HAL
open science

Evaluation des délais de réactivité des architectures de commande distribuées sur réseau Ethernet

Gaëlle Marsal, Bruno Denis, Jean-Marc Faure

To cite this version:

Gaëlle Marsal, Bruno Denis, Jean-Marc Faure. Evaluation des délais de réactivité des architectures de commande distribuées sur réseau Ethernet. Conférence Internationale Francophone d'Automatique, CIFA 2006, Bordeaux (France), May 2006, France. CDRom paper N°283. hal-00361041

HAL Id: hal-00361041

https://hal.science/hal-00361041v1

Submitted on 13 Feb 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Évaluation des délais de réactivité des architectures de commande distribuées sur réseau Ethernet

Gaëlle MARSAL[1], Bruno DENIS[1], Jean-Marc FAURE[1,2]

[1] LURPA - ENS de Cachan - Université Paris Sud
61, avenue du Président Wilson, 94235 CACHAN CEDEX, FRANCE

[2] SUPMECA
3, rue Fernand Hainaut, 93407 ST OUEN CEDEX, FRANCE

Prenom.Nom@lurpa.ens-cachan.fr
http://www.lurpa.ens-cachan.fr

Résumé— **L'idée d'avoir un média universel comme bus de terrain est séduisante. Ainsi, le développement de composants d'automatisation communicant sur un réseau Ethernet commuté utilisant des protocoles de communication standards et ouverts est d'actualité. Cependant, ce type de réseau introduit des mécanismes de partage de ressources et d'asynchronisme entre processus parallèles. Pour les architectures devant assurer la commande de systèmes à événements discrets, la conjonction de ces mécanismes rend difficile l'évaluation du délai de réactivité, retard entre l'occurrence d'un événement et celle de sa conséquence sur le processus. Nous proposons dans cet article une méthode basée sur la simulation d'un modèle dynamique en réseau de Petri hiérarchique coloré temporisé qui prend en compte les mécanismes d'asynchronisme et de partage de ressources. Deux cas tests permettent de mettre en évidence l'intérêt de la modélisation proposée.**

Mots-clés— **Réseau de terrain, systèmes à événements discrets, systèmes réactifs, réseau de Petri, simulation.**

I. Introduction

Les avantages du média Ethernet en tant que réseau de terrain sont nombreux. Citons par exemple l'uniformisation des média de communication dans l'entreprise et la reconfiguration facilitée par la communication possible entre tous les composants. Pour profiter de ces avantages en gardant une excellente réactivité de la commande, de nombreux constructeurs et chercheurs ont travaillé sur des protocoles spécifiques comme Profinet [1] ou DP-Ethernet [2]. Cependant, ces protocoles sont dépendants d'un constructeur particulier et ne permettent pas l'utilisation de composants réseaux standards [3].

Dans cette étude, au contraire, nous nous intéressons aux réseaux Ethernet commutés utilisant des protocoles standards et ouverts, plus particulièrement Modbus/TCP, de type client/serveur. Cependant, l'utilisation de ces réseaux introduit des délais difficiles à déterminer. La première cause de délai est l'ajout de composants (commutateurs) qui induisent des retards variables. En effet, ces composants sont des ressources partagées, et suivant leur charge, ils vont retarder plus ou moins les trames. De nombreux travaux se focalisent sur ce point, [4], [5], [6]. Une deuxième cause de délai est le partage des modules d'entrées/sorties (E/S). Ceci est dû au fait que les protocoles client/serveur ne permettent pas de contrôle global d'accès au média, contrairement aux protocoles maître/esclave ou producteur/distributeur/consommateur. Enfin, la dernière cause de délai est l'asynchronisme entre l'exécution du programme de commande et la scrutation des entrées sorties. Sur ce dernier aspect, une approche intéressante est présentée dans [7]. Elle se base sur une approche formelle de model-checking sur des automates communicants afin de déterminer des retards minimum et maximum. La formalisation du problème sous forme d'une équation diophantienne permet de valider les résultats obtenus.

Chaque approche présentée se concentre sur un problème isolé du reste. Toutefois, dans le cas de la commande d'un système à événements discrets (SED) à laquelle se limite cet article, la performance attendue au final est le délai de réactivité. Ce délai est défini par le retard entre un événement "cause" généré par le processus commandé et l'événement "conséquence" fourni en réponse au processus (dr_i sur la figure 1). Le chronogramme sur la droite de cette figure illustre la variabilité du délai, ce qui implique que les performances de réactivité doivent être caractérisées par la distribution du délai de réactivité ou de certaines des caractéristiques de cette distribution (minimum et maximum par exemple).

Pour évaluer ce délai de réactivité, il est utile dans un

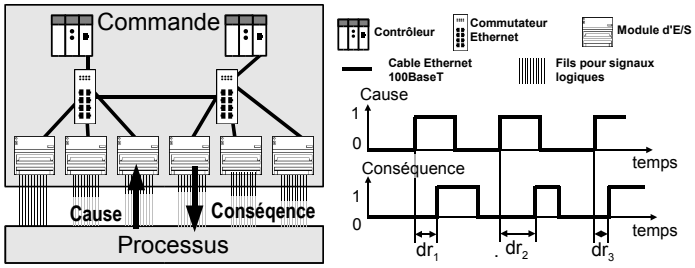


Fig. 1. Délai de réactivité

premier temps de le décomposer en délais élémentaires, qui correspondent aux délais induits par les différents composants de l'architecture. Une méthode classique pour évaluer le délai de réactivité consiste alors à sommer ces dé-

lais élémentaires calculés indépendamment de la charge des composants et de leur dépendance. Pour éviter ces approximations, c'est-à-dire prendre en compte la charge et leur dépendance, nous proposons une méthode de détermination du délai de réactivité basée sur la simulation d'un modèle réseau de Petri hiérarchique coloré temporisé (RdPHCT) de l'architecture de commande. Les comportements de chaque composant sont modélisés isolément par un RdPHCT, en prenant en compte les ressources partagées (commutateurs et modules d'E/S) et en définissant les communications possibles entre les composants.
Dans un premier temps, nous nous appliquerons à montrer l'influence des délais sur les performances de la commande des SED. Puis, la méthode classique de détermination du délai de réactivité basée sur la sommation de délais élémentaires sera présentée. Ensuite, le modèle RdPHCT de l'architecture sera détaillé. Enfin, deux architectures tests permettront de comparer ces deux méthodes et de conclure sur la validité des hypothèses faites pour chacune.

II. Réactivité des architectures de commande

A. Architectures de commande distribuées sur Ethernet

Seules les topologies de réseau de type Ethernet commuté, c'est-à-dire comprenant uniquement des commutateurs comme composants réseaux actifs, sont étudiées. De plus, l'étude présentée se limite aux systèmes de commande composés de contrôleurs industriels à moniteur périodique et de modules d'E/S déportées. Les contrôleurs échangent leurs données (valeurs d'E/S) avec les modules d'E/S via le réseau Ethernet commuté. Ces échanges se font selon un protocole client/serveur, dans notre cas Modbus TCP/IP. Le coupleur Ethernet du contrôleur est le client tandis que les modules sont les serveurs.
Cet article traite plus particulièrement des contrôleurs modulaires (un module CPU et un module Ethernet), des modules d'E/S déportées serveur Ethernet et des commutateurs Ethernet sans qualité de service. Le contrôleur a deux applications, une par module, le moniteur d'exécution périodique qui exécute le programme de commande dans le module CPU et le client Ethernet dans le module Ethernet qui envoie cycliquement des requêtes aux modules d'E/S déportés pour acquérir l'état des entrées et rafraîchir l'état des sorties. Ces deux applications s'exécutent de manière asynchrone, en parallèle, et communiquent par mémoire partagée. Pour ce qui est des communications, nous nous focalisons sur la couche application et en particulier sur la scrutation cyclique des entrées et sorties.

B. Délai de réactivité

Seul le délai de réactivité illustré par la figure 1 est étudié, c'est-à-dire le retard entre un événement "cause" généré par le processus commandé et l'événement "conséquence" fourni en réponse au processus. Dans le cadre de la commande des SED, deux cas de figure peuvent être distingués pour la validation d'une architecture par rapport aux spécifications :

- le délai de réactivité concerne une commande cyclique, alors la distribution du retard doit être évaluée,
- ou il concerne la réaction à une alarme, alors la borne maximale du délai suffit pour conclure.

Ce délai de réactivité résulte à la fois du comportement du contrôleur et des protocoles de communication. Il peut être exprimé de diverses manières. Nous avons choisi de l'étudier par une approche basée sur les composants. Ainsi la figure 2 illustre la décomposition du délai de réactivité en délais dans les composants de l'architecture. L'occupation des différentes ressources dans le temps est représentée sous forme de chronogrammes, et les flèches verticales représentent les échanges d'information, que ce soit des trames Ethernet ou des signaux logiques.
En haut de cette représentation se trouve le processus,

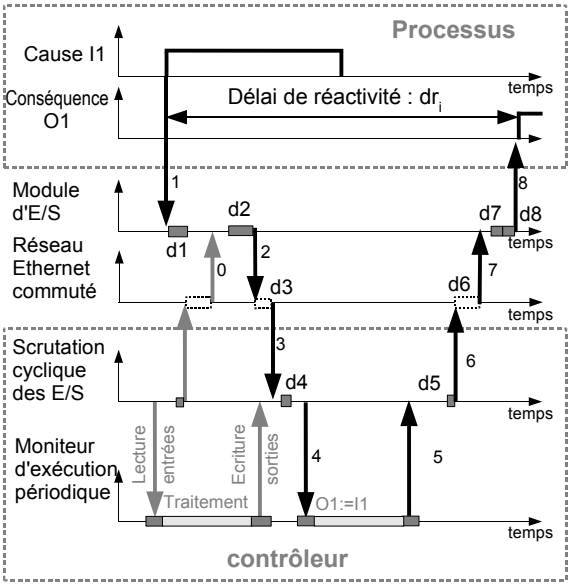


Fig. 2. Décomposition du délai de réactivité

où se produit un événement "cause", ici sur l'entrée I1. La flèche 1 représente la propagation de l'information du processus au module d'E/S, dont nous supposons la durée négligeable. L'occupation pendant un temps $d1$ représente le délai dû au filtrage. Cette valeur de l'entrée est envoyée dans la prochaine réponse (flèche 2) à une requête venant du contrôleur, après un temps de traitement d2 pour décoder la requête puis générer la réponse. Ensuite la trame passe un temps $d3$ dans le réseau pour arriver au coupleur Ethernet du contrôleur. Après extraction des données de la trame pendant une durée $d4$, les valeurs d'entrées seront prises en comptes à la prochaine phase de lecture des entrées du moniteur d'exécution du contrôleur. Après traitement puis mise à jour des sorties en mémoire, une trame contenant l'événement "conséquence" (la valeur de la sortie O1) est envoyée dès le prochain cycle de scrutation (flèche 6). La traversée du réseau durera un temps $d6$, potentiellement différent de $d3$, suivant le trafic instantané dans le réseau. Enfin après traitement dans le module, la sortie sera mise à jour. Il est possible de décomposer le délai de traversée de chaque composant en trois parties, (voir Fig. 3 pour le cas d'un module d'E/S) :

- les délais intrinsèques, dans le cas du module d'E/S, $d1$ pour le filtrage du signal logique et $d2$ pour le traitement de la trame Ethernet,
- le délai introduit par l'asynchronisme entre processus parallèles, d_{asynch}

– et le délai introduit par l'attente de disponibilité de la ressource, d_{ch}.

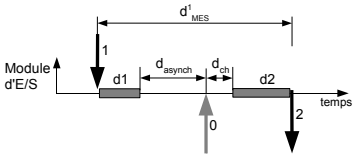


Fig. 3. Causes de délai (exemple d'un module d'E/S)

En notant d^1_{MES} le délai passé dans un module d'E/S entre l'occurrence de I1 (1) et l'envoi de la trame (2), on peut écrire $d^1_{MES} = d1 + d2 + d^1_{asynch} + d^1_{ch}$. Le délai d^1_{asynch} est l'attente d'une requête, au maximum un cycle de scrutation du client noté D_{SCR}, et d^1_{ch} est l'attente de la ressource processeur du serveur qui peut être en train de traiter d'autres requêtes. Concernant le client, le temps entre l'arrivée d'une réponse sur le coupleur et sa prise en compte par le programme de commande peut être décomposé comme $d^1_{SCR} = d4 + d^1_{asynch} + d^1_{ch}$. Le délai d^1_{asynch}, qui est le temps entre $d4$ et la flèche 4, représente l'attente de la prochaine phase de lecture des entrées par le programme de commande. Ce temps sera donc compris entre zéro et une période d'exécution du programme notée D_{CTR}. Le délai d^1_{ch}, durée entre la flèche 3 et $d4$, est l'attente de la libération du processeur, potentiellement occupé à traiter d'autres trames. La décomposition du temps entre l'écriture des sorties par le programme (flèche 5) et l'envoi par le coupleur (flèche 6) peut être décrit par l'équation $d^2_{SCR} = d5 + d^2_{asynch} + d^2_{ch}$. Ici, d^2_{asynch} représente l'attente du prochain envoi de requête vers le module lié à la sortie O1. Ce temps sera donc compris entre zéro et un cycle de scrutation D_{SCR}. Sur la figure 2, la somme $d^2_{asynch} + d^2_{ch}$ correspond au temps entre la flèche 5 et $d5$. Pour ce qui est du délai dû au contrôleur d_{CTR}, il suffit de prendre en compte la période d'exécution du programme, d'où $d_{CTR} = D_{CTR}$. Il est possible de faire de même pour chaque commutateur du réseau Ethernet. Pour des raisons de lisibilité, la figure 2 présente directement le délai résultant de la traversée de l'ensemble du réseau, par exemple entre 2 et 3 $d_{RES} = d3$. Si tous ces délais sont déterminés, alors le délai de réactivité en est la somme, c'est-à-dire

$$dr = d^1_{MES} + d^1_{RES} + d^1_{SCR} + D_{CTR} + d^2_{SCR} + d^2_{RES} + d^2_{MES}$$

avec les termes de type d^1_{XYZ} (respectivement d^2_{XYZ}) qui représentent les délais induit par le composant XYZ lors de la communication du processus vers le contrôleur (respectivement du contrôleur vers le processus).

III. Modèle classique : sommation des délais élémentaires

La section précédente présente le délai de réactivité comme une somme de délais induits par chacun des composants. Ces délais sont eux même décomposables en délais intrinsèques, caractéristiques des composants, et extrinsèques, dus à des asynchronismes et des partages de ressources. La méthode classique présentée prend en compte les caractéristiques des composants et les asynchronismes. Elle introduit deux hypothèses, la première est la charge nulle des composants et la deuxième est l'indépendance des sources de délai. Cette dernière hypothèse se traduit par un délai de réactivité minimum qui est la somme des délais élémentaires minimum, c'est-à-dire en considérant une synchronisation parfaite, $d_{asynch} = 0$. De même, le délai de réactivité maximum est calculé comme la somme des délais élémentaires maximum, c'est-à-dire en considérant un asynchronisme maximal. Cependant cette méthode ne tient pas compte du partage de ressources, donc dans le calcul du délai de réactivité la composante d_{ch} est toujours nulle, quel que soit le composant impliqué. Les équations suivantes récapitulent les délais élémentaires minimum, avec $d_{asynch} = 0$ et $d_{ch} = 0$:

$$\begin{aligned}
d^1_{MESmin} &= d1 + d2 \ ; \ d^2_{MESmin} = d7 + d8 \\
d^1_{RESmin} &= d^2_{RESmin} = d3 \\
d^1_{SCRmin} &= d4 \ ; \ d^2_{SCRmin} = d5 \\
d_{CTRmin} &= D_{CTR}
\end{aligned}$$

Puis suivent les délais élémentaires maximum, avec d_{asynch} au maximum et $d_{ch} = 0$:

$$\begin{aligned}
d^1_{MESmax} &= d1 + d2 + D_{SCR} \ ; \ d^2_{MESmax} = d7 + d8 \\
d^1_{RESmax} &= d^2_{RESmax} = d3 \\
d^1_{SCRmax} &= d4 + D_{CTR} \ ; \ d^2_{SCRmax} = d5 + D_{SCR} \\
d_{CTRmax} &= D_{CTR}
\end{aligned}$$

En effet, l'asynchronisme maximal sur le module d'E/S est donné par $d_{asynch} = D_{SCR}$, puis pour les scrutations d'entrées et sorties, par $d_{asynch} = D_{CTR}$ pour l'attente de la lecture des entrées, et $d_{asynch} = D_{SCR}$ pour l'attente de l'envoi d'une requête. Au final, les valeurs maximale et minimale du délai s'expriment ainsi :

$$\begin{aligned}
dr_{min} &= d1 + d2 + 2 * d3 + d4 + D_{CTR} + d5 + d7 + d8 \\
dr_{max} &= dr_{min} + D_{CTR} + 2 * D_{SCR}
\end{aligned}$$

Cette méthode permet donc de calculer les valeurs extrêmes du délai de réactivité en connaissant uniquement les données intrinsèques aux composants (délai de filtrage et de traitement des trames dans les modules d'E/S, délai de traversée d'un commutateur à vide, délai de traitement des trames dans les coupleurs, cycle de scrutation et période du programme du contrôleur) déterminées à partir de documentations constructeur ou d'expérimentations.

IV. Modèle Dynamique : Réseau de Petri Hiérarchique Coloré Temporisé

Pour prendre en compte la charge de l'architecture et les phénomènes de synchronisation qui s'y produisent, nous proposons un modèle dynamique simulable. Le formalisme retenu est celui des réseaux de Petri hiérarchiques colorés temporisés (RdPHCT) défini par Jensen ([8]) et supporté par le logiciel de simulation CPNTools.

A. Structure du modèle global

Dans un RdP hiérarchique, une transition est une abstraction du comportement du niveau inférieur. On utilise ce pouvoir de modélisation pour associer une transition à chaque classe de composant matériel de l'architecture. Ainsi au plus haut niveau de représentation du modèle (fig. 4) on retrouve une transition pour décrire le comportement des processeurs des contrôleurs CPU, des coupleurs

de communication Ethernet des contrôleurs Eth_board, des modules d'E/S IO_module, des commutateurs Switch, et une émulation du processus Process pour générer un scénario de sollicitation de l'architecture de commande.
Entre ces transitions, les places Process_event, Bus_exchange

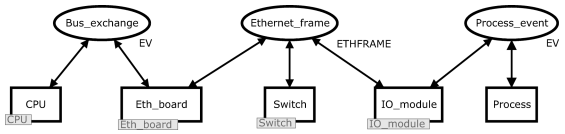
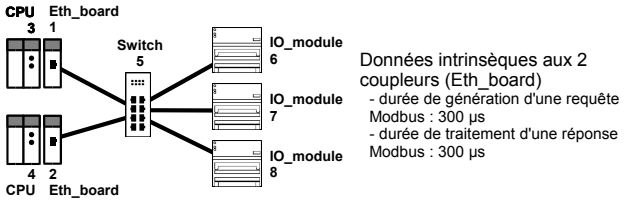


Fig. 4. Vue globale du modèle

et Ethernet_frame modélisent les échanges de données par des échanges de jetons. Les jetons de type EV (dans les places Bus_exchange et Process_event) représentent des événements, soit générés par le processus à destination d'un contrôleur, soit générés par un contrôleur à destination du processus. Les jetons de type ETHFRAME (dans la place Ethernet_frame) modélisent les requêtes Modbus encapsulées dans des trames Ethernet circulant dans le réseau commuté.
Pour des raisons pratiques il n'est pas possible de détailler l'ensemble du modèle dans ce papier. Nous avons donc choisi de nous limiter à une classe de composants représentatifs de ce type d'architectures : les coupleurs Ethernet des contrôleurs (transition Eth_board) avec leur client Modbus.
Il est à noter que le modèle global a été validé expérimentalement [9].

B. Modélisation du comportement des coupleurs Ethernet

Pour faciliter les explications du comportement des coupleurs, un exemple support est retenu. Il est composé de deux contrôleurs qui scrutent respectivement trois et deux modules d'E/S (fig. 5). Chaque composant est identifié par un entier unique dans l'architecture. Par exemple, le coupleur de communication (Eth_board) du contrôleur en haut et à gauche de la figure 5 est identifié par le numéro 1.



Données de configuration de l'architecture

Coupleur (Eth_board)	Liste des modules scrutés (IO_module)	Durée mini du cycle de scrutation
1	6, 7, 8	5 ms
2	7, 8	10 ms

Fig. 5. Exemple support

B.1 Structure du modèle du coupleur Ethernet

Dans le modèle global (fig. 4) les interfaces de la transition hiérarchique Eth_board sont les deux places Bus_exchange et Ethernet_frame. On retrouve ces deux places dans le modèle détaillé de la transition Eth_board présenté figure 6. Ce modèle inclus également une transition hiérarchique Ethb_TCP_IP_layers qui modélise le comportement de la pile TCP/IP et le média Ethernet des coupleurs de communications.
La place ETHB_CPU représente la disponibilité de la ressource CPU de chaque coupleur. Elle accueille donc des jetons de type ETHB (des entiers qui identifient les coupleurs) et son marquage initial, composé des deux jetons 1 et 2 en un exemplaire chacun (1'1++1'2), indique que les CPU des coupleurs Ethernet 1 et 2 sont disponibles.
La succession des places et transitions T1, P1, T2 et P2 décrit le regroupement et l'encapsulation dans des requêtes Modbus des événements que le contrôleur émet vers le processus. De manière symétrique la succession des places et transitions P3, T3, P4 et T4 décrit que les événements émis par le processus sont extraits des réponses Modbus puis, mis à la disposition du moniteur d'exécution du contrôleur via le bus d'échange.
Les places et transitions P8, T6, P7, P5, T5 et P6 (en gris dans le modèle) assurent le séquencement temporel des requêtes Modbus qui scrutent les modules d'E/S.

B.2 Modélisation de l'asynchronisme avec le moniteur d'exécution

Indépendamment du cycle de scrutation des modules d'E/S, la transition T1 regroupe au fur et à mesure de leur arrivée les événements ev dans des listes dédiées à chaque paire émetteur (un coupleur Ethernet ethb) destinataire (un module d'E/S iom). Le type de jeton dans P1 est ETHBxIOMxEVL, et représente le produit cartésien des ensembles ETHB (type qui identifie les coupleurs), IOM (type qui identifie les modules) et EVL (liste ordonnée de jetons ev). Au franchissement de T1 un événement ev est associé au triplet (ethb,iom,evl) correspondant grâce à la garde de transition non explicitée guardT1. Le poids (ethb,iom,ev : :evl) de l'arc T1->P1 indique que le jeton (ethb,iom,evl) extrait de P1 y est replacé en ajoutant à la liste evl l'événement ev issu de la place Bus_echange grâce à l'opérateur " : :" (par exemple ev3 : :[ev2,ev1] devient la liste [ev3,ev2,ev1]). Le marquage initial de P1 comprend 5 jetons correspondant à chaque paire utile coupleur-module avec une liste vide d'événements ([ ]). De façon symétrique T4 extrait de P4 un jeton (ethb,iom,ev : :evl) pour envoyer vers le moniteur d'exécution l'événement ev et replacer dans P4 un jeton (ethb,iom,evl) pour lequel la liste d'événement est diminuée d'un événement.

B.3 Modélisation du traitement des requêtes Modbus

Lorsque le moment est venu (ce moment est déterminé par la place P6 qui sera décrite ultérieurement), la transition T2 prélève dans P1 la liste des événements evl à transmettre du coupleur ethb vers le module iom et construit une requête Modbus modélisée par le triplet (ethb,iom,evl), c'est-à-dire (client, serveur, liste d'événements). Cette requête est empilée dans une liste FIFO placée dans P2 qui contient des jetons de type couple : (coupleur, liste de requêtes). Dans la transition hiérarchique qui gère la pile TCP/IP, l'opérateur " : :" est utilisé pour dépiler par la gauche de la liste les requêtes Modbus. Il faut donc empiler par la droite les requêtes au niveau de T2 pour obtenir la stratégie FIFO désirée. Cela est obtenu par l'opérateur de concaténation de listes "^^". Ainsi l'expression mbfl^^[(ethb,iom,evl)] concatène à droite de la liste

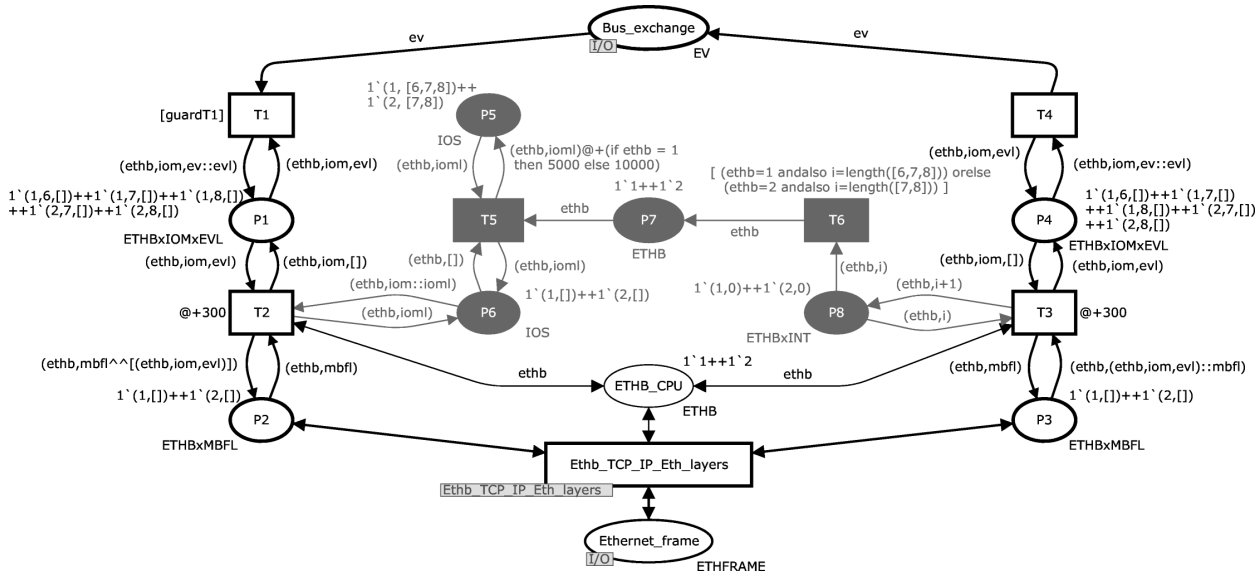


Fig. 6. Modèle du module de communication Eth_board

des requêtes mbfl, une liste composée d'une seule requête. On remarquera que la construction d'une requête Modbus (modélisée par T2) nécessite la ressource CPU (place ETHB_CPU) et dure 300μs soit 300 unités de temps du modèle (@+300). De façon symétrique, des réponses Modbus (ethb,iom,evl) sont organisées en listes dans P3. T3 extrait la liste les événements evl qu'elle empile dans P4.

B.4 Modélisation du cycle de scrutation

A chaque début de cycle de scrutation d'un coupleur client (Tir de T5), les requêtes Modbus à destination de chaque module serveur sont toutes envoyées à la suite les unes des autres. Les réponses reviennent au fur et à mesure et dans un ordre dépendant des disponibilités des ressources. Lorsqu'elles sont toutes revenues (Tir de T6), un nouveau cycle de scrutation redémarre si la valeur de temps de cycle configurée est atteinte ou dépassée. La place P5 contient les couples de coupleurs Ethernet et de listes de module d'E/S à scruter. Le marquage initial décrit que le coupleur 1 scrute les modules 6, 7 et 8 (un jeton (1,[6,7,8])), et que le coupleur 2 scrute les modules 7 et 8 (un jeton (2,[7,8])). La place P7 contient les coupleurs dont le cycle courant est terminé. Le franchissement de T5 change la date associée au jeton de P5 impliqué dans le tir au moyen de l'expression placée sur l'arc T5->P5. Il s'agit d'une expression conditionnelle qui dépend du coupleur. Ainsi pour le coupleur 1, le jeton aura comme nouvelle date la date de franchissement de T5 augmentée de 5000 unités de temps soit 5ms, tandis que pour le coupleur 2, la nouvelle date du jeton sera la date de franchissement de T5 augmentée de 10000 unités de temps soit 10ms. La place P8 assure le comptage du nombre de réponses i pour chaque coupleur ethb. Lorsque le nombre de réponses i est égal à la taille de la liste des modules scrutés (par exemple i=lenght([7,8,9]) pour le coupleur 1), la garde de la transition P6 permet le franchissement de cette dernière et modélise ainsi la fin du cycle en cours.

V. Application

Pour comparer les deux approches (sommation de délais élémentaires et simulation d'un modèle dynamique), deux configurations d'une même architecture (Fig. 7) sont étudiées. Les périodes des différents composants ont été fixées avec une résolution d'une microseconde (période du contrôleur 1 dans la configuration 1 : 16.003 ms) afin d'éviter des synchronisations non justifiées durant la simulation.
Ces deux configurations (table I) sont caractéristiques des architectures étudiées. Dans la configuration 1, le contrôleur 1 joue le rôle de superviseur en scrutant toutes les entrées et sorties déportées à une large période (environ 120 ms). Dans la deuxième configuration, les trois contrôleurs scrutent chacun neuf modules d'E/S avec des périodes proches, ils ont cette fois tous un rôle de contrôleur.

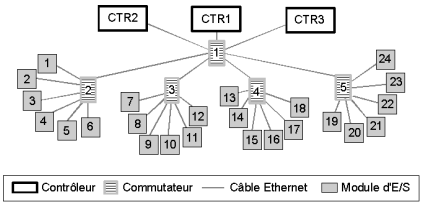


Fig. 7. Architecture test

Les résultats de la simulation de la configuration 1, pour 8000 délais simulés, sont présentés Fig. 8.

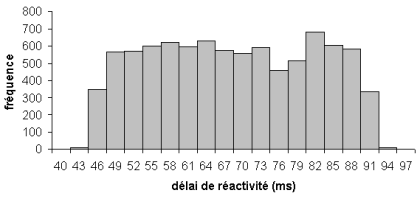


Fig. 8. Résultat de simulation

Dans la table II sont présentés les délais de réactivité minimum et maximum obtenus par chaque modèle pour les deux configurations tests.

	conf. 1	conf. 2
période contrôleur 1 (C1)	16.003	7.001
période contrôleur 2 (C2)	18.000	7.000
période contrôleur 3 (C3)	20.005	9.000
durée mini de scrutation de C1	120.007	13.000
durée mini de scrutation de C2	45.000	5.500
durée mini de scrutation de C3	45.003	17.001
délai commutateur	0.1	
délai modules d'E/S	0.72	
modules d'E/S scrutés par C1	1 à 24	1 à 9
modules d'E/S scrutés par C2	1 à 14	8 à 17
modules d'E/S scrutés par C3	11 à 24	16 à 24

TABLE I

Configuration temporelle en millisecondes

	conf. 1		conf. 2	
	min	max	min	max
modèle classique (1)	19.84	127.84	8.84	26.84
modèle RdPHCT (2)	44.77	92.06	11.85	34.97
erreur relative (%) par rapport au modèle (2)	55.68	38.86	25.42	23.25

TABLE II

Comparaison des délais minimum et maximum évalués par les deux méthodes (en millisecondes)

Rappelons que dans le cadre de la commande des SED, deux cas nous intéresses, la réponse à une commande cyclique ou à une alarme. Dans le cas d'une commande cyclique, le concepteur s'intéresse à la répartition des valeurs et à des caractéristiques statistiques de cette répartition. Tandis que dans le cas d'une alarme, le concepteur s'intéresse alors à sa valeur maximale.

Pour une commande cyclique, le modèle classique permet uniquement l'évaluation du minimum, du maximum et de la moyenne avec l'hypothèse d'une répartition uniforme des délais, alors que la simulation permet d'obtenir la distribution complète. Les éléments comparables sont les valeurs minimales, maximales et l'étendue des valeurs. Pour les deux configurations les écarts sont significatifs entre les deux modèles, jusqu'à plus de 55% pour le délai minimum de la configuration 1. En ce qui concerne l'étendue des valeurs, pour la configuration 1, elle est caractérisée par un facteur 6 entre les extremum en utilisant la méthode classique alors que ce facteur est de 2 si l'on utilise notre méthode. Ainsi, en utilisant le modèle classique, une configuration convenable pourrait être rejetée.

Pour une alarme, les deux modèles permettent le calcul d'une borne maximale. Cependant, d'une part, les résultats pour la configuration 1 donnent une différence de 39% entre les deux maxima. Cela pourrait conduire à refuser une architecture qui convient. D'autre part, la configuration 2 donne une différence de 23 % environ entre les maxima, mais cette fois, le délai maximum calculé par sommation ne majore pas celui déterminé par simulation. En effet, comme il avait été mentionné dans la section III, ce modèle ne permet pas d'avoir une borne maximum du délai de réactivité car il ne tient pas compte du partage de ressources. Dans ce cas, le modèle classique pourrait conduire à accepter une architecture alors qu'il ne répond pas aux spécifications.

Les hypothèses du modèle classique, indépendance des délais et non-partage des ressources, sont trop fortes et ne permettent pas de représenter le comportement du système. La simulation du modèle RdPHCT donne des résultats plus complets car ce modèle prend en compte les deux mécanismes négligés par les hypothèses du modèle classique. Que le délai de réactivité concerne une commande cyclique ou une réponse à une alarme, il convient donc de privilégier la modélisation et l'analyse par RdPHCT.

VI. Conclusion

Nous nous sommes intéressés ici à la réactivité des architectures de commande distribuées sur des réseaux Ethernet à protocoles ouverts de type client/serveur. Elle est caractérisée par le délai entre l'occurrence d'un événement "cause" venant du processus et l'occurrence de l'événement "conséquence" correspondant au niveau du processus, appelé délai de réactivité.

Nous avons proposé d'évaluer ce délai par simulation de modèles en RdPHCT, ce qui permet de prendre en compte les mécanismes de partage des ressources et de parallélisme de processus. La comparaison des résultats obtenus par simulation avec un calcul classique de sommation des délais a montré l'importance de ces deux mécanismes, la nécessité de les prendre en compte, et par conséquent l'intérêt de la modélisation du comportement dynamique par RdPHCT.

Deux perspectives se dessinent. D'un point de vue scientifique, nous travaillons actuellement au couplage de la simulation avec une méthode formelle, ce qui permettrait d'obtenir des bornes maximum et minimum strictes, en se basant sur [10]. D'un point de vue ingénierie, les modèles en RdPHCT sont délicats à manipuler et doivent être réservés à des experts. Pour cela, un outil d'assistance pour concepteur d'architecture, basé sur cette modélisation est en cours de développement.

Références

[1] Paolo Ferrari, Alessandra Flammini, Daniele Marioli, et Andrea Taroni. Experimental evaluation of PROFINET performances. *WFCS04*, Vienna, Austria, September 2004.

[2] Stefano Vitturi. DP-Ethernet : the Profibus DP protocol implemented on Ethernet. *Computer Communications*, 26 :1095–1104, 2003.

[3] Jurgen Jasperneite et Peter Neumann. How to guarantee real-time behavior using Ethernet. *INCOM04*, Salvador, Brasil, April 2004.

[4] Nicolas Krommenacker, Jean-Philippe Georges, Eric Rondeau, et Thierry Divoux. Designing, modelling and evaluating switched Ethernet networks in factory communication systems. *1st Workshop on RTLIA*, Vienna, Austria, June 2002.

[5] Kyung Chang Lee et Suk Lee. Performance evaluation of switched Ethernet for real-time industrial communications. *Computer Standards & Interfaces*, 24(5) :411–423, November 2002.

[6] Yeqiong Song. Time constrained communication over switched Ethernet. *FeT'2001*, Nancy, France, November 2001. IFAC, Elsevier Scientific.

[7] Belgacem Ben Hédia, Fabrice Jumel, et Jean-Philippe Babau. Qualité de service des pilotes d'équipements pour les systèmes d'acquisition de données. *MSR05*, pages 47–62, Grenoble, France, October 2005.

[8] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 1, Basic Concepts. Springer-Verlag, monographs in theoretical computer science edition, 2nd corrected printing 1997.

[9] Gaëlle Poulard, Bruno Denis, et Jean-Marc Faure. Modélisation par réseau de Petri coloré des architectures de commande distribuées sur réseau de terrain Ethernet et TCP/IP. *MOSIM04*, pages 405–412, Septembre 2004.

[10] Daniel Witsch, Birgit Vogel-Heuser, Jean-Marc Faure, et Gaëlle Marsal. Performance analysis of industrial Ethernet networks by means of timed model-checking. *INCOM'2006*, May 2006.