les modèles sont généralement très grossiers quant à la partie opérative du système et le respect de certaines contraintes, que nous qualifierons de contraintes de niveau applicatif et dont l'exemple typique est la contrainte de temps entre entre un changement de vitesse et la réduction de couple associée, n'est pas vérifiable. Toutes ses raisons nous ont conduits à développer, dans le cadre d'un second contrat industriel avec PSA [Song and Navet, 1997], un observateur réseau "intelligent" capable de vérifier en-ligne le respect de certaines contraintes de niveau applicatif que l'on aura spécifiées au préalable.

Toujours cette optique de coupler des informations obtenues à l'aide de techniques différentes pour mieux appréhender le système, mais cette fois dans le cadre de l'ordonnancement de tâches sur des systèmes d'exploitation se conformant au standard Posix1003.1b, nous avons développé une approche utilisant conjointement une analyse d'ordonnançabilité, qui donne des garanties dans le pire cas, à la simulation qui fournit des informations plus riches et permet ainsi d'optimiser le système relativement à des critères autres que la faisabilité. En effet, d'une façon générale, il existe plusieurs solutions d'ordonnancement qui assurent le respect des contraintes strictes d'une application et il faut donc se poser la question du choix de la solution la mieux adaptée à l'application considérée. Nous donnerons des éléments de réponse à cette question, d'une part en définissant des critères permettant de choisir entre différentes configurations faisables et d'autre part, en proposant une technique pour parcourir l'espace des solutions faisables.

3.2 Mécanismes exécutifs

Nous nous plaçons dans le contexte de l'ordonnancement de messages avec du trafic à contraintes strictes et du trafic à contraintes souples. Dans cette seconde partie de la thèse, nous traiterons du problème de l'ordonnancement de ces deux types de trafic avec des objectifs différents : 1) garantir les échéances pour le trafic à contraintes strictes et 2) minimiser le temps de réponse moyen pour le trafic à contraintes souples. Si ce problème a été très largement étudié dans le contexte de l'ordonnancement de tâches (cf. section 5.1 pour un état de l'art), à notre connaissance peu de solutions efficaces existent pour l'ordonnancement de messages. Notre objectif sera d'améliorer des politiques existantes et de proposer des solutions innovantes. Outre une évaluation quantitative des performances par simulation, nous nous attacherons

également à prouver formellement l'efficacité des solutions d'ordonnancements étudiées.

De nombreux réseaux locaux sont amenés à fonctionner dans des environnements soumis à des perturbations électromagnétiques qui peuvent engendrer des erreurs de transmission. Typiquement, ce problème se pose dans toute son acuité pour les applications de multiplexage véhicule. L'environnement d'utilisation, par ses effets sur le comportement de certains composants du système (ici, essentiellement sur le médium de transmission), ne permet plus d'envisager une prévisibilité absolue. Dans ses conditions, il nous paraît raisonnable de se tourner vers des garanties probabilistes sur la qualité de service que nous exprimerons en termes de probabilité de respect des échéances. Toutes les applications n'ayant pas des exigences identiques en matière de sûreté de fonctionnement, il est nécessaire que le concepteur de l'application puisse spécifier le niveau de qualité de service requis. Le niveau de perturbations électromagnétiques pouvant varier grandement au cours du temps, il nous faudra également envisager que les politiques d'ordonnancement s'adaptent "en-ligne" aux conditions courantes d'utilisation.

4 Organisation du document

La première partie de ce document est composé de deux chapitres : le premier est consacré au réseau CAN et à son utilisation dans le domaine du multiplexage véhicule et le second à la validation d'applications temps réel distribuées autour d'un réseau CAN. Dans la seconde partie, nous étudierons l'ordonnancement de tâches sous Posix1003.1b en proposant d'une part une analyse d'ordonnançabilité (chapitre 3) et d'autre part, une approche basée sur une technique d'optimisation pour choisir au mieux les politiques d'ordonnancement et les priorités d'un ensemble de tâches (chapitre 4). Enfin en troisième partie, nous traiterons du problème de l'ordonnancement conjoint de trafic à contraintes strictes et souples avec deux approches, une à priorités dynamiques très efficace mais nécessitant des modifications au niveau du contrôleur de communication (chapitre 5), et l'autre, à priorités fixes, qui s'est revélée sensiblement moins performante mais qui est utilisable sur des composants standards (chapitre 6).

Première partie

Le Réseau CAN dans les Systèmes Temps Réel Embarqués

Chapitre 1

CAN, un réseau conçu pour le multiplexage véhicule

Introduction

L'apparition de l'électronique dans les véhicules dès la fin des années 1970 a créé de nouveaux besoins en communication entre les différentes organes d'un véhicule. L'ajout de nouvelles liaisons "point-à-point" entre équipements est devenue de plus en plus problématique pour des raisons de maintenance, de poids et surtout de coûts (la longueur totale des câbles de certains véhicules produits dans les années 80 est supérieure à 2km pour un poids de plus de 100kg [Lawrenz, 1997]). C'est pour répondre à ce besoin nouveau de communication multiplexée, c'est à dire utilisant un médium de transmission commun, que le protocole de communication CAN (Controller Area Network, cf. [ISO, 1994b] et [ISO, 1994a]) a été développé à partir de 1983 par l'équipementier automobile Allemand Bosch. Bien que conçu spécifiquement pour l'industrie automobile, CAN a été paradoxalement tout d'abord beaucoup plus employé dans le domaine des systèmes automatisés de production. Dans ce chapitre, nous présenterons tout d'abord une application typique de multiplexage véhicule avec ses contraintes de temps et de sûreté de fonctionnement (SdF). Nous détaillerons ensuite les mécanismes de fonctionnement du réseau CAN.

1.1 Les systèmes électroniques embarqués dans l'automobile

Dans cette première section, nous décrirons une application typique de multiplexage véhicule, domaine dans lequel CAN est devenu un standard de fait. Une application embarquée dans un véhicule est composée de d'un certain nombre de sites (calculateurs, capteurs, actionneurs) reliés par un ou plusieurs réseaux de communication. Les fonctions (ou tâches) s'exécutent en parallèle sur les différents sites, la synchronisation se faisant par des échanges

de messages. Un tel système doit respecter un certain nombre de contraintes temporelles et de sûreté de fonctionnement (SdF); un exemple typique est celui d'un changement de rapport de vitesse qui doit provoquer une réduction du couple moteur dans une fenêtre temporelle bornée (cf. paragraphe 2.3). La vérification du respect des contraintes d'un système embarqué dans l'automobile est primordiale car les conséquences du non-respect de ces contraintes peuvent être graves voire catastrophiques vis-à-vis du bon fonctionnement du véhicule et donc de la sécurité de ses passagers. C'est pourquoi, la recherche de méthodes/outils pour cette validation suscite de nombreux travaux de recherche tant dans le milieu industriel qu'universitaire.

1.1.1 Une application multiplexée typique

La tendance actuelle chez les constructeurs automobile (PSA, BMW, SAAB, MERCEDES...) dans la conception de systèmes de multiplexage véhicule est d'interconnecter deux sous-réseaux :

- un réseau pour le contrôle temps réel du véhicule, ou réseau "moteur", avec des stations telles que le contrôle moteur, la boite de vitesse, le capteur d'angle du volant et l'ABS (*Anti-lock Braking System*). Le débit de transmission est généralement supérieur ou égal à 250 kbit/s avec CAN comme protocole de communication.
- un réseau pour le transfert de données dans l'habitacle du véhicule (appelé réseau "confort"). Le débit de ce réseau est généralement moins élevé ($\leq$125kbit/s). Le protocole utilisé n'est pas forcément CAN mais peut être le J1850 [SAE, 1996] (normalisé par le SAE - *Society of Automotive Engineers*, et utilisé par les constructeurs américains) ou aussi VAN (*Vehicle Area Network*, normalisé par l'ISO et l'AFNOR [Association Française de Normalisation-AFNOR, 1990] et utilisé par PSA).

Le problème principal est celui du respect des contraintes temporelles des messages échangés sur le réseau "moteur" (angle du volant, état des ABS, régime moteur..) car le non-respect de ces contraintes peut avoir des conséquences graves sur le fonctionnement du véhicule et donc sur la sécurité de ses passagers. Au contraire, les contraintes pesant sur les échanges dans le réseau habitacle (ex: climatisation) sont généralement plus lâches et les conséquences de leur non-respect relativement mineures.

Dans le second chapitre, nous nous proposons d'étudier une messagerie[4] composée de 12 trames périodiques. Ses principales caractéristiques sont consignées dans le tableau 1.1 où le DLC (*Data Length Code*) est la taille du champ de données de la trame. Les 6 sites participant aux échanges sont le contrôle moteur, le capteur d'angle du volant, la boite de vitesses automatique,

4. Dans son acception "ensemble de messages" et donc sans référence à la couche 7 du modèle OSI.

l'ABS, la suspension et le calculateur carrosserie (interface avec le réseau habitacle de type VAN). Le débit du bus est de 250kbit/s, l'échéance d'une trame est égale à sa période et nous considérerons qu'il n'y a pas de gigue en émission (variations dans les instants de mise à disposition des messages, cf. paragraphe 2.2.1). Notons que pour des raisons de sécurité, une même information peut provenir de différentes sources, par exemple, la vitesse courante du véhicule peut être émise par le contrôle moteur et par l'ABS. Cette messagerie nous a été fournie par PSA (Peugeot-Citroën Automobiles) et a été implantée sur un véhicule prototype. À notre connaissance un seul autre "benchmark" du domaine du multiplexage véhicule a été publié dans [SAE, 1993], il s'agit de la messagerie d'un prototype de véhicule électrique.

Priorité	Site émetteur	DLC	Période
1	contrôle moteur	8	10 ms
2	capteur angle volant	3	14 ms
3	contrôle moteur	3	20 ms
4	boite vitesse	2	15 ms
5	ABS	5	20 ms
6	ABS	5	40 ms
7	ABS	4	15 ms
8	calcul. carrosserie	5	50 ms
9	suspension	4	20 ms
10	contrôle moteur	7	100 ms
11	boite vitesse	5	50 ms
12	ABS	1	100 ms

TAB. 1.1 – *Messagerie de l'application PSA.*

1.1.2 Contraintes de Temps et de SDF

Comme pour tous les systèmes de contrôle-commande, la partie commande d'une application embarquée dans un véhicule prend régulièrement connaissance de l'état du processus physique par le biais de valeurs provenant de capteurs, calcule la prochaine action à effectuer sur la base des mesures antérieures et sur la base du modèle de comportement attendu et répercute ses consignes sur le système contrôlé par le biais d'actionneurs. On distingue généralement deux techniques de diagnostic d'un procédé qui peuvent par ailleurs coexister dans un même système: par échantillonnage périodique (*time triggered*) et par événement (*event triggered*). Ces deux techniques de diagnostic génèrent respectivement un trafic périodique et un trafic apériodique. Par la nature même des informations, les messages à transmettre sont généralement de durée de vie limitée. Ce qui veut dire du point de vue du système informatique que le temps de réponse d'un message (défini comme l'intervalle de temps entre l'activation de

la tâche qui produit le message et la réception complète de celui-ci par le ou les destinataires, cf. paragraphe 2.2.1) doit être borné.
Des contraintes plus complexes existent au niveau applicatif. Ainsi, la réponse à un stimulus doit intervenir dans une fenêtre temporelle bornée. Un exemple typique de contraintes de niveau applicatif est le délai entre un changement de vitesse (le stimulus) et la réduction de couple moteur associée (réponse au stimulus - cf. figure 2.8). Une grande partie des organes d'un véhicule n'étant pas conçue par le constructeur automobile lui même mais par divers équipementiers, une vérification du respect de ces contraintes par modélisation n'est que difficilement envisageable, la solution est alors la vérification par observation sur un prototype ou sur une plate-forme d'essai (cf. paragraphe 2.3)
Selon les conséquences du non-respect des contraintes (la criticité), on distingue des contraintes temps réel strictes (HRT - *Hard Real-Time*) et des contraintes dites souples (SRT - *Soft Real-Time*). Un système de communication doit fournir une garantie absolue vis-à-vis des temps de réponse des messages HRT et minimiser, autant que possible, celui des messages SRT. Dans la première partie de cette thèse, nous ne traiterons pas de la minimisation des temps de réponse du trafic SRT et renvoyons le lecteur à la troisième partie ainsi qu'à [Tindell and Hansson, 1995; Gaujal and Navet, 1999b; Gaujal *et al.*, 1999a; Navet and Song, 1999c; Navet and Song, 1999b].

Une application embarquée étant critique du point du point de vue de la sécurité, les contraintes de SdF doivent être identifiées et leur respect vérifié durant la phase de conception. Parmi les nombreuses contraintes de SdF, nous nous focaliserons dans cette étude sur la fiabilité de la transmission car d'une part l'environnement d'utilisation peut être extrêmement hostile en termes de perturbations d'origine électromagnétique et d'autre part, des contraintes économiques très fortes pèsent sur le choix du support physique de transmission (généralement la paire torsadée non-blindée, cf. figure 1.2). Le lecteur intéressé par une étude détaillée de la SdF dans les systèmes embarqués dans l'automobile pourra consulter [Ziegler *et al.*, 1994].

1.1.3 Facteurs perturbant le respect des contraintes temporelles et de SdF

En considérant que seuls les messages périodiques ou sporadiques ont des contraintes de temps strictes (HRT), on peut effectivement déterminer une borne sur le temps de réponse pour chaque niveau de priorité [Tindell and Burns, 1994b; Tindell and Burns, 1994a] (cf. paragraphe 2.2.1). Mais comme nous l'avons précédemment évoqué, les applications embarquées dans les véhicules sont souvent amenées à fonctionner dans des environnements soumis à de fortes perturbations électromagnétiques (*electromagnetic interference* - EMI) ce qui peut avoir des conséquences sur la sécurité des passagers. Le *Wall Street Journal*

(édition du 8 septembre 1997, cité dans [The Risks Digest, 1997a]) et le *NCR Handelsblad* (édition des 25 et 26 septembre 1997, cité dans [The Risks Digest, 1997b]) relatent des accidents impliquant des véhicules, ayant entrainés des blessures graves et le décès de plusieurs personnes et causés, selon toutes vraisemblances, par des EMI. Ces EMI [Noble, 1992; Zanoni and Pavan, 1993] peuvent provenir d'équipements électriques internes au véhicule (interrupteurs, contacts aimantés, ...) ou de sources extérieures (radio, radar, ...). Si les EMI peuvent affecter l'ensemble des équipements électroniques du véhicule (ou ECU - *Electronic Control Units*), nous nous focaliserons par la suite sur le médium de transmission qui est un composant du système particulièrement sensible. Dans des conditions de perturbations électromagnétiques données, la fréquence des erreurs de transmission dépend fortement du type de support utilisé; des mesures ont ainsi montré qu'une paire torsadée non blindée est 6 fois plus sensible aux EMI qu'une paire torsadée blindée (cf. figure 1.2). L'utilisation d'un réseau "tout optique" qui offre une très haute immunité aux EMI n'est à l'heure actuelle pas envisageable compte tenu des contraintes de coût imposées par l'industrie automobile.

Dans le protocole CAN, une trame entachée d'une erreur n'est pas considérée par les stations du réseau, elle sera retransmise le plus tôt possible par la station émettrice (cf. paragraphe 1.2.4). L'"overhead" induit par une erreur est le temps nécessaire pour transmettre la trame jusqu'au bit sur lequel on a détecté l'erreur plus la trame d'erreur, plus enfin toutes les trames plus prioritaires arrivées depuis la tentative initiale de transmission (en cas de retransmission sur l'occurrence d'une erreur, la trame garde sa priorité initiale). Cet overhead peut amener la trame à ne pas respecter l'échéance temporelle qui lui est associée.

Par ailleurs, des ECUs, et notamment ceux fournis par des équipementiers, peuvent ne pas respecter les hypothèses que l'on a faites sur le trafic qu'ils génèrent (variation de périodicité, de taille de messages, de priorité, transmission de trames non répertoriées, ...). Ces noeuds, appelés "babbling idiots" dans [Tindell and Hansson, 1995], modifient les caractéristiques du trafic et peuvent conduire le système à ne pas respecter le comportement temporel prédit par des études analytiques ou des simulations.

1.2 Le réseau CAN

Le réseau CAN est un bus à diffusion, avec accès au médium priorisé et résolution des collisions non-destructive. Les stations ne possèdent pas d'adresse propre et aucune d'entre elles ne joue un rôle prépondérant dans le protocole. Toutes les trames contiennent un identificateur, unique sur le système, qui a pour fonction de définir la priorité de transmission (plus l'identificateur est petit, plus la trame est prioritaire) et de permettre de filtrer les mes-

sages en réception. Les données (éventuellement segmentées en plusieurs trames) peuvent être transmises périodiquement, sporadiquement ou à la demande (RTR - *remote transmission request*).

Un profil CAN minimum est constitué d'une architecture à trois couches: physique, liaison de données (LdD) et application (cf. figure 1.1). La couche liaison de données est implantée sous forme d'un composant électronique appelé contrôleur de communication. De nombreuses sociétés (Intel, Philips, Siemens, Motorola, Nec, ...) se partagent ce marché important (11 millions de contrôleurs vendus en 1996 [PZ Marketing, 1997]) en forte croissance (122 millions d'unités prévus pour 2000 [PZ Marketing, 1997]). Les documents [ISO, 1994b; ISO, 1994a] ne normalisent que la couche physique et liaison de données mais plusieurs propositions ont été faites pour la couche application (CAN Application Layer - CAL [(CiA), 1995]) ou pour des profils complets intégrant les deux couches normalisées (DeviceNet [Allen-Bradley, 1994; IEC TC17B - WG3, 1998a], SDS [CENELEC, 1997; IEC TC17B - WG3, 1998b], et CANopen [(CiA), 1996] qui utilise un sous-ensemble de CAL).

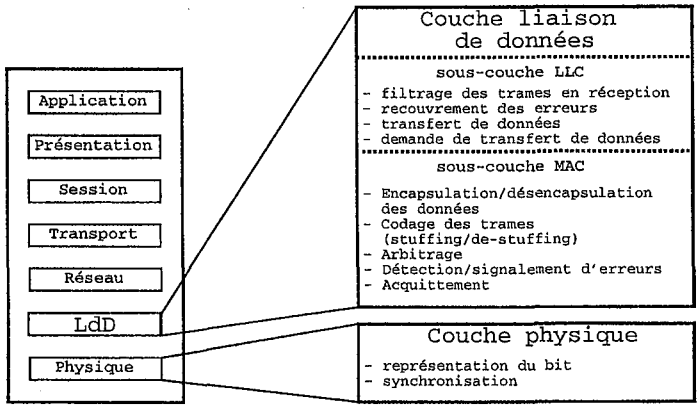


FIG. 1.1 – *Principales fonctions de la couche LdD et de la couche physique de CAN.*

Les principales fonctions et services assurés par les sous-couches *Medium Access Control* (MAC) et *Logical Link Control* (LLC) de la couche LdD ainsi que ceux de la couche physique sont répertoriés sur la figure 1.1. Le lecteur intéressé par le réseau CAN pourra consulter [Paret, 1996; Paret, 1999; Kiencke and Kytölä, 1996; Lawrenz, 1995; Lawrenz, 1997; ISO, 1994b; ISO, 1994a] et s'abonner à la liste de diffusion[5] CAN.

5. http://www.scruz.net/ ~cichlid/can-archive/maillist.html

1.2.1 Couche physique de CAN

Les normes ISO spécifient deux supports de transmission sur paires différentielles pour le multiplexage automobile: CAN "low-speed" [ISO, 1994b] et CAN "high-speed" [ISO, 1994a] avec un débit allant de quelques kbit/s à 1Mb/s. La méthode d'accès au médium utilisée (cf. paragraphe 1.2.2) impose une relation entre débit maximum et la longueur physique du médium; à titre d'exemple, un débit de 1Mb/s ne peut être atteint que sur une distance d'au maximum 40m et un débit de 5kbit/s sur 10km [Paret, 1996]. Un support physique autre que la paire différentielle (fibres optiques, ondes infrarouges ...) est envisageable à condition:

1. qu'il soit capable de véhiculer des bits ayant deux niveaux: *dominant* et *récessif.*
2. qu'il se comporte comme une porte logique *ET* avec les niveaux dominants et récessifs correspondant respectivement aux valeurs logiques 0 et 1. Ainsi, si une station émet un bit *dominant* et que simultanément une autre station émet un bit *récessif*, le bus doit se trouver dans l'état *dominant.*

La sensibilité d'un réseau CAN aux perturbations électromagnétiques dépendra naturellement des conditions d'utilisation mais aussi des caractéristiques de la couche physique: valeur et emplacement des résistances de fin de ligne et qualité du support physique. Ainsi, des mesures [Barrenscheen and Otte, 1997] ont montré que dans les mêmes conditions d'utilisation, la fréquence des erreurs de transmission étaient 6 fois plus importante avec la paire non-torsadée/non-blindée qu'avec la paire torsadée/blindée (cf. figure 1.2). Notons que bien qu'ayant de meilleures caractéristiques intrinsèques qu'un support mono-filaire, la paire torsadée non-blindée, telle qu'utilisée par certains constructeurs automobile, est loin d'assurer une protection absolue des trames contre des perturbations électromagnétiques.

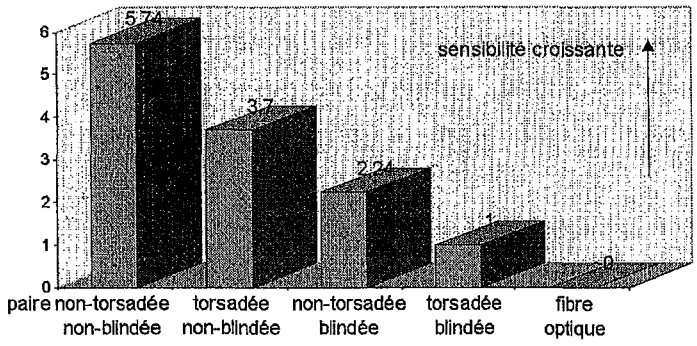


FIG. 1.2 – *Sensibilité relative de différents supports aux perturbations électromagnétiques (chiffres adaptés de [Barrenscheen and Otte, 1997]).*

La technique d'encodage des bits utilisée est le *Non-Return-to-Zero* (NRZ) avec un *bit-stuffing* de longueur 5. Le *bit-stuffing* consiste à insérer 1 bit de niveau opposé tous les m bits successifs de même niveau (où m est la longueur du *bit-stuffing*), elle a pour principal intérêt de créer artificiellement des fronts sur le signal et ainsi de permettre une resynchronisation des stations. Le *bit-stuffing* s'applique sur la trame jusqu'au champ d'acquittement (cf. figure 1.5). Dans le pire cas (cf. figure 1.3), le *bit-stuffing* augmente la taille de la trame CAN de $\lfloor (n-1)/4 \rfloor$ bits où n est le nombre de bits jusqu'au délimiteur de CRC (non-inclus, cf. figure 1.5).

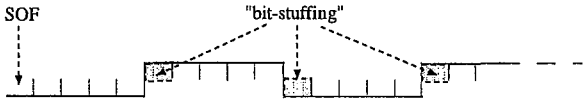


FIG. 1.3 – *Bit-stuffing - le pire cas.*

1.2.2 Résolution des collisions

La phase de résolution des collisions sur le bus, appelée arbitrage, suit le principe de la procédure *CSMA/CA*[6] (Carrier Sense Multiple Access/Collision Avoidance). L'arbitrage est fait sur l'identificateur (cf. figure 1.5) et sur le bit RTR. Dans l'hypothèse où plusieurs stations émettent simultanément, la trame de plus forte priorité (i.e. avec le plus petit identificateur) gagnera l'accès au bus. En effet, le bus se comportant comme une porte logique ET et la valeur binaire de l'identificateur étant transmise du bit le plus significatif au moins significatif, toutes les autres stations détecteront, chacune à leur tour, qu'ayant émis un bit récessif, le niveau du bus est dominant. Elles stopperont alors leur transmission et attendront que le bus redevienne libre pour tenter une réémission. Le fait que l'émetteur d'un bit doit pouvoir avoir un retour d'information pendant la durée même de ce bit impose une contrainte forte sur le débit de transmission. En effet, le signal doit avoir le temps d'effectuer un aller-retour sur toute la longueur du bus pendant un temps bit qui sera donc nécessairement plus grand que deux fois le délai de propagation. Notons qu'un contrôleur de communication ayant plusieurs messages en attente d'émission effectue un arbitrage interne pour ne présenter sur le bus que le message le plus prioritaire.
Sur la figure 1.4, la station 1 détecte la perte d'arbitrage lorsqu'elle "écoute" sur le bus un bit dominant alors qu'elle a émis un bit récessif. À cet instant, elle sait qu'une autre station émet une trame de plus petit identificateur et stoppera son émission.

6. Si Kopetz (cf. [Kopetz, 1997] page 161) par exemple parle de CSMA/CA, la phase d'arbitrage de CAN est souvent décrite comme suivant la procédure CSMA/AMP (Arbitration on Message Priority), cf. [Lawrenz, 1997] page 87, et plus rarement CSMA/CR (Collision Resolution), cf. [Intel Corporation, 1999].

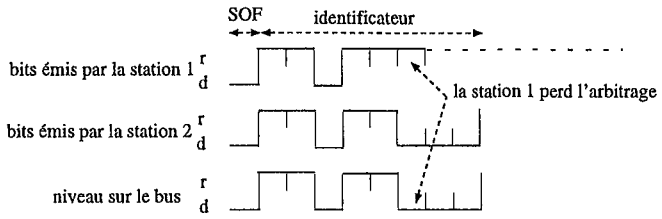


FIG. 1.4 – *Arbitrage "bit-à-bit" - un exemple.*

1.2.3 Format de la trame CAN

Les normes définissent deux formats de trames (cf. figure 1.5) : le *CAN 2.0A* (ou CAN standard) avec un champ identificateur de 11 bits le *CAN 2.0B* (ou CAN étendu) avec un champ identificateur de 29 bits. Le nombre de trames dans une application embarquée ne justifiant pas l'emploi du CAN étendu, nous considérerons par la suite que les trames suivent le format "standard" qui permet de coder 2032 identificateurs[7] . La trame CAN se compose des champs suivants :

- début de trame (*start of frame* - SOF): 1 bit dominant qui indique le début d'une trame,
- arbitrage: 11 bits pour l'identifieur plus le bit RTR qui est récessif[8] pour une trame de demande de données (dite aussi trame de requête) et dominant pour un trame de données simples. Une trame de requête ne comporte pas de données et aura le même identifieur que la trame de données correspondante. Son utilité est de demander l'émission d'une donnée sur le réseau par la station productrice (coopération de type client/serveur),
- contrôle: 2 bits réservés pour permettre des extensions (notamment le CAN étendu) et 4 bits pour la taille des données (*data length code* - DLC),
- données: de 0 à 8 octets,
- champ de CRC (*cyclic redundancy code*) : 15 bits et 1 bit délimiteur,
- acquittement (*acknowledgment* - ACK): 1 bit d'acquittement et 1 bit délimiteur. Le bit d'acquittement est émis au niveau récessif puis, chaque station ayant reçu la trame sans erreur, superposera au bit récessif initial, un bit dominant. Notons qu'un acquittement correct signifie simplement qu'une station au moins sur le bus a reçu correctement la trame mais il n'assure absolument pas que le consommateur de la donnée l'a effectivement reçu,

7. au lieu des $2^{11} = 2048$ identificateurs attendus et ce, pour rester compatible avec un des premiers contrôleurs CAN, l'Intel 82526, qui n'est pas capable de gérer les identificateurs dans la plage 2033-2048 .

8. Ce qui implique que pour un identificateur donné, une trame de données gagnera l'arbitrage sur une trame de demande de données.

- fin de trame (*end of frame* - EOF): 7 bits récessifs,
- inter-émission (*interframe*): 3 bits récessifs.

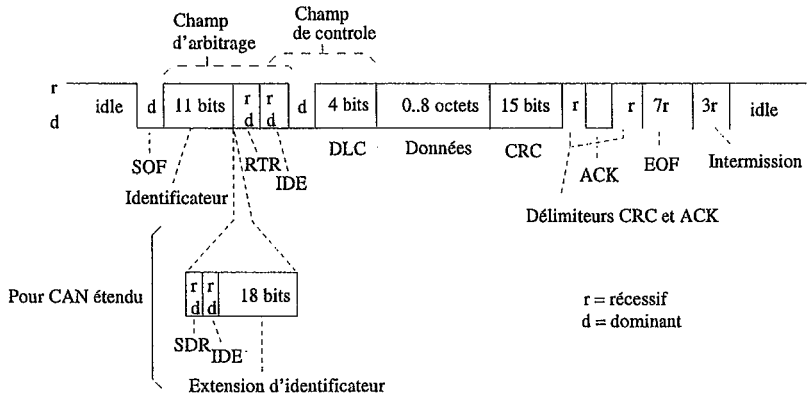


FIG. 1.5 – *Format des trames* CAN 2.0A *et* CAN 2.0B *au niveau du* MAC.

La taille maximale de la trame CAN standard en tenant compte du *bit-stuffing* est de 135 bits[9] (cf. équation (2.3)) et son rendement maximum (8 octets de données et pas d'overhead causé par le bit-stuffing) est de 57.6%.

1.2.4 Traitement des erreurs de transmission

Le protocole CAN n'utilise pas de technique de correction automatique des erreurs de transmission. Lorsqu'une station détecte une erreur dans une trame en cours d'émission, elle le signale à l'ensemble des autres stations, puis la trame corrompue participera automatiquement (i.e. sans intervention de la couche applicative) au prochain arbitrage une fois le bus redevenu libre. Néanmoins rien n'assure que la trame corrompue gagnera immédiatement le bus, elle peut perdre l'arbitrage interne à la station ou l'arbitrage sur le bus. Cinq types d'erreurs sont détectés sur un réseau CAN :

- erreur du niveau d'un bit (*bit error*): la valeur du bit émis est différente du niveau sur le bus (sauf pendant les phases d'arbitrage et d'acquittement),
- erreur de bit-stuffing: 6 bits consécutifs de même niveau,
- erreur du format d'un champ (*form error*): détection d'une valeur non conforme dans un champ fixé par le protocole (champ délimiteurs, EOF et inter-émission),

9. Etrangement, la taille maximum de la trame CAN semble toujours avoir posée des problèmes aux chercheurs du domaine, par exemple dans [Tindell and Burns, 1994b] elle est de 130 bits, dans [Paret, 1999] elle est de 138 bits alors que dans [Lawrenz, 1997] elle est successivement de 130 bits (page 95) puis de 135 bits (en appliquant la formule (6.1-2) page 255). A notre connaissance, la valeur de 135 bits, qui nous paraît exacte, a été publiée pour la première fois dans [Tindell and Hansson, 1995].

- erreur de CRC: le CRC reçu dans la trame est différent du CRC calculé,
- erreur d'acquittement (*ACK error*): le bit d'acquittement reçu par l'émetteur est récessif, l'acquittement n'a pas eu lieu.

Une station ayant détecté une erreur, le signale par l'émission d'un drapeau d'erreur (*error flag*) constitué de 6 bits dominants consécutifs. Ce drapeau d'erreur provoquera inévitablement soit une *form error* soit une erreur de bit-stuffing, de sorte que toutes les stations sur le bus détecteront l'erreur et émettront à leur tour un drapeau d'erreur. Après les différents drapeaux d'erreur (éventuellement superposés, cf. figure 1.6), la trame d'erreur se poursuit par un délimiteur d'erreur (8 bits récessifs) et par un champ inter-émission (3 bits récessifs). Au total, la trame d'erreur comportera de 17 à 23 bits[10]. Une étude réalisée par la société Bosch [Unruh *et al.*, 1989] a montré que la probabilité de non détection d'une erreur, ou erreur résiduelle, est extrêmement faible: de l'ordre de 10^{-13}. C'est pourquoi nous ferons l'hypothèse dans le second chapitre que toutes les erreurs sont correctement détectées.
La figure 1.6 montre la détection d'une erreur dans le cas le plus favorable, c'est à dire le cas dans lequel l'erreur est connue de toutes les stations seulement 6 bits après son occurrence permettant à la retransmission d'intervenir après 17 bits. La figure 1.7 illustre le pire cas, 23

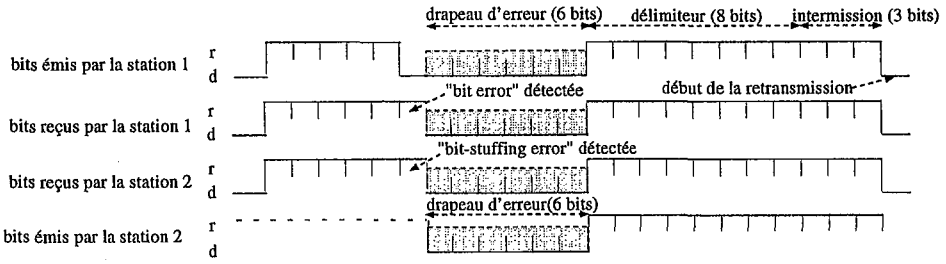


FIG. 1.6 – *Détection d'erreur: le cas le plus favorable.*

bits sont nécessaires pour la détection de l'erreur par toutes les stations.
Si CAN assure avec une très forte probabilité que le contenu d'une trame ne sera pas falsifié sans que cela ne soit détecté, il est possible qu'une même trame soit reçue deux fois par certaines stations. Ce problème est causé par une incohérence temporelle entre les instants auxquels les stations réceptrices et la station émettrice considèrent une trame comme étant non-erronée. En effet, une station réceptrice considère une trame comme correcte après l'avoir acquittée alors que la station émettrice continue à vérifier sa correction jusqu'au dernier bit

10. 31 bits dans des circonstances exceptionnelles: la trame corrompue a été émise par une station "erreur passive" (cf. paragraphe 1.2.5) qui, de par son état, doit attendre 8 bits supplémentaires (*suspend transmission flag*), d'autre part toutes les autres stations du réseau n'ont pas de trame à transmettre jusqu'au début de la retransmission de la trame corrompue.

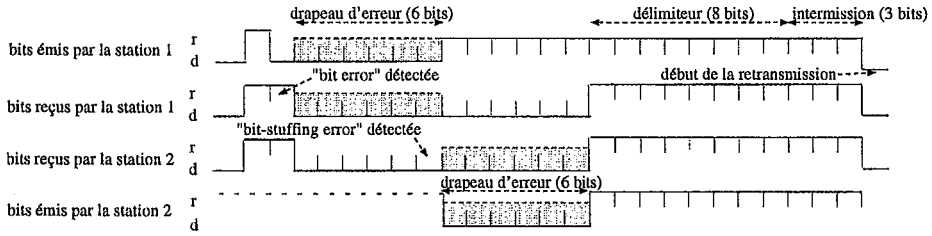


FIG. 1.7 – *Détection d'erreur: le pire cas.*

de la trame. Si une erreur se produit après l'acquittement, l'émetteur du message le signalera et retransmettra la trame alors que les stations réceptrices ont déjà considéré le message comme correct. De ce fait, les stations réceptrices peuvent se voir délivrer deux fois le même message. Les trames CAN doivent donc absolument ne comporter ni des données représentant des commutations d'états ("toggle messages") ni des données relatives (ex : incrémentation de 15 unités).

1.2.5 Confinement des erreurs : les états des stations

Pour éviter qu'une station défectueuse ne perturbe le bon fonctionnement du système, par exemple par l'émission répétée de trames d'erreurs, le protocole CAN intègre des mécanismes de "confinement" d'erreurs dont l'objectif est : 1) de détecter des dysfonctionnements d'origine matériels et 2) d'isoler les stations défectueuses. Pour cela, les contrôleurs de communication possèdent deux compteurs d'erreurs de transmission distincts :

- le compteur d'erreurs en émission (*transmit error counter* où TEC) qui compte les erreurs intervenues sur les trames que le station émet.
- le compteur d'erreurs en réception (*receive error counter* où REC) qui compte les erreurs intervenues sur les trames que le station reçoit.

Chaque fois qu'une trame est correctement reçue ou transmise, le compteur associé est décrémenté, de la même façon, chaque fois qu'une erreur de transmission est détectée, le compteur associé est incrémenté. En fonction de la valeur des compteurs, la station se trouvera dans un des trois états prévus par le protocole (cf. figure 1.8) :

- *Erreur active* (REC<128 et TEC<128) : La station émet et reçoit normalement. C'est l'état par défaut à l'initialisation de la station.
- *Erreur passive* (REC>127 ou TEC>127) : La station peut émettre mais doit attendre 8 bits supplémentaires (*suspend transmission flag*) après la fin de transmission de la dernière trame. D'autre part, la station ne pourra plus émettre le drapeau d'erreur de 6 bits dominants consécutifs en cas de détection d'une erreur.

- *bus-off* (TEC>255) : La station se déconnecte du bus et ne pourra se reconnecter qu'après une réinitialisation (*normal mode request*) et après avoir observé 128 occurrences de 11 bits récessifs consécutifs, une séquence de 11 bits récessifs consécutifs correspondant aux champs ACK, EOF et inter-émission d'une trame de donnée dont la transmission n'a pas été entachée d'erreurs.

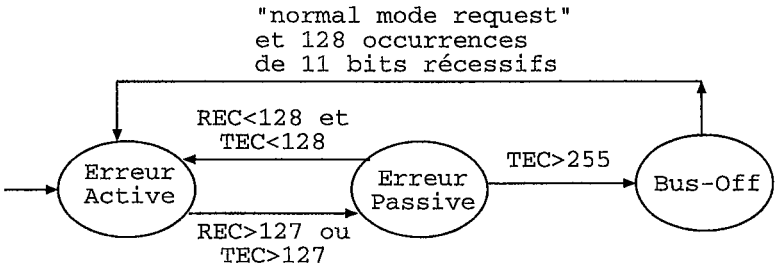


FIG. 1.8 – *Conditions des changements d'états.*

Les règles d'incrémentation et de décrémentation des compteurs TEC et REC sont relativement complexes (cf. [ISO, 1994b] pages 48-49). Les règles décrites ci-dessous ne prennent pas en compte la possibilité qu'une erreur se produise pendant le signalement d'une erreur (pas d'erreur de niveau d'un bit pendant un drapeau d'erreur). De plus, nous ne considérerons pas deux exceptions aux règles générales qui sont nécessaires pendant la phase de mise en marche du système (cf. [Lawrenz, 1997] page 93) lorsqu'il est possible qu'une seule station soit active. Sous ces hypothèses, les règles sont les suivantes :

1. transmission correcte :
   - Pour toute station en réception : si le REC se situe entre 1 et 127, celui-ci est décrémenté de 1, si sa valeur est 0, le REC reste à 0 et si sa valeur est supérieure à 127, la REC prend une valeur entre 119 et 127.
   - Pour la station en émission : si le TEC est non-nul, celui-ci est décrémenté de 1, sinon il reste à 0.
2. transmission erronée :
   - Pour toute station en réception : le REC est incrémenté de 1 si sa valeur est inférieure à 128, dans le cas contraire, il reste à 128.
   - Pour la station en émission: le TEC est incrémenté de 8.

Notons que quelque soit l'issue d'une transmission, seule la valeur de l'un des 2 compteurs est éventuellement modifiée, les règles précédemment exposées faisant l'hypothèse que l'on ne peut être à la fois émetteur et récepteur d'une même trame.

1.3 Conclusion

Dans ce chapitre introductif, nous détaillons les principes de fonctionnement du réseau CAN et présentons une application typique de multiplexage véhicule qui nous servira à plusieurs reprises de jeu d'essai au cours de cette thèse. Nous nous sommes en particulier intéressé aux contraintes de l'application, qu'ils s'agissent de contraintes temporelles ou plus généralement de contraintes de sûreté de fonctionnement, dont la vérification du respect sera le centre de nos préoccupations dans le second chapitre.

Chapitre 2

Validation d'applications réparties autour du réseau CAN

Résumé

Ce chapitre a pour objet de présenter des approches de validation d'applications temps réel distribuées autour d'un réseau CAN. Le processus de validation est conduit en résolvant des modèles, analytiques ou de simulation, ainsi que par l'observation sur maquettes ou prototypes. Une fois les principes du processus de validation exposés, nous donnerons des éléments de réponse (probabilité de non-respect des échéances, temps d'atteinte de l'état bus-off ...) et les appliquerons sur l'application de multiplexage véhicule présentée dans la section 1.1.1 du chapitre précédent. Nous insisterons particulièrement sur des approches probabilistes bien adaptées à la prise en compte des erreurs de transmission qui représentent le principal aléa sur les communications dans un environnement où les perturbations d'origine électromagnétique sont importantes.

2.1 Processus de validation

Le processus de validation que nous développerons dans cette étude est réalisée sur l'Architecture Opérationnelle (AO) pendant la phase de conception de l'application. L'AO est définie comme la projection de l'architecture fonctionnelle (AF) sur l'architecture matérielle (AM) [Bayart and Simonot-Lion, 1995]. L'AF décrit formellement la structure d'une application ainsi que son comportement attendu c'est à dire l'ensemble des services et traitements respectivement à fournir et effectuer par les fonctions ainsi que les échanges de données inter-fonctions. L'AF ne tient compte ni de la répartition des fonctions sur les sites ni de l'Architecture Matérielle (AM) support définie comme l'ensemble des composants qui assurent les ressources matérielles et logicielles nécessaires à l'exécution de l'application (e.g. machines, OS, réseaux, protocoles ...). Notons que dans le cas d'applications embarquées dans les vé-

hicules, l'AM est soumise à de très forte contraintes économiques. Le résultat attendu de la phase de conception est une AO validée et optimale. Les techniques présentées dans ce chapitre se veulent une aide à la conception et à la validation de l'AO. Dans cette optique, deux approches complémentaires sont utilisées :

- *L'évaluation sur modèles*, modèles analytiques et modèles de simulation.
- *L'évaluation sur prototype*, par observation du comportement d'un prototype.

Dans ce chapitre, nous traiterons de l'évaluation sur modèles analytiques (section 2.2) et sur prototype (section 2.3) qui permettent chacune des vérifications différentes et complémentaires. Nous avons fait le choix de ne pas développer dans le cadre de document de thèse nos travaux portant sur la construction de modèles de simulation et renvoyons à [Navet, 1995; Navet *et al.*, 1995; Navet *et al.*, 1996; Navet, 1996; Song *et al.*, 1996; Navet and Song, 1997b]. Le lecteur intéressé par la simulation d'applications distribuées autour du réseau CAN pourra par ailleurs consulter [Bélissent, 1996; Lawrenz, 1997; Kiencke *et al.*, 1997; Simonot-Lion *et al.*, 1997; Tang *et al.*, 1997; Courrier *et al.*, 1998a].

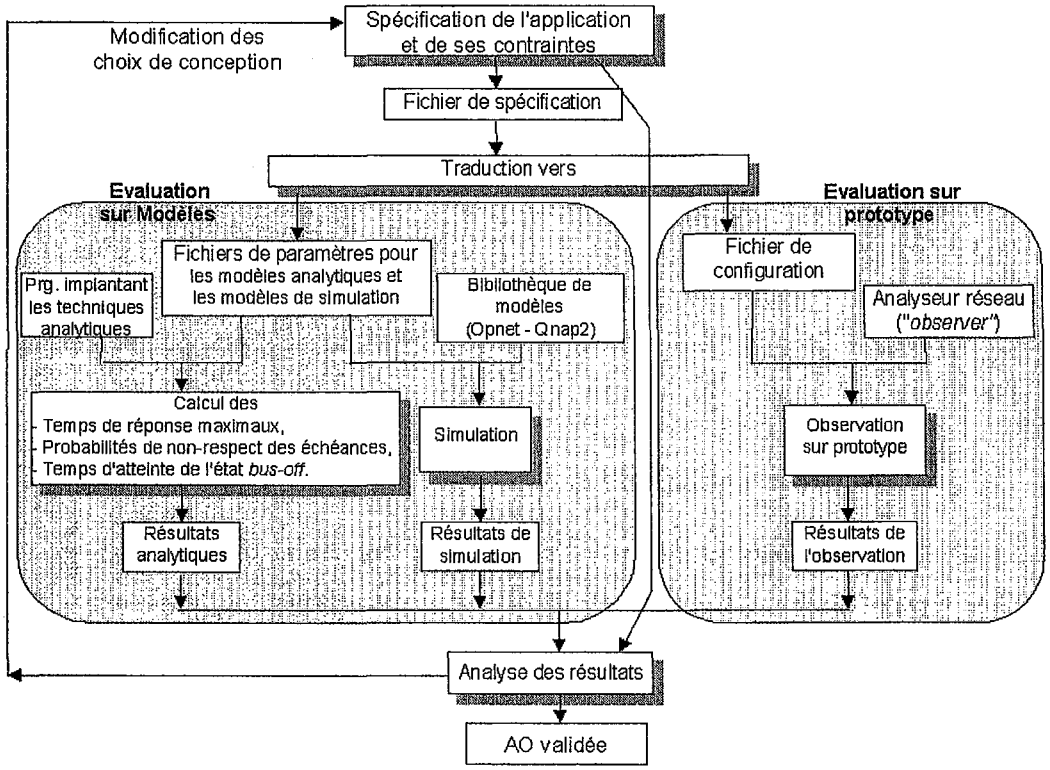


FIG. 2.1 – *Le processus de validation.*

La figure 2.1 représente le déroulement du processus de validation de l'AO. L'application est spécifiée en termes de tâches, flux de données entre tâches (messages) et distribution des tâches

sur les stations. Le lecteur intéressé par le langage de spécification pourra consulter [Bélissent, 1996; Navet and Song, 1997b]. Après avoir vérifié la correction syntaxique et lexicale de la spécification, celle-ci est traduite en fichiers de paramètres et de configuration pour les modèles de simulation (sous Opnet [Bélissent, 1996; Navet and Song, 1997b], sous Qnap2 [Navet, 1996]) , les modèles analytiques et pour l'observateur réseau (cette dernière traduction n'étant pas pour l'instant automatisée). Nous envisageons d'étendre le langage de spécification à la prise en compte des contraintes de temps ce qui permettra une vérification automatique de leur bon respect. Pour l'instant, c'est au concepteur de l'application de vérifier le respect des contraintes et, si nécessaire, il devra modifier ses choix jusqu'à ce que le système remplisse ses objectifs.

2.2 Méthodes analytiques

Les méthodes de validation analytiques présentées dans cette section s'appliquent aux messages périodiques ou sporadiques, dans ce dernier cas, en considérant la période égale au temps minimum inter-arrivée. L'ensemble des messages de l'application est $\mathcal{M} = \{m_1, \cdots, m_p\}$ ou m_k est le message de priorité k. Nous verrons tout d'abord comment obtenir une borne sur le temps de réponse pour chaque niveau de priorité sous l'hypothèse d'un médium de communication fiable. Nous relaxerons cette hypothèse et présenterons comment calculer le temps de réponse maximal avec un nombre d'erreurs borné. Ensuite, à l'aide d'un modèle d'erreurs probabiliste, plus réaliste dans la mesure où il ne fait pas d'hypothèses sur le nombre d'erreurs maximum pouvant survenir dans un intervalle de temps, nous verrons comment évaluer la probabilité de non-respect des échéances. Enfin, nous étudierons comment obtenir le temps moyen d'atteinte de l'état bus-off d'une station ainsi que la variance des temps d'atteinte de l'état bus-off.

Un certain nombre d'hypothèses sont faites sur le contrôleur de communication: (1) il doit être capable de participer à l'arbitrage immédiatement après une émission (et ainsi ne pas laisser le bus à une trame moins prioritaire), (2) si plusieurs trames sont en attente d'émission, le contrôleur doit toujours présenter sur le bus la trame la plus prioritaire pour la phase d'arbitrage. Notons que certains anciens contrôleurs ne comportant qu'un seul buffer en émission étaient connus pour ne pas respecter ces hypothèses, cf. [Tindell and Burns, 1994a] page 6.

Certains contrôleurs, comme le très répandu Intel 82527, ne réservent qu'un buffer en émission par identificateur. Pour éviter qu'une instance d'un message en attente de transmission ne soit "écrasée" sur l'arrivée de l'instance suivante du même message, nous imposons que l'échéance d'un message m_k, notée D_k, ne soit pas plus grande que sa période T_k. Notons que l'analyse présentée par la suite, peut-être étendue à des échéances supérieures aux périodes

en appliquant directement ce qui a déjà été proposé dans le cadre de l'ordonnancement de tâches [Audsley *et al.*, 1993; George *et al.*, 1996].

2.2.1 Temps de réponse maximum avec transmission fiable

Nous définissons dans la suite le temps de réponse d'un message m_k comme le temps entre l'activation (*release*) de la tâche qui produit m_k, notée τ_k, et la réception complète du message par le ou les destinataires. Le message m_k hérite sa période de τ_k mais entre deux mises à disposition successives du message, il se peut qu'il y ait une certaine variabilité (appelée aussi gigue). En effet, le temps entre l'activation de la tâche τ_k et l'invocation de la requête de transfert de données de m_k dépend, par exemple, de l'activation ou non de tâches plus prioritaires sur le processeur du noeud considéré. On note J_k la borne sur la gigue du message m_k. Si l'on n'a pas d'indication sur l'instant auquel τ_k génère m_k au cours de son exécution, par sécurité on peut considérer que m_k n'est disponible qu'à la toute fin de l'exécution de τ_k; J_k est alors égal au pire temps de réponse de τ_k. Si l'exécutif support se conforme au standard Posix1003.1b, et dispose donc des politiques FPP (*Fixed Priority Preemptive*) et Round-Robin, les temps de réponse pourront être dérivés de l'analyse développée dans le chapitre 3.

La durée de transmission C_k d'un message m_k pouvant être bornée (cf. équation (2.3)), pour calculer le pire temps de réponse il nous suffit de calculer le temps maximal que le message m_k devra attendre avant de gagner l'accès au bus (ce temps noté I_k est aussi appelé le temps d'interférence). L'accès au bus peut être bloqué par:

- des messages plus prioritaires, qui peuvent, selon leurs périodes, être transmis une ou plusieurs fois pendant I_k,
- un message moins prioritaire en cours de transmission, et dans le pire des cas, ce temps sera la durée de transmission de la plus longue trame moins prioritaire.

D'après [Tindell and Burns, 1994b], le temps de réponse du message m_k est donc

$$R_k = C_k + J_k + I_k \tag{2.1}$$

avec I_k la limite, quand n tend vers l'infini, de la suite

$$I_k^0 = 0, \quad I_k^n = \max_{i>k}(C_i) + \sum_{j<k} \left\lceil \frac{I_k^{n-1} + J_j + \tau_{bit}}{T_j} \right\rceil \cdot C_j \tag{2.2}$$

et C_j la borne sur le temps de transmission du message m_j comportant d_j octets de données (avec τ_{bit} le temps de transmission d'un bit) :

$$C_j = \left(47 + 8d_j + \left\lfloor \frac{34 + 8d_j - 1}{4} \right\rfloor\right) \cdot \tau_{bit} \tag{2.3}$$

I_k est calculé en partant de $I_k^0 = 0$ jusqu'à convergence ou jusqu'à ce que $I_k^n > D_k - C_k$. Dans ce dernier cas, il n'y a pas de garantie que m_k respecte son échéance et $\mathcal{M}$ est dit non-ordonnançable. La constante τ_{bit} présente dans le deuxième terme de I_k^n est justifiée par le fait que toute trame disponible moins de τ_{bit} après une fin de transmission peut participer à l'arbitrage suivant.

Application Numérique Outre les messages HRT décrit dans le tableau 1.1, nous ferons l'hypothèse que des messages SRT de priorités inférieures et de taille 100 bits sont échangés sur le réseau. L'existence de ces messages SRT doit être considérée dans l'analyse car, l'ordonnancement étant non-préemptif, ceux-ci peuvent retarder la transmission d'un message HRT (cf. premier terme de I_k^n dans l'équation (2.4)). Avec un débit qui est de 250kbit/s et en considérant les gigues sur les dates de mise à disposition des messages comme étant nulles, les temps de réponse des 12 messages HRT sont :

k	1	2	3	4	5	6	7	8	9	10	11	12
R_k	1,04	1,38	1,72	2,02	2,44	2,86	3,24	3,66	4,04	4,46	4,86	5,12

TAB. 2.1 – *Temps de réponse des messages HRT de l'application (en ms).*

Le temps de réponse maximale de chacun des messages étant plus petit que son échéance (égale à la période du message), sous l'hypothèse d'un médium fiable, le respect des contraintes de date au plus tard sur la réception des messages est garanti.

2.2.2 Temps de réponse maximum avec erreurs de transmission

Tindell et Burns dans [Tindell and Burns, 1994a] ont étendu l'analyse du temps de réponse maximum en introduisant la possibilité d'occurrences d'erreurs de transmission. Pour cela, ils proposent le modèle d'erreurs suivant :

- quelque soit le temps t considéré, on a au plus une rafale d'erreurs de taille n_{error},
- mise à part cette rafale, deux erreurs consécutives sont espacées au minimum d'un temps T_{error} (ces erreurs sont considérées sporadiques).

Ce modèle d'erreurs, "déterministe" dans la mesure où le le nombre d'erreurs pendant un temps t peut être borné par $\left(n_{error} + \left\lceil \frac{t}{T_{error}} \right\rceil - 1\right)$, permet le calcul du pire temps de réponse en remplaçant la suite (2.2) par

$$I_k^0 = 0, \quad I_k^n = E_k(I_k^{n-1} + C_k) + \max_{i>k}(C_i) + \sum_{j<k} \left\lceil \frac{I_k^{n-1} + J_j + \tau_{bit}}{T_j} \right\rceil \cdot C_j \qquad (2.4)$$

et E_k la fonction qui prend en compte les overheads induits par les erreurs :

$$E_m(t) = \left(n_{error} + \left\lceil \frac{t}{T_{error}} \right\rceil - 1\right) \cdot \left(23\tau_{bit} + \max_{j \leq k}(C_j)\right) \tag{2.5}$$

L'overhead induit par une erreur est composé d'une part de la trame d'erreur ($23\tau_{bit}$ dans le pire cas considéré, cf. paragraphe 1.2.4) et d'autre part de la retransmission de la trame corrompue. Dans le pire cas, toutes les erreurs se produisent au dernier bit de la plus longue trame susceptible d'être retransmise. En effet, bien qu'une erreur puisse se produire sur la trame moins prioritaire en cours de transmission à l'instant de mise à disposition de m_k, cette trame, compte tenu de l'arbitrage priorisé pour l'accès au bus, ne pourra être retransmise avant la fin de transmission de m_k et l'overhead induit par une telle erreur sera nécessairement moindre que celui induit par une erreur sur une trame pouvant être retransmise (m_j t.q. $j \leq k$).

Application Numérique Dans les mêmes conditions que pour l'application numérique du paragraphe 2.2.1, nous calculons maintenant les temps de réponse avec erreurs de transmission pour $n_{error} = 3$ et $T_{error} = 2{,}5$ms.

k	1	2	3	4	5	6	7	8	9	10	11	12
R_k	3,56	3,90	4,24	4,54	4,96	6,02	6,40	6,82	7,20	8,25	8,65	8,91

TAB. 2.2 – *Temps de réponse des messages HRT de l'application avec erreurs de transmission (en ms) pour* $n_{error} = 3$ *et* $T_{error} = 2{,}5ms$.

On voit sur le tableau 2.2 que même sous des conditions d'erreurs relativement pessimistes, les contraintes sur les temps de réponse des messages sont respectées.

Il nous apparaît néanmoins difficile d'appliquer cette analyse dans le contexte des applications embarquées dans les véhicules. En effet, le taux de perturbation du médium de communication peut varier grandement au cours du temps et nous ne savons pas, en général, comment borner le nombre d'erreurs qui peuvent survenir dans un intervalle de temps. Nous préférons donc explorer une autre voie qui est de modéliser le phénomène aléatoire des erreurs de transmission, par un processus stochastique.

2.2.3 Probabilité de non-respect des échéances dans le pire cas

Considérant d'une part qu'il est irréaliste de faire l'hypothèse d'un médium 100% fiable dans les conditions d'utilisation des applications embarquées dans le véhicules, et étant donnée d'autre part que nous ne savons pas donner une borne sur le nombre d'erreurs pouvant survenir dans un intervalle de temps, nous avons développé une analyse permettant d'évaluer la probabilité qu'une trame ne respecte pas son échéance.

2.2.3.1 Le modèle d'erreurs

Nous proposons un modèle d'erreurs, représenté sur la figure 2.2, qui considère à la fois la fréquence des perturbations (le nombre d'occurrences suit une loi de Poisson) et la gravité des perturbations (lorsqu'une perturbation survient, elle entraîne soit une erreur individuelle, soit une rafale d'erreurs). Dans la pratique, on observe effectivement deux types d'erreurs : des erreurs individuelles (le véhicule se trouve dans une zone qui n'est pas particulièrement perturbée) et des erreurs en rafales (*burst errors*) qui se produisent dans des zones fortement perturbées comme les zones balayées par des radars à proximité des aéroports et les abords immédiats de lignes à haute tension. Les rafales d'erreurs peuvent être appréhendées comme des intervalles de temps pendant lesquels le bus est inaccessible. Notons que ce modèle d'erreurs pourra être également utilisé dans le cadre d'applications de contrôle de processus industriels pour lesquelles des rafales d'erreurs se produisent lors de la mise en marche ou de l'arrêt d'équipements électriques voisins du réseau.

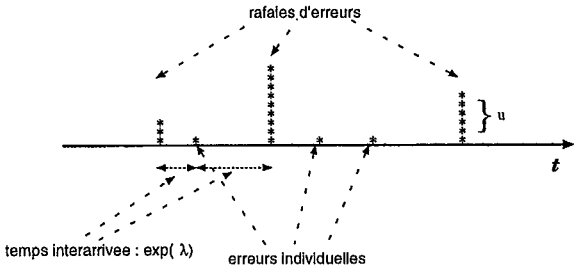


FIG. 2.2 – *Le modèle d'erreurs.*

D'un point de vue mathématique, ce modèle peut être décrit par un processus de Poisson Généralisé (*Generalized Poisson Process*, cf. [Parzen, 1962]). Soit $X(t)$ le nombre d'erreurs de transmission dans l'intervalle $[0, t]$,

$$X(t) = \sum_{i=0}^{N(t)} y_i \qquad \text{avec } y_0 = 0 \tag{2.6}$$

où

- $N(t)_{t\geq 0}$ suit une loi de Poisson de paramètre λ avec $P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}$
- $y_i = \begin{cases} u, \text{ avec une probabilité} & \alpha \\ 1, \text{ avec une probabilité} & 1-\alpha \end{cases}$

avec y_i la variable aléatoire (v.a.) qui donne le type de l'erreur (i.e. erreur individuelle ou rafale d'erreurs) et u la v.a. qui donne la taille des rafales. La distribution de u et la valeur des paramètres α et λ seront fixés à l'aide de mesures collectées sur un prototype ou avec l'expérience accumulée sur des systèmes similaires. Notons que dans cette optique, certains contrôleurs de communication CAN récents possèdent des fonctionnalités intéressantes comme des compteurs d'erreurs accessibles en lecture (contrôleurs NEC, cf. [Hausmann and Gebing, 1997]) ou même la possibilité de générer une interruption sur l'occurrence d'une erreur (contrôleurs Philips et Motorola, cf [Hank, 1997; Motorola LTD, 1997]). De telles fonctionnalités nous permettent même d'envisager des politiques d'ordonnancement de messages qui s'adaptent "en-ligne" au niveau de perturbation sur le bus (cf. section 5.6 pour un premier pas dans cette direction).

2.2.3.2 Seuil d'erreurs tolérables

En partant de l'analyse du pire temps de réponse proposée par Tindell et Burns dans [Tindell and Burns, 1994a] (cf. équation (2.4)), nous proposons une approche différente qui est de calculer pour chaque message de l'application, le nombre maximum d'erreurs tolérables tel que la contrainte associée au message soit respectée. Si l'on note $R_k(i)$ le temps de réponse du message m_k avec i erreurs de transmission, il s'agit de trouver pour chaque message,

$$\eta_k = \max\{n \in \mathbb{N} \mid R_k(n) \leq D_k\}. \tag{2.7}$$

avec

$$R_k(i) = C_k + J_k + I_k(i) \tag{2.8}$$

et $I_k(i)$ la limite, quand n tend vers l'infini, de la suite

$$I_k^0(i) = 0, \quad I_k^n(i) = \mathcal{E}_k(i) + \max_{i>k}(C_i) + \sum_{j<k} \left\lceil \frac{I_k^{n-1} + J_j + \tau_{bit}}{T_j} \right\rceil \cdot C_j \tag{2.9}$$

où $\mathcal{E}_k$ est la fonction qui intègre l'overhead induit par i erreurs de transmission

$$\mathcal{E}_k(i) = i \cdot \left(23\tau_{bit} + \max_{j \leq k}(C_j) \right) \tag{2.10}$$

Pour chaque trame de l'application, η_k se calcule à l'aide de l'algorithme suivant :

$\eta_k := 0;$
while $(R_k(\eta_k) < D_k)$ **do** $\eta_k := \eta_k + 1;$ **od**
$\eta_k := \eta_k - 1;$

Si moins de $\eta_k + 1$ erreurs se produisent entre la mise à disposition de la trame m_k et sa fin de transmission, c'est à dire pendant le temps $R_k(\eta_k)$, alors la trame m_k respectera forcément son échéance. La probabilité de non-respect de l'échéance dans le pire cas (*worst-case deadline failure probability*, cf. [Navet *et al.*, 1999]) est définie comme la probabilité qu'il y ait plus de η_k erreurs pendant le temps $R_k(\eta_k)$

$$P[X(R_k(\eta_k)) > \eta_k] = 1 - \sum_{n=0}^{\eta_k} P[X(R_k(\eta_k)) = n] \tag{2.11}$$

Cette probabilité est qualifiée de probabilité "dans le pire cas" car les deux hypothèses sous-jacentes sont que toute erreur de transmission est détectée sur le dernier bit de la trame (cf. équation (2.10)) et que, à chaque retransmission, le temps nécessaire pour gagner l'accès au bus est le plus grand possible (équation (2.9)).

Application Numérique Pour les 12 trames HRT de notre application et dans les conditions de l'application numérique du paragraphe 2.2.1, le nombre maximum d'erreurs tolérables η_k et $R_k(\eta_k)$ le temps de réponse avec η_k erreurs de transmission sont:

k	1	2	3	4	5	6	7	8	9	10	11	12
η_k	14	19	27	19	25	52	17	61	22	123	58	122
$R_k(\eta_k)$	9.92	13.96	19.70	14.60	19.46	39.42	14.56	49.41	19.54	99.11	49.92	99.92

TAB. 2.3 – *Seuil d'erreurs tolérables et temps de réponse correspondant (en ms).*

Sur le tableau 2.3, on remarque que globalement les trames de l'application peuvent subir un grand nombre d'erreurs de transmission tout en respectant leurs échéances ce qui est un gage de sûreté de fonctionnement. Le nombre d'erreurs tolérables par une trame dépend à la fois de l'échéance de la trame (ici sa période) et de sa priorité. Avec des échéances identiques, un message de forte priorité pourra logiquement subir plus d'erreurs de transmission (cf. message 3,5 et 9). Néanmoins, dans notre application, l'échéance a une plus grande influence que la priorité car le temps de transmission d'un message est petit (< 0.54ms à 250kbit/s) comparé aux différences d'échéances.

2.2.3.3 Calcul de $P(X(t) = k)$

$X(t)$ étant un processus de Poisson Généralisé, il possède une fonction génératrice de la forme $X(z) = E[z^{y(z)-1}]$ (cf. [Parzen, 1962]) et $P[X(t) = k]$ peut être obtenu par dérivation

successive de $X(z)$ à l'aide de la *formule de Taylor* :

$$P[X(t) = k] \doteq \frac{1}{k!} \frac{d^k X(z)}{dz^k}|_{z=0} \tag{2.12}$$

Dans la pratique, la fonction génératrice n'est pas toujours facile à obtenir et ses dérivées peuvent rapidement devenir impossibles à manipuler même avec un logiciel de calcul symbolique comme *Maple*. D'autre part, des problèmes de précisions numériques dans les calculs apparaissent pour des valeurs de k grandes ($k \geq 40$) qui peuvent conduire à des résultats absurdes. Ces difficultés nous ont contraints à envisager une autre méthode de calcul.
D'après la définition du processus $X(t)$ (cf. équation 2.6), on a :

$$P[X(t) = k] = \sum_{m=0}^{k} P(X(t) = k \mid N(t) = m) \frac{e^{-\lambda t}(\lambda t)^m}{m!} \tag{2.13}$$

- si $k = 0$, on a $N(t) = 0$ étant donné que $y_i \geq 1$, donc $P[X(t) = 0] = e^{-\lambda t}$
- si $k \geq 1$, on a $P[X(t) = k] = \sum_{m \geq 1}^{k} P(X(t) = k \mid N(t) = m) \frac{e^{-\lambda t}(\lambda t)^m}{m!}$

Comme $P[X(t) = k \mid N(t) = m]$ est la probabilité pour qu'il y ait k erreurs dans m paquets de taille y_i, et comme $N(t)$ et les v.a. y_i sont indépendantes, si on pose $S_m = y_1 + y_2 + \ldots + y_m$, alors:

$$P[X(t) = k] = \sum_{m=1}^{k} P[S_m = k] \frac{e^{-\lambda t}(\lambda t)^m}{m!} \tag{2.14}$$

S_m étant une chaîne de Markov, on peut écrire:

$$P[S_{m+1} = j] = \sum_{1 \leq i \leq j} P[S_{m+1} \mid S_m = i] \, P[S_m = i] \tag{2.15}$$

on pose $a_k = P(S_1 = k) = P(y_1 = k) \doteq P(y = k)$, d'après l'équation de *Chapman-Kolmogorov* [Parzen, 1962] :

$$P[S_{m+1} = j] = \sum_{i=1}^{j} a_{j-i} \, P[S_m = i] \tag{2.16}$$

On remarque que pour $m > k$, $P[S_m = k] = 0$ car $y_i \geq 1$. L'équation 2.16 nous fournit l'algorithme pour calculer $P[S_m = k]$. Par exemple pour $k = 2$, et en posant $P_m = \frac{e^{-\lambda t}(\lambda t)^m}{m!}$,

on a :

$$\begin{aligned} P[X(t)=2] &= \sum_{m=1}^{2} P(S_m=2)\,P_m \\ &= P(S_1=2)\,P_1 + P(S_2=2)\,P_2 \end{aligned}$$

d'après l'équation 2.16 :

$$\begin{aligned} P(S_1=2) &= a_2 \\ P(S_2=2) &= \sum_{i=1}^{2} a_{2-i}\,P(S_1=i) \\ &= a_1\,P(S_1=1) + a_0\,P(S_1=2) \;=\; a_1\,a_1 \end{aligned}$$

on a donc :

$$P[X(t)=2] = a_2\,P_1 + a_1 a_1\,P_2$$

Les algorithmes implantant cette analyse sont présentés dans les figures 2.3 ($P[S_m = k]$) et 2.4 ($P[X(t) = k]$).

```
1  funct real P_S(integer m,integer k)
2    real r := 0
3    if (m > k)  r := 0;
4  else if (m = 1)  r := a_k;
5    else if (computed[m,k] = −1)
6           for i := 1 to k do
7               r := r + (a_{k−i} · P_S(m − 1,i))
8           od
9           computed[m,k] := r;
10     else r := computed[m,k];
11       fi
12     fi
13   fi
14   return r;
15 end
```

FIG. 2.3 – *Algorithme pour calculer* $P[S_m = k]$, *où* computed[,] *est un tableau à deux dimensions qui sert à sauvegarder les valeurs déjà calculées et dont tous les éléments devront être initialisé à* -1 *avant exécution.*

Cette méthode de calcul de $P[X(t) = k]$ basée sur les chaînes de Markov résout les problèmes de précision dans les calculs et ne nécessite pas la connaissance de la fonction génératrice de $X(t)$. D'autre part, la distribution de y_i peut être donnée sous la forme d'un histogramme

```
1 funct real compute_Xt(integer k,real t,real λ)
2   real r := 0; integer i;
3   for i := 1 to k do
4       r := r + P_S(i,k) · Poisson(i,t,λ);
5   od
6   return r;
7 end
```

FIG. 2.4 – *Algorithme pour calculer* $P[X(t) = k]$ *avec* $Poisson(i,t,\lambda)$ *défini comme* $\frac{e^{-\lambda t}(\lambda t)^i}{i!}$.

de fréquence provenant de mesures effectuées sur prototypes ce qui, d'un point de vue pratique, est très appréciable.

Application Numérique Pour cette application numérique, nous considérerons que la taille des rafales d'erreurs suit la v.a. u, de paramètre p (avec $q = 1 - p$), dont la distribution est

$$P(u = k) = kp^2q^{k-1} \tag{2.17}$$

Nous avons construit par essais successifs cette loi basée sur la distribution géométrique de façon à obtenir $P(u = 0) = 0$ (la taille d'une rafale d'erreurs n'est jamais nulle) et une queue de distribution importante à droite pour prendre en compte des risques extrêmes (cf. [Mandelbrot, 1996]). Les caractéristiques stochastiques de u, y_i et $X(t)$ sont données en annexe A pour u obéissant à cette loi dite "géométrique modifiée", pour $u = 1$ (i.e. pas de rafales d'erreurs) et u suivant une loi uniforme. L'utilisation de la loi uniforme pour u s'impose lorsque l'on possède trop peu de données pour construire un histogramme de fréquences ou pour ajuster les paramètres d'une distribution existante. Sur la figure 2.5 sont représentées les probabilités de non-respect des échéances des 12 trames HRT de notre application pour $\lambda = 10$ et $\lambda = 30$ avec $\alpha = 0{,}1$ et la distribution de u donnée par l'équation (2.17) de paramètre $p = 0{,}04$. On observe que globalement une forte priorité est un gage de fiabilité de la transmission. Néanmoins, la priorité seule ne peut expliquer ces résultats et il est certain que la période d'émission des trames influent sur la probabilité de non-respect. Le lecteur pourra se référer à [Navet *et al.*, 1999] (u suivant la loi "géométrique modifiée") et [Navet and Song, 1998b] ($u = 1$) pour plus de détails.

La probabilité de non-respect des échéances nous informe sur la fiabilité du système qui est considérée comme l'indicateur principal de sûreté de fonctionnement [Ziegler *et al.*, 1994]. Elle donnera aussi des indications au concepteur de l'application pour le choix des périodes et priorités des messages, ainsi que sur le choix du bon support de transmission au regard de ses objectifs de coûts et de SdF.

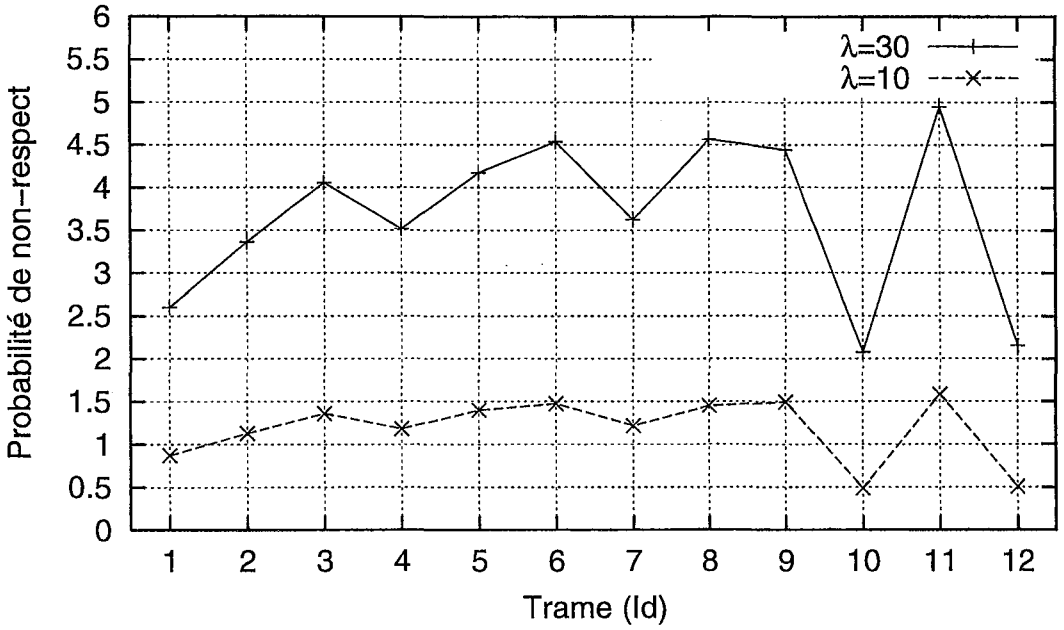


FIG. 2.5 – *Probabilité de non-respect des échéances pour* $\lambda = 10$ *et* $\lambda = 30$, $\alpha = 0{,}1$, $p = 0{,}04$.

Le travail présenté dans cette section a été réutilisé dans [Rüdiger, 1998a; Rüdiger, 1998b] où l'auteur introduit une fonction de coût C qui reflète la capacité du système à respecter ces échéances. L'espérance de C est définie comme

$$E[C] = \sum_{m_k \in \mathcal{M}} c_k \cdot P[R_k > D_k]$$

où c_k est le coût induit par le non-respect de l'échéance associée à la trame m_k et $P[R_k > D_k]$ est estimée à l'aide de l'analyse précédemment exposée, l'objectif étant bien évidemment de minimiser $E[C]$. Un outil intégrant cette analyse a été développé et est présenté dans [Weseloh and Rüdiger, 1999].

2.2.4 Temps d'atteinte de l'état bus-off

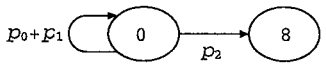
Les mécanismes de confinement d'erreurs de CAN, résumés dans le paragraphe 1.2.5 du chapitre précédent, ont pour objectif de détecter des dysfonctionnements d'origine matériels et de neutraliser (état *erreur passive*) ou d'isoler les stations défectueuses (état *bus-off*). Néanmoins, lorsque le médium n'est pas parfaitement fiable, il peut arriver qu'une simple succession d'erreurs de transmission entraîne le passage de la station dans un de ces deux modes de marche dégradés. Il s'agit donc d'évaluer la probabilité d'occurrence d'un tel événement pour, si cela se justifie, prévoir une stratégie de repli adéquate (ex : changement de mode de marche sur déconnexion d'une station).

2.2.4.1 Modélisation

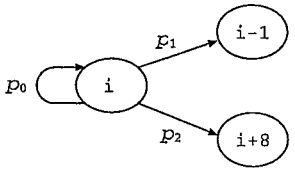
Nous nous intéressons au temps moyen d'atteinte de l'état *bus-off* et à la variance des temps d'atteinte. Pour cela, nous modélisons le compteur d'erreurs en émission (abrégé en TEC pour *Transmit Error Counter*) par une chaîne de Markov. Un processus stochastique est une chaîne de Markov si son espace d'état est discret et si le déroulement futur du processus ne dépend que de son état présent et pas de son passé. Clairement, le TEC peut être modélisé par une chaîne de Markov que nous choisirons en temps discret. Pour cela, nous devrons faire l'hypothèse que le temps est discrétisé, l'unité de temps (le slot) étant le temps moyen de transmission d'un message émis par la station considérée. Cela revient à imposer qu'une station ne peut prendre le bus qu'au début d'un slot, ce qui, sur une période de temps suffisamment longue, ne change rien au nombre de messages effectivement transmis.

On note p_0 la probabilité que la station considérée n'ait pas de trame à transmettre sur un slot de temps, p_1 la probabilité que la station émette une trame qui ne sera pas corrompue et p_2 la probabilité que la station considérée transmette une trame qui sera corrompue par une erreur de transmission. La règle générale est que le TEC est incrémenté de 8 si une trame émise est corrompue et décrémentée de 1 si la transmission est réussie, néanmoins il faut distinguer plusieurs cas de figures en fonction de la valeur courante du TEC, notée i :

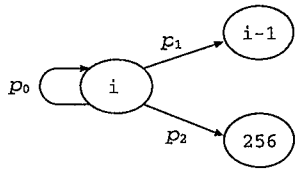
- $i = 0$:



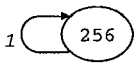
- $i \in \{1..248\}$:



- $i \in \{249..255\}$:



- $i = 256$:



L'état 256, qui correspond au passage de la station dans l'état bus-off, est un état particulier, dit *absorbant*, duquel on ne peut ressortir une fois qu'il est atteint et qui stoppe le processus. Cela correspond exactement au comportement réel d'une station CAN (cf. paragraphe 1.2.5). Avec les règles précédemment exposées, on obtient la matrice des probabilités de transition suivante de taille $257 * 257$ (la chaine de Markov comportant 257 états) :

$\mathcal{P} =$

	0	1	2	3	4	5	6	7	8	9	..	253	254	255	256
0	$p_0 + p_1$	0	0	0	0	0	0	0	p_2	0	..	0	0	0	0
1	$p1$	$p2$	0	0	0	0	0	0	0	p_2	..	0	0	0	0
2	0	$p1$	$p2$	0	0	0	0	0	0	0	..	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	..	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	..	.	.	.	.
254	0	0	0	0	0	0	0	0	0	0	..	p_1	p_0	0	p_2
255	0	0	0	0	0	0	0	0	0	0	..	0	p_1	p_0	p_2
256	0	0	0	0	0	0	0	0	0	0	..	0	0	0	1

La matrice $\mathcal{P}$ peut être mise sous la forme suivante dite *forme canonique* :

$$\mathcal{P} = \begin{bmatrix} 1 & 0 \\ \mathcal{R} & \mathcal{Q} \end{bmatrix} \tag{2.18}$$

où $\mathcal{Q}$ est la matrice originale sans la 257ième ligne et la 257ième colonne, $\mathcal{R}$ est le 257ième vecteur colonne de $\mathcal{P}$ moins le 257ième élément. Pour toute matrice stochastique qui peut être partitionnée sous la même forme que $\mathcal{P}$ (cf. équation (2.18)), l'existence de la matrice

$$\mathcal{S} = \sum_{r=0}^{\infty} \mathcal{Q}^r = (I - \mathcal{Q})^{-1} \tag{2.19}$$

a été prouvée dans [Bhat, 1984]. Cette matrice joue un rôle fondamental dans la détermination des temps moyen d'atteinte et de la variance des temps moyen d'atteinte. Nous notons $\mathcal{S}_\rho$ le vecteur colonne dont la $k$ième composante est la somme des éléments de la $k$ième ligne de $\mathcal{S}$, de façon similaire, $\mathcal{S}_{\rho 2}$ est le vecteur colonne dont la $k$ième composante est le carré de la $k$ième

composante de $\mathcal{S}_\rho$. Il a été prouvé dans [Bhat, 1984] (pages 77-79) que $||E(N_i)|| = \mathcal{S}_\rho$ où N_i est la variable aléatoire qui donne le temps nécessaire pour atteindre pour la première fois l'état absorbant en partant de l'état i et $||E(N_i)||$, le vecteur colonne qui donne l'espérance du premier temps d'atteinte de l'état absorbant en partant de i. De façon similaire, on a $||V(N_i)|| = (2\mathcal{S} - I)\mathcal{S}_\rho - \mathcal{S}_{\rho 2}$ avec $||V(N_i)||$ le vecteur colonne qui donne la variance des temps d'atteinte. Comme le TEC d'une station qui redevient active après un état bus-off est remis à 0, le premier temps d'atteinte est aussi le temps moyen d'atteinte.

2.2.4.2 Calcul de p_0, p_1, p_2

On note p_0^k la probabilité que la station k n'ait pas de messages à transmettre pendant le prochain slot de temps est égale à 1 moins la charge générée par la station qui est composée de la charge nominale plus l'overhead induit par les erreurs de transmission. À chaque erreur de transmission correspond une retransmission qui elle même peut être entachée d'une d'erreur (et ainsi de suite). On obtient donc :

$$\begin{aligned} p_0^k &= 1 - \left[\left(\sum_{m_i \in \mathcal{M}_k} \frac{C_i}{T_i} \right) + \left(\sum_{m_i \in \mathcal{M}_k} \frac{C_i}{T_i} \right) TeT_k + \left(\sum_{m_i \in \mathcal{M}_k} \frac{C_i}{T_i} \right) {TeT_k}^2 + \ \ldots \right] \\ &= 1 - \left(\sum_{m_i \in \mathcal{M}_k} \frac{C_i}{T_i} \right) / (1 - TeT_k) \end{aligned} \tag{2.20}$$

où $\mathcal{M}_k$ est le sous-ensemble de $\mathcal{M}$ composé uniquement des messages émis par la station k et TeT_k est le taux d'erreur trame de la station k qui peut être estimé par le produit du taux d'erreur bit (TeB), commun à toutes les stations du système, et de la taille moyenne des trames émises par la station k, noté $\tilde{S}_k$:

$$\tilde{S}_k = \left(\sum_{m_i \in \mathcal{M}_k} \frac{S_i}{T_i} \right) / \left(\sum_{m_i \in \mathcal{M}_k} \frac{1}{T_i} \right)$$

avec S_i la taille en bit du message m_i.

p_1^k est la probabilité que la station k émette une trame avec succès :

$$p_1^k = (1 - p_0^k) \cdot (1 - TeT_k) = \sum_{m_i \in \mathcal{M}_k} \frac{S_i}{T_i} \tag{2.21}$$

p_2^k est la probabilité que k émette une trame qui sera corrompue :

$$p_2^k = (1 - p_0^k) \cdot TeT_k = 1 - p_0^k - p_1^k \tag{2.22}$$

2.2.4.3 Application Numérique

Nous nous intéressons aux stations "contrôle moteur" (station numéro 1) et "calculateur carrosserie" (station numéro 5) qui émettent respectivement les sous-ensembles de messages $\mathcal{M}_1 = \{m_1, m_3, m_{10}\}$ et $\mathcal{M}_5 = \{m_8\}$ (cf. figure 1.1). La taille moyenne des trames du contrôle moteur, $\tilde{S}_1$, est de 118,75 bits, alors que $\tilde{S}_5 = 105$ bits. Sur la figure 2.6, on observe que le

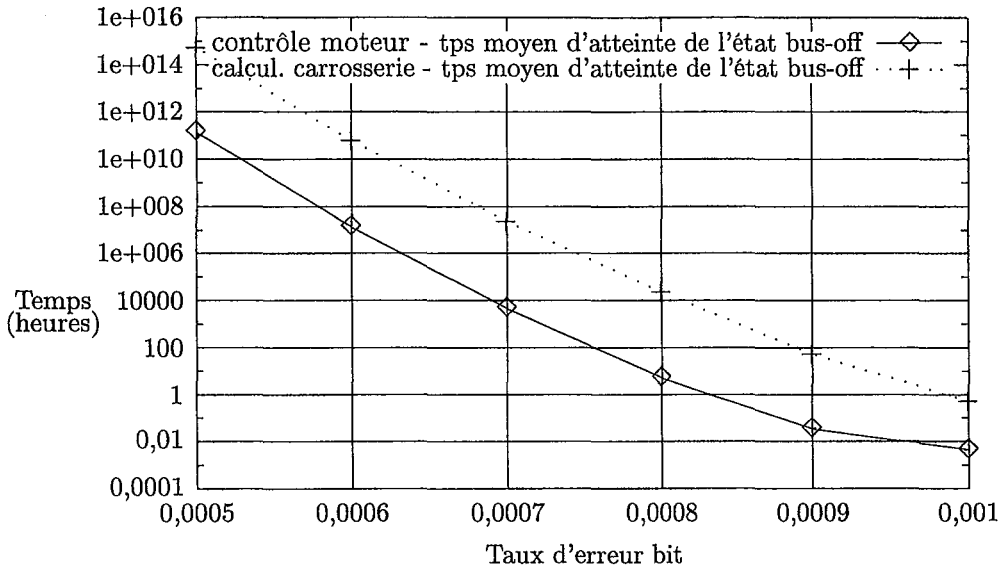


FIG. 2.6 – *Temps moyen d'atteinte de l'état bus-off pour les stations "contrôle moteur" et "calculateur carrosserie" avec un TeB variant de 0,0005 à 0,001.*

temps moyen d'atteinte varie grandement en fonction du TeB, ainsi il faut en moyenne moins de 20 secondes au contrôle moteur pour passer en bus-off avec un TeB de 0.001 (soit un TeT de près de 12%) alors qu'avec un TeB de 0,0007, le temps moyen est de plus de 4700 heures (soit environ la durée d'utilisation totale d'un véhicule). D'autre part, les courbes de la figure 2.6 suggère que plus la charge générée par une station est grande (la charge nominale induite par le contrôle moteur est de 7,6% contre 0,84% pour le calculateur carrosserie), plus la station est susceptible de passer souvent dans l'état bus-off. Notons que l'écart type des temps d'atteinte est très important (du même ordre de grandeur que la moyenne), ce qui dans la pratique implique que les temps d'atteinte que l'on pourra observer seront très variables.

Le programme implantant cette analyse a été codé sous Maple. Les premiers essais ont rapidement révélés que la précision par défaut sous Maple (i.e. 10 décimales significatives) était insuffisante. Une heuristique proposée dans [Gomez *et al.*, 1995] est d'augmenter la précision des calculs de 5 ou 10 unités puis d'observer la précision du résultat obtenu : si elle augmente

du même ordre de grandeur alors le résultat est sans doute fiable. Une précision de 25 chiffres significatifs (*Digits:=25* en Maple) donne des résultats suffisamment précis pour un temps de calcul qui reste raisonnable ($\approx$ 20mn par point de la figure 2.6). Néanmoins, pour s'assurer de la pertinence des résultats, nous avons comparé la moyenne des temps d'atteinte obtenue analytiquement à une moyenne obtenue par simulation. Cette moyenne par simulation a été calculée sur un échantillon de 100 temps d'atteinte, chacun résultat de l'exécution du morceau de code suivant :

```
1  integer time := 0; integer TEC := 0; boolean BusOff := FALSE; real tmp;
2  while (BusOff = FALSE) do
3          time := time + 1;
4          tmp := rand(0,1);
5          if (tmp > p0^k) then
6                                  if (tmp ≤ p0^k + p1^k) /* transmission ok */
7                                    then
8                                          if (TEC > 0) then TEC := TEC − 1; fi
9                                     else                 /* erreur de transmission */
10                                         TEC := TEC + 8;
11                                         if (TEC ≥ 256) then BusOff := TRUE; fi
12                                 fi
13         fi
```

De cette comparaison nous pouvons tirer deux enseignements. D'une part, les temps de simulation deviennent rapidement prohibitifs; il nous a été impossible de calculer les moyennes des temps d'atteinte pour des TeB $<$ 0,0008, ce qui justifie pleinement l'emploi d'une technique analytique pour estimer la fréquence de ces événements trop rares pour la simulation. D'autre part, pour des valeurs du TeB $\geq$ 0,0008, les résultats de simulation sont très voisins des résultats analytiques avec des écarts inférieurs à 5%.

2.2.4.4 Conclusion et perspectives

Dans la littérature traitant du protocole CAN (normes, articles, fiches descriptives de contrôleurs de communication ...), à notre connaissance, le choix des mécanismes de confinement d'erreurs, et particulièrement le choix de la valeur 256 comme seuil de déconnexion des stations, n'est à aucun moment justifié. On peut donc légitimement s'interroger sur la pertinence de ces choix et les premiers résultats de cette étude nous y incitent. En effet, on observe que dans des conditions de perturbations qui paraissent tout à fait envisageables (temporairement) sur un réseau de multiplexage véhicule, les temps moyens d'atteinte de l'état bus-off sont courts et le protocole ne remplit alors pas son office qui est de déconnecter une station du bus exclusivement sur l'occurrence de dysfonctionnements d'origine matériels généralement définitifs. Il nous apparaîtrait donc plus judicieux de permettre la programmation du seuil de

déconnexion, de façon à ce qu'il soit adapté aux conditions d'utilisation propres au système considéré. On peut même envisager que ce seuil puisse évoluer en-ligne en fonction du TeB courant, qui pourra être évalué selon les mécanismes proposés dans la section 5.6.

Un prolongement naturel de cette étude est l'évaluation des temps d'atteinte de l'état erreur-passive (cf. paragraphe 1.2.5), qui sans provoquer de déconnexions physiques, peut néanmoins avoir des conséquences presque aussi néfastes. En effet, si une station erreur-passive garde la capacité d'émettre, elle devra attendre 8 bits supplémentaires une fois le bus devenu libre. Ses trames perdront donc toute priorité entraînant par la même le non-respect quasi-certain des performances prévues par l'analyse ou la simulation. Rappelons que le passage d'une station dans l'état erreur passive est conditionné par l'atteinte de la valeur 128 par l'un au moins des deux compteurs d'erreurs. La modélisation sous forme d'une chaîne de Markov ne pose pas de problèmes particuliers (cf. figure 2.7) : on pourra identifier un état par un couple (i,j) ou i est la valeur du REC et j la valeur du TEC. Le nombre d'état de la chaîne de Markov étant $257 * 128$, la matrice de transition a une taille de $(257 * 128)^2 \approx 1{,}09 * 10^9$, ce qui, associé aux contraintes de précision dans les calculs, ne nous permet pas pour l'instant d'obtenir des résultats numériques. Néanmoins la matrice de transition est fortement creuse car à partir d'un état (i,j), il n'y a généralement pas plus de quatre états atteignables (cf. figure 2.7). Nous pensons donc que des techniques de calcul développées pour les matrices creuses, utilisées conjointement avec une bibliothèque de fonctions permettant des opérations "flottantes" en précision arbitraire, pourrait vraisemblablement résoudre ce problème.

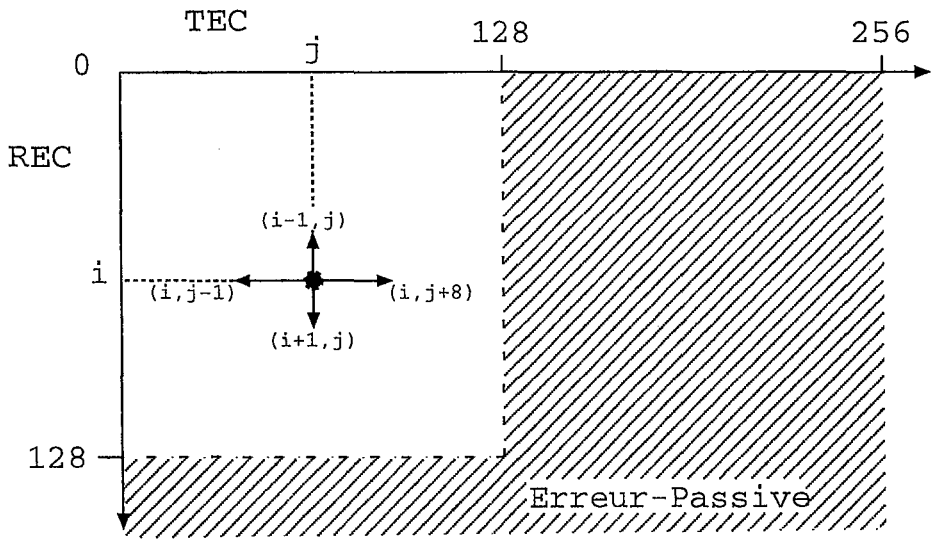


FIG. 2.7 – *Conditions de passage dans l'état "erreur passive" (partie grisée) et modélisation schématique.*

2.3 Evaluation sur prototypes

Dans l'industrie automobile, la construction de prototypes et de maquettes est partie intégrante de la phase de conception. Par l'observation sur un prototype ou sur une maquette, il est possible, comme nous allons l'étudier sur un exemple, de vérifier le respect de certaines contraintes de niveau applicatif. D'autre part, l'observation nous permet de s'assurer de la pertinence de nos modèles, qu'ils soient analytiques ou de simulation, et de fixer au mieux les paramètres de ces modèles. Enfin, un observateur réseau est indispensable pour s'assurer que les différents organes électroniques du véhicule, qui sont majoritairement conçus et fabriqués par des équipementiers, respectent leur spécification en termes de trafic réseau généré.
Dans le cadre d'un contrat industriel avec PSA [Song and Navet, 1997], nous avons développé un observateur réseau, appelé *"Observer"*, qui enregistre toutes les trames échangées sur le réseau et vérifie (1) que le trafic a les caractéristiques voulues et (2) le respect de certaines contraintes temporelles préalablement spécifiées. Un langage de configuration a été conçu pour définir :

- les trames de l'application (identifieur, longueur des données, périodes ...),
- les informations (ou "signaux") à afficher (vitesse, couple ...),
- les contraintes de niveau applicatif à vérifier, sous la forme d'une contrainte de temps entre un stimulus et la réponse associée.

L'analyse du langage ainsi que l'interface graphique ont été programmées dans le cadre d'un stage *AFPA* [André, 1997] et d'un stage de *DESS Informatique* [Coujoulou and Vuillaume, 1997]. *"Observer"* est écrit en langage C et tourne sur un PC portable équipé d'une carte CAN au format PCMCIA [Softing GmbH, 1995]. Certaines parties du code ont été soigneusement optimisées car le nombre de trames à "capturer" et analyser en ligne peut être important : sur un réseau CAN à 250kbit/s chargé à 50%, en considérant une taille moyenne de trames de 100 bits, environ 1250 trames sont échangées par seconde. *"Observer"* a été utilisé avec succès pour vérifier le respect d'une contrainte de type "stimulus-réponse" sur un prototype équipé de la messagerie décrite sur la figure 1.1, soit environ 700 trames par seconde. Pour plus de trames et/ou plus de contraintes, il faudra envisager une vérification hors-ligne par analyse d'un fichier trace.

De façon surprenante, il n'est pas impossible, par exemple lors de l'initialisation du système, d'observer sur le réseau des trames non-répertoriées qui peuvent, par leur niveau de priorité et l'accroissement de la charge réseau, perturber le bon fonctionnement du système. Par défaut, ces trames non-répertoriées sont affichées et enregistrées dans un fichier ainsi que, pour toutes les trames de l'application, les variations de période (observées à la réception) qui seront analysées (valeur maximum, valeur moyenne, écart type) à la fin de l'exécution.

Une contrainte typique dont le respect doit être vérifié est le délai entre un changement de vitesse avec une boite automatique (BVA) et la réduction de couple associé effectuée par le contrôle moteur (CM). Ce délai doit toujours rester inférieur à une certaine valeur, fixée par les concepteurs du véhicule en fonction de certains impératifs (mécaniques, confort, ...). Dans notre application, la BVA transmet toutes les 15 ms une trame contenant le rapport de vitesse engagé. De son coté, le contrôle moteur transmet le couple toutes les 10 ms. Lorsqu'un changement de vitesse est détecté par l'examen de la trame provenant de la BVA (le stimulus), *Observer* doit analyser chaque trame provenant du CM jusqu'à ce que le couple se situe en dessous de la valeur spécifiée (réponse au stimulus).

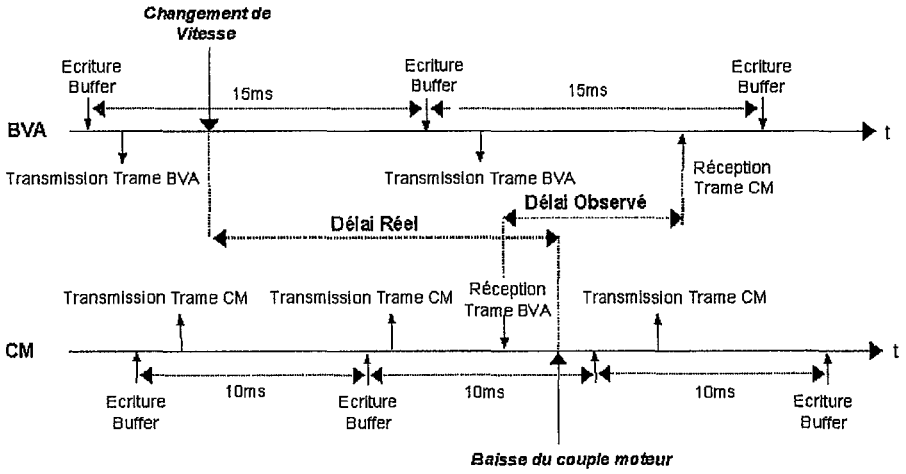


FIG. 2.8 – *Delai "stimulus-réponse" : changement de vitesse - réduction de couple.*

Comme cela est représenté sur la figure 2.8, on remarque que le délai que l'on peut observer (DO) entre le stimulus et la réponse associée, n'est qu'une estimation du délai réel (DR). Néanmoins, cette information est utile car elle nous permet de donner une borne sur le délai réel, en effet nous savons que

$$DR \leq T_{bva} + J_{bva} + DO + C_{bva} - C_{cm} \tag{2.23}$$

où T_{bva} et J_{bva} sont respectivement la période et la gigue de la trame contenant le rapport de vitesse, C_{bva} étant son temps de transmission et C_{cm} étant le temps de transmission de la trame envoyée par le contrôle moteur.

Une démonstration de cet observateur réseau a été effectuée lors d'ateliers de travail sur le réseau CAN organisés pendant l'*École Temps Réel 1997* [Navet and Song, 1997a].

2.4 Conclusion et perspectives

Ce chapitre donne une synthèse des résultats de recherche récents dans le domaine de la validation d'applications distribuées autour d'un réseau CAN. Le processus de validation que nous avons présenté est conduit en résolvant des modèles, analytiques ou de simulation, ainsi que par l'observation sur maquettes ou prototypes. Le tableau 2.4 résume les résultats que l'on peut attendre de chaque technique et leur complémentarité. Les techniques analy-

Contraintes		**Analyse**	**Simulation**	**Observation**
temporelles	tps de réponse max.	√		
	tps de réponse max. avec erreurs	√		
	tps de réponse moyen		√	
	gigue en réception			√
sur la robustesse de l'application	proba. de non-respect des échéances	√	√	
	tps moyen d'atteinte de l'état bus-off	√		
de niveau applicatif	délai stimulus-réponse			√
sur l'utilisation de ressources	charge réseau	√	√	√

TAB. 2.4 – *Les contraintes et leur(s) technique(s) de vérification.*

tiques fournissent des bornes sur les métriques de performance considérées (temps de réponse, temps de réponse avec erreurs de transmission, probabilité de non-respect des échéances) ou permettent d'évaluer l'occurrence d'événements rares (temps d'atteinte de l'état bus-off d'une station CAN). La simulation ne fournit pas de garanties mais des résultats plus proches du cas effectivement rencontré dans la pratique qui permettent au concepteur d'évaluer les performances moyennes de son application et de l'optimiser. Les modèles, qu'ils soient analytiques ou de simulation, mettent clairement l'accent sur le système de communication en utilisant des modèles très simples d'ECU. Il est naturellement possible de développer des modèles de simulation plus détaillés, comme cela a été entrepris dans [Courrier *et al.*, 1998b; Courrier *et al.*, 1998a], néanmoins la validation des modèles, la fixation des paramètres et surtout l'interprétation des résultats deviennent alors problématiques. D'autre part, les ECUs, fournis en grande majorité par des équipementiers, existent généralement dès la phase de conception du véhicule, il est alors intéressant de les utiliser plutôt que des modèles. Ceci nous a conduit à développer un observateur réseau qui est bien adapté à la vérification de contraintes du type stimulus-réponse.

Les modèles analytique de ce chapitre avec les compléments de l'annexe A ont fait l'objet de trois publications dans des congrès [Navet *et al.*, 1998a; Navet and Song, 1998a; Navet and Song, 1998b] et d'une publication dans un journal [Navet *et al.*, 1999]. Dans l'objectif de mieux faire connaître les techniques de validation des applications embarquées dans les véhicules, nous avons également publiés deux articles introductifs dans des revues de diffusion

de la connaissance [Navet, 1998; Navet *et al.*, 1998b]. Cette partie devrait paraître dans son intégralité sous la forme d'un chapitre de livre [Navet and Song, 1999a] et a été partiellement reprise pour un exposé à l'*École Temps Réel 1999* [Song *et al.*, 1999].

Dans le cadre du stage de DEA de Gérald Cabus [Cabus, 1999] que nous avons co-encadré avec Y.Q. Song, nous nous sommes intéressés au problème de l'ordonnancement conjoint de tâches et de messages sur l'exécutif OSEK/VDX [OSEK Group, 1997] proposé comme standard dans l'industrie automobile. En particulier, nous avons modélisé le service d'activation d'une tâche sur réception d'un message et avons développé une analyse permettant de calculer une borne sur le temps de réponse de bout en bout du système informatique (délai R_I sur la figure 3 de l'introduction). Cette analyse d'ordonnançabilité dans le cas distribué, appelée également analyse "holistique", est relativement classique dans la littérature [Tindell and Clark, 1994; Saad-Bouzefrane, 1997; Saad-Bouzefrane and Cottet, 1997; Bate and Burns, 1998] et le premier objectif de ce stage de DEA a été d'adapter les travaux existants à l'exécutif OSEK/VDX qui a pour particularité d'offrir deux politiques d'ordonnancement de tâches, à savoir FPP préemptif et FPP non-préemptif. Le second objectif du stage de DEA était de proposer une méthode d'allocation des tâches sur les différents sites et d'affectation des priorités des tâches et des messages qui engendre des solutions d'ordonnancement faisables. Ce problème étant connu pour être NP-complet (cf. [Tindell *et al.*, 1992]), il était nécessaire de mettre en oeuvre une technique d'optimisation. Notre choix s'est porté sur les algorithmes génétiques (cf. paragraphe "étude de l'existant" de la section 4.1 pour des éléments de justification). La qualité de chaque solution d'ordonnancement envisagée est estimée avec les résultats de l'analyse d'ordonnançabilité : elle est inversement proportionnelle au nombre d'échéances non respectées. Outre les nécessaires opérateurs génétiques de croisement et de mutation, Gérald Cabus a proposé des heuristiques de placement des tâches et de fixation des priorités pour la création de la population initiale de l'algorithme génétique. Notons néanmoins que l'utilisation effective de ces travaux dans un contexte industriel nécessiterait que le concepteur du système, c'est à dire le constructeur automobile, ait une connaissance complète des tâches embarquées dans les ECUs ce qui n'est pas le cas actuellement.

Dans les chapitres 3 et 4, nous étudierons l'ordonnancement de tâches sur des exécutifs se conformant au standard Posix1003.1b qui offrent des fonctionnalités d'ordonnancement plus riches qu'OSEK/VDX et verrons que d'autres objectifs que la simple faisabilité du système peuvent être poursuivis.

Deuxième partie

Ordonnancement de tâches sous Posix1003.1b

Chapitre 3

Analyse d'ordonnançabilité des systèmes Posix1003.1b

Résumé

Les analyses d'ordonnançabilité dans un contexte mono-processeur ont fait l'objet de très nombreuses études depuis l'article fondateur de Lui et Layland [Liu and Layland, 73] parmi lesquelles on peut citer [Joseph and Pandya, 1986; Sha *et al.*, 1990; Mok and Chen, 1997; George *et al.*, 1996]. Récemment, un cadre théorique permettant d'analyser des systèmes dans lesquels plusieurs politiques peuvent coexister a été proposée dans [Migge and Jean-Marie, 1998; Migge and Jean-Marie, 1999; Migge, 1999]. Dans ce chapitre, nous nous servons de ce travail pour proposer une analyse d'ordonnançabilité des systèmes Posix1003.1b effectuée en collaboration avec les auteurs de [Migge and Jean-Marie, 1998] et qui fait l'objet d'une soumission en cours d'évaluation [Migge *et al.*, 1999]. Outre de décider de la faisabilité ou non-faisabilité d'un ensemble de tâches, c'est à dire du respect ou du non-respect des contraintes de dates au plus tard sur les fins d'exécution de ces tâches, les bornes sur les temps de réponse proposées dans ce chapitre nous permettront de dériver les gigues sur la production d'éventuels messages et donc de vérifier, à l'aide de l'analyse développée dans la section 2.2, si les contraintes pesant sur les communications inter-tâches dans le cas distribué, sont respectées.

3.1 Introduction

Contexte de l'étude. Une des conclusions de [Stankovic *et al.*, 1996], dans lequel les 13 auteurs s'interrogent sur les développements futurs de l'informatique temps réel, est que des considérations économiques pousseront de plus en plus à l'utilisation de composants logiciels et matériels disponibles commercialement (ou composants sur étagères - *off-the-shelf*) pour construire des systèmes temps réel. C'est particulièrement vrai pour les systèmes d'exploitation

(OSs) temps réel où de nombreux exécutifs commerciaux comme *Lynx* ou *QNX*, conformes au standard Posix1003.1b, possèdent des caractéristiques intéressantes dans une optique temps réel telles que des temps de changements de contextes minimes, des sémaphores, des timers programmables et de nombreux niveaux de priorités pour les tâches. L'alternative "faire ou acheter" a généralement pour réponse "acheter" quand les temps de développements, la portabilité, l'existence de drivers de périphériques ou de profils de communication sont des critères de décision. D'un autre coté, l'utilisation d'OSs du commerce posent souvent des problèmes de certification et leur comportement en présence de fautes est souvent médiocre et mal défini [Salles *et al.*, 1997; Koopman *et al.*, 1997].

Posix (*Portable Operating System Interface*) [(ISO/IEC), 1996] est une famille de standards qui définissent les fonctionnalités et les interfaces qu'un OS de type Unix doit comporter. L'objectif premier de Posix est de faciliter la portabilité au niveau du code source. Posix1003.1b, anciennement Posix.4, définit des extensions temps réel à la norme de base concernant principalement les signaux, la communication inter-processus, la possibilité de "bloquer" tout ou partie d'un processus en mémoire vive, les entrées sorties synchrones et asynchrones, les timers et les politiques d'ordonnancement.

La plupart des OSs temps réel du marché se conforment, au moins partiellement, à ce standard même si il a été critiqué [Moses, 1995] pour ne pas correspondre aux besoins d'applications embarquées de petite taille.

L'ordonnancement sous Posix1003.1b Posix1003.1b définit trois politiques d'ordonnancement qui s'appliquent par processus : chaque processus s'exécute avec une politique et à une priorité donnée, toutes deux héritées du "père", mais que le processus peut ensuite modifier pendant son exécution. Les trois politiques disponibles sont :

- *sched_fifo* sous laquelle, à chaque instant, le processeur est alloué à la tâche de plus forte priorité susceptible de s'exécuter avec un ordre FIFO pour les tâches de même priorité. Sous l'hypothèse de priorités fixes au cours du temps, *sched_fifo* est équivalent à FPP.
- *sched_rr* qui est une politique de type Round-Robin permettant à des processus de même priorité de se partager la ressource. Notons qu'un ensemble de processus *sched_rr* au même niveau de priorité n'obtiendra le processeur que si aucune tâche plus prioritaire, qu'elle soit *sched_rr* ou *sched_fifo* n'est active. La valeur du quantum de temps peut être une constante globale, fonction du niveau de priorité ou spécifique à chaque processus.
- *sched_other*, une politique dont la définition est laissée libre à l'implémenteur de l'OS, et dont le standard ne requiert que l'existence et la documentation. *sched_other* peut simplement renvoyer à *sched_fifo* ou *sched_rr* ou implanter une politique Unix classique

de temps partagé[11]. Étant donné qu'il n'est pas possible de garantir un comportement identique d'un système à l'autre, l'utilisation de cette politique n'est pas recommandée pour des questions de portabilité, et nous ne la considérerons pas dans notre analyse.

A chaque politique d'ordonnancement est associé un intervalle de priorités; en fonction de l'implémentation les trois intervalles de priorités se recoupent ou non.

Le standard Posix1003.1c [Butenhof, 1997] définit les *threads* qui sont des processus légers évoluant dans le même espace d'adressage que le processus auquel elles se rattachent. Deux types de threads existent :

- Les threads ordonnancées globalement (*system contention scope*) : indépendamment du processus dans lequel elles résident, ces threads sont en concurrence avec toutes les autres tâches et toutes les autres threads ordonnancées globalement pour le processeur. L'analyse qui est développée par la suite, s'applique également aux threads de ce type même si nous parlerons de tâches.
- Les threads ordonnancées localement (*process contention scope*) : ces threads n'entrent en concurrence qu'avec les threads du même processus pendant le temps processeur qui est alloué au processus. Le standard ne précise malheureusement pas comment ces threads sont ordonnancées par rapports aux threads ordonnancées globalement et aux tâches. L'ordonnancement de ces threads étant dépendant de l'implémentation, nous ne les considérerons pas dans la suite.

Dans le reste de ce chapitre, nous définissons une tâche comme une activité récurrente qui est exécutée par l'activation répétitive d'un processus (ou thread) ou par un processus unique qui se met en attente à la fin d'un cycle d'exécution et se ré-active au début du suivant.

Hypothèses de travail L'analyse d'ordonnançabilité des systèmes Posix1003.1b sera proposée sous un certain nombre d'hypothèses relativement peu contraignantes qui visent d'une part à restreindre les politiques *sched_fifo* et *sched_rr* de telle façon que l'analyse des temps de réponse deviennent possibles (hypothèses 3, 4 et 5), et d'autre part, à éviter certains détails qui rendraient la compréhension plus difficile (hypothèses 1 et 2). Les hypothèses sont les suivantes :

1. les gigues sur les dates d'activation des tâches ne sont pas considérées,
2. les temps de changement de contexte sont négligés,
3. toutes les tâches sous *sched_fifo* doivent posséder des priorités différentes et toutes les instances d'une même tâche *sched_fifo* doivent être activées avec la même priorité (sous cette hypothèse *sched_fifo* est équivalent à FPP),

11. Généralement une politique de type Round-Robin où le quantum est fonction de la priorité du processus et de son ancienneté dans le système (plus le processus est ancien, plus petit sera son quantum).

4. toutes les instances d'une même tâche *sched_rr* doivent être activées avec la même priorité (sous cette hypothèse, *sched_rr* est équivalent à la politique Round-Robin (RR)),
5. des tâches ordonnancées sous des politiques différentes ne peuvent partager le même niveau de priorité.

Les gigues sur les dates d'activation (hypothèse 1) et les temps de changement de contexte (hypothèse 2) peuvent être intégrés à l'analyse d'ordonnançabilité de la même façon que cela a été fait pour le cas strictement FPP dans [Burns *et al.*, 95]. En pratique, l'hypothèse 5 est facilement contournable : il suffit de d'ordonnancer toutes les tâches sous RR et d'allouer aux tâches initialement FPP un quantum égal à leur temps d'exécution. Dans la suite du chapitre, *sched_fifo* sera désignée sous le nom de FPP et *sched_rr* sous le nom de RR.

Organisation du chapitre Les deux premières sections sont consacrées à la description de concepts introduits dans [Migge and Jean-Marie, 1998; Migge and Jean-Marie, 1999]. Ces concepts seront utilisés pour une analyse des temps de réponse de tâches ordonnancées sous FPP (cf. section 3.4) puis sous RR (cf. section 4.4). Notons que si dans la littérature le cas FPP a été étudié dans toutes ses extensions possibles (entre autres références, cf. [Liu and Layland, 73; Joseph and Pandya, 1986; Audsley, 1991a; Tindell *et al.*, 1992; Tindell, 1992; Tindell, 1993; Tindell, 94; Tindell *et al.*, 1994; Sun *et al.*, 1997]), à notre connaissance, l'analyse des temps de réponse sous RR est originale. En section 3.6, nous verrons comment intégrer l'existence de sections critiques dans le calcul des bornes sur les temps de réponse. Enfin, la section 3.7 est consacrée à une étude de cas.

3.2 Couches de priorités

Dans ce paragraphe, nous expliquons de façon informelle le concept de couches de priorités introduit dans [Migge and Jean-Marie, 1999]. Soit un ensemble de tâches $\mathcal{T} = \{\tau_1, \ldots, \tau_m\}$, imaginons que ces tâches soient groupées en différentes couches $\lambda_1, \lambda_2, \ldots, \lambda_n$, les couches étant priorisées entres elles ($\lambda_1 > \lambda_2 > \ldots > \lambda_n$ avec $\forall k$ t.q. $\tau_k \in \lambda_l$, j t.q. $\tau_j \in \lambda_{l+1}$ $\tau_k > \tau_j$) et chaque couche étant ordonnancée sous une politique particulière. Dans le cadre de Posix1003.1b, deux politiques seront possibles, à savoir FPP et RR.

On note $\tau_{k,n}$, la $n^{\text{ième}}$ instance d'une τ_k qui se situe dans une couche λ_l. En ne considérant que des politiques non-idling et préemptives (ex: FPP, EDF, RR, Least-Laxity, FIFO), la règle est que $\tau_{k,n}$ est susceptible d'être exécutée dès que, et aussi longtemps qu'aucune instance d'une tâche n'est active dans une des couches de priorités supérieures $\lambda_1, \ldots, \lambda_{l-1}$. Dans le temps processeur imparti à une couche λ_l, l'ordonnancement est réalisé selon la politique de λ_l. Pour

FPP, $\tau_{k,n} \in \lambda_l$ est exécuté dès que, et aussi longtemps qu'il n'y a aucune instance active d'une tâche plus prioritaire (τ_j avec $j < k$) appartenant à la même couche. Pour RR, chaque tâche de la couche aura une priorité identique et se verra attribuer un certain quantum de temps pendant lequel la ressource lui sera réservée. Un exemple de découpage en couches de priorité est représenté sur le tableau 3.1.

3.3 Fonctions d'arrivée de travail

Dans [Migge and Jean-Marie, 1998], les auteurs introduisent le concept de fonction d'arrivée de travail (*work arrival function* - WAF) qui estime la quantité de travail générée par un ensemble de tâches pendant un certain intervalle de temps. Avec $C_{k,n}$ et $A_{k,n}$ respectivement le temps d'exécution et la date d'activation[12] de la $n^{\text{ième}}$ instance de la tâche τ_k, la WAF pour τ_k est

$$S_k(a,b) = \sum_{n \in \mathbb{N}} C_{k,n} \cdot \mathbb{I}_{[a \leqslant A_{k,n} < b]}. \tag{3.1}$$

Sa valeur est simplement la somme du travail généré par les différentes instances de τ_k sur l'intervalle $[a,b[$. Considérons par exemple une tâche périodique τ_k, de période T_k et dont toutes les instances nécessitent C_k unités de temps sur le processeur, alors sur $[0,t[$

$$S_k(0,t) = \left\lceil \frac{t - A_{k,0}}{T_k} \right\rceil \cdot C_k. \tag{3.2}$$

Pour pouvoir déterminer des bornes sur le temps de réponse, il faut s'intéresser à la quantité maximale de travail pouvant arriver sur un intervalle de temps, quantité donnée par fonction d'arrivée de travail dite majorante (*majorizing work arrival function* - MWAF).

Définition 1 *La fonction d'arrivée de travail majorante $s_k(.)$ de la tâche τ_k est une borne sur le travail induit par les instances de τ_k sur tout intervalle de temps de durée t*

$$S_k(u,u+t) \leqslant s_k(t) \qquad \forall\, u \geqslant 0, t \geqslant 0. \tag{3.3}$$

Avant de déterminer la MWAF pour les tâches périodiques, nous devons introduire le concept de période d'interférence.

Définition 2 *Une période d'interférence de niveau k est un intervalle de temps $[U^k,V^k[$ tel qu'il n'y ait pas d'instance de tâches de priorité supérieure à k en attente de traitement ni à*

12. Dans le sens du premier instant auquel la tâche est susceptible de s'exécuter et non le premier instant auquel la tâche accède effectivement à la ressource.

son début U^k, ni à sa fin V^k, alors qu'à tout autre instant, il existe au moins une telle instance en attente.

Considérons toujours une tâche périodique τ_k, de période T_k et dont toutes les instances nécessitent C_k unités de temps sur le processeur. Soit t un instant dans une période d'interférence de niveau k $[U^k,V^k[$, A_{k,n_0} l'instant d'activation de la première instance de τ_k après ou à U^k et A_{k,n_1} la dernière date d'activation strictement avant t. Supposons qu'il y ait i instance de τ_k dans $[U^k,t[$ (i.e. $i = n_1 - n_0 + 1$), la quantité de travail sur cet intervalle est $\sum_{n=n_0}^{n_1} C_{k,n}$, soit $i \cdot C_k$. La dernière instance est disponible à l'instant

$$A_{k,n_1} = A_{k,n_0} + \sum_{n=n_0}^{n_1-1} T_{k,n} \geqslant U^k + (i-1) \cdot T_k.$$

Par hypothèse $A_{k,n_1} < t$, donc $i < 1 + (t - U^k)/T_k$ et comme i est une valeur entière, il y a donc au plus $i = \lceil (t - U^k)/T_k \rceil$ instances ce qui implique que la demande de la tâche en termes de temps CPU est bornée par

$$s_k(t - U^k) = \left\lceil \frac{t - U^k}{T_k} \right\rceil \cdot C_k. \tag{3.4}$$

La MWAF est particulière au type des tâches considérées. Dans cette étude, nous ne nous intéressons qu'aux tâches périodiques, mais des tâches plus complexes peuvent être considérées, par exemple les tâches sporadiquement périodiques [Tindell *et al.*, 1994] (i.e. temps de cycle entre deux instances variables), les tâches multi-cadres [Mok and Chen, 1997] (i.e. temps d'exécution variables) et les tâches sporadiquement périodiques multi-cadres (cf. [Migge, 1999] pour les MWAFs correspondantes).

Les fonctions d'arrivée de travail permettent de décrire les formules donnant les temps de réponse pour une politique donnée indépendamment du type de tâche considéré. Ainsi, l'analyse se décompose en deux parties distinctes : caractériser le flux d'arrivée de travail en termes de MWAF et proposer la formule donnant la borne sur le temps de réponse pour la politique particulière considérée. L'analyse d'une nouvelle politique peut donc se baser sur des MWAFs connues et les MWAFs de nouveaux types de tâches peuvent être utilisées pour l'analyse d'une politique connue.

3.4 Borne sur le temps de réponse : le cas FPP

Sous la politique FPP, les priorités des instances de tâches sont fixes au cours du temps et à chaque instant le processeur est alloué à la tâche de plus forte priorité susceptible de s'exécuter, les tâches de priorité identique étant départagées selon le principe FIFO.

Le temps de réponse d'une instance $\tau_{k,n}$ est par définition $R_{k,n} = E_{k,n} - A_{k,n}$ où $E_{k,n}$ est sa date de fin d'exécution. Si l'on note $S_{1..i}(a,b) = \sum_{k=1}^{i} S_k(a,b)$, la charge de travail sur $[a,b[$ des tâches de priorités 1 à i et $W_{1..i}(t)$ la fonction qui évalue le travail encore en attente de traitement (*pending work*) pour les tâches de priorités 1 à i à un instant t, $E_{k,n}$ est

$$E_{k,n} = \min\{t > A_{k,n} \mid W_{1..k}(A_{k,n}) + C_{k,n} + S_{1..k-1}(A_{k,n},t) = t - A_{k,n}\}. \tag{3.5}$$

C'est le premier instant t après $A_{k,n}$ tel que la somme du travail plus prioritaire et de même priorité (les instances précédentes de τ_k) en attente à l'instant $A_{k,n}$ (i.e. $W_{1..k}(A_{k,n})$), du temps d'exécution de l'instance considérée (i.e. $C_{k,n}$) et des arrivées de travail plus prioritaire sur $[A_{k,n},t[$ (i.e. $S_{1..k-1}(A_{k,n},t)$) est égale au temps d'exécution disponible ($t - A_{k,n}$). Notons que les WAFs sont définies telles que $S_{1..k}(a,b)$ ne prend pas en compte les tâches exactement disponibles à l'instant t, celles-ci ne pouvant préempter $\tau_{k,n}$ si celle-ci finit son exécution en t.

Pour pouvoir déterminer des bornes sur le temps de réponse, il est nécessaire de préciser la quantité $W_{1..k}(A_{k,n})$. On note $U_{k,n}$, le début de la période d'interférence dans laquelle $\tau_{k,n}$ est activée. $W_{1..k}(A_{k,n})$ dans l'équation (3.5) est fonction de la quantité de travail plus prioritaire que τ_k arrivé après $U_{k,n}$ et du temps $A_{k,n} - U_{k,n}$, plus précisément

$$W_{1..k}(A_{k,n}) = S_{1..k}(U_{k,n},A_{k,n}) - (A_{k,n} - U_{k,n}). \tag{3.6}$$

Avec $S_k^n = S_k(U_{k,n},A_{k,n}) + C_{k,n}$, la quantité de travail de $\tau_{k,n}$ et des instances précédentes de τ_k, l'instant de fin d'exécution de $\tau_{k,n}$ satisfait toujours

$$E_{k,n} = \min\{t > U_{k,n} \mid S_{1..k-1}(U_{k,n},t) + S_k^n = t - U_{k,n}\}. \tag{3.7}$$

Il a été prouvé dans [Migge *et al.*, 1999] que le temps de réponse de toute instance d'une tâche FPP est borné par le temps de réponse d'une instance de la même tâche dans la première période d'interférence du scénario d'activation majorant (*majorizing release pattern*). Le scénario d'activation majorant étant la séquence des mises à disposition $a_{k,n}$ (de temps d'exécution $c_{k,n}$, de temps de réponse $r_{k,n}$ et d'instant de fin d'exécution $e_{k,n}$) induit par la MWAF :

$$R_{k,n} \leqslant \max_{j \mid a_{k,j} < e_{k,j-1}} r_{k,j}.$$

Les arguments de la preuve sont donnés en annexe B.1. Concrètement, ce résultat nous dit qu'il suffit d'analyser les temps de réponse des instances de la première période d'interférence

de niveau k du scénario d'activation majorant pour trouver la borne sur le temps de réponse de la tâche τ_k. La fin d'exécution de $\tau_{k,n}$ est donné par

$$e_{k,n} = \min\{x > 0 \mid s_{1..k-1}(x) + s_k^n = x\} \ . \tag{3.8}$$

avec $s_{1..i}(t) = \sum_{k=1}^{i} s_k(t)$ et $s_k^n = \sum_{i=0}^{n} c_{k,i}$. Il nous suffit de calculer $r_{k,j} = e_{k,j} - a_{k,j}$ pour $j = 0,1,2,\ldots$ jusqu'à la fin de la période d'interférence (i.e. $a_{k,j} < e_{k,j-1}$) . Les algorithmes implantant cette analyse sont présentés dans la figure 3.1. La variable c de la fonction *RespTimeBound*() comptabilise le travail des instances de la tâche considérée, noté s_k^n dans l'équation (3.8). La fonction $f(k,t)$ calcule la demande de travail des tâches plus prioritaires $s_{1..k-1}(t)$ sur l'intervalle $[0,t[$, elle est utilisée aux lignes 5-8 pour calculer l'instant de fin d'exécution de l'instance considérée $e_{k,n} = \min\{t > 0 \mid f(k,t) + s_k^n = t\}$. Les temps de réponse des instances de la tâche τ_k sont examinées jusqu'à la fin de la période d'interférence (i.e. tant que $a < e$ à la ligne 13).

```
1  funct time RespTimeBound(task k)
2    max := 0; n := 0; a := 0;
3    c := c_k; w := c_k;
4    repeat
5      repeat
6        e := w;
7        w := f(k,e) + c;
8      until w = e;
9      r := e − a;
10     if r > max then max := r; fi
11     a := a + t_k;
12     c := c + c_k
13   until a ⩾ e;
14   return max;
15 end
```

```
funct time f(task k,time t)
  w := 0
  for i := 1 to k − 1 do
      w := w + s(i,t);
  od
  return w;
end

funct time s(task i,time t)
  w := 0; a := 0; j := 0;
  while a < t do
        w := w + c_i;
        a := a + t_i;
  od
  return w;
end
```

FIG. 3.1 – *Algorithmes de calcul de la borne sur les temps de réponse d'une tâche τ_k ordonnancées sous FPP (avec $\forall n \quad c_{k,n} = c_k$).*

3.5 Borne sur le temps de réponse : le cas Round-Robin

L'idée sous-jacente de la politique Round-Robin est de réserver une fraction de la ressource, en l'occurrence le processeur, à toutes les tâches de même priorité. Dans une couche Round-Robin, chaque tâche peut s'exécuter pendant un certain quantum de temps sans être préemptée

par une tâche de la même couche de priorité. Une tâche RR pourra bien sûr être préemptée par une ou plusieurs tâches plus prioritaires devenues actives pendant sont exécution.

Considérons tout d'abord uniquement les tâches d'une même couche RR. Soit τ_k une tâche RR, de façon répétitive, τ_k se verra attribuer un quantum de temps Ψ_k. Si la tâche n'a pas d'instance active (resp. si elle a moins de travail à effectuer que Ψ_k), alors le quantum (resp. le reste du quantum) est perdu. Le temps entre deux quanta peut varier, selon la demande des autres tâches, mais est borné $\Psi^l = \sum_{\tau_k \in \lambda_l} \Psi_k$.

Dans le cas RR, le concept de période d'interférence de niveau k demande à être légèrement redéfini. Par rapport au cas FPP, dans la mesure où toutes les tâches d'une même couche RR ont une priorité identique, la définition doit se faire en considérant k comme l'identifiant de la tâche τ_k et non plus comme sa priorité.

Définition 3 *Une période d'interférence RR pour une tâche τ_k est un intervalle de temps $[U,V[$ tel qu'à son début U et à sa fin V il n'y a pas d'instance de τ_k ou de tâches de priorité supérieure à celle de τ_k en attente de traitement, alors qu'à tout autre instant, il existe au moins une telle instance en attente.*

Considérons une période d'interférence RR pour la tâche τ_k. La tâche τ_k utilisera complètement tous les quanta mis à sa disposition excepté peut-être le dernier. Avant qu'une instance $\tau_{k,n}$ ne gagne le processeur, toutes les instances précédentes de τ_k doivent déjà avoir été exécutées. La somme du travail à effectuer pour τ_k dans $[U,E_{k,n}[$ est donc $S_k^n = S_k(U,A_{k,n}) + C_{k,n}$, ce qui correspond à $\lceil S_k^n/\Psi_k \rceil$ cycles Round-Robin. Dans chaque cycle RR, les autres tâches de la couche peuvent utiliser $\overline{\Psi}_k = \Psi^l - \Psi_k$ unités de temps processeur. Sur l'intervalle $[U,E_{k,n}[$ la demande des autres tâches est donc bornée par $\lceil S_k^n/\Psi_k \rceil \cdot \overline{\Psi}_k$. Néanmoins, cette demande ne peut excéder la somme des quantités de travail induites par les autres tâches de la même couche RR actives à l'instant U et des instances arrivant sur $[U,t[$, respectivement

$$\overline{W}_k(U) = \sum_{\tau_i \in \lambda_l, \tau_i \neq \tau_k} W_i(U) \quad \text{et} \quad \overline{S}_k(U,t) = \sum_{\tau_i \in \lambda_l, \tau_i \neq \tau_k} S_i(U,t)$$

Naturellement les tâches de la couche RR peuvent être préemptées par des tâches plus prioritaires qu'elles soient RR ou FPP. Notons que par définition de la période d'interférence RR de la tâche τ_k , $W_{1..m_{l-1}}(U) = 0$. La demande des tâches autres que τ_k sur un intervalle $[U,t[$ est donc bornée par

$$\overline{\Psi}_k(U,t) = \min\left(\left\lceil \frac{S_k^n}{\Psi_k} \right\rceil \cdot \overline{\Psi}_k, \overline{W}_k(U) + \overline{S}_k(U,t) \right) + S_{1..m_{l-1}}(U,t). \tag{3.9}$$

En utilisant (3.9), on en déduit que

$$E_{k,n} = \min\{t > U \mid \overline{\Psi}_k(U,t) + S_k^n = t - U\}$$

où $E_{k,n}$ est l'instant de fin d'exécution de $\tau_{k,n}$.

Avec des arguments similaires au cas FPP, on peut prouver que le temps de réponse de toute instance d'une tâche RR est borné par le temps de réponse d'une instance de la même tâche dans la première période d'interférence du scénario d'activation majorant. Le lecteur est invité à se référer à [Migge *et al.*, 1999]. A nouveau, ce résultat nous dit qu'il nous suffit de calculer $r_{k,j} = e_{k,j} - a_{k,j}$ pour $j = 0,1,2,\ldots$ jusqu'à la fin de la période d'interférence (i.e. $a_{k,j} < e_{k,j-1}$) .

La fonction f de la figure 3.1 demande à être adaptée pour le calcul de la borne sur le temps de réponse d'une tâche se trouvant dans une couche RR. Sur la figure 3.2, $s_k^*(t)$ (à la ligne 7) et une borne sur la quantité $\overline{W}_k(U) + \overline{S}_k(U,t) + S_{1..m_{l-1}}(U,t)$ de l'équation 3.9 qui est donnée dans l'annexe B.2. D'après la définition des MWAFs, on sait que :

$$\left\lceil \frac{S_k^n}{\Psi_k} \right\rceil \cdot \overline{\Psi}_k + S_{1..m_{l-1}}(U,t) \leqslant \left\lceil \frac{s_k^{\tilde{n}}}{\Psi_k} \right\rceil \cdot \overline{\Psi}_k + s_{1..m_{l-1}}(t-U).$$

La quantité $s_{1..m_{l-1}}(t-U)$ est calculée aux lignes 3-5 de la figure 3.2, $\left\lceil \frac{s_k^{\tilde{n}}}{\Psi_k} \right\rceil \cdot \overline{\Psi}_k$ étant ensuite ajouté à la ligne 6.

```
1 funct time f(task k,instance n,time t)
2   w := 0;
3   for i := 1 to m_{l-1} do
4       w := w + s(i,t);
5   od
6   w := w + ceil(s_k^n/Ψ_k) * Ψ̄_k;
7   return min(w,s_k^*(t));
8 end
```

FIG. 3.2 – *Demande de travail des tâches plus prioritaires que τ_k et des tâches de la même couche RR.*

3.6 Prise en compte des sections critiques

Lorsqu'une tâche écrit dans une zone de mémoire partagée, elle ne doit pas être interrompue par une tâche pouvant lire ou écrire dans cette même zone de mémoire, celle-ci pouvant se

trouver dans un état incohérent. D'une façon plus générale, il existe des intervalles de temps, appelés des *sections critiques*, pendant lesquels une tâche occupe une ressource et ne doit pas être préemptée par des tâches utilisant la même ressource, c'est le principe de l'exclusion mutuelle. L'emploi de sémaphores est une solution classique à ce problème de partage de ressources, néanmoins les sémaphores peuvent induire des problèmes d'inversions de priorités et d'interblocages. Diverses techniques ont été développées pour résoudre ces problèmes, le lecteur pourra se référer à [Martineau, 1994] pour un état de l'art exhaustif. Dans cette section, nous nous intéresserons à une solution simple et efficace, connue sous le nom de protocole à priorité plafond (*Priority Ceiling Protocol* - PCP), proposée dans [Sha *et al.*, 1990].

3.6.1 FPP et sections critiques

Le PCP est classiquement utilisé en conjonction avec la politique FPP, nous le noterons alors PCP-FPP. Son objectif est de limiter le temps pendant lequel une tâche peut attendre une ressource utilisée par des tâches moins prioritaires.

Principe du PCP-FPP: *Lorsqu'une tâche entre dans une section critique, elle verrouille le sémaphore correspondant à sa priorité et est promue à la valeur plafond, c'est à dire la valeur maximale des priorités des tâches susceptibles d'accéder à la ressource. Une fois qu'elle est sortie de la section critique, la tâche libère le sémaphore et retrouve sa priorité initiale.*

Il a été prouvé dans [Sha *et al.*, 1990] que sous PCP-FPP, une tâche τ_k dans une période d'activité de niveau k[13] pouvait être gênée par une seule tâche de priorité inférieure dans une section critique. La formule (3.7) donnant les instants de fins d'exécution doit être aménagée pour tenir compte du PCP-FPP, elle devient

$$E_{k,n} = \min\{t > U_{k,n} \mid S_{1..k-1}(U_{k,n},t) + B_{k,n} + S_k^n = t - U_{k,n}\} \tag{3.10}$$

avec $B_{k,n}$ la longueur de l'éventuelle section critique qui est bornée par b_k, la taille maximum des sections critiques des tâches de priorité inférieure qui peuvent verrouiller une ressource utilisée par τ_k. La formule donnant la borne sur les fins d'exécution devient donc :

$$e_{k,\tilde{n}} = \min\{t > 0 \mid s_{1..k-1}(t) + b_k + s_k^n = t\}. \tag{3.11}$$

Du point de vue d'une tâche FPP, les tâches comportant des sections critiques dans des couches RR moins prioritaires se comportent exactement comme des tâches FPP moins prioritaires.

13. La définition de période d'activité (*busy-period*) dans [Sha *et al.*, 1990] est légèrement différente de celle de période d'interférence utilisée dans ce chapitre. Néanmoins, comme une période d'interférence constitue un sous-intervalle d'une période d'activité, la preuve de Sha et al. reste valable.

3.6.2 Round-Robin et sections critiques

La question qui se pose maintenant est de savoir comment la politique RR se comporte avec le PCP qui a été conçu pour FPP. Souvenons-nous que toutes les tâches d'une même couche RR ont la même priorité; promouvoir une priorité à la valeur plafond, en cas de partage de ressources au sein de la même couche RR, n'a donc strictement aucun effet et nous sommes exposés aux problèmes d'interblocage. Imaginons une couche RR constituée de 2 tâches τ_1 et τ_2 accédant chacune aux ressources ζ_1 et ζ_2 dans un ordre différent (cf. figure 3.3). Supposons que τ_1 a besoin de ζ_1 au début de son exécution et ζ_2 après Ψ_1 unités de temps d'exécution. De façon symétrique, τ_2 nécessite ζ_2 au début de son exécution et ζ_1 après Ψ_2 unités de temps d'exécution. Si τ_1 est activée à l'instant t et τ_2 légèrement plus tard, alors ζ_1 est verrouillé à l'instant t par τ_1 qui s'exécute jusque $t + \Psi_1$, c'est à dire jusqu'à la fin de son quantum. Ensuite τ_2 gagne le processeur et verrouille ζ_2. A l'instant $t+\Psi_1+\Psi_2$, τ_1 devrait être exécutée à nouveau mais a besoin de la ressource ζ_2 qui est verrouillée par τ_2; τ_1 est donc bloquée. D'un autre côté τ_2 est aussi bloquée puisqu'elle a besoin de ζ_1 qui appartient encore à τ_1 : nous nous trouvons dans une situation d'interblocage.

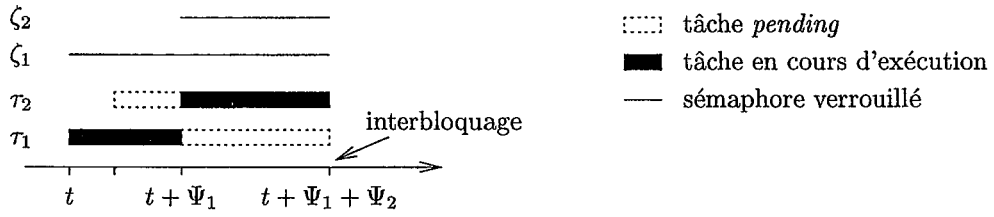


FIG. 3.3 – *Exemple d'interblocage dans une couche RR.*

Une telle situation ne peut se produire si les tâches RR ne sont pas interrompues pendant une section critique. Normalement, l'ordonnanceur RR interrompt une tâche τ_k dès que celle-ci a utilisé le processeur pendant Ψ_k. Pour empêcher cette interruption, il suffit d'ignorer le quantum lorsque la tâche est dans une section critique. Le résultat est qu'une tâche RR ne peut être interrompue que lorsqu'elle a relâchée tous ses verrous, aucun interblocage ne peut donc se produire car, à chaque instant, au plus une tâche possède une ou plusieurs ressources partagées. La règle permettant d'éviter les interblocages sous RR en présence de ressources partagées, appelé PCP-RR, peut être énoncée de la façon suivante :

PCP-RR: *La priorité d'une tâche RR entrant en section critique est promue à la valeur plafond et son quantum de temps est ignoré tant qu'elle se trouve en section critique.*

D'un point de vue pratique, deux primitives de services sont théoriquement nécessaires pour implanter PCP-RR : une pour rehausser la priorité[14] et l'autre pour modifier la valeur du

14. Sous Posix1003.1b, les fonctions *sched_setparam()* et *sched_setscheduler()* permettent ce rehaussement

quantum de temps qui malheureusement n'est généralement pas disponible[15]. Nous proposons une solution de repli qui est de réserver pour chaque couche RR une priorité immédiatement supérieure à celle de la couche qui servira de valeur plafond si la tâche de plus forte priorité utilisant la ressource se trouve dans la couche RR. Ainsi, une tâche RR promue à la priorité plafond sera la seule à ce niveau de priorité et se comportera donc exactement comme une tâche FPP, c'est à dire qu'elle possédera un quantum de temps potentiellement infini. Si la priorité plafond est celle d'une tâche FPP, par l'hypothèse de travail numéro 5, une tâche RR devra changer de politique, et donc devenir FPP pendant le temps de la section critique.

PCP-RR implique qu'une tâche τ_i peut utiliser le processeur pendant plus de Ψ_i unités de temps sans interruption. Si τ_i débute sa plus grande section critique z_i à l'instant ou son quantum expire, alors globalement elle aura utilisée le processeur pendant $\Psi_i + z_i$ unités de temps. Le fait qu'une tâche dépasse ou non son quantum dépendra du nombre, de la longueur et de la localisation des section critiques dans le code de cette tâche. Une borne sur les temps de réponse peut être néanmoins proposée en considérant le cas le plus pessimiste où toutes les autres tâches de la couche consomment $\Psi_i + z_i$ unités de temps par cycles RR, mais où la tâche considéré n'utilise que Ψ_k unités de temps. C'est une surestimation acceptable si les sections critiques sont petites en comparaison des quanta. Il ne suffit pas de considérer les sections critiques au sein de la couche, il faut aussi intégrer le retard induit par des tâches de priorité inférieure. On note b_k la plus grande section critique d'une tâche de priorité inférieure à la couche RR qui a besoin d'une ressource également utilisée par au moins une des tâches de la couche ou par une tâche dans une couche de priorité supérieure. Notons que b_k est identique pour toutes les tâches d'une même couche RR. Pour les sections critiques de tâches au sein de la couche RR, nous introduisons la notation $\overline{z}_k = \sum_{\tau_i \in \lambda_l, \tau_i \neq \tau_k} z_i$. La borne sur les instants de fin d'exécution avec PCP-RR devient :

$$e^*_{k,n} = \min\{x > 0 \mid \overline{\psi}_k(x) + b_k + s^n_k = t\} \tag{3.12}$$

avec

$$\overline{\psi}_k(x) = \min\left(\left\lceil \frac{s^{\tilde{n}}_k}{\Psi_k} \right\rceil \cdot \left(\overline{\Psi}_k + \overline{z}_k\right) + s_{1..m_{l-1}}(x),\ s^*_k(x)\right). \tag{3.13}$$

À nouveau, nous invitons le lecteur à se référer à l'annexe B.2 pour le calcul de $s^*_k(x)$ en présence de sections critiques.

Sous PCP-RR, il est inévitable que des tâches RR puissent utiliser la ressource au delà de leur quantum. L'équité entre tâches de même priorité, qui est la finalité de RR, n'est donc plus

de priorité. Par l'intermédiaire de ces fonctions, il est également possible pour une tâche de changer de politique.

15. Se référer au paragraphe "hypothèses de travail" de la Section 4.1 pour plus de détails sur les quanta de temps.

respectée. Pour rétablir cette équité, nous proposons dans [Migge *et al.*, 1999] une évolution de PCP-RR avec laquelle les tâches doivent "rembourser" les unités de temps qu'elles ont perçues au delà de leur quantum. Imaginons que dans un cycle RR, la tâche τ_k s'exécute pendant $\Psi_k + z$ unités de temps, alors au cours du prochain cycle RR, son quantum sera réduit de z unités de temps.

3.7 Étude de cas

On considère une application de contrôle-commande qui s'exécute sur une machine uniprocesseur avec un OS support se conformant à Posix1003.1b. La structure de cette application, inspirée de [Gallmeister, 1995] page 152, est représentée sur la figure 3.4.

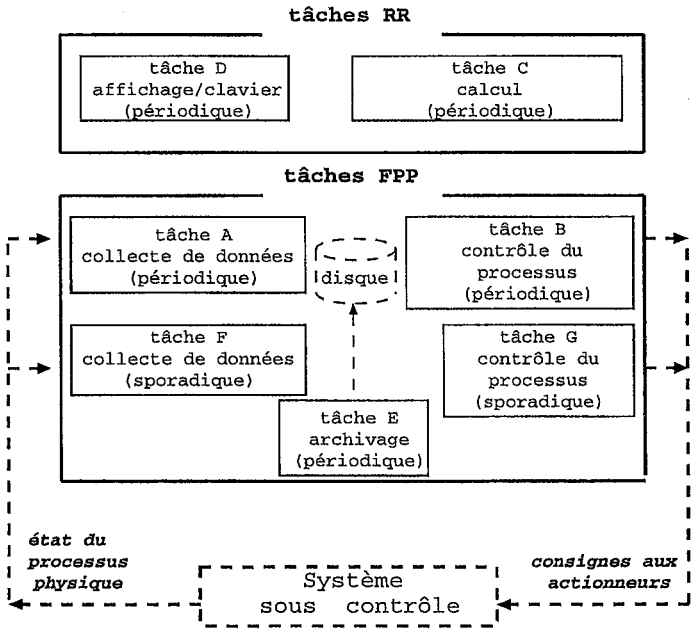


FIG. 3.4 – *Structure de l'application.*

Les informations sur l'état du système contrôlé sont collectées par la tâche A. La tâche B analyse les données et, en fonction du modèle de comportement attendu, prend un certain nombre de décisions qui sont répercutées sous la forme de consignes aux actionneurs. La tâche C est une tâche de calcul en arrière-plan qui, par exemple, tient à jour des statistiques sur l'évolution du système ou implante des techniques d'intelligence artificielle qui pourront aider la tâche B dans ses prises de décisions. L'évolution du système est affiché graphiquement pour l'opérateur de contrôle par la tâche D et archivé sur disque par la tâche E. Si le système se trouve dans un état anormal, une alarme est déclenchée et la tâche F est alors prioritairement

exécutée (cf. l'assignation des priorités dans le tableau 3.1) pour collecter des informations complémentaires. La tâche G prendra alors les décisions urgentes adéquates (ex: arrêt du système, changement de mode de marche) qui seront répercutées sur le processus physique. Toutes les tâches sont périodiques ou sporadiques et possèdent une priorité et une politique, toutes deux fixes sur la durée de vie de l'application. Dans cette étude de cas, pour rester consistant avec les notations des sections précédentes, l'allocation des priorités se fera selon le principe "plus petite la valeur numérique, plus grande la priorité de la tâche" qui est l'inverse de Posix1003.1b. Notons que dans cette étude de cas, nous avons effectué un choix a priori en ce qui concerne les priorités et les politiques des différentes tâches. Nous verrons dans le chapitre suivant comment choisir au mieux les politiques et les priorités pour une application particulière.

Les tâches C et E partagent une ressource qui est un fichier résidant en mémoire (*memory mapped file*, cf. [Gallmeister, 1995] pages 119-124), la tâche E mettant à jour régulièrement sur le disque le fichier physique sous-jacent. La durée des sections critiques, notée précédemment z_k, est de 6 ms pour la tâche C et de 15 ms pour la tâche E. Les caractéristiques des tâches et les résultats de l'analyse d'ordonnançabilité sont indiqués sur le tableau 3.1 où $\rho_k = C_k/T_k$ est le taux d'utilisation du CPU de la tâche τ_k et $\rho_{1..m}$ le taux d'utilisation cumulé de la tâche de plus forte priorité jusqu'à la tâche considérée.

couche	prio.	Ψ_k	τ_k	C_k	T_k	D_k	z_k	R_k	ρ_k	$\rho_{1..m}$
FPP	1		F	3	15	6		3	0,20	0,20
	2		G	3	15	7		6	0,20	0,40
	3		A	7	50	50		13	0,14	0,54
	4		B	6	50	50		25	0,12	0,66
RR	5	5	C	10	100	150	6	120	0,10	0,76
	5	4	D	40	500	700		277	0,08	0,84
FPP	6		E	20	500	500	15	282	0,04	0,88

TAB. 3.1 – *Les tâches composant l'application et les résultats de l'analyse d'ordonnançabilité (unité de temps: ms).*

3.8 Conclusion

Posix1003.1b ([(ISO/IEC), 1996] page xvii) définit le temps réel dans les systèmes d'exploitation comme "la capacité de l'OS à fournir une qualité de service requise dans un intervalle de temps borné". Cette exigence de "prévisibilité" des performances se retrouve au niveau applicatif et une analyse d'ordonnançabilité, tel que proposée dans ce chapitre, fournit un élément de réponse qui, dans une optique temps réel, est primordial car ce sont les performances

du pire cas envisageable qui sont évaluées.

L'analyse d'ordonnançabilité de ce chapitre est vraisemblablement faisable, et ce que nous avions envisagé initialement, sans utiliser les concepts et notations proposées dans [Migge and Jean-Marie, 1998; Migge and Jean-Marie, 1999; Migge, 1999]. Néanmoins, il apparaît très pratique en termes de réutilisabilité de pouvoir séparer la caractérisation du flux d'arrivée de travail, de l'expression du temps de réponse qui est propre à la politique considérée. Nous verrons également par la suite que ces concepts et notations se sont révélés précieux pour certaines preuves du chapitre 5. Il nous semble toutefois qu'il faudra s'interroger dans l'avenir sur le choix du bon niveau de formalisme. Doit-on rester fidèle à "l'école de York" ou se tourner vers un formalisme plus rigoureux et plus expressif mais qui d'une part nécessite un investissement plus important, et d'autre part ne facilite pas la diffusion des résultats?

Si dans la littérature les analyses d'ordonnançabilité sont pléthoriques, on peut s'étonner que jusqu'à maintenant, à notre connaissance, aucune n'ait abordé le problème des systèmes dans lesquels plusieurs politiques peuvent coexister. Dans la pratique pourtant, la grande majorité des OSs temps réel offrent, et depuis longtemps, plusieurs politiques d'ordonnancement[16]. Beaucoup de travail reste à faire dans ce domaine, en particulier se pose le problème de l'allocation des politiques aux tâches. Dans le cadre de Posix1003.1b, quel est l'apport d'un ordonnancement utilisant RR par rapport à un ordonnancement strictement FPP? Plus généralement, parmi un ensemble de solutions d'ordonnancement faisables, laquelle est la plus adaptée à l'application considérée? Dans le chapitre suivant, nous donnerons des éléments de réponse à ces questions, d'une part en définissant des critères permettant de choisir entre différentes configurations faisables, d'autre part en proposant une approche pour parcourir l'espace des solutions faisables.

16. Dans un "survey" sur les OSs temps réel paru dans [Real-Time Magazine, 1997], 50 des 56 OSs pour lesquels les mécanismes d'ordonnancement sont décrits (dans les catégories *Embedded RTOSs* et *RTOSs - Large RT Systems*) disposent d'au moins deux politiques d'ordonnancement (avec au plus 7 politiques).

Chapitre 4

Optimisation de l'ordonnancement de tâches dans les systèmes Posix1003.1b

Résumé

Une analyse d'ordonnançabilité fournit une aide appréciable au concepteur de l'application mais, en général, elle permet uniquement d'affirmer qu'un ordonnancement est *faisable* ou non, sans proposer de solutions faisables. D'autre part, comme cela a été souligné par Gerber et Hong dans [Gerber and Hong, 1997], une analyse d'ordonnançabilité n'est que de peu d'aide pour l'optimisation du système qui est pourtant souvent une étape incontournable. Nous proposons dans ce chapitre une approche basée sur un algorithme génétique pour fixer au mieux politiques d'ordonnancement et priorités dans les systèmes Posix1003.1b . D'autre part, nous montrons que l'utilisation conjointe de Round-Robin et de FPP augmente l'ordonnançabilité du système ainsi que la satisfaction de critères additionnels choisis en fonction de l'application. Ce travail a fait l'objet d'un rapport de recherche INRIA [Navet and Migge, 1999a] et d'un article en cours d'évaluation [Navet and Migge, 1999b].

4.1 Introduction

Définition du problème Une analyse d'ordonnançabilité telle que proposée dans le chapitre 3, permet simplement d'affirmer qu'une certaine allocation de politiques et de priorités est faisable ou non. Une stratégie d'allocation optimale[17], similaire à l'algorithme d'Audsley [Audsley, 1991b] qui s'applique dans le cas strictement FPP, n'existe pas quand RR et FPP sont utilisés conjointement. En fait, le problème ne consisterait pas uniquement, comme dans le cas strictement FPP, à fixer les priorités, mais aussi à choisir pour chaque tâche la politique, et pour chaque tâche sous RR, le quantum de temps. Si plusieurs allocations fai-

17. Optimal dans le sens où si une allocation faisable existe, celle-ci sera trouvée par l'algorithme.

sables existent, un tel algorithme ne nous aiderait de toutes façons pas à choisir la meilleure allocation pour l'application considérée lorsque des critères autres que la faisabilité sont pris en compte.

Dans ce chapitre, nous proposons une approche permettant d'obtenir (1) une solution faisable et (2) une solution qui se comporte bien par rapport à des critères additionnels choisis par le concepteur de l'application. Ces critères additionnels peuvent être par exemple, de minimiser les gigues sur les fins d'exécution de certaines tâches qui doivent produire leurs résultats de la façon la plus périodique possible, ou de maximiser la fraîcheur d'un ensemble de données utilisée en entrée d'un algorithme de contrôle implanté sous la forme d'une tâche périodique (cf. section 4.2.3.2).

Limites des analyses d'ordonnançabilité L'utilisation exclusive de l'analyse d'ordonnançabilité pour trouver des solutions d'ordonnancement réellement utilisables soulève un certain nombre de questions.

D'une part, dans le contexte de Posix1003.1b où les interactions entre les deux politiques à disposition sont difficilement prévisibles, il n'est pas toujours évident de trouver ne serait-ce qu'une seule solution faisable. Le problème se pose dans toute son acuité pour de grands ensembles de tâches induisant un fort taux d'occupation du processeur. Il est évidemment possible de se restreindre à l'utilisation de la politique FPP et d'appliquer l'algorithme d'Audsley, néanmoins cette solution n'est clairement pas satisfaisante car ne pas utiliser RR, c'est réduire l'ordonnançabilité globale du système comme cela sera étudié en section 4.4.1.

De plus, une analyse d'ordonnançabilité est d'une aide limitée pour l'optimisation (*"tuning"*) d'un système, d'une part à cause des hypothèses nécessairement pessimistes sur les pires temps d'exécution (la difficulté de déterminer des pires temps d'exécution "raisonnables" est discutée dans [Gerber, 1994]), et d'autre part, car ses résultats sont orientés "pire cas" (la seule trajectoire étudiée du système est la plus pessimiste en termes de flux d'arrivée de travail). Étant donné un ensemble de configurations faisables, comment choisir la meilleure uniquement au vu des résultats de l'analyse d'ordonnançabilité? A notre avis, la connaissance d'informations probabilistes sur le comportement du système est indispensable pour choisir judicieusement. Dans cette étude, ces informations probabilistes sont obtenues par simulation.

Aperçu de l'approche Compte tenu de la combinatoire du choix des priorités et des politiques pour un ensemble de tâches (cf. section 4.2.1 pour la complexité exacte du problème), l'utilisation d'une technique d'optimisation est requise. Dans cette étude, nous proposons l'utilisation d'un algorithme génétique (AG) basé sur une approche hybride détermi-

niste/stochastique :

- l'analyse d'ordonnançabilité est utilisée pour distinguer les configurations faisables des configurations non-faisables,
- la qualité de chaque configuration faisable est évaluée par simulation à l'aide de temps d'exécution stochastiques dérivés soit de mesures effectuées sur une plate-forme, soit par analyse [Lin *et al.*, 1990]. L'utilisation de la simulation est rendue nécessaire par la nature même des critères de qualité définis (cf. section 4.2.3.2).

Cette approche permettra au concepteur d'obtenir une "bonne" solution du point de vue de la satisfaction de critères dépendant de l'application, tout en préservant la faisabilité qui est essentielle dans une optique temps réel.

Étude de l'existant Dans [Tindell *et al.*, 1992], les auteurs ont proposé l'utilisation du recuit-simulé (*simulated annealing*) pour allouer les tâches d'une application temps réel sur différents noeuds interconnectés par un protocole à jeton. Le problème qui a été étudié n'est pas de choisir les priorités et les politiques (la politique est FPP pour toutes les tâches et les priorités sont assignées avec la stratégie *deadline monotonic*[18]), mais de savoir sur quels sites placer les tâches de telle façon que les contraintes physiques (mémoire vive, processeur) et les contraintes temporelles soient satisfaites. Outre de proposer une solution au problème abordé, cet article a eu pour mérite de montrer que l'emploi de techniques d'optimisation pouvait être efficace pour résoudre des problèmes d'ordonnancement temps réel. Le recuit-simulé a été également utilisé dans [DiNatale and Stankovic, 1995] pour trouver des ordonnancements faisables dans le cas distribué en minimisant les gigues de fin d'exécution des tâches. Les auteurs font l'hypothèse que les tâches sont pré-affectées sur les différents noeuds et calculent sur chaque noeud, des ordonnancements (non-préemptifs) faisables à l'aide du recuit simulé. L'optimisation, qui ne fait pas appel à la simulation, est basée sur les hypothèses de pire cas (i.e. pire temps d'exécution, pire temps de réponse).

Parmi les techniques d'optimisation existantes, nous avons choisi d'utiliser les algorithmes génétiques (AGs) car ils ont déjà été employés avec succès pour résoudre des problèmes d'ordonnancements tel que le *Job-Shop* [Davis, 1985; Nakano and Yamada, 1991; Falkenauer and Bouffoix, 1991; Djerid *et al.*, 1995; Portmann, 1996], le voyageur de commerce [Oliver *et al.*, 1985; Goldberg and Lingle, 1985], l'ordonnancement d'instructions au niveau d'un processeur [Beaty *et al.*, 1990; Beaty *et al.*, 1996] ou encore des problèmes d'ordonnancement dans des environnements multiprocesseurs [Kidwell, 1993; Schwehm and Walter, 1994; Dussa-Zieger, 1996; Kwok and Ahmad, 1997; Chen *et al.*, 1998; Dussa-Zieger and Schwehm, 1998; Tsuchiya

18. Plus l'échéance de la tâche relativement à sa date d'activation est petite, plus grande sera sa priorité.

et al., 1998]. Les performances des AGs ont été comparées avec celles d'autres techniques d'optimisation tel que la méthode Tabou (*Tabu Search*) [Beaty, 1993] ou le recuit-simulé [Thiel, 1998] et les résultats ont montré que les AGs étaient une approche "robuste" [Goldberg, 1989] qui, pour une large variété de problèmes, donne de bons résultats. Il a été montré dans [Ingber and Rosen, 1992] (cité dans [Chen *et al.*, 1998]) que les AGs étaient nettement moins sensibles à la valeur des paramètres initiaux que le recuit-simulé dont les performances sont très dépendantes du choix de ces paramètres. De plus, le recuit-simulé ne faisant évoluer qu'une seule solution, il est difficilement parallélisable.

Hypothèses de travail Dans ce chapitre nous plaçons un certain nombre de restrictions au problème de fixer les politiques et les priorités d'un ensemble de tâches périodiques sur un exécutif Posix1003.1b :

1. les gigues sur les dates d'activation des tâches ne sont pas considérées,
2. les temps de changement de contexte sont négligés,
3. les tâches ne possèdent pas de sections critiques pendant lesquelles elles ne peuvent être préemptées par d'autres tâches nécessitant la même ressource partagée,
4. les priorités utilisées doivent être contiguës (un niveau de priorité non-utilisé n'a de toutes façons aucune influence sur l'ordonnancement),
5. deux tâches sous des politiques d'ordonnancement différentes ne peuvent partager la même priorité,
6. toutes les tâches sous *sched_fifo* doivent avoir des priorités différentes qui ne change pas au cours de l'exécution de l'application (sous ces hypothèses *sched_fifo* devient FPP),
7. (a) les tâches RR ne peuvent changer leur priorité en cours d'exécution et (b) le quantum de temps est une constante globale.

L'existence de sections critiques (hypothèse 3) est prise en compte par l'analyse d'ordonnançabilité mais n'a pas été considérée dans ce chapitre par souci de simplicité. L'hypothèse 7(a) peut être relaxée en attribuant pour l'analyse d'ordonnançabilité un quantum de temps unique par tâche; néanmoins, comme à notre connaissance, aucun système d'exploitation ne possède cette fonctionnalité[19] qui n'est pas requise par le standard Posix1003.1b, nous nous restreindrons à un quantum de temps global pour toutes les couches RR. Enfin, les hypothèses 1, 2, 5, 6, 7(b) sont héritées de l'analyse d'ordonnançabilité.

19. A titre d'exemple, QNX V4.24 [QNX Software Systems, 1997] impose un quantum de temps de 50 ms (ce qui paraît bien excessif compte tenu de la rapidité des processeurs actuels) alors que VxWorks V5.3.1 [Wind Rivers Systems, 1997], avec la fonction *kernelTimeSlice()*, permet de choisir dynamiquement le quantum qui reste néanmoins commun à toutes les tâches RR.

Organisation du chapitre La section 4.2 est consacrée à la description de l'approche et de l'AG l'implantant. En section 4.3, nous évaluerons les performances de l'AG sur des problèmes de taille 20 tâches (paragraphe 4.3.1) et 30 tâches (paragraphe 4.3.2) puis proposerons une version parallélisée de l'AG. Enfin, dans la section 4.4, pour justifier de l'emploi de RR sous Posix1003.1b, nous montrerons que cette politique, contrairement à l'idée généralement admise, a son utilité dans les applications temps réel.

4.2 Choix des priorités et des politiques

Lorsque l'on traite du problème de fixer les priorités et les politiques, il existe une contrainte qu'il est indispensable de prendre en compte: il peut exister des tâches pour lesquelles la priorité et/ou la politique sont a priori imposées. En effet, dans le développement d'une application réelle, il est vraisemblable que l'on ait à utiliser des routines provenant de tiers ou du fournisseur de l'OS pour assurer par exemple des tâches de communication ou d'affichage. Typiquement, c'est le cas des drivers de périphériques qui peuvent nécessiter de s'exécuter à une priorité fixée (généralement parmi les plus hautes) et/ou avec une politique fixée (généralement *sched_fifo*).

Définition 4 *Une allocation des priorités et des politiques qui respectent les contraintes a priori de l'application est une solution possible.*

Définition 5 *Une solution possible qui passe avec succès l'analyse d'ordonnançabilité (i.e.* $\forall \tau_k \in \mathcal{T} \quad R_k \leqslant D_k$*) est une solution faisable.*

4.2.1 Complexité du problème

Si plusieurs tâches sont assignées au même niveau de priorité, alors elles sont nécessairement ordonnancées sous RR (cf. hypothèses 5 et 6). D'un autre coté, une couche RR avec une seule tâche est strictement équivalente à FPP.

Assigner n tâches à un ou plusieurs niveaux de priorités revient à diviser un ensemble de n éléments en un ou plusieurs sous-ensembles non-vides (cf. hypothèse 4). Supposons que cette partition crée k sous-ensembles. Le nombre de possibilités pour créer k sous-ensembles avec n tâches est par définition égal au nombre de Stirling du second ordre (cf. [Abramowitz and Stegun, 1970] page 824)

$$\frac{1}{k!}\sum_{i=0}^{k}(-1)^{(k-i)}\binom{k}{i}i^n.$$

Il y a $k!$ différentes possibilités d'ordonner (en termes de priorité) les k sous-ensembles de tâches. De plus, les n tâches peuvent être subdivisées en $k = 1,2,\ldots,n$ sous-ensembles. Il y a donc

$$\sum_{k=1}^{n}\sum_{i=0}^{k}(-1)^{(k-i)}\binom{k}{i}i^n$$

différentes possibilités d'assigner les tâches au(x) niveau(x) de priorité, les politiques étant induites par le nombre de tâches par niveau de priorité.

Comme on peut le voir sur la figure 4.1, l'espace des solutions croit plus qu'exponentiellement mais moins que n^n. Par exemple, pour $n = 10$, une recherche exhaustive nécessiterait plus de trois ans et deux mois en faisant l'hypothèse d'un temps de calcul de une seconde par possibilité.

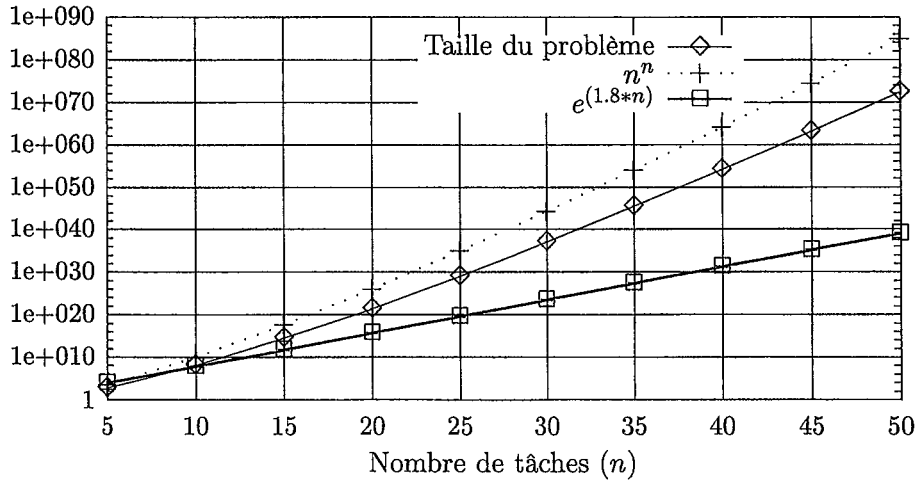


FIG. 4.1 – *Complexité du problème pour un nombre de tâches variant de 5 à 50.*

4.2.2 Principe de l'algorithme

Un projet informatique commence généralement par une description informelle des fonctionnalités et des contraintes que devra respecter le système. En particulier, pour les applications temps réel, l'accent est mis sur les contraintes temporelles. Par exemple, pour le système de contrôle d'un processus chimique décrit dans le paragraphe 4.4.2, une des contraintes est que *la température de la cuve doit être corrigée toutes les 12 ms.* En partant des fonctionnalités à offrir et des contraintes à respecter, le concepteur du système en dérive l'ensemble des tâches qui vont composer l'application.

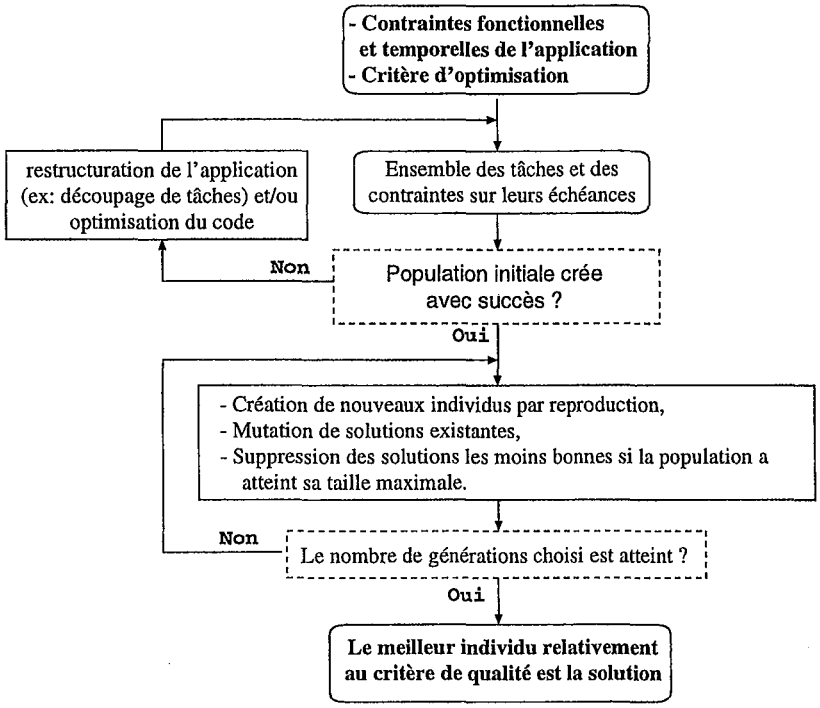


FIG. 4.2 – *Vue d'ensemble de l'approche.*

Le problème que l'on s'attachera à résoudre est d'affecter les priorités et les politiques aux tâches de telle façon (1) que la faisabilité soit garantie et (2) que le système se comporte bien relativement à des critères d'optimisation dépendant de l'application (cf. paragraphe 4.2.3.2). Nous proposons pour cela un AG qui, en faisant évoluer une population de solutions faisables, va parcourir l'espace des solutions au problème (cf. figure 4.2). Une des difficultés est de constituer la population initiale de l'AG, pour cela des heuristiques ainsi que le hasard seront utilisés (cf. paragraphe 4.2.3.5). Si une population initiale d'au moins deux solutions faisables[20] n'est pas créée, alors l'AG a échoué et le code de l'application doit être, au moins partiellement, ré-écrit pour diminuer les temps d'exécution des tâches et/ou l'application doit être restructurée, par exemple, à l'aide de la technique de segmentation de tâches proposée dans [Gerber and Hong, 1995; Gerber and Hong, 1997]. Les auteurs suggèrent que les tâches soient segmentées en une partie critique du point de vue temporel (ex: lecture de capteurs, production de consignes pour les actionneurs) et une partie non-observable (ex: mise à jour de l'état du système) moins contrainte d'un point de vue temporel. La partie non-observable constitue une nouvelle tâche, à l'échéance plus lointaine, ce qui a pour bénéfice d'augmenter l'ordonnançabilité globale du système.

20. Deux individus sont nécessaires pour appliquer l'opérateur de croisement, cf. paragraphe 4.2.3.3.

4.2.3 L'algorithme génétique

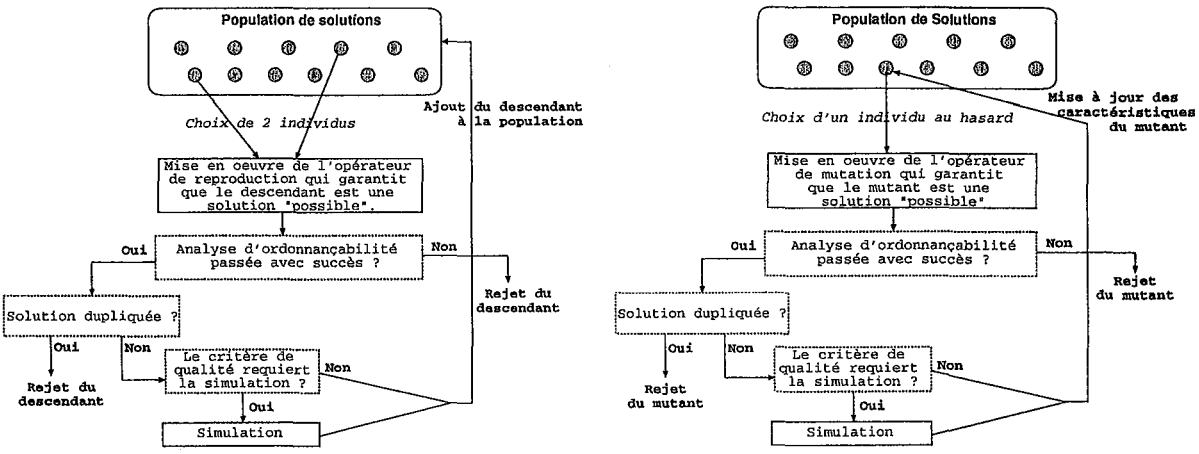
Les AGs ont été développés par J. Holland et ses collègues à l'Université du Michigan au début des années 1970 [Holland, 1975]. Ce sont des algorithmes de recherche qui explorent un espace de solutions en imitant certains processus observés dans l'évolution naturelle de populations vivantes, en particulier, la survie et la reproduction des individus les mieux adaptés à leur milieu et la disparition des plus faibles. Chaque individu de la population, qui est solution au problème étudié, est caractérisé par son empreinte génétique qui, d'un point de vue informatique, n'est rien d'autre qu'une structure de données plus ou moins complexe. La qualité ou force (*fitness*) d'un individu est mesurée par la valeur de la fonction objectif spécifique au problème traité.

La première étape d'un AG est la création de la population initiale. Ensuite chaque étape (ou génération) consiste en la création de nouveaux individus par croisement (*crossover*) et par mutations, puis en une phase de sélection à laquelle ne survivront que les meilleurs individus. Le résultat est une population qui est globalement meilleure à chaque génération avec occasionnellement une amélioration du meilleur individu. Après le nombre choisi de générations, le meilleur individu de la population sera la solution (sous-optimale) au problème. Il existe en fait une grande variété d'AGs, mais les points clés sont le choix des structures de données qui composeront le ou les chromosome(s) et la conception d'opérateurs de croisement et de mutations efficaces. Dans cette étude, nous adopterons le schéma de fonctionnement suivant inspiré de [Davis, 1991] :

1. Création de n_{ini} solutions possibles (cf. définition 4) composant la population initiale en utilisant les heuristiques *Rate Monotonic*[21] (1/4 n_{ini}), *Deadline Monotonic* (1/4 n_{ini}) ainsi que le hasard pur, cf. section 4.2.3.5. Suppression des solutions non-faisables (cf. définition 5) et des solutions dupliquées. On note p_{size} la taille de la population qui est alors $\leqslant n_{ini}$.
2. Création par croisement de m_{co} nouvelles solutions possibles qui sont appelées les descendants (*offsprings*). Si un descendant est non-faisable ou si une solution identique existe déjà dans la population, il n'est pas conservé. Dans le cas contraire, l'évaluation de sa qualité précède son incorporation dans la population. Cette étape est détaillée sur la figure 4.3(a).
3. Mutation de m_{mut} individus selon les mécanismes de la figure 4.3(b). Comme pour la phase de croisement, les solutions non-faisables ou dupliquées sont supprimées alors que les autres sont intégrées à la population.

21. Plus la période de la tâche est petite, plus grande sera sa priorité.

4. Si la population est plus nombreuse que la taille maximum désirée ($p_{size} > p_{max}$), suppression des $p_{size} - p_{max}$ moins bonnes solutions de la population.
5. Retour à l'étape 2 si le nombre de générations choisi n'est pas atteint, fin de l'AG sinon.



(a) Création d'un descendant par croisement.

(b) Mutation d'une solution existante.

FIG. 4.3 – *Les opérateurs génétiques.*

4.2.3.1 Codage des chromosomes

Un ou plusieurs chromosomes décrivent un individu, c'est à dire une solution au problème considéré. Chaque chromosome est composé de gènes dont les valeurs sont appelées les allèles. Au commencement des AGs, les chromosomes étaient codés sous la forme de champs de bits mais l'expérience a montré que lorsqu'il s'agissait de représenter les solutions de problèmes complexes, le codage binaire était nettement moins pratique à manipuler que des représentations symboliques[22] [Portmann, 1996]. Pour notre problème particulier, nous proposons de coder (de façon symbolique) l'ensemble du matériel génétique (ou génotype) d'une solution sur un seul chromosome, chaque tâche du problème étant constituée de quatre gènes :

priorité	δ_{prio}	politique d'ordonnancement	δ_{pol}

FIG. 4.4 – *Les 4 gènes caractérisant une tâche.*

22. Dans la littérature [Michalewicz, 1992], un GA dans lequel le codage des chromosomes n'est pas strictement binaire, est parfois qualifié d'algorithme "évolutionniste" (*evolutionary algorithm*).

où le gène

- priorité est la priorité de la tâche $\in \{1..card(\mathcal{T})\}$.
- δ_{prio} est une valeur binaire : | fixed | si la priorité de la tâche est une contrainte a priori
  | not_fixed | sinon
- politique est une valeur binaire : | FPP | si la tâche est FPP
  | RR | si la tâche est RR
- δ_{pol} est une valeur binaire : | fixed | si la politique de la tâche est une contrainte a priori
  | not_fixed | sinon

Un chromosome solution d'un problème à n tâches est un tableau résultant de la concaténation des différents gènes des tâches :

FIG. 4.5 – *Un chromosome C, solution d'un problème à n tâches.*

Nous définissons les fonctions suivantes qui s'appliquent sur un chromosome pour extraire certaines propriétés des tâches :

- prio(C,i) $\in \{1..card(\mathcal{T})\}$: priorité de τ_i dans le chromosome C.
- $\delta_{prio}(C,i) \in$ {fixed,not_fixed} : valeur de δ_{prio} pour la tâche τ_i dans le chromosome C.
- pol(C,i) $\in$ {FPP,RR} : politique d'ordonnancement de τ_i dans le chromosome C.
- $\delta_{pol}(C,i) \in$ {fixed,not_fixed} : valeur de δ_{pol} pour la tâche τ_i dans le chromosome C.

Pour qu'un chromosome représente une solution possible, les invariants suivants doivent toujours être vérifiés :

1. $\forall \tau_i,\tau_j \in \mathcal{T}$ avec $i \neq j$, si (pol(C,i) = pol(C,j) = FPP) alors prio(C,i) $\neq$ prio(C,j) : deux tâches FPP ne peuvent posséder la même priorité (hypothèse 6).
2. $\forall \tau_i,\tau_j \in \mathcal{T}$ si (pol(C,i) $\neq$ pol(C,j)) alors prio(C,i) $\neq$ prio(C,j) : deux tâches ordonnancées sous des politiques différentes ne peuvent partager le même niveau de priorité (hypothèse 5).
3. $\forall \tau_i \in \mathcal{T}$, prio($C$,$i$) $\leq card(T)$: pour réduire l'espace des solutions, nous imposons que toutes les priorités soient dans [1,$card(T)$] (hypothèse 4).

D'un point de vue implémentation, un chromosome est un tableau de gènes, chaque gène $C[i]$ étant une structure composée de la priorité ($C[i].prio$), de la valeur de δ_{prio} ($C[i].\delta_{prio}$), de la politique ($C[i].pol$) et de la valeur de δ_{pol} ($C[i].\delta_{pol}$). Pour chaque individu de la population,

en plus du chromosome, une donnée de type Réel est nécessaire pour sauvegarder la qualité de l'individu une fois celle-ci calculée. Nous définissons la fonction fitness() qui prend un chromosome en argument et retourne la valeur de la fonction objectif.

4.2.3.2 Fonctions objectifs

La fonction objectif permet d'évaluer la qualité de chaque individu de la population en fonction d'un ou de plusieurs critères combinés. Il est certain qu'un critère d'optimisation est très dépendant de l'application considérée, néanmoins nous avons identifié les 3 critères suivants dont la satisfaction est souhaitable dans un grand nombre d'applications, en particulier les applications de contrôle-commande :

- Minimisation de la gigue sur les fins d'exécution : après l'analyse d'ordonnançabilité, nous savons que les instants de fin de deux instances successives de la même tâche τ_i sont séparés d'un temps variant de $T_i - R_i + C_i$ à $T_i + R_i - C_i$. Si le travail de τ_i doit être délivré le plus régulièrement possible, l'objectif est de minimiser ces variations ou gigues. Dans ce cas, la fonction objectif, notée $\mathcal{C}^-$ dont il faudra minimiser la valeur, peut être définie comme

$$\mathcal{C}^- = \sum_{i \in \mathcal{T}} \sigma(\widetilde{R}_i) \cdot \Phi_i \tag{4.1}$$

  avec $\widetilde{R}_i = \{R_{i,n} \mid n \vdash E_{i,n} \leq t_{sim}\}$, l'échantillon des temps de réponse des instances de la tâche τ_i collectés pendant une simulation de durée t_{sim}, avec $\sigma()$ la fonction calculant l'écart type d'un échantillon et Φ_i, le poids donné à la tâche τ_i dans le calcul (ex: $\Phi_i = 0$ signifie que τ_i n'est pas prise en compte).
- Maximisation de la fraîcheur des données : soit une tâche τ_i qui nécessite des données produites par un sous-ensemble de tâches $\mathcal{T}_i$. D'une façon générale, les données produites par une instance $\tau_{h,a}$ d'une tâche $\tau_h \in \mathcal{T}_i$ n'est disponible qu'après la fin de son exécution, $E_{h,a}$. Pour être utilisable par une instance $\tau_{i,n}$, la donnée doit être disponible avant le début d'exécution de $\tau_{i,n}$ noté $B_{i,n}$. Plus le temps entre $E_{h,a}$ et $B_{i,n}$ est court, plus la donnée sera récente et donc généralement pertinente. La fraîcheur des données en entrée d'une tâche τ_i peut être mesurée par la fonction objectif suivante qu'il faudra minimiser

$$\mathcal{C}_i^- = \sum_{n \vdash E_{i,n} \leq t_{sim}} \sum_{\tau_h \in \mathcal{T}_i} \sum_{a \vdash E_{h,a} \leq B_{i,n} < E_{h,a+1}} (B_{i,n} - E_{h,a}) \tag{4.2}$$

  Avec plusieurs tâches de poids éventuellement différents, la fonction objectif devient

$$\mathcal{C}^- = \sum_{\tau_i \in \mathcal{T}} \mathcal{C}_i^- \cdot \Phi_i. \tag{4.3}$$

- Maximisation de la cohérence temporelle de données : lorsqu'une tâche τ_i a besoin de plusieurs données $X_1,..,X_n$, les dates de production de ces données doivent être les plus proches possibles de façon à ce que les données soient les plus cohérentes d'un point de vue temporel. On fait l'hypothèse que les données $X_1,..,X_n$ ne sont disponibles qu'à la fin d'exécution des tâches productrices dont l'ensemble est noté $\mathcal{I}_i$. Pour $\tau_k \in \mathcal{I}_i$, on définit :

$$j_{k,i,n} = \max\{j \mid E_{k,j} \leqslant B_{i,n}\}$$

qui est l'indice de la dernière instance de τ_k ayant terminé son exécution avant la $n^{\text{ième}}$ instance de τ_i. La fonction objectif, qui doit être minimisée, peut s'exprimer sous la forme

$$\mathcal{C}_i^- = \sum_{n \vdash B_{k,n} \leq t_{sim}} \sigma(\widetilde{E}_{i,n}) \tag{4.4}$$

où $\sigma(\widetilde{E}_{i,n})$ est l'écart type de $\widetilde{E}_{i,n} = \{E_{k,j_{k,i,n}} \mid \tau_k \in \mathcal{I}_i\}$, l'échantillon constitué des dates de fin d'exécution de toutes les tâches dont les résultats sont utilisés par $\tau_{i,n}$. Si on doit maximiser la cohérence des données en entrée de plusieurs tâches, la fonction objectif devient

$$\mathcal{C}^- = \sum_{\tau_i \in \mathcal{T}} \mathcal{C}_i^- \cdot \Phi_i. \tag{4.5}$$

Deux autres fonctions objectifs envisageables sont la minimisation du temps de réponse moyen ou maximum. Notons que l'on peut facilement construire des fonctions objectifs qui seraient des compromis entre plusieurs critères d'optimisation. La fonction objectif pouvant être évaluée par simulation, et c'est le cas pour les trois fonctions précédemment détaillées, il est possible de construire des critères beaucoup plus fins comme "minimiser la fraîcheur de la donnée X_1 produite par la tâche τ_k après un temps d'exécution de t_1 unités de temps et consommée par τ_i après un temps d'exécution de t_2 unités de temps.

4.2.3.3 Opérateur de croisement

L'opérateur de croisement, appelé aussi opérateur de reproduction, s'applique sur des couples d'individus sélectionnés avec une probabilité fonction de leur qualité. Ce point est crucial car dans le processus de sélection naturelle, il est généralement fait l'hypothèse que de bons parents créeront des descendants à leur image. La technique que nous utilisons pour choisir les paires de géniteurs est basée sur le principe d'une roulette [Goldberg, 1989] sur

laquelle chaque case correspondrait à un individu et où la probabilité de tirer une case est fonction de la qualité de l'individu :

1. Classer les nb chromosomes composant la population dans un ordre quelconque $C_1, C_2, \ldots C_{nb}$.
2. Calculer $S = \sum_{j=1}^{nb} \text{fitness}(C_j)$, la somme des valeurs de la fonction objectif sur toute la population.
3. Générer au hasard un nombre réel $\alpha \in [0,1]$.
4. Sélectionner un chromosome comme suit :
   - Si la valeur de la fonction objectif doit être maximisée, choisir C_i tel que :

$$\frac{\sum_{j=1}^{i-1} \text{fitness}(C_j)}{S} \leqslant \alpha < \frac{\sum_{j=1}^{i} \text{fitness}(C_j)}{S}$$

     pour $i = nb$, l'inégalité de droite devient $\leq$.
   - Si la valeur de la fonction objectif doit être minimisée, trouver $max_{fit} = \max\{1 \leqslant i \leqslant nb \mid \text{fitness}(C_i)\}$, et choisir C_i tel que :

$$\frac{\sum_{j=1}^{i-1} (max - \text{fitness}(C_j))}{nb \cdot max_{fit} - S} < \alpha \leqslant \frac{\sum_{j=1}^{i} (max - \text{fitness}(C_j))}{nb \cdot max_{fit} - S}$$

     l'inégalité de gauche devient $\leq$ pour $i = 1$.
5. Retour à l'étape 3 si les deux chromosomes n'ont pas été encore sélectionnés.

Une fois les deux géniteurs sélectionnés, nous appliquons un opérateur de croisement à deux points inspiré du célèbre *Partially Mapped X* (PMX) opérateur proposé pour le problème du voyageur de commerce dans [Goldberg and Lingle, 1985]. Le croisement des individus C_0 et C_1 crée un descendant C_2 selon la procédure :

1. Choix au hasard de deux entiers α_1, α_2 tels que $1 \leqslant \alpha_1 \leqslant \alpha_2 \leqslant n$ où n est le nombre de tâches du problème.

2. Le croisement s'applique entre les gènes α_1 et α_2 de la façon suivante :

```
1  for i := α1 to α2 do
2      if δprio(C0,i) = not_fixed
3          then C2[i].prio := random(C0[i].prio, C1[i].prio, 0,5);
4               C2[i].δprio := not_fixed;
5          else C2[i].prio := C0[i].prio;
6               C2[i].δprio := fixed;
7      fi
8      if δpol(C0,i) = not_fixed
9          then C2[i].pol := random(C0[i].pol, C1[i].pol, 0,5);
10              C2[i].δpol := not_fixed;
11         else C2[i].pol := C0[i].pol;
12              C2[i].δpol := fixed;
13     fi
14 od
```

Où random(a,b,α) est une fonction qui retourne la valeur a avec une probabilité α et la valeur b avec la probabilité $1-\alpha$.

3. Les gènes $C_2[1]..C_2[\alpha_1-1]$ sont tous pris du même parent, C_0 ou C_1 avec une probabilité de 0,5.

4. Les gènes $C_2[\alpha_2+1]..C_2[n]$ sont tous pris du même parent, C_0 ou C_1 avec une probabilité de 0,5.

5. "Réparation" du chromosome C_2 si les invariants 1,2 ou 3 ne sont pas vérifiés. La prédominance est donnée aux gènes nouvellement créés, qui se situent entre α_1 et α_2 dans le chromosome C_2, par rapport aux gènes préexistants. Cette étape garantit que le chromosome C_2 est une solution possible au problème. La procédure de réparation est détaillée dans l'annexe C.1.

4.2.3.4 Opérateur de mutation

Pour éviter le problème de dégénérescence qui peut se produire lors de croisement multiples d'individus d'une même population confinée, on introduit une étape de mutation. Les individus sur lesquels s'appliquent la mutation sont choisis au hasard, exceptée toutefois la meilleure solution qui est exclue du processus de mutation. Une fois un chromosome C choisi, l'opérateur

de mutation fait évoluer ses gènes de la façon suivante :

1.
```
1 repeat
2   i := random(1,n);
3 until (δprio(C,i) = not_fixed) ∨ (δpol(C,i) = not_fixed);
4 if δprio(C,i) = not_fixed
5   then C[i].prio := uniform(1,n);
6 fi
7 if δpol(C,i) = not_fixed
8   then C[i].pol := random(RR,FPP,0.5);
9 fi
```

où uniform(a,b) est une fonction qui retourne un entier tiré au hasard selon une loi uniforme entre a et b.

2. "Réparation" du chromosome C si les invariants 1,2 ou 3 ne sont pas vérifiés. Comme pour l'opérateur de croisement, la prédominance est donnée aux gènes nouvellement créés (cf. annexe C.1).

4.2.3.5 Création de la population initiale

La création de la population initiale de solutions est une étape cruciale de l'algorithme. En effet, si après un certain nombre d'essais successifs (valeur de n_{ini}), moins de deux solutions[23] ont été trouvées, l'opérateur de croisement ne peut être appliqué et l'algorithme a échoué. Dans ce cas, comme mentionné au paragraphe 4.2.2, l'application doit être restructurée ou au moins partiellement ré-écrite. La population initiale est composée d'individus créés à l'aide des heuristiques *Rate Monotonic* et *Deadline Monotic* ainsi que de solutions créées aléatoirement :

- Heuristique *Rate Monotonic* (RM) : les priorités sont fixées selon le principe "plus petite la période de la tâche, plus grande la priorité", les politiques sont ensuite assignées aléatoirement.
- Heuristique *Deadline Monotonic* (DM) : les priorités sont fixées selon le principe "plus petite l'échéance de la tâche (relativement à son instant d'arrivée), plus grande la priorité", les politiques sont ensuite assignées aléatoirement.
- Création aléatoire : les priorités ainsi que les politiques sont fixées aléatoirement.

Comme nous le verrons au paragraphe 4.3.2, les heuristiques RM et DM, bien que fort simples, se sont avérées efficaces dans nos expérimentations.

23. Bien que cela n'ait pas été implanté, il serait envisageable dans le cas où une seule solution a été trouvée, de ne pratiquer que des mutations jusqu'à ce qu'une deuxième solution voit le jour.

4.3 Implémentation et expérimentations

L'approche développée dans ce chapitre a été implantée sous la forme de trois exécutables distincts écrits en C++: un pour l'algorithme génétique ($\approx$ 4200 lignes de code), un pour la simulation de l'ordonnancement de tâches sous Posix1003.1b ($\approx$ 3600 lignes de code) et le programme d'analyse d'ordonnançabilité[24]. L'algorithme génétique appelle l'analyseur d'ordonnançabilité pour déterminer si une solution est faisable et, le cas échéant, la simulation qui permettra d'évaluer la qualité de cette solution est effectuée (cf. figure 4.3). Beaucoup de soins ont été apportés à l'écriture du code de simulation, en effet, c'est de très loin le programme qui nécessite le plus de temps de calcul. Ainsi, il a été écrit sans recours à une bibliothèque de simulation et les dates d'activation de tâches sont précalculées, dans la limite de la mémoire disponible, et stockées dans l'ordonnanceur de la simulation.

Une des difficultés a été pour nous de fixer les paramètres de la simulation, à savoir le nombre de trajectoires à simuler[25] et la durée de simulation. Il nous a fallu trouver un compromis entre précision des résultats et temps de simulation. Nous considérons qu'un résultat est précis si d'autres simulations effectuées dans les mêmes conditions (à l'exception de la valeur d'initialisation du générateur de nombres aléatoires qui doit être différente sous peine de retrouver des résultats strictement identiques) donnent des résultats proches. Pour les expérimentations des paragraphes 4.3.1 et 4.3.2, nous avons simulé chaque ordonnancement solution au problème sur 10 trajectoires ayant chacune une durée de 10 PPCMs des périodes avec des résultats satisfaisants en termes de précision. Dans les expérimentations de la section 4.4, les périodes sont générées aléatoirement et le PPCMs peut, par conséquent, devenir prohibitivement grand. Après différents essais, nous avons fixé le nombre de trajectoires à 10 avec chacune une durée de 50 fois la plus grande période de toutes les tâches.

Il est toujours délicat d'évaluer les performances d'un algorithme d'optimisation étant donné que le résultat optimal d'un jeu d'essais dont la taille justifie l'emploi d'une technique d'optimisation n'est en général pas connu. Dans notre cas, nous choisissons d'évaluer les performances de l'AG (qui sera noté par la suite F-GA pour *Full Genetic Algorithm*) avec une version faible du même algorithme (notée W-GA pour *Weak Genetic Algorithm*) qui ne possède pas les opérateurs de croisement et de mutation. Cela nous permettra d'estimer le gain apporté par l'AG par rapport à un simple parcours aléatoire de l'espace des solutions. Dans W-GA, excepté pour la création de la population initiale, les nouveaux individus sont créés

24. Écrit par J. Migge et disponible à l'adresse *http://www.inria.fr/mistral/personnel/Jorn.Migge/rts.html.*

25. Dans le cas général, on ne peut faire l'hypothèse que les tâches sont toutes activées pour la première fois simultanément, il faut donc simuler plusieurs trajectoires avec diverses décalages entre les premières dates d'activation.

exclusivement aléatoirement. Le schéma de fonctionnement de W-GA est le suivant :

1. Création de n_{ini} solutions possibles pour la population initiale avec l'heuristique RM (1/4 n_{ini}), l'heuristique DM et le hasard pur (1/2 n_{ini}). Suppression des individus non-faisables ou dupliqués puis évaluation de la fonction objectif. Soit p_{size} la taille de la population qui est $\leq n_{ini}$.
2. Création aléatoire de $m_{co}+m_{mut}$ nouvelles solutions possibles. Suppression des individus non-faisables ou dupliqués puis évaluation de la fonction objectif avant incorporation dans la population.
3. Si la population est plus nombreuse que la taille maximum désirée ($p_{size} > p_{max}$), suppression des $p_{size} - p_{max}$ moins bonnes solutions de la population.
4. Retour à l'étape 2 si le nombre de générations choisi n'est pas atteint, fin de W-GA sinon.

Les expérimentations des paragraphes 4.3.1 et 4.3.2 ont été effectuées avec les paramètres $n_{ini} = 50$, $m_{co} = 40$, $m_{mut} = 20$, et $p_{max} = 100$, le nombre de générations successives étant fixé à 100. Le temps d'exécution d'une tâche τ_k pendant la simulation est supposé suivre une loi uniforme entre $[C_k/2,C_k]$ où C_k est le pire temps d'exécution de τ_k considéré dans l'analyse d'ordonnançabilité.

4.3.1 Performance de l'algorithme sur un problème à 20 tâches

Le jeu d'essai comportant 20 tâches est décrit sur le tableau C.1(a) de l'annexe C.2. L'utilisation totale du processeur est de 85,7% et l'objectif est de minimiser la gigue de fin d'exécution des tâches τ_9, τ_{10}, τ_{11}, τ_{12}, τ_{13}, τ_{14}, τ_{15}, τ_{16}, τ_{17}, τ_{18}, τ_{19} et τ_{20}, chaque tâche ayant le même poids dans le calcul de la fonction objectif (i.e. dans l'équation 4.1 $\forall i \in \{9,10,11,12,13,14,15,16,17,18,19,20\}$ $\Phi_i = 1$, sinon $\Phi_i = 0$). Certaines contraintes a priori sont imposées : les tâches τ_1 et τ_2 sont FPP avec respectivement les priorités 1 et 2, les tâches τ_{15} et τ_{20} sont ordonnancées sous RR au plus faible niveau de priorité du système et la priorité 3 est assignée à la tâche τ_6. Le quantum de temps pour RR est de 2 unités de temps.

Sur la figure 4.6, on observe une amélioration importante de la force moyenne des individus jusqu'à environ la trentième génération avec F-GA. Ensuite, la population converge, ce qui signifie que les individus deviennent très similaires en termes de qualité et des améliorations significatives de la meilleure solution deviennent plus improbables.. La convergence de l'algorithme à la $i^{\text{ième}}$ génération peut se mesurer par l'écart type de l'échantillon des valeurs de la fonction objectif noté σ_i, ainsi, à la fin de la première génération $\sigma_1 = 9{,}75$ alors que $\sigma_{31} = 0{,}6$ et $\sigma_{100} = 0{,}31$. Avec W-GA, la valeur moyenne s'améliore plus lentement, seulement 9% après 100 générations contre 36,1% avec F-GA. Cela nous suggère que nos opérateurs

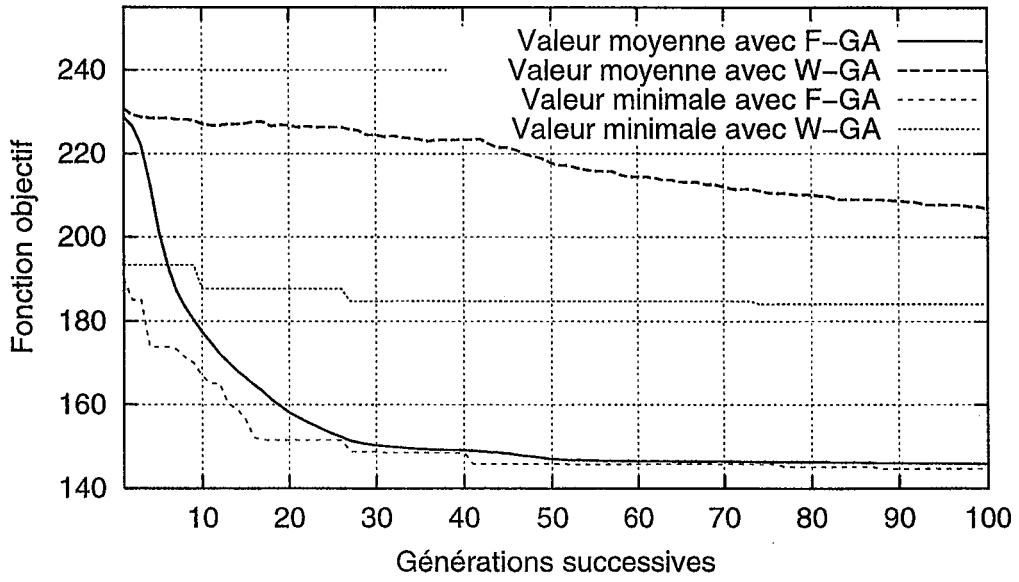


FIG. 4.6 – *Évolution de la valeur moyenne et de la valeur minimale de la fonction objectif pendant 100 générations sur un problème à 20 tâches pour F-GA et W-GA.*

génétiques sont nettement plus efficaces qu'une simple recherche aléatoire dans l'espace des solutions. Finalement, la meilleure solution avec F-GA, qui est décrite sur le tableau C.1(b) de l'annexe C.2, est 21,5% meilleure que celle trouvée avec W-GA.

4.3.2 Performance de l'algorithme sur un problème à 30 tâches

La description du problème à 30 tâches est donnée sur le tableau C.2(a) de l'annexe C.3. Les 30 tâches induisent un taux d'utilisation du processeur de 82,8% et l'objectif est de minimiser la gigue sur fin d'exécution de l'ensemble des tâches du système qui ont toutes le même poids dans le calcul de la fonction objectif. Il est imposé que les tâches τ_{20} et τ_{21} soient RR avec la plus faible priorité, que τ_{13} et τ_{14} soient FPP et que τ_1, τ_2 et τ_3 aient respectivement les priorités 1,2 et 3 sous FPP. Le quantum de temps pour RR étant toujours de 2 unités de temps.

La figure 4.7 montre que W-GA donne de très mauvais résultats sur ce problème à 30 tâches. En fait, toutes les solutions de W-GA ont été trouvées avec les heuristiques de la population initiale. Cela nous permet de penser que (1) les heuristiques proposées sont efficaces et que (2) une recherche aléatoire dans l'espace des solutions est complètement inadaptée pour des problèmes de taille importante. Comparée avec le problème à 20 tâches, la vitesse de convergence avec F-GA est plus lente, ce qui nous indique logiquement que plus la taille

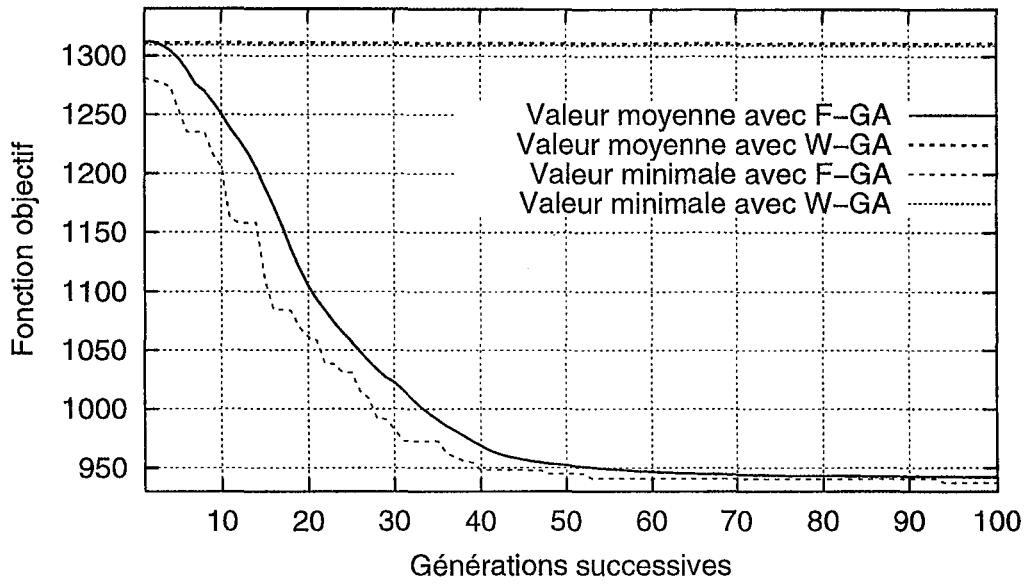


FIG. 4.7 – *Évolution de la valeur moyenne et de la valeur minimale de la fonction objectif pendant 100 générations sur un problème à 30 tâches pour F-GA et W-GA.*

du problème est grande, plus le nombre de générations doit être important. Finalement, la valeur moyenne des individus de la $100^{\text{ième}}$ génération de F-GA est 28,1% meilleure que celle des individus de W-GA, avec un gain de 28,4% pour F-GA en termes de meilleure solution au problème (cf. tableau C.2(b) en annexe C.3).

4.3.3 Parallélisation de l'algorithme

L'approche développée dans ce chapitre est coûteuse en temps de calcul; le jeu d'essais comportant 30 tâches a par exemple nécessité près de deux jours de calcul sur une station *SUN ultra 5*. Il n'est d'une part pas indispensable d'exécuter l'AG sur 100 générations. En effet, la population de solutions converge bien avant (vers la $40^{\text{ième}}$ génération sur le problème à 30 tâches) et après convergence, les améliorations que l'on peut attendre sont limitées. Une autre possibilité est de réduire le temps de simulation pour l'évaluation du critère de qualité, la contrepartie étant que les résultats perdent en précision et donc en validité. Il est toutefois clair que l'approche proposée n'est pas adaptée pour des reconfigurations en-ligne et le temps de calcul requis peut parfois poser problème même lors de la phase de conception du système, c'est pourquoi nous avons envisagé de paralléliser l'AG.

Les AGs, an tant qu'abstractions du processus d'évolution naturel par essence hautement parallèle, sont très facilement parallélisables. De nombreux travaux ont été réalisés dans ce

domaine et le lecteur pourra se référer à [Cantù-Paz, 1995] pour un état de l'art sur les algorithmes génétiques parallèles (AGPs). Trois approches distinctes existent :

- Parallélisation globale : dans ce modèle, il n'y a qu'une seule population de solutions mais l'évaluation de la qualité des individus et éventuellement l'application des opérateurs génétiques sont fait en parallèle. La communication entre les processeurs se fait au début et à la fin de l'exécution des fonctions que l'on a distribuées. Le gain à espérer est sous-linéaire en le nombre de processeurs impliqués.
- Parallélisation à grains épais[26] (*Coarse grained parallelization*) : la population totale est partitionnée en différentes sous-populations qui évoluent sur des processeurs distincts. Les sous-populations échangent occasionnellement (ex: toutes les n générations, toutes les t unités de temps) des individus. Ce modèle introduit un opérateur de migration pour transmettre des individus d'une sous-population à l'autre.
- Parallélisation à grains fins (*Fine grained parallelization*) : cette troisième approche est similaire à la seconde dans le sens où plusieurs sous-populations coexistent. La différence tient au nombre de sous-populations qui est beaucoup plus élevé rendant ce modèle bien adapté aux ordinateurs massivement parallèles.

Nous avons choisi l'approche à grains épais car elle possède des avantages importants : d'une part, son implantation ne pose pas de difficultés majeures, ensuite elle ne requiert pas de machines particulières et enfin le gain que l'on peut espérer n'est pas uniquement en vitesse d'exécution mais aussi dans la qualité des résultats. En effet, les différentes sous-populations explorerons, selon toute vraisemblance, des régions différentes de l'espace des solutions et la migration entre les populations sera source de diversité génétique. Cette migration est contrôlée par différents paramètres : la topologie qui définit les interconnexions entre sous-populations, la taux de migration qui spécifie le nombre d'individus qui doivent migrer et enfin la fréquence des migrations. Pour notre algorithme, chaque sous-population met à disposition de toutes les autres un dixième de ses individus à la fin de chaque génération, la moitié des individus étant choisie au hasard, l'autre moitié étant constituée des meilleurs de leur population. Pour chaque population, les valeurs des paramètres n_{ini}, m_{co}, m_{mut} et p_{max} restent inchangées.

Une question importante est de savoir si l'existence de plusieurs sous-populations améliore les performances globales de l'algorithme. Pour cela, nous avons comparé les résultats des versions parallèles et non-parallèles sur le jeu d'essai à 30 tâches après le même nombre de générations : 2 populations sur 50 générations sur 2 processeurs distincts pour la version parallèle contre une population sur 100 générations pour la version non-parallèle.
Sur la figure 4.8, on observe que la version non-parallèle se comporte mieux en termes de

26. La taille du grain correspond au ratio entre le temps de calcul et le temps de communication, un grain épais équivalant à un ratio élevé.

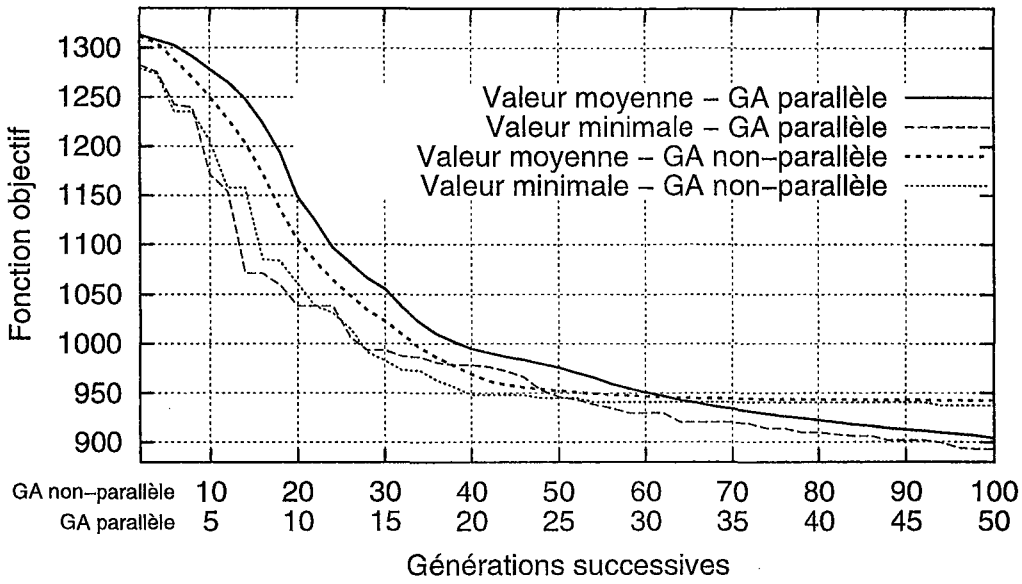


FIG. 4.8 – *Comparaison de l'évolution de la valeur moyenne et de la meilleure valeur de la fonction objectif entre la version parallèle (2 sous-populations) et non-parallèle de F-GA sur 100 générations.*

meilleure valeur de la fonction objectif jusqu'à la $50^{\text{ième}}$ génération et jusqu'à la $62^{\text{ième}}$ pour la valeur moyenne. Ensuite, les 2 populations de l'AGP commencent à converger et de meilleurs résultats sont trouvés. Finalement, après 100 générations, l'AGP fournit une amélioration de 4,72% pour la meilleure valeur et de 4,06% pour la valeur moyenne. Cela nous suggère que l'emploi de l'AGP est avantageux, même sur une machine monoprocesseur, à condition de l'exécuter suffisamment longtemps pour laisser le temps aux différentes sous-populations de converger. Des expérimentations complémentaires menées avec 3 et 4 sous-populations confirment que la version parallèle de l'AG se comporte mieux dès que chacune des sous-populations a convergé.

4.4 Round-Robin: une politique pour le temps réel?

A notre connaissance, l'utilisation de la politique RR pour l'ordonnancement de tâches dans les applications temps réel n'a jamais pas été sérieusement envisagée. Traditionnellement, l'utilisation de RR est cantonnée à des tâches de faibles priorités "quand il n'y a rien de plus important à faire" (cf. [Gallmeister, 1995] page 163). Dans cette section, notre objectif est de montrer que l'emploi de RR ne doit pas être écarté a priori car il existe des situations dans lesquelles RR est un choix efficace tant du point de vue de l'ordonnançabilité du système que du point de vue de son optimisation.

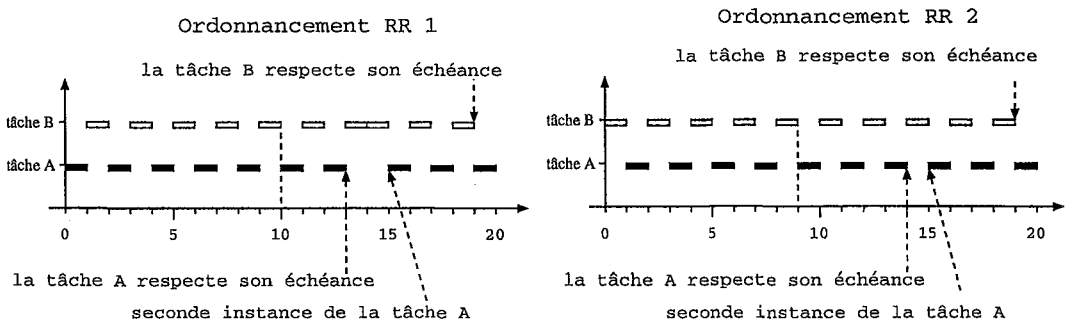
4.4.1 Round-Robin et ordonnançabilité

Dans le cadre de Posix1003.1b, la disponibilité de RR augmente l'ordonnançabilité globale d'un système simplement car il existe des ensembles de tâches qui ne sont pas ordonnançables sans RR. A titre d'exemple, imaginons une application composée de deux tâches A et B respectivement de périodes 15 ms et 50 ms, de temps d'exécution 7 ms et 10 ms et d'échéances 15 ms et 20 ms. Il a été montré dans le chapitre précédent que pour trouver les temps de réponse maximaux, et donc pour décider de l'ordonnançabilité ou de la non-ordonnançabilité d'un système, il suffisait d'examiner la première période d'interférence du scénario d'activation majorant (qui correspond à une trajectoire commençant à un instant critique [Liu and Layland, 73], c'est à dire un instant auquel toutes les tâches sont activées simultanément). La figure 4.9(a) représente l'ordonnancement des tâches A et B sous RR au même niveau de priorité à partir d'un instant critique avec un quantum de temps de 1 ms. Généralement, l'ordonnanceur RR est implémenté de telle façon qu'à un même niveau de priorité, la première tâche qui entre dans la file d'attente, se voit attribuer le premier quantum de temps (pour QNX Neutrino par exemple cf. [QNX Software Systems, 1999]). Quelle que soit la première tâche, sous RR le système est ordonnançable. Au contraire, sous FPP aucun ordonnancement n'est faisable comme représenté sur la figure 4.9(b).

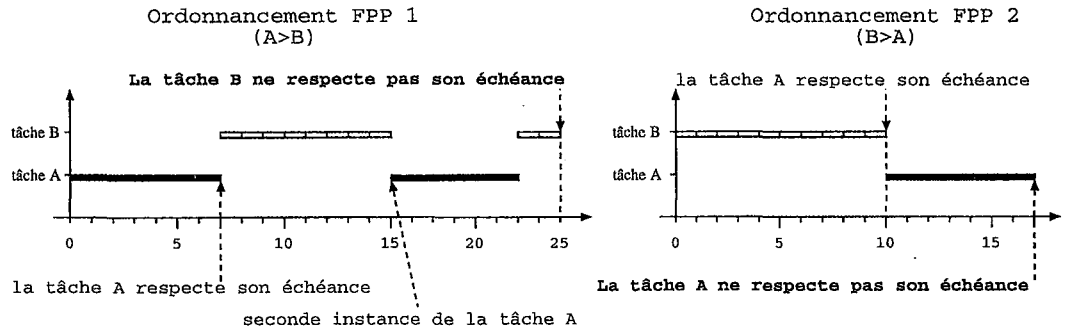
Cet exemple simple illustre le fait que l'emploi de RR peut rendre faisables des ensembles de tâches qui ne le seraient pas sous FPP. Néanmoins d'une façon générale, c'est l'utilisation combinée de RR et de FPP qui est la plus efficace. Pour estimer le gain en ordonnançabilité que l'on peut attendre de RR plus FPP par rapport à la politique FPP seule, nous avons généré au hasard 2000 ensembles de 10 tâches non-faisables sous FPP et, à l'aide des heuristiques de la population initiale de l'AG, avons cherché des solutions faisables avec RR plus FPP.

Pour générer des ensembles de tâches non-faisables, nous avons supposé l'échéance de chacune des tâches égale à sa période, et sous cette hypothèse, il a été prouvé [Liu and Layland, 73] que la stratégie d'allocation des priorités RM est optimale sous FPP, ce qui signifie que si RM ne conduit pas à une solution faisable, il n'en existe pas. Les périodes des tâches sont choisies aléatoirement entre 50 et 1000 unités de temps; les temps d'exécution étant ensuite ajustés de telle façon que la charge totale se situe entre 75% et 90%. Pour chaque ensemble de tâches non-faisable sous FPP, 2000 tentatives pour trouver une solution faisable avec FPP plus RR ont été effectuées à l'aide des heuristiques du paragraphe 4.2.3.5, le quantum de temps pour RR étant fixé à 2 unités de temps.

Dans ces conditions, pour 15,3% des ensembles de tâches (306 sur 2000), au moins une solution a été trouvée avec RR plus FPP et il est très probable que certaines solutions n'aient pas été trouvées. Des expérimentations préliminaires suggèrent que l'efficacité de RR plus FPP



(a) Les tâches A et B ordonnancées sous RR.



(b) Les tâches A et B ordonnancées sous FPP.

FIG. 4.9 – *Exemple de 2 tâches faisables sous RR et non-faisables sous FPP.*

en termes d'ordonnançabilité peut être grandement améliorée en individualisant le quantum de temps pour chaque tâche RR. Une heuristique simple, qui a donné de bons résultats sur quelques jeux d'essais mais dont l'efficacité demande à être démontrée dans des expérimentations plus poussées, est de fixer le quantum de temps d'une tâche τ_k proportionnellement au rapport C_k/D_k.

4.4.2 Round-Robin pour l'optimisation du système

Notre objectif est maintenant d'évaluer dans quelle mesure l'utilisation conjointe de RR et de FPP est meilleure que FPP seule pour l'optimisation du système selon les critères définis dans la section 4.2.3.2. Dans un système de contrôle-commande, l'emploi de RR peut être dans certains cas un moyen efficace de réduire le délai entre des mesures effectuées sur le système par des tâches différentes et ce, sans avoir besoin de recourir à des techniques de découpage de tâches [Klein *et al.*, 1993]. De la même façon, RR peut aider à minimiser le délai entre les

résultats (ex: consignes aux actionneurs) de plusieurs tâches.

Imaginons par exemple un processus chimique contrôlé par une application constituée de trois tâches périodiques (cf. figure 4.10) . Les tâches A (C=4 ms, T=12 ms, D=12 ms) et B (C=4 ms, T=12 ms, D=12 ms) mesurent et régulent respectivement la température et la pression dans la cuve. La collecte de ces grandeurs est effectuée au début de l'exécution des deux tâches et nécessite moins de 1 ms. Température et pression doivent évoluer dans le temps et l'objectif des tâches A et B est de garder leur valeur le plus proche possible d'un optimum. A la fin de leur exécution, après avoir donné les consignes aux actionneurs, les tâches A et B mettent à la disposition de la tâche C (C=2 ms, T=12 ms, D=12 ms), ordonnancée sous FPP avec une priorité inférieure à A et B, la température et la pression courante. La tâche C est responsable de l'injection de la bonne quantité de produits chimiques dans la cuve au regard de l'état courant (i.e. température, pression) de la réaction.

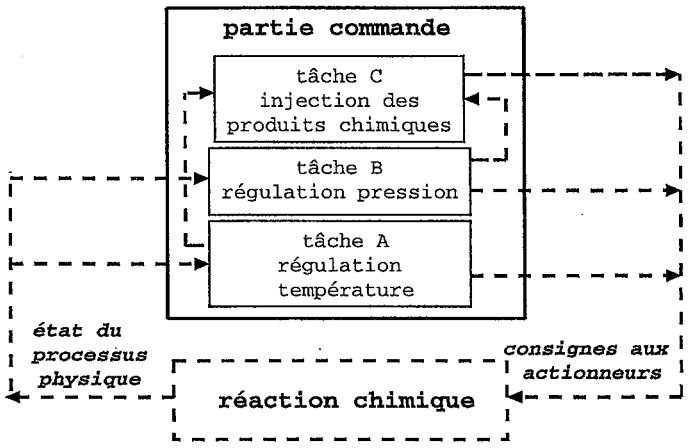


FIG. 4.10 – *Structure de l'application.*

Comme on peut le constater sur la figure 4.11, les tâches sont ordonnançables avec A et B sous RR (avec un quantum de 1 ms) et sous FPP. La solution RR a l'avantage que les données consommées par la tâche C soient plus cohérentes temporellement, les dates de production étant plus proches dans le temps qu'avec la stratégie FPP. Sous RR, la tâche C aura ainsi une vision plus exacte de l'état du système ce qui lui permettra d'injecter la bonne quantité de produits dans la réaction et permettra éventuellement d'accélérer le processus et de réduire les coûts. Il est évident que le besoin de cohérence temporelle n'est pas universel, mais en fonction de l'algorithme de contrôle et du processus physique sous-jacent, ce peut être un objectif à atteindre.

Pour quantifier l'amélioration apportée par l'utilisation conjointe de RR et de FPP par rapport à FPP seule, des expérimentations ont été menées sur 100 ensembles de tâches, générés

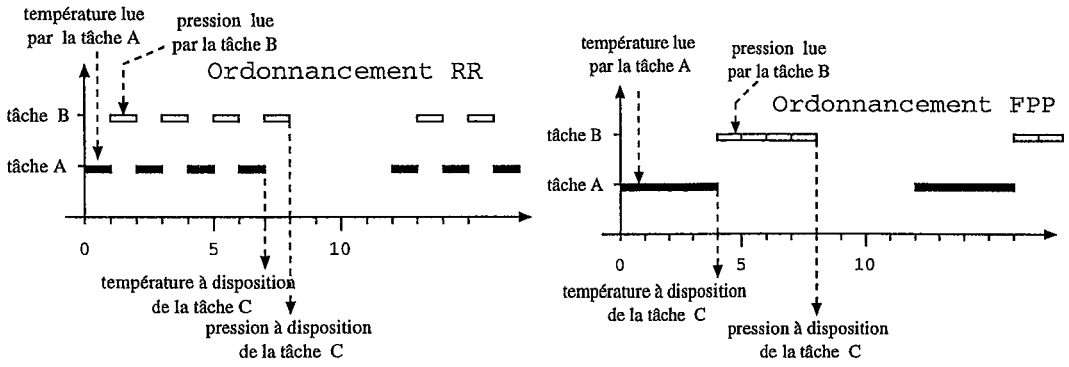


FIG. 4.11 – *Tâches A et B ordonnancées sous RR et FPP.*

aléatoirement, et composés de 10 à 20 tâches pour une charge totale variant de 30% à 70%. Pour chaque ensemble de tâches, l'AG, avec les paramètres précisés en section 4.3, a été exécuté à deux reprises : une fois avec la restriction de ne jamais utiliser RR et l'autre fois sans cette restriction. Nous nous sommes limités à des tâches de temps d'exécution constants de façon à éviter toute interférence de la distribution des temps d'exécution sur la valeur de la fonction objectif. Pour le critère de gigue sur les fins d'exécution (cf. équation 4.1), $card(\mathcal{T})/2$ tâches de $\mathcal{T}$ ont été choisies au hasard pour être prises en compte dans le critère. Pour les critères de fraîcheur et de cohérence temporelle des données (cf. respectivement équation 4.3 et 4.5), $card(\mathcal{T})/2$ tâches consommant des données ont été tirées au sort, chaque tâche consommant entre 1 et $card(\mathcal{T})/2$ données produites par des tâches également choisies au hasard. Quel que soit le critère, chaque tâche sélectionnée a le même poids dans le calcul de la fonction objectif (i.e. $\Phi_i = 1$).

Comme cela est représenté sur la figure 4.12, l'utilisation de RR a permis d'améliorer la valeur de la fonction objectif du meilleur individu de la population dans 36% des cas pour la gigue sur les fins d'exécution, dans 68% des cas pour la fraîcheur des données et 71% des cas pour la cohérence temporelle des données.

4.5 Conclusion et perspectives

Dans le domaine de l'informatique temps réel, les analyses d'ordonnançabilité, en particulier dans le cas FPP, ont fait l'objet d'un très grand nombre d'études. En général, à un même problème, il existe plusieurs solutions d'ordonnancement faisables qui ne sont pas équivalentes du point de vue de l'application, et le choix de la meilleure solution est un problème qui n'a été que très rarement abordé. Dans ce chapitre, nous proposons un AG dont l'objectif est de choisir au mieux les priorités et les politiques d'un ensemble de tâches relativement à un

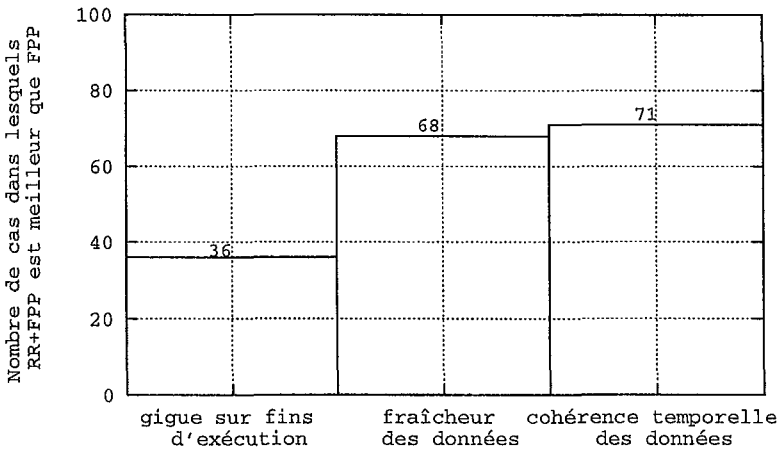


FIG. 4.12 – *Amélioration de la fonction objectif de la meilleure solution par l'utilisation combinée de RR et de FPP.*

critère tel que minimiser les gigues sur fins d'exécution ou maximiser la cohérence temporelle ou la fraîcheur de données utilisée en entrée d'une ou plusieurs tâches. L'AG, qui s'est montré efficace dans les expérimentations de la section 4.3 et dont nous avons proposé une version parallèle, utilise l'analyse d'ordonnançabilité pour distinguer les configurations faisables des configurations non-faisables et la simulation pour évaluer la qualité de chaque solution faisable.

Le choix de Posix1003.1b comme domaine d'application de cette étude est motivé par la très large adoption de ce standard, ou de certaines des fonctionnalités standardisées[27], par les systèmes d'exploitation temps réel disponibles sur le marché. Un de nos objectifs était de montrer que contrairement à l'idée reçue, la politique RR a son utilité dans les applications temps réel car il existe des situations dans lesquelles cette politique est efficace tant du point de vue de l'ordonnançabilité du système que du point de vue de son optimisation.

Sur des problèmes très contraints, même si il s'est révélé clairement plus efficace qu'un parcours aléatoire de l'espace des solutions, l'AG échoue assez fréquemment dans la phase de création de la population initiale. Pour améliorer sa capacité à trouver les premières solutions faisables, il est envisageable de travailler, dans une première étape, sur une population de solutions non-faisables. Pour cela, il semble d'abord nécessaire de définir une mesure de la faisabilité plus fine que oui ou non (par exemple à l'aide des "laxités", i.e. $D_k - R_k$) qui constituerait la fonction objectif dans cette première phase de l'algorithme. Ensuite, une fois trouvées les premières solutions faisables, l'AG reprendrait son déroulement habituel.

27. Windows NT4.0 par exemple, offre, avec certaines contraintes sur le nombre et le choix des priorités [Liu *et al.*, 1999], l'équivalent des politiques *sched_fifo* et *sched_rr* , ce qui rend l'analyse de ce chapitre et du chapitre précédent applicable aux systèmes temps réel basée sur l'exécutif Windows NT4.0.

Il serait naturel d'étendre ce travail au cas distribué, en particulier avec CAN comme protocole de communication, car des noeuds Posix1003.1b interconnectés par un réseau CAN constituent, à notre avis, une solution efficace pour des systèmes temps réel construits autour de composants du commerce.

Nous pensons qu'il est possible qu'en cas d'échec, l'AG fournisse des informations qui aide le concepteur à identifier les goulots d'étranglement du système. A moyen terme, on peut envisager un outil de conception (*CASE Tool* - Computer Aided Software Engineering) dédié au temps réel qui intègre l'analyse des pires temps d'exécution, par exemple à l'aide du travail entrepris sur le compilateur *gcc* dans [Macos and Mueller, 1998], ainsi que la fixation automatique des priorités et des politiques telle que proposée dans ce chapitre.

Troisième partie

Ordonnancement conjoint de trafic à contraintes strictes et souples avec garanties sur la Qualité de Service

Chapitre 5

Priorités dynamiques pour l'ordonnancement de messages

Résumé

Dans les systèmes temps réel, deux types de contraintes temporelles sont généralement identifiées : des contraintes strictes et des contraintes souples. La politique d'ordonnancement à changement de priorité (*"Dual-Priority policy"* - DP) donne des garanties sur le temps de réponse de tâches à contraintes strictes tout en diminuant le temps de réponse de tâches à contraintes souples. Dans [Tindell and Hansson, 1995], les auteurs ont proposé d'appliquer cette politique à l'ordonnancement de messages. Nous montrerons par simulation que les performances de DP, en comparaison à celles de la politique classique d'ordonnancement en arrière plan (*Background Scheduling* ou BS qui consiste à donner une priorité plus faible à tous les messages à contraintes souples) classiquement utilisée, sont excellentes sur une application typique de multiplexage véhicule utilisant CAN. Cette évaluation de performance quantitative sera suivie d'une évaluation plus qualitative où nous prouverons, par une méthode trajectorielle, que DP se comporte toujours mieux que BS sous une hypothèse FIFO sur le trafic à contraintes souples. Le résultat complémentaire est qu'il existe des exemples non-FIFO sur lesquels l'ordonnancement en arrière-plan est meilleur que la politique à changement de priorité.

Dans un environnement bruité, l'utilisation de DP a la fâcheuse conséquence de rendre le trafic à contraintes strictes particulièrement exposé à des dépassements d'échéances causés par les erreurs de transmission. Nous expliquerons pourquoi le trafic temps réel est hautement sensible aux erreurs de transmission, évaluerons le risque et proposerons un mécanisme simple pour donner des garanties probabilistes sur le respect des échéances des messages. Le mécanisme proposé sera comparé en termes de performance à la politique DP initiale. Enfin, nous proposerons une méthode de fixation "en-ligne" des paramètres du modèle d'erreurs,

bien adaptée à des environnements dans lesquels le niveau de perturbation est potentiellement sujet à de fortes variations ou difficilement prévisible, implantable sur des micro-contrôleurs de faible puissance. Dans ce chapitre, les métriques de performances retenus sont le pourcentage d'échéances dépassées pour le trafic temps réel, le temps de réponse moyen et la variance des temps de réponse pour le trafic à contraintes souples.

5.1 Introduction

Définition du problème. Deux types de contraintes sont généralement identifiées dans les systèmes temps réel, des contraintes strictes (*Hard Real-Time* - HRT) et des contraintes souples (*Soft Real-Time* - SRT). Nous faisons l'hypothèse que le trafic temps réel (TR), celui qui est soumis à des contraintes strictes, est périodique ou sporadique avec des échéances qui doivent être garanties alors que le trafic non-temps réel (non-TR), qu'il soit périodique ou apériodique, peut occasionnellement ne pas satisfaire ses contraintes (contraintes souples). Dans ce chapitre, nous traiterons du problème de l'ordonnancement de ces deux types de trafic, sur un réseau potentiellement non fiable, avec des objectifs différents : garantir des échéances pour le trafic TR et minimiser le temps de réponse moyen pour le trafic non-TR.

Étude de l'existant La stratégie la plus simple pour ordonnancer conjointement du trafic TR et non-TR est de donner systématiquement les plus fortes priorités au trafic TR. Comme cela était prévisible, cette politique dite d'ordonnancement en arrière-plan (*Background Scheduling* - BS) s'est révélée très peu efficace en termes de temps de réponse du trafic non-TR sur de nombreuses expérimentations (cf. [Bernat, 1998; Davis and Wellings, 1995a] pour l'ordonnancement de tâches et la section 5.3 pour l'ordonnancement de messages). Dans le cadre de l'ordonnancement de tâches, de nombreuses politiques plus efficaces ont été proposées, parmi lesquelles on peut citer Earliest Deadline Last (EDL, [Chetto and Chetto, 1989]), Deferrable Server (DS, [Lehoczky *et al.*, 1987]), Priority Exchange (PE, [Lehoczky *et al.*, 1987]), Extended Priority Exchange (EPE, [Sprunt *et al.*, 1989]), Static Slack Stealing algorithm (SSS, [Lehoczky and Ramos-Thuel, 1992]) et Dynamic Slack Stealing (DSS, [Davis *et al.*, 1993]). Comme cela a été souligné dans [Bernat, 1998] et [Davis, 1994], elles possèdent toutes des désavantages : DS, PE et EPE ne fournissent pas toujours d'excellentes performances, EDL et DSS sont algorithmiquement complexes et donc induisent d'importants overheads alors que SSS peut parfois nécessiter le stockage de quantités d'information très importantes. Le lecteur intéressé par un état de l'art et une description de ses solutions au problème de l'ordonnancement conjoint de tâches TR et non-TR pourra se référer à [Bernat, 1998].

Une alternative simple et élégante a été introduite pour l'ordonnancement préemptif de tâches dans [Davis, 1994; Davis and Wellings, 1995b] sous le nom de *"Dual-Priority policy"* (DP) que nous traduirons par politique à changement de priorité[28]. Cette politique vise à réduire le temps de réponse de tâches à contraintes de temps souples en repoussant "le plus tard possible" l'exécution des tâches critiques si des tâches à contraintes souples sont en attente d'exécution. D'un point de vue pratique, l'implantation de DP pour l'ordonnancement de tâches ne pose aucune difficulté majeure; elle peut être effectuée au niveau du noyau de l'OS comme cela a été réalisé pour l'OS temps réel *Monstre* [Riera, 1997] (cité dans [Bernat, 1998]) ou même au niveau applicatif, par exemple, en utilisant des primitives du langage ADA (cf. les travaux publiés dans [Burns and Wellings, 1996; Bernat, 1998]).

Dans la pratique, cette politique possède deux avantages clés sur les approches concurrentes : (1) de par sa simplicité elle n'induit pas d'overheads significatifs pendant l'exécution de l'application, (2) elle ne présuppose que très peu d'hypothèses restrictives sur le système ce qui la rend applicable sur une large variété de problèmes. Ainsi, la connaissance du premier instant d'activation d'une source de trafic périodique n'étant pas nécessaire, DP est utilisable pour l'ordonnancement de messages sur un réseau où les stations ne sont pas synchronisées. Tindell et Hansson dans [Tindell and Hansson, 1995] ont proposé d'appliquer cette politique à l'ordonnancement de messages sur le réseau CAN et ont suggéré des modifications au niveau du contrôleur de communication permettant une mise en oeuvre efficace mais sans évaluer les performances obtenues ni mettre en évidence le problème des erreurs de transmission.

Objectifs Notre objectif est tout d'abord d'évaluer quantitativement les performances de la politique DP pour l'ordonnancement de messages. Nous nous poserons ensuite la question de savoir si le gain en performances que nous avons observé avec l'emploi de DP, dans les conditions particulières de nos expérimentations, reste valable dans un contexte plus général et prouverons que DP se comporte toujours mieux que BS à la condition expresse que le trafic non-TR soit FIFO. Cette preuve sera faite par une méthode trajectorielle où une trajectoire est déterminée par les dates d'activation des messages et leur temps de transmission.

Ensuite, nous expliquerons pourquoi l'utilisation de DP est problématique dans des environnements bruités, quantifierons le problème en termes de pourcentage d'échéances non-respectées et proposerons un mécanisme fournissant des garanties probabilistes sur le respect des échéances. Nous montrerons également que le mécanisme proposé dégrade peu les performances de la politique DP tout en garantissant une qualité de service exprimée en termes de

28. *"Dual-priority policy"* aurait pu être traduit par politique à priorités doubles mais nous voulions insister sur l'existence d'un changement de priorité en cours d'exécution. D'autre part, le principe de cette politique peut être étendu à 3,4,..n niveaux de priorités.

probabilité de respect des échéances. Dans cette étude, un modèle d'erreurs très classique a été retenu : l'occurrence des erreurs suit une loi de Poisson. Ce travail pourra être adapté à des modèles d'erreurs plus expressifs comme celui proposé dans le chapitre 2.

Organisation du chapitre. Dans la section 5.2, nous préciserons le cadre de l'étude, à savoir les notations et la définition des deux politiques d'ordonnancement considérées. En section 5.2.2, nous évaluerons quantitativement les performances de DP en termes de temps de réponse, de variance des temps de réponse et de pourcentage d'échéances non-respectées. Ensuite en section 5.4, nous adopterons un point de vue plus qualitatif en comparant trajectoriellement DP et BS. La section 5.5 est consacrée au mécanisme permettant de donner des garanties probabilistes sur le respect des échéances. Enfin, en section 5.6, nous étudierons la possibilité de fixer "en-ligne", c'est-à-dire pendant l'exécution de l'application, les paramètres du modèle d'erreurs.

5.2 Cadre de l'étude

Le contexte de l'étude est celui de l'ordonnancement non-préemptif de messages sur un bus à priorité de la famille CSMA. Sur un tel bus, chaque message du système possède un identificateur unique qui servira à effectuer l'arbitrage pour l'accès au bus : si plusieurs stations décident simultanément d'émettre, le message de plus forte priorité gagnera le bus. Cet arbitrage basé sur la priorité nous permet de calculer, sous certaines hypothèses, une borne sur le temps de réponse de chacun des messages TR. Dans ce chapitre, les applications numériques et expérimentations sont effectuées en considérant CAN comme protocole support, néanmoins ce travail est également valable pour des protocoles comme VAN, Batibus, D2B, PALNET, ABUS, CCD ou encore le J1850 (cf. [Lawrenz, 1997] pour les caractéristiques de la plupart des bus à priorité).

Le système que nous étudions peut être modélisé par un ensemble de m messages récurrents et d'une ressource partagée, le médium de transmission, sur lequel les messages sont émis. Relativement aux contraintes temporelles, l'ensemble des messages de l'application $\mathcal{M}$ peut être subdivisé en deux sous-ensembles :

- les messages à contraintes strictes, $\mathcal{H} = \{m_1, \cdots, m_p\}$,
- et les messages à contraintes souples $\mathcal{S} = \{m_{p+1}, \cdots, m_m\}$.

On note $m_{k,n}$ la $n^{\text{ième}}$ instance du message m_k, $A_{k,n}$ sa date d'activation et $C_{k,n}$ sa durée de transmission (avec $\forall n \ C_{k,n} = C_k$). Le trafic TR est supposé périodique (resp. sporadique) de période (resp. de temps minimum interarrivée) T_k pour le message m_k. Le flux d'arrivée du trafic non-TR est non-contrôlé et aucune garantie n'est donnée sur les échéances associées à

ce type de trafic.

Le médium de transmission est partagé selon une politique $S \in \{BS, DP\}$. On note $B^S_{k,n}$ l'instant de début de transmission de $m_{k,n}$ et $E^S_{k,n}$ son instant de fin de transmission. $R^S_{k,n}$ est le temps de réponse de $m_{k,n}$ avec par définition $R^S_{k,n} = E^S_{k,n} - A^S_{k,n}$ [29]. Si une instance $m_{k,n}$ est activée avant l'instant t et si sa transmission n'est pas terminée à l'instant t, (i.e. $A_{k,n} \leq t < E^S_{k,n}$), alors $m_{k,n}$ est dite en attente (*pending*) à l'instant t. Chaque instance d'un message TR a une échéance D_k relative à son instant d'arrivée. Le système est dit *faisable* ou *ordonnançable* sous S si chaque instance TR respecte son échéance :

$$\forall k, \quad \forall n, \quad R^S_{k,n} \leq D_k. \tag{5.1}$$

Le médium de transmission, avec la politique S, est dit occupé (*busy*) à l'instant t si un message est en cours de transmission à cet instant. On définit une fonction $\beta^S(t)$, dite fonction d'utilisation de la ressource, continue à droite et valant 1 si le médium est occupé et 0 sinon.

Nous ferons l'hypothèse par la suite que chaque station a la connaissance des bornes sur les temps de réponse de tous les messages TR. Cette hypothèse est peu contraignante dans la pratique puisque d'une part les garanties temporelles imposent une définition statique de l'application (i.e. pas de nouveaux messages TR en cours de exécution), et que d'autre part, l'espace mémoire requis est non-significatif.

5.2.1 La politique BS

La politique d'ordonnancement en arrière plan (BS) est une politique à priorités fixes sous laquelle les priorités sont allouées de telle façon que tous les messages non-TR aient des priorités plus faibles que celles de tous les messages TR. Comme nous l'étudierons dans le paragraphe 5.3.1, ses performances à forte charge en ce qui concerne le trafic non-TR sont faibles. Néanmoins, elle possède un avantage clef qui est sa facilité d'implantation.

Priorités BS étant une politique à priorités fixes, toutes les instances d'un même message ont une priorité qui ne varie pas au cours du temps. La priorité de l'instance $m_{k,n}$ est donnée par la fonction de priorité $\pi^{BS}(k,n) = (k,n)$. Nous adopterons la convention "plus petite la valeur numérique, plus grande la priorité", si $\pi^{BS}(k,n) < \pi^{BS}(k',n')$, alors l'instance $m_{k,n}$ a une priorité plus forte que $m_{k',n'}$. La première composante de la priorité permet de donner un

29. Dans ce chapitre, nous nous focaliserons sur des mécanismes protocolaires de niveaux MAC en faisant largement abstraction des couches supérieures, c'est pourquoi le temps de réponse considéré n'intègre pas d'éventuelles gigues de niveau applicatif.

ordre sur des instances de messages différents, la seconde de distinguer les instances d'un même message (le système est FIFO pour les instances d'un même message qui serait en attente de la ressource au même instant).

Règle d'allocation de la ressource Dans notre étude, le protocole sous-jacent étant un bus à priorité, le principe est que dès que la ressource est libre, l'instance de plus forte priorité commence sa transmission.

5.2.2 La politique DP

La différence fondamentale entre BS et DP est que sous DP la priorité des tâches TR change dynamiquement au cours du temps. DP nécessite de partitionner la plage des priorités en 3 intervalles contigus [Davis, 1994] : "faible TR", "haut TR" et "non-TR". Toutes les priorités dans la plage "faible TR" sont plus faibles que celles de la plage "non-TR" qui sont elles-mêmes plus faibles que celles de la plage "haut TR". Les priorités de la plage "non-TR" sont réservées aux tâches non-TR alors qu'une tâche TR se voit initialement affecter une priorité dans la plage "faible TR", puis, lorsqu'elle devient urgente, sa priorité est promue dans la plage "haut TR". Ainsi la tâche TR urgente devient plus prioritaire que toutes les tâches non-TR. Le même principe s'applique à l'ordonnancement de messages [Tindell and Hansson, 1995] : au lieu de transmettre les messages TR dès que disponibles, ils peuvent être retardés au profit de messages non-TR en attente d'émission, et ce, jusqu'à ce qu'ils deviennent urgents. L'instant à partir duquel une instance TR $m_{k,n}$ devient urgente se calcule simplement si l'on connaît une borne sur son temps de réponse : la date la plus tardive de promotion de la priorité de $m_{k,n}$ de la plage "faible TR" à la plage "fort TR" garantissant le respect de l'échéance est $(A_{k,n} + D_k - R_k^{BS})$ ou R_k^{BS} est la borne sur le temps de réponse de toutes les instances du message m_k (cf. paragraphe 2.2.1 du chapitre 2) dans sa priorité forte, c'est à dire dans la plage "haut TR".
Plus formellement, nous définissons un *intervalle critique* pour le message TR m_k comme tout intervalle de temps de la forme $]A_{k,n} + D_k - R_k^{BS}, A_{k,n} + D_k]$, $\forall n \in \mathbb{N}$. Les priorités des messages TR changent au cours du temps, $\forall k \leq p$ (où $p = card(\mathcal{H})$),

$$\pi^{DP}(k,n,t) = \begin{cases} (k,n) & \text{si } t \text{ est dans un intervalle critique pour } m_k \\ & \text{(priorité "haut TR")}, \\ (m+k,n) & \text{sinon (priorité "faible TR")}. \end{cases} \tag{5.2}$$

Au contraire, la priorité des instances d'un message non-TR m_k reste fixe au cours du temps dans la plage "non-TR" : $\forall k > p$, $\forall t$, $\pi^{DP}(k,n,t) = (k,n)$.
Une fois les priorités définies, la règle d'allocation de la ressource est la même que sous BS.

Implémentation La politique DP n'est applicable à bon escient que si les hypothèses faites sur les périodes, les dates de changement de priorité et sur la taille des messages sont vérifiées. Dans le cadre de CAN, Tindell et Hansson ont proposé l'idée d'un contrôleur de communication "intelligent" ne permettant l'émission d'un message que si celui-ci remplit les hypothèses faites dans le calcul de R_k^{BS}. Ce contrôleur de communication aurait également la charge d'effectuer les changements de priorités à l'aide d'un timer dédié pour chaque niveau de priorité[30]. Même si le surcoût engendré par ces modifications est vraisemblablement peu significatif, à notre connaissance, un tel contrôleur n'est pas disponible commercialement.

Dans [Tindell and Hansson, 1995], les auteurs affirment que si une solution "hardware" se justifie dans des systèmes à contraintes fortes, une couche logicielle pourra remplir le même office dans des systèmes moins contraints, sans donner plus de détails sur l'implémentation de cette couche logicielle. Si en théorie cette solution est envisageable, dans la pratique elle n'est, à notre avis, pas toujours possible car elle présuppose que le contrôleur de communication offre un service d'annulation d'une requête de transfert de données. Sous cette hypothèse, le changement de priorité se fera d'abord en annulant la requête de transfert en basse priorité puis par l'émission d'une requête de transfert de données avec la priorité haute. L'overhead induit par l'exécution de ces deux primitives de service devra être borné et intégré dans le calcul de R_k^{BS}; le calcul de l'overhead devra tenir compte du fait que cette couche logicielle "middleware" entrera nécessairement en compétition pour le processeur avec les autres tâches de la station. Malheureusement, à notre connaissance, aucun protocole de niveau Laison de Données (LdD) n'offrent de service d'annulation de requête de transfert de données. Dans l'hypothèse même de l'existence de cette primitive de service, la solution d'un couche logicielle, en augmentant la borne temps de réponse, aurait le désavantage certain de réduire l'ordonnançabilité de la partie TR du trafic. Il serait alors plus astucieux de considérer une politique, qui pourrait s'appeler *Inverse Dual-Priority*, sous laquelle les changements dynamiques de priorités porteraient non pas sur les trames TR, mais sur les trames non-TR.

5.2.3 Un exemple de trajectoire sous DP et BS

Un exemple d'une trajectoire sous DP et BS est illustré sur la figure 5.1. Les rectangles représentent des instances de messages, ils sont grisés lorsque l'instance est en cours de transmission et non-grisés lorsque l'instance est en attente de la ressource. Sur cet exemple, quatre instances de messages différents sont en compétition, m_1 et m_2 sont des messages TR alors que

30. Niveau de priorité doit ici se comprendre comme première composante du vecteur de priorité. Pour qu'un seul timer suffise par message, il faut éviter d'avoir plusieurs instances d'un même message en attente de transmission au même instant et donc imposer $D_k \leq T_k$.

m_3 et m_4 sont des messages non-TR (avec les notations de la section 5.2, $p = 2$ et $m = 4$). Les dates d'activations sont $A_{1,1} = 0, A_{2,1} = 2, A_{3,1} = 4, A_{4,1} = 6$. Les temps de transmission sont respectivement $C_1 = 5, C_2 = 3, C_3 = 5, C_4 = 4$. A l'instant 0, la seule instance active est $m_{1,1}$, elle débute donc sa transmission immédiatement sous DP comme sous BS : $B_{1,1}^{BS} = B_{1,1}^{DP} = 0$. Lorsque $m_{1,1}$ est transmise (au temps 5), $m_{2,1}$ and $m_{3,1}$ sont en attente. Sous BS, leurs priorités respectives sont $\pi^{BS}(2,1) = (2,1)$ et $\pi^{BS}(3,1) = (3,1)$, $m_{2,1}$ gagne donc la ressource : $B_{2,1}^{DP} = 5$. Sous DP, leurs priorités sont $\pi^{DP}(2,1,5) = (2+4,1)$ (nous faisons l'hypothèse que l'instance $m_{2,1}$ n'est pas dans un intervalle critique au temps 5) et $\pi^{DP}(3,1,5) = (3,1)$. $m_{3,1}$ gagne alors la ressource et $B_{3,1}^{DP} = 5$. Sous DP, lorsque $m_{3,1}$ a terminé sa transmission au temps 10, $m_{2,1}$ and $m_{4,1}$ sont en attente. Leurs priorités sont respectivement $\pi^{DP}(2,1,10) = (2,1)$ (m_2 est dans un intervalle critique) et $\pi^{DP}(4,1,10) = (4,1)$: $m_{2,1}$ obtient donc la ressource avec $B_{2,1}^{DP} = 10$. Finalement, sous DP le temps de réponse de $m_{3,1}$ et $m_{4,1}$ (les instances non-TR) est respectivement de 6 et 11 contre 9 et 11 sous BS.

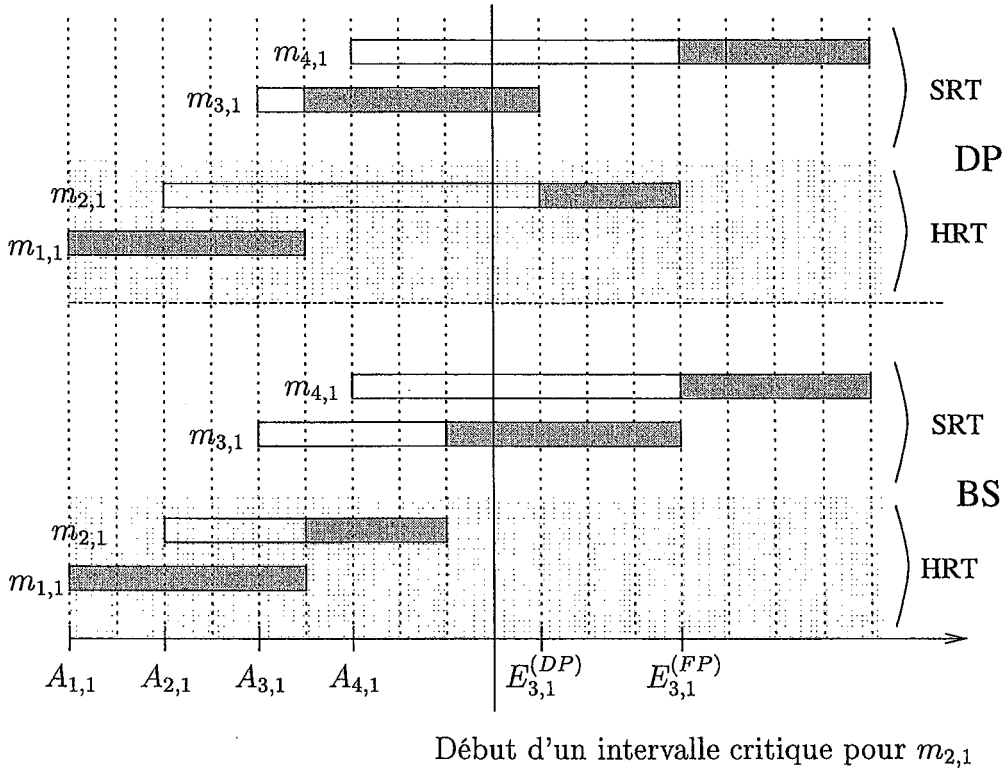


FIG. 5.1 – *Une trajectoire sous les politiques DP et BS.*

Remarque 1 *Un ensemble de messages sous DP n'est faisable que si le même ensemble de*

messages est faisable sous BS. Néanmoins, les temps de réponses des trames TR sont plus grands ou égaux sous DP que sous BS, le gain ne porte que sur la partie non-TR du trafic.

5.3 Performances de DP : évaluation quantitative

La trop grande complexité d'une analyse probabiliste exacte du système, nous a contraints à effectuer une étude par simulation réalisée à l'aide du simulateur à événements discrets *QNAP2*. Le temps de réponse moyen (cf. figure 5.2) ainsi que la variance des temps de réponse du trafic non-TR (cf. figure 5.3) ont été mesurés avec et sans l'utilisation de la politique DP pour une charge réseau globale variant de 60 % à 95 %[31]. Ensuite, nous considérerons la possibilité d'erreurs de transmission et évaluerons le pourcentage d'échéances non-respectées pour différentes conditions de perturbation du bus (cf. figure 5.4).

5.3.1 DP avec un médium de transmission fiable

Nous considérons par la suite une étude de cas sur un réseau CAN qui consiste en un ensemble de messages TR périodiques (décrit sur le tableau 5.1) plus un flux de messages non-TR dont les temps interarrivées sont exponentiellement distribués. Tous les messages périodiques ont une échéance (relative à leur instant d'arrivée) égale à leur période.

k	1	2	3	4	5	6	7	8	9	10	11	12
T_k (ms)	10	14	20	15	20	40	15	50	20	100	50	100
$D_k - R_k^{BS}$	8	11	16	10	14	33	7	41	10	88	37	86.2

TAB. 5.1 – *Messages périodiques du jeu d'essai.*

La partie TR de l'application provient de l'application embarquée décrite au paragraphe 1.1.1 du chapitre 1. Le débit du réseau CAN est fixé à 125kbit/s et nous ferons l'hypothèse que les messages TR ont une taille de 125 bits contre 100 bits pour les messages non-TR. Dans ces conditions, la partie TR du trafic génère une charge réseau de 53,46 %. Nous considérerons par la suite que les stations commencent toutes à émettre simultanément (à la mise en marche du véhicule).

5.3.1.1 Temps de réponse moyen du trafic non-TR

La figure 5.2 montre clairement que la politique DP réduit fortement le temps de réponse moyen du trafic non-TR dans nos conditions d'expérimentations. Le gain observé par rapport

31. Pour une charge réseau plus proche de 100 %, le système devient instable et les résultats sur le trafic non-TR perdent toute signification.

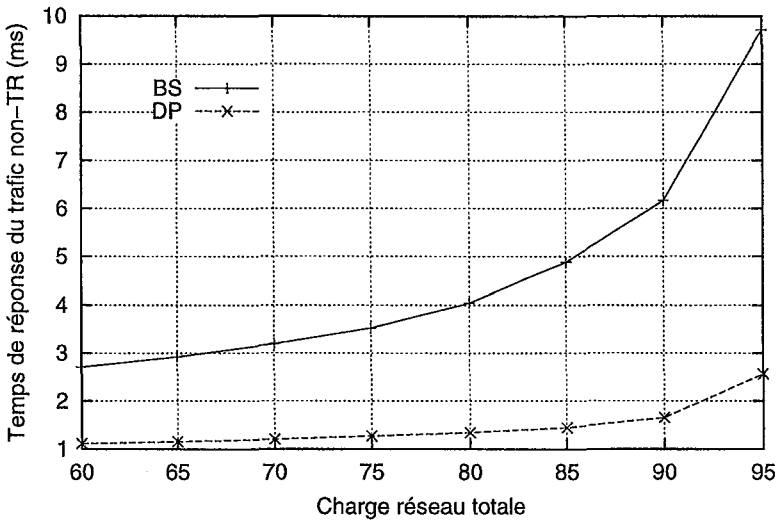


FIG. 5.2 – *Temps de réponse moyen du trafic non-TR sous DP et BS.*

à BS sur le temps de réponse moyen varie d'un facteur 2,4 pour une charge globale de 60 % jusqu'à 3,8 pour une charge de 95 %. Il est aussi remarquable que les performances de DP restent excellentes jusqu'à 90 % de charge.

5.3.1.2 Variance des temps de réponse pour le trafic non-TR

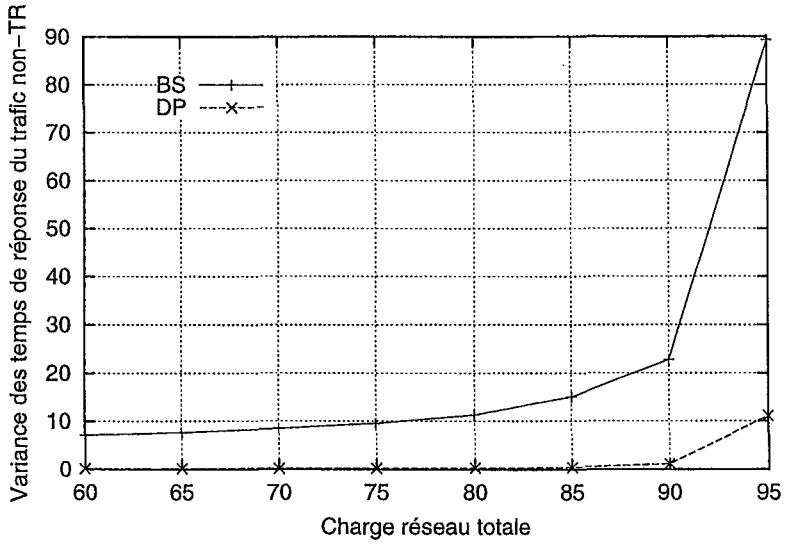


FIG. 5.3 – *Variance des temps de réponse pour le trafic non-TR sous DP et BS.*

La variance étant un estimateur de la dispersion d'un échantillon autour de la valeur moyenne, dans notre contexte, une variance faible induira un certain "déterminisme" dans

la transmission dans la mesure où généralement le temps de réponse d'un message non-TR sera peu éloigné de la moyenne. Sur la figure 5.3, on observe clairement que l'utilisation de DP diminue très fortement la variance des temps de réponse du trafic non-TR, ceux-ci étant presque constant pour une charge inférieure à 80 %.

5.3.2 Le problème des erreurs de transmission

Les réseaux locaux sont souvent amenés à fonctionner dans des environnements perturbés d'un point de vue électromagnétique. C'est particulièrement le cas pour CAN devenu un standard de fait dans les applications embarquées dans les véhicules, le problème étant rendu plus délicat par des contraintes économiques fortes ayant pour conséquences l'emploi d'un support de transmission relativement peu fiable [Barrenscheen and Otte, 1997] telle que la paire torsadée non blindée. L'objectif de ce paragraphe est de mettre en évidence que la politique DP est peu sûre lorsque des erreurs de transmission peuvent survenir.

Une instance d'un message TR m_k se voit à son activation $A_{k,n}$ allouer une priorité dans la plage "faible-TR", si un nombre suffisant de messages non-TR occupent le bus, l'instance $m_{k,n}$ devra attendre de passer dans la plage "haut-TR" pour gagner le bus. Ce changement de priorité a lieu à l'instant $(A_{k,n}+D_k-R_k^{BS})$. A l'instant $(A_{k,n}+D_k-R_k^{BS}+\epsilon)$ le dernier message non-TR a fini d'être transmis avec dans le pire des cas $\epsilon = \max_{i>k}(C_i)$ (cf. équation (2.2) du chapitre 2). Si une erreur de transmission se produit après l'instant $(A_{k,n} + D_k - R_k^{BS} + \max_{i>k}(C_i) - 23\tau_{bit})$ (où $23\tau_{bit}$ est la taille de la trame d'erreur), une ou plusieurs instances de messages TR, dont $m_{k,n}$, peuvent être amenées à ne pas respecter leur échéance.

Pour quantifier ce problème, des simulations ont été effectuées avec une charge totale (incluant la surcharge générée par les erreurs de transmission) de 100 % (dont toujours 53,46 % de trafic TR) pour des taux d'erreur trame (TeT) de 2,5 %, 5 % et 7,5 % en utilisant la politique DP. Pour un TeT de 5 %, une instance du message de plus forte priorité de notre application (priorité 1 sur la figure 5.4) a alors plus de 3 chances sur 100 de ne pas respecter son échéance. Notons que l'hypothèse d'une charge réseau de 100 % n'est pas forcément irréaliste sur une période de temps restreinte. Il est à noter que sans l'emploi de la politique DP, c'est à dire sous BS, aucun dépassement d'échéance TR n'a été observé pour les trois TeT considérés sur des simulations de même durée.

5.4 Performances de DP : évaluation qualitative

A notre connaissance, il n'a jamais été prouvé que DP améliore les temps de réponse du trafic non-TR. Dans cette section, nous donnons une condition suffisante sous laquelle DP

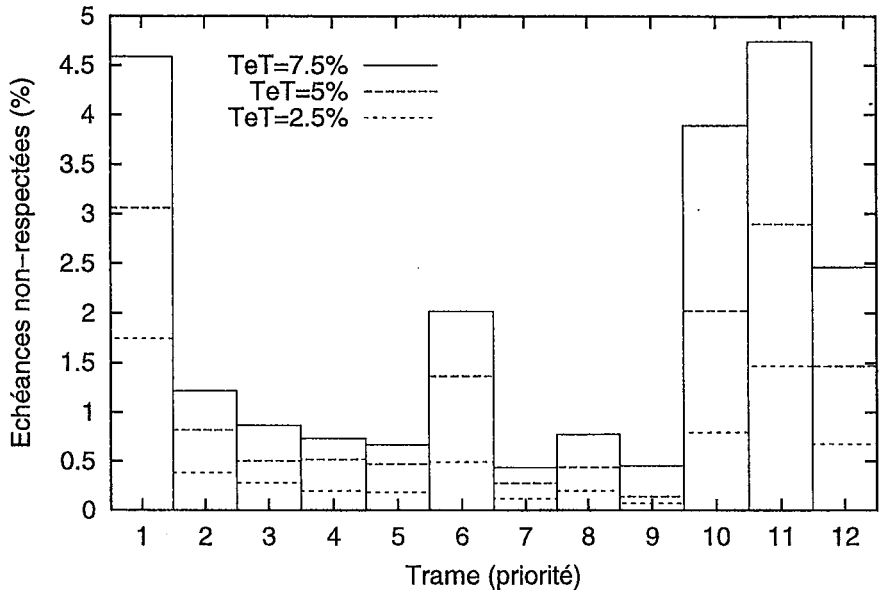


FIG. 5.4 – *Pourcentage d'échéances non-respectées pour le trafic TR sous la politique DP.*

est meilleur que BS sur toute trajectoire possible du système dans le cas non-préemptif. La condition est que le trafic non-TR soit FIFO, c'est à dire que l'ordre des dates de fin de transmission des instances de messages non-TR soit celui des dates d'arrivées. Le travail de cette section a été effectué en collaboration avec B. Gaujal et J. Migge et fait l'objet d'un rapport de recherche [Gaujal *et al.*, 1999a] et d'une soumission en cours d'évaluation [Gaujal *et al.*, 1999b] dans lesquels nous montrons également que dans le cas préemptif, la même propriété FIFO est suffisante pour que DP soit toujours meilleur que BS. D'autre part, nous proposons un mécanisme permettant de garantir la propriété FIFO sur le trafic non-TR pour l'ordonnancement de messages[32] sur un bus à priorité. L'idée est d'utiliser la date de mise à disposition de la trame comme valeur de sa priorité, ainsi, plus la trame sera "ancienne", plus sa priorité sera importante et la propriété FIFO sur le trafic non-TR sera assurée.

32. Dans un contexte centralisé comme celui de l'ordonnancement de tâches, l'ordre FIFO sur le trafic non-TR se réalise sans difficulté : à chaque arrivée d'une tâche non-TR, il suffit de lui donner une priorité plus faible que toutes les tâches non-TR en attente d'exécution.

5.4.1 Période d'interférence

La charge de travail totale en attente (*total workload*) à un instant t peut être définie [Migge and Jean-Marie, 1998] sous la forme de la fonction

$$W(t) = \sum_{(k,n)} C_{k,n} \mathbf{1}_{\{A_{k,n}<t\}} - \int_0^t \beta(t)dt. \tag{5.3}$$

avec $C_{k,n}$ la durée de transmission de $m_{k,n}$. La fonction $W(t)$ (continue à gauche) représente la quantité de travail qui est arrivée avant t et qui reste à accomplir. Le premier terme de l'équation (5.3) est le travail arrivé avant t, le second terme est le travail accompli entre 0 et t, 0 étant le premier instant de disponibilité de la ressource.

Lemme 1 *A tout instant, la charge totale de travail en attente sur une même trajectoire est identique sous BS et DP.*

Démonstration: Pour prouver ce lemme, nous allons utiliser le fait que les deux politiques sont non-oisives (*non-idling*) ce qui signifie que si il y a du travail en attente, la ressource est occupée. D'après la définition des fonctions d'utilisation de la ressource (cf. paragraphe 5.2), nous avons :

$$W^{BS}(t) = \sum_{(k,n)} C_{k,n} \mathbf{1}_{\{A_{k,n}<t\}} - \int_0^t \beta^{BS}(t)dt, \tag{5.4}$$

$$W^{DP}(t) = \sum_{(k,n)} C_{k,n} \mathbf{1}_{\{A_{k,n}<t\}} - \int_0^t \beta^{DP}(t)dt. \tag{5.5}$$

Si $\beta^{BS}(t) = \beta^{DP}(t)$ pour tout t alors on a forcément $W^{BS}(t) = W^{DP}(t)$. Faisons l'hypothèse que les fonctions d'utilisation de la ressource ne sont pas identiques sous les deux politiques. Par la continuité à droite de $\beta^{DP}(t)$ et $\beta^{BS}(t)$, alors il existe un temps $t_0 > 0$ t.q.

$$\begin{aligned} \beta^{DP}(t) &= \beta^{BS}(t), \quad \forall t \; s.t. \; 0 \leq t < t_0, \\ \beta^{DP}(t_0) &\neq \beta^{BS}(t_0). \end{aligned}$$

Si l'on a $\beta^{DP}(t_0) = 0$ et $\beta^{BS}(t_0) = 1$, comme les dates d'activation et les temps de transmission de toutes les instances de messages sont identiques pour les deux politiques (on raisonne sur la même trajectoire), et en utilisant le fait que les fonctions d'utilisation sont égales jusqu'à l'instant t_0[33], on a

$$W^{BS}(t_0) = W^{DP}(t_0).$$

33. La valeur des fonctions $\beta()$ en t_0 n'influe pas sur la valeur de l'intégrale, il faudrait une infinité non-dénombrable de points pour cela.

En utilisant le fait que DP est non-oisif, $\beta^{DP}(t_0) = 0$ signifie que $W^{DP}(t_0) = 0$. Pour BS, $\beta^{BS}(t_0) = 1$ et $W^{BS}(t_0) = 0$ est possible si il y a une arrivée exactement en t_0. Comme les instants des arrivées coïncident toujours sous DP et BS, cela implique qu'il y ait également une arrivée pour DP à l'instant t_0, ce qui contredit le fait que $\beta^{DP}(t_0) = 0$. Le même argument tient pour prouver que $\beta^{DP}(t_0) = 1$ et $\beta^{BS}(t_0) = 0$ est impossible. □

Définition 6 (période d'interférence, cluster) *Une période d'interférence est un intervalle de temps* $[t_1,t_2[$ *tel que* $\beta^S(t) = 1$ *pour tout* $t \in [t_1,t_2[$ *et* $W(t_1) = 0$, $W(t_2^+) = 0$. *L'ensemble des instances activées sur une même période d'interférence forme un "cluster", noté* $\mathcal{C}$.

Un corollaire immédiat du Lemme 1 est que les périodes d'interférences et les clusters coïncident sous BS et DP.

5.4.2 Un contre-exemple de DP>BS

Nous montrons maintenant un exemple sur lequel le temps de réponse d'une instance non-TR est plus petit sous BS que sous DP. Considérons la trajectoire représentée sur la figure 5.5 avec deux instances de tâches TR ($A_{1,1} = 0$ avec $C_1 = 5$, $A_{2,1} = 3$ avec $C_2 = 3$) et trois instances de tâches non-TR ($A_{3,1} = 6$ avec $C_3 = 5$, $A_{4,1} = 4$ avec $C_4 = 7$). Sans intervalles critiques pour les instances $m_{1,1}$ et $m_{2,1}$, nous obtenons $E_{1,1}^{BS} = 5$, $E_{2,1}^{BS} = 8$, $E_{3,1}^{BS} = 13$, $E_{4,1}^{BS} = 20$ pour BS et $E_{1,1}^{DP} = 5$, $E_{2,1}^{DP} = 20$, $E_{3,1}^{DP} = 17$, $E_{4,1}^{DP} = 12$ pour DP. Pour l' instance $m_{3,1}$, BS se comporte mieux que DP ($E_{3,1}^{BS} = 13 < E_{3,1}^{DP} = 17$) ce qui constitue un contre exemple de l'idée, que nous avions initalement, selon laquelle DP est toujours meilleur que BS.

5.4.3 Le cas FIFO

Dans ce paragraphe, toutes les instances de messages non-TR sont transmises dans un ordre FIFO. Sans perte de généralité, nous pouvons faire l'hypothèse qu'il existe un seul message non-TR m_m, avec l'indice $m = p+1$, qui rassemblent toutes les instances de messages non-TR. La première étape est de comparer DP et BS sur une seule période d'interférence de cluster $\mathcal{C}$. La comparaison globale sera dérivée de la comparaison de toutes les périodes d'interférences. Sur une période d'interférence, nous transformons DP en une politique à priorités fixes, appelée EP, qui se comporte exactement comme DP si l'on choisit judicieusement les priorités. On considère les instants de fin d'exécution de toutes les instances dans le cluster $\mathcal{C}$ sous DP et l'on alloue les priorités en fonction des dates de fin d'exécution, plus précisément

$$\pi^{EP}(k,n) = card\{m_{i,j} \in \mathcal{C} | E_{i,j}^{DP} < E_{k,n}^{DP}\}.$$

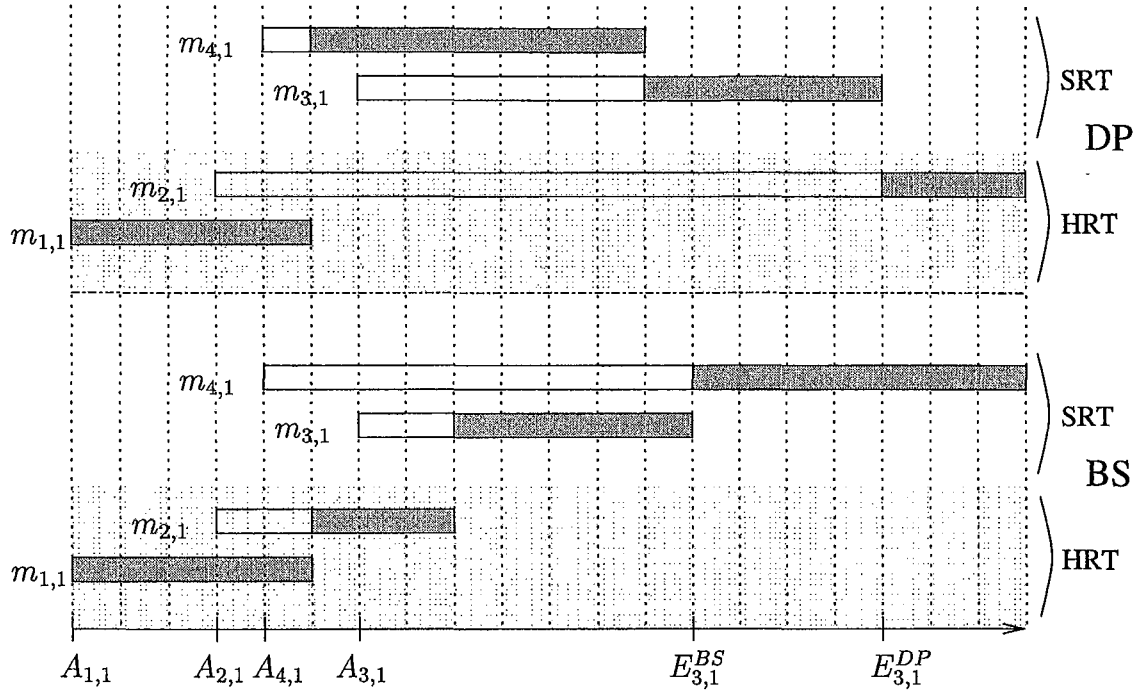


FIG. 5.5 – *Une trajectoire sur laquelle BS est meilleur que DP pour* $m_{3,1}$.

Lemme 2 *Pour toute instance* $m_{k,n}$ *dans* C, $E^{DP}_{k,n} = E^{EP}_{k,n}$.

Démonstration: Notons tout d'abord que comme EP est une politique non-oisive, elle a les mêmes clusters et les mêmes périodes d'interférences que DP. La preuve est faite par récurrence sur le nombre de messages du cluster. Si le cluster est composé d'un seul message, alors de façon évidente, $E^{DP}_{k,n} = A_{k,n} + C_{k,n} = E^{EP}_{k,n}$. Le cluster est maintenant composé de i instances. Nous enlevons le message transmis en dernier sous DP, disons l'instance $m_{a,b}$. Nous obtenons un cluster réduit de $i-1$ instances. Par hypothèse de récurrence, $E^{DP}_{k,n} = E^{EP}_{k,n}$ pour toutes instances de ce cluster réduit.
Sous DP, les instants de fin de transmission de toutes les instances dans le cluster original sont les mêmes que dans le cluster réduit car $m_{a,b}$ est transmis en dernier. Pour EP, l'instance $m_{a,b}$ a la plus petite priorité par construction, elle est donc transmise en dernier. □

Lemme 3 *Soit une instance arbitraire d'un message non-TR* $m_{a,b}$. *Alors* $E^{EP}_{a,b} \leq E^{BS}_{a,b}$.

Démonstration: Montrons tout d'abord que sous EP les priorités des instances non-TR sont ordonnées selon l'ordre d'arrivée. Soit $m_{m,n}$ une instance arbitraire non-TR, sous DP, les priorités non-TR sont statiques et FIFO (par hypothèse). Comme $A_{m,n} < A_{m,n+1}$, $\pi^{DP}(m,n,t) < \pi^{DP}(m,n+1,t)$ pour tout t. Si l'instance $m_{m,n}$ n'a pas été transmise au temps

$A_{m,n+1}$, alors les deux instances sont en attente. A la première opportunité de transmission, $m_{m,n}$ qui est plus prioritaire que $m_{m,n+1}$ par la règle FIFO, gagnera le bus plus tôt que $m_{m,n+1}$. On aura donc $E_{m,n} < E_{m,n+1}$ et par la définition de la politique EP $\pi^{EP}(m,n) < \pi^{EP}(m,n+1)$.

L'ordre parmi les priorités des instances non-TR est le même sous EP et BS, pour les instances TR les priorités sous EP sont différentes par rapport à BS. En appliquant le Lemme 4 donné dans l'annexe D, qui dit que le temps de réponse d'une instance du message de plus faible priorité, ne peut que s'améliorer si l'on alloue les priorités d'une façon différente à BS, nous obtenons $E^{EP}_{m,n} \leq E^{BS}_{m,n}$ pour tout n.

□

Théorème 1 *Pour tout message non-TR m_k, le temps de réponse des instances de m_k sont plus petits ou égaux sous DP que sous BS car pour tout n, $E^{DP}_{k,n} \leq E^{BS}_{k,n}$.*

Démonstration: La preuve est une conséquence directe des Lemmes 2 et 3. □

5.5 Garanties probabilistes sur le respect des échéances

Les performances de la politique DP se sont révélées très bonnes sur la partie non-TR du trafic en termes de temps de réponse moyen et de variance des temps de réponse. Néanmoins, le problème soulevé et quantifié sur un exemple dans le paragraphe 5.3.2 permet difficilement d'envisager l'utilisation de cette politique sur un médium non fiable. Notre objectif est ici de proposer un mécanisme qui, quitte à dégrader légèrement la politique DP initiale, donne des garanties sur la probabilité de respecter les échéances. Toutes les applications n'ayant pas des exigences identiques en matière de sûreté de fonctionnement (SdF), nous proposons de spécifier le niveau de qualité de service (QoS) requis par l'intermédiaire d'un paramètre, noté α et défini comme une borne sur la probabilité de respect des échéances du trafic TR, dont la valeur sera fonction des contraintes de SdF de l'application considérée. Par exemple, avec α choisi égal à 0,001, aucune trame TR de l'application ne manquera son échéance en moyenne plus de 1 fois sur 1000.

5.5.1 Principes

Pour offrir des garanties tout en gardant le principe de la politique DP, la stratégie est simplement d'avancer l'instant de changement de priorité si la valeur du paramètre α l'exige. Pour cela, il s'agit de trouver, pour chaque trame TR m_k de l'application, le nombre minimal d'erreurs de transmission η_k tel que la qualité de service requise sera atteinte et d'en déduire

l'instant de changement de priorité correspondant. Si l'on note $R_k^{BS}(n)$ la borne sur le temps de réponse de toute les instances de m_k avec n erreurs de transmission et $X(t)$ le processus stochastique comptant le nombre d'erreurs de transmission sur le temps, il faut déterminer η_k tel que la probabilité d'observer plus de η_k erreurs de transmission pendant $R_k^{BS}(\eta_k)$ soit inférieure ou égale à α:

$$\eta_k = \min\{n \in \mathbb{N} \mid P[X\left(R_k^{BS}(n)\right) > n] \leq \alpha\}. \tag{5.6}$$

où $P[X\left(R_k^{BS}(n)\right) > n] \leq \alpha$ peut être ré-écrit sous la forme:

$$\left(1 - \sum_{i=0}^{n} P[X(R_k^{BS}(n) = i]\right) \leq \alpha \tag{5.7}$$

Si moins de $\eta_k + 1$ erreurs se produisent pendant $R_k^{BS}(\eta_k)$ (avec $R_k^{BS}(\eta_k) \leq D_k$), la trame m_k respectera forcément son échéance. Pour le calcul de $R_k^{BS}(n)$, outre le temps d'interférence et le temps de transmission, il faut considérer une fonction dite de reprise sur erreurs, notée $\mathcal{E}_k(n)$, qui prend en compte l'overhead induit par les erreurs, c'est-à-dire pour chaque erreur, la trame d'erreur de 23 bits puis la retransmission de la trame corrompue. Notons que cette fonction a été initialement introduite dans [Tindell and Burns, 1994a] (sous le terme de *error recovery overhead function*) comme une fonction du temps, elle devient pour nous une fonction du nombre d'erreurs.

Le calcul $R_k^{BS}(\eta_k)$ est effectué avec les équations (2.8), (2.9) et (2.10) du chapitre 2, en partant de $\eta_k = 0$ et en incrémentant η_k tant que l'inégalité 5.7 n'est pas satisfaite et tant que $R_k^{BS} \leq D_k$. Si à l'issue du calcul, on se trouve dans la situation $R_k^{BS}(\eta_k) > D_k$, la QoS requise ne peut être garantie, dans le cas contraire, le changement de priorité aura lieu $\left(D_k - R_k^{BS}(\eta_k)\right)$ après la mise à disposition de la trame m_k.

Dans ce chapitre, nous avons retenu un modèle d'erreurs très classique qui fait l'hypothèse que l'occurrence des erreurs suit une loi de Poisson. On peut naturellement utiliser des modèles d'erreurs plus expressifs comme celui proposé dans le chapitre 2. Néanmoins la fixation en-ligne des paramètres devient alors plus problématique et l'emploi de techniques statistiques de fixation des paramètres plus évoluées que celle de la section 5.6 devient nécessaire avec toujours la contrainte de limiter les overheads en temps de calcul.
Dans le cas où le modèle d'erreurs suit une loi de Poisson de paramètre λ, on a:

$$P[X(t) = k] = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \tag{5.8}$$

La valeur de λ sera fixée à l'aide de mesures collectées sur un prototype ou avec l'expérience accumulée sur des systèmes similaires. Des données collectées sous forme d'un TeT, on en déduit la valeur de λ :

$$TeT = \frac{\lambda}{\widetilde{n}} \tag{5.9}$$

avec λ le nombre moyen de trames corrompues par seconde et $\widetilde{n} = (R \cdot \rho)/\widetilde{S}$ le nombre moyen de trames par seconde où R est le débit de transmission (bit/s), ρ la charge moyenne du réseau (somme de la charge non-TR ρ_a et de la charge TR ρ_p) et $\widetilde{S}$ la taille moyenne des trames :

$$\widetilde{S} = \left(\widetilde{S_a} \cdot \frac{\rho_a}{\rho}\right) + \left(\widetilde{S_p} \cdot \frac{\rho_p}{\rho}\right).$$

$\widetilde{S_a}$ la taille moyenne du trafic non-TR est dépendant de l'application et $\widetilde{S_p}$ la taille moyenne du trafic TR est égale à :

$$\widetilde{S_p} = \sum_{m \in \mathcal{H}} \frac{S_m}{T_m} / \sum_{m \in \mathcal{H}} \frac{1}{T_m}$$

avec $\mathcal{H}$ est l'ensemble de messages TR et S_k la taille du message TR m_k (cf. équation 2.3 pour le réseau CAN).

Dans notre jeu d'essai, nous avons considéré des trames TR de tailles identiques donc uniformément affectées par les erreurs de transmission. Dans l'hypothèse contraire, et surtout si les tailles sont très dissemblables, il faut dériver du λ global, le λ_k correspondant à la trame m_k :

$$\lambda_k = \frac{\lambda}{\widetilde{S}} \cdot S_k \tag{5.10}$$

Sur l'application décrite dans le paragraphe 5.3.1 avec un TeT de 5 % pour le trafic TR ($\lambda = 53{,}13$) et α fixé à 0,001, on obtient les résultats présentés dans le tableau 5.2 pour η_k, le pire temps de réponse correspondant $R_k^{BS}(\eta_k)$ et la date de changement de priorité (relativement à l'instant d'arrivée des trames) égale à $D_k - R_k(\eta_k)$.

La QoS requise est spécifiée par l'intermédiaire du paramètre α commun à toutes les trames, il est parfaitement possible d'ajuster ce paramètre pour chacune des trames en fonction de leur criticité, par exemple, en pondérant α de la priorité de la trame.

Trame (id)	η_k	$R_k(\eta_k)$ (ms)	$D_k - R_k(\eta_k)$ (ms)
1	3	5,55	4,45
2	3	6,55	7,45
3	3	7,55	12,45
4	4	9,74	5,26
5	4	10,74	9,26
6	4	12,74	27,26
7	5	14,92	0,08
8	5	18,92	31,08
9	5	19,92	0,08
10	6	26,10	73,90
11	6	27,10	22,90
12	7	30,29	69,71

TAB. 5.2 – *Valeurs de* η_k, $R_k^{BS}(\eta_k)$ *et* $D_k - R_k(\eta_k)$ *pour* $\alpha = 0{,}001$ *et* $TeT = 5\%$.

5.5.2 Évaluation des performances

La proposition décrite dans le paragraphe précédent donne des garanties sur la probabilité de respecter les échéances. En avançant l'instant de changement de priorité, elle a néanmoins la contrepartie négative de réduire la fenêtre temporelle durant laquelle le trafic non-TR est prioritaire et donc de dégrader les performances pour ce type de trafic. Dans ce paragraphe, nous nous attacherons à quantifier la perte de performances en termes de temps de réponse moyen et de variance des temps de réponse.

5.5.2.1 Temps de réponse moyen du trafic non-TR pour différentes valeurs de α

La figure 5.6 présente le temps de réponse moyen du trafic non-TR collectés par simulation pour différentes valeurs de α avec une charge réseau de 85 % et 90 % et un TeT de 5 %. Sur cette figure, on voit clairement que le mécanisme proposé ne dégrade que faiblement les temps de réponse. Par exemple, pour $\alpha = 0{,}001$ et 90 % de charge réseau, le temps de réponse moyen est toujours 2,4 fois plus petit qu'avec BS et seulement 1,4 fois plus grand qu'avec la politique DP initiale tout en garantissant une QoS relativement élevée.

5.5.2.2 Variance des temps de réponse du trafic non-TR pour différentes valeurs de α

De la même façon que pour les temps de réponse, nous observons que la perte de performances est limitée pour la variance des temps de réponse du trafic non-TR. Ainsi, pour 90 % de charge réseau et toujours un TeT de 5 %, la variance est de 5,59 avec la politique DP initiale, 12,61 avec la politique modifiée pour $\alpha = 0{,}001$ et 46,38 avec BS. Sur la figure 5.7, on peut aussi remarquer que la variance des temps de réponse diminue rapidement avec la charge

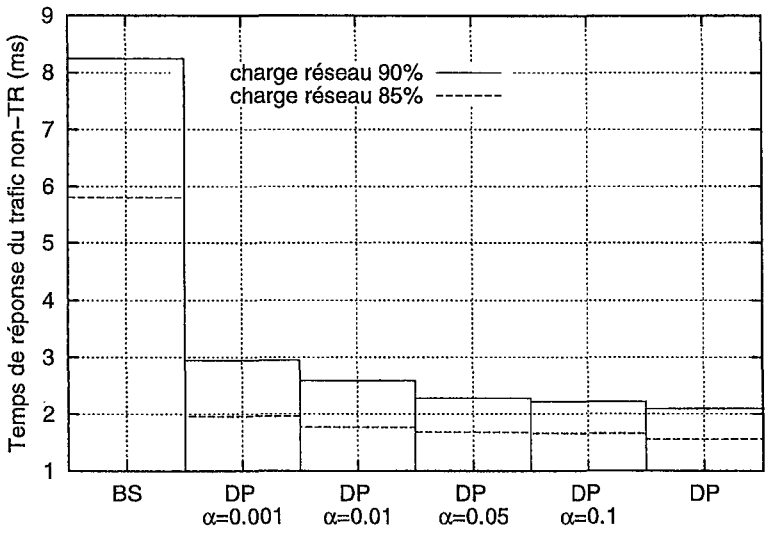


FIG. 5.6 – *Temps de réponse moyen du trafic non-TR pour différentes valeurs de α avec $TeT = 5$ %*

réseau ce qui signifie logiquement que moins la charge est élevée, moins les performances du trafic le moins prioritaire sont variables.

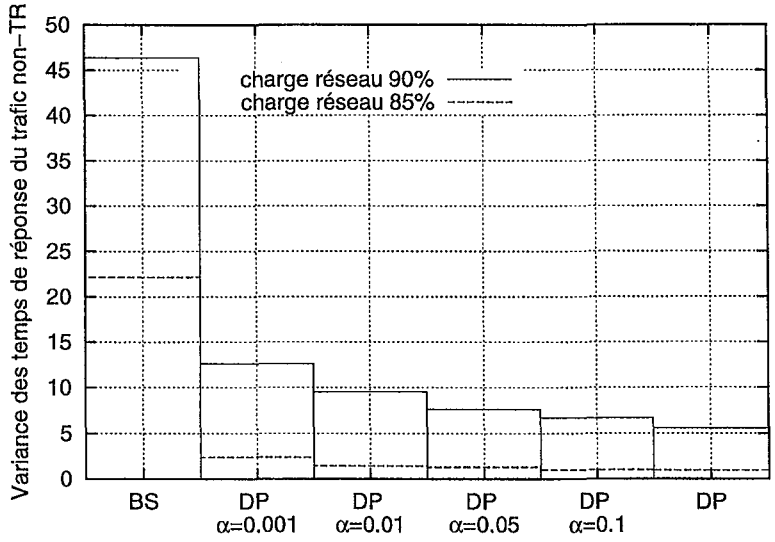


FIG. 5.7 – *Variance des temps de réponse pour le trafic non-TR pour différentes valeurs de α avec $TeT = 5$ %*

5.6 Vers une politique DP adaptive

Nous avons considéré jusqu'ici que le TeT a une valeur fixée sur la durée de vie du système. Il existe cependant une grande variété de systèmes dans lesquels le TeT peut varier grandement au cours du temps. C'est typiquement le cas des systèmes embarqués dans les véhicules sur lesquels nous nous focaliserons dans cette section. Des études antérieures [Noble, 1992] ont par exemple montré qu'à proximité des aéroports, dans des zones balayées par des radars, la fréquence des erreurs de transmission augmentait très fortement. Dans ce paragraphe, nous proposerons un mécanisme simple, implantable sur des micro-contrôleurs de puissance modérée, pour la mise en oeuvre d'une politique DP dont les paramètres s'adaptent au niveau de perturbations courant sur le bus. Notons que le protocole CAN assure qu'une erreur de transmission sera détectée par l'ensemble des stations du réseau, chaque station pourra donc se faire une image identique du niveau de perturbation.

5.6.1 Hypothèses sur les contrôleurs de communication

La perspective d'une politique qui s'adapte en-ligne au niveau de perturbations courant est rendue possible par l'apparition d'une nouvelle génération de contrôleurs de communication CAN qui possèdent des fonctionnalités intéressantes pour le signalement des erreurs telle que la possibilité de scruter les compteurs d'erreurs [Hausmann and Gebing, 1997] ou de générer une interruption sur l'occurrence d'une erreur [Philips Semiconductors, 1997]. Si un "polling" répétitif des compteurs d'erreurs est éventuellement envisageable dans la phase de mise au point du système pour la fixation du paramètre λ (sur une station dédiée à cet effet), cette solution ne nous paraît pas envisageable en phase d'exploitation. En effet, les compteurs d'erreurs évoluent selon des règles d'incrémentation et de décrementation relativement complexes qui obligent à une scrutation fréquente[34] générant une charge CPU inconciliable avec un fonctionnement normal.

Nous considérerons donc dans la suite que les contrôleurs sont capables de générer une interruption vers le CPU sur l'occurrence d'une erreur, la routine de service d'interruption (ISR) se contentant d'incrémenter un compteur. Le calcul du TeT courant se fera périodiquement au niveau du CPU. La période choisie, notée $\mathcal{T}$, devra être compatible avec les exigences de l'application en termes de charge CPU induite (en tenant compte des overheads causés par les

34. Le processus d'application doit être capable de lire les compteurs d'erreurs après la fin de transmission de chaque trame circulant sur le bus pour se faire une idée exacte des perturbations. Le problème est qu'au niveau du processus d'application, l'instant de fin de transmission d'une trame non destinée à la station n'est pas connu, obligeant donc à une scrutation avec une période plus petite que le temps de transmission de la plus petite trame du système soit, à 125kbit/s, environ 0,4 ms.

interruptions) mais devra être suffisamment petite pour refléter les fluctuations du niveau de perturbation. Comme une perturbation n'est détectable par le contrôleur de communication que lorsqu'elle se produit pendant la transmission d'une trame, le processus d'application, pour estimer le niveau de perturbation, devra également être capable de quantifier le nombre de trames transmises sur le dernier intervalle de temps $\mathcal{T}$. La seule possibilité avec les contrôleurs de communication actuels est d'accepter tous les messages, d'incrémenter un compteur de trames dans l'ISR, et d'effectuer le filtrage au niveau applicatif. Il serait cependant beaucoup plus efficace de disposer d'un compteur de trames au niveau du contrôleur de communication. Cette modification très peu coûteuse, en dehors même de son intérêt dans le cadre de la politique DP, permettrait à toutes les stations d'un réseau CAN d'estimer la charge réseau courante pour éventuellement adapter leur politique d'ordonnancement de messages (priorités, périodes). Après lecture par le processus d'application, les compteur d'erreurs et de trames, qu'ils se trouvent au niveau applicatif ou dans le contrôleur de communication, seront remis à zéro pour éviter des problèmes de débordement de capacité.

On note $E(a,b)$ la fonction qui donne le nombre d'erreurs ayant eu lieu dans l'intervalle $[a,b[$ et $F(a,b)$, la fonction qui compte le nombre de trames dont l'émission a pris fin sur l'intervalle $[a,b[$. Le TeT observé par une station pendant l'intervalle de temps $[a,b[$, noté $TeTobs_{[a,b[}$, est par définition $TeTobs_{[a,b[} = E\,(a,b)\ /\ F\,(a,b)$.

5.6.2 Modèle de variabilité des erreurs

Des contacts industriels (contrats industriels [Navet and Song, 1996; Song and Navet, 1997]) et quelques articles abordant le sujet [Noble, 1992] [Zanoni and Pavan, 1993] [Barrenscheen and Otte, 1997], nous ont permis d'avoir une idée relativement précise sur la réalité du phénomène des erreurs de transmission sur les réseaux embarqués dans les véhicules. Nous considérerons toujours que le flux d'arrivée des erreurs est poissonnien mais que le taux d'arrivée des erreurs peut varier au cours du temps. Nous ferons l'hypothèse qu'il existe des zones dans lesquelles les perturbations sont de niveaux similaires et dont la durée de traversée est de l'ordre de quelques secondes dans le cas d'un véhicule. Le temps de traversée d'une zone sera donné par la variable aléatoire S qui obéit à la loi uniforme de paramètre S_{min}, S_{max} (avec $S_{min} \leq S_{max}$) :

$$P(S = k) = \frac{1}{S_{max} - S_{min} + 1} \text{ pour } k \in [S_{min}, S_{max}]$$

De façon similaire, le TeT pour chaque zone sera donné par une variable aléatoire suivant une loi uniforme de paramètre TeT_{min} et TeT_{max}. Ce modèle nous servira d'hypothèse de

travail pour valider nos choix en matière de prévision. En particulier, il faut se poser la question du modèle de prévision et de la "taille" de sa mémoire.

5.6.3 Modèle de prévision

Après plusieurs essais infructueux se heurtant à la forte variabilité du flux d'arrivée poissonnien, nous avons considéré l'utilisation d'une méthode de prévision robuste et simple connue sous le nom de "lissage exponentiel" ([Wonnacott and Wonnacott, 1995] page 805), qui sur nos essais a donné des résultats satisfaisants (cf. figure 5.8). Le lissage exponentiel pondère une observation par son ancienneté avec des pondérations qui décroissent de façon exponentielle :

$$TeT_t = (1-\gamma) \cdot TeTobs_{[(t-1)\cdot\mathcal{T}, t\cdot\mathcal{T}[} + \gamma \cdot TeT_{t-1} \tag{5.11}$$

où $t \cdot \mathcal{T}$ est l'instant du $t^{\text{ième}}$ calcul. La valeur de TeT_0 sera fixée à une valeur moyenne ou simplement plausible et γ, la pondération, a été estimée par simulation de façon à minimiser la somme des carrés des erreurs de prévision : $min\left(\sum_{t\in\mathbb{N}}(TeTobs_{[(t-1)\cdot\mathcal{T}, t\cdot\mathcal{T}[} - TeT_t)^2\right)$. Ainsi, pour $\mathcal{T} = 100$ ms, $TeT_{min} = 0{,}02$, $TeT_{max} = 0{,}08$, $S_{min} = 2000$ ms, $S_{max} = 6000$ ms, la valeur optimum de γ se situe à 0,82 alors que pour $\mathcal{T} = 200$ ms la valeur optimale est 0,6. La figure 5.8 représente l'approximation d'un flux d'erreurs poissonnien obtenue avec $\gamma = 0{,}6$, pour $T = 200$ ms.

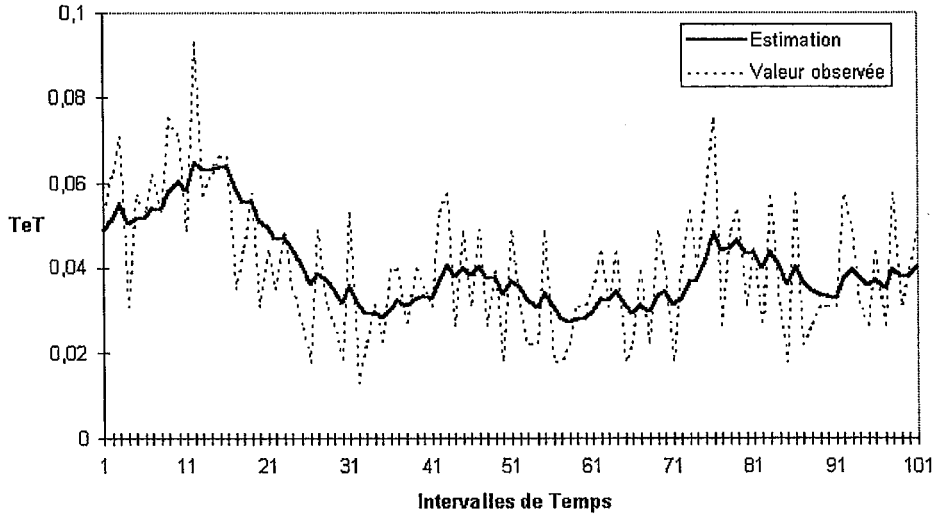


FIG. 5.8 – *Lissage exponentiel sur un flux d'arrivée poissonnien avec* $\gamma = 0{,}6$, $\mathcal{T} = 200$ *ms,* $TeT_{min} = 0{,}02$, $TeT_{max} = 0{,}08$, $S_{min} = 2000$ *ms,* $S_{max} = 6000$ *ms.*

L'utilisation du lissage exponentiel a comme avantage un temps de calcul très faible et de plus constant pour chaque estimation, et d'autre part, en réduisant le passé à une seule valeur,

il est très économe en mémoire. Un autre intérêt de cette technique est qu'elle permet des déconnexions temporaires des stations (état "bus-off" du protocole CAN, cf. paragraphe 1.2.5) et qu'elle ne nécessite ni horloge globale et ni synchronisation des stations lors de la mise en marche du système.

5.6.4 Application à la politique DP

À chaque instant les stations connaissent le TeT courant estimé avec l'équation [5.11], il faut maintenant utiliser cette information pour déterminer l'instant de changement de priorité de chaque instance de toute trame m_k, c'est-à-dire déterminer η_k telle que l'inéquation (5.6) soit vérifiée puis en déduire $A_{k,n} + D_k - R_k^{BS}(\eta_k)$. On ne peut se contenter de réaliser le calcul de l'instant de changement de priorité à l'instant de mise à disposition de la trame étant donné que le niveau de perturbation peut changer par la suite. Il faudra donc effectuer le calcul plusieurs fois pour ajuster au fur et à mesure la date de changement de priorité. Nous proposons par exemple d'effectuer ce calcul à la fin de la procédure de fixation du TeT. Cette solution est toutefois imparfaite car après de brusques variations du niveau de perturbation, qui correspondent dans notre modèle à des changements de zone, il est possible de se retrouver dans une situation où, après le calcul, la trame aurait déjà dû être promue dans la plage "haut-TR". Néanmoins, les changements de zone étant par hypothèse peu fréquents, cette solution reste valable. D'un point de vue implémentation, de façon à ne pas surcharger le CPU, nous proposons de pré-calculer pour chaque trame m_k, les valeurs de $R_k^{BS}(\eta_k)$ sur une large plage de valeurs du TeT et de les stocker, puis lors de l'utilisation, de se référer à la valeur pré-calculée de $R_k^{BS}(\eta_k)$ correspondant au TeT immédiatement supérieur à la valeur du TeT courant.

Pour évaluer les performances de cette proposition, nous avons envisagé deux situations, 1) le TeT moyen est connu mais le flux d'erreurs et soumis à des variations et 2) le TeT n'est pas prévisible. Dans le premier cas, les résultats de simulations effectuées pour différentes valeurs de α, γ, T et différents paramètres du modèle d'erreur (cf. paragraphe 5.6.2), nous amènent à conclure que la fixation en-ligne des paramètres n'améliorent pas les performances du trafic non-TR. On assiste même à une dégradation peu significative des temps de réponse du trafic de l'ordre de 1 à 2 % avec des instants de changements de priorité pré-calculés ce qui s'explique logiquement par le choix d'une valeur de $R_k^{BS}(\eta_k)$ pessimiste puisque correspondant au TeT immédiatement supérieur au TeT courant. L'intérêt de la fixation en-ligne des paramètres du modèle d'erreurs se situe ici au niveau du respect des échéances, et bien que sur nos simulations le pourcentage d'erreurs de transmission sans la fixation "en-ligne" n'ait jamais dépassé le seuil α, on assiste à une diminution sensible du nombre d'échéances dépassées et ce d'autant plus que les erreurs sont sujetes à de fortes variations. Dans la situation où le TeT n'est a priori

pas connu, cette proposition s'impose naturellement puisqu'en fixant arbitrairement un TeT, on risque un fort pourcentage d'échéances non respectées pour le trafic TR (TeT sous-estimé) ou des performances médiocres pour le trafic non-TR (TeT surestimé).

5.7 Conclusion et perspectives

Dans cette étude, nous avons évalué les performances de la politique DP sur un jeu d'essai réaliste du domaine de l'automobile embarqué dans un premier temps sous l'hypothèse d'un médium fiable. Les performances de la politique DP se sont révélées excellentes pour le trafic à contraintes de temps souples, en termes de temps de réponse moyen et de variance des temps de réponse. Nous avons mis en évidence que cette politique utilisée sur un médium non fiable expose fortement le trafic TR à des dépassements d'échéances. Le problème identifié, nous avons proposé un mécanisme simple permettant de garantir une QoS exprimée comme une probabilité de respect des échéances. Cette proposition dégrade faiblement les performances de la politique DP et donne des garanties probabilistes sur le respect des échéances. La QoS requise devra être choisie en fonction des contraintes de l'application considérée et pourra être éventuellement individualisée en fonction de la criticité de chacune des trames. Enfin, nous avons proposé des mécanismes permettant de fixer "en-ligne" les paramètres du modèle d'erreurs dont l'utilisation s'impose lorsqu'il n'est pas possible de déterminer *a priori* le niveau de perturbation sur le bus ou lorsque celui-ci et soumis à de fortes variations. Cette étude a fait l'objet d'un article [Navet and Song, 1999b] (évaluation de performances et garanties probabilistes sur la QoS), d'une publication dans un congrès [Navet and Song, 1999c] (garanties probabilistes sur la QoS et fixation en ligne des paramètres du modèle d'erreurs).
Nous avons également prouvé que DP se comporte toujours mieux que BS si les messages non-TR sont FIFO. Les simulations, présentées en annexe D, confirment ce résultat. Dans un rapport de recherche [Gaujal *et al.*, 1999a] (soumis pour publication [Gaujal *et al.*, 1999b]), nous proposons un mécanisme protocolaire garantissant la propriété FIFO sur un bus à priorité et montrons que dans le cadre de l'ordonnancement préemptif de tâches, sous la même condition FIFO, DP améliore les temps de réponse du trafic non-TR. Nous ne savons pas actuellement dans quelle mesure d'autres politiques que BS et DP peuvent être comparées trajectoriellement. Si c'était possible, nous pourrions envisager, à moyen terme, d'établir un ensemble de règles permettant de choisir a priori, en fonction des caractéristiques des trafics TR et non-TR, la meilleure politique pour l'application considérée.

Dans ce chapitre, comme de façon générale au cours de cette thèse, nous avons choisi comme cadre d'application les protocoles de la famille des bus à priorité, et particuliérement le réseau CAN. Néanmoins, ce travail peut être adapté à d'autres protocoles donnant des

garanties sur le temps de réponse, nous pensons en particulier aux protocoles à jetons (*Token Bus, Token Ring, FDDI ...*).

La politique DP est du type "meilleur effort" (*"best effort"*) pour la classe de trafic non-TR; une perspective de recherche serait de proposer des mécanismes permettant d'offrir des garanties, même probabilistes, sur le respect du trafic non-TR. Une piste qu'il nous paraît intéressant d'étudier est d'associer à la politique DP des mécanismes de contrôle de flux sur le trafic non-TR qui nous permettent d'accepter un message et de le garantir avec une probabilité choisie ou de le refuser. Sur l'occurrence d'une requête de transfert d'une trame non-TR, on pourrait imaginer par exemple deux réponses possibles, (1) le message est accepté et son échéance garantie avec la probabilité spécifiée lors de la requête ou (2) le message est refusé car la probabilité de respect des échéances ne peut être assurée.

L'inconvénient majeur de la politique DP, qui par ailleurs est d'un principe fort simple et qui s'est révelée remarquablement efficace, est le fait que le changement dynamique des priorités soit incompatible avec l'utilisation des contrôleurs de communication existants. Dans le chapitre suivant, nous proposerons une politique qui poursuit les mêmes objectifs tout en restant implantable sur le hardware existant.

Chapitre 6

Lissage de flux sous contraintes temps réel

Résumé

Nous nous plaçons dans le même cadre d'étude que lors chapitre précédent, deux types de trafic coexistent : du trafic à contraintes strictes dit Temps Réel (TR) et du trafic à contraintes souples dit non-Temps Réel (non-TR). L'objectif est toujours de minimiser le temps de réponse du trafic non-TR en garantissant les contraintes pesant sur le trafic TR. Dans cet optique, nous proposons une politique d'ordonnancement à lissage de flux (*"traffic shaping"*) dont la faible complexité permet son utilisation "en-ligne" et qui, contrairement à DP, reste compatible avec les contrôleurs de communication existants.

L'idée de base est qu'il est possible de réduire le temps de réponse du trafic à contraintes souples si les périodes d'activité engendrées par le trafic à contraintes strictes sont "régulièrement" réparties dans le temps, créant ainsi des intervalles de temps durant lesquels la ressource (le processeur ou le médium de communication) peut être utilisée par du trafic non-TR avec un minimum de délai. La politique que nous proposons s'applique aussi bien à l'ordonnancement de tâches que de messages, néanmoins dans ce chapitre nous l'appliquerons exclusivement à l'ordonnancement de messages. Nous prouverons que cette politique préserve la *faisabilité* du système, capitale pour le temps réel, c'est à dire qu'étant donné un ensemble de messages TR ordonnançable sous BS, celui-ci sera également ordonnançable sous cette politique. Les performances de la politique proposée seront comparées, en termes de temps de réponse moyen et de variance des temps de réponse du trafic non-TR, avec les politiques BS et DP.

6.1 Introduction

Objectifs Comme dans le chapitre précédent, nous traiterons du problème d'ordonnancer deux types de trafic avec des objectifs différents :

1. garantir que les contraintes temporelles du trafic TR seront respectées,
2. minimiser autant que possible le temps de réponse du trafic non-TR tout en satisfaisant le premier objectif.

Néanmoins, nous nous attacherons à proposer une solution qui, contrairement à DP, reste compatible avec les contrôleurs de communication disponibles commercialement. Une autre contrainte que nous nous imposons est que la complexité algorithmique de l'approche soit faible de façon à minimiser les overheads.

Organisation du chapitre Dans la section 6.2, nous introduisons le cadre et les notations du problème. Ensuite, en section 6.3, nous verrons que des travaux menés dans un contexte différent sont applicables à notre problème particulier mais qu'ils ne le résolvent que partiellement. La section 6.4 est consacrée à la politique à lissage de flux : ses principes de fonctionnement, la preuve qu'elle préserve toujours la faisabilité d'un ensemble de messages et l'algorithme permettant son l'implantation. Certaines extensions au cadre initial de la Section 6.2 seront proposées en section 6.5. Enfin, en section 6.6, nous évaluerons les performances de notre proposition par rapport aux politiques BS et DP.

6.2 Cadre de l'étude

Le cadre de l'étude et les notations sont les mêmes que dans le chapitre précédent (cf. section 5.2) avec néanmoins une hypothèse supplémentaire qui est que les instants auxquels les stations émettent une requête de transfert de données doivent être des multiples de la granularité du temps, c'est à dire des valeurs entières. De la même façon que la période T_k, la borne sur le temps de réponse R_k sera un multiple de la granularité du temps (en arrondissant le résultat donné par l'équation (2.1) du chapitre 2 à la valeur entière immédiatement supérieure). Pour des applications de multiplexage véhicule, une granularité de 0,5 ms semble être un bon choix (cf. le jeu d'essai du tableau 1.1 dans le chapitre 1) alors que 5 ou 10 ms suffiront pour la majorité des applications multimédia. Dans une première étape, nous ferons les hypothèses suivantes qui seront relevées ultérieurement dans la Section 6.5 :

- l'échéance de tout message m_k est égale sa période ($T_k = D_k$),
- il n'y a pas de gigue en émission ($A_{k,n} = n \cdot T_k$),
- les stations sont synchronisées dans le sens où elles commencent toutes à émettre simultanément à la mise en marche du système,

- le médium de communication est parfaitement fiable; il n'y a pas d'erreurs de transmission.

On introduit une nouvelle notation: $\mathbf{R}_k = D_k - R_k$. Notons que sous BS, le système est faisable ssi $\forall k$ t.q. $m_k \in \mathcal{H}$, $\mathbf{R}_k \geq 0$. La politique que nous proposons est basée sur le fait qu'il est possible de choisir l'instant d'émission de toute instance TR $m_{k,n}$ entre $A_{k,n}$ et $A_{k,n}+\mathbf{R}_{k,n}$ (avec pour tout n, $\mathbf{R}_{k,n} = \mathbf{R}_k$) de telle façon que les performances du trafic non-TR soient améliorées.

6.3 Minimisation du temps de réponse du trafic non-TR : une première approche

Le système étudié se compose d'un certain nombre de stations, interconnectées par un bus à priorité, qui génèrent à la fois du trafic TR et du trafic non-TR. Sur chacune des stations, il existe une file d'attente[35] dans laquelle les messages en attente d'émission sont triés par ordre de priorité; c'est l'arbitrage "interne" à la station. Le message le plus prioritaire de chacune des stations participera à l'arbitrage sur le bus qui assure que le plus prioritaire d'entres tous sera effectivement transmis. Conceptuellement, notre système peut donc se résumer à une file d'attente globale, priorisée, avec un serveur unique qui est la ressource partagée, à savoir le médium de transmission. Considérons que le flux d'entrée de la file d'attente est constitué uniquement du trafic non-TR et que les transmissions TR correspondent à des vacances du serveur. Le problème se résument à choisir judicieusement les instants de vacances, c'est à dire les instants de transmission du trafic TR, de façon à minimiser le temps de réponse du trafic non-TR. Moyennant quelques hypothèses, nous pouvons réutiliser directement des résultats présentés dans un cadre plus général [Gaujal *et al.*, 1998] et obtenir un résultat optimal. Les hypothèses que notre système devra satisfaire sont les suivantes :

- les messages non-TR arrivent dans la file $G/G/1$ selon un processus stationnaire,
- le temps de transmission de tous les messages du système est exactement de une unité de temps,
- tous les messages TR sont plus prioritaires que les messages non-TR, ces derniers ayant une priorité relative donnée par leur ordre d'arrivée (i.e. le système est FIFO pour les messages non-TR),

On note $T = \text{ppcm}_k T_k$. Un message TR (périodique par hypothèse) m_k sera transmis q_k fois sur l'intervalle $[0,T]$. Pour chaque slot de temps, il faut décider si l'on émet ou non un message

35. L'implantation au niveau du contrôleur de communication est généralement différente, mais fonctionnellement le résultat est identique.

TR. On construit une suite a_1,a_2,...,a_T ou la valeur de a_n nous indique si l'on doit transmettre ou non durant le $n^{\text{ième}}$ slot de temps. Il faut trouver la meilleure politique qui satisfait

$$\sum_{n=1}^{T} a_n = \sum_{k=1}^{p} q_k \tag{6.1}$$

c'est à dire la séquence des a_i qui minimise l'espérance du temps moyen d'attente des messages non-TR dans la file $G/G/1$ (avec $p = card(\mathcal{H})$ dans l'équation 6.1, cf. section 5.2). Ce type de système a été étudié dans [Gaujal *et al.*, 1998] ou il a été prouvé que cette séquence se construit de la façon suivante : soit $u = \sum_{k=1}^{p} q_k/T$ la densité moyenne d'émission de messages TR sur un ppcm, la séquence optimale est

$$a_n^* = \lceil nu \rceil - \lceil (n-1)u \rceil \text{ avec } a_0^* = 0. \tag{6.2}$$

Un exemple de cette construction est donné sur la figure 6.1.

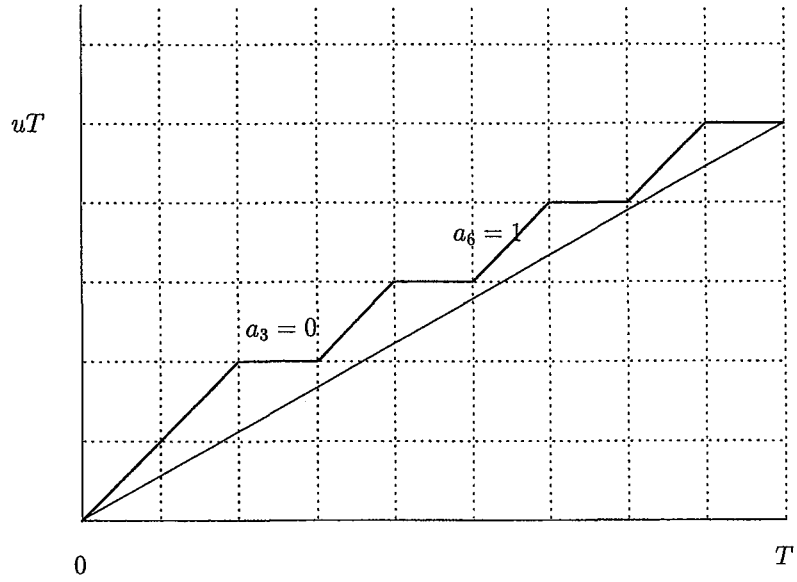


FIG. 6.1 – *Construction de a^* pour $T = 9$ et $u = 5/9$.*

L'enseignement fourni par cette analyse est que pour réduire l'espérance du temps de réponse du trafic non-TR, les périodes d'activités engendrées par le trafic TR doivent être espacées le plus régulièrement possible dans le temps. Néanmoins, l'allocation optimale donnée par l'équation (6.2) souffre d'inconvénients majeurs :

1. L'allocation n'est optimale que sous l'hypothèse FIFO pour le trafic apériodique.

2. Le temps de transmission de tous les messages du système doit être exactement de une unité de temps ce qui dans la pratique, au minimum, engendre de très fortes contraintes sur la conception du système, ou qui est même irréalisable sur certains réseaux. Ainsi avec CAN, pour un même nombre d'octets de données, le nombre de bits transmis effectivement sur le réseau dépendra du contenu même des informations (cf. le mécanisme de bit-stuffing au paragraphe 1.2.1).
3. La politique ne donne aucune garantie sur le respect des échéances ce qui ne permet pas d'envisager sont utilisation dans les applications TR.

Le problème que nous avons maintenant à résoudre est de distribuer la charge TR de façon régulière dans le temps tout en assurant la faisabilité du système. Pour cela, la définition de la densité u doit être raffinée pour prendre en compte les échéances. Nous perdrons l'optimalité de l'allocation mais nous gagnerons la faisabilité du système.

6.4 Lissage de flux avec une procédure en-ligne

Dans cette section, nous proposons une méthode qui se comporte mieux que BS en termes de temps de réponse du trafic non-TR et qui, contrairement à l'approche optimale présentée en section 6.3, garantit le respect des échéances. Cette approche se décompose en deux étapes :

1. La première étape est globale dans le sens où l'on ne distingue pas les messages. Il s'agit simplement de sélectionner les instants (ou slots) d'émission,
2. La seconde étape consiste à assigner une instance d'un message de $\mathcal{H}$ à chaque instant sélectionné pour une émission de telle façon que les échéances soient toujours respectées.

6.4.1 Densités et séquences d'émission

Nous décrivons ici la première étape de la procédure qui peut être vue comme un processus de sélection des instants d'émission. Comme nous en avons fait l'hypothèse, les instants d'émission sont discrétisés. Pour chaque slot de temps (dont l'instant de début est identifié par la variable i) et chaque message TR m_k, nous définissons le doublet (p_i^k, q_i^k) défini de façon unique par la division euclidienne de i par T_k :

$$i = p_i^k T_k + q_i^k \text{ avec } q_i^k < T_k.$$

Pour chaque message m_k, on construit une séquence de densités $\{u_i^k\}_{i\in\mathbb{N}}$:

$$u_i^k = \begin{cases} \frac{1}{T_k - R_k + 1} & \text{si } q_i^k \leq \mathrm{R}_k, \\ 0 & \text{sinon.} \end{cases} \tag{6.3}$$

Intuitivement, la séquence des densités représente la "quantité" de message m_k qui doit être alloué pour chaque slot de temps.
La double égalité suivante est une simple conséquence de la définition de u_i^k. Pour tout entier b et tout message TR m_k,

$$\sum_{i=bT_k}^{bT_k+\mathbf{R}_k} u_i^k = \sum_{i=bT_k}^{(b+1)T_k-1} u_i^k = 1. \tag{6.4}$$

La densité totale pour le slot i est définie par :

$$u_i = \sum_{k=1}^{p} u_i^k. \tag{6.5}$$

La somme des densités jusqu'au slot i est :

$$U_i = \sum_{j=0}^{i} u_j. \tag{6.6}$$

L'objectif du processus de sélection est de transformer la séquence des densités u_i à valeurs dans $\mathbb{R}$ en la séquence des sélections, notée v_i qui prend ses valeurs dans $\mathbb{N}$, de telle façon que v_i majorent u_i (i.e. $\forall t$, $\sum_{i=1}^{t} v_i \geq \sum_{i=1}^{t} u_i$) tout en restant le plus proche possible de u_i. La séquence des sélections v suivante satisfait cette propriété :

$$v_0 = \lceil u_0 \rceil, \quad v_i = \lceil U_i \rceil - \lceil U_{i-1} \rceil. \tag{6.7}$$

La construction d'une séquence de sélections à l'aide des relations (6.7) est assez classique dans les travaux sur le contrôle d'admission, cf. par exemple [Hajek, 1985]. Un exemple de construction de v est illustrée sur la Figure 6.2.

Résultat 1 *La complexité de l'étape 1 est linéaire en le nombre de slots et en le nombre de messages considérés.*

Démonstration: Le calcul des quantités u_i^k,u_i,U_i et v_i est linéaire sur i et sur k (cf. équations (6.4), (6.5), (6.6) et (6.7)). □

Remarque 2 *Il est suffisant de calculer v_i jusque $T = ppcm_k(T_k)$, la séquence des sélections se répétant après T.*

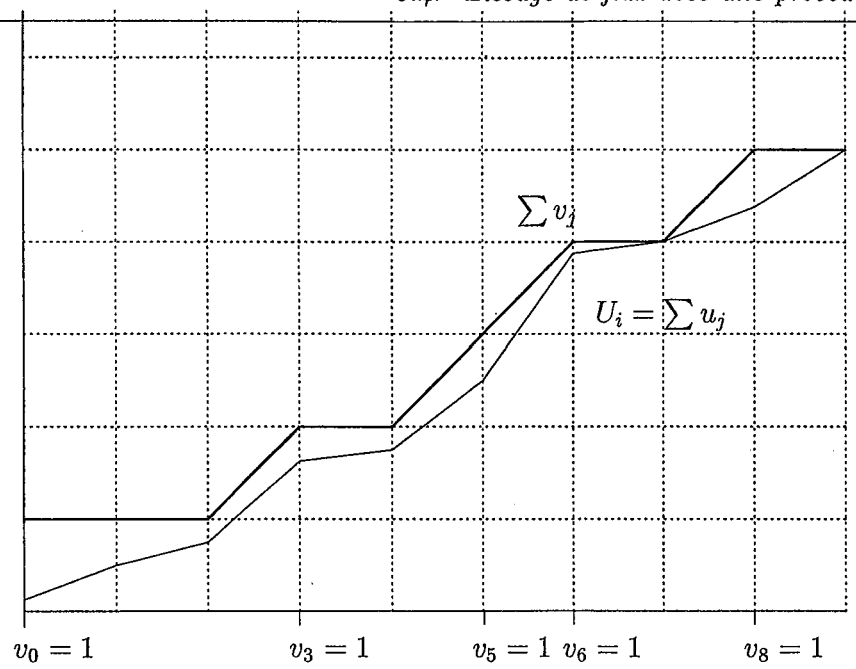


FIG. 6.2 – *Exemple de sélection pour laquelle un message sera émis aux instants 0,3,5,6 et 8.*

6.4.2 Allocation fonction des dates d'échéances

Une fois terminée la phase de sélection des instants d'émission, il faut choisir quel message émettre à chaque instant d'émission ($v_i = 1$). Cette procédure de sélection doit être peu complexe d'un point algorithmique de façon à limiter au maximum les overheads. Nous avons choisi la règle d'allocation "plus petite échéance d'abord" (*"Earliest Deadline first"*, EDF) et prouverons sa faisabilité.

Cette procédure d'allocation peut être vu comme la construction d'une séquence $\{a_i\}$, à valeurs dans $\mathcal{H}'$ où $\mathcal{H}'$ est l'ensemble des indices des messages composant $\mathcal{H}$, telle que a_i n'est définie que lorsque $v_i = 1$.

Remarque 3 *Une allocation est faisable si on alloue exactement une émission à tout message TR m_k entre l'instant de mise à disposition d'une de ses instances (bT_k) et le dernier instant d'émission garantissant le respect de la contrainte de temps associée à l'instance ($bT_k + \mathbf{R}_k$) :*

$$\forall k \in \mathcal{H}', \forall b \in \mathbb{N} \quad card\{a_j = k \mid bT_k \leq j \leq bT_k + \mathbf{R}_k\} = 1 \tag{6.8}$$

Comme $i = p_i^k T_k + q_i^k, q_i^k < T_k$ (division Euclidienne de i par T_k), un message m_k est en attente d'émission (*"pending"*) à l'instant i ssi $card\{a_j = k \mid j < i\} < p_i^k$. L'ensemble des indices de tous les messages en attente d'émission à l'instant i est noté P_i.

Définition 7 *Règle d'allocation "plus petite échéance d'abord": à tout instant i t.q. $v_i = 1$ sera alloué le message dont l'échéance est la plus proche, c'est à dire le message m_k t.q. :*

$$k = argmin_{k \in P_i}\{p_i^k T_k + \mathbf{R}_k\}, \tag{6.9}$$

où les égalités seront résolues arbitrairement (mais selon la même règle sur toutes les stations).

Théorème 2 *Si pour tout m_k, $\mathbf{R}_k \geq 0$, alors le processus de sélection associé à la règle d'allocation EDF préserve la faisabilité du système.*

Notons tout d'abord que s'il existe un m_k pour lequel $\mathbf{R}_k < 0$, il n'y a aucune chance de trouver une allocation qui soit faisable. Le théorème dit que si l'ensemble des messages TR est ordonnançable, alors le processus de sélection du paragraphe 6.4.1 en conjonction avec l'allocation EDF conserve la faisabilité. La preuve est donnée dans la section E.1 de l'annexe E.

6.4.3 Implantation informatique

L'algorithme qui implante l'analyse développée dans les sections 6.4.1 et 6.4.2 est présenté sur la figure 6.3. La procédure *Shaping*(), qui doit être exécutée sur chaque station indépendamment, décide pour chaque slot de temps, si un message TR doit être être transmis (étape de sélection) et, le cas échéant, choisit le message selon la règle EDF.
L'algorithme pourra être exécuté régulièrement avant chaque slot de temps mais il est également possible de mettre à profit les périodes d'inactivités de la station pour pré-calculer les émission sur de grandes périodes de temps. Dans ce cas, la requête de transfert de données (ligne 32 de la procédure) devra être retardée jusqu'à l'instant calculé.
Le paramètre *Pending* est un tableau de booléens servant à stocker les messages en attente de transmission : $Pending[k] = true$ signifie que le message m_k est en attente d'émission et qu'il peut peut donc être programmé pour un prochain slot. Le paramètre U_i est la somme des densités jusqu'à l'instant courant i.
L'algorithme peut se décomposer en 3 étapes :

1. Ajout des densités d'émission apportés par le prochain slot de temps (lignes 5-9 sur la figure 6.3).
2. Décision de la transmission ou non d'un message (lignes 10-22 sur la figure 6.3).
3. Si nécessaire, choix du message à transmettre (lignes 23-31 sur la figure 6.3) et, dans le cas où le message est produit par la station qui exécute l'algorithme, émission de la requête de transfert de données (fonction $queueMessage()$ à la ligne 32 de la figure 6.3).

Avant le premier appel à la procédure *Shaping*(), c'est à dire à la mise en marche de la station, i, *carry* et U_i doivent être initialisés à 0 et toutes les valeurs de $Pending[]$ à $false$.

Le paramètre *carry* (ligne 14, ligne 18) se justifie par le fait que la somme des densités sur

```
1  proc Shaping(real U_i, integer i, boolean Pending[p], integer carry)
2    real u_i, U_{i-1}, min;
3    integer m, a_i;
4    u_i := 0;
5    for m := 1 to p do
6        if (q_i^m ≤ R_m) then u_i := u_i + 1/(T_m − R_m + 1)
7                              if (q_i^m = 0) then Pending[m] := true; fi
8        fi
9    od
10   U_{i-1} := U_i;
11   U_i := U_i + u_i;
12   if (⌈U_i⌉ − ⌈U_{i-1}⌉ ≥ 1)
13     then v_i := 1;
14          carry := carry + ⌈U_i⌉ − ⌈U_{i-1}⌉ − 1;
15      else
16          if (carry > 0)
17            then v_i := 1;
18                 carry := carry − 1;
19             else
20                 v_i := 0;
21          fi
22   fi
23   min := +∞;
24   if (v_i = 1) then
25                  for m := 1 to p do
26                      if (Pending[m] ∧ (min > p_i^m T_m + R_m))
27                        then a_i := m;
28                             min := p_i^m T_m + R_m;
29                      fi
30                  od
31                  Pending[a_i] := false;
32                  if (sentByThisNode(a_i)) then queueMessage(a_i); fi
33   fi
```

FIG. 6.3 – *L'algorithme implantant la politique à lissage de flux.*

un même slot de temps peut être plus grande que un alors qu'il n'est pas possible d'allouer plus d'un message par intervalle de temps. Nous devons donc garder la trace des messages sélectionnés mais non-émis à l'instant i pour en tenir compte dans les instants suivants.
Comme cela a été prouvé précédemment, on voit sur la figure 6.3 que la complexité de la procédure est linéaire en le nombre de messages TR du système. Notons que la fréquence d'exécution de cet algorithme, et donc l'overhead induit par cette politique d'ordonnancement,

est fonction de la granularité choisie pour le temps.

6.5 Généralisation

Dans cette section, nous allons relaxer certaines hypothèses faites dans la section 6.2 en intégrant dans notre analyse des échéances plus petites que la période, des stations désynchronisées avec décalages connus et inconnus, des gigues en émission et des changements de mode de marche. Notons que le mécanisme donnant des garanties probabilistes sur le respect des échéances, proposé pour DP dans la section 5.5, peut être également réutilisé dans le cadre de la politique à lissage de flux.

6.5.1 Échéances plus petites que la période

Jusqu'à maintenant nous avons fait l'hypothèse que l'échéance de toute trame m_k était égale à sa période (i.e. $T_k = Dk$) ce qui est très usuel dans les applications de contrôle-commande basée sur la théorie de l'échantillonnage. Nous relaxons ici cette hypothèse en considérant que les échéances peuvent être inférieures aux périodes. Avec $D_k \leq T_k$, l'équation (6.3) devient :

$$u_i^k = \begin{cases} \frac{1}{D_k - R_k + 1} & \text{si } q_i^k \leq \mathbf{R}_k, \\ 0 & \text{sinon.} \end{cases} \tag{6.10}$$

Les autres résultats de la section 6.4 restent inchangés.

6.5.2 Stations désynchronisées avec des décalages connus

Le décalage entre l'instant d'activation de la première instance du message m_k et le point de référence 0 est noté ϕ_k. Nous avions jusqu'alors fait l'hypothèse que tous les décalages étaient nuls. Ici, nous permettons des décalages non-nuls en considérant toutefois que les stations connaissent ces décalages éventuels. Le calcul de u_i doit être adapté et l'équation (6.10) devient :

$$u_i^k = \begin{cases} \frac{1}{D_k - R_k + 1} & \text{si } i \geq \phi_k \text{ et } q_{i-\phi_k}^k \leq \mathbf{R}_k, \\ 0 & \text{sinon.} \end{cases} \tag{6.11}$$

L'ensemble des messages *pending* P_i doit également être redéfini, il devient

$$P_i' = \{k \text{ t.q. } card\{a_j = k \mid j < i\} < p_{i-\phi_k}^k\} \backslash \{k \mid i < \phi_k\}.$$

La suite du processus de sélection et d'allocation reste inchangé.

6.5.3 Stations désynchronisées avec des décalages arbitraires

Si les décalages initiaux ne sont pas connus, nous devons ajouter une phase d'initialisation pour synchroniser les stations un peu comme cela existe dans le protocole FIP (*Factory Instrumentation Protocol*, cf. [Thomesse *et al.*, 1991]). Il faut donc utiliser un protocole de synchronisation qui permette de définir un signal de départ global une fois que toutes les stations sont actives. Le signal de départ servira d'origine du temps pour le calcul des périodes. Pour les bus à priorités, nous suggérons le protocole suivant : à chaque station, on assigne un identificateur unique dans [1,n] où n est le nombre de stations. Une fois la station 1 active, elle émet une trame de priorité 1, si la station 2 est présente elle transmet immédiatement après la fin de transmission de la trame de priorité 1 sa trame de priorité 2. Si la station 2 n'a pas signalée sa présence après un certain délai, la station 1 réemet. Le même principe est valable pour les stations $3 \ldots n$: si une station n'est pas encore active, la station 1 réemet et le processus recommence. Lorsque l'on observe la séquence de trames $1,2,\ldots n$, on sait que toutes les stations sont actives et l'instant de fin de transmission de la $n^{\text{ième}}$ trame pourra servir d'origine globale pour le calcul des périodes.

Quel que soit le protocole, une synchronisation parfaite n'est pas envisageable, en particulier à cause du délai de propagation et de la granularité des horloges locales mais cette procédure de synchronisation nous permet raisonnablement d'envisager une réduction de tous les décalages à des valeurs plus petites que la granularité du temps choisie pour notre système. Sous cette hypothèse, toutes les stations sont synchronisées à plus ou moins une unité de temps ce qui peut engendrer des décalages de au plus une unité de temps sur les slots sélectionnés. Ces petites incohérences temporelles peuvent avoir néanmoins de lourdes conséquences et une notion plus forte de faisabilité doit être définie.

Définition 8 (marges de taille $\mathcal{K}$) *Si pour tout $m_p \in \mathcal{H}$ on a $\mathbf{R}_p \geq \mathcal{K}$, alors le système est faisable avec une marge de taille $\mathcal{K}$.*

Après l'étape de synchronisation, chaque station sélectionne les slots et alloue un message TR en attente à chaque slot sélectionné. Comme les stations ne sont synchronisées qu'à une unité de temps près, il est possible que les dates de sélections différent également d'une unité de temps. Néanmoins l'ordre dans lequel les messages sont alloués reste identique. Imaginons qu'une station 1 sélectionne un slot à un l'instant $n \cdot T_k + \mathbf{R}_k$ alors que la station 2 le sélectionne à l'instant $n \cdot T_k + \mathbf{R}_k + 1$. Si une instance de m_k est en attente, la station 1 devra allouer ce slot à cette instance. Pour la station 2, l'instance de m_k ne fait plus partie des messages pending car, ayant dépassé $n \cdot T_k + \mathbf{R}_k$, il n'y a plus de garantie sur le respect de l'échéance et le slot sera alloué à une instance d'un autre message. A partir de ce moment, les ensembles pendings sont différents et les décisions des stations peuvent devenir incohérentes. Il faut donc

empêcher qu'une instance de m_k soit programmée en transmission après $n \cdot T_k + \mathbf{R}_k - 1$, ce qui revient à distribuer la densité d'émission des instances de m_k sur une période de temps plus courte. L'équation (6.10) devient alors :

$$u_i^k = \begin{cases} \frac{1}{D_k - R_k} & \text{si } q_i^k \leq \mathbf{R}_k - 1, \\ 0 & \text{sinon.} \end{cases} \tag{6.12}$$

Sous l'hypothèse que le protocole de synchronisation réduise les décalages des différentes horloges locales à moins de une unité de temps, et si le système possède une marge de taille 1, alors la faisabilité est conservée.

6.5.4 Gigues en émission

Comme cela a été expliqué dans le chapitre 2, une instance d'un message peut être soumise à une certaine gigue : au lieu d'être disponible exactement à $n \cdot T_k + \phi_k$ (avec $\phi_k = 0$ dans le cas synchronisé), l'instance $m_{k,n}$ ne l'est effectivement qu'à un instant de l'intervalle $[n \cdot T_k + \phi_k - J_k, n \cdot T_k + \phi_k[\cup]n \cdot T_k + \phi_k, n \cdot T_k + \phi_k + J_k[$ (ou J_k est une borne sur la gigue qui est multiple de l'unité de temps). Si une instance $m_{k,n}$ est prête avant $n \cdot T_k + \phi_k$, il suffit de ne pas l'intégrer dans l'ensemble des messages en attente avant $n \cdot T_k + \phi_k$ (cf. équation (6.14)). Le problème est plus délicat lorsqu'une instance arrive plus tard que prévu, dans le pire cas au temps $n \cdot T_k + \phi_k + J_k$ car il lui reste un intervalle de temps plus court pour être transmis. Cette réduction du temps disponible est causé par l'augmentation de R_m qui doit maintenant intégrer la possibilité de gigue (cf. équation (2.1) dans le chapitre 2). En notant $j = i - \phi_k$ pour tenir compte de l'éventuelle désynchronisation, l'équation (6.11) devient :

$$u_i^k = \begin{cases} \frac{1}{D_k - R_k + 1} & \text{si } j \geq 0 \text{ et } J_k \leq q_j^k \leq \mathbf{R}_k, \\ 0 & \text{sinon.} \end{cases} \tag{6.13}$$

où R_k tient compte des gigues des messages. A nouveau, nous devons redéfinir P_i l'ensemble des messages en attente de transmission

$$P_i'' = \{k \text{ t.q. } card\{a_j = k \mid j < i\} < p_j^k \text{ et } p_j^k > J_k\} \backslash \{k \mid i < \phi_k\}. \tag{6.14}$$

6.5.5 Changements de mode de marche

Dans les applications TR de grande taille, on distingue généralement un certain nombre de phases opérationnelles prédéterminées, appelées aussi les *modes de marche* de l'application. Un exemple typique, donné dans [Kopetz *et al.*, 1995], est celui d'un système informatique embarqué dans un avion qui accomplira des tâches différentes lorsque l'appareil se trouve

au sol et en l'air. Les messages échangés sur le réseau dépendent des tâches actives et donc du mode de marche courant de l'application. Une des exigences définies dans [Kopetz *et al.*, 1995] pour des changements de mode de marche sûres de fonctionnement est la *consistence* qui impose que les différentes stations effectuent le changement de mode de marche simultanément. D'un point de vue pratique, le changement peut être signifié par un message spécial à qui l'on donnera la priorité la plus forte. Ensuite, chaque station devra réinitialiser les variables globales i, U_i, $Pending[]$ et $carry$ de l'algorithme présenté sur la figure 6.3 et considérer le trafic correspondant au mode de marche courant.

6.6 Expérimentations

Les performances de la politique à lissage de flux ont été évaluées en résolvant par simulation, à l'aide du logiciel *QNAP2*, un réseau de files d'attente modélisant le système. La partie TR du trafic est celle du jeu d'essai de la figure 5.1 à laquelle est adjoint un flux de messages non-TR dont les temps interarrivées sont exponentiellement distribués. Le débit du réseau CAN est fixé à 125kbit/s et nous ferons l'hypothèse que les messages périodiques ont une taille de 95 bits contre 75 bits pour les messages non-TR. Dans ces conditions, la partie TR du trafic génère une charge réseau de 41,02 %. Le temps de réponse moyen ainsi que la variance des temps de réponse du trafic non-TR ont été mesurés avec DP, BS et lissage de flux pour une charge réseau globale variant de 50 % à 90 %.

6.6.1 Le cas synchronisé

Dans le cas synchronisé toutes les stations commencent à émettre simultanément à la mise en marche du système.

Temps de réponse moyen du trafic non-TR Sur la figure 6.4, on observe que DP donne les meilleurs résultats parmi les trois politiques en concurrence. La stratégie à lissage de flux se comporte toutefois très bien jusqu'à une charge réseau de 70%, et pour des niveaux de charge supérieurs, elle reste toujours nettement plus efficace que BS. Dans nos expérimentations, le gain observé sur le temps de réponse moyen par rapport à BS varie entre un facteur 1,9 pour une charge de 50% et un facteur 1,4 pour une charge de 90%. Il est également intéressant de constater que quelle que soit la charge réseau, le gain absolu reste presque constant à environ 0,9 ms.

Variance des temps de réponse du trafic non-TR La figure 6.5 montre clairement que les politiques DP et lissage de flux diminuent fortement la variance des temps de réponse du

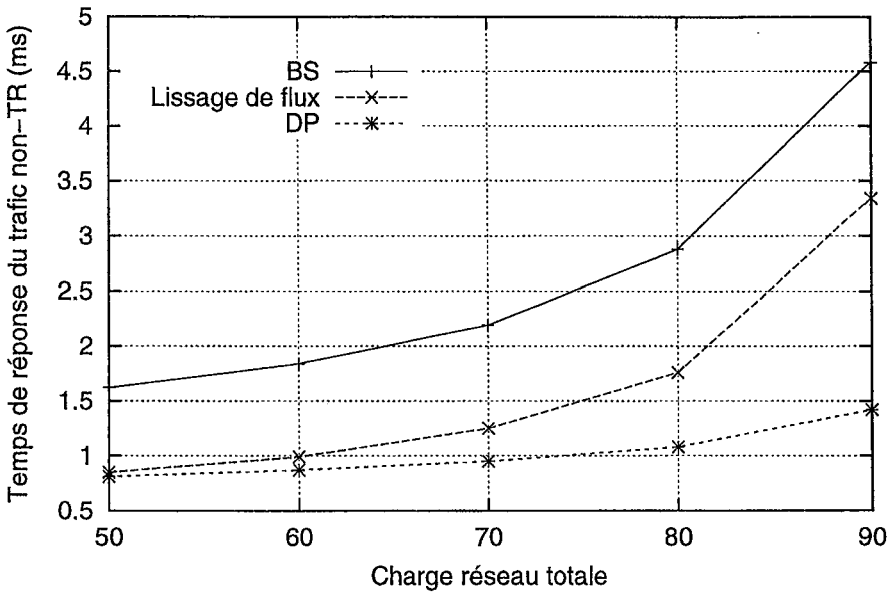


FIG. 6.4 – *Temps de réponse moyen du trafic non-TR avec BS, DP et lissage de flux dans le cas synchronisé.*

trafic non-TR. A nouveau, la politique à lissage de flux est toujours meilleure que BS et reste proche de DP jusqu'à 70-80%.

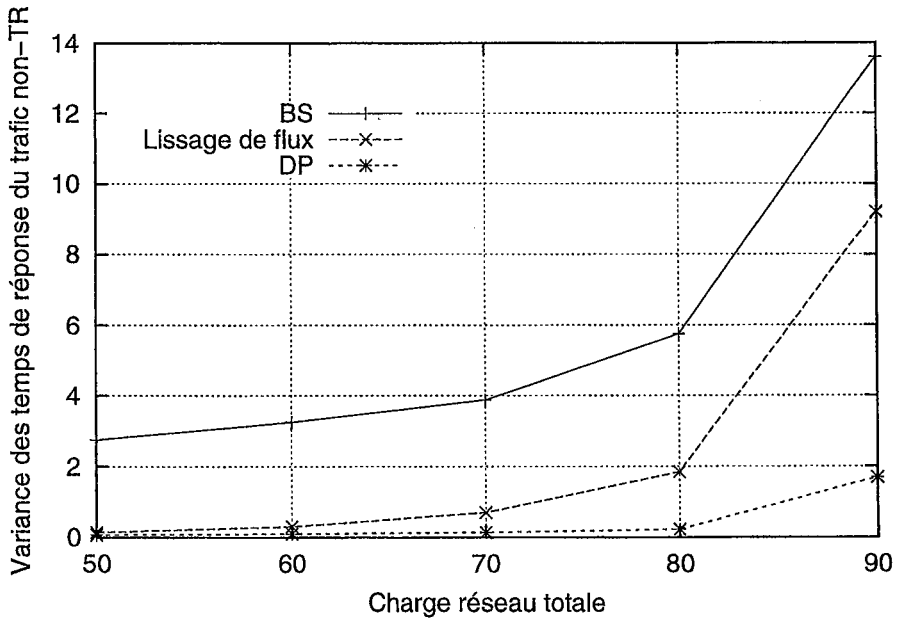


FIG. 6.5 – *Variance des temps de réponse du trafic non-TR avec BS, DP et lissage de flux dans le cas synchronisé.*

6.6.2 Le cas désynchronisé

Dans le cas désynchronisé, toutes les "sources" de messages ne deviennent pas actives au même instant : les simulations ont été réalisées avec un décalage initial ϕ_k tiré au hasard dans l'intervalle $[0,R_k]$.

Temps de réponse moyen du trafic non-TR En observant les figures 6.4 et 6.6, on remarque que le gain du lissage de flux par rapport à BS diminue quand les stations ne sont pas synchronisées : il se situe maintenant entre un facteur 1,28 et un facteur 1,19. Les performances de la politique à lissage de flux sont très proches dans le cas synchronisé et désynchronisé alors que BS se comporte beaucoup mieux dans le cas désynchronisé. L'explication est simplement que les décalages initiaux "lissent", même imparfaitement, le trafic. Néanmoins dans la pratique, ces décalages initiaux ne sont pas des variables indépendantes et il donc très vraisemblable que les performances de BS se situent en deçà des résultats présentés sur la figure 6.6.

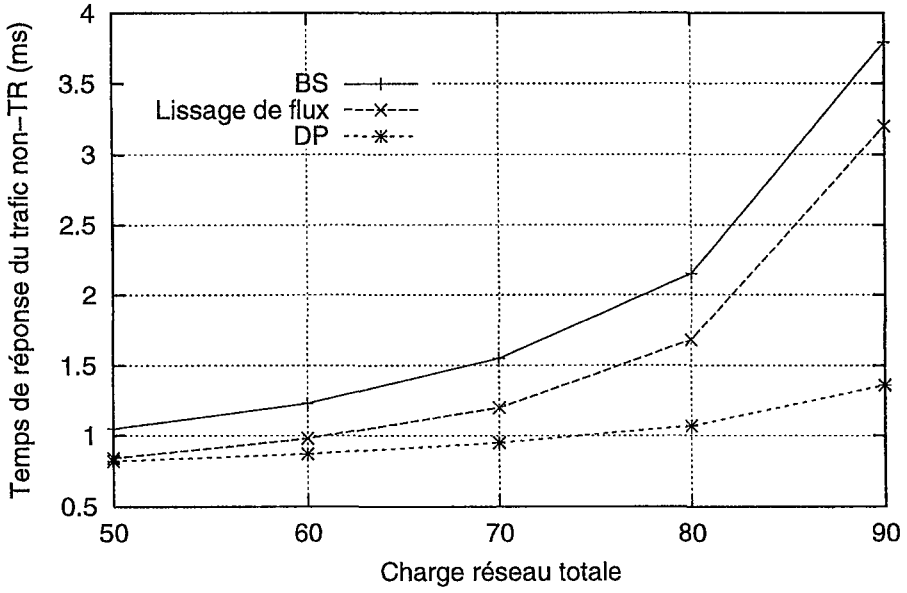


FIG. 6.6 – *Temps de réponse moyen du trafic non-TR avec BS, DP et lissage de flux dans le cas désynchronisé.*

Variance des temps de réponse du trafic non-TR Comme on peut l'observer sur la figure 6.7, et même si le gain est réduit par rapport au cas synchronisé, le lissage de flux est toujours meilleur que BS en termes de variance des temps de réponse du trafic non-TR. La remarque faite dans le paragraphe précédent reste valable; dans la pratique, les décalages

seront bien moins parfaits que ceux obtenus pendant la simulation ce qui aura pour effet d'augmenter la variance des temps de réponse sous BS.

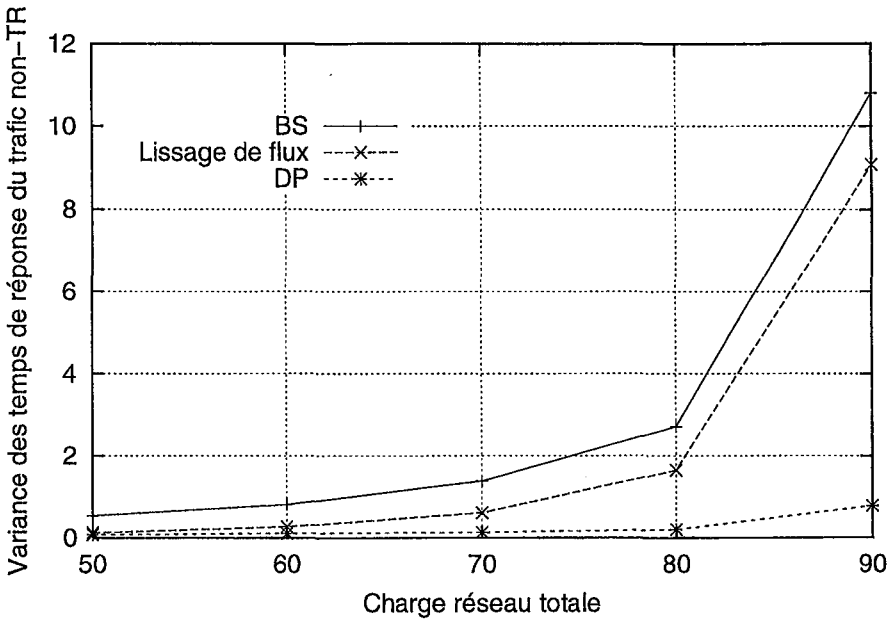


FIG. 6.7 – *Variance des temps de réponse du trafic non-TR avec BS, DP et lissage de flux dans le cas désynchronisé.*

6.7 Conclusion

Les expérimentations de la section 6.6 ont montré que les performances de notre proposition étaient toujours nettement supérieures à celles de la politique classique BS, et ceci, tout en préservant la faisabilité du système comme cela a été prouvé dans l'annexe E. Si à forte charge, DP est clairement plus efficace, notre proposition a en pratique l'avantage de ne pas requérir de modifications au niveau du contrôleur de communication. De plus, sa complexité algorithmique étant linéaire en le nombre de messages TR du système, elle peut être utilisée en-ligne avec un overhead minime en temps de calcul.

Cette politique d'ordonnancement est du type oisif, ce qui signifie que la ressource n'est pas nécessairement occupée si il y a du travail en attente. A notre connaissance, cette classe de politique est très peu étudiée dans la littérature récente. On peut penser, d'une façon générale, que ces politiques ne peuvent pas rivaliser en termes de performances avec les meilleures politiques non-oisives même lorsque l'ordonnancement est non-préemptif. Néanmoins, elles ne doivent pas être écartées a priori car il existe des situations ou la facilité d'implantation (qui passe ici par la compatibilité avec le matériel existant) prime sur les performances pures.

Ce travail, effectué en commun avec B. Gaujal, a fait l'objet d'un rapport de recherche [Gaujal and Navet, 1999a] et d'un article dans un journal [Gaujal and Navet, 1999b].

Conclusion et perspectives générales

Un des objectifs de cette thèse est de proposer des techniques de validation de systèmes temps réel qui soient d'une part pertinentes et rigoureuses, et d'autre part, pragmatiques. Le premier champ d'application de nos travaux est celui des applications embarquées dans les véhicules. Deux contrats industriels [Navet and Song, 1996; Song and Navet, 1997], réalisés pour PSA en début de thèse, nous ont permis de prendre conscience des contraintes et des problèmes particuliers aux applications embarquées de multiplexage véhicule. Par la suite, notre préoccupation a été de compléter les solutions initiées lors de ces contrats industriels.
L'idée forte qui ressort de ces travaux est que la validation doit être menée en couplant les techniques de vérification et ce, tout au long du cycle de vie de l'application. Initialement, nous nous sommes essentiellement intéressés aux techniques de simulation (construction de modèles, interfaçage de modèles, génération automatique de code), puis nous nous sommes tournés vers des techniques analytiques donnant des bornes sur les métriques de performance considérées (temps de réponse, probabilité de non-respect des échéances) ou permettant d'évaluer l'occurrence d'événements trop rares pour être évalués par simulation (temps d'atteinte de l'état bus-off). Parallèlement, nous avons développé un observateur réseau permettant de s'assurer du respect de certaines hypothèses faites lors de la conception des modèles, de fixer les valeurs de certains paramètres de ces modèles et de vérifier le respect de contraintes de niveau applicatif.
Notre souci a été d'intégrer des considérations de sûreté de fonctionnement aux analyses classiques de performances temporelles dans le pire cas. À cet égard, la contribution qui nous paraît la plus importante est la probabilité de non-respect des échéances associée au modèle stochastique d'erreurs. Conjointement avec Y.Q. Song ce travail nous a valu le prix *CAN In Automation Research Award* en 1997.
Beaucoup de travail reste à fournir dans le domaine de la conception d'applications embarquées dans les véhicules qui est une préoccupation relativement récente de l'informatique temps réel et qui évolue rapidement avec l'émergence de nouveaux standards comme OSEK/VDX et vraisemblablement, à moyen terme, de nouvelles méthodes de travail entre équipementiers et constructeurs automobiles. Cette réflexion sur la conception des applications embarquées est

l'enjeu de projets nationaux en cours de réalisation comme AEE[36] en France.

Une tendance forte de l'informatique temps réel est l'utilisation croissante de composants logiciels ou matériels du commerce. Ceci est vrai en particulier pour les systèmes d'exploitation temps réel où de nombreux exécutifs se conformant au standard Posix1003.1b sont disponibles commercialement. Une caractéristique remarquable de ces systèmes d'exploitation est qu'il est possible d'ordonnancer conjointement des tâches sous des politiques d'ordonnancement différentes. Les deux politiques standardisées *sched_fifo* et *sched_rr* sont, sous certaines hypothèses peu restrictives, respectivement équivalentes à FPP et Round-Robin. Dans un premier temps, nous avons proposé une analyse d'ordonnançabilité de ces systèmes en présence d'éventuelles sections critiques. Considérant ensuite qu'il existe généralement plusieurs solutions d'ordonnancement faisables à un même problème, nous nous sommes posés la question du choix de la meilleure solution pour l'application considérée. Des critères généraux permettant de choisir entre différentes configurations faisables ont été définis et nous avons expérimenté une approche, basée sur un algorithme génétique, pour parcourir l'espace des solutions faisables. Si la faisabilité est un pré-requis pour la mise en oeuvre d'une solution d'ordonnancement dans un contexte temps réel, notre conviction est qu'il faut également fonder son choix sur l'étude des répercutions de l'ordonnancement sur le processus physique sous contrôle. Les résultats de recherche dans cette direction sont à notre connaissance encore peu nombreux et il nous semble que des progrès significatifs ne pourront être réalisés qu'en intégrant plus étroitement les modèles issus de l'automatique (cf. [Juanole and Blum, 1999] pour une étude dans cette direction).

Le deuxième axe de recherche de cette thèse est la proposition de mécanismes d'ordonnancement qui garantissent le respect des échéances du trafic à contraintes strictes tout en minimisant les temps de réponse du trafic à contraintes souples.

Dans cette optique, nous avons tout d'abord étudié la politique d'ordonnancement à priorités dynamiques dite Dual-Priority. Des études antérieures ont montré par simulation que ses performances, dans le contexte de l'ordonnancement préemptif de tâches, sont très intéressantes et nous pensions qu'elle pourrait également se révéler efficace pour l'ordonnancement de messages. Nous avons évalué les performances de Dual-Priority sous l'hypothèse d'un médium fiable d'un point de vue quantitatif (en termes de temps de réponse et de variance des temps de réponse), et d'un point de vue plus qualitatif, en prouvant par une méthode trajectorielle qu'elle se comporte toujours mieux que la politique classique dite d'ordonnancement en arrière

36. Projet *Architecture Électronique Embarquée* auquel participe l'équipe TRIO et qui a débuté en Septembre 1998, cf. http://aee.inria.fr/. En Allemagne, le projet *Softmobil* poursuit des objectifs similaires, cf. http://www4.informatik.tu-muenchen.de/~beeck/gvp/ .

plan sous une hypothèse FIFO sur le trafic à contraintes souples. À l'heure actuelle, nous nous savons pas dans quelle mesure il est possible d'utiliser cette technique de comparaison trajectorielle sur d'autres politiques mais cette perspective permettrait de guider le concepteur d'application dans le choix des politiques à mettre en oeuvre.

Nous avons ensuite mis en évidence que Dual-Priority utilisé sur un médium potentiellement non fiable expose fortement le trafic à contraintes strictes à des dépassements d'échéances. Le problème identifié, nous avons proposé un mécanisme simple, dégradant faiblement les performances, permettant de garantir une QoS exprimée en termes de probabilité de respect des échéances. Enfin, nous avons proposé des mécanismes permettant de fixer "en-ligne" les paramètres du modèle d'erreurs dont l'utilisation s'impose lorsqu'il n'est pas possible de déterminer a priori le niveau de perturbation sur le bus ou lorsque celui-ci et soumis à de fortes variations. Les recherches en informatique temps réel ont essentiellement portées sur des systèmes "statiques" (trafic parfaitement identifié, environnement prévisible ..), il faut maintenant faire porter nos efforts sur la conception de systèmes capables de s'adapter dynamiquement à une évolution de l'application ou de son environnement et de fournir une qualité de service constante (exprimée par exemple de façon probabiliste) :

> "What is required is a science of performance guarantees that can provide a more formal analysis [than simulation or testing] of these dynamic real-time systems when the environment is not fully predictable or controllable and when failures occur ... there is a need for probabilistic guarantees that are applied to the general notion of QoS requirements" [Stankovic *et al.*, 1996].

Le travail que nous avons effectué sur la politique Dual-Priority trouve sa place dans cette problématique récente de la recherche en informatique temps réel.

Si Dual-Priority est une politique généralement très efficace, elle possède l'inconvénient certain de requérir des modifications au niveau du contrôleur de communication. Nous proposons une politique, basée sur une technique de lissage de flux, qui poursuit les mêmes objectifs tout en restant compatible avec les contrôleurs existants. Cette politique, comme cela a été prouvé, préserve la faisabilité du système et sa complexité algorithmique est linéaire en le nombre de messages à contraintes strictes du système ce qui lui permet d'être utilisée en-ligne avec un overhead minime en temps de calcul. D'autre part, la politique proposée pourra fonctionner en présence de stations désynchronisées avec décalages initiaux connus et inconnus, de gigues en émission et de changements de mode de marche. En termes de performances, cette politique à lissage de flux reste proche de Dual-Priority à charge faible ou modérée et se comporte toujours mieux que la politique d'ordonnancement en arrière plan.

Annexe A

Compléments sur le chapitre 2

Dans cette annexe, nous précisons les caractéristiques stochastiques du modèle d'erreurs proposé dans le chapitre 2 pour trois distributions différentes de la variable aléatoire (v.a.) u qui donnent la taille d'une rafale d'erreurs. Nous étudierons le cas où u suit la loi que nous avons appelée "géométrique modifiée", où u obéit à une loi uniforme et le cas où $u = 1$.

A.1 Le cas u suivant la loi géométrique modifiée

A.1.1 Caractéristiques de u

La v.a. u donne la taille des rafales d'erreurs. Nous allons calculer son espérance, sa variance après avoir vérifié que $\sum_{k=0}^{\infty} P[k;\ p] = 1$.

$$\sum_{k=0}^{\infty} P[k;\ p] = \frac{p^2}{q} \sum_{k=0}^{\infty} k q^k = \frac{p^2 q}{q(1-q)^2} = 1$$

espérance: $E(u) \doteq \sum_{k=0}^{\infty} k P(k;\ p)$

$$E(u) = \sum_{k=0}^{\infty} p^2 k^2 q^{k-1} = qp^2 \sum_{k=0}^{\infty} k^2 q^{k-2} = qp^2 \sum_{k=0}^{\infty} \left(k(k-1) q^{k-2} + k q^{k-2} \right)$$

$$E(u) = qp^2 \sum_{k=0}^{\infty} (q^k)'' + p^2 \sum_{k=0}^{\infty} (q^k)'$$

$$E(u) = qp^2 \left(\frac{1}{1-q}\right)'' + p^2 \left(\frac{1}{1-q}\right)' = qp^2 \left(\frac{1}{(1-q)^2}\right)' + p^2 \left(\frac{1}{(1-q)^2}\right)$$

$$E(u) = 1 + qp^2 \left(\frac{2}{(1-q)^3}\right) = 1 + \frac{2q}{p} = (2p^{-1} - 1)$$

variance: $Var(u) \doteq E[u^2] - (E[u])^2$

$$E[u]^2 = (2p^{-1} - 1)^2 = 4p^{-2} - 4p^{-1} + 1$$

$$E(u^2) = \sum_{k=0}^{\infty} k^2 P[k] = \sum_{k=0}^{\infty} k^3 q^{k-1} p^2 = p^2 q \sum_{k=0}^{\infty} k^3 q^{k-1}$$

$$E(u^2) = p^2 q \sum_{k=0}^{\infty} ((k+1)k(k-1)q^{k-2} + kq^{k-2}) = p^2 q \sum_{k=0}^{\infty} \left[(q^{k+1})''' + q^{-1}(q^k)' \right]$$

$$E(u^2) = p^2 q \left[(q \sum_{k=0}^{\infty} q^k)''' + q^{-1} (\sum_{k=0}^{\infty} q^k)' \right] = p^2 q \left[(\frac{q}{1-q})''' + q^{-1} (\frac{1}{1-q})' \right]$$

$$E(u^2) = p^2 q \left[\frac{6}{(1-q)^4} + \frac{1}{q(1-q)^2} \right] = p^2 (1-p) \left(6p^{-4} + \frac{1}{p^2(1-p)} \right)$$

$$E(u^2) = 6p^{-2} - 6p^{-1} + 1$$

$$Var(u) = 6p^{-2} - 6p^{-1} + 1 - 4p^{-2} + 4p^{-1} - 1 = 2p^{-2} - 2p^{-1} = \frac{2q}{p^2}$$

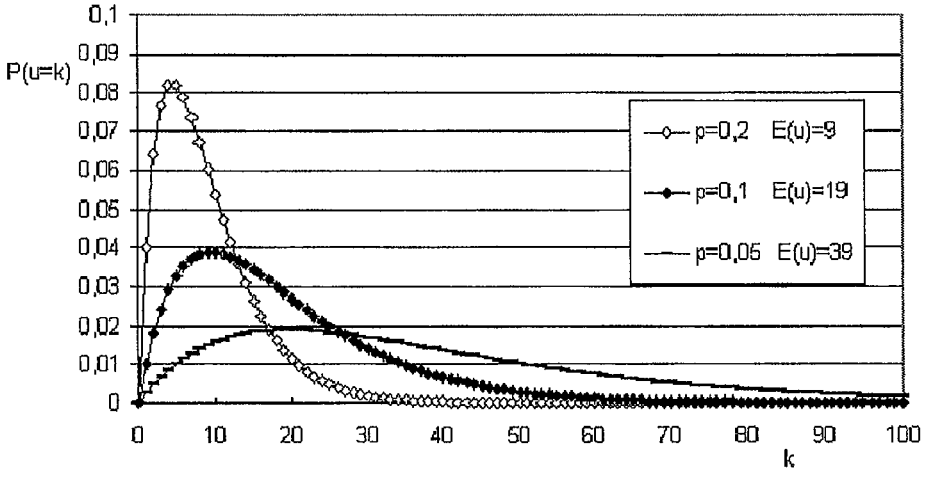


FIG. A.1 – *Distribution de u pour différentes valeurs de p.*

A.1.2 Caractéristiques de y

La taille des erreurs (erreurs atomiques et rafales d'erreurs) suit la v.a. y_i définie par:

$$y_i = \begin{cases} u \text{ avec une probabilité} & \alpha \\ 1 \text{ avec une probabilité} & 1-\alpha \end{cases}$$

Comme $\forall i \in \mathbb{N}$, y_i suit la même loi de probabilité, nous pouvons ignorer l'indice i. On sait que:

$$\begin{aligned} P(y=k) &= P[y=k \mid u]\,P[u] + P[y=k \mid 1]\,P[1] \\ &= \alpha k p^2 q^{k-1} + P[y=k \mid 1]\,(1-\alpha) \end{aligned}$$

comme $P[y=k \mid 1] = \begin{cases} 0 & k \geq 2 \\ 1 & k=1 \end{cases}$

on a:

$$P(y=k) = \begin{cases} \alpha k p^2 q^{k-1} & k \geq 2 \\ 1-\alpha+\alpha p^2 & k=1 \end{cases}$$

fonction génératrice: $y(z) \doteq E[z^y] = \sum_{k=0}^{\infty} z^k P[y=k]$

$$y(z) = 0 + (1-\alpha+\alpha p^2)z + \sum_{k=2}^{\infty} \alpha k p^2 q^{k-1} z^k$$

$$y(z) = (1-\alpha+\alpha p^2)z + \alpha p^2 z \left[\sum_{k=0}^{\infty} k(qz)^{k-1} - 1\right]$$

$$y(z) = (1-\alpha)z + \alpha p^2 z \left(\sum_{k=0}^{\infty}(qz)^k\right)\prime = (1-\alpha)z + \frac{\alpha p^2 z}{(1-qz)^2} \text{ avec } qz < 1$$

espérance: $E[y] \doteq \left.\frac{dy(z)}{dz}\right|_{z=1}$

$$E[y] = (1-\alpha) + \left.\frac{\alpha p^2(1-qz)^2 + 2q(1-qz)\alpha p^2 z}{(1-qz)^4}\right|_{z=1}$$

$$E[y] = (1-\alpha) + \left.\frac{\alpha p^2}{(1-qz)^2} + \frac{2\alpha p^2 qz}{(1-qz)^3}\right|_{z=1}$$

$$E[y] = 1 + \frac{2\alpha q}{p}$$

variance: $Var(y) \doteq E[y^2] - E[y]^2$

$$E[y^2] = \left.\frac{d^2y(z)}{dz^2}\right|_{z=1} + \left.\frac{dy(z)}{dz}\right|_{z=1}$$

$$\left.\frac{d^2y(z)}{dz^2}\right|_{z=1} = \left.\frac{2\alpha p^2 q(1-qz)}{(1-qz)^4}\right|_{z=1} + \left.\frac{2\alpha p^2 q(1-qz)^3 + 6\alpha(1-qz)^2 p^2 q^2 z}{(1-qz)^6}\right|_{z=1}$$

$$\left.\frac{d^2y(z)}{dz^2}\right|_{z=1} = \frac{4\alpha p^2 q}{(1-q)^3} + \frac{6\alpha p^2 q^2}{(1-q)^4} = \frac{4\alpha pq + 6\alpha q^2}{p^2}$$

$$Var(y) = \frac{(1-q+2\alpha q)}{(1-q)} - \frac{(1-q+2\alpha q)^2}{(1-q)^2} + \frac{4\alpha p^2 q}{(1-q)^3} + \frac{6\alpha p^2 q^2}{(1-q)^4}$$
$$Var(y) = \frac{2\alpha q\,(p+3q-2\alpha q)}{p^2}$$

A.1.3 Caractéristiques de $X(t)$

$X(t)$ étant un processus de Poisson généralisé, il a une fonction génératrice de la forme [Parzen, 1962] $X(z) = E[z^{y(z)-1}]$, avec $y(z)$ la fonction génératrice de la v.a. y. On a donc :

$$X(z) \doteq \sum_{k=0}^{\infty} z^k P[X(t)=k] = e^{\lambda t\left[(1-\alpha)z+\frac{\alpha p^2 z}{(1-qz)^2}-1\right]} \tag{A.1}$$

espérance: $E[X(t)] \doteq \left.\frac{dX(z)}{dz}\right|_{z=1}$

$$\begin{aligned} E[X(t)] &= \lambda t\left(1-\alpha+\tfrac{\alpha p^2}{(1-q)^2}+\tfrac{2\alpha p^2 q}{(1-q)^3}\right) e^{\lambda t\left[-\alpha+\frac{\alpha p^2}{(1-q)^2}\right]} \\ &= \lambda t\left(1+\tfrac{2\alpha q}{p}\right) = \lambda t\, E[y] \end{aligned}$$

variance: $Var[X(t)] \doteq E[X(t)^2] - E[X(t)]^2$

$$E[X(t)^2] = \left.\frac{d^2X(z)}{dz^2}\right|_{z=1} + \left.\frac{dX(z)}{dz}\right|_{z=1}$$

$$\begin{aligned} E[X(t)^2] &= \lambda t\left(4\frac{\alpha p^2 q}{(1-qz)^3}+6\frac{\alpha p^2 z q^2}{(1-qz)^4}\right) \mathrm{e}^{\lambda t\left[(1-\alpha)z+\frac{\alpha p^2 z}{(1-qz)^2}-1\right]}\Big|_{z=1} \\ &+ \lambda^2 t^2\left(1-\alpha+\frac{\alpha p^2}{(1-qz)^2}+2\frac{\alpha p^2 z q}{(1-qz)^3}\right)^2 \mathrm{e}^{\lambda t\left[(1-\alpha)z+\frac{\alpha p^2 z}{(1-qz)^2}-1\right]}\Big|_{z=1} \\ &+ \lambda t\left(1-\alpha+\frac{\alpha p^2}{(1-qz)^2}+2\frac{\alpha p^2 z q}{(1-qz)^3}\right) \mathrm{e}^{\lambda t\left[(1-\alpha)z+\frac{\alpha p^2 z}{(1-qz)^2}-1\right]}\Big|_{z=1} \end{aligned}$$

$$E[X(t)^2] = \frac{2\lambda t\alpha q\,(2+q) + \lambda^2 t^2\,(p+2\alpha q)^2}{p^2}$$

$$\begin{aligned} E[X(t)]^2 &= \lambda^2 t^2\left(1-\alpha+\frac{\alpha p^2}{(1-q)^2}+2\frac{\alpha p^2 q}{(1-q)^3}\right)^2 \left(\mathrm{e}^{lt\left[-\alpha+\frac{\alpha p^2}{(1-q)^2}\right]}\right)^2 \\ &= \lambda^2 t^2\left(1+\tfrac{2\alpha q}{p}\right)^2 \end{aligned}$$

$$\begin{aligned} Var[X(t)] &= \frac{2\lambda t\alpha q\,(2+q) + \lambda^2 t^2\,(p+2\alpha q)^2}{p^2} + \lambda t E[y] - \lambda^2 t^2 E[y]^2 \\ &= \frac{\lambda t}{p^2}\left(4\alpha q + 2\alpha q^2 + p^2 + 2\alpha q p\right) \\ &= \frac{\lambda t}{p^2}\left(1+(6\alpha-2)\,q+q^2\right) \end{aligned}$$

La figure A.2 représente la fonction de répartition de $X(t)$, $F(X(t)) \doteq P[X(t) \leq k]$ pour des valeurs de t et de k variables.

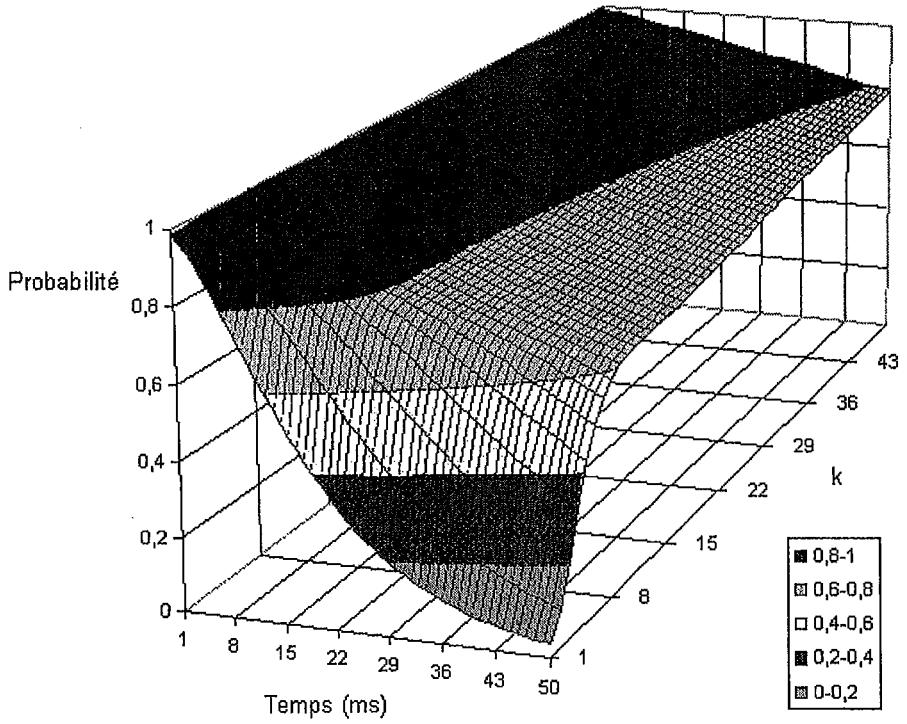


FIG. A.2 – $P[X(t) \leq k]$ *avec* $\alpha = 0.1$, $p = 0.04$, $\lambda = 100$ *pour un nombre d'erreurs tolérables* k *variant de 1 à 50 et pour le temps* t *variant de 1 à 50 ms.*

A.2 Le cas u suivant la loi uniforme

A.2.1 Caractéristiques de u

On suppose $a < b$ et $a > 1$.

$$P(u = k) = \frac{1}{b - a + 1}$$

fonction génératrice :

$$u(z) \doteq E[z^u] = \sum_{k=0}^{\infty} z^k P[u = k] = -\frac{z^{(b+1)} - z^a}{(b - a + 1)(1 - z)}$$

espérance: $E[u] \doteq \frac{du(z)}{dz}\Big|_{z=1}$

Ici, avec $u'(z) = \frac{(1-z)(-(b+1)z^b + az^{a-1}) - z^{b+1} + z^a}{(b-a+1)(1-z)^2}$, on se retrouve dans un cas indéterminé $\frac{0}{0}$.

Néanmoins, on peut écrire $E[u] = \lim_{z \to 1} u'(z)$ et d'après la règle d'Hopital on sait que $\lim_{z \to 1} \frac{g(z)}{h(z)} = \lim_{z \to 1} \frac{g'(z)}{h'(z)}$, d'où

$$\lim_{z \to 1} u'(z) = \lim_{z \to 1} \frac{-(b+1)bz^{b-1} + (a-1)az^{a-2} + (b+1)^2 z^b - a^2 za - 1 - (b+1)z^b + az^{a-1}}{-2(b - a + 1)(1 - z)}$$

$$\begin{aligned} \text{Or } \lim_{z \to 1} \{ & -(b+1)bz^{b-1} + (a-1)az^{a-2} + (b+1)^2 z^b - a^2 za - 1 - (b+1)z^b + az^{a-1} \} \\ & = -b^2 - b + a^2 - a + b^2 + 1 + 2b - a^2 + b - 1 + a = 0 \end{aligned}$$

et $\lim_{z \to 1} \{-2(b - a + 1)(1 - z)\} = 0$. On est donc à nouveau dans un cas indéterminé $\frac{0}{0}$ et on réapplique la règle d'Hopital :

$$\begin{aligned} \lim_{z \to 1} \frac{g(z)}{h(z)} &= \lim_{z \to 1} \frac{g'(z)}{h'(z)} = \lim_{z \to 1} \frac{g''(z)}{h''(z)} \\ &= \lim_{z \to 1} \frac{-(b+1)b(b-1)z^{b-2} + a(a-1)(a-2)z^{a-3} + b(b+1)^2 z^{b-1} - a^2(a-1)z^{a-2} - b(b+1)z^{b-1} + a(a-1)z^{a-2}}{2(b-a+1)} \\ &= \frac{b^2 + b - a^2 + a}{2(b - a + 1)} = \frac{(b - a + 1)(a - b)}{2(b - a + 1)} = \frac{a + b}{2} \end{aligned}$$

variance: $Var(u) \doteq E[u^2] - E[u]^2$

$$\begin{aligned} E[u^2] &= \lim_{z\to 1-} \left(\frac{d^2u(z)}{dz^2} + \frac{du(z)}{dz} \right) \\ &= \lim_{z\to 1-} \big(-(z^{(1+a)}\,a - 3\,z^b + 3\,z^a - 5\,z^a\,a + 5\,z^{(a-1)}\,a + 3\,z^{(b+1)}\,b - z^{(b-1)}\,b^2 \\ &\quad - z^{(b-1)}\,b - 2\,z^{(a-1)}\,a^2 - z^{(a-2)}\,a + 2\,z^b\,b^2 + z^{(a-2)}\,a^2 - z^{(b+1)}\,b^2 + z^a\,a^2 \\ &\quad - z^{(b+2)}\,b - z^b\,b + z^{(b+1)} - z^{(1+a)})/((b-a+1)\,(z-1)^3)\big) \\ &= \frac{1}{3}\,b^2 + \frac{1}{6}\,b + \frac{1}{3}\,b\,a - \frac{1}{6}\,a + \frac{1}{3}\,a^2 \end{aligned}$$

$$\begin{aligned} Var(u) &= \frac{1}{3}\,b^2 + \frac{1}{6}\,b + \frac{1}{3}\,b\,a - \frac{1}{6}\,a + \frac{1}{3}\,a^2 - \left(\frac{a+b}{2}\right)^2 \\ &= \frac{(b-a+1)^2 - 1}{12} \end{aligned}$$

A.2.2 Caractéristiques de y

La taille des erreurs (erreurs atomiques et rafales d'erreurs) suit la v.a. y_i définie par :

$$P(y_i = k) = \begin{cases} 0 & k \in [0,1[\cup]1,a[\cup]b,+\infty[\\ 1-\alpha & k = 1 \\ \alpha\,\dfrac{1}{b-a+1} & k \in (a,b] \end{cases}$$

À nouveau, dans la suite, nous négligerons l'indice i.

fonction génératrice: $y(z) \doteq E[z^y] = \sum_{k=0}^{\infty} z^k P[y=k]$

$$\begin{aligned} y(z) &= (1-\alpha)\,z + \alpha \sum_{k=a}^{b} \frac{1}{b-a+1}\,z^k \\ &= (1-\alpha)\,z - \frac{\alpha\,z^{b+1} - \alpha\,z^a}{(a-b-1)(z-1)} \end{aligned}$$

espérance: $E[y] = \lim_{z\to 1-} u'(z)$

$$\begin{aligned} E[y] &= \lim_{z\to 1-} \big(-(\alpha + 2\,z - b + a + 2\,b\,z - z^2 - b\,z^2 + a\,z^2 - 2\,a\,z + \alpha\,b \\ &\quad - \alpha\,a + \alpha\,z^2 - 2\,\alpha\,z + \alpha\,z^b - \alpha\,z^a + \alpha\,b\,z^2 - 2\,\alpha\,b\,z - \alpha\,a\,z^2 + 2\,\alpha\,a\,z \\ &\quad + \alpha\,z^b\,b - \alpha\,z^{(a-1)}\,a - \alpha\,z^{(b+1)}\,b + \alpha\,z^a\,a - 1)/((b-a+1)\,(z-1)^2)\big) \\ &= \frac{1}{2}\,\alpha\,b - \alpha + 1 + \frac{1}{2}\,\alpha\,a = \frac{1}{2}\alpha(b+a) - \alpha + 1 \end{aligned}$$

variance: $Var(u) \doteq E[u^2] - E[u]^2$

$$\begin{aligned}
E[y^2] &= \lim_{z\to 1-}\left(\frac{d^2y(z)}{dz^2}+\frac{dy(z)}{dz}\right)\\
&= \lim_{z\to 1-}\Big(\frac{\alpha\, z^{(b+1)}\,(b+1)^2}{(b-a+1)\,z^2\,(z-1)} - \frac{\alpha\, z^{(b+1)}\,(b+1)}{(b-a+1)\,z^2\,(z-1)} - 2\,\frac{\alpha\, z^{(b+1)}\,(b+1)}{(b-a+1)\,z\,(z-1)^2}\\
&\quad + 2\,\frac{\alpha\, z^{(b+1)}}{(b-a+1)\,(z-1)^3} - \frac{\alpha\, z^a\, a^2}{(b-a+1)\,z^2\,(z-1)} + \frac{\alpha\, z^a\, a}{(b-a+1)\,z^2\,(z-1)}\\
&\quad + 2\,\frac{\alpha\, z^a\, a}{(b-a+1)\,z\,(z-1)^2} - 2\,\frac{\alpha\, z^a}{(b-a+1)\,(z-1)^3} + \frac{\alpha\, z^{(b+1)}\,(b+1)}{(b-a+1)\,z\,(z-1)}\\
&\quad - \frac{\alpha\, z^{(b+1)}}{(b-a+1)\,(z-1)^2} - \frac{\alpha\, z^a\, a}{(b-a+1)\,z\,(z-1)} + \frac{\alpha\, z^a}{(b-a+1)\,(z-1)^2} + 1 - \alpha\Big)\\
&= \frac{1}{3}\alpha\, b^2 + \frac{1}{6}\alpha\, b + \frac{1}{3}\alpha\, a\, b - \alpha + 1 - \frac{1}{6}\alpha\, a + \frac{1}{3}\alpha\, a^2
\end{aligned}$$

$$\begin{aligned}
Var(y) &= \frac{1}{3}\alpha\, b^2 + \frac{1}{6}\alpha\, b + \frac{1}{3}\alpha\, a\, b - \alpha + 1 - \frac{1}{6}\alpha\, a + \frac{1}{3}\alpha\, a^2 - (\frac{1}{2}\alpha\, b + 1 - \alpha + \frac{1}{2}\alpha\, a)^2\\
&= \frac{1}{3}\alpha\, b^2 - \frac{5}{6}\alpha\, b + \frac{1}{3}\alpha\, a\, b + \alpha - \frac{7}{6}\alpha\, a + \frac{1}{3}\alpha\, a^2 - \frac{1}{4}\alpha^2\, b^2 + \alpha^2\, b - \frac{1}{2}\alpha^2\, b\, a - \alpha^2\\
&\quad + \alpha^2\, a - \frac{1}{4}\alpha^2\, a^2\\
&= \frac{1}{12}\alpha\,(4\,a^2 - 3\,\alpha\, a^2 + 12\alpha\, a - 14\,a - 6\,\alpha\, b\, a + 4\,b\, a - 3\,\alpha\, b^2 + 12\,\alpha\, b - 12\,\alpha + 12\\
&\quad - 10\,b + 4\,b^2)
\end{aligned}$$

A.2.3 Caractéristiques de $X(t)$

fonction génératrice:

$$X(z) = e^{\lambda t\left[(1-\alpha)\,z - \frac{\alpha\, z^{(b+1)} - \alpha\, z^a}{(a-b-1)\,(z-1)} - 1\right]}$$

espérance:

$$\begin{aligned}
E[X(t] = \lim_{z\to 1} X'(z) &= \lim_{z\to 1} \lambda\, t\left(1 - \alpha - \frac{\alpha\,(b+1)\, z^b - \alpha\, a\, z^{a-1}}{(a-b-1)\,(z-1)} + \frac{\alpha\, z^{(b+1)} - \alpha\, z^a}{(a-b-1)\,(z-1)^2}\right)\\
&\qquad e^{\lambda t\left[(1-\alpha)\,z - \frac{\alpha\, z^{(b+1)} - \alpha\, z^a}{(a-b-1)\,(z-1)} - 1\right]}\\
&= \frac{1}{2}\lambda\, t\,(\alpha\, a - 2\,\alpha + 2 + \alpha\, b)
\end{aligned}$$

variance: $Var((X(t)) \doteq E[X(t)^2] - E[(X(t)]^2$

$E[X(t)^2] = \lim_{z\to 1} X''(z) + \lim_{z\to 1} X'(z)$

$$\begin{aligned}\lim_{z\to 1} X''(z) &= \frac{1}{12}\lambda t(3\lambda\alpha^2 a^2 t + 4\alpha a^2 + 6\lambda\alpha^2 bat + 12\lambda\alpha at - 8\alpha a - 12\lambda\alpha^2 at + 4\alpha ba \\ &\quad +3\lambda\alpha^2 b^2 t - 12\lambda\alpha^2 bt - 24\lambda\alpha t + 12\lambda t + 12\lambda\alpha^2 t + 12\lambda\alpha bt - 4\alpha b + 4\alpha b^2)\end{aligned}$$

$$Var(X(t)) = \frac{1}{6}\lambda t(2\alpha b^2 + \alpha b + 2\alpha ba - 6\alpha + 6 - \alpha a + 2\alpha a^2)$$

A.3 Le cas $u = 1$

Dans le cas $u = 1$, il n'y a pas de rafales d'erreurs et la probabilité qu'il y ait plus de k erreurs dans $[0,t]$ est :

$$P[X(t) > k] = 1 - \sum_{m=0}^{k} \frac{e^{-\lambda t}(\lambda t)^m}{m!} \tag{A.2}$$

$X(t)$ suit un processus de Poisson, son espérance ainsi que sa variance valent donc λt.

Annexe B

Compléments sur le chapitre 3

B.1 Borne sur les temps de réponses : le cas FPP

L'objectif est de prouver que le temps de réponse de toute instance d'une tâche FPP est borné par le temps de réponse d'une instance de la même tâche dans la première période d'interférence du scénario d'activation majorant (*majorizing release pattern*). Le scénario d'activation majorant étant la séquence des mises à disposition $a_{k,n}$ (de temps d'exécution $c_{k,n}$, de temps de réponse $r_{k,n}$ et d'instant de fin d'exécution $e_{k,n}$) induit par la MWAF :

$$R_{k,n} \leqslant \max_{j \mid a_{k,j} < e_{k,j-1}} r_{k,j}.$$

Soit $\tilde{n}$ l'index de l'instance du scénario d'activation majorant tel que

$$a_{k,\tilde{n}} \leqslant A_{k,n} - U_{k,n} < a_{k,\tilde{n}+1}. \tag{B.1}$$

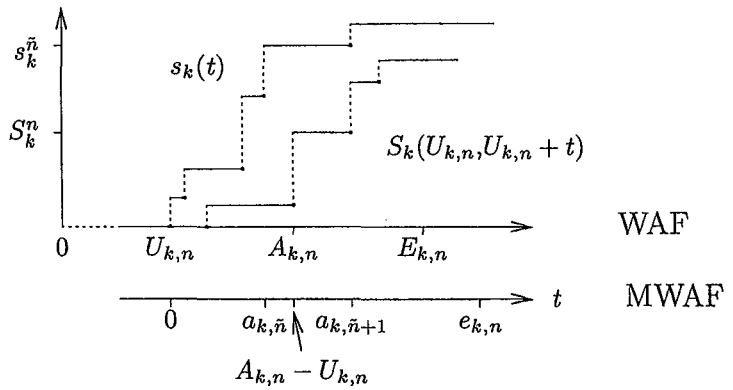


FIG. B.1 – *Illustration des inégalités* (3.3) *et* (B.1).

On note $s_k^n = \sum_{i=0}^{n} c_{k,i}$, la première étape de la preuve est de montrer que $S_k^n \leqslant s_k^{\tilde{n}}$ comme

cela est illustrée sur la figure B.1. Pour cela, deux cas doivent être distingués. Notons tout d'abord que $S_k^n = S_k(U_{k,n},A_{k,n}) + C_{k,n} = S_k(U_{k,n},A_{k,n+1})$.

- $A_{k,n+1} - U_{k,n} \leqslant a_{k,\tilde{n}+1}$: dans ce cas $s_k(A_{k,n+1} - U_{k,n}) \leqslant s_k(a_{k,\tilde{n}+1})$ puisque s_k est croissante. Avec (3.3), nous obtenons

$$S_k^n = S_k(U_{k,n},A_{k,n+1}) \leqslant s_k(A_{k,n+1} - U_{k,n}) \leqslant s_k(a_{k,\tilde{n}+1}) = s_k^{\tilde{n}}.$$

- $A_{k,n+1} - U_{k,n} > a_{k,\tilde{n}+1}$: dans ce cas $S_k(U_{k,n},A_{k,n+1}) = S_k(U_{k,n},U_{k,n} + a_{k,\tilde{n}+1})$, car par la définition de $\tilde{n}$, $a_{k,\tilde{n}+1} > A_{k,n} - U_{k,n}$. Avec (3.3), nous obtenons

$$S_k^n = S_k(U_{k,n},A_{k,n+1}) = S_k(U_{k,n},U_{k,n} + a_{k,\tilde{n}+1}) \leqslant s_k(a_{k,\tilde{n}+1}) = s_k^{\tilde{n}}.$$

On a prouvé que $S_k^n \leqslant s_k^{\tilde{n}}$. Avec l'inégalité 3.3, on peut maintenant écrire

$$S_{1..k-1}(U_{k,n},t) + S_k^n \leqslant s_{1..k-1}(U_{k,n} - t) + s_k^{\tilde{n}}$$

Soit f une fonction croissante $\mathbb{R} \longmapsto \mathbb{R}$ avec $f(0) > 0$ avec pour premier point fixe

$$x_1 = \min\{x > 0 \mid f(x) = x \} \tag{B.2}$$

Si f peut être bornée par une fonction $\hat{f}$ croissante alors le premier point fixe de $\hat{f}$ est soit égal soit supérieur au premier point fixe de f. Dans notre cas avec $f(x) = S_{1..k-1}(U_{k,n},U_{k,n}+x)+S_k^n$, $\hat{f}(x) = s_{1..k-1}(x) + s_k^{\tilde{n}}$ et $U_{k,n}$ le début de la période d'interférence contenant $\tau_{k,n}$, alors $x_1 = E_{k,n} - U_{k,n} \leqslant \hat{x}_1 = e_{k,\tilde{n}}$. Comme $A_{k,n} - U_{k,n} \geqslant a_{k,\tilde{n}}$, on a $R_{k,n} \leqslant r_{k,\tilde{n}}$.

Jusqu'à maintenant nous avons identifié pour chaque temps de réponse $R_{k,n}$ une borne $r_{k,\tilde{n}}$ où $\tilde{n}$ est le plus petit indice t.q. $S_k^n \leqslant s_k^{\tilde{n}}$. Le maximum des $r_{k,\tilde{n}}$ sera une borne sur tous les temps de réponse de la tâche τ_k. Il s'agit maintenant de savoir ou s'arrêter dans l'analyse des $r_{k,\tilde{n}}$.

Soit $\tau_{k,n}$ une instance de τ_k dans une période d'interférence quelconque. Dans l'intervalle $[U_{k,n},A_{k,n}[$, le processeur est occupé par des tâches plus prioritaires. D'après (3.3), comme la MWAF génère au moins autant de travail que la WAF, le processeur sera occupé pendant $[0,A_{k,n} - U_{k,n}[$ sur le scénario d'activation majorant. Comme par définition $a_{k,\tilde{n}} \leqslant A_{k,n} - U_{k,n}$, $\tau_{k,\tilde{n}}$ fait partie de la première période d'interférence du scénario d'activation majorant. Il suffit donc de calculer $r_{k,j}$ pour $j = 0,1,2,\ldots$ tant que $a_{k,j} < e_{k,j-1}$.

B.2 Borne sur les temps de réponses : le cas Round-Robin

L'objectif est ici de borner la quantité $\overline{W}_k(U) + \overline{S}_k(U,t) + S_{1..m_{l-1}}(U,t)$ de l'équation 3.9. La quantité de travail provenant des autres tâches de la couche RR qui reste à effectuer à l'instant U (début de la période d'interférence de la tâche RR τ_k considérée) est notée $\overline{W}_k(U)$. Cette quantité dépend du passé, plus précisément du résultat de l'ordonnancement depuis le début de la dernière période d'interférence de niveau m_l noté U^{m_l} (où m_l est le plus grand indice des tâches de la couche RR et U^{m_l} est donc le dernier instant auquel il n'y a pas de travail de niveau $1..m_l$). Notons que dans l'intervalle $[U^{m_l},U[$, seules les tâches $\tau_1,\ldots,\tau_{m_l}$ peuvent être exécutées. On peut écrire

$$W_{1..m_l}(U) = W_{1..m_{l-1}}(U) + W_k(U) + \overline{W}_k(U)$$

Par la définition de la période d'interférence RR, on sait que $W_k(U) = W_{1..m_{l-1}}(U) = 0$. On a donc :

$$\overline{W}_k(U) = W_{1..m_l}(U) = S_{1..m_l}(U^{m_l},U) - (U - U^{m_l}).$$

Comme $S_{1..m_l}(U^{m_l},U) = S_{1..m_{l-1}}(U^{m_l},U) + S_k(U^{m_l},U) + \overline{S}_k(U^{m_l},U)$ et parce que les WAFs peuvent être additionnés (i.e. $\overline{S}_k(U^{m_l},U)+\overline{S}_k(U,t) = \overline{S}_k(U^{m_l},t)$ et $S_{1..m_l}(U^{m_l},U)+S_{1..m_{l-1}}(U,t) = S_{1..m_l}(U^{m_l},t)$), on a :

$$\overline{W}_k(U) + \overline{S}_k(U,t) + S_{1..m_{l-1}}(U,t) = S_k(U^{m_l},U) + \overline{S}_k(U^{m_l},t) + S_{1..m_{l-1}}(U^{m_l},t) - (U - U^{m_l}) \tag{B.3}$$

Par la définition des MWAFs, on peut écrire :

$$\begin{aligned}\overline{W}_k(U) + \overline{S}_k(U,t) + S_{1..m_{l-1}}(U,t) \leqslant & s_k(U - U^{m_l}) + \overline{s}_k(t - U^{m_l}) + s_{1..m_{l-1}}(t - U^{m_l}) - (U - U^{m_l}) \\ = & s_k(u) + \overline{s}_k(t - U + u) + s_{1..m_{l-1}}(t - U + u) - u. \end{aligned} \tag{B.4}$$

avec $u = U - U_{m_l}$. Le problème est maintenant d'estimer dans quel intervalle peut varier cette quantité u (i.e. les valeurs de U^{m_l} et U dépendent de l'instance et de la trajectoire). La plus grande période d'interférence de niveau m_l est $v^{m_l} = \min\{x > 0 \mid s_{1..m_l}(x) = x\}$. On sait que $[U^{m_l},U[\subset [U^{m_l},V^{m_l}[$ (ou V^{m_l} est la fin de la période d'interférence de niveau m_l) et comme d'autre part $V^{m_l} - U^{m_l} \leqslant v^{m_l}$ (toute période d'interférence de niveau m_l est plus petite que la première période d'interférence de niveau m_l du scénario d'activation majorant),

on a $u = U - U^{m_l} \leqslant v^{m_l}$. Une borne sur $\overline{W}_k(U) + \overline{S}_k(U,t) + S_{1..m_{l-1}}(U,t)$, notée $s_k^*(t-U)$, est donc :

$$s_k^*(x) = \max_{0 \leqslant u \leqslant v^{m_l}} s_k(u) + \overline{s}_k(u+x) + s_{1..m_{l-1}}(u+x) - u. \tag{B.5}$$

Lorsque l'on considère la possibilité de sections critiques, la taille de la plus grande période d'interférence du scénario d'activation majorant v^{m_l} peut augmenter mécaniquement par l'accroissement de la quantité de travail plus prioritaire ou de même priorité. En notant b_k, la taille de la plus grande section critique (d'une tâche de priorité inférieure à la couche RR) de priorité plafond $\geq$ à celle de la couche RR, on obtient $v^{m_l} = \min\{x > 0 \mid b_k + s_{1..m_l}(x) = x\}$. Notons que cette section critique b_k ne peut se trouver qu'en tout début d'une période d'interférence si la tâche (de priorité inférieure) qui possède cette section critique a commencé son exécution avant le début de la période d'interférence. En effet, si la tâche de priorité inférieure ne débute pas son exécution avant le début de la période d'interférence, elle n'aura plus l'opportunité d'avoir le processeur (et donc de rentrer en section critique) avant la fin de la période d'interférence.

Annexe C

Compléments sur le chapitre 4

C.1 Procédure de "réparation" d'un chromosome

Il est possible qu'un chromosome C résultant de l'application d'un opérateur génétique, qu'il s'agisse du crossover ou de la mutation, ne vérifie pas les invariants 1,2 ou 3 (cf. paragraphe 4.2.3.1) qui ont été dérivés des hypothèses de travail. Il s'agit donc pour nous de corriger certains gènes de C avant de l'intégrer à la population de solutions.

Si on alloue aux tâches uniquement les politiques d'ordonnancement, il est toujours possible de trouver une assignation de priorités telle que les invariants soient vérifiés. La procédure de réparation ne consistera donc qu'à modifier certaines priorités de telle façon que C devienne compatible avec les invariants. Schématiquement, la procédure de réparation d'un chromosome C est la suivante :

```
1 foreach (i,j) such that (prio(C,i) = prio(C,j)) do
2   if not(pol(C,i) = pol(C,j) = RR)
3     then
4           k = choose(i,j);
5           allocatePriority(C,k);
6   fi
7 od
```

1. Lignes 1 et 2 : il n'y a que dans le cas où deux tâches sont RR qu'elles peuvent partager le même niveau de priorité, cf. invariants 1 et 2.
2. Ligne 4 : il s'agit maintenant de choisir entre i et j la tâche qui devra changer de priorité. Dans la mesure du possible, nous nous efforçons de conserver les gènes nouvellement créés par l'opérateur génétique.
3. Ligne 5 : une nouvelle priorité est allouée à la tâche k. Si k est ordonnancée sous FPP, elle se verra attribuer une priorité non-utilisée, si k est RR, nous choisirons également parmi les priorités des autres tâches RR.

C.2 Description du jeu d'essai comportant 20 tâches

Le jeu d'essai comportant 20 tâches qui a été utilisé pour estimer les performances de l'algorithme génétique dans la section 4.3.1 est décrit dans le tableau C.1(a). Le tableau C.1(b) présente la meilleure solution trouvée avec l'algorithme génétique avec comme critère de qualité de minimiser la gigue de fin d'exécution des tâches τ_9, τ_{10}, τ_{11}, τ_{12}, τ_{13}, τ_{14}, τ_{15}, τ_{16}, τ_{17}, τ_{18}, τ_{19} et τ_{20}, chaque tâche possédant le même poids dans le critère. Le quantum pour les tâches Round-Robin est fixé à 2 unités de temps.

On note $\rho_k = C_k/T_k$ le taux d'utilisation du CPU de la tâche τ_k et $\rho_{1..m}$ le taux d'utilisation cumulé de la tâche de plus forte priorité jusqu'à la tâche considérée. La laxité (*laxity*) de τ_k est définie comme $D_k - R_k$ où R_k est la borne sur le temps de réponse donnée par l'analyse d'ordonnançabilité.

Sur le tableau C.1(a), une des contraintes a priori est que les tâches τ_{20} et τ_{15} soient ordonnançées sous Round-Robin au niveau de priorité le plus faible que nous devons intialement considérer comme étant la priorité 19, les priorités des autres tâches n'étant pas encore déterminées. La meilleure solution préserve le niveau de priorité le plus faible mais comme nous imposons que les priorités utilisées soient contigues et que les tâches τ_8 et τ_{18} sous RR n'utilisent qu'une priorité, la priorité 19 devient 18 ce qui, en termes d'ordonnancement, ne change strictement rien.

tâche	C	T	D	prio.	pol.
τ_{20}	10	1000	1000	19	RR
τ_{19}	13	1000	1000		
τ_{18}	8	800	800		
τ_{17}	12	800	400		
τ_{16}	5	600	600		
τ_{15}	10	500	750	19	RR
τ_{14}	10	500	500		
τ_{13}	15	300	300		
τ_{12}	5	300	300		
τ_{11}	4	250	250		
τ_{10}	7	225	225		
τ_9	15	200	200		
τ_8	5	175	350		
τ_7	6	150	500		
τ_6	6	150	150	3	
τ_5	8	120	120		
τ_4	9	100	100		
τ_3	5	75	120		
τ_2	6	60	60	2	FPP
τ_1	7	50	50	1	FPP

(a) Le jeu d'essai à 20 tâches et ses contraintes a priori.

couche	tâche	prio.	R	laxité	ρ_k	$\rho_{1..m}$
Round Robin	τ_{20}	18	444	556	1.0	85.7
	τ_{15}	18	444	306	2.0	84.7
FPP	τ_{13}	17	297	3	5.0	82.7
	τ_{19}	16	282	718	1.3	77.7
	τ_{17}	15	269	131	1.5	76.4
	τ_9	14	189	11	7.5	74.9
	τ_3	13	120	0	6.7	67.4
	τ_4	12	99	1	9.0	60.7
	τ_5	11	90	30	6.7	51.7
	τ_{14}	10	82	418	2.0	45.1
	τ_{16}	9	72	528	0.8	43.1
	τ_{12}	8	67	233	1.7	42.2
	τ_7	7	49	451	4.0	40.6
	τ_{10}	6	43	182	3.1	36.6
	τ_{11}	5	36	214	1.6	33.5
Round Robin	τ_8	4	30	320	2.9	31.9
	τ_{18}	4	32	768	1.0	29.0
FPP	τ_6	3	19	131	4.0	28.0
	τ_2	2	13	47	10.0	24.0
	τ_1	1	7	43	14.0	14.0

(b) Meilleur ordonnancement trouvé par l'algorithme génétique à la $88^{\text{ième}}$ génération avec un critère de qualité égal à 144,66.

TAB. C.1 – *Le jeu d'essai à 20 tâches et sa solution sous-optimale.*

C.3 Description du jeu d'essai comportant 30 tâches

Le tableau C.2(a) décrit le jeu d'essai à 30 tâches utilisé dans la section 4.3.2. Le critère de qualité est de minimiser la gigue de fin d'exécution de toutes les tâches du jeu d'essai, chaque tâche ayant le même poids dans le calcul du critère. La meilleure solution trouvée par l'algorithme génétique est présentée sur le tableau C.2(b). Comme pour le jeu d'essai à 20 tâches, le quantum pour les tâches Round-Robin est de 2 unités de temps.

tâche	C	T	D	prio.	pol.
τ_{30}	50	5000	5000		
τ_{29}	40	2500	2500		
τ_{28}	15	2000	2000		
τ_{27}	60	1500	1000		
τ_{26}	19	1500	1500		
τ_{25}	10	1500	1000		
τ_{24}	10	1200	1200		
τ_{23}	12	1200	1200		
τ_{22}	15	1100	550		
τ_{21}	10	1000	1000	30	RR
τ_{20}	13	1000	1200	30	RR
τ_{19}	14	1000	1000		
τ_{18}	40	1000	500		
τ_{17}	16	1000	600		
τ_{16}	11	750	400		
τ_{15}	8	750	800		
τ_{14}	12	750	150		FPP
τ_{13}	14	750	200		FPP
τ_{12}	15	500	500		
τ_{11}	5	500	500		
τ_{10}	10	500	500		
τ_{9}	15	300	300		
τ_{8}	5	300	300		
τ_{7}	4	250	250		
τ_{6}	7	250	250		
τ_{5}	15	200	200		
τ_{4}	5	200	175		
τ_{3}	6	150	150	3	FPP
τ_{2}	5	50	50	2	FPP
τ_{1}	7	50	50	1	FPP

(a) Le jeu d'essai à 30 tâches et ses contraintes a priori.

couche	tâche	prio.	R	laxité	ρ_k	$\rho_{1..m}$
Round Robin	τ_{21}	26	977	23	1.0	82.8
	τ_{20}	26	980	220	1.3	81.8
FPP	τ_{27}	25	945	55	4.0	80.5
	τ_{30}	34	729	4271	1.0	76.5
	τ_{29}	23	597	1903	1.6	75.5
	τ_{18}	22	492	8	4.0	73.9
	τ_{26}	21	434	1066	1.3	69.9
	τ_{28}	20	383	1617	0.8	68.7
	τ_{19}	19	368	632	1.4	67.9
	τ_{25}	18	342	658	0.7	66.5
	τ_{9}	17	294	6	5.0	65.9
	τ_{17}	16	279	321	1.6	60.9
	τ_{12}	15	240	260	3.0	59.3
	τ_{5}	14	193	7	7.5	56.3
	τ_{13}	13	178	22	1.9	48.8
	τ_{14}	12	146	4	1.6	46.9
	τ_{16}	11	134	266	1.5	45.3
	τ_{10}	10	123	377	2.0	43.8
	τ_{23}	9	113	1087	1.0	41.8
	τ_{15}	8	89	711	1.1	40.8
Round Robin	τ_{11}	7	72	428	1.0	39.8
	τ_{22}	7	81	469	1.4	38.8
Round Robin	τ_{6}	6	47	203	2.8	37.4
	τ_{24}	6	49	1151	0.8	34.6
FPP	τ_{8}	5	32	268	1.7	33.8
Round Robin	τ_{7}	4	26	224	1.6	32.1
	τ_{4}	4	27	148	2.5	30.5
FPP	τ_{3}	3	18	132	4.0	28.0
	τ_{2}	2	12	38	10.0	24.0
	τ_{1}	1	7	43	14.0	14.0

(b) Meilleur ordonnancement trouvé par l'algorithme génétique à la $94^{\text{ième}}$ génération avec un critère de qualité égal à 937,12.

TAB. C.2 – *Le jeu d'essai à 30 tâches et sa solution sous-optimale.*

Annexe D

Compléments sur le chapitre 5

D.1 Influence d'un changement de priorité

Cet annexe est consacré à la preuve d'un Lemme technique utilisé dans la preuve du Théorème 1 du chapitre 5. Ce Lemme nous dit que les temps de réponse des instances du message de plus faible priorité sous BS, ne peuvent que s'améliorer si l'on alloue les priorités d'une façon différente.

Pour l'accès au bus, BS met en oeuvre une politique d'ordonnancement du type FPP non-préemptif. Sous cette politique, le temps de réponse d'une instance $m_{k,n}$ satisfait toujours l'équation suivante :

$$R_{k,n} = C_{k,n} + \min\left\{t \geq 0 \mid \Omega_{k,n}(A_{k,n}) + \Gamma_{k,n}(t) + \rho_{k,n}(A_{k,n}) = t\right\}, \tag{D.1}$$

où

- $\Omega_{k,n}(x)$ est la charge de travail plus prioritaire présente à l'instant x.
- $\Gamma_{k,n}(x) = \sum_{(i,j)} \mathbf{1}_{\{\pi(i,j)<\pi(k,n)\}} \mathbf{1}_{\{A_{k,n} \leq A_{i,j} \leq A_{k,n}+x\}} C_{i,j}$ est le travail plus prioritaire qui arrive entre $A_{k,n}$ et $A_{k,n} + x$.
- $\rho_{k,n}(x)$ est le temps de transmission restant pour une instance moins prioritaire en cours de transmission à l'instant x.

L'instant de fin d'exécution $E_{k,n}$ satisfait

$$\begin{aligned} E_{k,n} &= A_{k,n} + R_{k,n} \\ &= A_{k,n} + C_{k,n} + \min\left\{t \geq 0 \mid \Omega_{k,n}(A_{k,n}) + \Gamma_{k,n}(t) + \rho_{k,n}(A_{k,n}) = t\right\}. \end{aligned}$$

Nous construisons une nouvelle fonction de priorité π' de la façon suivante : $\pi'(m,n) = \pi(m,n)$ pour toutes les instances du message de plus faible priorité m_m. Pour tout autre message m_k, la valeur de $\pi'(k,i)$ est choisie arbitrairement (éventuellement plus grande que $\pi'(m,n)$).

Toutes les quantités se référant à la nouvelle fonction de priorité seront notées avec un signe prime.

Lemme 4 *Pour toute instance du message de plus faible priorité,* $m_{m,n}$, $E'_{m,n} \leq E_{m,n}$.

Démonstration: Considérons une instance du message plus faible priorité $m_{m,n}$. Cette instance appartient à un cluster $\mathcal{C}$ composé de toutes les instances s'exécutant dans une période d'interférence $[t_1,t_2[$. Dans la suite, nous ne considérerons que les instances de messages qui appartiennent à $\mathcal{C}$, les autres messages ne pouvant influer sur l'instant de fin d'exécution de $m_{m,n}$.

Sous les priorités initiales, comme à sa date d'activation $m_{m,n}$ est forcément l'instance de plus faible priorité active (c'est une instance du message le moins prioritaire, c'est la dernière instance activée avec une affectation des priorités telle que l'ordre FIFO soit respectée) , $\rho_{m,n}(A_{m,n}) = 0$ et

$$E_{m,n} = A_{m,n} + C_{m,n} + \min\left\{t \geq 0 \mid \Omega_{m,n}(A_{m,n}) + \Gamma_{m,n}(t) = t\right\}. \tag{D.2}$$

Sous les priorités π', $E'_{m,n}$ satisfait

$$E'_{m,n} = A_{m,n} + C_{m,n} + \min\left\{t \geq 0 \mid \Omega'_{m,n}(A_{m,n}) + \Gamma'_{m,n}(t) + \rho'_{m,n}(A_{m,n}) = t\right\}. \tag{D.3}$$

On peut remarquer dans l'équation (D.3) que le terme ρ' réapparaît ce qui se justifie par le fait que $m_{m,n}$ n'est pas forcément l'instance la moins prioritaire sous π'.

Pour comparer ces deux instants de fin d'exécution, nous allons examiner la valeur des termes composant les équations (D.2) et (D.3).

Comme $m_{m,n}$ est l'instance de plus faible priorité active à l'instant $A_{m,n}$, on a

$$\begin{aligned} \Omega_{m,n}(A_{m,n}) &= W(A_{m,n}) && \text{[D.4]} \\ &= W'(A_{m,n}) && \text{[D.5]} \\ &\geq \Omega'_{m,n}(A_{m,n}) + \rho'_{m,n}(A_{m,n}), && \text{[D.6]} \end{aligned}$$

où l'équation (D.4) vient du fait que lorsque l'instance $m_{m,n}$ est activée, elle a la priorité la plus faible sur l'ensemble du système, l'équation (D.5) du fait que la charge totale de travail en attente de deux politiques non-oisives (*non-idling*) est égale (cf. Lemme 1) et l'équation (D.6) de la définition de Ω' et ρ' qui ne sont que des parties de la charge de travail totale (il manque la charge de travail induite par les instances moins prioritaires qui ne sont pas en cours de transmission à l'instant $A_{m,n}$).

On sait en plus que

$$\Gamma_{m,n}(t) = \sum_{(i,j)} C_{i,j} \mathbf{1}_{\{\pi(i,j)<\pi(m,n)\}} \mathbf{1}_{\{A_{m,n} \leq A_{i,j} \leq A_{m,n}+t\}}, \quad [D.7]$$

$$\geq \sum_{(i,j)} C_{i,j} \mathbf{1}_{\{\pi'(i,j)<\pi'(m,n)\}} \mathbf{1}_{\{A_{m,n} \leq A_{i,j} \leq A_{m,n}+t\}} \quad [D.8]$$

$$= \Gamma'_{m,n}(t). \quad [D.9]$$

L'inégalité (D.8) vient du fait que l'ensemble des instances $(m_{i,j})$ t.q. $\pi(i,j) > \pi(m,n)$ qui arrive après $A_{m,n}$ est exactement l'ensemble $\{m_{m,i} \mid i > n\}$ qui est inclu dans l'ensemble des instances t.q. $\pi'(i,j) > \pi'(m,n)$ et arrivant après $A_{m,n}$ (puisque π' ne modifie pas les priorités parmi les instances de m_m). En considérant les ensembles complémentaires, on a donc, pour tout (i,j),

$$\mathbf{1}_{\{\pi(i,j)<\pi(m,n)\}} \mathbf{1}_{\{A_{m,n} \leq A_{i,j} \leq A_{m,n}+t\}} \geq \mathbf{1}_{\{\pi'(i,j)<\pi'(m,n)\}} \mathbf{1}_{\{A_{m,n} \leq A_{i,j} \leq A_{m,n}+t\}}.$$

En combinant les équations (D.6) et (D.9), on obtient $E_{m,n} \geq E'_{m,n}$.

□

D.2 Implications du Théorème 1

De nombreuses simulations publiées dans la littérature montrent que l'emploi de DP à la place de BS apporte un gain substantiel en termes de temps de réponse du trafic non-TR. A forte charge (> 70%), les gains sont généralement supérieurs à 60% (cf. les expérimentations 5.3 and 5.4 dans [Davis and Wellings, 1995a], la figure 4 dans [Bernat and Burns, 1997] et la figure 5.2 du chapitre 5). Dans cette section, nous présentons des résultats de simulation dans le cas non-préemptif qui ont pour objectif de montrer que si la propriété FIFO sur le trafic non-TR n'est pas respectée, les performances de DP peuvent devenir médiocres.

Dans la première expérimentation, la partie TR de l'application est composée des 12 messages de taille 125 bits du jeu d'essai décrit sur la figure 5.1. Toutes les stations commencent à émettre simultanément et le réseau du bus est de 125kbit/s. A ce trafic TR s'ajoute un flux de 15 messages non-TR de taille 100 bits dont les temps interarrivées sont exponentiellement distribués. La charge totale du réseau est de 90% dont 53,46% pour la partie TR.
De façon à montrer l'importance de l'ordre des priorités parmi le trafic non-TR, nous avons imposé que les instances de messages non-TR soit activée de telle façon que l'on obtienne l'ordre le plus proche possible de LIFO :

$$m_{27,1}, m_{26,1}, m_{25,1}, \cdots, m_{15,1}, m_{14,1}, m_{13,1}, m_{27,2}, \cdots$$

Les simulations sous DP et BS ont été chacune réalisées sur plus de 10000 instances de messages TR. Les résultats, représentés sur la figure D.1, montre le gain en termes de temps de réponse

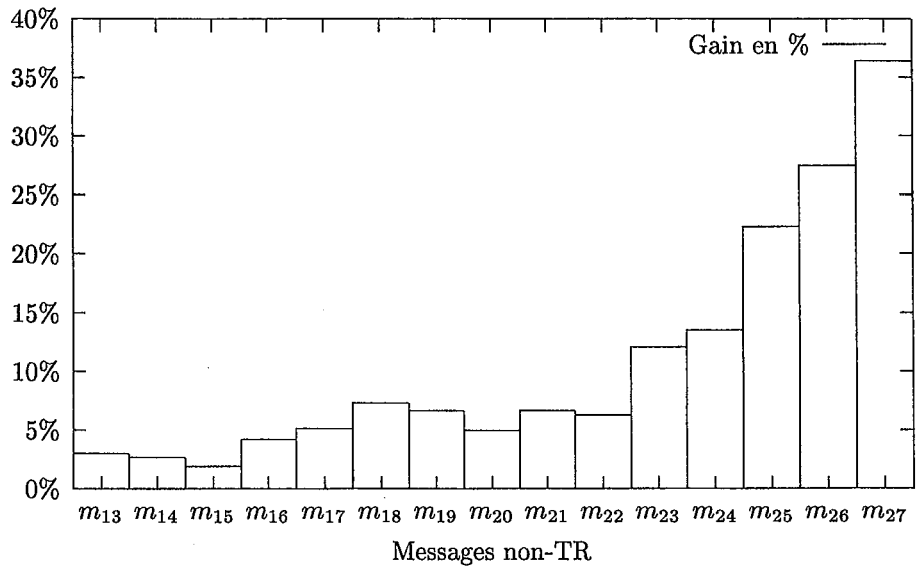


FIG. D.1 – *Gain en termes de temps de réponse moyen sous DP pour chaque message non-TR (avec 12 messages TR).*

moyen pour chaque message non-TR en utilisant la politique DP à la place de BS. Il est intéressant de noter que si le gain reste important pour les messages de plus faible priorité (m_{27}, m_{26} et m_{25}), ils sont beaucoup plus restreints pour les messages de priorités intermédiaires et fortes. Par exemple, le gain pour m_{15} est de l'ordre de 2% ce qui est très inférieurs à tous les résultats publiés dans la littérature.

Les conditions de la seconde expérimentation ont été fixées de façon à reproduire le comportement observé sur la figure 5.5 où BS fait mieux, pour certains messages non-TR, que DP. Le flux de messages non-TR est toujours composé de 15 messages de taille 100 bits mais il n'y a plus qu'un seul message TR m_1 de période 1 ms et de longueur 115 bits. Le débit étant maintenant de 250kbit/s, le message m_1 génère une charge de 46% alors que la charge totales est fixée à 95%. A nouveau, les messages non-TR sont activés dans un ordre quasi-LIFO :

$$m_{16,1},m_{15,1},m_{14,1},\cdots,m_{4,1},m_{3,1},m_{2,1},m_{16,2},\cdots$$

Les résultats de simulation sont représentés sur la figure D.2. Il est remarquable de constater que 4 des 15 messages non-TR (m_7, m_9, m_{13} et m_{16}) ont un temps de réponse moyen plus petit sous BS que sous DP. En particulier, la perte pour m_9 ordonnancé sous DP est de l'ordre de 5% ce qui est la différence la plus importante en valeur absolue.

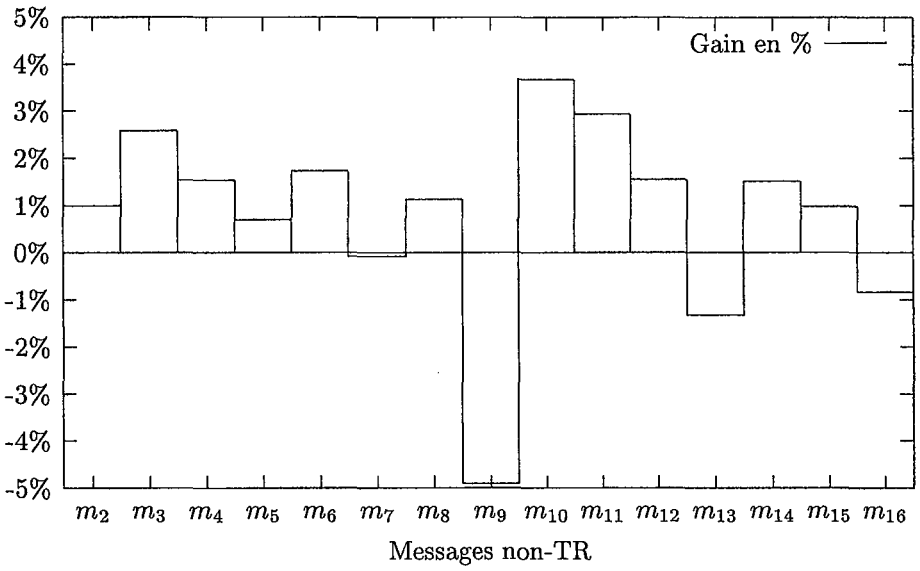


FIG. D.2 – *Gain en termes de temps de réponse moyen sous DP pour chaque message non-TR (avec 1 message TR).*

Annexe E

Compléments sur le chapitre 6

E.1 Preuve du théorème 2

Démonstration: La preuve se fait par récurrence sur le nombre de messages. Nous allons montrer que la faisabilité est vérifiée pour 1 message, l'admettre pour $p-1$ messages et montrer que l'ajout du $p^{\text{ième}}$ message préserve la faisabilité.

Avec un seul message ($\mathcal{H}' = \{1\}$), montrons que l'étape de sélection décide que $v_i = 1$ ssi $i = bT_1$, $\forall b \in \mathbb{N}$. C'est vrai pour $b = 0$, on a $v_0 = \lceil u_0 \rceil = \lceil u_0^1 \rceil = 1$ car $u_0^1 > 0$ par construction (cf. équation (6.3)). Nous l'admettons pour la $(b-1)$ième période. On sait que la somme des u_i^1 sur $[(b-1)T_1, bT_1 - 1]$ est égale à 1 par l'équation (6.4) et que $u_{bT1}^1 > 0$, on a donc $v_{bT_1} = \lceil U_{bT_1} \rceil - \lceil U_{pT_1-1} \rceil = 1$. On obtient donc $P_i = \{1\}$ ssi $i = bT_1$ sinon P_i est vide. La règle d'allocation EDF est clairement faisable puisque $a_i = 1$ pour tout $i = bT_1$ satisfait l'équation (6.8).

Nous avons maintenant $p > 1$ messages. Pour les premiers $p-1$ messages, nous faisons l'hypothèse que EDF est faisable et voulons prouver qu'elle le reste avec un message de plus (le message m_p). Toutes les quantités se référant au système avec $p-1$ messages (appelé système 1 dans la suite) seront préfixées par ^{1} alors que toutes les quantités calculées sur le système avec p messages (système 2) seront préfixées par ^{2}. La première assertion dont nous aurons besoin est que pour tout $b > 1$,

$$^2U_{bT_p-1} = {}^1U_{bT_p-1} + b, \qquad \text{[E.1]}$$

$$^2U_{bT_p+\mathbf{R}_p} = {}^1U_{bT_p+\mathbf{R}_p} + b + 1, \qquad \text{[E.2]}$$

L'égalité (E.1) est vérifiée pour $b = 1$. On admet que $^2U_{(b-1)T_p-1} = {}^1U_{(b-1)T_p-1} + b - 1$. On peut écrire $^2U_{bT_p-1} = {}^2U_{(b-1)T_p-1} + \sum_{i=(b-1)T_p}^{bT_p-1} u_i^p + {}^1u_{T_p}$. Avec l'équation (6.4) et l'hypothèse de récurrence, on obtient $^2U_{bT_p} = {}^1U_{bT_p} + b$. On pourra procéder de la même façon pour

(E.2). Les équations (E.1) et (E.2) nous enseigne que si l'on ajoute le message m_p au système 1, il y a exactement un point de sélection supplémentaire dans chaque intervalle du type $[bT_p,bT_p+\mathbf{R}_p]$ et le même nombre d'émissions dans $]bT_p+\mathbf{R}_p,(b+1)T_p[$.

Dans le système 1, appelons ${}^1i_1,\cdots,{}^1i_k$ les slots sélectionnés dans l'intervalle $[bT_p,bT_p+\mathbf{R}_p]$ et ${}^1i_{k+1},\cdots,{}^1i_{k+n}$ les slots dans $]bT_p+\mathbf{R}_p,(b+1)T_p[$. De la même façon, dans le système 2 on note ${}^2i_1,\cdots,{}^2i_k,{}^2i_{k+1}$ les instants d'émissions dans $[bT_p,bT_p+\mathbf{R}_p]$ et ${}^2i_{k+2},\cdots,{}^2i_{k+n+1}$ dans $]bT_p+\mathbf{R}_p,(b+1)T_p[$. La seconde assertion qu'il va falloir démontrer est que :

$$ {}^2i_j \leq {}^1i_j, \quad \forall 1 \leq j \leq k, \qquad \text{[E.3]} $$

$$ {}^2i_{j+1} = {}^1i_j, \quad \forall k+1 \leq j \leq k+n. \qquad \text{[E.4]} $$

Pour prouver cette assertion, considérons n'importe quel slot $i \in [bT_p,bT_p+\mathbf{R}_p]$. Avec l'équation (E.1), on peut écrire $\lceil {}^2U_i \rceil = \lceil {}^1U_i + \sum_{j=bT_p}^{i} u_i^p \rceil + b$ et $\lceil {}^2U_{bT_p-1} \rceil = \lceil {}^1U_{bT_p-1} \rceil + b$. On a donc :

$$ \sum_{j=bT_p}^{i} {}^2v_j = \lceil {}^2U_i \rceil - \lceil {}^2U_{bT_p-1} \rceil \geq \lceil {}^1U_i \rceil - \lceil {}^1U_{bT_p-1} \rceil = \sum_{j=bT_p}^{i} {}^1v_j. $$

Cela est vrai pour tout i et prouve que ${}^2i_j \leq {}^1i_1, \quad \forall 1 \leq j \leq k$ (équation E.3).

Pour l'équation (E.4) et pour tout slot $i \in]bT_p+\mathbf{R}_p,(b+1)T_p[$, $\lceil {}^2U_i \rceil = \lceil {}^1U_i \rceil + b + 1$. On a ainsi :

$$ \sum_{j=bT_p+\mathbf{R}_p+1}^{i} {}^2v_j = \lceil {}^2U_i \rceil - \lceil {}^2U_{bT_p+\mathbf{R}_p+1} \rceil = \lceil {}^1U_i \rceil - \lceil {}^1U_{bT_p+\mathbf{R}_p+1} \rceil = \sum_{j=bT_p}^{i} {}^1v_j. $$

Comme cela vaut pour tout i, et en considérant le fait que dans le système 2 il y a une allocation supplémentaire dans l'intervalle $[bT_p,bT_p+\mathbf{R}_p]$, cela montre que ${}^2i_{j+1} = {}^1i_j, \quad \forall k+1 \leq j \leq k+n$ (équation E.3).

La partie centrale de la preuve est faite par récurrence sur b. Nous nous intéressons maintenant au processus d'allocation proprement dit. Nous voulons prouver la troisième assertion qui est que pour tout $b \in \mathbb{N}$, on a :

$$ {}^2P_{bT_p} = {}^1P_{bT_p} \cup \{p\}, \qquad \text{[E.5]} $$

$$ card\{a_j = p \mid bT_p \leq j \leq bT_p + \mathbf{R}_p\} = 1 \qquad \text{[E.6]} $$

où P_i est l'ensemble des messages pendings à l'instant i. Considérons d'abord le premier intervalle $[0,\mathbf{R}_p]$. De façon évidente, on a : ${}^2P_0 = \mathcal{H}'$ et ${}^1P_0 = \mathcal{H}'/\{p\}$ ou exprimé autrement :

${}^2P_0 = {}^1P_0 \cup \{p\}$. Considérons que le système avec $p-1$ messages a k slots sélectionnés sur $[0,\mathbf{R}_p]$ et n sélections sur $]\mathbf{R}_p,T_p[$, respectivement notées ${}^1i_1,\cdots,{}^1i_k$ et ${}^1i_{k+1},\cdots,{}^1i_{k+n}$.
Les équations (E.1) et (E.2) impliquent que le système avec p messages a $k+1$ slots sélectionnés dans $[0,\mathbf{R}_p]$ et n sélections dans $]\mathbf{R}_p,T_p[$, noté respectivement ${}^2i_1,\cdots,{}^2i_k,{}^2i_{k+1}$ et ${}^2i_{k+2},\cdots,{}^2i_{k+n+1}$.
Dans le système 1, la règle EDF alloue les slots aux messages $\mathbf{m}_1,\cdots,\mathbf{m}_{n+k}$ dont la transmission doit être prévue respectivement avant $\mathbf{R}_{\mathbf{m}_1} \leq \cdots \leq \mathbf{R}_{\mathbf{m}_{n+k}}$ (le message $\mathbf{m}_k$ n'est pas le message de priorité k, c'est simplement le $k^{\text{ième}}$ a être programmé dans l'intervalle considéré d'où cette nouvelle notation). La faisabilité du système 1 qui est une hypothèse nous dit que ${}^1i_j \leq \mathbf{R}_{\mathbf{m}_j}$. Considérons maintenant le système 2 qui a un message pending supplémentaire; le message p, pour lequel il faut aussi respecter ${}^2i_p \leq \mathbf{R}_{\mathbf{m}_p}$.
Faisons maintenant l'hypothèse que $\mathbf{R}_{\mathbf{m}_h} \leq \mathbf{R}_p \leq \mathbf{R}_{\mathbf{m}_{h+1}}$ pour $h < k$[37].

- Pour $1 \leq j \leq h$, la $j^{\text{ième}}$ sélection est alloué au message $\mathbf{m}_j$: ${}^2a_{i_j} = \mathbf{m}_j$. La faisabilité du message $\mathbf{m}_j$ est satisfaite car avec l'équation (E.3), on a : ${}^2i_j \leq {}^1i_j \leq D_{\mathbf{m}_j}$.
- Pour le slot $h+1$, l'allocation EDF sélectionne le message p. La faisabilité vis-à-vis du message p est garantie car ${}^2i_{h+1} \leq {}^2i_{k+1} \leq \mathbf{R}_p$ par la définition de h.
- Pour $h+2 \leq j \leq k+1$, EDF sélectionne le message $\mathbf{m}_{j-1}$. Là encore la faisabilité vis-à-vis du message $\mathbf{m}_{j-1}$ est assurée car ${}^2i_j \leq {}^2i_{k+1} \leq \mathbf{R}_p \leq \mathbf{R}_{\mathbf{m}_{j-1}}$.
- Finalement, pour $k+2 \leq j \leq k+n$, le message $\mathbf{m}_{j-1}$ est choisi et la faisabilité de $\mathbf{m}_{j-1}$ est toujours satisfaite car par l'équation (E.4), on a ${}^2i_j = {}^1i_{j-1} \leq \mathbf{R}_{\mathbf{m}_{j-1}}$.

Dans tous ces cas, le système 2 alloue les même messages que le système 1 plus le message p, tous à temps, pendant l'intervalle $[0,T_p[$.
Faisons maintenant l'hypothèse que les équations (E.5) et (E.6) sont vraies jusqu'au slot bT_p, alors ${}^2P_{bT_p} = {}^1P_{bT_p} \cup \{p\}$, et la propriété de faisabilité est vérifiée jusque bT_p. En reproduisant le raisonnement utilisé sur le premier intervalle, les messages alloués dans $[bT_p,(b+1)T_p[$ pour le système complet (système 2) sont les mêmes que dans le système 1, plus le message p, tous étant alloués à temps. Ainsi, le message p satisfait son échéance dans $[bT_p,(b+1)T_p[$ (équation (E.6) l'ensemble des messages pendings au slot $(b+1)T_p$ satisfait l'équation (E.5)).
□

37. Si $\mathbf{R}_p$ est plus petit que $\mathbf{R}_1$ ou si $\mathbf{R}_p$ est plus grand que $\mathbf{R}_k$ alors h n'existe pas, ces cas peuvent cependant être considérés comme des cas particuliers des arguments qui suivent.

Table des figures

Liste des tableaux

Listes des abréviations utilisées

ABS: Anti-lock Braking System
AF : Architecture Fonctionnelle
AG : Algorithme Génétique
AGP : Algorithme Génétique Parallèle
AO : Architecture Opérationnelle
AM : Architecture Matérielle
BS : Background Scheduling
BV : Boite de Vitesse
BVA : Boite de Vitesse Automatique
CAN : Controller Area Network
CM : Contrôle Moteur
CRC : Cyclic Redundancy Code
CSMA/AMP: Carrier Sense Multiple Access/ Arbitration on Message Priority
CSMA/CA : Carrier Sense Multiple Access/ Collision Avoidance
CSMA/CR : Carrier Sense Multiple Access/ Collision Resolution
DLC : Data Length Code
DM : Deadline Monotonic
DO : Délai Observé
DP : Dual-Priority
DR : Délai Réel
ECU : Electronic Control Unit
EDF : Earliest Deadline First
EMI : Electromagnetic Interference
F-GA : Full Genetic Algorithm
FIFO : First-In First-out
FPP : Fixed Priority Preemptive
HRT : Hard Real-Time
ISR : Interrupt Service Routine
LdD : Liaison de Données
MWAF : Majorizing Work Arrival function
non-TR : non-Temps Réel
PCP : Priority Ceiling Protocol
PCP-RR :Priority Ceiling Protocol pour Round-Robin
QOS : Quality of Service
REC : Receive Error Counter
RM : Rate Monotonic
RR : Round-Robin
SAE : Society of Automotive Engineers
SdF : Sûreté de Fonctionnement
SRT : Soft Real-Time
TeB : Taux d'erreur Bit
TEC : Transmit Error Counter
TeT : Taux d'erreur Trame
TR : Temps Réel
VAN : Vehicle Area Network
WAF : Work Arrival function
W-GA : Weak Genetic Algorithm

Bibliographie

[Abramowitz and Stegun, 1970] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions*. Dover Publications (ISBN 0-486-61272-4), 1970.

[Allen-Bradley, 1994] Allen-Bradley. Devicenet specification, 1994. vol. 1 & 2.

[André, 1997] O. André. Généralisation d'un observateur d'événements sur le réseau CAN, Novembre 1997. Stage de fin d'études.

[Association Française de Normalisation-AFNOR, 1990] Association Française de Normalisation-AFNOR. Véhicules routiers - transmission de données, Décembre 1990. R13-708.

[Audsley *et al.*, 1993] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. *Software Engineering Journal*, 8(5):284–292, Septembre 1993.

[Audsley, 1991a] N.C. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary deadlines. Technical Report YCS 164, University of York, November 1991.

[Audsley, 1991b] N.C. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Technical Report YCS164, University of York, Novembre 1991.

[Barrenscheen and Otte, 1997] J. Barrenscheen and G. Otte. Analysis of the physical CAN bus layer. In 4^{th} *international CAN Conference, ICC'97*, pages 06.02–06.08, Octobre 1997.

[Bate and Burns, 1998] I. Bate and A. Burns. Investigation of the pessimism in distributed systems timing analysis. In *Proceedings of the 10th Euromicro Workshop on Real-Time System*, pages 107–114, Juin 1998.

[Bayart and Simonot-Lion, 1995] M. Bayart and F. Simonot-Lion. Impact de l'émergence des réseaux de terrain et de l'instrumentation intelligente dans la conception des architectures des systèmes d'automatisation de processus. Contrat MESR - 92-p-239, Centre de Recherche en Informatique de Nancy, Vandoeuvre-lès-Nancy, Février 1995.

[Beaty *et al.*, 1990] S.J. Beaty, J. Whitley, and G. Johnson. Motivation and framework for

using genetic algorithms for microcode compaction. In *Proceedings of the 23rd Annual Workshop in Microprogramming and Microarchitecture (MICRO-23)*, Décembre 1990.

[Beaty *et al.*, 1996] S.J. Beaty, S. Colcord, and P. Sweany. Using genetic algorithms to fine-tune instruction-scheduling heuristics. In *Second International Conference on Massively Parallel Computing Systems (MPCS'96)*, Mai 1996.

[Beaty, 1993] S.J. Beaty. Genetic algorithm versus tabu search for instruction scheduling. In *International Conference on Neural Networks and Genetic Algorithms*, Avril 1993.

[Bélissent, 1996] P. Bélissent. Validation des systèmes temps réel distribués autour de CAN. Technical report, Centre de Recherche en Informatique de Nancy (CRIN), Septembre 1996. Mémoire de DEA, UHP Nancy I.

[Bernat and Burns, 1997] G. Bernat and A. Burns. Combining (n, m)-hard deadlines and dual priority scheduling. In *Proceedings of the 18th IEEE Real-Time Systems Symposium*, December 1997.

[Bernat, 1998] G. Bernat. *Specification and Analysis of Weakly Hard Real-Time Systems.* Phd thesis, Universitat de les Illes Balears, Departament de Ciències Matemàtiques i Informàtica, January 1998.

[Bhat, 1984] U.N. Bhat. *Elements of Applied Stochastic Processes.* John Wiley & Sons, 1984. ISBN 0-471-87826-X.

[Burns and Wellings, 1996] A. Burns and A.J. Wellings. Dual priority scheduling in ADA95 and real-time Posix. In *Proceedings of the 21st IFAC/IFIP Workshop on Real-Time Programming*, pages 45–50, 1996.

[Burns *et al.*, 95] A. Burns, K. Tindell, and A. Wellings. Effective analysis for engeneering real-time fixed priority schedulers. *IEEE Transactions on Software Engineering*, 21(5):475–480, May 95.

[Butenhof, 1997] D.R. Butenhof. *Programming with POSIX Threads.* Addison-Wesley, 1997. ISBN 0-201-63392-2.

[Cabus, 1999] G. Cabus. Etude de l'ordonnancement conjoint de tâches et de messages - application à OSEK/VDX, Septembre 1999. Mémoire de DEA, INPL.

[Cantù-Paz, 1995] E. Cantù-Paz. A summary of research on parallel genetic algorithms. Technical Report IlliGAL Report No. 95007, University of Illinois at Urbana-Champain, 1995.

[CENELEC, 1997] European Committee for Electrotechnical Standardization CENELEC. Low voltage switchgear and controlgear - part 5: Control circuit devices and switching elements - smart distributed systems (SDS), 1997. document CLC/TC(SEC)146 Smart Distributed Systems.

[Chen *et al.*, 1998] H. Chen, N.S. Flann, and D. Watson. Parallel genetic simulated annealing : A massively parallel SIMD algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 9(2):126–136, Février 1998.

[Chetto and Chetto, 1989] H. Chetto and M. Chetto. Some Results of the Earliest Deadline Scheduling Algorithm. *IEEE Transactions on Software Engineering*, 15(10):1261–1269, October 1989.

[(CiA), 1995] CAN in Automation International Users and Manufacturers Group (CiA). CAN application layer (CAL), 1995. Cia/DS201-207.

[(CiA), 1996] CAN in Automation International Users and Manufacturers Group (CiA). CANopen communication profile for industrial systems, 1996. CiA/DS301 (Version 3.0).

[Coujoulou and Vuillaume, 1997] P. Coujoulou and M. Vuillaume. Analyse des contraintes de temps réel sur un réseau CAN, 1997. Mémoire de DESS Informatique, Option IRS, UHP Nancy I.

[Courrier *et al.*, 1998a] M. Courrier, F. Simonot-Lion, and Y.Q. Song. Microscopic modeling of support system for in-vehicle embedded systems. In *International IFIP Workshop on Distributed and Parallel Embedded Systems, DIPES'98*, 1998.

[Courrier *et al.*, 1998b] M. Courrier, S. Wolf, F. Simonot-Lion, and Y.-Q. Song. Modélisation de composants matériels et exécutifs en vue de la validation d'architecture opérationelle par évaluation de performances. Technical report, Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA), 1998. Rapport intermédiaire du contrat PSA 033, Rapport 98-R-056.

[Davis and Wellings, 1995a] R. Davis and A.J. Wellings. Dual priority scheduling. In *Proceedings of the 16th IEEE Real-Time Systems Symposium*, pages 100–109, December 1995.

[Davis and Wellings, 1995b] R. Davis and A.J. Wellings. Dual priority scheduling. In *Real-Time Systems Symposium*, pages 100–109, 1995.

[Davis *et al.*, 1993] R. Davis, K. Tindell, and A. Burns. Scheduling slack time in fixed priority pre-emptive systems. In *Proceedings of the 14th IEEE Real-Time Systems Symposium*, pages 222–231, December 1993.

[Davis, 1985] L. Davis. Job-shop scheduling with genetic algorithms. In *Proceedings of the First Int. Conf. on Genetic Algorithms*, pages 136–140, 1985.

[Davis, 1991] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New-York, 1991.

[Davis, 1994] R. Davis. Dual priority scheduling: A means of providing flexibility in hard

real-time systems systems. Technical report, Department of Computer Science, University of York (UK), Mai 1994. Technical Report YCS230.

[DiNatale and Stankovic, 1995] M. DiNatale and J.A. Stankovic. Applicability of simulated annealing methods to real-time scheduling and jitter control. In *Proceedings of the 16th IEEE Real-Time Systems Symposium*, Décembre 1995. also available at http://retis.sssup.it/papers/abstracts.html.

[Djerid *et al.*, 1995] L. Djerid, M.-C. Portmann, and P. Villon. Performance analysis of previous and new proposed cross-over genetic operators designed for permutation scheduling problems. In *Proceedings International Conference on Industrial Engineering and Production Management*, pages 487–497, Marrakech (Maroc), Avril 1995.

[Dussa-Zieger and Schwehm, 1998] K. Dussa-Zieger and M. Schwehm. Scheduling of parallel programs on configurable multiprocessors by genetic algorithms. *International Jounal of Approximate Reasoning*, 19(1-2):23–38, Juillet 1998.

[Dussa-Zieger, 1996] K. Dussa-Zieger. Task scheduling on configurable parallel systems by genetic algorithms. In *Proceedings of Telecommunication, Distribution, Parallelism (TDP'96)*, pages 485–494, Juillet 1996.

[Falkenauer and Bouffoix, 1991] E. Falkenauer and S. Bouffoix. A genetic algorithm for job shop. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, pages 824–829, 1991.

[Gallmeister, 1995] B.O. Gallmeister. *Programming for the Real World - Posix 4*. O'Reilly & Associates, 1995. ISBN 1-56592-074-0.

[Gaujal and Navet, 1999a] B. Gaujal and Navet. Traffic shaping in real-time distributed systems: a low-complexity approach. Rapport de recherche RR-3719, INRIA, Juin 1999.

[Gaujal and Navet, 1999b] B. Gaujal and N. Navet. Traffic shaping in real-time distributed systems: a low-complexity approach. *Computer Communications*, 22(17):1562–1573, 1999.

[Gaujal *et al.*, 1998] B. Gaujal, E. Altman, and A Hordijk. Optimal open-loop control of vacations. Rapport de recherche RR-3261, INRIA, 1998.

[Gaujal *et al.*, 1999a] B. Gaujal, N. Navet, and J. Migge. Dual-priority versus background scheduling: a path-wise comparison. Rapport de recherche RR-3734, INRIA, Juillet 1999.

[Gaujal *et al.*, 1999b] B. Gaujal, N. Navet, and J. Migge. Dual-priority versus background scheduling: a path-wise comparison. *En cours d'évaluation pour Real-Time Systems*, 1999.

[George *et al.*, 1996] L. George, N. Rivierre, and M. Spuri. Preemptive and non-preemptive real-time uniprocessor scheduling. Rapport de recherche RR-2966, INRIA, Septembre 1996.

[Gerber and Hong, 1995] R. Gerber and S. Hong. Slicing real-time programs for enhanced

schedulability. Technical Report UMD Technical Report CS-TR-3477, UMIACS TR 95-62, University of Maryland, Mai 1995.

[Gerber and Hong, 1997] R. Gerber and S. Hong. Slicing real-time programs for enhanced schedulability. *ACM Transactions on Programming Languages and Systems*, 19(3), Mai 1997.

[Gerber, 1994] R. Gerber. Languages and tools for real-time systems: Problems, solutions and opportunities. Technical Report UMD Technical Report CS-TR-3362, UMIACS-TR-94-117, University of Maryland, Octobre 1994.

[Goldberg and Lingle, 1985] D.E. Goldberg and R. Lingle. Alleles, loci, and the travelling salesman problem. In *Proceedings of the First Int. Conf. on Genetic Algorithms*, pages 154–159, 1985.

[Goldberg, 1989] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Welsey, 1989.

[Gomez *et al.*, 1995] C. Gomez, B. Salvy, and P. Zimmermann. *Calcul Formel: Mode d'Emploi - Exemples en Maple.* Masson, 1995. ISBN 2-225-84780-0.

[Hajek, 1985] B. Hajek. Extremal splittings of points process. *Mathematics of Operation Research*, 10(4):543–556, 1985.

[Hank, 1997] P. Hank. Pelican: A new can controller supporting diagnosis and system optimization. In 4^{th} *international CAN Conference, ICC'97*, pages 04.12–04.18, Octobre 1997.

[Hausmann and Gebing, 1997] G. Hausmann and E. Gebing. The realisation of specific automotive applications with "full" CAN functionality at "basic" CAN cost on highly integrated 8-bit microcontroller of NEC's 78k/0 family. In 4^{th} *international CAN Conference, ICC'97*, pages 4.02–4.11, 1997.

[Holland, 1975] J.A. Holland. *Adaptation in Natural and Artificial Systems.* Mit Press, Cambridge, Mass., 1975.

[IEC TC17B - WG3, 1998a] IEC TC17B - WG3. Devicenet specification, 1998. Draft 62026-3.

[IEC TC17B - WG3, 1998b] IEC TC17B - WG3. SDS specification, 1998. Draft 62026-5.

[Ingber and Rosen, 1992] L. Ingber and B. Rosen. Genetic algorithm and very fast simulated reannealing: A comparison. *Mathematical Computer Modeling*, 16(11):87–100, 1992.

[Intel Corporation, 1999] Intel Corporation. Introduction to in-vehicle networking, 1999. available at http://www.intel.com/design/auto/autolxbk.htm.

[ISO, 1994a] International Standard Organization ISO. *Road Vehicles - Interchange of Digital Information - Controller Area Network for high-speed Communication.* ISO, 1994. ISO 11898.

[ISO, 1994b] International Standard Organization ISO. *Road Vehicles - Low Speed serial data communication - Part 2: Low Speed Controller Area Network.* ISO, 1994. ISO 11519-2.

[(ISO/IEC), 1996] (ISO/IEC). *9945-1:1996 (ISO/IEC)[IEEE/ANSI Std 1003.1 1996 Edition] Information Technology - Portable Operating System Interface (POSIX) - Part 1 : System Application : Program Interface.* IEEE Standards Press, 1996. ISBN 1-55937-573-6.

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

[Juanole and Blum, 1999] G. Juanole and I. Blum. Influence de fonctions de base (communication-ordonnancement) des systèmes distribués temps-réel sur les performances d'applications de contrôle-commande. In *Colloque Francophone sur L'ingénierie des Protocoles, CFIP'99*, pages 217–231, Avril 1999.

[Kidwell, 1993] M.D. Kidwell. Using genetic algorithms to schedule distributed tasks on a bus-based system. In *Proceedings of Fifth International Conference on Genetic Algorithms*, pages 368–374, 1993.

[Kiencke and Kytölä, 1996] U. Kiencke and T. Kytölä. CAN, a ten years' anniversarial review. In *3th international CAN Conference, ICC'96*, pages 2.2–2.7, Octobre 1996.

[Kiencke *et al.*, 1997] U. Kiencke, J. Dirk, and S. Schneider. Performance analysis of a distributed automotive real-time system. In 4^{th} *international CAN Conference, ICC'97*, pages 7.02–7.08, 1997.

[Klein *et al.*, 1993] M.H. Klein, T. Ralya, B. Pollak, R. Obenza, and Harbour M.G. *A Practitioner's Handbook for Real-Time Analysis.* Kluwer Academic Publishers, 1993.

[Koopman *et al.*, 1997] P. Koopman, J. Sung, C. Dingman, and D. Siewiorek. Comparing operating systems using robustness benchmarks. In *Symposium on Reliable Distributed Systems*, pages 72–79, Octobre 1997.

[Kopetz *et al.*, 1995] K Kopetz, R. Nossal, R. Hexel, A Krueger, D. Millinger, R. Pallierer, C. Temple, and M. Krug. Mode handling in the time-triggered architecture. *Control Eng. Practice*, 3(8):1163–1169, 1995.

[Kopetz, 1997] H. Kopetz. *Real-Time Sytems : Design Principles for Distributed Embedded Applications.* Kluwer Academic Publishers, Boston, 1997.

[Kwok and Ahmad, 1997] Y.K. Kwok and I. Ahmad. Efficient scheduling of arbitrary task graphs to multiprocessors using a parallel genetic algorithm. *Journal of Parallel and Distributed Computing - Special Issue on Parallel Evolutionary Computing*, 47(1):58–77, Novembre 1997.

[Lawrenz, 1995] W. Lawrenz. Worldwide status of CAN - present and future. In *2nd international CAN Conference, ICC'95*, pages 0.12–0.25, Septembre 1995.

[Lawrenz, 1997] W. Lawrenz. *CAN System Engineering - From Theory to Applications.* Springer-Verlag, 1997. ISBN 0-387-94939-9.

[Lecuivre, 1996] Jérôme Lecuivre. *Evaluation d'Architectures Temps Réel Réparties : Application à CCE.* Thèse d'université, INPL, 1996.

[Lefebvre *et al.*, 1995] M. Lefebvre, E. Gressier, and S. Natkin. MMS sur TCP/IP: une nouvelle solution pour l'échange de données en informatique de production. In *Real-Time Systems, RTS'98*, pages 90–108, Janvier 1995.

[Lehoczky and Ramos-Thuel, 1992] J. Lehoczky and S. Ramos-Thuel. Aperiodic Task Scheduling for Hard-Real-Time Systems. In *Proceedings IEEE Real-Time Systems*, pages 110–123, December 1992.

[Lehoczky *et al.*, 1987] J. Lehoczky, L. Sha, and Y. Ding. Enhancing aperiodic responsiveness in hard real-time environment. In *Proceedings IEEE Real-Time Systems*, pages 261–270, December 1987.

[Lin *et al.*, 1990] K. Lin, K. Kenny, S. Natarayan, and J. Liu. Flex: A language for real-time systems programming. In A. Tilborg and G. Koob, editors, *Foundations of Real-Time Computing: Formal Specifications and Methods*, pages 251–290. Kluwer Academic Publishers, 1990.

[Liu and Layland, 73] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in hard real-time environment. *Journal of the ACM*, 20(1):40–61, February 73.

[Liu *et al.*, 1999] J.W. Liu, R. Rakjumar, Z. Deng, M. Seri, A. Frei, L. Zhang, and C.S. Shih. RFC: How to get the most predictability out of Windows NT, 1999. IEEE-CS TC-RTS Newsletter for Thu. Aug. 05, 1999.

[Macos and Mueller, 1998] D. Macos and F. Mueller. Integrating gnat/gcc into a timing analysis environment. In *10th EUROMICRO Workshop on Real Time Systems*, 1998. WIP Session.

[Mandelbrot, 1996] B. Mandelbrot. Du hasard bénin au hasard sauvage. *Pour la Science*, pages 12–17, Avril 1996. dossier hors-série.

[Martineau, 1994] P. Martineau. *Ordonnancement en-ligne dans les systèmes informatiques temps-réel.* Thèse de doctorat, Ecole Centrale de Nantes, Octobre 1994.

[Michalewicz, 1992] J.A. Michalewicz. *Genetic Algorithm + Data Structure = Evolution Program.* Springer Verlag, 1992.

[Migge and Jean-Marie, 1998] J.M. Migge and A. Jean-Marie. Timing analysis of real-time

scheduling policies: A trajectory based model. Research Report 3561, INRIA, November 1998.

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

[Migge *et al.*, 1999] J.M. Migge, A. Jean-Marie, and N. Navet. Timing analysis of compound scheduling policies: Fixed piorities and round robin. Submitted to Journal of Scheduling, 1999.

[Migge, 1999] J.M. Migge. *Scheduling of recurrent tasks on one processor: A trajectory based Model.* PhD thesis, Université Nice Sophia-Antipolis, 1999.

[Minsky, 1965] M.L. Minsky. Matters, minds and models. In *Proc. International Federation of Information Processing Congress*, pages 45–49, 1965.

[Mok and Chen, 1997] A. Mok and D. Chen. A multiframe model for real-time tasks. *IEEE Transactions on Software Engineering*, 23(10):635–645, october 1997.

[Moses, 1995] J. Moses. Is posix appropriate for embedded systems? *Embedded Systems Programming*, July 1995.

[Motorola LTD, 1997] Motorola LTD. Bosch controller area network (CAN) version 2.0 - annexe b : Toucan, 1997. Rev. 1.

[Nachef, 1992] A. Nachef. Modélisation des systèmes distribués. *RAIRO, Techniques et Sciences Informatiques*, 12(2):163–192, 1992.

[Nakano and Yamada, 1991] R. Nakano and T. Yamada. Conventional genetic algorithm for job shop problems. In *Proceedings of the Fourth Int. Conf. on Genetic Algorithms*, pages 474–479, 1991.

[Navet and Migge, 1999a] N. Navet and J. Migge. Fine tuning the scheduling of tasks through a genetic algorithm : Application to Posix1003.1b compliant systems. Rapport de recherche RR-3730, INRIA, Juillet 1999.

[Navet and Migge, 1999b] N. Navet and J. Migge. Fine tuning the scheduling of tasks through a genetic algorithm: Application to Posix1003.1b compliant uniprocessor systems. *Version révisée en cours d'évaluation pour IEE Proceedings - Software*, 1999.

[Navet and Song, 1996] N. Navet and Y.-Q. Song. Evaluation de performances de la messagerie CAN du véhicule prototype PSA - action 1 du contrat PSA-CRIN. Technical report, Centre de Recherche en Informatique de Nancy (CRIN), 1996. Rapport de fin de contrat 96-R-182.

[Navet and Song, 1997a] N. Navet and Y.-Q. Song. Animation de deux ateliers de travail sur

la validation d'applications réparties autour du réseau CAN, Septembre 1997. École d'été Temps Réel.

[Navet and Song, 1997b] N. Navet and Y.-Q. Song. CAN modeling : towards integrating analytic methods and simulation. In *OPnet European Users Group (OPEUG), Paris, France*, Octobre 1997.

[Navet and Song, 1998a] N. Navet and Y.-Q. Song. Design of reliable real-time applications distributed over CAN (controller area network). In *INCOM98, IFAC 9th Symposium on Information Control in Manufacturing*, Juin 1998.

[Navet and Song, 1998b] N. Navet and Y.-Q. Song. On fault tolerance and worst-case response time analysis in CAN. In *23nd IFAC/IFIP Workshop on Real-Time Programming, WRTP'98*, Juin 1998.

[Navet and Song, 1999a] N. Navet and Y.-Q. Song. *Les Communications Industrielles*, chapter Le Réseau CAN dans les Systèmes Temps Réel Embarqués. À paraître sous la direction de Francis Lepage, éditions Hermès, 1999.

[Navet and Song, 1999b] N. Navet and Y.-Q. Song. Reliability improvement of the dual-priority protocol under unreliable transmission. *Control Engineering Practice*, 7(8):975–981, 1999.

[Navet and Song, 1999c] N. Navet and Y.-Q. Song. Une politique à changement de priorité pour l'ordonnancement de messages dans des environnements bruités. In *Colloque Francophone sur L'ingénierie des Protocoles, CFIP'99*, pages 249–264, Avril 1999.

[Navet *et al.*, 1995] N. Navet, J. Lecuivre, and Y.-Q. Song. A Simulation Framework for Performance Evaluation of Communication Profiles: Application to the MMS/TCP/Ethernet Profile. In M. D. Cin, U. Herzog, G. Bolch, and A. R. Kaylan, editors, *Proceedings 7th European Simulation Symposium, Erlangen-Nuremberg (Germany)*, pages 757–761, Octobre 1995.

[Navet *et al.*, 1996] N. Navet, P. Bélissent, and Y.-Q. Song. VACANS : Un outil d'aide à la validation d'applications distribuées autour de CAN. In *6ème Atelier d'Evalution de Performances*, Novembre 1996.

[Navet *et al.*, 1998a] N. Navet, Y.-Q. Song, and F. Simonot. Performances et tolérance aux fautes des applications temps réel distribuées autour du réseau CAN. In *Real-Time Systems, RTS'98*, pages 197–216. Teknea, Janvier 1998.

[Navet *et al.*, 1998b] N. Navet, Y.-Q. Song, and J.-P. Thomesse. Le Réseau CAN et les Erreurs de Transmission. *CiMax Terrain*, 16:15–18, Mars 1998.

[Navet *et al.*, 1999] N. Navet, Y.-Q. Song, and F. Simonot. Worst-case deadline failure proba-

bility in real-time applications distributed over CAN (controller area network). *à paraître dans Journal of Systems Architecture*, 1999.

[Navet, 1995] N. Navet. Modélisation, simulation et evaluation de profils de communication, Septembre 1995. Mémoire de DEA, UHP Nancy I.

[Navet, 1996] N. Navet. Le Réseau CAN et sa modélisation sous forme de réseaux de files d'attente. In *Automatique, Génie Informatique, Image - AGI'96, Tours, France*, pages 293–297, Juin 1996.

[Navet, 1998] N. Navet. Controller Area Network: CANs use within automobiles. *IEEE Potentials*, 17(4):12–14, Octobre 1998.

[Noble, 1992] I.E. Noble. EMC and the automotive industry. *Electronics & Communication Engineering Journal*, pages 263–271, Octobre 1992.

[Oliver *et al.*, 1985] I.M. Oliver, G.J. Smith, and J.R.C. Holland. A study of permutation crossover operators on the travelling salmesman problem. In *Proceedings of the Second Int. Conf. on Genetic Algorithms*, pages 224–230, 1985.

[OSEK Group, 1997] OSEK Group. OSEK/VDX operating system, Octobre 1997. Version 2.0 rev. 1, disponible à l'adresse http://www-iiit.etec.uni-karlsruhe.de/~ osek/.

[Paret, 1996] D. Paret. *Le Bus CAN (Controller Area Network)*. Dunod éditeur, 1996. ISBN 2-10-003164-3.

[Paret, 1999] D. Paret. *Le Bus CAN: Applications*. Dunod éditeur, 1999. ISBN 2-10-003659-9.

[Parzen, 1962] E. Parzen. *Stochastic Processes*. Holden-Day (ISBN 0-8162-6664-6), 1962.

[Philips Semiconductors, 1997] Philips Semiconductors. SJA 1000 stand-alone CAN controller data sheet, Novembre 1997. Preliminary Specification.

[Popper, 1973] J. Popper. *La dynamique des systèmes, principes et applications*. Éditions d'Organisation, 1973.

[Portmann, 1996] M.-C. Portmann. Scheduling methodology: Optimization and compu-search approaches i. In A. Artiba and S. E. Elmahgraby, editors, *Production and Scheduling of Manufacturing System*, pages 271–300. Chapman & Hall. 1997, 1996.

[PZ Marketing, 1997] PZ Marketing. *CAN Newsletter*. PZ Marketing, Juin 1997. J25361F.

[QNX Software Systems, 1997] QNX Software Systems. QNX operating system, system architecture, 1997. QNX 4.24.

[QNX Software Systems, 1999] QNX Software Systems. Neutrino system architecture guide - chapter 2: The neutrino microkernel (part 2), 1999. available at http://www.qnx.com/literature/nto_sysarch/kernel2.html#SCHEDULING.

[Real-Time Magazine, 1997] Real-Time Magazine. RTOS buyer's guide, 1997. 97-3.

[Riera, 1997] F. Riera. *Monstre: A Microkernel orientat a les necessitats de sistemes de temps real empotrats.* Final year project, Universitat de les Illes Balears, Departament de Ciències Matemàtiques i Informàtica, 1997.

[Rüdiger, 1998a] R. Rüdiger. Evaluating the temporal behaviour of CAN-based systems by means of a cost functional. In *5th international CAN Conference, ICC'98*, Novembre 1998. Also available at http://www.fh-wolfenbuettel.de/fb/i/organisation/personal/ruediger/forschung/research.htm.

[Rüdiger, 1998b] R. Rüdiger. Prioritätswartesysteme für dir modellbildung von CAN-systemen. Technical report, Fachbereich Mathematik und Technik, FH Bielefeld, 1998. Available at http://www.fh-wolfenbuettel.de/fb/i/organisation/personal/ruediger/forschung/research.htm.

[Saad-Bouzefrane and Cottet, 1997] S. Saad-Bouzefrane and F. Cottet. A performance analysis of distributed hard real-time applications. In *IEEE International Workshop on Factory Communication Systems, WFCS'97*, pages 167–176, Octobre 1997.

[Saad-Bouzefrane, 1997] S. Saad-Bouzefrane. *Étude temporelle des applications temps réel distribuées à contraintes strictes basée sur une analyse d'ordonnançabilité.* Thèse d'université, Université de Poitiers, 1997.

[SAE, 1993] Society of Automotive Engineers SAE. Class C application requirement considerations. Technical report, Society of Automotive Engineers, Juin 1993. J2056/1.

[SAE, 1996] Society of Automotive Engineers SAE. Class B data communications network interface - sae j1850 standard, 1996. Rev. NOV96.

[Salles *et al.*, 1997] F. Salles, J. Arlat, and J.-C. Fabre. Can we rely on COTS microkernels for building fault-tolerant systems. In *Proceedings of the 6th IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pages 189–194, Octobre 1997.

[Schwehm and Walter, 1994] M. Schwehm and T. Walter. Mapping and scheduling by genetic algorithms. In *Third Joint International Conference on Vector and Parallel Processing (CONPAR94), Linz (Austria)*, number 854 in Lecture Notes in Computer Science, pages 833–841. Springer Verlag, 1994.

[Sha *et al.*, 1990] L. Sha, R. Rajkumar, and J.P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, September 1990.

[Simonot-Lion *et al.*, 1997] F. Simonot-Lion, Y.-Q. Song, and J. Raymond. Validating real-

time applications distributed over CAN : an interoperability verification. In 4^{th} *international CAN Conference, ICC'97*, pages 07.09–07.18, octobre 1997.

[Softing GmbH, 1995] Softing GmbH. Cancard user manual (v2.03, rev.01), 1995.

[Song and Navet, 1997] Y.-Q. Song and N. Navet. Validation d'applications distribuées autour du réseau can par vérification en-ligne - développement d'un observateur réseau. Technical report, Centre de Recherche en Informatique de Nancy (CRIN), 1997. Rapport de fin de contrat 97-R-110.

[Song *et al.*, 1996] Y.-Q. Song, F. Simonot-Lion, and N. Navet. Validation of distributed real-time systems thanks to performance evaluation of their physical architecture. In *CESA'96, IMACS Multiconference/IEEE SMC, Symposium on Discrete Events and Manufacturing systems / Computational Engineering in Systems Applications*, pages 507–512, Juillet 1996.

[Song *et al.*, 1999] Y.-Q. Song, F. Simonot-Lion, and N. Navet. De l'évaluation de performances à la validation de l'architecture opérationnelle - cas du système embarqué dans l'automobile. In *École d'été Temps Réel*, pages 213–227, Septembre 1999.

[Sprunt *et al.*, 1989] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic task scheduling for hard-real-time systems. *Real-Time Systems*, 1:27–60, 1989.

[Stankovic *et al.*, 1996] J.A. Stankovic, A. Burns, K. Jeffay, M. Jones, G. Koob, I. Lee, J. Lehoczky, J. Liu, A. Mok, K. Ramamritham, J. Ready, L. Sha, and A. Van Tilborg. Strategic directions in real-time and embedded systems. *ACM Computing Surveys*, 28(4):751–763, Décembre 1996.

[Stankovic, 1988] J.A. Stankovic. Misconceptions about real-time computing. *IEEE Computer*, 21(10):10–19, Octobre 1988.

[Sun *et al.*, 1997] Jun Sun, Mark K. Gardner, and Jane W. S. Liu. Bounding completion times of jobs with arbitrary release times, variable execution times, and resource sharing. *IEEE Transactions on Software Engineering*, 23(10):603–615, October 1997.

[Tang *et al.*, 1997] K.H. Tang, R. McLaughlin, J. Moyne, and J. Shah. DeviceNet modeling on a DeviceNet communications network. In 4^{th} *international CAN Conference, ICC'97*, pages 10.12–10.21, Octobre 1997.

[The Risks Digest, 1997a] The Risks Digest. GM car acceleration due to EMI. *http:// catless.ncl.ac.uk/ Risks/ 19.38.html*, 19(38), Septembre 1997.

[The Risks Digest, 1997b] The Risks Digest. Mad bus disease. *http:// catless.ncl.ac.uk/ Risks/ 19.40.html*, 19(40), Octobre 1997.

[Thiel, 1998] D. Thiel. Un algorithme génétique pour la résolution de problèmes d'affecta-

tion quadratique comparé aux performances du recuit simulé. *RAIRO - APII - JESA*, 31(9):1541–1564, 1998.

[Thomesse *et al.*, 1991] J.-P. Thomesse, P. Lorenz, J.-P. Bardinet, P. Leterrier, and T. Valentin. Factory instrumentation protocol : model, products and tools. *Control Engineering Practice*, 38(12):65–67, 1991.

[Tindell and Burns, 1994a] K. Tindell and A. Burns. Guaranteed message latencies for distributed safety-critical hard real-time control networks. Technical report, Department of Computer Science, University of York (UK), Mai 1994. Technical Report YCS229.

[Tindell and Burns, 1994b] K. Tindell and A. Burns. Guaranteeing message latencies on controller area network (CAN). In *1st International CAN Conference, ICC'94*, 1994.

[Tindell and Clark, 1994] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessors and Microprogramming*, 40:117–134, 1994.

[Tindell and Hansson, 1995] K. Tindell and H. Hansson. Babbling idiots, the dual priority protocol, and smart CAN controllers. In *2nd international CAN Conference, ICC'95*, pages 7.22–7–28, Septembre 1995.

[Tindell *et al.*, 1992] K. Tindell, A Burns, and A. Wellings. Mode changes in priority preemptively scheduled systems. In *Proceedings IEEE Real-Time Systems Symposium*, December 1992.

[Tindell *et al.*, 1994] K. Tindell, A. Burns, and A.J. Wellings. An extendible approach for analysing fixed priority hard real time sytems. *Real-Time Systems*, 6(2), 1994.

[Tindell *et al.*, 1995] K. Tindell, Burns, and A.J. Wellings. Calculating controller area network (CAN) message response times. *Control Engineering Practice*, 3(8):1163–1169, 1995.

[Tindell, 1992] K.W. Tindell. An extendible approach for analysing hard real time tasks. Technical Report YCS 189, University of York, 1992.

[Tindell, 1993] K.W. Tindell. *Fixed Priority Scheduling of Hard Real-Time Systems.* PhD thesis, University of York, December 1993.

[Tindell, 94] K.W. Tindell. Adding time-offsets to schedulability analysis. Technical Report YCS221, Department of Computer Science, University of York, 94.

[Toussaint, 1997] Joël Toussaint. *Modélisation d'applications temps réel réparties pour la validation de propriétés temporelles Méthodologie de construction de modèles et algorithmes de validation.* Thèse d'université, INPL, 1997.

[Tsuchiya *et al.*, 1998] T. Tsuchiya, T. Osada, and T. Kikuno. Genetics-based multiprocessor scheduling using task duplication. *Microprocessors and Microsystems*, 22(3-4):197–207, 1998.

[Unruh *et al.*, 1989] J. Unruh, H.-J. Mathony, and K.-H. Kaiser. Error detection analysis of automotive communication protocols. Technical report, Robert Bosch GmbH, 1989.

[Vega, 1996] Luis Vega. *Modèles de coopération et de communication entre processus temps réel répartis : expression de contraintes de temps pour la vérification de propriétés temporelles dans la communication.* Thèse d'université, INPL, 1996.

[Weseloh and Rüdiger, 1999] M. Weseloh and R. Rüdiger. Applying modern software design principles: A can tool based on extensibility. In *6^{th} international CAN Conference, ICC'99, to appear*, Novembre 1999. Also available at http://www.fh-wolfenbuettel.de/fb/i/organisation/personal/ruediger/forschung/research.htm.

[Wind Rivers Systems, 1997] Wind Rivers Systems. Vxworks programmer's guide, 5.3.1, 1997. Edition 1, DOC-12067-ZD-00.

[Wonnacott and Wonnacott, 1995] T.H. Wonnacott and R.J. Wonnacott. *Statistique.* Economica, 1995. 4ème édition.

[Zanoni and Pavan, 1993] E. Zanoni and P. Pavan. Improving the reliability and safety of automotive electronics. *IEEE Micro*, 13(1):30–48, 1993.

[Ziegler *et al.*, 1994] C. Ziegler, D. Powell, and P. Desroches. Dependability of on-board automotive computer systems. In *IEEE Intelligent Vehicles 1994 Symposium*, pages 568–575, Octobre 1994.

Résumé

Notre premier objectif est de proposer des méthodes et des outils de vérification du respect des contraintes temporelles d'une application temps réel. Le principal cadre d'application de nos travaux est celui des applications embarquées dans l'automobile distribuées autour d'un réseau CAN. La validation est menée en couplant les techniques de vérification : simulation, analyse et observation sur prototypes. L'apport principal de cette thèse réside en la conception de modèles analytiques qui fournissent des bornes sur les métriques de performance considérées (temps de réponse, probabilité de non-respect des échéances) ou permettant d'évaluer l'occurrence d'événements rares (temps d'atteinte de l'état bus-off d'une station CAN). Nous proposons également une analyse d'ordonnançabilité des applications s'exécutant sur des systèmes d'exploitation se conformant au standard Posix1003.1b . Ensuite, considérant qu'il existe généralement plusieurs solutions d'ordonnancement faisables à un même problème, nous avons défini des critères de choix et avons expérimenté une approche, utilisant un algorithme génétique, pour parcourir l'espace des solutions.

Notre second objectif est d'étudier des mécanismes d'ordonnancement qui garantissent le respect des échéances du trafic à contraintes strictes tout en minimisant les temps de réponse du trafic à contraintes souples. Nous évaluons les performances de la politique Dual-Priority pour l'ordonnancement de messages. Pour son utilisation dans des environnements bruités, nous proposons un mécanisme simple donnant des garanties sur la qualité de service exprimée en termes de probabilité de respect des échéances et s'adaptant en-ligne à des conditions de perturbations variables. Nous proposons également une politique concurrente, basée sur une technique de lissage de flux, qui est d'une mise en oeuvre plus aisée. Cette politique préserve la faisabilité du système et sa faible complexité algorithmique permet son utilisation en-ligne.

Mots-clés: Validation, Optimisation, Temps Réel, Ordonnancement, Systèmes Embarqués.

Abstract

In this thesis, we first investigate several approaches aimed at validating the temporal constraints of real-time systems. This work is applied to in-vehicle embedded applications distributed on a CAN network. The validation process is performed using different and complementary techniques : simulation, analysis and prototype monitoring. The main contribution is the design of analytical models that provide bounds on the considered performance metrics (response time, deadline failure probability) or that enable us to evaluate the occurrence of rare events (bus-off hitting time of a CAN node). We also propose a schedulability analysis of real-time applications running on Posix1003.1b compliant operating systems. Considering that usually there exist several solutions to the same scheduling problem, we define general criteria for chosing among feasible solutions as well as a genetic algorithm for exploring the solution space.

The second objective is to address the problem of jointly scheduling hard real-time and soft real-time traffic with different performance objectives : ensuring that the timing requirements of hard real-time traffic are met while minimizing as much as possible the response time of soft real-time traffic. We first evaluate the performance of the Dual-Priority policy for the scheduling of messages. Then we suggest a simple mechanism that provides probabilistic guarantees to prevent hard real-time frames from missing their deadlines under unreliable transmission. We also provide an on-line adaptive procedure for setting the parameters of the error model which is well suited for systems where the bus perturbation level may vary greatly over time. Finally, we propose a traffic shaping policy having the same goals as Dual-Priority but more easily applicable. This policy preserves feasibility and its complexity being linear in the number of hard real-time messages, it can be used on-line.

Keywords: Validation, Optimisation, Real-Time, Scheduling, Embedded Systems.

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE DU DOCTORAT DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :

Monsieur JUANOLE Guy, Professeur, LAAS, Toulouse,

Monsieur COTTET Francis, Professeur, LISI - ENSMA, Futuroscope,

Monsieur RUSINOWITCH Mikaël, Directeur de Recherche, INRIA LORRAINE, Villers les Nancy,

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Monsieur NAVET Nicolas

à soutenir devant un jury de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :

"Evaluation de performances temporelles et optimisation de l'ordonnance de taches et messages".

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 5 4 5 0 1
VANDŒUVRE CEDEX

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : **"INFORMATIQUE"**

Fait à Vandoeuvre le, **28 octobre 1999**

Le Président de l'I.N.P.L.,

J. HARDY

TEL. 33/03.83.59.59.59
FAX. 33/03.83.59.59.55

Résumé

Notre premier objectif est de proposer des méthodes et des outils de vérification du respect des contraintes temporelles d'une application temps réel. Le principal cadre d'application de nos travaux est celui des applications embarquées dans l'automobile distribuées autour d'un réseau CAN. La validation est menée en couplant les techniques de vérification : simulation, analyse et observation sur prototypes. L'apport principal de cette thèse réside en la conception de modèles analytiques qui fournissent des bornes sur les métriques de performance considérées (temps de réponse, probabilité de non-respect des échéances) ou permettant d'évaluer l'occurrence d'événements rares (temps d'atteinte de l'état bus-off d'une station CAN). Nous proposons également une analyse d'ordonnançabilité des applications s'exécutant sur des systèmes d'exploitation se conformant au standard Posix1003.1b . Ensuite, considérant qu'il existe généralement plusieurs solutions d'ordonnancement faisables à un même problème, nous avons défini des critères de choix et avons expérimenté une approche, utilisant un algorithme génétique, pour parcourir l'espace des solutions.

Notre second objectif est d'étudier des mécanismes d'ordonnancement qui garantissent le respect des échéances du trafic à contraintes strictes tout en minimisant les temps de réponse du trafic à contraintes souples. Nous évaluons les performances de la politique Dual-Priority pour l'ordonnancement de messages. Pour son utilisation dans des environnements bruités, nous proposons un mécanisme simple donnant des garanties sur la qualité de service exprimée en termes de probabilité de respect des échéances et s'adaptant en-ligne à des conditions de perturbations variables. Nous proposons également une politique concurrente, basée sur une technique de lissage de flux, qui est d'une mise en oeuvre plus aisée. Cette politique préserve la faisabilité du système et sa faible complexité algorithmique permet son utilisation en-ligne.

Mots-clés: Validation, Optimisation, Temps Réel, Ordonnancement, Systèmes Embarqués.

Abstract

In this thesis, we first investigate several approaches aimed at validating the temporal constraints of real-time systems. This work is applied to in-vehicle embedded applications distributed on a CAN network. The validation process is performed using different and complementary techniques : simulation, analysis and prototype monitoring. The main contribution is the design of analytical models that provide bounds on the considered performance metrics (response time, deadline failure probability) or that enable us to evaluate the occurrence of rare events (bus-off hitting time of a CAN node). We also propose a schedulability analysis of real-time applications running on Posix1003.1b compliant operating systems. Considering that usually there exist several solutions to the same scheduling problem, we define general criteria for chosing among feasible solutions as well as a genetic algorithm for exploring the solution space.

The second objective is to address the problem of jointly scheduling hard real-time and soft real-time traffic with different performance objectives : ensuring that the timing requirements of hard real-time traffic are met while minimizing as much as possible the response time of soft real-time traffic. We first evaluate the performance of the Dual-Priority policy for the scheduling of messages. Then we suggest a simple mechanism that provides probabilistic guarantees to prevent hard real-time frames from missing their deadlines under unreliable transmission. We also provide an on-line adaptive procedure for setting the parameters of the error model which is well suited for systems where the bus perturbation level may vary greatly over time. Finally, we propose a traffic shaping policy having the same goals as Dual-Priority but more easily applicable. This policy preserves feasibility and its complexity being linear in the number of hard real-time messages, it can be used on-line.

Keywords: Validation, Optimisation, Real-Time, Scheduling, Embedded Systems.