Modelisation et évaluation temporelle des architectures de commande en réseau

Boussad Addad, Saïd Amari

Modélisation et évaluation temporelle des architectures de commande en réseau

BOUSSAD ADDAD[1], SAÏD AMARI[1,2]

[1] Laboratoire Universitaire de Recherche en Production Automatisée, Lurpa
ENS- Cachan, 61 av du Président Wilson, 94235 Cachan Cedex, France

[2] Université ParisXIII
Place 8 mai 1954, St Denis, France

Prenom.Nom@lurpa.ens-cachan.fr

***Résumé* — Ce papier présente une nouvelle méthode d'évaluation du temps de réponse dans les architectures de commande sur réseau Ethernet commuté. Elle se base sur la modélisation du système avec tous ses constituants à l'aide des graphes d'événements temporisés et sa représentation d'état en équations linéaires dans le formalisme de l'algèbre Max-plus. La résolution de ces équations et l'introduction de variables secondaires, nous a permis de développer un algorithme de calcul du temps en question. Avec une analyse plus poussée des équations, nous avons abouti à des formules analytiques de calcul direct du temps de réactivité en fonction des caractéristiques temporelles de l'architecture. Les bornes minimale et maximale du temps de réponse sont au cœur de l'analyse. Des simulations de l'algorithme et des mesures expérimentales sur une plate-forme de laboratoire ont été utilisées pour la validation des résultats et la mise en évidence de l'intérêt de cette nouvelle méthode.**

***Mots clés* — Systèmes commandés en réseau, algèbre Max-Plus, graphes d'événements temporisés, temps de réponse.**

I. INTRODUCTION

Le media Ethernet en tant que réseau de terrain est une idée séduisante. Cela apporte en effet beaucoup d'avantages tant sur le plan économique avec de bas prix grâce à une production à grande échelle des composants, que technique avec des débits allant actuellement jusqu'à des dizaines de Gb/s. De nombreux constructeurs se sont alors tournés vers cette alternative mais malheureusement en définissant des protocoles spécifiques pas toujours compatibles avec les réseaux standards. Les causes sont notamment dues aux contraintes temps réel de certain systèmes de commande et donc la nécessité de définir de nouveaux protocoles avec des méthodes de gestion de l'accès au medium Ethernet spécifiques. Les caractéristiques temporelles de ces systèmes sont bien maitrisées et connues. Leur temps de réponse peut être inferieur à la milliseconde avec une gigue de quelques microsecondes. Cependant, la plupart des systèmes de commande ne nécessitent pas des contraintes aussi rigoureuses et des protocoles simples sont souvent suffisants. En revanche, des études qui s'intéressent à ces protocoles sont rares et par conséquent la connaissance de leurs caractéristiques reste très approximative. Dans notre étude justement, nous nous intéressons aux réseaux Ethernet commutés qui utilisent des protocoles ouverts et standards comme Modbus TCP/IP de type client/serveur [1]. C'est un protocole simple mais difficilement évaluable. A cause de la présence des commutateurs et du partage des ressources d'une part, et aux asynchronismes entres les composants d'autre part. Ce qui rend difficile la prédiction du temps de traversée d'un nœud du réseau par une trame. Différents travaux sont menés afin d'évaluer ces temps en adoptant la politique du pire cas, calcul réseau [2] ou la simulation [3]. Cependant, comme la majorité des études, ces travaux se focalisent sur une partie de l'architecture à savoir le réseau et ignore le reste comme la partie contrôleur et module déportés d'E/S. En réalité, l'asynchronisme des modules du PLC (programmable logic controller) et le partage des ressources, induisent des retards considérables dans l'architecture et doivent être pris en compte. A notre connaissance des méthodes qui prennent en considération tous les constituants de l'architecture sont essentiellement basés sur la simulation. Une méthode basée sur un modèle avec automates communicants et le model-checking [4]-[5]-[6], permet d'avoir une réponse de type booléen quant à la possibilité de dépassement ou non d'une borne de temps. Cette méthode présente l'inconvénient de ne pas donner une distribution des temps de réactivité et aussi l'explosion du nombre d'états dans le modèle. Une autre méthode se base sur un modèle en Réseaux de Pétri colorés hiérarchiques temporisés et leur simulation avec l'outil CPNTools [7]-[8]-[9]. Cette méthode permet une bonne évaluation des retards mais est onéreuse en temps de calcul et les cas critiques ne sont pas surement atteints.

Dans ce papier, nous proposons une méthode analytique de calcul des bornes et de la distribution du temps de réactivité. Plusieurs configurations d'architectures de commande sont étudiées. Ce papier est organisé comme suit. Quelques notions sur l'algèbre Max-plus et les GETs (graphes d'événements temporisés) sont rappelées en section II. Ensuite, la modélisation de l'architecture en utilisant ces outils est présentée en section III. La section IV est alors consacrée à l'explication de notre méthode d'évaluation et les résultats importants qui en découlent. En section V, nous les comparons à des mesures prises sur la plate forme brevetée PRISME [10] du laboratoire LURPA, et ainsi nous vérifierons la validité des formules analytiques obtenues. Finalement, une conclusion et quelques perspectives sont données en section VI.

II. RAPPELS

A. Algèbre Max-plus

Un ensemble D muni d'une loi interne notée $\oplus$ est un monoïde si cette loi est associative et admet un élément neutre ε tel que, $\forall a \in D, a \oplus \varepsilon = \varepsilon \oplus a = a$. Un semi anneau est un monoïde muni d'une seconde loi interne notée $\otimes$ et qui est

associative, distributive par rapport à $\oplus$ et admet un élément neutre noté e et absorbant pour $\oplus$, $\forall a \in D, a \otimes e = e \otimes a = e$. Un dioïde est un semi anneau avec une addition idempotente, $\forall a \in D, a \oplus a = a$. Il est dit commutatif si $\otimes$ est commutative.

Dans le formalisme max-plus, on considère l'ensemble $\overline{\mathbb{R}}_{\max} = \mathbb{R} \cup \{-\infty\} \cup \{+\infty\}$. La structure $(\overline{\mathbb{R}}_{\max}, \max, +)$ est un dioïde commutatif avec comme loi $\oplus$ le maximum usuel avec son élément neutre $\varepsilon = -\infty$ et l'addition classique pour la loi $\otimes$ et $e = 0$ comme élément neutre.

Cette algèbre est étendue pour le cas vectoriel et matriciel. Ainsi, pour $n \in \mathbb{N}$ et $v, w \in \overline{\mathbb{R}}_{\max}^{n}$, le vecteur $v \oplus w$ a les composantes $v_i \oplus w_i = \max(v_i, w_i)$ pour $i = 1$ à n. Pour $p, q \in \mathbb{N}$, $A \in \overline{\mathbb{R}}_{\max}^{p \times n}$ et $B \in \overline{\mathbb{R}}_{\max}^{n \times q}$, la multiplication matricielle dans $\overline{\mathbb{R}}_{\max}$ est définie par $A \otimes B$ ou $A.B$ où : $(A.B)_{ij} = \bigoplus_{k=1}^{n} (A_{ik} \otimes B_{kj}) = \max_k (A_{ik} + B_{kj})$.

Pour une matrice carrée $M \in \overline{\mathbb{R}}_{\max}^{n \times n}$, on définit l'étoile de Kleene comme $M^* = \bigoplus_{i \in \mathbb{N}} M^i$ où M^0 est la matrice unité avec uniquement des e en diagonale et des ε ailleurs. Ainsi pour $v \in \overline{\mathbb{R}}_{\max}^{n}$, $x = M^*.v$ est la solution maximale de l'inéquation $x \geq M.x \oplus v$ et l'équation $x = M.x \oplus v$ [11].

B. Graphes d'événements temporisés (GETs)

Un graphe d'événements est un réseau de Petri ordinaire où toutes les places ont au plus une transition en amont et en aval. Un graphe d'événements est dit GET ou temporisé si des temporisations sont associées aux transitions ou aux places. Dans la suite de notre étude, nous adopterons cette dernière, i.e. des places temporisées. On note n le nombre de transitions avec au moins une place en amont et m le nombre de transitions source *tu*. L'unique place reliant deux transition t_j et t_i est notée p_{ij} et sa temporisation τ_{ij}. Dans notre étude, nous associerons simplement la temporisation τ_i à la place p_i. Pour l'étude du comportement dynamique du graphe, on associe à chaque transition t_i la date de son franchissement pour la $k^{ème}$ fois. Elle est notée $u_s(k)$ pour les transitions source et $\theta_i(k)$ pour les autres.

Exemple (Fig. 1):

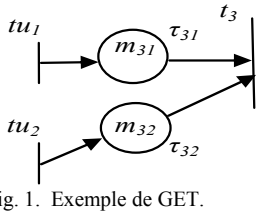

Fig. 1. Exemple de GET.

Pour le graphe de l'exemple, nous avons l'équation suivante : $\theta_3(k) = \max(\tau_{31} + u_1(k - m_{31}), \tau_{32} + u_2(k - m_{32}))$, qui est une équation linéaire dans l'algèbre Max-plus : $\theta_3(k) = \tau_{31} \otimes u_1(k - m_{31}) \oplus \tau_{32} \otimes u_2(k - m_{32})$.

En général, le comportement d'un graphe d'événements peut être décrit par l'équation linéaire explicite :

$$\theta(k) = \bigoplus_{m>0} (A_0^*.A_m.\theta(k-m) \oplus A_0^*.B_m.u(k-m)), \qquad (1)$$

où les vecteurs $\theta(k)$ et $u(k)$ comportent les temps de franchissement des n et m transitions du système pour la $k^{ème}$ fois. La matrice A_m appartient au dioïde $\overline{\mathbb{R}}_{\max}^{n \times n}$ et dont l'élément $A_{m,ij}$ représente la temporisation τ_{ij} associée à la place p_{ij} si elle existe et l'élément neutre ε sinon. Aussi pour $B_m \in \overline{\mathbb{R}}_{\max}^{n \times m}$, B_m représente les retards dans les places en aval des transitions sources. A_0^* est l'étoile de Kleene (définie précédemment) de la matrice A_0. D'une manière analogue à la représentation d'état des systèmes linéaires classiques, l'équation explicite peut être ramenée à une forme de représentation d'état en remplaçant toutes les places avec des marquages $m_{ij} > 1$ avec m_{ij} places et $(m_{ij} - 1)$ transitions intermédiaires. Ainsi, on aboutit à un système étendu dont le vecteur d'état $x(k)$ appartient à $\overline{\mathbb{R}}_{\max}^{N}$ où $N = n + n'$ et n' le nombre de transitions intermédiaires ajoutées. Le système étendu peut être décrit alors par l'équation :

$$x(k) = \hat{A}_0.x(k) \oplus \hat{A}_1.x(k-1) \oplus \hat{B}.u(k), \qquad (2)$$

ou même

$$x(k) = A.x(k-1) \oplus B.u(k), \qquad (3)$$

où $A = \hat{A}_0^*.\hat{A}_1$ et $B = \hat{A}_0^*.\hat{B}$. Ce qui permet de conclure que la dynamique du système ne dépend que des transitions source et des conditions initiales [12].

III. MODELISATION DE L'ARCHITECTURE

Dans cet article, l'architecture de commande fonctionne suivant le protocole client/serveur où le contrôleur est le client et les modules d'E/S sont les serveurs (Fig. 2). Le PLC retenu dispose de deux modules, le CPU pour le traitement programme et le coupleur réseau Ethernet ETHb pour l'envoi des requêtes et la scrutation des E/S ou RIOM (remote input output module). Ils fonctionnent de manière cyclique mais asynchrone, ce qui rend l'évaluation des performances difficile et pose un problème quant aux valeurs des périodes du CPU et l'ETHb à fixer pour atteindre un niveau de performance désiré. Le processeur de traitement exécute périodiquement les taches : lecture, traitement et écriture avec comme seule contrainte, la période de cycle à ne pas dépasser. Le coupleur réseau ETHb de son coté, indépendamment du reste, copie puis envoie des requêtes aux modules E/S et attend que les réponses reviennent. Si avant de terminer le temps de cycle toutes les réponses sont arrivées, il se met en attente et sinon il recommence directement un nouveau cycle de scrutation.

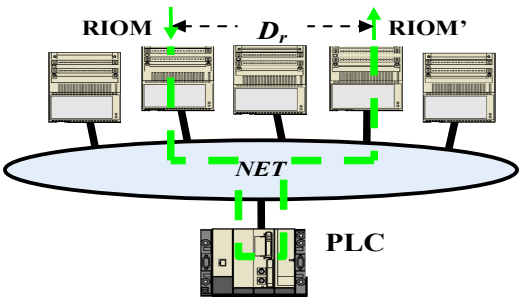

Fig. 2. Schéma de l'architecture de commande.

Pour mieux aborder le problème de détermination du temps de réponse, nous considérons le cas relativement simple d'un système avec seul PLC et un seul module RIOM (Fig. 3) puis nous étendrons l'étude pour une architecture plus générale.

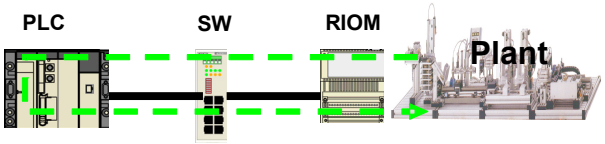

Fig. 3. Architecture de commande : cas 1.

A. CAS 1 : un seul PLC et un seul RIOM

Le temps de réactivité est défini comme le délai entre l'occurrence d'un événement sur l'entrée du RIOM et l'arrivée de sa conséquence issue du PLC sur le système commandé (Fig. 3). Deux cas peuvent être considérés dans une architecture de commande : ça peut être par exemple le temps entre le moment de détection d'un danger et le déclenchement d'une alarme. Dans ce cas, la borne maximale du temps de réaction est très importante à déterminer. Mais pour une commande de type cyclique comme un réglage de vitesse, c'est plutôt la distribution de ce temps qui est nécessaire à évaluer. Dans notre étude, on va s'intéresser au cas général où il s'agit de déterminer ce temps pour tout événement sachant son moment d'occurrence et l'état du système.

Compte tenu du fonctionnement de l'architecture de commande décrit précédemment, on obtient le modèle en GETs de la figure (Fig. 4). Le modèle présente la particularité de comporter deux GETs principaux : celui du CPU et le reste de l'architecture. En réalité, ce modèle sert plutôt à décrire l'état des constituants indépendamment, en faisant abstraction sur le contenu des jetons (trames) en circulation dans les GETs. C'est un modèle très abstrait qui tient compte seulement de la couche applicative de l'architecture et ignore les autres couches comme TCP/IP. C'est en effet les caractéristiques temporelles du système qui nous intéressent et l'encapsulation d'un message dans une trame Ethernet par exemple pour être envoyée vers le RIOM, peut simplement être représentée par une seule place avec un délai égal au retard global induit par la phase de l'envoi. Le même principe est suivi pour toute étape dans notre système où ce genre de traitement a lieu. Toutefois, la fusion des équations des deux modules à l'aide de nouvelles variables est nécessaire pour pouvoir suivre la trace des trames dans l'architecture et déterminer le temps de leur parcours.

Les places p_1, p_2, p_3 et p_4 avec les temporisations τ_1, τ_2, τ_3 et τ_4 du CPU représentent respectivement les phases d'attente, de traitement programme pendant T_{CLC} (avec lecture écriture incluses), non disponibilité du CPU et enfin sa disponibilité. Ainsi, on voit le fonctionnement périodique du CPU avec un temps de cycle τ_3 noté T_{CPU}. De manière similaire, τ_{14} représente la période de scrutation T_{SCR} du coupleur et la place p_{14} la non disponibilité du module pendant au moins τ_{14}. Le début d'envoi de la requête se fait avec le franchissement de la transition t_4 et se termine avec t_5 (τ_6 ou T_{EM} étant le temps nécessaire pour l'envoi d'une trame). Un jeton dans p_{13} indique que la requête est envoyée et le coupleur se met en attente des réponses. Les places p_7 et p_{12} modélisent respectivement le retard aller T_{NETf} et retour T_{NETb} induits par le réseau. Cette séparation est possible car les liaisons considérées sont full duplex et il n'y a pas de risque de collision ou conflit. Pour éviter d'encombrer le modèle, τ_{12} inclut aussi le temps nécessaire pour la copie de la réponse du buffer d'entrée du coupleur vers la mémoire partagée avec le CPU (ce temps étant aussi négligeable devant les autres délais de l'architecture). Les places p_8, p_9, p_{10} et p_{11} représentent le module E/S. En accord avec le protocole client serveur, le module reste en attente dans p_9 jusqu'à réception d'une requête dans son buffer d'entrée p_8. Il commence le traitement avec le franchissement de t_7 pendant un temps τ_{10} noté $T_{I/O}$ puis met le résultat dans son buffer de sortie p_{11}. Les transitions en gris modélisent la transition source (donnée en provenance du capteur) et sortie (donnée vers l'actionneur). Elles ne sont pas prises en considération car le système est non contraint et tant que le capteur est fonctionnel, il y a une valeur disponible à sa sortie.

IV. EVALUATION DU TEMPS DE REPONSE

En appliquant la méthode de la section II au modèle de l'architecture avec les conditions initiales de la (Fig. 4), on obtient les équations :

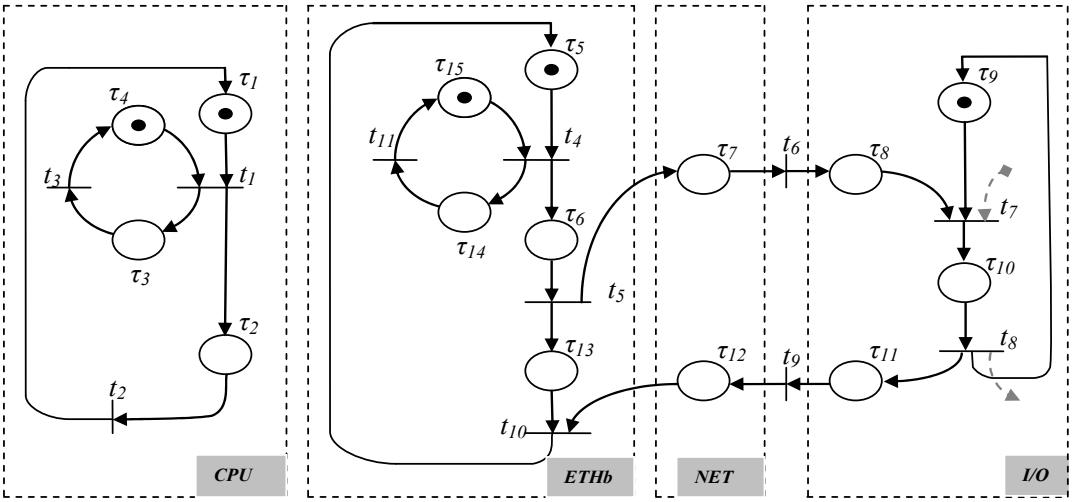

Fig. 4. Modèle en GETs de l'architecture de commande.

$$\begin{cases} \theta_1(k) = \max(\theta_2(k-1)+\tau_1, \theta_3(k-1)+\tau_4) \\ \theta_2(k) = \theta_1(k)+\tau_2 \\ \theta_3(k) = \theta_1(k)+\tau_3 \end{cases} \quad (5)$$

$$\begin{cases} \theta_4(l) = \max(\theta_{10}(l-1)+\tau_5, \theta_{11}(l-1)+\tau_{15}) \\ \theta_5(l) = \theta_4(l)+\tau_6 \\ \theta_6(l) = \theta_5(l)+\tau_7 \\ \theta_7(l) = \max(\theta_6(l)+\tau_8, \theta_8(l-1)+\tau_9) \\ \theta_8(l) = \theta_7(l)+\tau_{10} \\ \theta_9(l) = \theta_8(l)+\tau_{11} \\ \theta_{10}(l) = \max(\theta_5(l)+\tau_{13}, \theta_9(l)+\tau_{12}) \\ \theta_{11}(l) = \theta_4(l)+\tau_{14} \end{cases} \quad (6)$$

A. Algorithme de calcul du temps de réponse

Les systèmes (5) et (6) sont linéaires dans l'algèbre Max-plus et peuvent se mettre sous la forme (3). Nous avons affecté des indices différents pour les deux GETs (*k* et *l*) pour représenter le fonctionnement asynchrone des deux modules du PLC. C'est la difficulté supplémentaire et principale de notre étude.

La résolution des équations dans $\overline{\mathbb{R}}_{\max}$ donne :

$$\begin{cases} \theta_1(k) = (k-1)\cdot T_{CPU} \\ \theta_2(k) = (k-1)\cdot T_{CPU} + T_{CLC} \\ \theta_3(k) = k\cdot T_{CPU} \end{cases} \quad (7)$$

$$\begin{cases} \theta_4(l) = (l-1)\cdot T_{SCR} \\ \theta_5(l) = \theta_4(l)+\tau_6 \\ \theta_6(l) = \theta_5(l)+\tau_7 \\ \theta_7(l) = \max(\theta_6(l)+\tau_8, \theta_8(l-1)+\tau_9) \\ \theta_8(l) = \theta_7(l)+\tau_{10} \\ \theta_9(l) = \theta_8(l)+\tau_{11} \\ \theta_{10}(l) = \max(\theta_5(l)+\tau_{13}, \theta_9(l)+\tau_{12}) \\ \theta_{11}(l) = \theta_4(l)+\tau_{14} \end{cases} \quad (8)$$

Parmi ces solutions, seules les équations représentant les événements suivants nous intéressent à ce stade de l'étude :

-- Lecture et début de calcul dans le CPU (θ_1)

-- Fin de calcul dans le CPU et écriture du résultat (θ_2)

-- Début de scrutation et envoi d'une requête (θ_4)

-- Arrivée de la réponse au coupleur Ethernet (θ_{10})

En posant le temps de retour de la réponse : $T_r = \tau_6 + \tau_7 + \tau_8 + \tau_{10} + \tau_{11} + \tau_{12}$, nous obtenons les équations suivantes :

$$\begin{cases} \theta_1(k) = (k-1)\cdot T_{CPU} \\ \theta_2(k) = (k-1)\cdot T_{CPU} + T_{CLC} \end{cases} \quad (9)$$

$$\begin{cases} \theta_4(l) = (l-1)\cdot T_{SCR} \\ \theta_{10}(l) = (l-1)\cdot T_{SCR} + T_r \end{cases} \quad (10)$$

A la $l^{ème}$ scrutation, la réponse arrive à l'instant $\theta_{10}(l)$ et pour qu'elle soit prise en considération, il faut attendre le m^{eme} (mais prochain immédiat par rapport à $\theta_{10}(l)$) début de traitement dans le CPU. L'indice m doit alors vérifier la condition: $m = \underset{i/\theta_1(i)>\theta_{10}(l)}{Arg\min}(\theta_1(i)-\theta_{10}(l))$. Nous introduisons alors une nouvelle variable $\hat{\theta}_1$ telle que : $\hat{\theta}_1(l) = \theta_2(m) = \theta_1(m) + T_{CLC}$.

Une fois le résultat de calcul obtenu et écrit dans la mémoire du coupleur, il faut attendre le prochain début cycle de scrutation pour qu'il soit pris en considération et envoyé vers le module E /S. On introduit alors une deuxième variable $\hat{\theta}_2$ telle que $\hat{\theta}_2(l) = \theta_4(n)$ et $n = \underset{j/\theta_4(j)>\theta_1(l)}{Arg\min}(\theta_4(j)-\hat{\theta}_1(l))$. Le temps d'arrivée du résultat vers le module E/S est alors $\theta_8(n)$ noté $\hat{\theta}_f(l)$. Ainsi pour le p^{eme} événement généré à l'instant $v(p)$ et pris en compte à la $l^{ème}$ scrutation, le temps de réponse de l'architecture relatif à cet événement est :

$$D_r = \theta_f(l) - v(p) \quad (11)$$

Ce délai est minimal si la donnée est prise en compte juste après son arrivée du capteur. Ce temps minimal est alors :

$$D_{MIN} = \theta_f(l) - \theta_7(l) + d_f$$

où d_f est le retard dû au filtrage des données dans le capteur.

Le délai maximal est obtenu si la donnée est arrivée juste après le début de traitement dans le RIOM lors de la scrutation courante. Il est donné par :

$$D_{MAX} = \theta_f(l) - \theta_7(l-1) + d_f \quad (12)$$

Le calcul de ces délais est toujours valable et c'est le cas si le renouvellement de la sortie du capteur se fait avec une fréquence inférieure à celle de scrutation. Dans le cas contraire, il y'a écrasement de données et elles ne sont utilisées durant aucune étape dans l'architecture. Un cas inutile à considérer car les bornes ne dépendent pas de la fréquence de renouvellement de la sortie du capteur.

Ainsi, nous avons un algorithme de calcul des délais de réactivité de l'architecture considérée. Il est rapide et facile à implémenter connaissant les caractéristiques du système. Des simulations de cet algorithme nous permettront de vérifier les formules de calcul de délais obtenues par la suite.

- *Lemme*

Le système est composé de deux processus cycliques mais asynchrones. Malgré cette caractéristique, le système reste périodique dans sa globalité avec une période T_{CR} telle que : $T_{CR} = k_1 \cdot T_{SCR} = k_2 \cdot T_{CPU}$ où $k_1, k_2 \in \mathbb{N}$ existent toujours.

Preuve : posons : $T_{SCR} = r \cdot T_{CPU}$ avec $r = \lfloor r \rfloor + \omega$, $\lfloor r \rfloor$ étant la partie entière de r et ω la partie fractionnaire. Pour $\omega = n_1/n_2$, il suffit de prendre : $k_1 = n_2$ et $k_2 = n_2 \cdot \lfloor r \rfloor + n_1$ pour trouver : $T_{CR} = \left\lfloor n_2 \cdot \lfloor r \rfloor + n_1 \right\rfloor \cdot T_{CPU}$ (13)

Cette période est minimale pour n_1 et n_2 premiers entre eux.

Donc l'algorithme précèdent est formel et balaie tous les états possibles du système pour un temps de simulation dépassant la période critique T_{CR}.

B. Calcul analytique du temps de réponse

Nous allons utiliser les équations (9) et (10) et le principe de l'algorithme précédent.

Posons : $T_r = \alpha \cdot T_{CPU} + \tau_r$, $T_{CLC} = \beta \cdot T_{CPU}$ où $\beta < 1$. α, τ_r sont respectivement le rapport et le reste de la division euclidienne de T_r par T_{CPU}.

Pour des raisons de complexité avérée par la suite, nous

commençons l'étude par le cas $r \in \mathbb{N}$.

- $r \in \mathbb{N}$

A la $l^{éme}$ scrutation, nous avons :

$$\theta_{10}(l) = (l-1)\cdot r \cdot T_{CPU} + \alpha \cdot T_{CPU} + \tau_r \quad (14)$$

Pour $k-1 = (l-1)\cdot r + \alpha + 1$ nous obtenons :

$$\theta_1(k) = \theta_{10}(l) + T_{CPU} - \tau_r \quad (15)$$

Comme $0 < T_{CPU} - \tau_r < T_{CPU}$ alors

$$\hat{\theta}_1(l) = \theta_1(k) + T_{CLC} = \left[1 + \alpha + \beta + (l-1)\cdot r\right]\cdot T_{CPU} \quad (16)$$

Nous avons aussi $\theta_4(n) = (n-1)\cdot r \cdot T_{CPU}$ et pour $(n = l+1)$ alors : $\theta_4(n) = l \cdot r \cdot T_{CPU}$ (17)

ou encore : $\theta_4(n) = \hat{\theta}_1(k) + \left[r - (1+\alpha+\beta)\right]\cdot T_{CPU}$ (18)

Ainsi sous la condition C_1 : $r > (1+\alpha+\beta)$ et c'est le cas optimal, nous avons : $\hat{\theta}_2(l) = \theta_4(n) = \theta_4(l+1)$ et donc :

$$\hat{\theta}_f(l) = \theta_8(l+1) \quad (18)$$

$$D_{MIN} = \theta_8(l+1) - \theta_7(l) + d_f \quad (19)$$

$$D_{MAX} = \theta_8(l+1) - \theta_7(l-1) + d_f \quad (20)$$

Finalement :

$$\begin{cases} D_{MIN} = T_{SCR} + T_{NETf} - T_{NETb} + T_{I/O} + d_f \\ D_{MAX} = 2T_{SCR} + T_{NETf} - T_{NETb} + T_{I/O} + d_f \\ D_r(p) = \theta_f(l) - v(p) \end{cases} \quad (21)$$

Ces résultats sont souvent valables car la fréquence de scrutation est très inférieure à la fréquence du CPU. Pour un cas plus général, nous obtenons les résultats suivants :

$$\begin{cases} D_{MIN} = q \cdot T_{SCR} + T_{NETf} - T_{NETb} + T_{I/O} + d_f \\ D_{MAX} = (q+1)\cdot T_{SCR} + T_{NETf} - T_{NETb} + T_{I/O} + d_f \\ D_r(p) = \theta_8(l+q) - v(p) \end{cases} \quad (22)$$

où $r \cdot q > (1+\alpha+\beta) > r\cdot(q-1)$.

- $r \in \mathbb{Q}^+$

Posons $\tau_r = \gamma \cdot T_{CPU}$ où $\gamma < 1$,

A la $l^{éme}$ scrutation, nous avons :

$$\theta_{10}(l) = (l-1)\cdot r \cdot T_{CPU} + (\alpha+\gamma)\cdot T_{CPU} \quad (23)$$

Soit i tel que : $i < \gamma + \omega\cdot(l-1) < (i+1)$ (*)

Pour $k-1 = (l-1)\cdot r + \alpha + 1 + i$ nous obtenons :

$$\theta_1(k) = \theta_{10}(l) + \left[i + 1 - (\gamma + \omega\cdot(l-1))\right]\cdot T_{CPU} \quad (24)$$

et comme $i < \gamma + \omega\cdot(l-1) < (i+1)$ alors :

$$\hat{\theta}_1(l) = \theta_{10}(l) + \left[i + 1 + \beta - (\gamma + \omega\cdot(l-1))\right]\cdot T_{CPU}. \quad (25)$$

Nous avons aussi :

$\theta_4(n) = (n-1)\cdot r \cdot T_{CPU}$ et pour $(n = l+1)$ alors : $\theta_4(n) = l \cdot r \cdot T_{CPU}$ ou encore :

$$\theta_4(n) = \hat{\theta}_1(k) + \left[r - \left[\alpha + \beta + i + 1 - \omega\cdot(l-1)\right]\right]\cdot T_{CPU} \quad (26)$$

De (*) nous déduisons: $\gamma < i + 1 + \omega\cdot(l-1) < 1 + \gamma$

Posons : $\Gamma_{l,i} = i + 1 + \omega\cdot(l-1)$

$$\Gamma_{MIN} = \min_{i\in\mathbb{N}, l\in\mathbb{N}}(\Gamma_{l,i})$$

$$\Gamma_{MAX} = \max_{i\in\mathbb{N}, l\in\mathbb{N}}(\Gamma_{l,i})$$

Ainsi, sous la condition C_2 : $r > (\alpha + \beta + \Gamma_{MAX})$ et c'est le cas optimal, nous avons :

$\hat{\theta}_2(l) = \theta_4(n) = \theta_4(l+1)$ et donc $\hat{\theta}_f(l) = \theta_8(l+1)$. Et ainsi nous obtenons les mêmes résultats que dans l'équation (21).

En général, si:

$r \cdot q_1 > (\Gamma_{MIN} + \alpha + \beta) > r\cdot(q_1 - 1)$

$r \cdot q_2 > (\Gamma_{MAX} + \alpha + \beta) > r\cdot(q_2 - 1)$

$r \cdot q_3 > (\Gamma_{l,i} + \alpha + \beta) > r\cdot(q_3 - 1)$

Alors :

$$\begin{cases} D_{MIN} = q_1 \cdot T_{SCR} + T_{NETf} - T_{NETb} + T_{I/O} + d_f \\ D_{MAX} = (q_2+1)\cdot T_{SCR} + T_{NETf} - T_{NETb} + T_{I/O} + d_f \\ D_r(p) = \theta_8(l+q_3) - v(p) \end{cases} \quad (27)$$

Avec ce cas général, la condition d'optimalité $r > (\alpha + \beta + \Gamma_{MAX})$ est plus dure car pour $\omega = n_1/n_2$, il suffit de prendre $(l-1) = n_2$ et $i = n_1$ pour avoir $\gamma_{l,i} = 1$. Ce qui implique que $\Gamma_{MAX} \geq 1$ et donc la condition C_2 est plus forte que C_1. C'est un résultat important car en prenant la période de scrutation un multiple de la période du CPU (en minimisant T_{CPU} au passage), on réduit le temps de réponse maximal de l'architecture.

C. *Cas 2: un PLC et N module E/S*

La modélisation du PLC ne change pas mais le nombre de modules E/S peut être quelconque. Les requêtes sont envoyées dans un ordre constant vers les RIOMs et le commutateur sans qualité de service gère les messages suivant la politique FIFO. Dans ce cas plus général, le GET précédent devient temporel et nous introduisons une équation supplémentaire relative aux temporisations τ_{12} et τ_7 pour modéliser la politique FIFO et résoudre le problème de partage du commutateur. Les RIOMs sont numérotés suivant le rang des requêtes qui leur sont destinées. Le destinataire de la $i^{éme}$ requête est alors affecté du numéro *i*. Le RIOM source de l'événement est affecté du numéro N_S et la destination de la conséquence de N_D. Avec une analyse similaire au cas 1 et avec les mêmes notations, on trouve les résultats suivants :

$$\begin{cases} D_{MIN} = q_1 \cdot T_{SCR} + (N_D - N_S)\cdot T_{em} + \tau_7^D - \tau_{12}^S + T_{I/O} + d_f \\ D_{MAX} = (q_2+1)\cdot T_{SCR} + (N_D - N_S)\cdot T_{em} + \tau_7^D - \tau_{12}^S + T_{I/O} + d_f \end{cases} \quad (28)$$

où $r \cdot q_1 > \left[\min_{l,i}(\Gamma_{l,i} + \alpha(l)) + \beta\right] > r\cdot(q_1 - 1)$

$r \cdot q_2 > \left[\max_{l,i}(\Gamma_{l,i} + \alpha(l)) + \beta\right] > r\cdot(q_2 - 1)$

$T_r(l) = (\alpha(l) + \gamma(l))\cdot T_{CPU}$, est le temps de retour de la requête du RIOM source de l'événement. τ_{12}^S et τ_7^D sont respectivement le temps de traversée du commutateur par la trame en provenance de la source *S* et la trame vers la destination *D*. Pour une architecture avec des temps de calculs dans les RIOMs constants ou très faiblement variables (c'est le cas en pratique), ces retards sont invariants et il suffit de les calculer une seule fois.

V. VALIDATION DU MODELE ET DE LA METHODE

Pour la validation, nous considérons deux configurations de l'architecture (Fig.2 et Fig.5). Nous comparons les résultats de l'algorithme et des formules précédentes avec ceux obtenus par mesures expérimentales sur la plate forme PRISME [10].

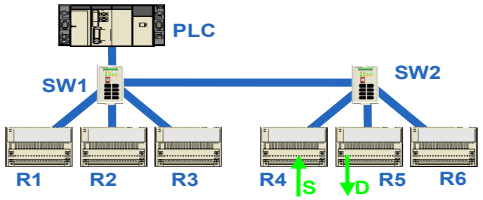

Fig. 5. Seconde configuration.

Dans la seconde configuration (Fig. 5), on s'intéresse au retard entre l'occurrence d'un événement généré sur l'entrée du module déporté R4 et sa conséquence sur la sortie du module R5. Les histogrammes de Fig. 6 représentent les résultats d'une compagne de 10,000 mesures et simulations de l'algorithme pour cette configuration. L'architecture est encore plus générale que les cas étudiés. Cela est fait dans le but de montrer la possibilité d'étendre nos résultats pour des cas plus complexes. La période du CPU est fixée à 5 ms et celle de scrutation à 10 ms avec une gigue de 15%. On obtient alors la table (I) qui donne les retards pour les différentes méthodes.

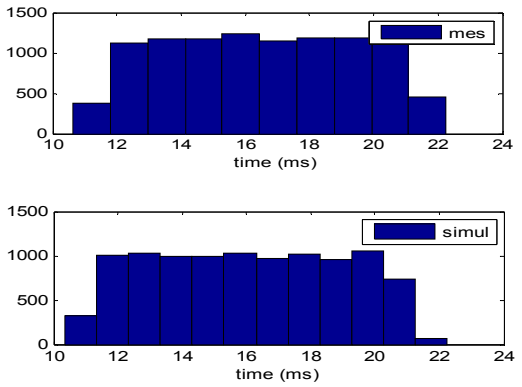

Fig. 6. Histogrammes des retards mesurées et simulés

TABLE I
RESULTATS DE MESURES, SIMULATIONS ET FORMULES ANALYTIQUES

		Les retards de réaction en ms		
		Min	Max	Moyenne
Cas 1	Mesures	10.40	21.90	16.10
	Simulation	10.05	22.24	15.82
	Formules	10.05	22.24	/
Cas 2	Mesures	10.65	22.25	16.40
	Simulation	10.31	22.49	16.07
	Formules	10.31	22.49	/

Dans les deux cas d'études on peut déjà conclure sur la validité des résultats analytiques car les bornes minimales sont toutes inférieures aux mesures et les bornes maximales sont supérieures. Les écarts de ces retards dans tous les cas ne dépassent pas 3.27%. Les simulations sont reproduites plusieurs fois avec un générateur d'événements à période aléatoire (pour compenser l'effet de gigue). Pour les extrema, aussi bien pour le min que le max, on retrouve exactement les mêmes valeurs que pour le calcul analytique. C'est en effet prévisible vu que les formules se basent sur le principe de l'algorithme. Et si on compare les résultats des deux configurations (cas1 et cas2), on constate qu'il y a environ un écart de 0.25 ms pour les retards max et min. C'est justement la valeur T_{EM} du temps de transmission des trames. Cela conforte les formules générales trouvées dans la section *IV.D*.

VI. CONCLUSION

Dans cette étude, on a modélisé des architectures de commande en réseau à l'aide de GETs. Grâce au formalisme Max-plus, on a obtenu un algorithme et des formules analytiques de calcul des temps de réponse. La confrontation des résultats avec des mesures expérimentales, nous a permis la validation aussi bien de l'algorithme que des formules. Il reste à étendre la validité des résultats de cette étude et leur généralisation, en se penchant sur des architectures plus complexes. Ainsi, sachant les exigences en temps de réponse d'une commande, on peut aisément à l'aide de ces formules poser la configuration adéquate de l'architecture afin de les satisfaire. Pour une étude plus approfondie, il serait intéressant de considérer aussi des architectures avec des flux apériodiques et qualité de service dans les commutateurs.

RÉFÉRENCES

[1] P. Neumann, "Communication in industrial automation -what is going on?" Control Engineering Practice, 2006.

[2] J. P. Georges, E. Rondeau, and T. Divoux, "Evaluation of switched Ethernet in an industrial context using network calculus," in *Proc. of 4th IEEE Int. Workshop on Factory Communication Systems*, 2002.

[3] N. Kakanakov, M. Shopov, G. Spasov and H. Hristev, "Performance evaluation of switched Ethernet as communication media in controller networks," International Conference on Computer Systems and Technologies (CompSysTech'07), pp. IIIA.8-1-6, 2007.

[4] K. C. Lee and S. Lee, "Performance evaluation of switched Ethernet for real-time industrial communications," Computer standards & interfaces, (24):411–423, 2002.

[5] J. Greifeneder, G. Frey, "Optimizing quality of control in networked automation systems using probabilistic models," in Proc. of 11th IEEE Int. Conf. on Emerging Technologies and Factory Automation, Prague, Czech Republic, September 2006.

[6] B. Ben-Hédia, F. Jumel and J.-P. Babau, "Formal evaluation of quality of service for data acquisition systems," in *Proc. of FDL'05*, 2005.

[7] D. Witsch, B. Vogel-Heuser, J. Faure, and G. Poulard-Marsal, "Performance analysis of industrial Ethernet networks by means of timed model-checking," in *Proc. of 12th IFAC Symposium on Information Control Problems in Manufacturing*, pp 101–106, 2006.

[8] D. A. Zaitsev, "Switched LAN simulation by colored Petri nets," *Mathematics and Computers in Simulation*," 65(3):245–249, 2004.

[9] G. Marsal, B. Denis, J.-M. Faure, G. Frey, " Evaluation of response time in Ethernet-based automation systems," in Proc. of 11th IEEE Int. Conf. on Emerging Technologies and Factory Automation, Prague, Czech Republic, September 2006.

[10] B. Denis, O. De Smet, J.-J. Lesage, J.-M. Roussel," Process of performance analysis and identification of systems as finite state automata", FR. Patent 01 110 933, August 2001.

[11] F. Baccelli, G. Cohen, G.-J. Olsder, and J.-P. Quadrat, *Synchronization and Linearity: An algebra for Discrete Event Systems*, Wiley, 1992.

[12] F. Baccelli, G. Cohen, and B. Gaujal, "Recursive equations and basic properties of timed Petri nets," *Discrete Event Dynamic Systems: Theory and Applications*, 1 (4), 1992.