Qualité de service des pilotes d'équipements pour les systèmes d'acquisition de données

Belgacem Ben Hédia, Fabrice Jumel, Jean-Philippe Babau

CITI - INSA Lyon - CPE
F69621 Villeurbanne Cedex - FRANCE
{belgacem.ben-hedia; fabrice.jumel; jean-philippe.babau }@insa-lyon.fr

RÉSUMÉ. *Dans le domaine des applications temps réel de suivie et de contrôle des procédés, la validation temps réel s'appuie sur une connaissance fine des caractéristiques temporelles des données manipulées (principalement sous forme de retards et pertes). Lorsque ces données sont fournies par un logiciel dédié, appelé pilote, il est donc nécessaire de modéliser l'impact de ce logiciel sur la qualité de service des données manipulées. Cette étude propose un modèle formel des pilotes d'acquisition basé sur des automates temporisés communicants et montre l'impact de paramètres du pilote (période de scrutation, …) sur la qualité de service fournie.*

ABSTRACT. *In the field of real time application (especially for control purpose), validation is based on a fine knowledge of temporal properties of used data (in form of losses and delays…) If the data are processed using a dedicated software (called driver), it's necessary to model the consequence of this software part on the quality of service of the data. In This study we present a formal model of equipment driver based on timed automata and we show the influence of the characteristics of the driver (polling period …) on the offered quality of service.*

MOTS-CLÉS : *temps réel, automates temporises, pilotes d'équipements, validation, qualité de service*

KEYWORDS: *real time, timed automata, equipment drivers, validation, quality of service*

1. Introduction

L'utilisation de l'informatique dans le cadre du contrôle de procédé est de plus en plus courante (automobile, ferroviaire, procédés industriels). Le contrôle de procédé correspond au traitement de données en provenance de capteurs, qui permettent de connaître l'état du procédé, et la création éventuelle de commandes qui, appliquées via des actionneurs, permettent de contrôler l'évolution du procédé. Dans les domaines visés par cette étude, la criticité des procédés contrôlés requiert une validation a priori des systèmes de contrôle. Le procédé évoluant indépendamment du système de contrôle, la validité de ce dernier dépend fortement de ses propriétés temporelles. Par exemple, pour réaliser un contrôle valide, l'état du procédé doit être connu avec un retard borné ou pseudo constant, selon le type d'application (Jumel, 2003).

La validation du bon fonctionnement des systèmes temps réel, en prenant en compte les problématiques d'ordonnancement de tâches ou de messages, a fait l'objet de nombreuses études (Audsley et al 1993 ; Klein et al, 1993 ;Liu et al, 1973). Ces études s'appuient sur une vue globale simplifiée du système étudié. Par contre, de nombreux travaux restent à faire pour valider des systèmes dans le cadre d'architecture complexe. Dans cette étude nous nous intéressons à la caractérisation de la partie logicielle "pilotes d'équipement" des systèmes d'acquisition de données. Cette couche logicielle a pour but de servir d'interface entre les capteurs physiques et les applications. L'objectif de cette étude est de fournir des modèles des systèmes d'acquisition pour évaluer la qualité du service fournie par les pilotes et d'analyser les paramètres pouvant l'influencer.

La première partie présente le contexte général de l'étude via le principe de fonctionnement d'un pilote d'équipement et les critères de qualité retenus dans le cadre des systèmes d'acquisition. La deuxième partie montre comment il est possible de modéliser les différents éléments d'un pilote en utilisant des automates temporisés. La troisième partie valide sur un exemple élémentaire, la démarche proposée. Enfin, la dernière partie montre, dans un cadre plus réaliste, l'impact des paramètres d'un pilote sur les performances d'un pilote d'acquisition.

2. Contexte de l'étude

2. 1. Principe de fonctionnement des pilotes

Les systèmes d'acquisition de données étudiées sont composés de capteurs physiques, d'interface de communication avec ces derniers, du pilote proprement dit et d'une application temps réel (cf. figure1). On distingue le pilote matériel, appelé par la suite interface de communication, qui permet de connecter un capteur à l'application (convertisseur analogique/numérique, carte dédiée, bus) du pilote

logiciel qui fournit des services d'accès aux données en provenance d'un capteur pour une application donnée. Les services d'accès aux données peuvent être conçus soit simplement de manière élémentaire au niveau applicatif par des commandes assembleurs (par exemple l'accès à un registre), soit via un logiciel dédié intégré dans le système d'exploitation et indépendant de l'application. La mise en place d'un tel logiciel dédié, appelé pilote par la suite, possède plusieurs avantages. Une fois développé, il peut être utilisé dans des contextes applicatifs divers. De plus, l'utilisation d'un pilote offre une abstraction de la couche matérielle.

L'architecture du pilote dépend bien entendu des spécificités des équipements raccordés (protocole de communication du capteur et de l'interface de communication) mais aussi de la forme du système d'exploitation. Dans le cadre des systèmes temps réel respectant les normes POSIX (ISO, 1996) ou équivalentes, des règles précises ont étés adoptées. Elles permettent de fournir une interface générique d'accès aux données via les services classiques de type gestion de fichiers. Un capteur, est alors vu comme un fichier. Les primitives standards de manipulation d'un capteur s'appuient donc sur les primitives d'accès aux fichiers : create/open (connexion au capteur), remove/close (déconnexion du capteur), read (lecture sur le capteur) et ioctl/write (configuration des propriétés du capteur).

Si un pilote intégré possède des avantages pour la réutilisation, sa mise en place influence les performances du système. En particulier, le mode de fonctionnement interne des pilotes nécessite l'utilisation de mémorisation intermédiaire de données et de mécanismes pour assurer le transit des données à l'intérieur des différentes couches du pilote. Ceci entraîne inévitablement des retards et peut engendrer des pertes de données. Un pilote entraîne donc des relations complexes entre les données récupérées directement sur les équipements physiques et les données utilisées par l'application.

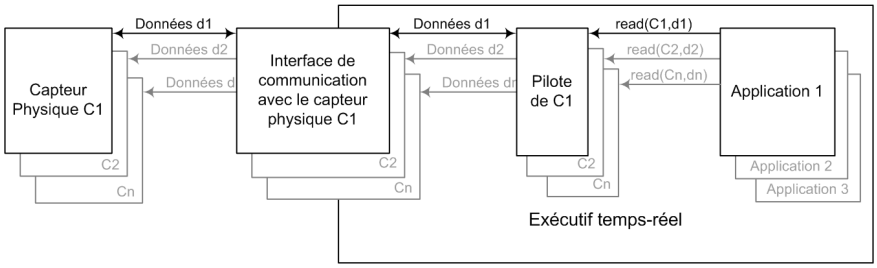


Figure 1. *Système d'acquisition de données*

2. 2. Etat de l'Art

Les études spécifiques de l'analyse de la qualité de service offerte par les pilotes d'acquisition sont à notre connaissance très rares. Le projet le plus proche, SLAM

(Thomas Ball, 2004), développé par Microsoft vise à la validation par analyse du code des pilotes (en particulier il s'agit d'assurer l'absence d'interblocage) mais ne permet pas de dériver des notions de qualité de service offerte. Dans le cadre de l'étude des pilotes d'équipement de stockage d'informations, on trouve un certain nombre d'études portant sur l'évaluation de performances. Ces approches s'appuient soit sur des techniques d'observation du système à l'exécution (traçage temporel) (Dedicated System, 2003 ; Robles, 1999), soit par la simulation de modèles ad hoc (David, 2003). Dans les études portant sur les périphériques de stockage, on s'intéresse principalement au retard car la présence de perte est, pour ce domaine, non acceptable et de nombreux mécanismes ont pour but de les prévenir. Les techniques d'étude utilisées sont donc de types statistiques ou probabilistes et les résultats obtenus ne permettent pas d'évaluer le retard maximum. Dans le cadre des systèmes temps réel, visés dans cette étude, les études de type probabilistes ne sont pas suffisantes pour garantir le bon fonctionnement du système. En effet, la validation s'appuie sur l'étude des pires cas d'exécution (par un exemple, un retard maximal).

Dans le domaine du temps réel, les techniques les plus utilisées pour permettre l'obtention de garanties s'appuient sur des études d'ordonnançabilité des tâches et des messages (techniques de type Rate Monotonic Analysis (Mammeri, 1998) ou fonction d'arrivée de travail (Migge, 2003)). Ces techniques considèrent des architectures simples, généralement statiques, et produisent des bornes des pires temps de réponse parfois peu réalistes. Pour pallier ces manques, de plus en plus d'études s'intéressent à l'utilisation de techniques basées sur une modélisation exhaustive des comportements temporels à l'aide d'automates temporisés communicants, d'automates hybrides ou de calcul symbolique (Belarbi et al, 2004 ; David, 2003 ; Hieu, 2003 ; Vestal, 2000 ; Waszniowski, 2003). La validation s'appuie alors sur un parcours exhaustif et l'utilisation de techniques de model-checking. Au delà des informations sur les temps de réponse, ces études permettent de produire, des informations générales sur le comportement de l'application (Belarbi et al, 2004 ; Juanole et al, 1999). Ces approches permettent de considérer des architectures plus complexes (comportements dynamiques et variables) que pour les approches citées ci-avant, même si les modèles traités restent limités en taille (quelques tâches).

Dans notre étude, le résultat recherché est la caractérisation temporelle des données produites par le pilote. Les architectures étudiées dépendent de nombreux paramètres (politiques de mémorisation, de communication) et sont caractérisées par des propriétés temporelles parfois complexes (lois d'arrivées des informations en provenance des capteurs, temps de réponses …). Nous avons opté pour une modélisation exhaustive des comportements temporels. Parmi, les différents choix de formalisme identifiés dont SDL (Mammeri, 2000), RdPT, Kronos (Bozga et al, 1998) et UPPAAL (David et al, 2003), nous avons fait le choix des automates temporisés communicants et plus précisément de IF (Bozga 1999 ; Bozga et al, 2002) couplé à CADP (Fernandez, 1999). Le choix de IF s'est imposé en raison de

la grande la facilité d'utilisation des modèles pour générer de manière exhaustive les graphes d'atteignabilité.

2. 3. Introduction au langage IF

IF (Intermediate Format) a été crée originalement pour servir de format intermédiaire entre des langages de haut niveau, en particulier SDL et les outils de model checking. Il peut cependant être utiliser directement et offre un pouvoir d'expression semblable à SDL.

Dans IF-2.0 (version utilisée pour l'étude), un système est décrit par un ensemble de processus sous forme d'automates temporisés communicants. La communication se fait de manière asynchrone via des signaux ou par partage de données. Au niveau d'un processus, les signaux sont gérés par file d'attente.

La sémantique opérationnelle de IF repose sur les modèles suivants:

le temps discret ; les systèmes de transitions étiquetées ;- les automates temporisés avec échéances pour la modélisation de l'aspect temporel (Bornot et al, 2000) ;- les automates communicants par files d'attentes (étendus pour prendre en compte les données.)

Pour chacun de ces modèles, la syntaxe, la sémantique dynamique et la définition des opérateurs de composition est décrite dans la thèse de Marius Bozga (Bozga, 1999).

Au niveau de la modélisation et l'écoulement du temps, les automates temporisés avec échéances, permettent de gérer explicitement l'urgence (dans le tir des transitions), contrairement au modèle classique qui repose exclusivement sur la notion d'invariant. L'urgence (*du tir des transitions)* est donc associée à l'aide d'un paramètre d'échéance : la transition est soit urgente (eager), soit retardable dans le respect des gardes imposés (delayable) , soit non urgente (lazy pour paresseuse).

L'intérêt principal de IF (dans notre étude) est la possibilité de générer facilement un modèle sémantique sous forme de systèmes à transitions étiquetées (LTS) de l'ensemble des exécutions possibles du système. Il est en outre possible d'instrumenter le modèle afin de calculer des propriétés lors de la génération du LTS. La sémantique particulière de IF permet de garantir la non introduction de blocage temporelle par composition de modèles. Cette propriété, non disponible avec les automates temporisés classiques basés sur la notion d'invariant, facilite considérablement la modélisation de systèmes complexes.

3. Modélisation d'un système d'acquisition de données

3.1. Introduction

Dans ce qui suit, nous proposons un modèle du système d'acquisition de données permettant l'analyse des performances et de la qualité de service fournies par le pilote. Chaque élément du système est représenté par un processus IF. La communication étant asynchrone en IF-2.0, une consultation de donnée est modélisée par un envoi de signal *read ou get* et l'attente d'un retour *valX()* (cf. figure 2).

Nous présentons maintenant les paramètres principaux de chaque élément modélisé.

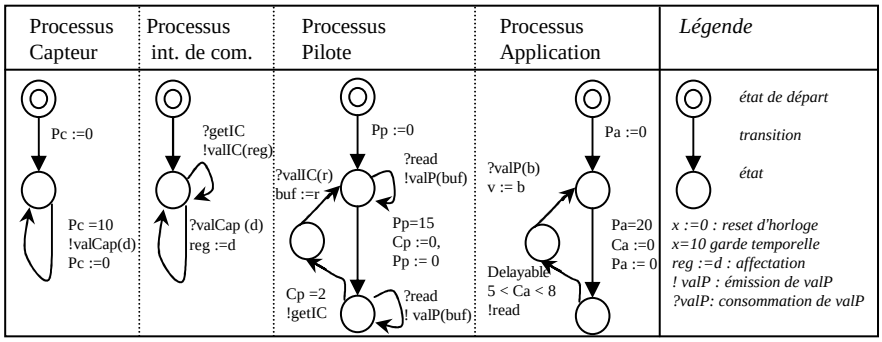


Figure 2. *Modèle de système d'acquisition pour une émission périodique (période Pc de 10) , un pilote à scrutation (période Pp de 15, durée Cp de 2) et une application en mode scrutation (période Pa de 20, durée Ca de 6 à 7)***.**

Un capteur physique traite, ou convertit, des informations en provenance du procédé. Les données sont produites suivant une loi de production donnée. Dans notre modèle, on peut considérer des lois périodiques (avec ou sans gigue), sporadiques ou en rafale. Dans cette étude seul le cas périodique est présenté.

L'interface de communication récupère les données produites par le capteur physique et les range suivant une politique donnée (FIFO,LIFO…) dans des registres (caractérisés par leurs tailles) accessibles au pilote. Si les registres sont pleins (données non consommées), la politique d'écrasement peut suivre plusieurs lois (perte de la donnée, ou écrasement de la première ou de la dernière donnée mémorisée).

Le pilote est la partie logicielle qui permet la prise en charge des données stockées dans les registres et assure leur transmission à l'application. Pour gérer les données qui proviennent du capteur et les appels des applications (primitive read), une architecture multitâche complexe peut être mise en oeuvre. Dans cette étude,

nous nous limitons à une architecture classique simple mono-tâche. La tâche consomme les données en lisant les registres de l'interface de communication puis les sauvegarde dans une variable. Lors de la consultation par l'application, le pilote renvoie les informations mémorisées dans la variable. Le fonctionnement du pilote est donc défini par la loi d'activation de la tâche (mode scrutation et mode événementiel) et par la politique de stockage des informations. En mode scrutation, le pilote consulte périodiquement les registres de l'interface de communication. En mode événementiel, le pilote fixe une politique d'envoi à l'interface de communication. A chaque nouvelle donnée, l'interface de communication vérifie une condition d'envoi fixée par le pilote et envoie si nécessaire les données au pilote. Pour les politiques de stockage, on distingue la taille des buffers de mémorisation des données, la politique de stockage (FIFO ou LIFO) et les politiques d'écrasement en cas de buffer plein.

3.1.4 L'application

L'application est la partie logicielle qui consomme les données en consultant les variables mémorisées par le pilote (appel à la primitive read). Nous considérons ici une application monotâche. L'application peut fonctionner suivant deux modes, scrutation ou événementiel. Leur logique d'activation est semblable à celle décrite dans la partie précédente.

Dans le mode scrutation, l'application est caractérisée par la période de scrutation. Dans le mode événementiel, l'application consomme les données lorsque elle sont produites par le pilote (i.e. l'application exécute toujours un read en attente active).

3.1.4 Prise en compte des caractéristiques temporelles du système

Dans IF les transitions se franchissent en durées nulles. Le temps n'est considéré qu'au travers des dates de tir de transition. Il est possible de fixer via des horloges, des intervalles dans lesquels les transitions peuvent et doivent être franchies (cf. eager, delayable et lazy). La modélisation des lois de production (périodique avec ou sans gigue, sporadique, rafales) ne pose donc pas de problème particulier. La modélisation d'une durée de traitement (lié au temps d'exécution des tâches) nécessite la mise en place d'états d'attente dans les différents processus. La durée d'attente modélise dans cet état la durée du traitement (cf. figure 2).

Les durées d'attente sont, dans le cas de l'étude, facile à évaluer car par hypothèses, le pilote est ordonnancé comme une tâche de forte priorité non préemptive.

3.2. Evaluation des Critères de qualité de service pour les pilotes d'acquisition

3.2.1. Définition des critères de qualité

Le but des pilotes présentés précédemment est de récupérer des informations sur l'évolution d'un procédé et donc sur son état. Les critères de qualité de service attendue pour ce type d'application sont donc basés sur les notions suivantes :

-Les retards : les données utilisées par l'application reflètent bien l'état du procédé mais à un instant plus ou moins passé.

-Les pertes : certaines informations ne seront jamais rendu disponible pour l'application et sont donc perdues.

Les données produites par les capteurs constituent des flux de données. Cela signifie que les données produites sont liées entre elle et que la qualité de service fournie est non seulement définie par des propriétés exprimées sur chaque donnée mais aussi sur le flux de données. En fonction des applications, les critères les plus pertinents donnés pour un flux vis à vis des pertes et des retards peuvent différer (Jumel, 2003) et ne font pas l'objet de cette étude.

Dans cette étude, les critères étudiés sont le retard minimal et le retard maximal ainsi que le pourcentage maximal de perte dans un intervalle de temps donné.

La notion de moyenne ne peut pas être étudié avec le formalisme retenu car il n'est pas possible de quantifier les différents chemins obtenus. Il serait nécessaire pour cela de joindre à la sémantique de IF une extension stochastique ou un équivalent ce qui n'est pas l'objet de notre travail actuel.

3.2.2 Instrumentation et analyse des Modèles

Afin de permettre l'observation de l'évolution temporelle du pilote et de cerner tous les cas d'exécution, des variables d'observation doivent être introduites dans le modèle. Ces variables sont des compteurs de données, reçues par l'application ou envoyées par le capteur, et des compteurs temporels (dates de production/consommation des données).

C'est à partir de ces différentes variables que les critères de qualité de service pourront être calculé au cours de la génération du LTS. L'introduction de variables d'observation, par exemple le compteur de messages, fait exploser le modèle lors de la génération du LTS. Afin de réaliser une analyse, il est nécessaire de fixer une durée de fonctionnement finie du système pour obtenir un LTS fini. Ce temps de fonctionnement doit permettre de garantir l'exhaustivité des cas simulés. Pour cela, on procède en deux étapes. On réalise un premier modèle sans les variables d'observation. Il est à noter que les variables n'impactent pas le comportement du

système étudié. Le modèle sans variable d'observation doit être fini (comportement cyclique) et nous fourni le temps d'exécution nécessaire à la simulation du modèle avec variable d'observation.

4. Validation des modèles

4.1 Présentation du système

Afin de valider et d'illustrer les modèles réalisés, nous présentons un premier exemple volontairement simple. Les paramètres du système d'acquisition de données sont fixés aux valeurs suivantes :

La période de production est périodique avec une période fixe *Pc* de 10 unités de temps ; La taille du registre de l'interface de communication est de 1 ; Le pilote travaille en mode scrutation. La période *Pp* n'est pas déterminée a priori ; Le pilote gère un buffer de 1 élément ; Le temps de réponse du pilote (*Cp*) est considéré comme constant et égal à 2 unités de temps ; La période de scrutation *Pa* de l'application est de 20 unités de temps et sa durée est nulle ; Enfin, on considère que toutes les activités (capteur, pilote, application) démarrent au même instant.

On cherche à quantifier pour cette architecture, les critères de qualité de service retard (maximal et minimal) et pourcentage de perte maximal

Les paragraphes suivants présentent les résultats obtenus grâce à la méthodologie présentée et leurs justifications analytiques.

4.2 Evaluation du pourcentage maximal de pertes

La figure 3 présente l'évolution du pourcentage de perte maximal en fonction d'une période de scrutation du pilote variant entre 3 et 42 unités de temps. Dans cette étude, on s'intéresse au maximum de perte pour tout intervalle de temps [t1, t2]. La forme de la courbe obtenue est relativement simple, le pourcentage de perte augmente avec la période de scrutation du pilote. On distingue trois paliers constants

- Sur le premier palier, la période de production du capteur (10) et la période de consommation par l'application (20) implique une perte minimale de 50%. A ce niveau, l'information pertinente est donnée par la limite du palier égale à 18 unités de temps. Cette limite est due à la durée de traitement du pilote de 2 unités de temps.

- Le deuxième palier correspond à la perte de 3 données sur 4 produites au niveau du pilote.

- Le troisième palier apparaît à partir d'une période de scrutation du pilote supérieur à 38 unités de temps et correspond à la perte de 5 données sur 6 produites.

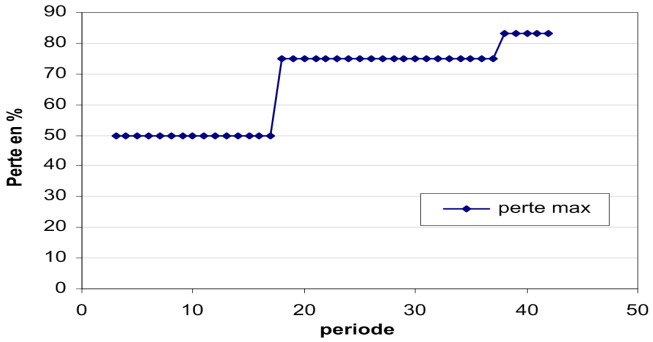


Figure 3. *Pourcentage de perte maximal en fonction de la période du pilote***.**

4.3 Evaluation des retards

La figure 4 présente l'évolution du retard minimal et du retard maximal en fonction de la période de scrutation du pilote. Pour le retard minimal, on constate trois valeurs possibles égales à 0, 10 et 20 unités de temps. Nous allons maintenant expliquer cette courbe et justifier des valeurs obtenues par une approche analytique.

Un retard nul correspond a l'enchaînement de la production, de la scrutation par le pilote (+durée d'exécution) et de la consommation par l'application. Ces correspondances peuvent être mises en équation :

$$n1 * Tc = n2 * T + Cexec = n3 * Ta \quad [1]$$

– *n1 * Tc* représente les différentes dates de production
– *Cexec* représente le temps de traitement du pilote
– *n2 * Tp + Cexec* représente les différentes dates de production par le pilote
– *n3 * Ta* représente les différentes dates de consommation par l'application

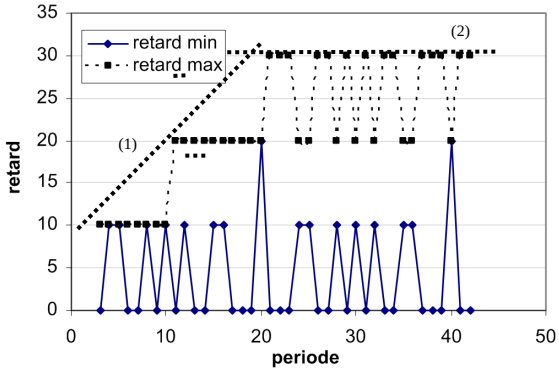


Figure 4. *Retards minimal et maximal en fonction de la période du pilote ; (1) et (2) : bornes obtenues par une analyse de type RMA.*

Il s'agit de savoir si l'équation suivante possède des solutions entières positives en *n1*, *n2* et *n3*. La résolution de ce type d'équations (dites diophantiennes) est relativement simple (basée sur le calcul du plus grand commun diviseur). Si une solution existe alors la correspondance est possible et le retard est nul. Ces résultats sont correctement pris en compte dans la génération exhaustive.

Dans les autres cas, le pilote ne se synchronise pas avec le capteur et l'application. Dans ce cas, au mieux, le retard est de 10 unité de temps (temps passé dans le registre).

Il existe un cas particulier pour les multiples de 20. Dans ce cas, on note un retard constant, et donc minimal, de 20. En effet, pour ce cas, on note un séquencement particulier et cyclique. Dans tous les autres cas, du fait de la désynchronisation entre les différentes activités, le pilote arrive toujours au moins une fois à récupérer et fournir une donnée avec un retard de 10 unité de temps.

Pour le retard maximal, une explication équivalente est possible mais n'est pas développée faute de place.

4.4 Approche analytique

Pour l'exemple présenté, il est intéressant de comparer les résultats obtenus à ceux issus d'une analyse de type RMA. Nous allons maintenant donner le principe d'évaluation des retards avec des hypothèses simplificatrices liées à l'exemple traité (périodicité, durées constantes).

- A priori, le retard minimal est nul et correspond à la synchronisation de la production, de l'acquisition par le pilote et par l'application

Pour le retard maximal, on distingue le cas où le pilote possède une période inférieure, respectivement supérieure, à celle de l'application.

- Dans le premier cas, au pire, la donnée est consommée par le pilote juste avant d'être modifiée par le capteur. Son retard est alors égal à la période du capteur, notée *Tc*. Ensuite, l'application peut la consommer, au pire, juste avant qu'elle soit, à nouveau, modifiée par le pilote, soit un retard maximal entre deux mises à jour égal à *Tp*. Au final, le retard maximal est égal à :$Tc + Tp$

- Dans le deuxième cas, le pilote n'arrive plus à fournir des données assez rapidement pour l'application. Dans ce cas, si il fournit une information, elle est consommée, au pire, à la prochaine activation de l'application (retard de Ta). En intégrant le retard maximal du au capteur (cf. ci-avant), au final, la donnée possède un retard maximal égal à (cf. droite (2) de la figure 4) : $Tc + Ta$.

Dans l'exemple présenté figure 4, ces résultats correspondent aux droites (1) et (2) de la figure 4. On voit alors aisément l'intérêt d'une approche exhaustive pour obtenir des résultats réalistes.

4.5 Analyse et Conclusion

On remarque évidemment que les propriétés de qualité de service se dégradent avec l'augmentation de la période de scrutation du pilote. Néanmoins, ce n'est pas en scrutant plus vite que les performances sont obligatoirement meilleures. Enfin, on trouve généralement un point limite égale à la période de production de capteur physique. Cet exemple simple a été présenté dans le but de valider l'approche proposée et aussi de l'illustrer.

Il est à noter que pour cet exemple, les résultats présentent beaucoup d'irrégularités. Cela illustre que la qualité de service fournie par un pilote ne suit pas une loi linéaire simple et qu'elle peut être difficile à prédire analytiquement. Ceci justifie, a posteriori, l'utilisation de modèles exhaustifs pour les caractériser. Il est à noter que les études menées sur d'autres paramètres tels que la politique de stockage (Ben Hedia, 2004) augmente la complexité des modèles manipulés et la forme des résultats obtenus. Il apparaît alors difficile de produire ou d'estimer a priori un modèle simple et réaliste des propriétés de qualité de service fournies par un pilote d'acquisition.

Enfin, cette première étude démontre que la qualité de service fournie par un composant logiciel de type « pilote d'acquisition » ne peut être fournie sans connaissance de son environnement d'exécution (capteur et l'application).

5. Application à un système plus réaliste

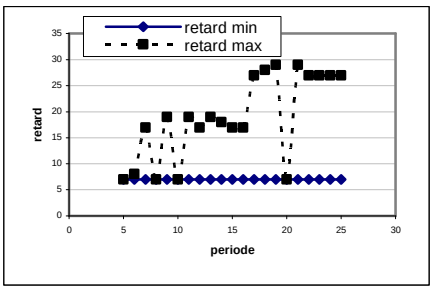
Nous présentons maintenant l'utilisation des modèles dans des contextes d'utilisation plus complexe où l'intérêt de l'approche semble plus pertinente encore.

5.1 Présentation

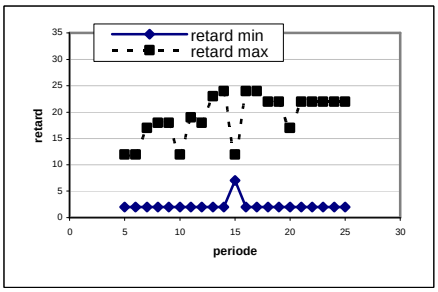
L'exemple précédent peut être, pour bien des raisons, considéré comme élémentaire. Tout d'abord le temps de traitement de l'application a été négligé, il est pris en compte pour l'obtention des retards présentés dans la figure 5a (considéré comme constant à 7 unités de temps). Le cas précédemment est très particulier en raison de la synchronisation des différents ensembles du système, la figure 5b présente les résultats obtenus pour une période de consommation de 15 unités de temps. Enfin dans des cas réalistes, les paramètres temporels considérés ne sont pas constants mais généralement variables tout en restant bornés. La figure 5c représente les résultats obtenus pour un temps de réponse de l'application variable dans l'intervalle [6,7]. Pour la figure 5d, on considère en plus que le temps de réponse du pilote est lui aussi variable et compris dans l'intervalle [1,2].

5.2 Résultats et Analyse

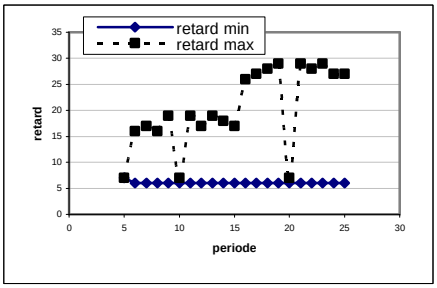
Comparons, les Figures 4 et 5a. Le cas 4 est en fait un cas très particulier car il offre de nombreux points de synchronisation (entre le capteur, le pilote et l'application) où le choix qui est fait dans l'ordre des opérations, à un même instant, va entraîner des résultats totalement différents. Ce phénomène explique la forme complexe des résultats qui sont détaillés dans la section précédente. Les courbes de la figure 5a sont plus lissées. Le retard minimal est maintenant de 7 (lecture toutes les (20 k +7) unités de temps d'une donnée à priori produites par le capteur toutes les (20 k) unités de temps). Quelques points remarquables demeurent, correspondants à un séquencement particulier, en particulier 10 et 20 où la synchronisation entre l'application, le capteur et le pilote amènent à l'existence d'un séquencement unique entraînant un retard possible unique de 7 unités de temps. La figure 5c est encore plus lissée, on retrouve comme uniques points particuliers, les points de synchronisation 5, 10 et 20. La figure 5d, est très semblable à la figure 5c, à part quelques petites différences de +/- 1 liés à l'influence de l'intervalle [1,2]. Enfin sur la figure 5b, le fait de changer la période de l'application de 20 unités de temps à 15 unités de temps entraîne des changements considérables au niveau, bien sûr, des valeurs obtenues, mais aussi de la forme de la courbe.



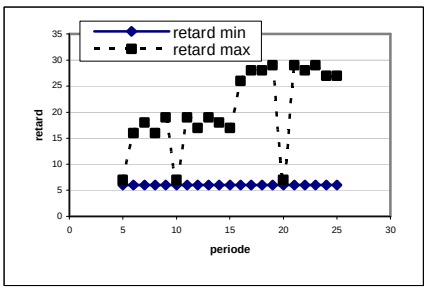
(a) Temps de réponse du pilote : 2
Période application : 20, Temps de réponse : 7



(b) Temps de réponse du pilote 2
Période application 15, Temps de réponse : 7



(c) Temps de réponse du pilote 2
Période application 20, Temps de réponse dans [6,7]



(d) Temps de réponse du pilote [1 2]
Période application 20, Temps de réponse dans [6,7]

Figure 5. *Evolution du retard vis-à-vis de la période de scrutation du pilote pour différentes configurations du système.*

5.3 Conclusions

Nous avons sur cet exemple montré la grande richesse d'expression des modèles proposés. En particulier, il est possible de prendre en compte l'existence de propriétés temporelles définis sous forme d'intervalles. Les résultats des analyses analytiques n'ont généralement pas de formes clauses et ne présentent donc pas d'intérêt particulier vis-à-vis de ceux obtenus par analyse exhaustive.

L'intérêt de l'analyse exhaustive est de concentrer le travail sur la création des modèles alors que pour une analyse classique un travail important doit être fait pour l'obtention des résultats. Bien entendu, le problème majeur de l'analyse exhaustif est le risque d'explosion combinatoire. La tableau 1 présente l'évolution du nombre d'états du LTS généré (sans compteurs). Les exemples a et b traités au niveau de la figure 5 ne posent pas de difficultés. On remarque, par contre, qu'un exemple plus complexe incluant une non périodicité de la production augmente considérablement le nombre d'états du LTS. Cet exemple n'a donc pas pu être traité car l'analyse des modèles (en rajoutant les compteurs) explose sur la machine de faible puissance utilisée pour réaliser les expérimentations.

Exemple	***a***	***b***	***C***
période de production	10	10	[10,11]
période du pilote	7	7	7
durée driver	2	[1,2]	[1,2]
période de l'application	20	20	20
durée de l'application	7	[6,7]	[6,7]
nombre d'états (sans compteur)	387	1676	19331

Tableau 1. *Evolution du nombre d'états en fonction de la complexité de l'exemple*

6. Conclusion générale

Nous avons proposé des modèles de spécification comportementale et temporelle d'un système d'acquisition de données. Ces modèles permettent de caractériser un pilote d'acquisition du point de vue de la qualité de service. Ces modèles ont été réalisés en IF et prennent en compte le capteur, l'interface de communication, le pilote et l'application. Les résultats présentés ont montré que la forme des propriétés de qualité de service est complexe et dépend de plusieurs paramètres. Cette étude peut aussi servir pour le dimensionnement ou l'adaptation d'un pilote (choix dune période de scrutation, d'une taille de buffers) pour répondre aux exigences d'un contexte d'utilisation précis.

Les perspectives de cette étude portent sur la prise en compte d'architectures plus complexes (multitâches, multiplexage des données de plusieurs capteurs,

structuration en couches) et sur des modèles temporels plus complexes (temps continu, modèles stochastiques).

Les problèmes d'explosion combinatoire soulevés aux paragraphes précédents feront bien entendu l'objet d'une étude approfondie. Nous évaluerons les réductions par équivalence et/ou l'introduction de résultats analytiques évalués a priori.

Bibliographie

N. C. Audsley, A. Burns, M. F. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling" *Software Eng. J., vol. 8, no. 5*, pp. 284–292, Sept. 1993.

T. Ball, "SLAM and Static Driver Verifier: Technology Transfer of Formal Methods inside Microsoft" , *Integrated Formal Methods 2004*, Cantebury, England

M. Belarbi, J.-P. Babau, J.-J. Schwarz, "Temporal Verification of Real-Time Multitasking Application Properties Based on Communicating Timed Automata", *Proc. in 8-th IEEE International Symposium on Distributed Simulation and Real Time Applications*, Budapest, 2004.

B. Ben Hedia, Modélisation formelle d'un driver d'acquisition des données, stage de master II de recherche, Insa de Lyon, Juillet 2004.

S. Bornot , J.Sifakis. "An Algebraic Framework for Urgency", *Information and Computation 163*,172-202 ,2000.

M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: "A model-checking tool for real-time systems". *In Proc. of the 10th Conference on Computer-Aided Verification*, Vancouver, Canada, 1998.

Bozga, Dorel Marius. Vérification symbolique pour les protocoles de communication. Thèse de doctorat, Verimag, Université Joseph Fourrier, 1999.

M. Bozga, S. Graf, L. Mounier, "IF-2.0: A Validation Environment for Component-Based Real-Time Systems", In Ed Brinksma, K.G. Larsen (Eds.) *Proceedings of CAV'02 (Copenhagen, Denmark) LNCS* vol. 2404 Springer-Verlag July 2002.

A. David, G. Behrmann, K. G. Larsen, and W. Yi. "A tool architecture for the next generation of uppaal", Technical report, Uppsala University, 2003.

Dedicated System, "Comparison report between QNX NEUTRINO RTOS v6.2, VxWorks AE 1.1, Windows CE .NET 2.52 and Red Hat Embedded Linux Development Suite", ELDS v1.1, Technical report, 2003.

J-C Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, M. Sighireanu, "CADP A Protocol Validation and Verification Toolbox". *8th International Conference on Computer Aided Verification CAV*, New Brunswick (USA), 1996.

F.J. Fokkink, N.Y. Ioustinova, E. Kesseler, J.C. van de Pol, Y.S. Usenko and Y.A. Yushtein, "Refinement and verification applied to an in-flight data acquisition unit", *in Proc. 13th Conference on Concurrency Theory - CONCUR'02*, Brno, Lecture Notes in Computer

ISO/IEC Standard 9945-1: 1996 [IEEE/ANSI Std 1003.1, 1996 Edition] *(POSIX)—Part 1: System Application: Program Interface (API) [C Language]*, 1996.

G. Juanole , I.Blum , "Influence de fonctions de base (communication-ordonnancement) des systèmes distribués temps-réel sur les performances d'applications de contrôle-commande", Rapport LAAS No98376, *7ème Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'99), Nancy (France)*, 26-29 Avril 1999,

F. Jumel , Définition et gestion d'une qualité de service pour les applications temps réel, thèse de doctorat, INPL, novembre 2003

M. Klein & all, *A practionner's Handbook for Real-Time Analysis*, Kluwer Academic Publishers, 1993.

C.L. Liu, J.W. Layland "Scheduling algorithms for multiprogramming in a hard real time environment" *Association for Computing Machinary*, 20 (1), 46-61,1973.

Z. Mammeri., "Expression et dérivation des contraintes temporelles dans les applications temps réel". *Dans : Journal Européen des Systèmes Automatisés, JESA-APII*, Hermès, V. 32 N. 5-6, p. 609-644, 1998.

Z. Mammeri. *SDL - Modélisation de protocoles et systèmes réactifs*, Hermès sciences, juillet 2000.

J. Migge, A. Jean-Marie, N. Navet "Timing analysis of compound scheduling policies : Application to posix1003.b ", *Journal of Scheduling, Kluwer Academic Publishers*, 6 (5), 457-482, 2003.

P. Ramanathan,,. "Overload Management in Real-Time Control Applications Using (m, k)-Firm Guarantee". *IEEE Transactions on Parallel and Distributed Systems* 10(6): 549-559, 1999

S. Vestal "Modeling and Verification of Real-Time Software Using Extended Linear Hybrid Automata" , *Proc Fifth NASA Langley Formal Methods Workshop*, 2000.

B.Sprunt, L. Sha, J. Lehoczky "Aperiodic Task Scheduling for Hard Real Time Systems" *Real Time Systems*, 1, 27-60, 1989.

Time-Triggered Technology TTTech Computertechnik AG. "Specification of the TTP/C protocol". ,Technical report, Vienna, Austria, July 1999.

L. Waszniowski, and Z. Hanzálek, "Analysis of Real Time Operating System Based Applications", *Proc. First International Workshop on Formal Modeling and Analysis of Timed Systems 2003*, LNCS Springer, Marseille, 2003.

A. Zahir, P. Palmieri, "OSEK/VDX-operating systems for automotive applications", *in OSEK/VDX Open Systems in Automotive Networks*, 1998