



N° ENSC-2011/

THESE DE DOCTORAT
DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Présentée par

Boussad ADDAD

Pour obtenir le grade de

DOCTEUR DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Spécialité :

ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE

**Evaluation analytique du temps de réponse des systèmes
de commande en réseau en utilisant l'algèbre (max,+)**

Thèse présentée et soutenue à Cachan le 1^{er} juillet 2011 devant le jury composé de :

C. FRABOUL.	Professeur – ENSEEIHT-IRIT	Président
T. DIVOUX	Professeur – Université de Nancy-CRAN	Rapporteur
J-L. BOIMOND	Professeur – Université d'Angers-LISA	Rapporteur
J. KOMENDA	MC – Institute of Mathematics – Czech republic	Examineur
J-J. LESAGE	Professeur – ENS-Cachan – LURPA	Directeur de thèse
S. AMARI	Maître de conférences – IUT ST Denis – LURPA	Co-Encadrant



Laboratoire Universitaire de Recherche en Production Automatisée
(ENS CACHAN EA / 1385)
61 Avenue du Président Wilson, Cachan Cedex, 64235, France

Remerciements

Cette thèse de doctorat est une expérience unique qui sera bien gravée dans ma mémoire. Elle m'a permis de découvrir le monde passionnant de la recherche et m'a ouvert l'esprit sur différentes facettes de la vie. Durant ces trois années, j'ai eu l'opportunité et le plaisir de rencontrer de nombreuses personnes qui m'ont aidé de près ou de loin à la réalisation de mes travaux de recherche et à atteindre mes objectifs.

Ainsi, je tiens à exprimer ma forte gratitude à mon directeur de thèse, le Professeur Jean-Jacques LESAGE, pour la confiance qu'il m'a accordée et tous les conseils qu'il m'a prodigués durant toutes ces années que j'ai passées au LURPA. Son aide m'a été précieuse et me le sera pour toujours.

Je tiens également à remercier mon Co-encadrant Saïd AMARI. Au-delà de mon encadrement qu'il a mené avec toute énergie, il a été un soutien permanent sur le plan personnel et professionnel. Je lui exprime toute ma reconnaissance.

Je remercie tous les autres membres du LURPA qui m'ont soutenu et plus particulièrement Bruno DENIS pour ses conseils lors de la réalisation de mes manipulations sur la plateforme expérimentale.

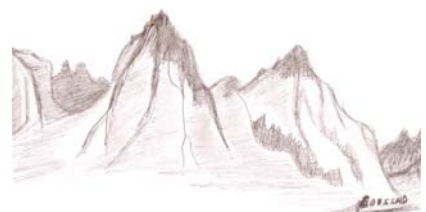
Je tiens aussi à remercier les Professeurs Thierry DIVOUX et Jean-Louis BOIMOND pour avoir accepté d'être rapporteurs de ma thèse et de prendre de leur temps pour la lire avec soin. Je remercie également le Professeur Christian FRABOUL pour avoir présidé ma soutenance de thèse et le Dr Jan KOMENDA pour avoir accepté d'être examinateur et membre du jury.

Je tiens bien évidemment à remercier mon cousin Ami Saïd, que je peux bien appeler « grand frère », Bilal AMGHAR, AZA-VALLINA Damien, AUDFREY Nicolas, Julien PROVOT ... et tous les autres qui m'ont aidé dans l'organisation de ma soutenance et surtout permettre à ma famille, restée au pays, d'y assister à distance. Cela m'a fait chaud au cœur et m'a donné énormément de motivation.

Enfin, j'adresse mes remerciements les plus profonds à toute ma famille qui m'entoure et me soutient en permanence malgré la distance qui nous sépare. Je leur dédie ce mémoire de thèse de doctorat et leur exprime tout mon amour. Que dieu me les protège.

*A ma très chère grand-mère « Azou-chabha »
A mes très chers parents
A mes très chères sœurs
A ma très chère future épouse*

Une pensée aux belles montagnes de Kabylie



"And in the end it's not the years in your life that count. It's the life in your years."
- Abraham Lincoln

Table des matières

Introduction générale

Chapitre 1 Etat de l'art & positionnement

1.1 Introduction	14
1.2 Exigences de réactivité des Systèmes de commande en réseau (SCR)	16
1.3 Etat de l'art et méthodes d'évaluation des performances des SCR	21
1.3.1 Simulation	21
1.3.2 Mesure expérimentale	23
1.3.3 Vérification formelle par model-checking	24
1.3.4 Méthodes analytiques	25
1.4 Positionnement et contributions	29
1.4.1 Avantages et inconvénients des méthodes existantes	29
1.4.2 Positionnement, contributions et démarche de la thèse	30

Chapitre 2 Fonctionnement & modélisation des SCR

2.1 Fonctionnement des SCR Client/Serveur	34
2.1.1. Architecture et composition des SCR	34
2.1.1.1.Fonctionnement des contrôleurs : mode périodique ou mode cyclique. .	34
2.1.1.2.Fonctionnement des modules d'E/S déportés	36
2.1.1.3.Réseau de communication : Hypothèse	37
2.1.2. Hypothèses sur le fonctionnement des SCR considérés	37
2.2 Modélisation en Graphes d'Evénements Temporisés (GET) et représentation (max,+) linéaire des SCR	38
2.2.1. Rappels sur les GET	38
2.2.2. Représentation d'état linéaire des GET l'algèbre (max,+)	40

2.2.3. Application aux SCR	42
2.2.3.1 Cas 1 : Mono Client Mono Serveur	43
2.2.3.2 Cas 2 : Mono Client Multiserveur	48
2.2.3.2 Cas 3 : Multi Client Multiserveur	51

Chapitre 3

Evaluation du temps de réponse des SCR

3.1 Evaluation du temps de réponse des SCR : analyse déterministe	55
3.1.1. SCR mono client mono serveur	55
3.1.1.1. Simplification et résolution des équations (max,+) linéaires	55
3.1.1.2. Calcul analytique des bornes du temps de réponse	59
3.1.1.2.1. Etude du cas $T_{COM} / T_{CPU} \in \mathbb{N}^+$	60
3.1.1.2.2. Etude du cas $T_{COM} / T_{CPU} \in \mathbb{Q}^+$	63
3.1.1.3. Calcul itératif de l'allure de la distribution du temps de réponse	65
3.1.2. SCR mono client multi serveurs	67
3.2 Evaluation du temps de réponse des SCR : analyse stochastique	70
3.2.1. Formulation du problème	70
3.2.2. Calcul analytique de la fonction de distribution du temps de réponse	72
3.3 Analyse de sensibilité du temps de réponse	74
3.3.1. Sensibilité aux délais de non synchronisation	74
3.3.2. Sensibilité aux délais de bout-en-bout	77
3.3.3. Conséquences de la sensibilité sur la qualité de commande	78

Chapitre 4

Evaluation des délais bout en bout dans les SCR

4.1 Introduction	85
4.1.1 Pourquoi Ethernet ?	85
4.1.2 La micro segmentation et le fonctionnement d'un commutateur Ethernet ...	86
4.2 Evaluation des délais de bout-en-bout dans un réseau Ethernet commuté	87
4.2.1 Etat de l'art	87

4.2.2 Analyse des scénarii d'interférence dans un SCR à base d'Ethernet	89
---	----

4.3 Déroulement itératif d'un scénario de traversée d'un réseau Ethernet commuté ..
..... **94**

4.3.1 Modélisation d'un commutateur à l'aide de GETC	94
--	----

4.3.2 Représentation d'état (max, +) des GETC	97
---	----

4.3.3 Modèle en équations (max, +) récurrentes d'un réseau Ethernet commuté ...	101
---	-----

4.4 Calcul de la borne maximale d'un délai de bout-en-bout **105**

4.4.1 Méthode de calcul combinatoire	105
--	-----

4.4.2 Algorithmes génétiques comme alternative pour des SCR de grande taille ..	106
---	-----

Chapitre 5
Validation expérimentale

5.1 Présentation de la plateforme expérimentale **115**

5.1.1 Architecture de commande considérée	115
---	-----

5.1.2 Outils et procédure de mesure expérimentale du temps de réponse	116
---	-----

5.2 Confrontation des résultats théoriques aux mesures sur un cas d'étude **118**

5.2.1 Calcul des bornes du temps de réponse	118
---	-----

5.2.2 Calcul de la fonction de distribution du temps de réponse	119
---	-----

5.3 Validation expérimentale de la formule de la borne maximale **123**

5.3.1 Impact de la période de scrutation sur la borne maximale	124
--	-----

5.3.2 Impact de la charge du module E/S capteur sur la borne maximale	126
---	-----

5.3.3 Impact de la charge du module E/S actionneur sur la borne maximale	127
--	-----

5.3.4 Impact de la charge d'autres module E/S sur le la borne maximale	128
--	-----

Conclusion & perspectives

Bibliographie

Introduction générale

Dans l'industrie, une composante majeure de l'économie, les impératifs d'optimisation de production, de réduction de coûts, de simplicité, de flexibilité, de fiabilité, ... ont conduit à des bouleversements à tous les niveaux de l'entreprise. Sur le terrain de la production, les procédés sont de plus en plus automatisés et l'intervention humaine se fait de plus en plus rare. Un ordinateur central suffit le plus souvent pour piloter un site entier de production. L'évolution de l'automatisation des processus industriels n'est cependant qu'à ses débuts. En effet, devant les inconvénients de ce schéma centralisé en termes de sûreté de fonctionnement et de coût, les systèmes centralisés laissent place aux systèmes de commande distribués. Les différents composants de terrain (contrôleurs, capteurs, actionneurs, superviseurs, etc.) échangent leurs données à travers des réseaux de communication, eux-mêmes connectés *via* des passerelles aux réseaux informatiques de l'entreprise. Ces systèmes de commande en réseau apportent beaucoup d'avantages mais des défis aussi. L'un de ces défis est évidemment l'interopérabilité et la coordination entre ces sous-systèmes fortement hétérogènes. Aussi, une communication à travers un réseau signifie le partage de ressources dont les capacités sont limitées. Par conséquent, des délais d'attente de disponibilité de ces ressources sont inévitables. Ainsi, un signal provenant d'un capteur n'arrive à un contrôleur qu'après un certain délai et de même pour un signal de commande allant d'un contrôleur vers un actionneur. Pire encore, ces signaux peuvent même être perdus en court de route à cause de débordement des ressources et ne jamais atteindre leur destination.

Bien entendu, ces délais ont un impact direct sur la réactivité de la boucle de commande et sur sa qualité (instabilité, dégradation de la dynamique, etc.). La question qui se pose alors est de trouver le moyen pour satisfaire ces besoins temps réel de la commande. Deux approches existent pour répondre à ces besoins ; la première consiste à s'assurer que les contraintes temps-réel sont satisfaites en amont, au moment même de la planification de l'architecture de commande et du protocole de communication. Les performances temporelles de ces solutions sont bien cernées par construction. Les paquets temps-réel sont en général soit séparés du trafic non contraint (transfert de fichiers son, vidéo, documentation, ...) ou bien traités en priorité en cas de contention lors de l'utilisation du médium de communication. L'inconvénient de ces solutions est souvent leur incompatibilité avec les réseaux standard comme Ethernet. Elles sont par conséquent relativement chères et pas assez flexibles.

La deuxième solution consiste à construire ou à planifier une architecture de commande aussi standard que possible mais nécessite d'être évaluée *a priori* avant sa mise en service. Si la propriété temps-réel recherchée est vérifiée lors de l'évaluation alors l'architecture est mise en œuvre. Sinon, des ajustements doivent être apportés jusqu'à satisfaire les exigences de la commande. Pour certaines applications à contraintes temps réel strictes (ex : temps de réponse de l'ordre de la milliseconde), cette solution ne peut leur être satisfaisante étant données leurs contraintes assez fortes. Heureusement, les applications industrielles ne sont pas aussi exigeantes dans leur majorité et la deuxième solution est largement répandue.

La présente thèse s'inscrit dans cette deuxième catégorie de solution. Notre objectif est d'évaluer des performances temporelles, non connues *a priori*, de systèmes de commande en réseau (SCR). Nous nous focalisons plus précisément sur ce qu'on appelle le *temps de réponse* d'un SCR. C'est le retard de causalité entre le changement d'un signal d'entrée (événement) et l'occurrence de sa conséquence sur une sortie du système commandé. Cette caractéristique temporelle est probablement des plus importantes mais des plus dures à évaluer. Des phénomènes de synchronisation dans les composants de terrain ou de partage de ressources dans le réseau de communication, doivent en effet être pris en compte. De ce fait, des modélisations et des analyses adéquates, assez simplifiées pour rester viables en pratique, mais assez précises pour ne pas dévier fortement de la réalité, doivent être apportées. Plusieurs travaux de recherche se sont penchés sur le sujet par le passé et des résultats intéressants ont été obtenus. Mais comme toute recherche n'est jamais finie, il reste des choses à faire avancer. C'est notre contribution à cela que nous tentons d'exposer dans ce manuscrit.

Cette thèse se décompose en cinq chapitres. Nous donnons dans cette introduction les grandes lignes du plan qui sera détaillé par la suite.

Dans le premier chapitre, nous introduisons les systèmes de commande en réseau et les exigences de leurs performances temporelles. Le temps de réponse étant l'exigence que nous abordons dans la thèse, nous rappelons les différents travaux de la littérature qui s'y sont intéressés. Nous pointons les avantages et les inconvénients des nombreuses méthodes existantes. Nous montrons ainsi les motivations de nos travaux de recherche et expliquons le plan de la démarche entreprise pour contribuer à leur résolution.

Dans le deuxième chapitre, nous présentons les SCR que nous avons étudiés i.e. ceux fonctionnant suivant le protocole Client/serveur où les contrôleurs sont les clients et les

modules d'E/S déportés sont les serveurs. Nous détaillons chacun de ces composants et énumérons les hypothèses retenues concernant leurs fonctionnements. Ensuite, nous en déduisons leurs modèles en utilisant des Graphes d'Événements Temporisés puis les équations (max,+) linéaires représentant leurs évolutions.

Le troisième chapitre est consacré à l'évaluation analytique du temps de réponse des SCR en exploitant les équations obtenues dans le chapitre précédent. Deux approches d'analyse sont considérées. Une première approche déterministe par laquelle des formules de calcul des bornes, minimale et maximale, du temps réponse ainsi que l'allure de sa distribution, sont obtenues. La deuxième approche est stochastique et s'intéresse au calcul analytique exact de la fonction de distribution du temps de réponse.

Le quatrième chapitre est consacré au calcul de délais de bout-en-bout (délais subis dans le seul réseau de communication) pour les injecter dans les formules obtenues dans le chapitre précédent et finalement calculer le temps de réponse d'un SCR donné.

Le cinquième chapitre est dédié à une validation expérimentale des différents résultats obtenus. De nombreuses mesures de temps de réponse, réalisées sur une plateforme réelle, sont comparées aux prédictions des formules analytiques. Une attention particulière est portée sur la validation de la formule de calcul de la borne maximale du temps de réponse.

Enfin, une synthèse des résultats obtenus ainsi que quelques perspectives envisagées pour des travaux futurs sont données pour conclure ce manuscrit.

Etat de l'art & positionnement

Sommaire

1.1 Introduction	14
1.2 Exigences de réactivité des Systèmes de commande en réseau (SCR)	16
1.3 Etat de l'art et méthodes d'évaluation des performances des SCR	21
1.3.1 Simulation	21
1.3.2 Mesure expérimentale	23
1.3.3 Vérification formelle par model-checking	24
1.3.4 Méthodes analytiques	25
1.4 Positionnement et contributions	29
1.4.1 Avantages et inconvénients des méthodes existantes	29
1.4.2 Positionnement, contributions et démarche de la thèse	30

Dans ce chapitre bibliographique, nous rappelons brièvement ce que sont les systèmes de commande en réseau (SCR). Ensuite, à travers un exemple, nous montrons l'une des principales exigences des SCR, la réactivité ou *le temps de réponse*. De là, nous énumérons les différents travaux existants quant à l'évaluation du temps de réponse. Quatre catégories sont exposées : simulation, mesure expérimentale, vérification par model-checking et enfin les méthodes analytiques. Devant les inconvénients de ces méthodes en termes de temps de calcul ou de garantie d'évaluation des bornes du temps de réponse, nous proposons une nouvelle approche analytique permettant d'y remédier. A travers un schéma synoptique, nous résumons la démarche suivie pour atteindre cet objectif.

~ *The formulation of a problem is often more essential than its solution,
which may be merely a matter of mathematical or experimental skill.* ~
- Albert Einstein

1.1 Introduction

Au début de leur application, les systèmes de contrôle-commande étaient analogiques. Les architectures étaient extrêmement simples et les composants relativement rudimentaires. L'avènement de la communication digitale et des réseaux allait révolutionner le domaine en l'espace de quelques décennies. Les réseaux appliqués à la commande ont été utilisés pour la première fois dans l'industrie dans les années 70, avec l'introduction des communications digitales entre des ordinateurs et des E/S (entrée/sortie). Différents constructeurs comme Honeywell, Yokogawa et US-based Bristol les ont utilisés pour la première fois en 1975. Par la suite, des réseaux de communication ont été utilisés dans des systèmes de commande mais pour relier essentiellement des unités de commande ou des contrôleurs aux consoles des opérateurs dans les salles de commande. La communication digitale dans les composants de moindre taille comme les modules d'E/S n'est apparue quant à elle que dans les années 80. Elle s'est largement répandue par la suite dans les années 90. De là, l'échange de données avec un module E/S n'est plus réservé exclusivement à un seul contrôleur. Les systèmes de commande sont ainsi devenus *distribués* dans le sens où plusieurs tâches sont distribuées ou partagées par plusieurs contrôleurs. Ces systèmes sont le plus souvent appelés systèmes de contrôle commande en réseau ou SCR (ou *Networked control systems* dans la littérature anglophone). Ceci a évidemment l'avantage direct d'une meilleure disponibilité des systèmes de commande puisqu'un contrôleur en panne n'implique pas forcément l'effondrement de toute l'architecture comme c'était le cas en commande centralisée. Le développement spectaculaire des systèmes de commande en réseau peut aussi s'expliquer par d'autres avantages que la communication digitale leur procure (Berge, 2002) :

- *Plus d'informations* : l'avantage majeur de la communication digitale est qu'elle permet en effet d'envoyer en série des flux de zéros et de uns. Au lieu d'un câble pour chaque variable comme c'était le cas en analogique, des milliers ou même des millions d'informations peuvent être acheminées grâce à un seul câble. En 1981 déjà, le constructeur australien Midac a installé un système en réseau sur le campus de l'Université de Melbourne en reliant plusieurs départements avec environ 20.000 variables échangées (informations possibles). La communication digitale permet aussi une multitude d'échanges d'information entre composants de plus en plus intelligents et géographiquement dispersés. Un opérateur voulant par exemple tester le bon fonctionnement d'un composant de terrain n'est plus obligé de se déplacer localement mais peut le faire à distance. Aussi, les frontières entre les différentes couches de l'entreprise disparaissent et des données destinées à diverses

applications comme la commande, le diagnostic, la maintenance, la supervision ..., se côtoient sur le même medium de communication.

- *Plus de robustesse* : dans les systèmes analogiques 4-20mA, un signal transmis depuis un capteur est sujet à des perturbations lors de son acheminement vers un contrôleur. Il peut arriver à sa destination avec des dégradations ou des distorsions majeures sans pour autant être considéré comme invalide. La communication digitale a l'avantage d'être plus robuste vu que seuls deux états valides sont considérés, zéro et un. De plus, des techniques de codage et de vérification d'erreur sont disponibles. Un message contenant une erreur peut être écarté et une demande de retransmission est aussi possible.

- *Plus de ramifications et moins de câbles* : l'autre bénéfice est la possibilité de relier plusieurs composants sur la même paire de câble pour former un réseau avec un medium partagé. Par conséquent, le volume des câbles utilisés, notamment sur les longues distances, est considérablement réduit et le coût des installations aussi.

Aussi, l'utilisation d'un medium avec des ramifications permet d'adopter plusieurs protocoles d'échange d'informations entre les différents nœuds du réseau, chaque protocole étant particulièrement plus pertinent pour une application donnée. De plus, chaque nœud étant défini par une adresse unique, toute ambiguïté est évitée. Les principaux protocoles d'échange de données dans les SCR sont les suivants (Vitturi, 2001) :

- *Maître/esclave* : avec ce protocole, un esclave ne peut émettre d'information que lorsqu'il reçoit l'aval du maître (ex : Profibus DP). Le maître interroge cycliquement les esclaves et leur donne la main, chacun son tour, pour pouvoir émettre des données.
- *Client/serveur* : cette fois, un client interroge de manière autonome un serveur pour demander une information et le serveur lui répond simplement (ex : Modbus TCP).
- *Producteur/consommateur* : dans ce cas, un nœud producteur n'attend pas la réception d'un aval ou d'une requête pour émettre de l'information. Il le fait indépendamment (à chaque fois qu'une variable change d'état par exemple) et l'information est envoyée précisément aux seuls consommateurs intéressés par la variable publiée (ex : WorldFIP).

Hélas, les réseaux de communication appliqués aux systèmes de commande apportent aussi leur lot d'inconvénients. Parmi ceux-ci, nous trouvons (Berge, 2002) :

- *Pas si décentralisé que ça* : au début de leur apparition, les SCR étaient considérés comme distribués comparés aux systèmes centralisés qui les ont précédés. Cela demeure cependant assez relatif puisque les SCR restent vulnérables aux pannes au niveau du réseau qui peuvent entraîner tout ou une partie du système dans la défaillance. Ceci est d'autant plus

valable avec l'utilisation de composants réseau (ex : Ethernet) qui ne sont pas originellement construits pour travailler dans des milieux hostiles comme c'est le cas en industrie. Toutefois, des solutions logicielles et matérielles, en utilisant par exemple des redondances des échanges de données et des composants, peuvent être apportées (Limal, 2009).

- *Manque d'interopérabilité* : lors de l'introduction de la communication digitale, les constructeurs ont développé leurs propres protocoles indépendamment les uns des autres. Des protocoles divers et variés ont inondé le marché et chaque produit n'est compatible qu'avec les produits du même constructeur. De plus, la documentation n'est pas toujours disponible et les technologies sont protégées par des brevets. Les verrous de cette situation étaient très dommageables à l'industrie. L'un des inconvénients majeurs est que les produits d'un seul constructeur ne sont le plus souvent pas suffisants pour couvrir tous les besoins d'un site entier de production. L'utilisation de produits de plusieurs fournisseurs, même non compatibles, était alors incontournable, conduisant à un melting-pot et des îlots dans un même site de production. Ce problème sera de moins en moins posé à l'avenir avec l'utilisation de standards comme Ethernet (voir Chapitre 4) mais ce ne sera probablement que partiel puisqu'il existe un énorme fossé entre les exigences des diverses applications des sous-systèmes des entreprises. Les différences entre le terrain (systèmes d'automatisation) et les systèmes d'information par exemple sont de taille. Les exigences dans les systèmes informatiques concernent le plus souvent la vitesse de traitement de l'information étant donné les quantités considérables de données échangées. Néanmoins, cela reste du trafic appelé *non temps-réel*. Le téléchargement d'un fichier peut bien être effectué en une durée plus au moins importante sans pour autant que cela soit gênant. De plus, les performances grandissantes des composants réseau avec des débits allant jusqu'au Gigabit ou même Térabit par seconde, répondent bien aux exigences de ces systèmes d'information. Sur le terrain ou dans les SCR par contre, bien que les paquets échangés soient relativement petits, des contraintes *temps-réel* plus ou moins strictes doivent être vérifiées. Une donnée envoyée, même petite soit-elle, doit non seulement atteindre sa destination mais avant une échéance bien définie. Dans cette thèse, nous nous limiterons à ce type d'exigences. Nous aborderons les exigences en termes de performances temporelles des SCR et plus précisément leur *réactivité*. Un exemple pratique pour illustrer cela est présenté dans la section ci-après.

1.2 Exigences de réactivité des SCR

Pour expliquer les exigences de réactivité des SCR, nous reprendrons l'exemple d'un système de production d'énergie décrit dans (Limal, 2009). Il consiste en une centrale thermique classique comme représenté sur la Fig. 1.1.

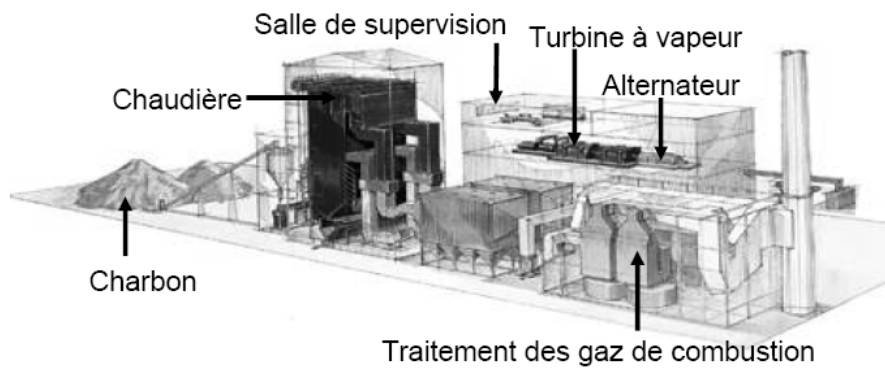


Fig. 1.1 : Exemple de centrale thermique classique (Limal, 2009)

Les centrales thermiques classiques brûlent des énergies fossiles (gaz, fuel, charbon, etc.) pour produire de la vapeur par l'intermédiaire d'une chaudière. La vapeur entraîne une turbine reliée à un alternateur. Ce dernier transforme l'énergie mécanique en énergie électrique. Si d'autres centrales thermiques existent (solaire, nucléaire, biomasse, hydroélectricité, éolienne, etc.), les procédés de production sont tous automatisés. L'automatisation permet par exemple de piloter l'ouverture d'une vanne d'un barrage hydroélectrique comme de réguler l'alimentation en carburant d'une turbine à combustion. L'automatisation intervient aussi dans les processus associés comme la gestion des produits issus de la combustion ou le contrôle de la tension produite par l'alternateur. Par conséquent, une architecture de contrôle commande est associée à tout processus automatisé dans la centrale. Un exemple en est donné dans la Fig. 1.2.

Sur l'architecture de la Fig. 1.2, nous pouvons remarquer qu'un même motif, un réseau de supervision, se répète à plusieurs emplacements pour piloter différents processus liés à la chaudière et à la turbine. En effet, le système de contrôle commande a été défini pour offrir une solution unifiée pour réaliser l'automatisation des processus ou le contrôle de machines tournantes avec les mêmes technologies au sein d'une même centrale de production d'énergie. Cette approche de contrôle commande unifiée permet de décrire chaque architecture par la combinaison de postes d'ingénierie et de supervision, de réseaux de supervision ainsi que ce cellule d'automatisation.

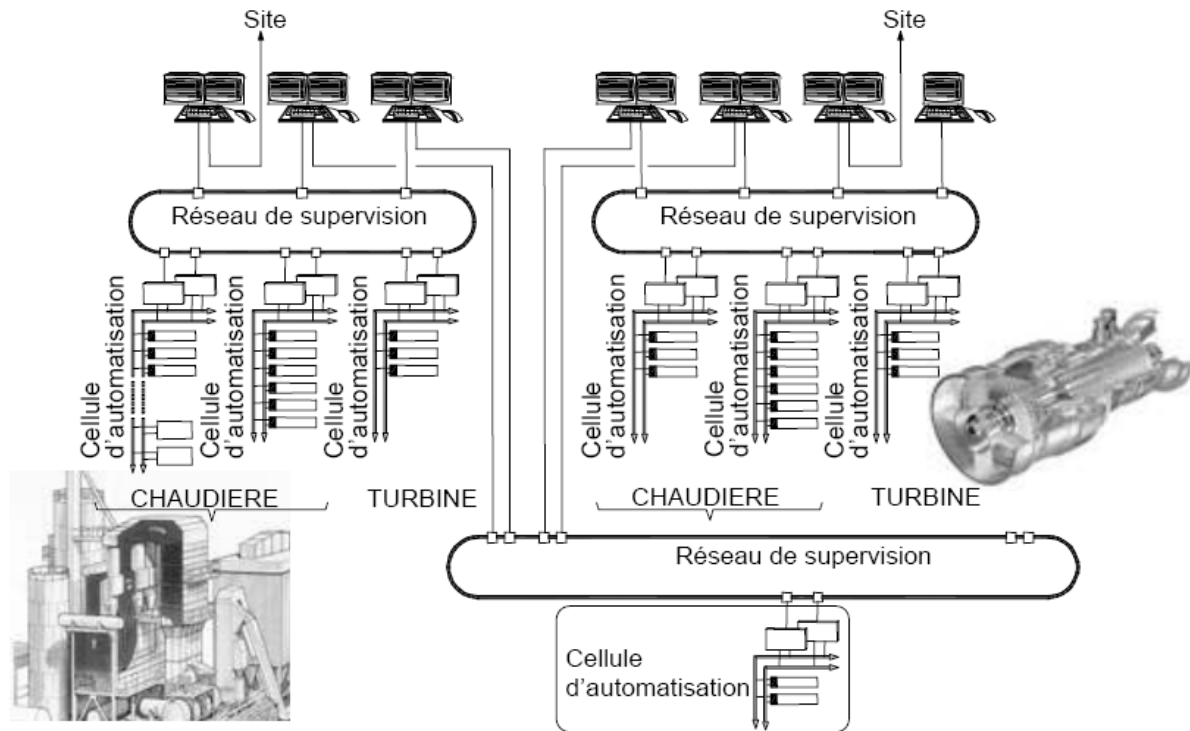


Fig. 1.2 : Exemple d'architecture de contrôle commande (Limal, 2009)

La cellule d'automatisation dont un exemple d'architecture est illustré sur la Fig. 1.3, est l'entité où sont automatisées une ou plusieurs parties du processus de production d'énergie. La cellule d'automatisation est un système de contrôle-commande distribué ou plus communément SCR comme le cas dans le présent document. Elle est constituée de noeuds (un contrôleur de cellule, des contrôleurs de terrain ou des modules d'E/S déportés) communiquant par l'intermédiaire d'un réseau de communications appelé réseau de terrain. C'est aux performances temporelles de cette cellule d'automatisation que nous nous intéressons exclusivement dans ce manuscrit. Les caractéristiques à maîtriser pour le pilotage du processus de production d'énergie y sont en effet localisées.

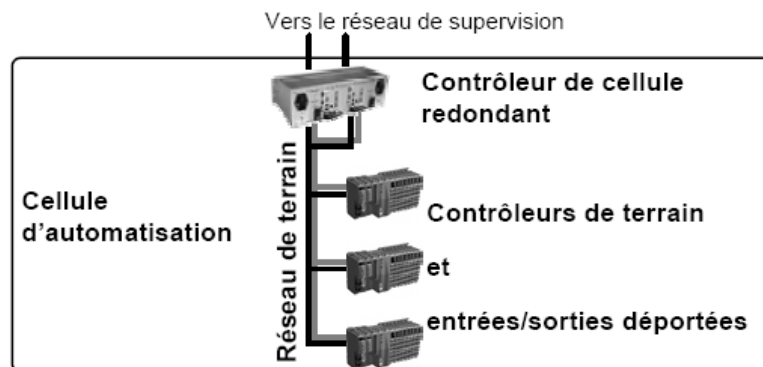


Fig. 1.3 : Cellule d'automatisation ou SCR (Limal, 2009)

Les exigences du SCR sont principalement le déterminisme, la disponibilité et la réactivité (Limal, 2009). Par déterminisme, on entend la capacité à prédire l'état suivant d'un système connaissant son état actuel et ses entrées. Autrement dit, pour chaque combinaison des entrées et de l'état du système, il existe une sortie unique. Par disponibilité - une composante de la sûreté de fonctionnement - on entend la capacité à maintenir une qualité de service et la satisfaction d'exigences temporelles comme la réactivité, malgré l'occurrence d'une défaillance. Ceci nous amène à la dernière exigence qu'est la réactivité et qui sera l'objet principal de cette thèse.

Comme nous pouvons remarquer sur la Fig. 1.4, il existe un délai non nul entre le changement d'état d'un signal d'entrée d'un module déporté E, acquis via un capteur par exemple, et l'occurrence de sa conséquence sur un signal de sortie d'un module déporté S, émis pour commander un préactionneur du processus.

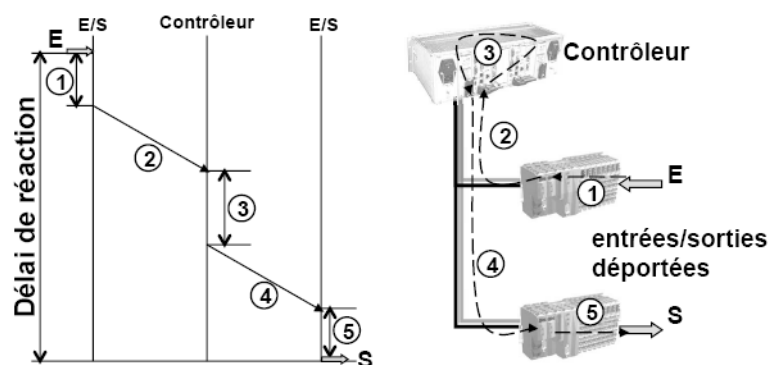


Fig. 1.4 : Réactivité ou temps de réponse d'un SCR

Ce délai de réactivité que nous appelons *temps de réponse*, peut être divisé en plusieurs délais élémentaires (détaillés par la suite dans le Chapitre 3) :

- 1) Délai d'attente dans le module déporté entrée E
- 2) Délai de traversée *aller* du réseau de terrain (appelé *délai de bout-en-bout* par la suite)
- 3) Délai d'attente, traitement, émission ... dans le contrôleur
- 4) Délai de traversée *retour* du réseau de terrain
- 5) Délai d'attente dans le module déporté sortie S

Le SCR doit respecter des contraintes temps-réel dépendant du processus automatisé. Il est suffisamment réactif si son temps de réponse est inférieur à une contrainte temporelle donnée. Sur la Fig. 1.5, l'émission sur la sortie S respecte la contrainte alors que l'émission sur S' ne la respecte pas.

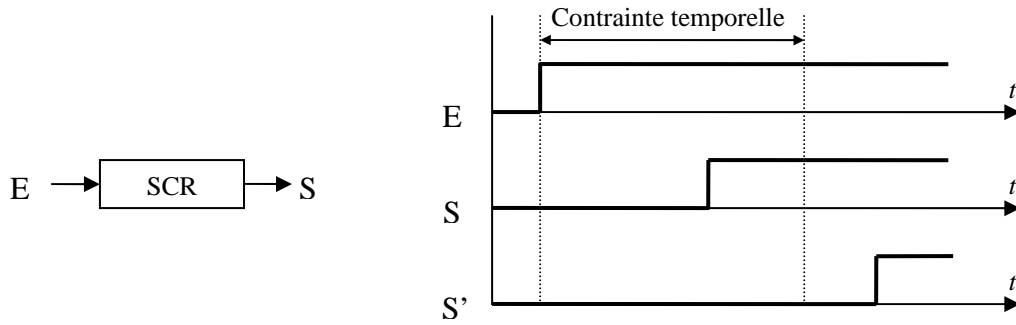


Fig. 1.5 : Illustration de la contrainte temps réel sur le temps de réponse

Suivant le contexte, les SCR sont soumis à des contraintes temps réel plus ou moins strictes :

- Contraintes temps réel strictes : les contraintes doivent être respectées dans tous les cas sous peine d'avoir des conséquences dangereuses.
- Contraintes temps réel souples : des dépassements d'échéances peuvent être tolérés sans engendrer des dangers mais la fréquence de ces dépassements doit être limitée pour garder une qualité de commande acceptable.

Comme nous le verrons par la suite, le temps de réponse n'est pas constant et peut varier d'un cycle à un autre. De ce fait, un SCR est plus réactif qu'un autre suivant les contraintes auxquelles il est soumis. Sur la Fig. 1.6, le premier SCR est plus performant que le second SCR dans un contexte temps réel strict. Le pire temps de réponse est en effet inférieur à la contrainte temporelle fixée. Si en revanche on se retrouve dans un contexte temps réel souple, le second SCR est plus performant puisque la valeur moyenne de son temps de réponse est moindre.

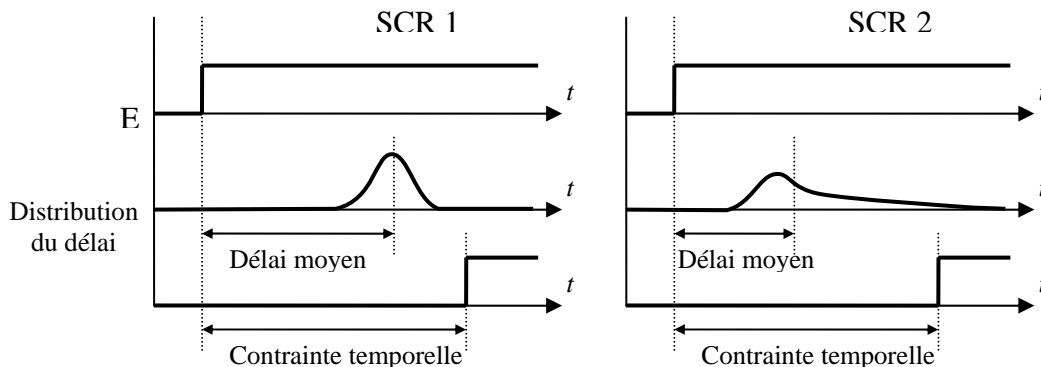


Fig. 1.6 : Comparaison entre les temps de réponse de deux SCR

Au final, pour juger de la performance d'un SCR en termes de réactivité, il faut évaluer son temps de réponse et non seulement des délais de bout en bout. De plus, suivant le contexte et

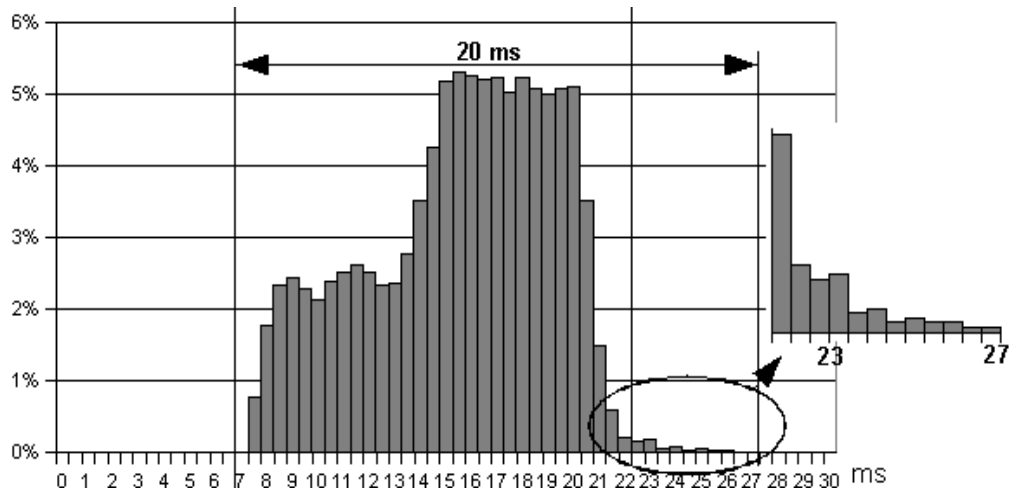


Fig. 1.8 : Répartition des temps de réponse obtenus par simulation (Marsal, 2006a)

Dans la même catégorie, (Liu *et al.*, 2007) ont utilisé Dymola, un environnement de modélisation et de simulation basé sur le langage de programmation orienté objet Modelica, pour créer des classes d'objets, correspondant aux différents composants des SCR. Ils les ont simulés par la suite pour calculer une distribution du temps de réponse. Les auteurs ont également utilisé un autre outil appelé TrueTime (sous Matlab/Simulink). Une comparaison entre les deux méthodes a été réalisée (Fig. 1.9). TrueTime donnait plus de choix quant aux protocoles de communication utilisés dans les SCR mais le temps de simulation était nettement moindre en utilisant Dymola.

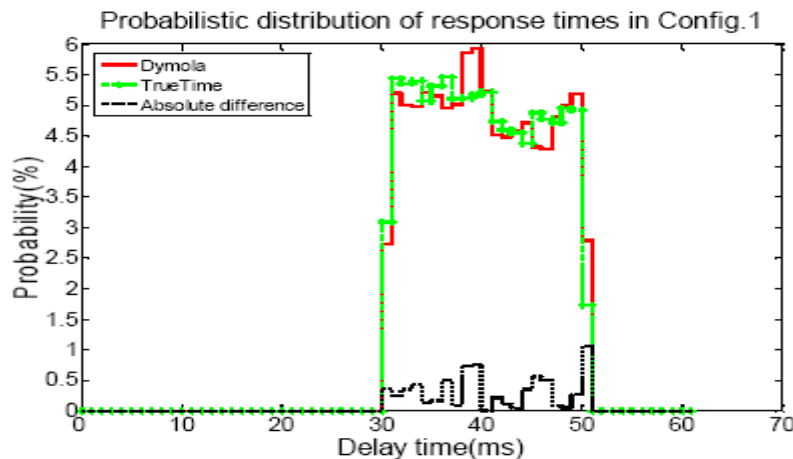


Fig. 1.9 : Temps de réponse simulés, Dymola vs. TrueTime (Liu *et al.*, 2007)

Les auteurs (Pereira *et al.*, 2004a) ont utilisé l'outil OMNeT++ dont les éléments (les composants) sont programmés à l'aide du langage de programmation C++ puis assemblés en composants plus élaborés (topologie des réseaux : étoile, arbre, ligne, ...) à l'aide d'un langage de plus haut niveau NED (NETwork Description). Les auteurs modélisent un SCR

Producteur/consommateur avec des modules OMNeT++ (exemple d'un modèle de contrôleur sur la Fig. 1.10) et simulent son fonctionnement pour déterminer les temps de réponse.

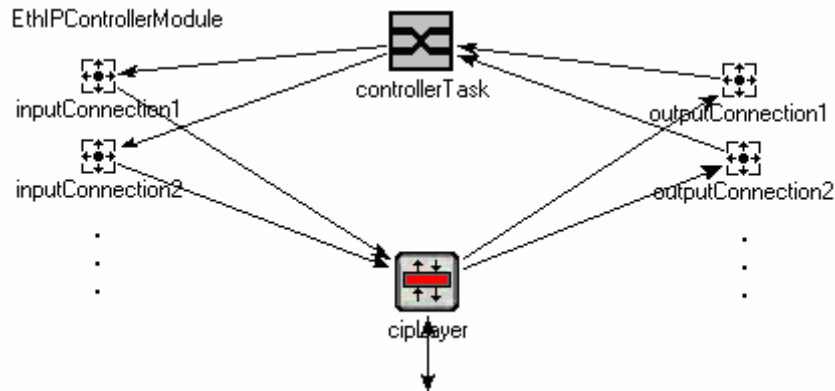


Fig. 1.10 : Modèle OMNET++ d'un contrôleur (Pereira *et al.*, 2004a)

1.3.2 Mesures expérimentales

Ces méthodes consistent à mesurer expérimentalement les performances temporelles d'un SCR. Dans (Denis *et al.*, 2007), les auteurs ont utilisé une plateforme expérimentale (décrite plus en détail dans le Chapitre 5 puisque nous nous en servons pour la validation de nos résultats théoriques) dédiée à l'identification des SED ainsi que la mesure des temps de réponse des SCR. Les auteurs ont évalué l'impact du trafic non temps réel sur la boucle de commande (Fig. 1.11). Il y a été conclu que l'effet des délais de bout-en-bout est relativement moins important que les délais dus à la charge des contrôleurs ou des modules déporté d'E/S.

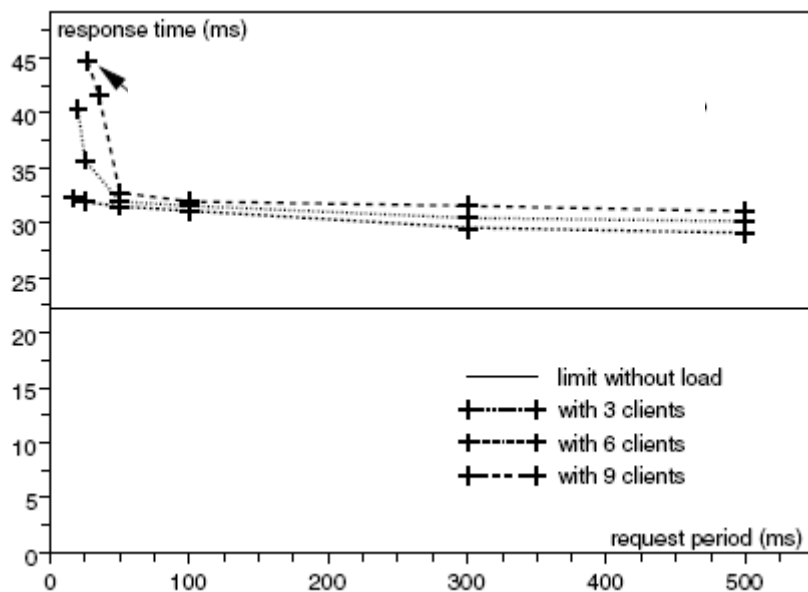


Fig. 1.11 : Impact de la charge d'un contrôleur sur le temps de réponse maximal (Denis *et al.*, 2007)

Un autre travail expérimental, concernant cette fois le temps de cycle d'un réseau PROFINET IO ainsi que sa gigue, a été présenté dans (Schafer *et al.*, 2007). Différentes méthodes de mesures sont utilisées et comparées, notamment un outil de mesure de haute résolution HRTA (High Resolution Timing Tool), l'outil bien connu Wireshark (implémenté sous Windows et sous Linux), la méthode Port Miroir et enfin la méthode de robinet passif (Passive Tap).

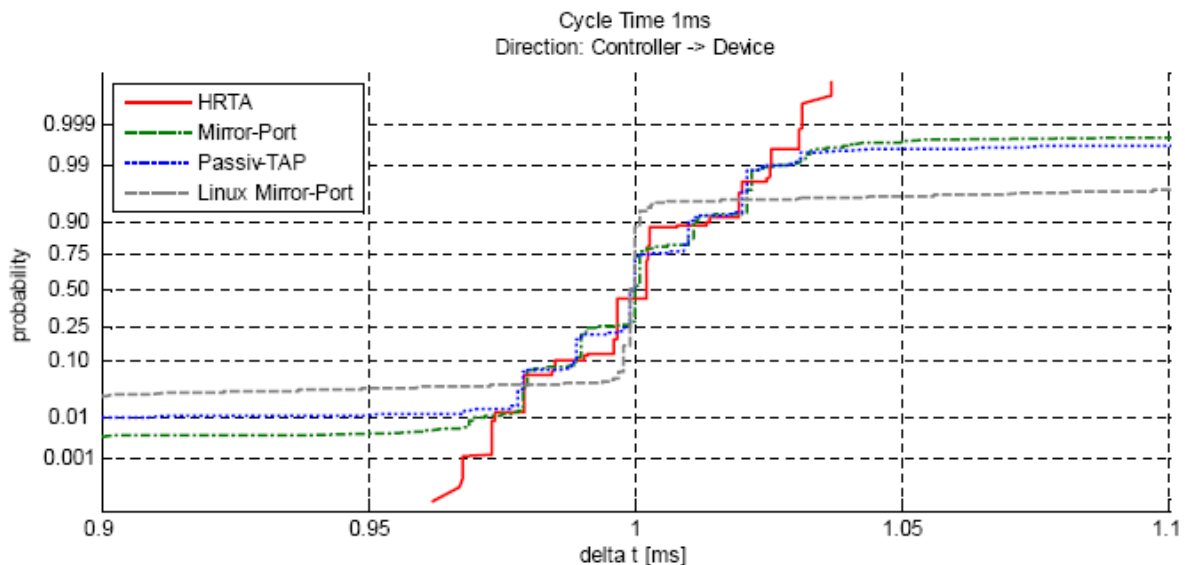


Fig. 1.12 : Mesure du temps de cycle avec plusieurs méthodes expérimentales (Schafer *et al.*, 2007)

Une expérimentation similaire a été menée par (Silvola *et al.*, 2007) sur un réseau PROFINET IO pour la mesure du temps de cycle réseau ainsi que sa gigue. Du trafic perturbateur UDP (non temps réel et de basse priorité) a été généré pour mesurer son impact sur les performances temporelles du SCR. Les commutateurs Ethernet utilisés prennent en compte l'ordonnancement avec priorité ou différenciation de trafic selon les spécifications IEEE 802.1D/Q.

D'autres travaux expérimentaux existent et concernent le plus souvent le protocole LAN Ethernet et les délais de bout-en bout. Nous allons y revenir dans le Chapitre 4.

1.3.3 Vérification formelle par model-checking

Cette méthode est à l'origine utilisée pour prouver, par vérification exhaustive, qu'une propriété est vérifiée ou non par un modèle. Elle a été adaptée pour l'évaluation de performances temporelles des SCR. Cela consiste à vérifier par exemple si le temps de réponse dans un SCR atteint une valeur donnée ou pas. Ainsi, par un mécanisme de

dichotomie par exemple, on peut calculer les bornes, minimale et maximale, du délai avec une bonne précision.

Cette méthode a été utilisée par (Krakora *et al.*, 2004) pour un SCR utilisant le protocole CAN et (Witsch *et al.*, 2006), (Ruel *et al.*, 2008a) pour un SCR utilisant Ethernet. Les auteurs ont réussi à calculer les bornes maximales des temps de réponse. Les modèles utilisés pour la vérification sont des automates à états finis temporisés (Fig. 1.13). Par la suite, cette méthode a été utilisée pour le calcul de la distribution du temps de réponse par (Greifeneder *et al.*, 2007). Cette fois, c'est le model-checking probabiliste qui a été utilisé. Il consiste à évaluer la probabilité que le temps de réponse dépasse une certaine valeur. Par accumulation de résultats de plusieurs vérifications, les auteurs obtiennent l'allure de la répartition du temps de réponse. L'application de cette méthode a été étendue par la suite pour différents protocoles, Client/serveur et Producteur/consommateur, dans (Greifeneder *et al.*, 2008a). Enfin, une comparaison de la vérification formelle par model-checking et de la simulation a été proposée dans (Greifeneder *et al.*, 2008b).

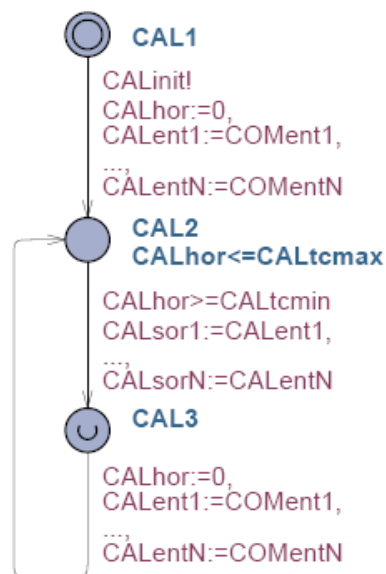


Fig. 1.13 : Modèle en automate à états fini temporisé d'un contrôleur (syntaxe UPAAL) (Ruel *et al.*, 2008a)

1.3.3.4 Méthodes analytiques

Les auteurs (Tovar *et al.*, 2001) ont présenté une méthode analytique de calcul du temps de réponse dans un SCR utilisant WorldFIP, un protocole Producteur/consommateur. Vu le caractère statique de ce protocole, impliquant un mécanisme d'arbitrage d'accès au médium (ordonnancement) connu a priori, des formules de calcul du pire temps de réponse ont été aisément formulées. Dans (Vitturi, 2001), l'auteur présente des formules de calcul du temps

de cycle de deux protocoles, Profibus DP et WorldFIP, et compare leurs performances à celle d'un réseau Ethernet en mode Maître/esclave et Producteur/Consommateur.

Les méthodes suivantes concernent les délais de bout-en-bout et non le temps de réponse ...

- *Calcul réseau* : la littérature abonde de méthodes d'évaluation des délais de bout-en-bout notamment quand il s'agit des réseaux basés sur Ethernet commuté. Le calcul réseau (en anglais *Network Calculus*) est incontestablement la méthode la plus connue et la plus utilisée actuellement. Elle a été introduite par Cruz (1991a), Le Boudec *et al.*, (2004), et Chang (2002) puis utilisée par la suite par d'autres auteurs (Georges *et al.*, 2005a), (Fraboul *et al.*, 2004). Cette méthode est basée sur la théorie mathématique des dioïdes (min,+) et (max,+). Un majorant d'un délai de traversée d'un composant est calculé en utilisant deux concepts ; courbe d'arrivée et courbe de service. En déterminant l'arriéré de traitement maximal à l'aide de ces courbes, on arrive à calculer un majorant de délai de traversée du composant (Fig. 1.14). En considérant un commutateur comme étant une cascade de composants (file, multiplexeur, démultiplexeur, etc.) dont les courbes d'arrivée sont soit connues (stations périphériques), soit calculées grâce à des résultats de la théorie des dioïdes précédents, un majorant du délai de traversée d'un commutateur est calculé. Le réseau étant constitué de plusieurs commutateurs en cascade, un majorant du délai global de traversée d'un réseau Ethernet commuté est obtenu. Toutes ces étapes sont détaillées de manière assez complète dans (Georges, 2006).

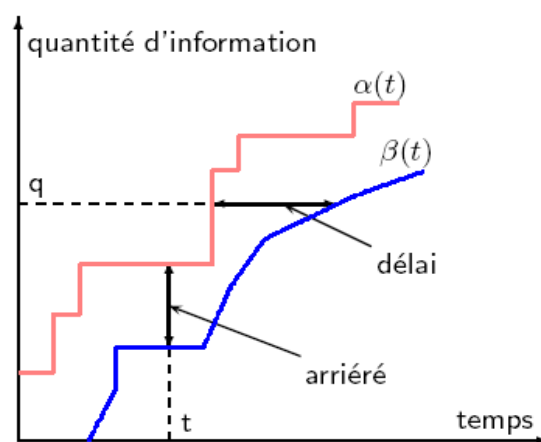


Fig. 1.14 : Calcul réseau et ses concepts : courbe d'arrivée $\alpha(t)$ et courbe de service $\beta(t)$ (Georges 2005b)

L'inconvénient qui est reproché au calcul réseau est souvent son pessimisme dans l'évaluation des majorants. En effet, la courbe d'arrivée est souvent une surestimation de la quantité réelle

des données reçues par un composant, l'objectif premier étant d'éviter toute sous-estimation des délais. Par ailleurs, le besoin de simplification pousse souvent à utiliser des courbes d'arrivée/service faciles à manipuler mathématiquement (courbes affines) et cela se fait au détriment de la précision de l'évaluation. Cette méthode a été améliorée par la suite en appliquant le groupement des flux provenant de la même source et transitant à travers un même port de sortie (jusqu'à environ 50% de gain sur une architecture industrielle (Bauer *et al.*, 2009)). Cette même technique de groupement a été appliquée dans la méthode par trajectoire, présentée ci-dessous.

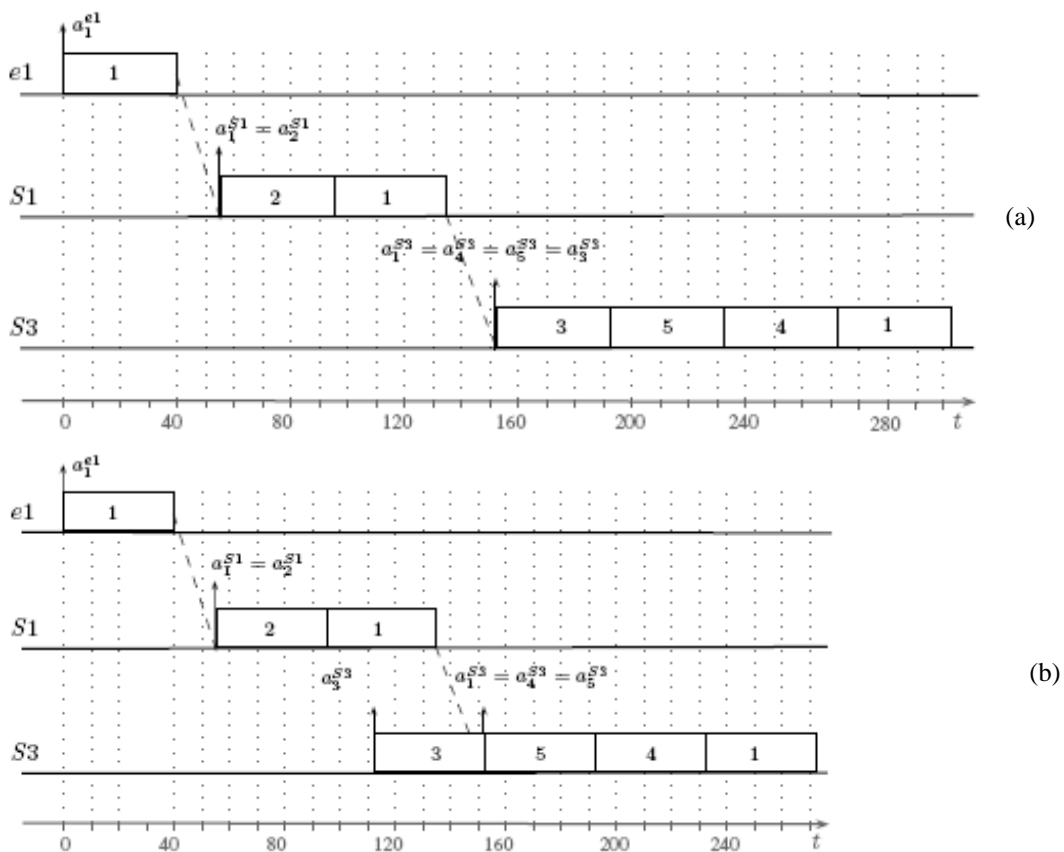


Fig. 1.15 : Approche par trajectoire (a) classique (b) améliorée avec groupement (Bauer *et al.*, 2009)

- *Approche par trajectoire* : l'approche par trajectoire est une méthode analytique qui permet de calculer un majorant de délai de bout-en-bout en identifiant les paquets d'autres flux que le paquet en question rencontre sur sa trajectoire lors de la traversée du réseau. Elle se base sur le concept de *période active* (en anglais *Busy Period*). Une période active à un niveau du réseau est un délai d'attente subi par le paquet en attendant la fin d'expédition des paquets qui l'ont précédé. En retardant au maximum le début de ces périodes actives, et avec un calcul récursif en remontant depuis le niveau de destination du paquet, on peut calculer un majorant

du délai de bout-en-bout (Martin, 2006). Dans sa version originale, cette méthode souffrait, tout comme le calcul réseau, du phénomène de sérialisation des messages provenant de la même source. Elle a été améliorée en utilisant le groupement des flux de la même source (Bauer *et al.*, 2009) , (Bauer *et al.*, 2010). La méthode est en moyenne 10% meilleure que le calcul réseau groupé dans les cas étudiés.

- *Approche pire cas* : les auteurs (Lee *et al.*, 2002) ont proposé cette méthode en se basant sur des conditions de stabilité du réseau Ethernet commuté (pas d'effondrement à cause des débordements). Ils calculent le nombre maximal de trames de longueur minimale dans une file d'attente, assurant cette stabilité, puis en déduisent le délai maximal de bufferisation. En y ajoutant les délais de propagation, les auteurs obtiennent un majorant du délai de traversée. Cette méthode est assez restrictive car elle considère un seul commutateur et elle est très pessimiste car elle se base sur une limite supérieure de fonctionnement du réseau. Dans les SCR d'ailleurs, cette limite n'est le plus souvent pas atteinte.

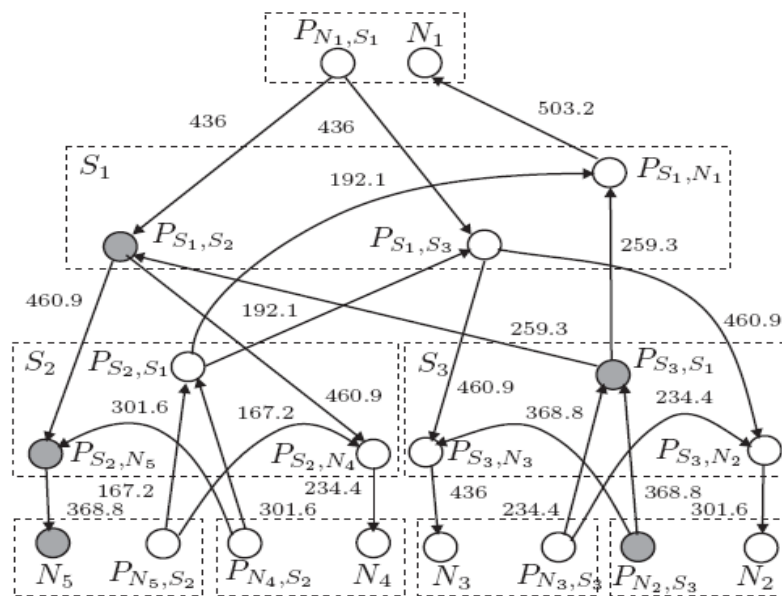


Fig. 1.16 : Approche du plus long chemin sur un graphe pondéré (Schmidt *et al.*, 2010)

- *Approche du plus long chemin* : c'est une méthode très récente. Elle est proposée par (Schmidt *et al.*, 2010) et est développée pour le protocole Ethernet/IP. Les auteurs considèrent des paquets de longueur minimale et dont les destinations sont inconnues (tout le temps émis en broadcast). Les auteurs calculent les longueurs maximales des files d'attente sur chaque port de sortie, i.e. le nombre de paquets transitant sur ce port et susceptibles de s'y trouver en

même temps. De là, ils construisent un graphe pondéré dont les sommets sont les nœuds du réseau et le poids de chaque arc est le délai de traversée (longueur de file d'attente convertie en délai) d'un nœud à un autre (Fig. 1.16). Le problème de calcul du pire délai de bout-en-bout revient alors à déterminer la trajectoire de longueur maximale sur le graphe.

- *Approche mixte (analytique et simulation)* : (Fan *et al.*, 2008) ont proposé un modèle analytique d'un réseau Ethernet commuté mais n'ont pas réussi à trouver une expression de calcul direct du délai de bout-en-bout. Ils ont alors proposé un algorithme itératif permettant de dérouler le fonctionnement du réseau et déterminer les longueurs des files d'attente à tout instant. En conservant en mémoire les longueurs maximales atteintes, les auteurs déterminent les délais maximaux de bufferisation. Un majorant du délai de bout-en-bout est alors déduit. Cette méthode a été comparée au calcul réseau et semble donner de meilleurs résultats dans différents exemples. Cela a été prouvé quand il s'agit d'un réseau avec un seul commutateur mais n'est pas prouvé dans le cas général.

1.4 Positionnement et contributions

1.4.1 Avantages et inconvénients des méthodes existantes :

Un constat évident peut être fait après ce parcours rapide des méthodes existantes d'évaluation des performances temporelles des SCR. La majorité des études, notamment analytiques, s'intéressent exclusivement aux délais de bout-en-bout. Or, comme cela a été mis en évidence précédemment, ces délais ne constituent qu'une partie du temps de réponse des SCR.

En ce qui concerne les méthodes énumérées précédemment et qui s'intéressent au temps de réponse des SCR, nous pouvons pointer les inconvénients suivants :

- **Simulation** : cette méthode permet d'évaluer l'allure de la distribution (histogramme) du temps de réponse même si elle ne donne pas la fonction exacte de cette distribution. Si cela n'est pas très grave en effet, le plus gros inconvénient de la simulation reste sa non-exhaustivité. Elle ne garantit pas de trouver les bornes réelles du temps de réponse puisqu'elle n'est pas exhaustive et ne permet pas d'être certain de balayer tous les cas possibles, notamment les cas critiques.
- **Mesure expérimentale** : on peut lui reprocher le même problème de manque d'exhaustivité même si la multiplication du nombre de mesures permet d'augmenter les chances de balayer les cas extrêmes. Cependant, cette méthode n'est pas toujours possible puisqu'intervenir sur une plateforme industrielle, surtout si celle-ci est en

marche, n'est pas toujours aisé. Le coût d'une telle opération, si besoin d'arrêter le système est, serait insupportable. Celui des outils de mesures l'est tout autant !

- **Vérification formelle :** la vérification formelle permet de remédier au problème d'exhaustivité. Malheureusement, cette méthode souffre du problème d'explosion combinatoire. Le nombre d'états qu'elle génère, même dans des SCR assez simples, devient très vite insupportable et la mémoire des ordinateurs de calcul saturée (Ruel *et al.*, 2008a). Quand le calcul arrive à terme, la durée peut atteindre des jours ! Des tentatives de résolution de ce problème ont été apportées notamment avec l'augmentation du niveau d'abstraction des SCR et de leur modélisation (Ruel *et al.*, 2008b). Si la méthode apporte des améliorations sur quelques exemples, le problème reste malheureusement non résolu même sur des cas simple. Les calculs n'arrivent pas toujours au bout (Ruel *et al.*, 2008b) !
- **Méthodes analytiques :** comme nous pouvons le constater, seuls deux exemples, (Tovar *et al.*, 2001) et (Vitturi, 2001), ont été trouvés s'agissant d'évaluer des performances globales (temps de réponse ou temps de cycle) des SCR avec des approches analytiques. De plus, ces exemples concernent des SCR dont les caractéristiques sont quasi-statiques puisque les stratégies d'ordonnancement sont connues a priori. D'ailleurs, ces SCR sont construits à base de protocoles dédiés et dont les performances temporelles sont assez bien cernées par construction. Une approche analytique dans un cadre plus large reste donc à trouver ...

1.4.2 Positionnement, contributions et démarche entreprise

Naturellement, notre objectif dans cette thèse est de contribuer à la résolution des problèmes exposés précédemment. Nous avons retenu une approche analytique et avons adopté la démarche représentée étape par étape sur le schéma synoptique de la Fig. 1.17.

Comme nous pouvons le voir sur le schéma, la démarche peut se décomposer en trois parties : la partie principale qui est au milieu et qui constitue l'ossature de la thèse, la partie à droite dédiée au calcul des délais de bout-en-bout et enfin la partie à gauche, consacrée à la validation expérimentale des différents résultats obtenus.

Dans la partie principale, nous modélisons les SCR à l'aide de graphes d'événements temporisés. Nous proposons un modèle générique d'un SCR constitué de M contrôleurs et de N modules déportés E/S. Nous déduisons les équations $(\max,+)$ linéaires du modèle et les utilisons pour évaluer le temps de réponse. Nous proposons deux approches d'analyse pour cela : une première approche déterministe pour évaluer les bornes, minimale et maximale, ainsi que l'allure de la distribution du temps de réponse (histogramme). Des formules

paramétrées de calcul direct sont données. La seconde approche est stochastique et permet de trouver la distribution exacte (la fonction de densité de probabilité) du temps de réponse. L'avantage de ces deux approches est qu'elles assurent l'exhaustivité de calcul du temps de réponse sans pour autant induire des durées ou des ressources de calcul conséquentes puisqu'il suffit de remplacer chaque paramètre des formules analytiques par sa valeur numérique.

A ce stade, nous pouvons évaluer les performances temporelles d'un SCR donné à condition de pouvoir instancier, en remplaçant les paramètres par leurs valeurs numériques, les formules paramétrées obtenues. Pour ce faire, nous avons besoin des valeurs, inconnues a priori, de certains délais de bout-en-bout, subis par certains messages échangés dans le SCR. C'est le but de la branche située à droite du schéma. Ces délais sont ceux subis lors de la traversée du seul réseau de communication.

Nous nous penchons alors sur le cas d'un réseau à base d'Ethernet et le modélisons à l'aide de réseaux de graphes d'événement temporisés en conflit (GETC) que nous représentons par la suite à l'aide d'équations $(\max,+)$. Le déroulement itératif de ces équations permet le calcul des délais de bout-en-bout. Par ailleurs, nous montrons que les GETC et leurs représentation $(\max,+)$ peuvent être utilisés aisément pour la modélisation de nombreux systèmes pratiques impliquant des ressources partagées (ex : systèmes de production), et pas seulement les réseaux de communication. Cela représente également une extension de l'utilisation de l'algèbre $(\max,+)$, usuellement réservée aux graphes d'événements temporisés dont l'utilisation est limitée aux systèmes sans conflits (sans ressources partagées). Les paramètres des formules obtenues dans la partie principale étant tous connus, nous pouvons à présent évaluer analytiquement le temps de réponse sur un cas d'étude.

Finalement, pour valider les résultats théoriques obtenus dans cette thèse, nous menons une campagne de mesures expérimentales sur de vrais SCR. C'est le but de la partie à gauche du schéma. Nous comparons les résultats théoriques à de nombreuses mesures réalisées dans différentes conditions. Le temps net cumulé est d'environ quarante heures d'acquisition et de traitement de données, soit environ un million de temps de réponse mesurés, étendu sur plusieurs jours. Les critères de comparaison retenus sont la précision pour les bornes du temps de réponse et le taux de recouvrement et l'allure pour la distribution.

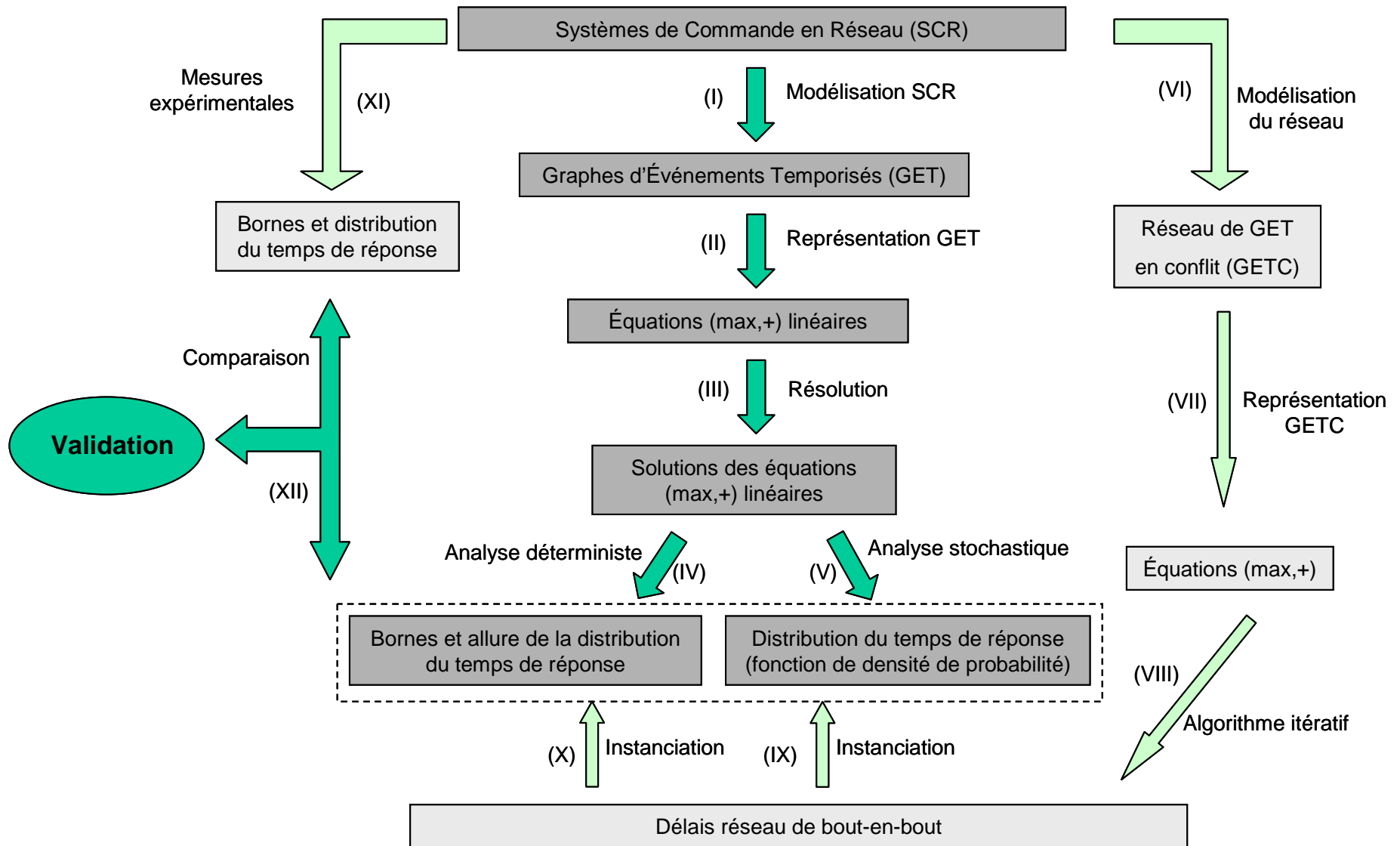


Fig. 1.17 : Démarche des travaux de la thèse

Fonctionnement & Modélisation des SCR

Sommaire

2.1 Fonctionnement des SCR Client/Serveur	34
2.1.1. Architecture et composition des SCR	34
2.1.1.1. Fonctionnement des contrôleurs : mode périodique ou mode cyclique ..	34
2.1.1.2. Fonctionnement des modules d'E/S déportés	36
2.1.1.3. Réseau de communication : Hypothèse	37
2.1.2. Hypothèses sur le fonctionnement des SCR considérés	37
2.2 Modélisation en Graphes d'Événements Temporisés (GET) et représentation (max,+) linéaire des SCR	38
2.2.1. Rappels sur les GET	38
2.2.2. Représentation d'état linéaire des GET l'algèbre (max,+)	40
2.2.3. Application aux SCR	42
2.2.3.1 Cas 1 : Mono Client Mono Serveur	43
2.2.3.2 Cas 2 : Mono Client Multiserveur	48
Cas 3 : Multi Client Multiserveur	51

Dans ce deuxième chapitre, nous décrivons la composition détaillée et le fonctionnement des composants principaux des SCR Client/Serveur : les contrôleurs (CPU et interface de communication compris), les modules E/S déportés (MES) et le réseau de communication. Nous précisons les hypothèses retenues relatives au fonctionnement de chaque composant. De là, nous passons à la modélisation des SCR en utilisant des Graphes d'Événements Temporisés (GET) puis à leur représentation en équations (max,+) linéaires. Mais avant cela, nous rappelons brièvement quelques notions fondamentales sur le dioïde (max,+) et son utilisation pour représenter la dynamique des GET.

~ As far as the laws of mathematics refer to reality, they are not certain,
as far as they are certain, they do not refer to reality. ~
- Albert Einstein

2.1 Fonctionnement des SCR Client/serveur

2.1.1 Architecture et composition des SCR Client/serveur

Les SCR Client/serveur sont composés principalement des éléments suivants : les contrôleurs (clients) et les modules déportés d'E/S (serveurs). Ces éléments sont interconnectés à travers un réseau de communication. Ce dernier peut prendre des formes diverses et variées (ligne, étoile, arbre, ...) et peut être constitué d'un ou plusieurs composants réseau (commutateurs, concentrateurs, câbles, ...).

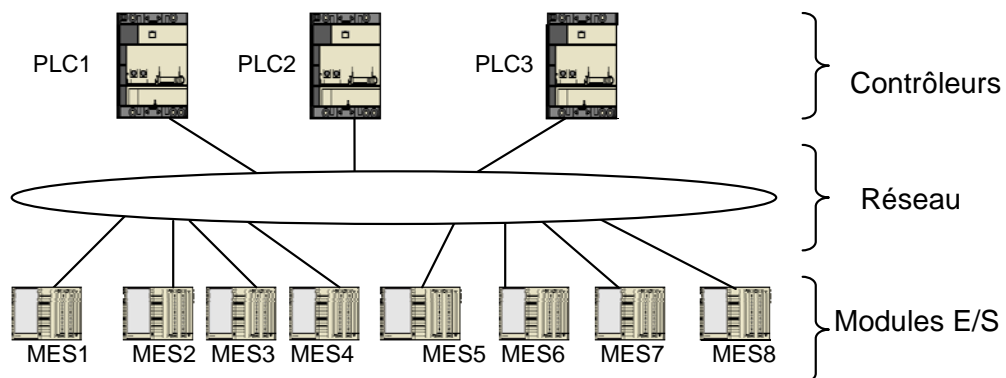


Fig. 2.1 : Exemple de système de commande en réseau ou SCR

2.1.1.1 Fonctionnement des contrôleurs : mode périodique ou mode cyclique

La fonction principale d'un contrôleur est naturellement l'exécution d'un programme de commande, implémenté par l'utilisateur, afin de générer les signaux de commande en fonction des informations renvoyées par les modules E/S déportés (MES). Le module qui se charge de cette fonction est un processeur de calcul CPU (Central Processing Unit). Aussi, pour transmettre ces signaux aux modules déportés ou en recevoir d'autres, un contrôleur contient une carte de communication (COM). Elle sert d'interface entre le CPU et le réseau de communication. Un contrôleur dans un SCR est finalement modulaire, avec un module de calcul et un module de communication (Fig. 2.2). De plus, ces deux modules ne sont pas synchronisés et fonctionnent indépendamment l'un de l'autre. Deux mémoires tampons, pour échanger les données des E/S de lecture et d'écriture, sont le seul lien physique qui existe entre les deux.

Le CPU peut fonctionner suivant deux modes, cyclique ou périodique. En mode périodique, la période étant T_{CPU} , le CPU suit les étapes suivantes : lecture des entrées, exécution du programme utilisateur, écriture des sorties puis attente de fin de période. Il commence donc par lire les images des entrées depuis la mémoire tampon 2 et les utilise pour exécuter le programme de commande. Il remet alors à jour les images des sorties qu'il écrit dans la

mémoire tampon 1. De là, il attend jusqu'à ce que la période de cycle T_{CPU} soit écoulée puis recommence un nouveau cycle de lecture, calcul, écriture et attente.

La seule différence entre le mode cyclique et le mode périodique est que le CPU n'attend pas que la période soit écoulée pour commencer un nouveau cycle (l'étape « Attente fin période » sur le schéma de la Fig. 2.2 n'existe pas). Juste après la fin de la phase d'écriture des sorties, le CPU commence la lecture des entrées.

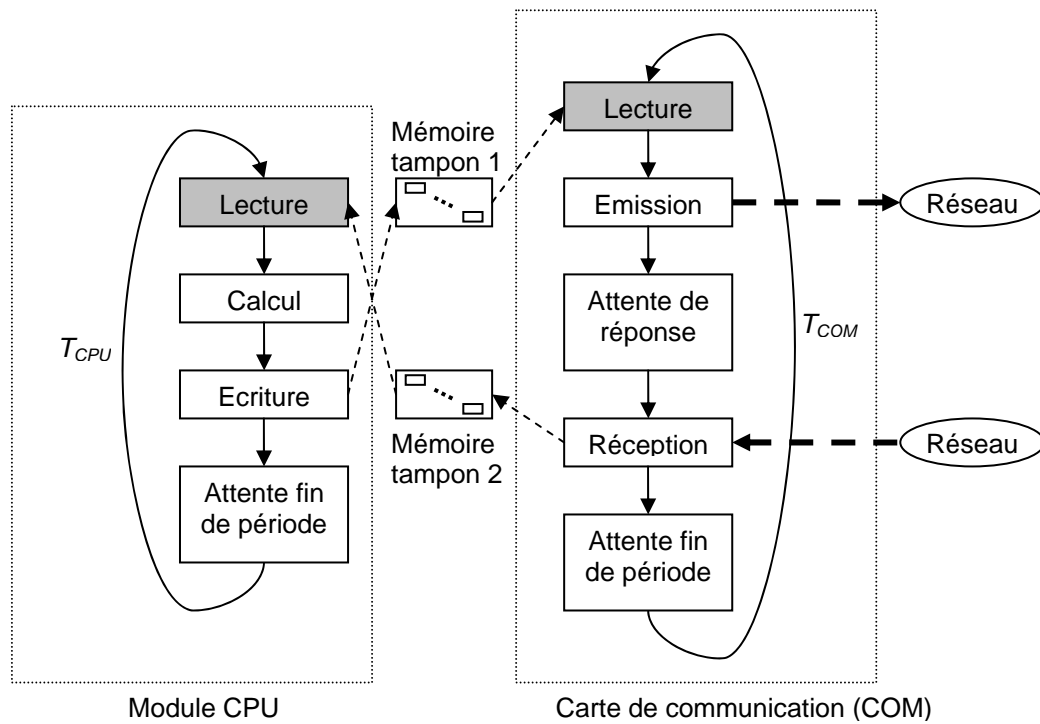


Fig. 2.2 : Constitution modulaire d'un contrôleur et son fonctionnement

La carte de communication COM fonctionne quant à elle toujours périodiquement (hypothèse retenue dans notre étude). En début de cycle, elle lit en bloc les valeurs des entrées de la mémoire tampon 1 puis commence à les émettre vers les modules déportés E/S. C'est ce qu'on appelle la scrutation ou le *I/O scanning*. L'ordre de scrutation des modules déportés est invariant dans le temps et fixé par l'utilisateur. La COM émet des requêtes contenant les signaux de commande les unes après les autres et attend les réponses. A chaque fois qu'une réponse est reçue, elle est copiée dans la mémoire tampon 2. En cas de réception de plusieurs réponses en rafale, elles sont mises en attente dans un buffer et traitées en mode FIFO (first in first out). Une fois que toutes les réponses sont reçues, la COM se met en phase d'attente jusqu'à ce que la période de scrutation T_{COM} soit écoulée. La COM commence alors un nouveau cycle de scrutation.

Notons que l'absence de synchronisation entre les deux modules, le CPU et la COM, entraîne des délais supplémentaires dans le temps de réponse. En effet, lorsque les sorties sont mises à jour par le CPU, elles ne sont pas prises en compte immédiatement par la COM, pour être envoyées vers les modules déportés, mais attendent le début d'un nouveau cycle de scrutation. De même, lors de la réception d'une réponse par la COM, celle-ci n'est pas immédiatement utilisée par le CPU dans le programme utilisateur mais attend également le début d'un nouveau cycle CPU. Ce phénomène est évidemment à prendre en compte et comme nous le verrons par la suite, son impact est loin d'être négligeable. Au contraire !

2.1.1.2 Fonctionnement des modules déportés d'E/S

Le fonctionnement des modules d'E/S déportés est plus simple. Un MES est continuellement en état d'attente jusqu'à la réception d'une requête envoyée par un contrôleur. En cas de réception d'une rafale de requêtes, ces dernières sont mises en attente et traitées en mode FIFO. Le fonctionnement est cyclique et le cycle de traitement d'une requête en attente débute immédiatement après la fin du cycle précédent. Chaque requête véhicule une donnée utile, destinée pour une opération de lecture d'une entrée (ex : lecture de la valeur d'un capteur) ou d'écriture d'une sortie (ex : changement de l'entrée d'un actionneur) ou bien la combinaison des deux. Notons qu'un signal en provenance d'un capteur subit un filtrage (conversion analogique/numérique) avant d'être pris en compte par le MES. La même remarque peut être formulée concernant le signal en partance vers un préactionneur. Ces deux opérations introduisent aussi des délais supplémentaires dans le temps de réponse.

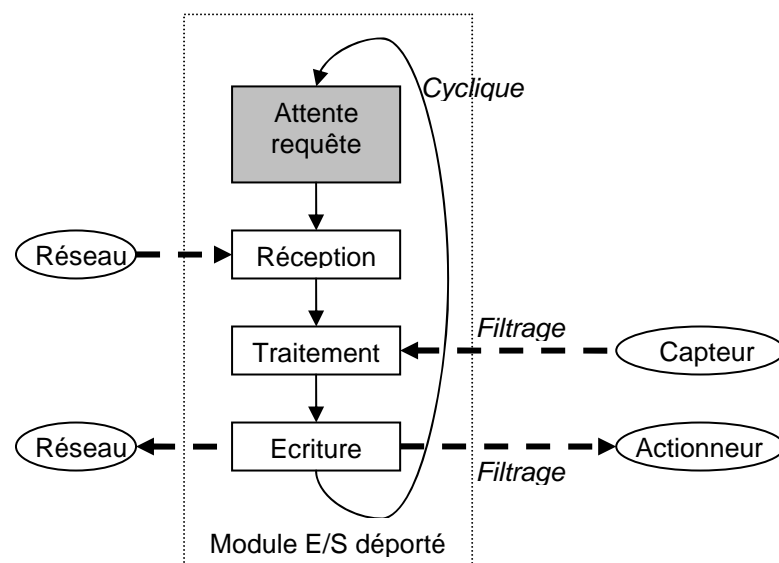


Fig. 2.3 : Fonctionnement d'un module d'E/S déporté

2.1.1.3 Réseau de communication : Hypothèse

Contrairement aux travaux existant dans la littérature, nous adoptons un niveau d'abstraction qui permet de séparer la partie applicative de la partie communication. Nous considérons que le réseau est un élément du SCR mais sans entrer dans le détail de sa composition : une boîte noire. Avec ce haut niveau d'abstraction, nous ne précisons ni le type, ni la technologie, ni même la composition du réseau utilisé pour la communication entre les contrôleurs et les serveurs. Nous supposons toutefois que le réseau est compatible avec le mode Client/serveur exposé précédemment et les délais qu'il induit sont bornés. A vrai dire, la chose qui nous importe le plus à ce stade est la contribution du réseau en terme de délai dans le temps de réponse. Nous supposons que le réseau de communication introduit dans le temps de réponse des *délais variables*, que nous appelons des *délais de bout-en-bout*, et que nous représenterons avec des paramètres. Ce sont les délais subis lors de la traversée du seul réseau de communication. Cette démarche nous permet d'obtenir des modèles et des résultats analytiques génériques (paramétrés) quelque soit le nombre de clients et de serveurs dans le SCR. Mieux encore, les paramètres représentant la composante variable qu'est le réseau de communication peuvent être calculés séparément puis injectés dans les formules obtenues pour évaluer le temps de réponse. C'est justement l'objet du Chapitre 4 en se penchant sur le cas particulier des délais de bout-en-bout dans les réseaux Ethernet commuté.

Ainsi, une fois que les résultats analytiques sont obtenus, une fois pour toute, la complexité de l'évaluation du temps de réponse d'un SCR se réduit à la complexité d'évaluation des seuls délais de bout-en-bout.

2.1.2 Hypothèses sur le fonctionnement des composants des SCR

Les hypothèses suivantes sont posées dans nos travaux :

- H1 : en mode périodique, la période du CPU d'un contrôleur reste constante durant toute la durée de fonctionnement du système. Rappelons que cette période est un paramètre réglé par l'utilisateur et elle ne peut être choisie que parmi un nombre fini de valeurs discrètes. De ce fait, nous pouvons supposer que la période T_{CPU} est un entier naturel. Les mêmes hypothèses sont posées pour la période T_{COM} . Elle est constante et appartient aux entiers naturels. Le rapport entre les deux est alors un nombre rationnel positif ($T_{COM} / T_{CPU} \in \mathbb{Q}^+$).
- H2 : les temps de lecture/écriture des données dans les mémoires tampons 1 et 2 sont négligés. En effet, ces temps ne sont que de l'ordre de la microseconde puisque seules

les données utiles de seulement quelques octets, de la couche application et qui correspondent aux valeurs des signaux des entrées et des sorties, sont manipulées durant ces deux phases. Les capacités des tampons sont également largement suffisantes pour éviter toute perte de données.

- H3 : les pannes des équipements et les pertes de paquets ne sont pas considérées. Rappelons sur ce point que lors de nos expérimentations, des observations sur pas moins de 5 millions de requêtes/réponses échangées, pas un seul paquet n'a été perdu. Cette hypothèse est souvent posée dans les SCR. Les paquets échangés dans les SCR sont en effet de petite taille vu que les données utiles le sont aussi. Aussi, même si les SCR sont parfois connectés à d'autres sous réseaux véhiculant du trafic non temps-réel lourd, le trafic entre ces deux domaines reste limité.
- H4 : les modules E/S déportés ne fonctionnent qu'en serveurs et ne peuvent émettre de message de manière autonome.
- H5 : pour le bon fonctionnement de la commande, la période du CPU est fixée de sorte qu'elle soit toujours suffisante, tout le temps, pour accomplir les phases de lecture, calcul et écriture. De simples manipulations pratiques permettent de s'assurer de cela.
- H6 : la période de scrutation T_{COM} est fixée de sorte que toutes les réponses aux requêtes émises lors d'un cycle soient reçues avant la fin de ce même cycle. Le cas contraire engendrerait des variations de la période de scrutation non désirables.
- H7 : les différents contrôleurs du SCR sont indépendants les uns des autres. Ils ne sont pas synchronisés et chacun fonctionne avec sa propre période d'I/O scanning T_{COM} . Chaque contrôleur peut également commencer à émettre ses requêtes vers les E/S qu'il scrute indépendamment des autres.

2.2 Modélisation en Graphes d'Événements Temporisés (GET) et représentation $(\max,+)$ linéaire des SCR

2.2.1 Rappels sur l'algèbre $(\max,+)$

Avant d'aborder la représentation $(\max,+)$ linéaire des GET, il convient de faire quelques rappels brefs sur les notions fondamentales et propriétés des structures algébriques dites dioïdes.

Définition 2.1 (Demi-anneau). On appelle Demi-anneau un ensemble D muni de deux lois internes \oplus et \otimes telles que :

- la loi additive \oplus est associative, commutative et admet un élément neutre, noté ε , tel que : $\forall a \in D, a \oplus \varepsilon = \varepsilon \oplus a = a$
- la loi multiplicative \otimes est associative et admet un élément neutre, noté e , tel que : $\forall a \in D, a \otimes e = e \otimes a = a$
- la loi \otimes est distributive par rapport à la loi additive : $\forall a, b, c \in D, (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$
 $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$
- l'élément neutre de la loi additive ε est absorbant pour la multiplication : $\forall a \in D, a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon$

Définition 2.2 (Dioïde). Un dioïde est un Demi-anneau dont la loi additive \oplus est idempotente : $\forall a \in D, a \oplus a = a$;

Un dioïde est dit *commutatif* si la loi \otimes est commutative : $\forall a, b \in D, a \otimes b = b \otimes a$;

Un dioïde est dit *complet* s'il est fermé pour les sommes infinies et la loi de distributivité définie précédemment s'étend aussi aux sommes infinies.

Notons que le terme puissance de a dans les dioïdes, noté a^k , désigne le produit $\underbrace{a \otimes \dots \otimes a}_{k \text{ fois}}$

et $a^0 = e$.

Exemple 2.1 : on peut aisément vérifier que l'ensemble $\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$ muni des deux lois, maximum et addition classique, additive et multiplicative respectivement est un dioïde commutatif. Il est le plus souvent appelé *Algèbre (max, +)*. L'élément neutre de l'opérateur maximum est $\varepsilon = -\infty$ et celui de l'addition classique est $e = 0$.

Exemple 2.2 (Dioïde matriciel). L'ensemble des matrices carrées de dimension n , à coefficients dans le dioïde scalaire (D, \oplus, \otimes) est un dioïde matriciel, noté $(D^{n \times n}, \oplus, \otimes)$ où les opérations sont définies comme suit :

$$\forall A, B \in D^{n \times n}, (A \oplus B)_{ij} = A_{ij} \oplus B_{ij}, \quad \forall i, j = 1, \dots, n$$

$$\forall A, B \in D^{n \times n}, (A \otimes B)_{ij} = \bigoplus_{k=1}^n A_{ik} \otimes B_{kj}, \quad \forall i, j = 1, \dots, n$$

L'élément identité de $D^{n \times n}$ est la matrice de dimension n , notée id_n , dont la diagonale n'est composée que de e . Cette matrice id_n contient des ε partout ailleurs.

L'élément zéro est la matrice composée entièrement de ε .

Théorème 2.1 (Baccelli *et al.*, 1992b, p. 189). *Les matrices A, B étant deux éléments du dioïde complet $(\mathbb{R}_{\max}^{n \times n}, \oplus, \otimes)$, l'équation suivante : $x = A \otimes x \oplus B$ admet une solution minimale donnée par : $x = A^* \otimes B$ où la matrice $A^* = \bigoplus_{k \geq 0} A^k$ est appelé matrice de Kleene de A .*

Théorème 2.2 (Baccelli *et al.*, 1992b, p. 148). *La matrice relative à un graphe fortement connexe est une matrice irréductible. Elle admet une valeur propre unique.*

Théorème 2.3 (Baccelli *et al.*, 1992b, p. 148). *Pour une matrice irréductible A , il existe deux entiers K et c tels que : $\forall k \geq K$, $A^{k+c} = \lambda^c \otimes A^k$;*

λ est l'unique valeur propre de A et le nombre minimal c vérifiant cela est appelé la cyclicité de A .

Faisons remarquer que nous avons rappelé brièvement uniquement les bases de l'algèbre $(\max, +)$ et les théorèmes qui seront utiles dans la thèse. Pour de plus amples détails, nous recommandons au lecteur le livre de Baccelli *et al.* (1992b).

2.2.2 Représentation d'état linéaire des GET dans l'algèbre $(\max, +)$

La théorie des réseaux de Petri (ou RdP) s'intéressait à l'origine exclusivement à l'étude de l'ordre d'occurrence des événements dans les systèmes modélisés et non aux dates de leurs occurrences (Murata, 1989). Pour des problèmes d'évaluation de performances – par exemple, quelle est la cadence de production dans un système? – l'introduction de la notion de *temps* est nécessaire. Pour ce faire, des temporisations sont associées, soit aux transitions (le RdP est alors dit T- temporisé), soit aux places (P- temporisé). Une temporisation associée à une transition correspond au temps nécessaire pour son franchissement. Une temporisation associée à une place correspond au temps de séjour d'un jeton entrant dans la place avant qu'il soit disponible pour le tir des transitions en aval de cette même place. Dans le cas général, l'introduction du temps complexifie l'analyse des RdP. La sous classe de RdP appelée Graphes d'Evénements Temporisés (GET) offre quant à elle des capacités d'analyse plus aisée grâce à l'algèbre $(\max, +)$. Ainsi, des problèmes d'évaluation de performances (Hillion *et al.*, 1989), de commande (Houssin *et al.*, 2007), de supervision d'automates et GET (Komenda *et al.*, 2009), ... ont été résolus.

- **Définition** : un graphe d'événements temporisés (GET) est un réseau de Petri ordinaire temporisé où chaque place a exactement une transition en amont et une transition en aval au maximum (Fig. 2.4).

Les GET permettent de représenter des phénomènes de synchronisation et de parallélisme mais pas de conflit ou de partage de ressources. Outre leur simplicité, l'avantage des GET est qu'ils peuvent être représentés par des équations faciles à manipuler.

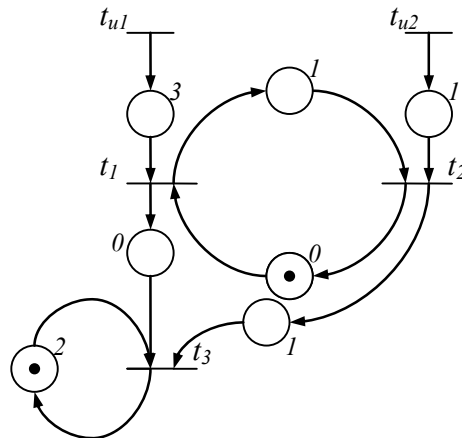


Fig. 2.4 : Exemple de GET

Pour obtenir ces équations, on associe à chaque transition t_i une variable $x_i(k)$, appelée *dateur*, qui représente la date de son franchissement pour la $k^{\text{ème}}$ fois. Le dateur associé à une transition d'entrée t_{uj} est noté $u_j(k)$.

Exemple 2.3 :

Supposons que le franchissement des transitions se fait à vitesse maximale c.-à-d. dès que le franchissement est possible. Intéressons nous par exemple à la transition t_1 du GET de la Fig.

2.4. Son franchissement pour la $k^{\text{ème}}$ fois est possible sous deux conditions :

i) Il existe un $k^{\text{ème}}$ jeton dans la place en aval de u_1 et celui-ci est disponible. Cette date de disponibilité correspond à 3 unités de temps après le franchissement de la transition u_1 pour la $k^{\text{ème}}$ fois.

ii) Il existe un $(k-1)^{\text{ème}}$ jeton dans la place liant t_2 à t_1 et celui-ci est disponible. Autrement dit, juste après le franchissement de la transition t_2 pour la $(k-1)^{\text{ème}}$ fois. Si la transition t_1 est franchie dès que possible (à vitesse maximale), nous pouvons alors écrire :

$$x_1(k) = \max[u_1(k) + 3, x_2(k-1) + 0]$$

De la même manière nous pouvons écrire les équations relatives aux autres transitions et obtenir :

$$\begin{cases} x_1(k) = \max[u_1(k) + 3, x_2(k-1) + 0] \\ x_2(k) = \max[u_2(k) + 1, x_1(k) + 1] \\ x_3(k) = \max[x_1(k) + 0, x_2(k) + 1, x_3(k-1) + 2] \end{cases}$$

Ces équations impliquent les deux lois du dioïde $(\mathbb{R}_{\max}, \oplus, \otimes)$, le maximum qu'on a précédemment noté \oplus et l'addition notée \otimes .

La réécriture des équations précédentes en utilisant ces opérateurs donne :

$$\begin{cases} x_1(k) = 3 \otimes u_1(k) \oplus 0 \otimes x_2(k-1) \\ x_2(k) = 1 \otimes u_2(k) \oplus 1 \otimes x_1(k) \\ x_3(k) = 0 \otimes x_1(k) \oplus 1 \otimes x_2(k) \oplus 2 \otimes x_3(k-1) \end{cases}$$

Ce système d'équations peut également être exprimé sous forme matricielle (pour rappel, $e = 0$) :

$$x(k) = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon \\ e & 1 & \varepsilon \end{pmatrix} \otimes x(k) \oplus \begin{pmatrix} \varepsilon & e & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 2 \end{pmatrix} \otimes x(k-1) \oplus \begin{pmatrix} 3 & \varepsilon \\ \varepsilon & 1 \\ \varepsilon & \varepsilon \end{pmatrix} \otimes u(k) ;$$

$$\text{où : } X(k) = (x_1(k) \quad x_2(k) \quad x_3(k))' \text{ et } U(k) = (u_1(k) \quad u_2(k))'$$

Cette expression est une représentation d'état sous la forme :

$$X(k) = A_0 \otimes X(k) \oplus A_1 \otimes X(k-1) \oplus B_0 \otimes U(k)$$

La structure $(\mathbb{R}_{\max}, \oplus, \otimes)$ étant un dioïde, cette équation peut aussi être mise sous la forme canonique standard en utilisant le Théorème 2.1. Nous obtenons alors :

$$X(k) = A \otimes X(k-1) \oplus B \otimes U(k)$$

$$\text{où : } A_0^* = \begin{pmatrix} e & \varepsilon & \varepsilon \\ 1 & e & \varepsilon \\ 2 & 1 & e \end{pmatrix}, A = A_0^* A_1 = \begin{pmatrix} \varepsilon & e & \varepsilon \\ \varepsilon & 1 & \varepsilon \\ \varepsilon & 2 & 2 \end{pmatrix} \text{ et } B = A_0^* B_0 = \begin{pmatrix} 3 & \varepsilon \\ 4 & 1 \\ 5 & 2 \end{pmatrix}.$$

Dans le cas où le GET est non contraint (sans transitions d'entrée t_{ij}), l'équation devient :

$$x(k) = A \otimes x(k-1) = A^k \otimes x(0).$$

2.2.3. Application aux SCR

Dans cette section de l'étude, nous faisons le choix d'utiliser des GET P-Temporisés. Comme nous le verrons, ce choix permet de proposer des modèles assez faciles à appréhender car il

traduit naturellement et directement les modèles fonctionnels présentés précédemment dans la section 2.1.

2.2.3.1 Cas 1 : des SCR mono client – mono serveur

Nous commençons la modélisation par le cas le plus simple avec un SCR impliquant un seul contrôleur et un seul module E/S.

- **Contrôleur** : compte tenu du fonctionnement du contrôleur décrit dans la section 2.1, nous obtenons le modèle assez intuitif et naturel de la Fig. 2.5. A noter que le positionnement initial des jetons reflète le fonctionnement au démarrage du système c.-à-d. début d'un cycle de communication et début d'un cycle CPU à la date $t=0$.

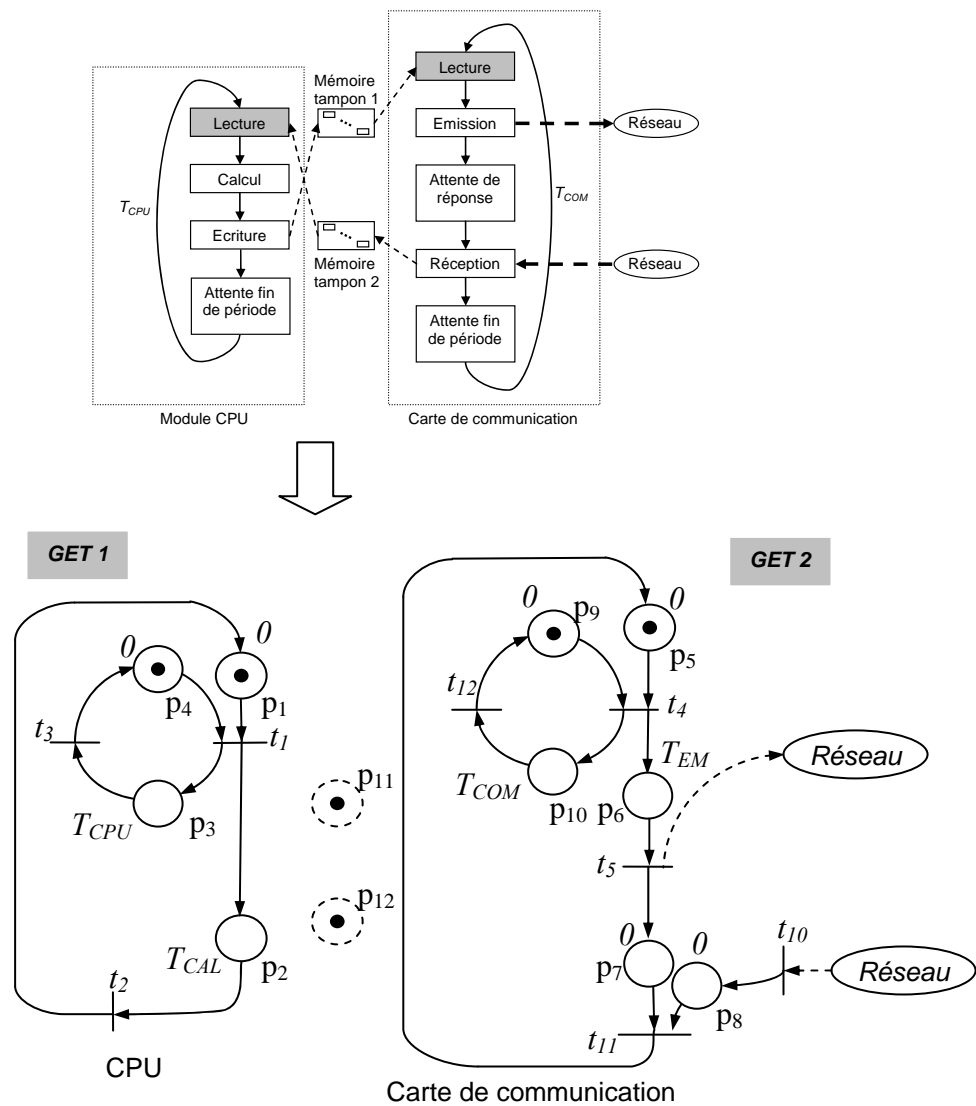


Fig. 2.5 : Modèle en GET du contrôleur : CPU et carte de communication compris

- Le franchissement de t_1 modélise le début d'un cycle CPU. Les phases lecture, calcul et écriture, représentées par la place p_2 , durent T_{CAL} unités de temps et se terminent avec le franchissement de la transition t_2 (mise à jour des sorties). En parallèle, un jeton entre dans la place p_3 , du fait de franchissement de t_1 , et y reste durant une durée égale à la période du CPU, soit T_{CPU} . Une fois cette durée écoulée, ce jeton quitte cette place et entre dans p_4 . Le CPU est alors prêt pour entamer un nouveau cycle puisque les places en amont de t_1 contiennent chacune un jeton disponible.
- Du côté de la carte de communication, le fonctionnement est très similaire. Le franchissement de t_4 modélise le début d'un cycle de scrutation et celui de t_5 la fin de l'émission de la requête vers le module déporté. La génération d'un jeton dans p_7 représente l'entrée en phase d'attente de la réponse à la requête. Une fois que la réponse est reçue, un jeton entre dans p_8 et la transition t_{11} est immédiatement franchie. La COM entre dans une autre phase d'attente jusqu'à la fin d'écoulement de la période de scrutation T_{COM} , représentée cette fois pas la temporisation de la place p_{10} . Une fois l'attente terminée, la transition t_{12} est franchie et un jeton entre dans p_9 . Un jeton étant également présent dans p_5 , puisque la réponse est reçue avant la fin de la période T_{COM} , la COM commence un nouveau cycle de scrutation.

Remarque 2.2.3 : les mémoires tampon 1 et 2, liant le CPU à la COM, sont représentées par les places p_{11} et p_{12} . Comme nous pouvons le constater, elles ne sont pas reliées au reste du modèle. Nous aurions pu les connecter aux transitions t_1 et t_{11} . Nous ne l'avons pas fait pour la simple raison que les temps lecture/écriture dans les mémoires tampon sont négligés (Hypothèse H2). Par conséquent, les jetons des places p_{11} et p_{12} seraient toujours disponibles et donc inutiles dans le modèle.

- A noter qu'il est possible de réduire le modèle de la Fig. 2.5. Il est par exemple possible de supprimer la place p_4 sans affecter la dynamique du système. Nous l'avons gardée car le fonctionnement du système est mieux représenté et plus facile à comprendre comme cela. Un jeton dans la place p_4 signifie simplement de la fin de la période du CPU. Le CPU est ainsi prêt pour débiter un autre cycle.

- **Module E/S** : Le modèle du module déporté E/S est représenté par le GET de la Fig. 2.6.

Le module est nominalelement en phase d'attente tant qu'il n'y a pas de requête à traiter. Cette attente est représentée par le jeton dans la place p_{14} . Le franchissement de t_6 signifie l'arrivée d'une requête dans le buffer d'entrée du module (représenté pas la place p_{13}). Immédiatement après la réception de cette requête, l'attente est interrompue avec le franchissement de la transition t_7 et le traitement de la requête peut commencer. Ce processus dure $T_{E/S}$ unités de temps avant de se terminer avec le tir de la transition t_8 . La réponse résultant du traitement est renvoyée par le MES avec le tir de t_9 .

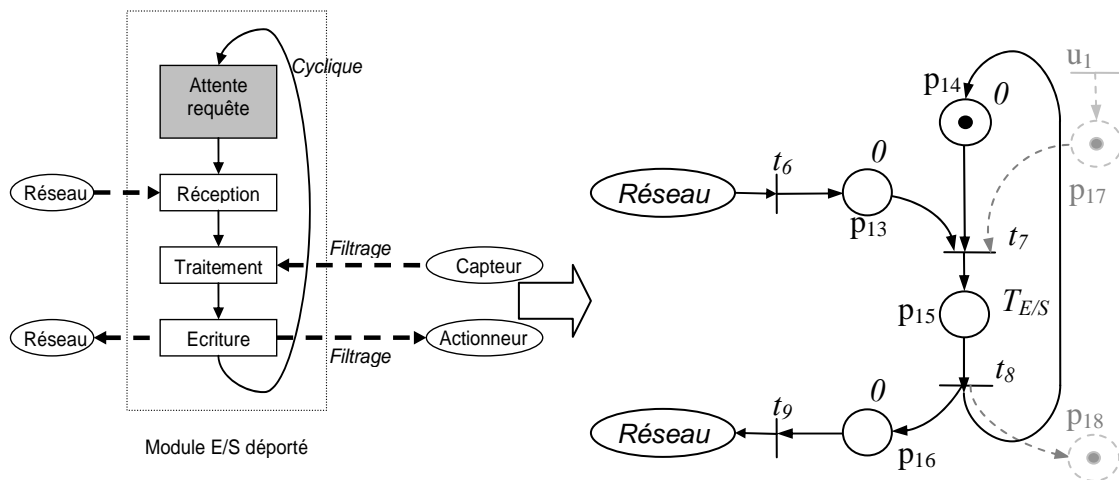


Fig. 2.6 : Modèle GET du module déporté E/S

- **Remarque 2.2.4** : remarquons que nous avons grisé une partie du modèle. Celle-ci représente la partie instrumentation (capteur et actionneur) mais nous ne la prenons pas en considération. La transition d'entrée u_1 aurait bien pu représenter le signal en provenance du capteur. Cependant, ceci donnerait un GET non autonome ou contraint. Or, nous savons que le module E/S traite toujours et immédiatement la requête qu'il reçoit en utilisant le signal capteur actuel quelle que soit sa valeur. Le modèle non contraint est alors suffisant pour représenter le fonctionnement du MES vis-à-vis du contrôleur. Toutefois, la valeur du signal du capteur, ou plus précisément son changement d'état qui représente un événement, nous intéressera par la suite lors du calcul du temps de réponse.

- **Réseau de communication** : Comme nous l'avions annoncé précédemment, le réseau de communication est représenté par des délais de bout-en-bout. Plus précisément deux délais : un délai que subit la requête lors de son envoi du contrôleur vers le MES et un autre que subit

la réponse dans le sens inverse (MES vers contrôleur). Les deux éléments de GET suivants modélisent ces deux délais :

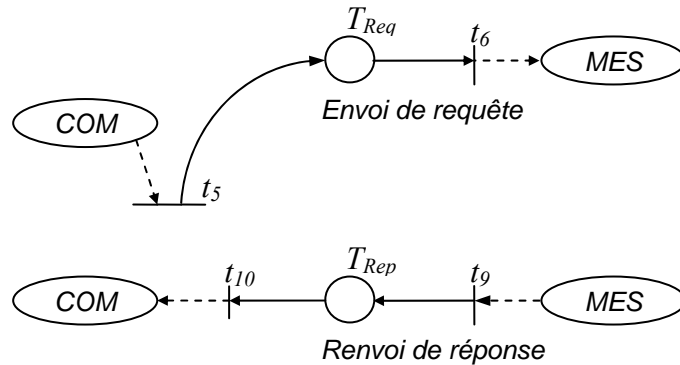


Fig. 2.7 : Délai de bout-en-bout (a) délai COM vers MES, (b) délai MES vers COM.

Comme nous l'avons déjà expliqué avec le modèle en GET de la COM, le franchissement de t_5 signifie la fin d'émission de la requête depuis le contrôleur. De là, la requête entre dans le réseau de communication et y subit un délai de bout-en-bout total T_{Req} . Le franchissement de t_6 modélise la sortie du réseau pour entrer dans le module E/S déporté. Dans le sens du retour, la réponse entre dans le réseau avec le franchissement de la transition t_9 et y subit un délai total T_{Rep} . Rappelons que ces délais de bout-en-bout sont *variables* d'un cycle de scrutation à un autre.

Au final, nous pouvons lier les modèles des différents composants pour reconstituer un modèle global du SCR, comme sur la Fig. 2.8.

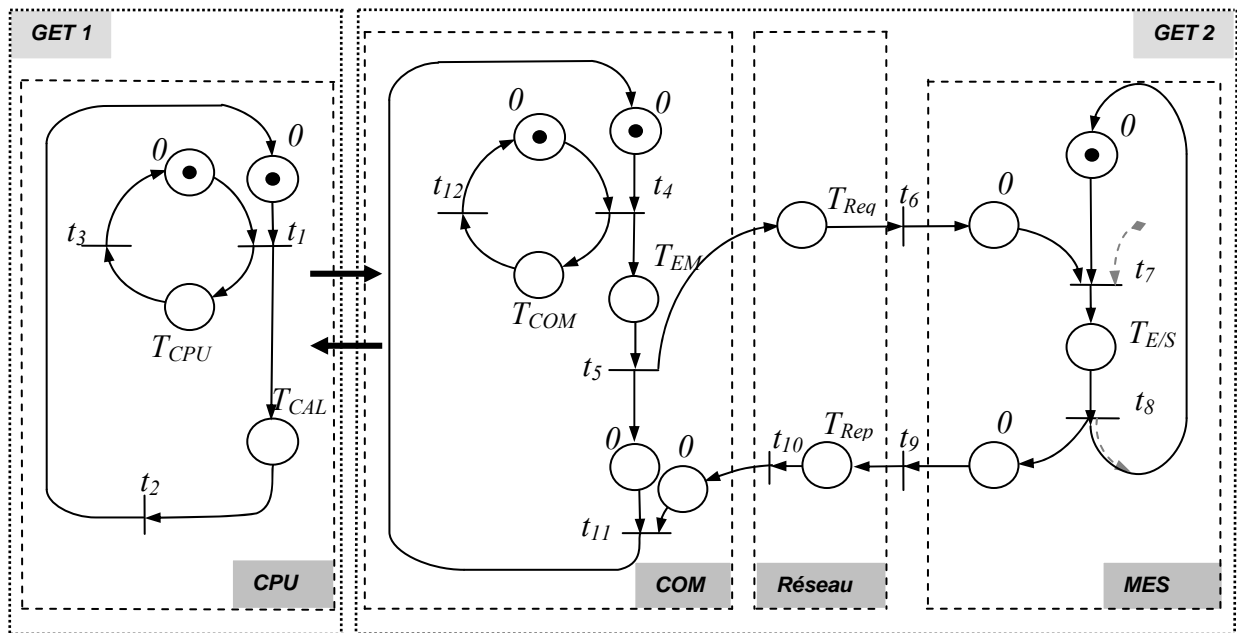


Fig. 2.8 : Modèle GET d'un SCR mono client - mono serveur

Représentation en équations (max,+) linéaires des GET

Le SCR est finalement modélisé avec deux GET communicants. Ils sont représentés par les deux systèmes d'équations (max,+) linéaires suivants :

$$\begin{cases} \theta_1(k) = e \otimes \theta_2(k-1) \oplus e \otimes \theta_3(k-1) \\ \theta_2(k) = T_{CAL} \otimes \theta_1(k) \\ \theta_3(k) = T_{CPU} \otimes \theta_1(k) \\ \theta_4(l) = e \otimes \theta_{11}(l-1) \oplus \theta_{12}(l-1) \otimes e \\ \theta_5(l) = T_{EM} \otimes \theta_4(l) \\ \theta_6(l) = T_{Req} \otimes \theta_5(l) \\ \theta_7(l) = e \otimes \theta_6(l) \oplus e \otimes \theta_8(l-1) \\ \theta_8(l) = T_{E/S} \otimes \theta_7(l) \\ \theta_9(l) = e \otimes \theta_8(l) \\ \theta_{10}(l) = T_{Rep} \otimes \theta_9(l) \\ \theta_{11}(l) = e \otimes \theta_5(l) \oplus e \otimes \theta_{10}(l) \\ \theta_{12}(l) = T_{COM} \otimes \theta_4(l) \end{cases}$$

Notons que nous avons associé deux indices différents, k et l , aux deux systèmes d'équations.

Ceci est dû au fait que les deux modules, le CPU et la COM, ne soient pas synchronisés.

Ces équations peuvent évidemment se mettre sous les formes standard suivantes :

$$\Theta_1(k) = A_1 \otimes \Theta_1(k-1)$$

$$\Theta_2(l) = A_2 \otimes \Theta_2(l-1)$$

où $A_1 = \begin{pmatrix} \cdot & e & e \\ \cdot & T_{CAL} & T_{CAL} \\ \cdot & T_{CPU} & T_{CPU} \end{pmatrix}$, $\Theta_1(k) = (\theta_1(k) \ \theta_2(k) \ \theta_3(k))^t$, $\Theta_2(k) = (\theta_4(k) \ \theta_5(k) \ \dots \ \theta_{12}(k))^t$, et

$$A_2 = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & e & e \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & T_{EM} & T_{EM} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & T_{Req} T_{EM} & T_{Req} T_{EM} \\ \cdot & \cdot & \cdot & e & \cdot & \cdot & T_{Req} T_{EM} & T_{Req} T_{EM} \\ \cdot & \cdot & \cdot & T_{E/S} & \cdot & \cdot & T_{E/S} T_{Req} T_{EM} & T_{E/S} T_{Req} T_{EM} \\ \cdot & \cdot & \cdot & T_{E/S} & \cdot & \cdot & T_{E/S} T_{Req} T_{EM} & T_{E/S} T_{Req} T_{EM} \\ \cdot & \cdot & \cdot & T_{Rep} T_{E/S} & \cdot & \cdot & T_{Rep} T_{E/S} T_{Req} T_{EM} & T_{Rep} T_{E/S} T_{Req} T_{EM} \\ \cdot & \cdot & \cdot & T_{Rep} T_{E/S} & \cdot & T_{EM} \oplus T_{Rep} T_{E/S} T_{Req} T_{EM} & T_{EM} \oplus T_{Rep} T_{E/S} T_{Req} T_{EM} & T_{EM} \oplus T_{Rep} T_{E/S} T_{Req} T_{EM} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & T_{COM} & T_{COM} \end{pmatrix} ;$$

Les représentations d'état s'écrivent aussi en utilisant l'état initial comme suit :

$$\Theta_1(k) = A_1^k \otimes \Theta_1(0)$$

$$\Theta_2(l) = A_2^l \otimes \Theta_2(0)$$

où $\Theta_1(0) = (\varepsilon \ e \ e)^t$ et $\Theta_2(0) = (\varepsilon \ \varepsilon \ \varepsilon \ \varepsilon \ e \ \varepsilon \ \varepsilon \ e \ e)^t$, les jetons des places marquées étant tous disponibles à l'instant initial. Ces dates sont choisies de la sorte pour modéliser le fait que le CPU et la COM commencent leurs cycles à l'instant initial.

2.2.3.2 Cas 2 : mono client – multi serveurs

Dans ce cas, N modules déportés sont interrogés par un seul contrôleur. Durant un cycle de scrutation, le contrôleur envoie une rafale de N requêtes (une requête après l'autre) vers ces modules dans un ordre invariant puis attend les réponses. Une fois que toutes les réponses sont reçues, il se met en attente jusqu'à la fin de la période de scrutation.

Remarquons que le module CPU est modélisé exactement comme dans le cas mono serveur. En effet, au début d'un cycle CPU, toutes les entrées sont lues en bloc (comme une seule variable et donc comme dans le cas mono serveur) et les sorties sont mises à jour en bloc également.

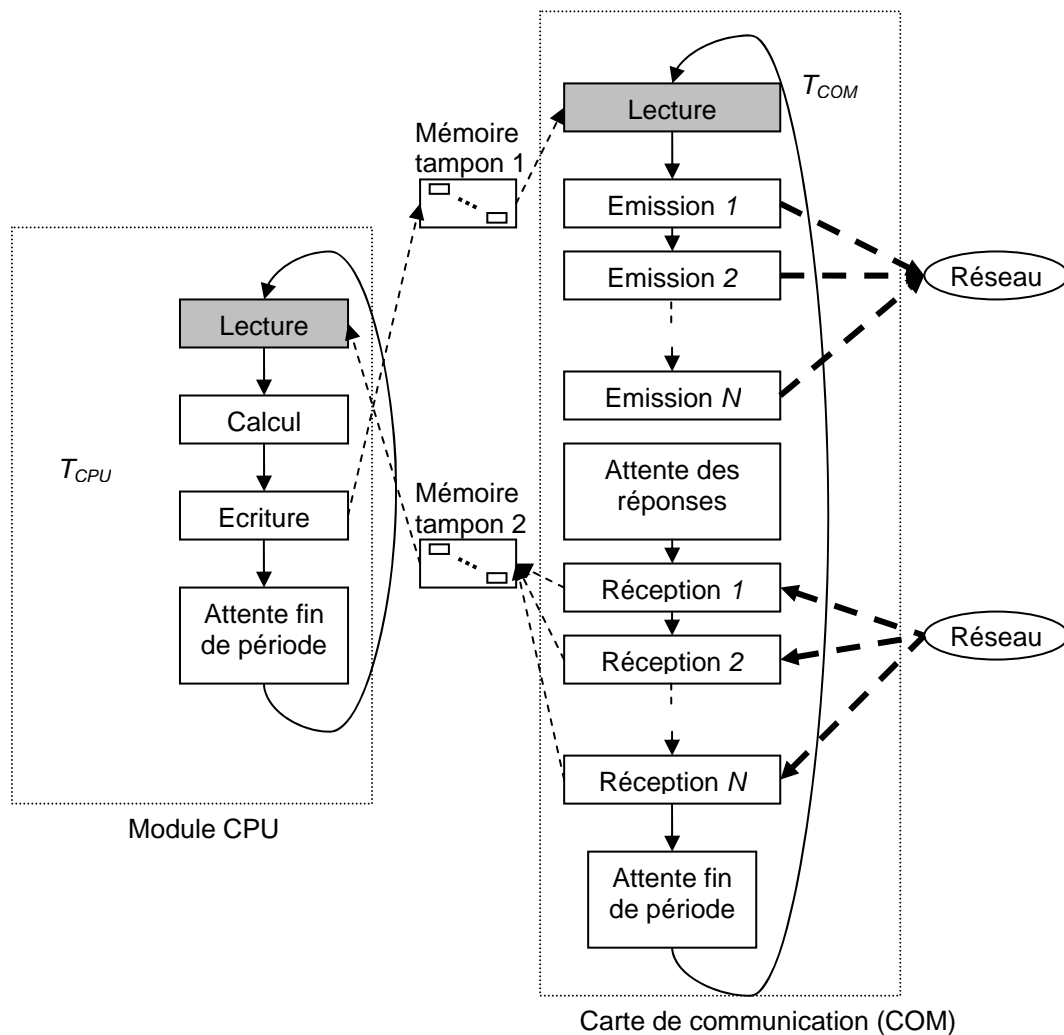


Fig. 2.9 : Fonctionnement d'un contrôleur interrogeant N module E/S

De manière similaire que dans le cas mono serveur, chaque requête/réponse échangée avec un serveur subit un délai lors de la traversée du réseau de communication. Chaque paquet étant affecté par un délai différent des autres, nous associons le délai de bout-en-bout T_{Req_i} à la $i^{ème}$ requête, envoyée vers le serveur numéro i noté MES_i . Sur la Fig. 2.10, seuls les délais *aller*, relatifs au requêtes sont représentés. Les délais *retour* des réponses sont représentés de manière similaire.

Le délai noté T_{EM_i} est le temps nécessaire pour l'émission de cette requête depuis le contrôleur. De même, nous notons T_{Rep_i} le délai subi par la réponse correspondante.

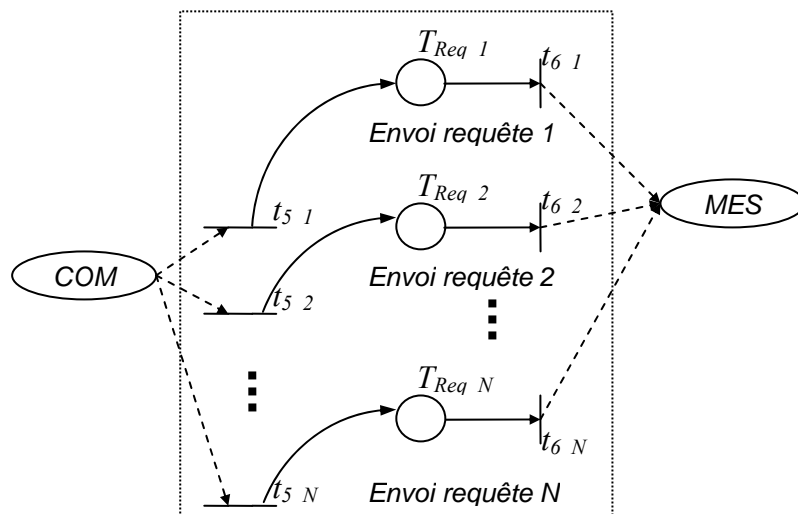


Fig. 2.10 : Délais de bout-en-bout des requêtes dans le cas multi serveur

Le modèle des modules E/S reste quant à lui inchangé puisque chaque module ne reçoit qu'une seule requête à la fois, du seul contrôleur du SCR.

Le modèle complet du SCR est donné sur la figure Fig. 2.11. Pour des raisons de clarté, un seul module E/S est représenté sur la figure (le modèle reste le même pour tous les MES).

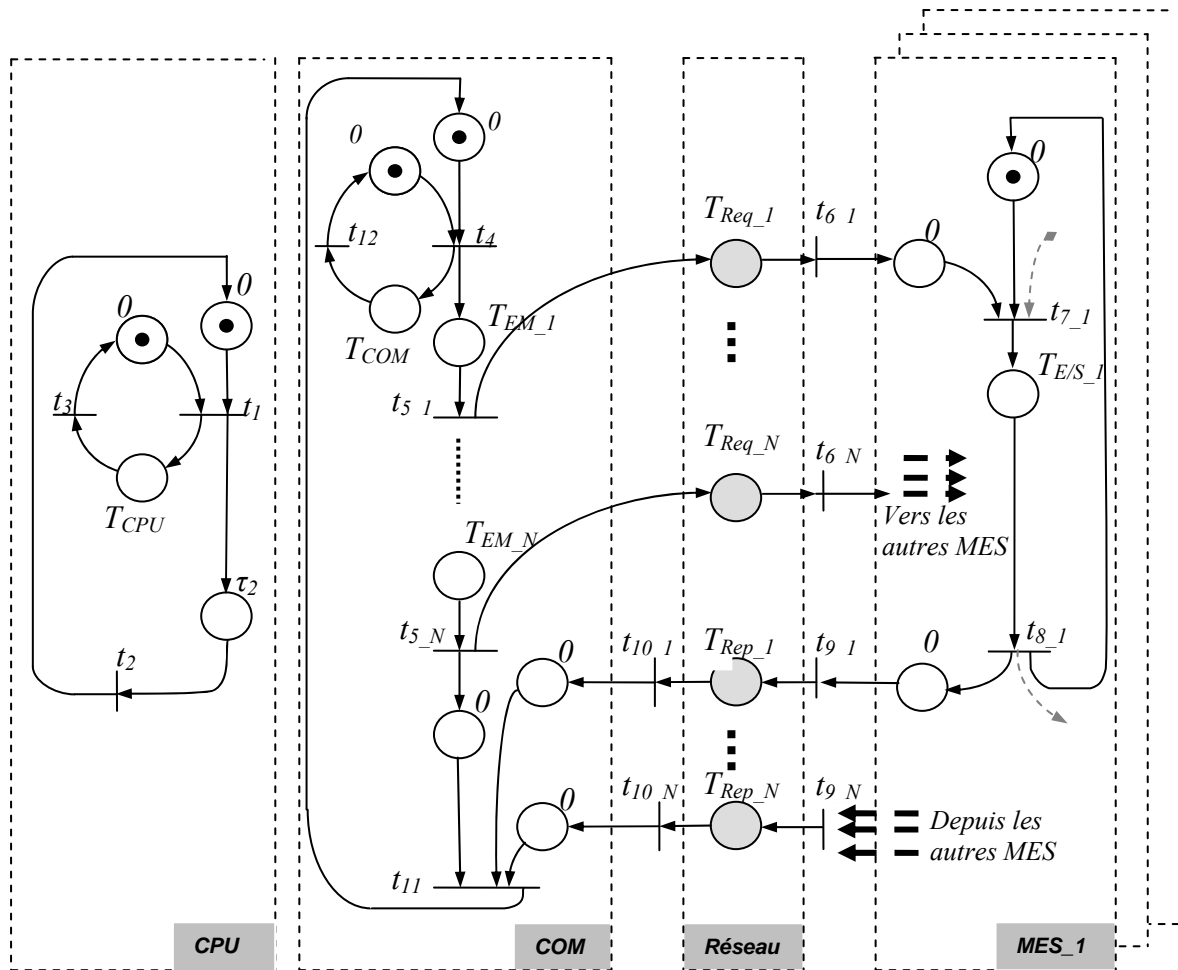


Fig. 2.11 : Modèle d'un SCR mono client - multi serveur

Les équations (max,+) correspondantes sont :

$$\begin{cases} \theta_1(k) = e \otimes \theta_2(k-1) \oplus e \otimes \theta_3(k-1) \\ \theta_2(k) = T_{CAL} \otimes \theta_1(k) \\ \theta_3(k) = T_{CPU} \otimes \theta_1(k) \end{cases}$$

$$\theta_4(l) = (\theta_{11}(l-1) \otimes e) \oplus (\theta_{12}(l-1) \otimes e)$$

Pour $i=1, \dots, N$

$$\theta_{5_i}(l) = \theta_{5_i(l-1)} \otimes T_{EM_i} \quad // \theta_{5_0}(l) = \theta_4(l) \text{ pour l'initialisation}$$

$$\theta_{6_i}(l) = \theta_{5_i}(l) \otimes T_{Req_i}$$

$$\theta_{7_i}(l) = (\theta_{6_i}(l) \otimes e) \oplus (\theta_{8_i}(l-1) \otimes e)$$

$$\theta_{8_i}(l) = \theta_{7_i}(l) \otimes T_{E/S_i}$$

$$\theta_{9_i}(l) = \theta_{8_i}(l) \otimes e$$

$$\theta_{10_i}(l) = \theta_{9_i}(l) \otimes T_{Rep_i} \quad \text{Fin}$$

$$\theta_{11}(l) = (\theta_{5_N}(l) \otimes e) \oplus \left[\bigoplus_{1 \leq j \leq N} (\theta_{10_j}(l) \otimes e) \right]$$

$$\theta_{12}(l) = \theta_4(l) \otimes T_{COM}$$

- **Remarque 2.2.5** : lors du calcul du temps de réponse, nous nous intéressons au retard entre le changement d'état sur une seule entrée donnée S (source de l'événement) sur le module MES_S, et une seule sortie D (destination de la réaction à l'événement) sur le module MES_D. Nous adoptons alors les notations suivantes :

- T_{Req_S} : est le délai de bout-en-bout subi par la requête envoyée vers la source S (MES_S).
- T_{Rep_S} : est le délai de bout-en-bout subi par la réponse renvoyée par la source S (MES_S).
- T_{Req_D} : est le délai de bout-en-bout subi par la requête envoyée vers la destination D (MES_D).

2.2.3.3 Cas 3 : Multi clients – multi serveurs

Dans le cas multi clients, plusieurs contrôleurs peuvent interroger le même module E/S. Si le modèle du contrôleur ne change pas par rapport au cas précédent, il n'en est pas de même pour le module E/S. Prenons le cas du premier module interrogé, soit MES_1, que nous supposons scruté par plusieurs clients (contrôleurs). Une requête entrant dans MES_1 n'est pas forcément traitée immédiatement après son arrivée puisqu'elle peut y trouver d'autres requêtes, en provenance d'autres clients, en attente de traitement. Elle doit donc attendre son tour durant un temps dépendant du nombre de paquets en attente (de jetons en attente dans la place Q_{E_1}). Un premier modèle représentant cela consiste à ajouter plusieurs transitions entrantes dans la place Q_{E_1} , représentant le buffer d'entrée de MES_1 (Fig. 2.12). Les jetons qui y entrent contribuent au franchissement de la transition t_{7_1} suivant le mode FIFO. De même pour le buffer de sortie (place Q_{S_1}). Il contient plusieurs transitions de sorties, chacune représentant le chemin d'une réponse renvoyée vers un client donné.

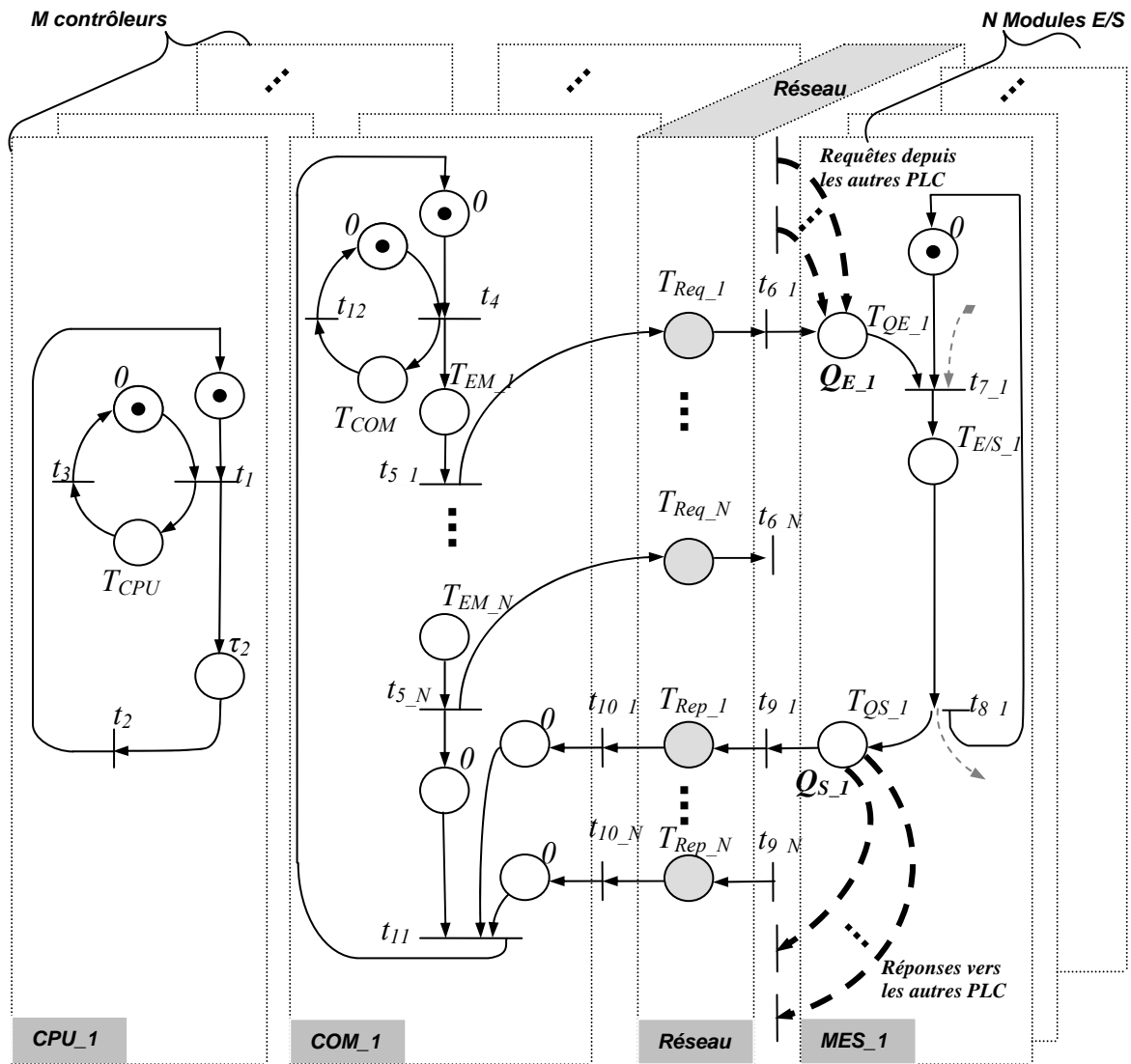


Fig. 2.12 : modèle d'un SCR avec M clients et N serveurs

Le problème majeur avec ce modèle est qu'il ne correspond pas à un GET. Les places Q_{E_1} et Q_{S_1} ne respectent pas l'unicité du nombre de transitions entrantes et sortantes. Donc, la représentation (max+) linéaire dont nous avons besoin n'est plus possible dans ce cas.

Nous résolvons ce problème, en deux étapes, comme suit :

i) Nous pouvons modéliser le cas multi clients avec deux approches possibles. Soit nous représentons tous les clients interrogeant le MES_1 comme sur la Fig. 2.12. Dans ce cas nous gardons toutes les transitions entrantes/sortantes des places Q_{E_1} et Q_{S_1} . Nous associons alors des délais nuls à ces places, comme dans le cas mono client, et l'attente dans ce cas dépendra de l'ordre d'entrée des jetons dans ces places. Le problème n'est cependant pas réglé puisque nous n'obtenons pas des GET.

Soit, au lieu d'intégrer tous les clients dans un seul modèle, nous nous intéressons qu'au seul contrôleur impliqué dans la boucle de commande dont le temps de réponse est recherché. De là, nous pouvons enlever les transitions concernant les autres clients des places Q_{E_1} et Q_{S_1} . Cependant, au lieu de leur associer des temps nuls (comme dans les modèles mono client), nous supposons que la requête du contrôleur en question y subit un délai d'attente non nul T_{QE_1} . C'est le délai d'attente dû aux autres requêtes envoyées par d'autres clients. Il en va de même pour la place Q_{S_1} où un délai T_{QS_1} non nul est subi.

ii) Une fois que la première étape est accomplie en procédant suivant la deuxième approche, nous aboutissons à un modèle GET où deux délais supplémentaires T_{QE_1} et T_{QS_1} doivent être calculés. Remarquons cependant que la place Q_{E_1} est en série avec la place modélisant le réseau et dont le délai est T_{Req_1} . Nous pouvons alors fusionner leurs délais dans la place représentant le réseau et considérer que le délai dans Q_{E_1} est nul, sans pour autant changer le fonctionnement du modèle. Ceci est logique et sans conséquence sur l'évaluation puisqu'un délai subi dans le réseau de communication ou bien à l'entrée du module d'E/S correspond à exactement la même situation du point de vu du délai de bout-en-bout (débutant à l'émission depuis le contrôleur de la requête et se terminant au moment du début de son traitement dans MES_1). Nous faisons de même pour la place Q_{S_1} et la place réseau dans le sens du retour.

Ainsi, nous aboutissons au même modèle que dans le cas mono client - multi serveurs. Ceci étant, nous considérons dans l'évaluation du temps de réponse que les deux premier cas, le troisième pouvant facilement se ramener au deuxième cas en appliquant les deux transformations expliquées précédemment. Pour l'évaluation du temps de réponse, le délai de bout-en-bout doit cependant comprendre le délai de bufferisation dans les MES.

Les modèles des SCR étant tous obtenus, nous pouvons passer à l'étape d'évaluation du temps de réponse. C'est l'objet du prochain chapitre.

Evaluation du temps de réponse des SCR

Sommaire

3.1 Evaluation du temps de réponse des SCR : analyse déterministe	55
3.1.1. SCR mono client mono serveur	55
3.1.1.1. Simplification et résolution des équations (max,+) linéaires	55
3.1.1.2. Calcul analytique des bornes du temps de réponse	59
3.1.1.2.1. Etude du cas $T_{COM}/T_{CPU} \in \mathbb{N}^+$	60
3.1.1.2.2. Etude du cas $T_{COM}/T_{CPU} \in \mathbb{Q}^+$	63
3.1.1.3. Calcul itératif de l'allure de la distribution du temps de réponse	65
3.1.2. SCR mono client multi serveurs	67
3.2 Evaluation du temps de réponse des SCR : analyse stochastique	70
3.2.1. Formulation du problème	70
3.2.2. Calcul analytique de la fonction de distribution du temps de réponse	72
3.3 Analyse de sensibilité du temps de réponse	74
3.3.1. Sensibilité aux délais de non synchronisation	74
3.3.2. Sensibilité aux délais de bout-en-bout	77
Conséquences de la sensibilité sur la qualité de commande	78

Dans ce chapitre, nous abordons l'évaluation du temps de réponse des SCR en utilisant deux approches : l'une est déterministe et l'autre est stochastique. Dans la première, nous développons des formules analytiques de calcul direct des bornes du temps de réponse ainsi qu'un algorithme de calcul itératif de l'allure de sa distribution. Dans la deuxième approche, nous donnons une formule de calcul exact de la fonction de distribution du temps de réponse.

Nous analysons ensuite la sensibilité du temps de réponse, aux différents délais qui le composent, grâce aux formules analytiques développées précédemment. Nous prouvons alors que la borne maximale du temps de réponse est particulièrement sensible aux délais de non synchronisation entre les composants du SCR ainsi qu'aux délais de bout-en-bout.

3.1 Evaluation du temps de réponse des SCR : analyse déterministe

Rappelons que le temps de réponse est la différence entre deux dates ; la date d'émission d'un signal par un capteur et la date d'arrivée du signal causal sur l'actionneur. Ces deux signaux sont caractérisés dans notre modèle par des événements sur les modules d'E/S déportés. Comme les modèles en GET que nous avons proposés représentent tous les événements possibles dans les SCR et les dates de leurs occurrences, il suffit de résoudre les équations (max,+) correspondantes et trouver les formules des dates utiles pour l'évaluation du temps de réponse.

3.1.1 SCR mono client - mono serveur

Nous commençons par l'étude du cas des SCR mono client - mono serveur car l'analyse en est relativement aisée et la généralisation sera plus facile.

3.1.1.1 Simplification et résolution des équations (max,+) linéaires

Nous allons utiliser deux hypothèses (H5 et H6 p.38) posées dans le Chapitre 2 pour simplifier et résoudre les systèmes d'équations (max,+) linéaires du modèle en GET des SCR mono client - mono serveur. Pour rappel, ces hypothèses stipulent que :

- H5 : pour le bon fonctionnement de la commande, la période du CPU est fixée de sorte qu'elle soit toujours suffisante, tout le temps, pour accomplir les phases de lecture, calcul et écriture. Ceci implique que : $T_{CPU} > T_{CAL}$.
- H6 : la période de scrutation est fixée de sorte que toutes les réponses aux requêtes émises lors d'un cycle soient reçues avant la fin de ce même cycle. Ceci implique que : $T_{COM} > T_{A/R}$, $T_{A/R}$ étant le temps d'aller-retour (le temps entre l'envoi de la requête et la réception de la réponse correspondante).

Nous commençons par la résolution des équations correspondant au GET représentant le CPU puis nous montrons qu'il est possible de ramener le deuxième GET (représentant le reste du SCR) à un GET semblable à celui du CPU. La résolution des équations est alors similaire.

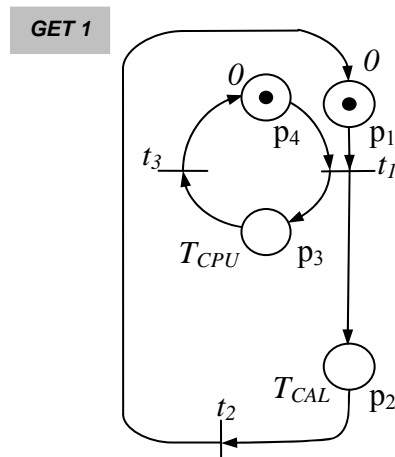


Fig. 3.1 : Modèle GET du CPU

Les systèmes d'équations (max,+) linéaires du GET 1 est le suivant :

$$\Theta_1(k) = A_1 \otimes \Theta_1(k-1) \quad (3.1)$$

La représentation d'état s'écrit également en utilisant l'état initial comme suit :

$$\Theta_1(k) = A_1^k \otimes \Theta_1(0) \quad (3.2)$$

où $\begin{pmatrix} \varepsilon & e & e \\ \varepsilon & T_{CAL} & T_{CAL} \\ \varepsilon & T_{CPU} & T_{CPU} \end{pmatrix}$ et $\Theta_1(0) = (\varepsilon \ e \ e)^t$

D'après le Théorème 2.3, pour un GET fortement connexe (ce qui est le cas du GET 1), il est possible d'écrire : $A_1^{k+c} = \lambda^c \otimes A_1^k$ pour un k assez grand, λ étant l'unique valeur propre finie de A_1 .

- **Proposition 3.1 :** sous l'hypothèse H5, la formule suivante est vraie pour $k \geq 1$;

$$A_1^k = T_{CPU}^{(k-1)} \otimes A_1 \quad (3.3)$$

Preuve (raisonnement par récurrence) :

- Il est évident que la formule est vraie pour $k=1$ puisque $A_1^1 = T_{CPU}^{(0)} \otimes A_1 = e \otimes A_1 = A_1$.

- Supposons maintenant qu'elle est vraie pour $k=n-1$ et prouvons que ceci implique qu'elle l'est aussi pour $k=n$.

Posons donc : $A_1^{(n-1)} = T_{CPU}^{(n-2)} \otimes A_1$. En multipliant cette égalité par A_1 , on trouve :

$$A_1^{(n-1)} \otimes A_1 = T_{CPU}^{(n-2)} \otimes A_1 \otimes A_1.$$

Ceci est équivalent à : $A_1^n = T_{CPU}^{(n-2)} \otimes A_1^2$.

Calculons le terme A_1^2 en fonction de A_1 :

$$\begin{aligned}
 A_1^2 &= \begin{pmatrix} \varepsilon & e & e \\ \varepsilon & T_{CAL} & T_{CAL} \\ \varepsilon & T_{CPU} & T_{CPU} \end{pmatrix} \otimes \begin{pmatrix} \varepsilon & e & e \\ \varepsilon & T_{CAL} & T_{CAL} \\ \varepsilon & T_{CPU} & T_{CPU} \end{pmatrix} \\
 &= \begin{pmatrix} \varepsilon & T_{CPU} \oplus T_{CAL} & T_{CPU} \oplus T_{CAL} \\ \varepsilon & T_{CAL} \otimes (T_{CPU} \oplus T_{CAL}) & T_{CAL} \otimes (T_{CPU} \oplus T_{CAL}) \\ \varepsilon & T_{CPU} \otimes (T_{CPU} \oplus T_{CAL}) & T_{CPU} \otimes (T_{CPU} \oplus T_{CAL}) \end{pmatrix} \\
 &= (T_{CPU} \oplus T_{CAL}) \otimes \begin{pmatrix} \varepsilon & e & e \\ \varepsilon & T_{CAL} & T_{CAL} \\ \varepsilon & T_{CPU} & T_{CPU} \end{pmatrix}.
 \end{aligned}$$

Comme $T_{CPU} > T_{CAL}$ d'après l'hypothèse H5, nous avons : $T_{CPU} \oplus T_{CAL} = T_{CPU}$. Nous avons alors : $A_1^2 = T_{CPU} \otimes A_1$ (ce qui prouve par la même occasion que la proposition 3.1 est vraie pour $k = 2$).

En remplaçant A_1^2 par sa valeur dans l'expression de A_1^n , nous aboutissons finalement à :

$$A_1^n = T_{CPU}^{(n-1)} \otimes A_1.$$

Ce qui prouve bien que la proposition 3.1 est vraie pour $k = n$. □

Faisons remarquer que la valeur propre de A_1 est $\lambda = (T_{CPU} \oplus T_{CAL}) = T_{CPU}$. Ceci montre aussi que la proposition est en accord avec le Théorème 2.3, la cyclicité de A_1 étant égale à 1.

De la même manière, on peut prouver que la matrice d'état des équations (max,+) du second GET sous l'hypothèse H6 vérifie la propriété : $A_2^l = T_{COM}^{(l-1)} \otimes A_2$.

Vu la taille de A_2 , nous allons procéder autrement que de calculer ses puissances.

Posons le temps d'aller-retour $T_{A/R} = T_{EM} + \bar{T}_{A/R}$ (temps entre l'envoi d'une requête et la réception de la réponse correspondante). Nous pouvons obtenir un GET équivalent au GET 2 comme montré sur la Fig. 3.2a, lequel est aussi équivalent au GET de la Fig. 3.2b. Finalement, ce dernier peut se mettre sous une forme similaire à celle du GET 1 du CPU (Fig. 2.3c) où T_{CPU} est remplacé par T_{COM} et T_{CAL} est remplacé par $T_{A/R}$.

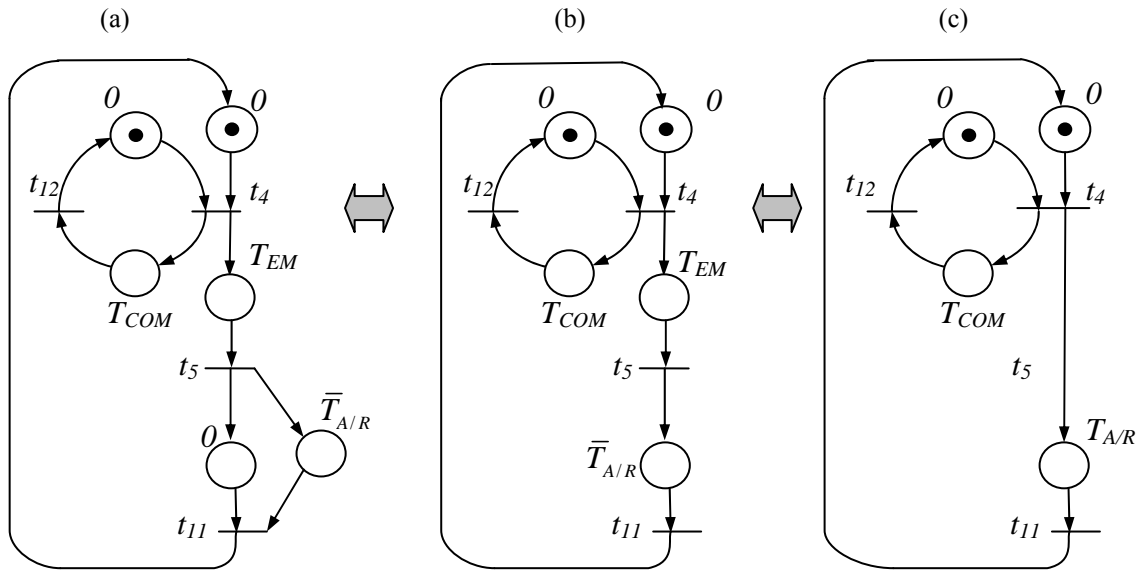


Fig. 3.2 : Transformation du GET 2 et un GET semblable au GET 1

Nous pouvons alors écrire la propriété suivante :

$$\bar{A}_2^l = T_{COM}^{(l-1)} \otimes \bar{A}_2 \text{ où } \bar{A}_2 \text{ est la matrice réduite de } A_2, \text{ correspondant au GET de la Fig. 3.2c.}$$

Les solutions des équations d'évolution des GET du SCR peuvent donc s'écrire :

$$\Theta_1(k) = T_{CPU}^{(k-1)} \otimes A_1 \otimes \Theta_1(0) \quad (3.4)$$

$$\Theta_2(l) = T_{COM}^{(l-1)} \otimes A_2 \otimes \Theta_2(0) \quad (3.5)$$

Parmi ces solutions, les plus utiles pour l'évaluation du temps de réponse sont celles représentant les débuts et fins des cycles du CPU et de la COM, soit θ_1 , θ_2 , θ_4 et θ_{11} . Elles sont données par :

$$\begin{cases} \theta_1(k) = T_{CPU}^{(k-1)} \\ \theta_2(k) = T_{CAL} \otimes T_{CPU}^{(k-1)} \\ \theta_4(l) = T_{COM}^{(l-1)} \\ \theta_{11}(l) = T_{A/R} \otimes T_{COM}^{(l-1)} \end{cases} \quad (3.6)$$

A ce stade, nous utiliserons les opérateurs de l'algèbre classique pour l'analyse et l'évaluation du temps de réponse. Les solutions précédentes peuvent ainsi s'écrire :

$$\begin{cases} \theta_1(k) = (k-1) \cdot T_{CPU} \\ \theta_2(k) = (k-1) \cdot T_{CPU} + T_{CAL} \\ \theta_4(l) = (l-1) \cdot T_{COM} \\ \theta_{11}(l) = (l-1) \cdot T_{COM} + T_{A/R} \end{cases} \quad (3.7)$$

3.1.1.2 Calcul analytique des bornes du temps de réponse

Le temps de réponse noté D_r peut être décomposé en un ensemble de délais élémentaires, subis aux différents niveaux du SCR comme le montre la Fig. 3.3.

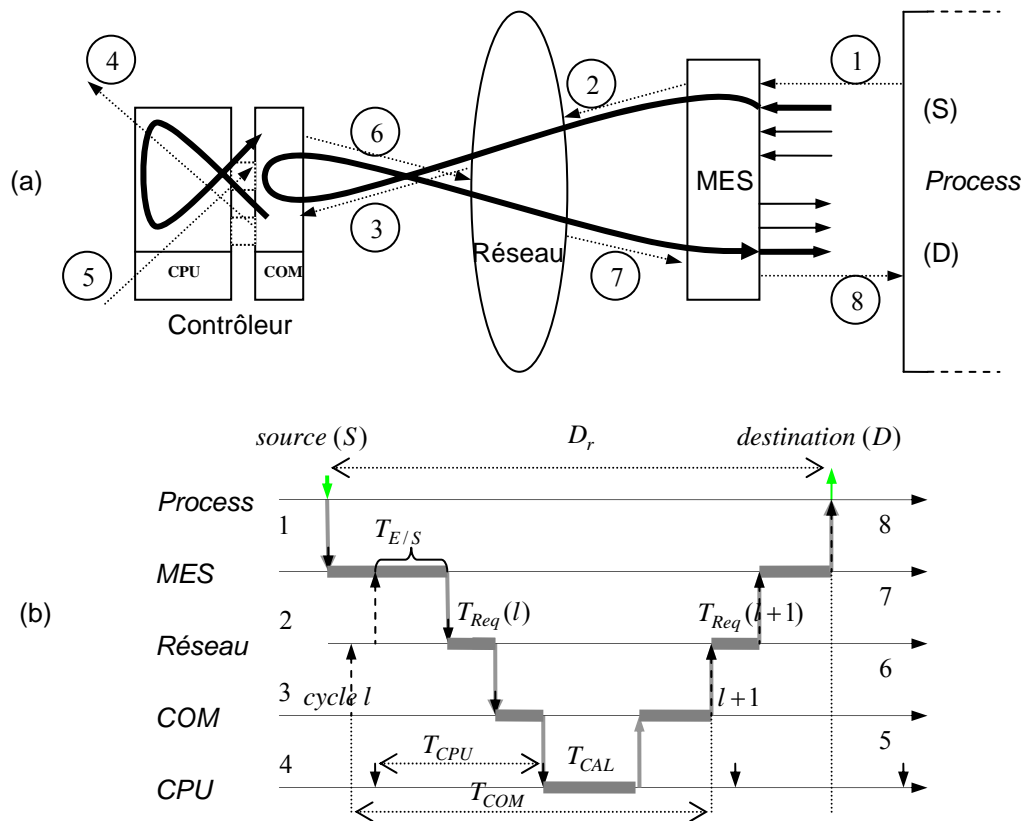


Fig. 3.3 : Temps de réponse et sa décomposition en délais élémentaires

Comme on peut le voir sur la Fig. 3.3, le temps de réponse peut être décomposé en huit étapes, suivant le niveau dans lequel se trouve l'événement généré par le capteur ;

- 1- Génération d'un événement par le capteur puis attente d'arrivée d'une requête de lecture de cet événement.
- 2- Après l'arrivée d'une requête et son traitement par le MES, une réponse est renvoyée.
- 3- Arrivée de la réponse au buffer d'entrée de la COM, sa mise en mémoire tampon puis attente du début d'un nouveau cycle CPU.
- 4- Démarrage d'un cycle CPU et exécution du programme utilisateur pour fabriquer l'événement de réaction (conséquence).
- 5- Mise en mémoire tampon de la conséquence et attente du début d'un nouveau cycle de la COM.
- 6- Envoi de la conséquence vers le MES.
- 7- Arrivée de la conséquence sur le MES puis son traitement par le MES.

8- Arrivée de la conséquence sur l'actionneur.

Comme nous pouvons le constater, le temps de réponse dépend de beaucoup de délais élémentaires qui peuvent varier d'un cycle à un autre. Le temps d'attente de début d'un nouveau cycle CPU par exemple peut, dans le meilleur des cas, être égal à zéro et au pire des cas égal à la période du CPU, soit T_{CPU} . Ce même raisonnement peut être appliqué aux différents niveaux du SCR.

Le temps de réponse est finalement propre à chaque cycle de scrutation de la COM.

Nous allons donc calculer le temps de réponse relatif à chaque cycle de scrutation en traquant l'événement généré avant l'arrivée de la requête envoyée au début de ce même cycle. Pour des raisons de complexité avérée par la suite, nous allons étudier deux cas, suivant que le rapport entre la période de la COM et du CPU soit un nombre entier ou un nombre rationnel. Souvenons nous que ce rapport T_{COM}/T_{CPU} est un nombre rationnel positif d'après l'hypothèse H1.

Posons les notations suivantes : $r = \frac{T_{COM}}{T_{CPU}}$, $\alpha = \frac{T_{A/R}}{T_{CPU}}$, $\beta = \frac{T_{CAL}}{T_{CPU}}$, $\lfloor x \rfloor$ partie entière du nombre rationnel x et $\lceil x \rceil$ sa partie fractionnaire.

3.1.1.2.1 Etude du cas $r \in \mathbb{N}^+$

Calculons le temps de réponse relatif au $l^{\text{ème}}$ cycle de scrutation. L'événement capteur est pris en compte durant ce cycle s'il est généré avant l'arrivée de la requête envoyée durant ce cycle, soit à une date $\theta_{gen}(l)$ telle que : $\theta_{gen}(l) < \theta_6(l)$. Par ailleurs, cet événement doit être généré après l'arrivée de la requête envoyée au $(l-1)^{\text{ème}}$ cycle. Dans le cas contraire, l'événement est pris en compte au $(l-1)^{\text{ème}}$ cycle et est donc propre à ce cycle et non au $l^{\text{ème}}$. Pour exprimer cela, nous pouvons écrire : $\theta_{gen}(l) = \theta_6(l) - \tau_l$, τ_l étant le temps d'attente de l'événement dans le MES avant l'arrivée de la $l^{\text{ème}}$ requête. Nous allons maintenant traquer cet événement en suivant les étapes 1 à 8 de la Fig. 3.3.

L'événement étant pris en compte avec l'arrivée de la $l^{\text{ème}}$ requête, la réponse correspondante arrive à la COM à la date $\theta_{11}(l)$. Cette réponse se retrouve dans la mémoire tampon à cette même date (le temps d'écriture étant nul d'après l'hypothèse H2). Cette réponse est utilisée dans l'exécution du programme de commande au prochain début de cycle CPU, soit à la date $\theta_1(k_l)$ telle que :

$$\theta_1(k_l) > \theta_{11}(l) \geq \theta_1(k_l - 1) \quad (3.8)$$

D'après les solutions (3.7), nous avons :

$$\begin{aligned}\theta_{11}(l) &= (l-1) \cdot T_{COM} + T_{A/R} \\ &= r \cdot (l-1) \cdot T_{CPU} + \alpha \cdot T_{CPU} \\ &= [r \cdot (l-1) + \alpha] \cdot T_{CPU}\end{aligned}\quad (3.9)$$

et

$$\theta_1(k_l) = (k_l - 1) \cdot T_{CPU} \quad (3.10)$$

Sachant que $\lfloor \alpha \rfloor + 1 > \alpha$ et $\lfloor \alpha \rfloor \leq \alpha$, il est clair que pour : $k_l - 1 = r \cdot (l-1) + \lfloor \alpha \rfloor + 1$, la condition (3.8) est vérifiée. En remplaçant k_l par cette valeur, nous obtenons :

$$\theta_1(k_l) = [r \cdot (l-1) + \lfloor \alpha \rfloor + 1] \cdot T_{CPU} \quad (3.11)$$

La réponse étant prise en compte au début du $k_l^{\text{ème}}$ cycle CPU, la conséquence est mise en mémoire tampon à la fin de l'exécution du programme de commande, soit à la date $\theta_2(k_l)$:

$$\theta_2(k_l) = \theta_1(k_l) + T_{CAL} = [r \cdot (l-1) + \lfloor \alpha \rfloor + 1 + \beta] \cdot T_{CPU} \quad (3.12)$$

Encore une fois, cette conséquence n'est envoyée vers le MES qu'au début du prochain cycle de scrutation. Sachant que l'événement capteur est récupéré avec l'envoi de la requête du $l^{\text{ème}}$ cycle, la conséquence est forcément envoyée lors d'un cycle ultérieur, soit le $(l + q_l)^{\text{ème}}$ cycle tel que le nombre $q_l \in \mathbb{N}^+$ soit supérieur ou égal à 1. Plus précisément, ce nombre doit vérifier :

$$\theta_4(l + q_l) > \theta_2(k_l) \geq \theta_4(l + q_l - 1) \quad (3.13)$$

Autrement dit, nous devons rechercher le nombre entier minimal q_l tel que :

$$\theta_4(l + q_l) > \theta_2(k_l) \quad (3.14)$$

Cette condition peut être exprimée en remplaçant les dateurs θ_4 et θ_2 par leurs solutions :

$$r \cdot (l + q_l - 1) \cdot T_{CPU} > [r \cdot (l-1) + \lfloor \alpha \rfloor + 1 + \beta] \cdot T_{CPU} \quad (3.15)$$

On peut vérifier aisément que cette condition est équivalente à :

$$r \cdot q_l > 1 + \lfloor \alpha \rfloor + \beta \quad (3.16)$$

Maintenant que nous savons que la requête contenant la conséquence est envoyée au $(l + q_l)^{\text{ème}}$ cycle de scrutation, celle-ci arrive forcément à destination à la date $\theta_8(l + q_l)$.

Le temps de réponse relatif au $l^{\text{ème}}$ cycle de scrutation peut être finalement exprimé comme :

$$D_r = \theta_8(l + q_l) - \theta_{gen}(l) \quad (3.17)$$

Place maintenant aux bornes du temps de réponse ...

Rappelons que l'événement pris en compte au $l^{\text{ème}}$ cycle de scrutation est généré à la date $\theta_{gen}(l)$ telle que :

$$\theta_6(l-1) < \theta_{gen}(l) < \theta_6(l) \quad (3.18)$$

Le meilleur cas (temps de réponse minimal) correspond à un événement pris en compte immédiatement après sa génération. Autrement dit, une requête arrive un instant après sa génération, soit à la date : $\theta_6(l) = \theta_{gen}(l) + 0^+$. La borne minimale du temps de réponse est finalement :

$$D_r^{\min}(l) = \theta_8(l + q_l) - \theta_6(l) \quad (3.19)$$

En remplaçant les dateurs $\theta_8(l + q_l)$ et $\theta_6(l)$ par leurs expressions (solutions (3.5) page 58), nous obtenons :

$$\begin{aligned} D_r^{\min}(l) &= \theta_8(l + q_l) - \theta_6(l) \\ &= (l + q_l - 1) \cdot T_{COM} + T_{EM} + T_{Req}(l + q_l) + T_{E/S} - [(l - 1) \cdot T_{COM} + T_{EM} + T_{Req}(l)] \end{aligned} \quad (3.20)$$

Finalement nous obtenons :

$$D_r^{\min}(l) = q_l \cdot T_{COM} + \Delta(l, q_l) + T_{E/S} \quad (3.21)$$

où le terme $\Delta(l, q_l) = T_{Req}(l + q_l) - T_{Req}(l)$ est dû exclusivement au réseau de communication (différence entre deux délais de bout-en-bout).

A l'inverse, le pire cas correspond à un événement généré un instant après l'arrivée d'une requête. Pour le $l^{\text{ème}}$ cycle, la borne maximale est atteinte quand : $\theta_{gen}(l) = \theta_6(l-1) + 0^+$. En remplaçant les dates dans (3.17) par leurs solutions, nous arrivons à l'expression :

$$D_r^{\max}(l) = (q_l + 1) \cdot T_{COM} + \Delta(l - 1, q_l + 1) + T_{E/S} \quad (3.22)$$

Les bornes que nous avons calculées sont relatives au $l^{\text{ème}}$ cycle. Le plus intéressant est évidemment de trouver les bornes dans l'absolu. Sachant que la période de communication est de loin supérieure aux délais réseau (conséquence de l'hypothèse H6), la borne maximale absolue est atteinte quand le nombre entier q_l est maximal et la borne minimale est atteinte quand q_l est minimal. Nous avons alors les expressions des bornes absolues suivantes :

$$D_r^{MIN} = q_{\min} \cdot T_{COM} + \Delta_{\min} + T_{E/S} \quad (3.23)$$

$$D_r^{MAX} = (q_{\max} + 1) \cdot T_{COM} + \Delta_{\max} + T_{E/S} \quad (3.24)$$

où $q_{\min} = \min_{l \in \mathbb{N}} \{q_l\}$ et $q_{\max} = \max_{l \in \mathbb{N}} \{q_{\max}\}$

Le calcul de ces valeurs, minimale et maximale, est expliqué par la suite dans la discussion des résultats.

3.1.1.2.2 Etude du cas $r \in \mathbb{Q}^+$ avec $\lceil r \rceil \neq 0$

Le raisonnement est similaire que précédemment. Recherchons le nombre k_l tel que :

$$\theta_1(k_l) > \theta_{11}(l) \geq \theta_1(k_l - 1) \quad (3.25)$$

Nous avons :

$$\begin{cases} \theta_1(k_l) = (k_l - 1) \cdot T_{CPU} \\ \theta_{11}(l) = \lceil r \rceil \cdot (l - 1) + \alpha \cdot T_{CPU} \end{cases} \quad (3.26)$$

Posons le nombre $i \in \mathbb{N}$ tel que :

$$i \leq \lceil \alpha \rceil + \lceil r \rceil \cdot (l - 1) < (i + 1) \quad (3.27)$$

Vérifions alors que pour : $k_l - 1 = \lfloor r \rfloor \cdot (l - 1) + \lfloor \alpha \rfloor + 1 + i$, la condition (3.25) est remplie. En remplaçant cette valeur de k_l dans l'expression de $\theta_1(k_l)$, nous obtenons :

$$\theta_1(k_l) = (k_l - 1) \cdot T_{CPU} = (\lfloor r \rfloor \cdot (l - 1) + \lfloor \alpha \rfloor + 1 + i) \cdot T_{CPU}$$

Comme $(i + 1) > \lceil \alpha \rceil + \lceil r \rceil \cdot (l - 1)$ d'après (3.27), nous avons alors :

$$1 + i + \lfloor r \rfloor \cdot (l - 1) + \lfloor \alpha \rfloor > \lceil \alpha \rceil + \lceil r \rceil \cdot (l - 1) + \lfloor r \rfloor \cdot (l - 1) + \lfloor \alpha \rfloor$$

$$\text{Ceci est équivalent à : } 1 + i + \lfloor r \rfloor \cdot (l - 1) + \lfloor \alpha \rfloor > \alpha + r \cdot (l - 1)$$

En multipliant l'inéquation par T_{CPU} , nous trouvons : $\theta_1(k_l) > \theta_{11}(l)$.

De la même manière, on prouve facilement que : $\theta_{11}(l) \geq \theta_1(k_l - 1)$.

La condition (3.25) est donc bien remplie et le nombre k_l recherché est donné par :

$$k_l = \lfloor r \rfloor \cdot (l - 1) + \lfloor \alpha \rfloor + 2 + i \quad (3.28)$$

Recherchons maintenant le nombre minimal $q_l \in \mathbb{N}^+$ tel que $\theta_4(l + q_l) > \theta_2(k_l)$, soit :

$$r \cdot (l + q_l - 1) \cdot T_{CPU} > (\lfloor r \rfloor \cdot (l - 1) + \lfloor \alpha \rfloor + 1 + i + \beta) \cdot T_{CPU},$$

ce qui est équivalent à :

$$r \cdot q_l > 1 + i - \lceil r \rceil \cdot (l - 1) + \lfloor \alpha \rfloor + \beta. \quad (3.29)$$

De l'inéquation (3.27), on peut facilement déduire que :

$$\lceil \alpha \rceil < i + 1 - \lceil r \rceil \cdot (l - 1) \leq 1 + \lceil \alpha \rceil \quad (3.30)$$

Posons : $\Gamma(i, l) = i + 1 - \lceil r \rceil \cdot (l - 1)$ tel que (3.27) soit vérifiée, soit $\lceil \alpha \rceil < \Gamma(i, l) \leq 1 + \lceil \alpha \rceil$.

La condition (3.29) de calcul de q_l devient alors :

$$r \cdot q_l > \Gamma(i, l) + \lfloor \alpha \rfloor + \beta \quad (3.31)$$

La borne maximale du temps de réponse est calculée en utilisant la condition :

$$r \cdot q_{\max} > \Gamma_{\max} + \lfloor \alpha \rfloor + \beta \quad \text{où } \Gamma_{\max} = \max_{i, l \in \mathbb{N}} \Gamma(i, l).$$

De manière similaire, on calcule la borne minimale avec $r \cdot q_{\min} > \Gamma_{\min} + \lfloor \alpha \rfloor + \beta$ où $\Gamma_{\min} = \min_{i, l \in \mathbb{N}} \Gamma(i, l)$.

Pour résumer, le calcul du temps de réponse relatif à un cycle donné l passe par le calcul de $\Gamma(i, l)$ puis du nombre minimal q_l vérifiant la condition ci-dessus (3.31). Une fois ce nombre trouvé, les formules de calcul reste les mêmes que dans le cas $r \in \mathbb{N}^+$.

- Remarque 3.1 : sachant que $\forall l, i \in \mathbb{N} : \Gamma(i, l) \leq 1 + \lceil \alpha \rceil$, il est possible de simplifier la condition de calcul de la borne maximale en supposant que le cas extrême où $\Gamma_{\max} = 1 + \lceil \alpha \rceil$ est atteint. Dans ce cas la condition (3.31) devient :

$$r \cdot q_{\max} > 1 + \alpha + \beta \quad (3.32)$$

Faisons remarquer cependant que cette simplification peut introduire du pessimisme dans certains cas où l'extremum n'est pas atteint. Son utilisation doit donc se faire avec précaution.

Discussion des résultats :

i) Un raisonnement intuitif conduit à penser qu'une plus petite période de scrutation T_{COM} mènerait vers un temps de réponse plus petit. En réalité, ce n'est pas le cas. Prenons un exemple où $T_{CPU} = 5ms$, $\beta = 0.6$ et $\alpha = 0.2$. Si on prend une période $T_{COM} = 10ms$, soit $r = 2$ et $r \in \mathbb{N}^+$, le nombre $q_l = 1$ vérifie la condition (3.16). Le temps de réponse maximal correspondant est d'environ $2 \times 10ms$ (plus le terme dû au réseau Δ , ce dernier étant très petit devant cette valeur). Prenons maintenant $T_{COM} = 9ms$. Cette fois $r \in \mathbb{Q}^+$ et le nombre minimal q_l vérifiant la condition (3.31) est $q_l = 2$. Dans ce cas le temps de réponse est d'environ $3 \times 9ms$, soit environ $27ms$.

Ce phénomène paradoxal peut s'expliquer comme suit : une fois qu'une réponse est reçue dans la COM, mieux vaut temporiser un peu avant de débiter un nouveau cycle de scrutation pour attendre que la réponse soit prise en compte par le CPU et que la conséquence soit mise à disposition de la COM. Ainsi, la conséquence est envoyée immédiatement au début du

cycle succédant au cycle précédent. A l'inverse, si le cycle débute trop tôt, la conséquence pourrait rater ce cycle COM et devrait attendre le début d'un autre cycle de scrutation, ce qui constitue délai d'attente plus long.

ii) Dans les SCR, les délais réseau sont très inférieurs à la période de la COM (hypothèse H6) et même à celle du CPU. Dans ce cas, $\alpha < 1$ et la condition (3.16) est vérifiée pour $q_l = 1$ et $r \geq 2$. Il est alors conseillé de configurer la période de scrutation comme le double de la période du CPU. En effet, le temps de réponse maximal dans ce cas est optimal (minimal) puisque $q_{\max} = 1$.

iii) La borne maximale est atteinte quand q_l est maximal. Comme nous pouvons le voir sur les conditions de calcul de q_{\max} , ceci implique que α et β soient maximaux également. Ceci constitue un résultat intéressant puisque le calcul des bornes du temps de réponse nécessite uniquement l'utilisation des bornes de $T_{A/R}$ et de T_{CAL} .

3.1.1.3 Calcul itératif de l'allure de la distribution du temps de réponse

Contrairement au calcul des bornes du temps de réponse qui nécessitent seulement les bornes des délais élémentaires, l'estimation de la distribution requiert, quant à elle, la connaissance de plusieurs valeurs de ces délais. Sachant que nous considérons une analyse déterministe, nous supposons que les valeurs de ces délais sont connues à chaque étape (cycle). Par exemple, nous supposons que les délais de bout-en-bout $T_{Req}(l)$ et $T_{Rep}(l)$ sont connus. Ces délais peuvent par exemple être évalués à l'avance par simulation, par mesure ou même par échantillonnage d'une distribution connue. Il en va de même pour tous les autres délais élémentaires.

Il est à noter qu'il est nécessaire de générer une série d'événements à des instants $\theta_{gen}(l)$ correspondant à une distribution donnée. A partir de là, nous pouvons évaluer le temps de réponse relatif à chaque cycle en utilisant la formule (3.17). En répétant la démarche sur un long horizon, nous trouvons un ensemble de temps de réponse que nous pouvons représenter par un histogramme. Ceci permet d'obtenir l'allure de la distribution du temps de réponse. Pour cela, nous pouvons adopter l'approche suivante :

Rappelons nous que le temps de réponse relatif au $l^{ème}$ cycle de scrutation est donné dans (3.17) par :

$$D_r = \theta_8(l + q_l) - \theta_{gen}(l) = \theta_8(l + q_l) - \theta_6(l) - \tau_l \quad (3.33)$$

En remplaçant les dateurs de cette équation par leurs expressions (solutions), nous obtenons :

$$\begin{aligned}
 D_r &= (l + q_l - 1) \cdot T_{COM} + T_{EM} + T_{Req}(l + q_l) + T_{E/S} - [(l - 1) \cdot T_{COM} + T_{EM} + T_{Req}(l)] - \tau_l \\
 &= q_l \cdot T_{COM} + T_{Req}(l + q_l) - T_{Req}(l) + T_{E/S} - \tau_l
 \end{aligned} \tag{3.34}$$

Pour calculer ce temps de réponse, il faut simplement connaître q_l et τ_l , les autres paramètres étant déjà connus.

Concernant q_l , il suffit d'utiliser la condition (3.16) ou (3.28) suivant que le rapport r soit entier ou rationnel. Pour τ_l , on peut le déduire de l'expression : $\theta_{gen}(l) = \theta_6(l) - \tau_l$. Il est égal à :

$$\tau_l = \theta_{gen}(l) - [(l - 1) \cdot T_{COM} + T_{EM} + T_{Req}(l)] \tag{3.35}$$

Un problème qui fait que cette formule ne peut être utilisée directement se pose cependant. En effet, lors de la génération des événements, qui sont inhérents au seul système commandé, on génère un vecteur où la $l^{ème}$ composante n'est pas forcément l'événement pris en compte au $l^{ème}$ cycle de scrutation. On génère donc un autre vecteur que nous notons $\bar{\theta}_{gen}$. Puisque nous calculons le temps de réponse relatif à un cycle donné, la question qui se pose est donc la suivante : quel cycle de scrutation correspond au $m^{ème}$ événement généré ?

Rappelons que le $m^{ème}$ événement, généré à la date $\bar{\theta}_{gen}(m)$, est pris en compte au $l_m^{ème}$ cycle de scrutation si et seulement si :

$$\theta_6(l_m - 1) \leq \bar{\theta}_{gen}(m) < \theta_6(l_m),$$

ce qui est équivalent à rechercher l_m tel que :

$$(l_m - 2) \cdot T_{COM} + T_{EM} + T_{Req}(l_m) \leq \theta_{gen}(m) < (l_m - 1) \cdot T_{COM} + T_{EM} + T_{Req}(l_m) \tag{3.36}$$

En résumé, pour chaque composante $\bar{\theta}_{gen}(m)$, il faut rechercher le cycle de scrutation l_m correspondant en utilisant (3.36). Une fois celui-ci trouvé, nous pouvons simplement utiliser la formule de calcul du temps de réponse :

$$D_r(l_m) = q_{l_m} \cdot T_{COM} + T_{Req}(l_m + q_l) - T_{Req}(l_m) + T_{E/S} - \tau_{l_m} \tag{3.37}$$

Le processus précédent de calcul du temps de réponse est répété pour chaque événement généré, suivant l'algorithme de la Fig. 3.3. En procédant itérativement de la sorte, sur un horizon suffisamment long, nous obtenons un histogramme qui donne l'allure de la distribution du temps de réponse. La moyenne du temps de réponse peut ainsi par exemple être calculée et utilisée comme critère d'évaluation de la réactivité d'un SCR.

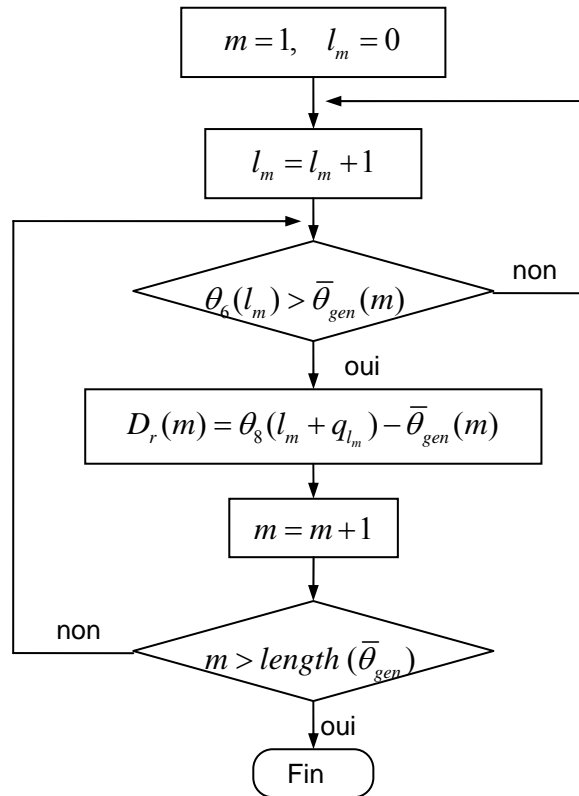


Fig. 3.3 : Détermination de l'allure de la distribution du temps de réponse par calcul itératif

- **Remarque 3.2** : cette méthode itérative peut être satisfaisante pour évaluer la distribution du temps de réponse ou sa moyenne mais elle reste une méthode non exhaustive, assimilable à de la simulation. Elle ne considère en effet que certains scénarii, correspondants aux seules dates du vecteur $\bar{\theta}_{gen}$. Elle ne donne donc pas de preuve formelle que la borne maximale de tous les temps de réponse trouvés soit effectivement la borne maximale réelle. Pour cela, il faut utiliser plutôt les formules des bornes (3.23) et (3.24).

3.1.2 SCR Mono client - multi serveurs

L'évaluation du temps de réponse est légèrement différente du cas mono serveur. En effet, la source de l'événement et la destination de la conséquence n'étant pas forcément les mêmes, nous devons apporter quelques ajustements dans l'analyse. Comme nous l'avions annoncé dans le Chapitre 2 lors de la modélisation du cas multiserveurs, N serveurs sont interrogés dans un ordre invariant. Le MES connecté au capteur (source des événements) est supposé être le $S^{ème}$ interrogé et le MES destinataire (actionneur) est le $D^{ème}$, $S, D \in \{1, \dots, N\}$. De ce fait, les délais subis par la requête envoyée vers la source sont indicés avec un « S » et ceux subis par la requête envoyée vers « D » sont indicés avec un « D ». Les solutions des

équations (max,+) des GET du cas multiserveurs sont aussi exprimées en utilisant ces indices. Ainsi par exemple, la date d'arrivée d'une requête au MES source, envoyée au $l^{\text{ème}}$ cycle de scrutation, est donnée par :

$$\theta_{6_S}(l) = (l-1) \cdot T_{COM} + \sum_{i=1}^S T_{EM_i} + T_{Req_S}(l) \quad (3.38)$$

Outre les notations, la différence majeure avec le cas mono serveur réside dans le fait que la transition t_{11} ne représente plus l'arrivée d'une réponse mais plutôt celle de toutes les réponses avant le début d'un nouveau cycle de scrutation. Or, chaque réponse de la source est copiée dans la mémoire tampon immédiatement après son arrivée. Par conséquent, dans la recherche du nombre entier k_l , ce n'est plus $\theta_{11}(l)$ qu'on doit utiliser mais une date notée $\theta_{11_S}(l)$, représentant la date d'arrivée de la réponse de la source. Elle est donnée par :

$$\theta_{11_S}(l) = (l-1) \cdot T_{COM} + \max \left(T_{A/R_S}(l), \sum_{i=1}^N T_{EM_i} \right) \quad (3.39)$$

L'opérateur *max* dans cette solution est introduit du fait que la réponse n'est plus copiée dans la mémoire tampon juste après sa réception, soit après $T_{A/R_S}(l)$, mais après que toutes les requêtes soient envoyées vers les MES, d'où le terme $\sum_{i=1}^N T_{EM_i}$.

Le paramètre α représentant le temps d'aller-retour est donné cette fois par :

$$\max \left(T_{A/R_S}(l), \sum_{i=1}^N T_{EM_i} \right) = \alpha \cdot T_{CPU} \quad (3.40)$$

Le reste de l'analyse est exactement le même que dans le cas mono serveur. Nous recherchons donc k_l tel que : $\theta_1(k_l) > \theta_{11_S}(l) \geq \theta_1(k_l - 1)$

Aussi bien dans le cas $r \in \mathbb{N}^+$ que $r \in \mathbb{Q}^+$, les expressions de k_l restent les mêmes. La condition de calcul du nombre minimal q_l reste aussi la même. Ainsi, si la condition $r \cdot q_l > \Gamma(i, l) + \lfloor \alpha \rfloor + \beta$ est vérifiée, la date d'arrivée de la conséquence à la destination D est $\theta_{8_D}(l + q_l)$. Le temps de réponse est finalement donné comme suit :

$$D_r = \theta_{8_D}(l + q_l) - \theta_{gen}(l) \quad (3.41)$$

où $\theta_{gen}(l) = \theta_{6_S}(l) - \tau_l$.

La borne minimale est obtenue quand $\theta_{gen}(l) = \theta_{6_S}(l) - 0^+$ ($\tau_l = 0^+$). Elle est donnée par :

$$D_r^{\min}(l) = \theta_{8_D}(l + q_l) - \theta_{6_S}(l) \quad (3.42)$$

La borne maximale est obtenue quand $\theta_{gen}(l) = \theta_{6_s}(l-1) + 0^+$ et a pour expression :

$$D_r^{\max}(l) = \theta_{8_D}(l + q_l) - \theta_{6_s}(l - 1) \quad (3.43)$$

En remplaçant les dates par leurs expressions, nous obtenons les bornes :

$$D_r^{\min}(l) = q_l \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + \Delta(l, q_l) + T_{E/S_D} \quad (3.44)$$

$$D_r^{\max}(l) = (q_l + 1) \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + \Delta(l - 1, q_l + 2) + T_{E/S_D} \quad (3.45)$$

où $\Delta(l, q_l) = T_{Req_D}(l + q_l) - T_{Req_S}(l)$

Remarquons que dans le cas où la source est également la destination ($S \equiv D$), nous nous retrouvons avec les même formules que dans le cas mono serveur. Seul le calcul de q_l est différent.

Discussion des résultats :

Dans ces formules générales (3.44)-(3.45), nous remarquons un terme supplémentaire négatif

$(-\sum_{i=1}^S T_{EM_i})$ relatif à la source. Ceci signifie que si nous augmentons l'ordre de scrutation du

MES source ou, autrement dit, retardons l'envoi de la requête vers la source, le temps de réponse diminue. Cet autre phénomène paradoxal de prime abord, peut être expliqué intuitivement. En effet, il est parfois préférable de retarder l'envoi d'une requête et attendre l'occurrence d'un événement plutôt que d'envoyer la requête *prématurément* et par conséquent de risquer de rater cet événement. Cet événement, pour être pris en compte, devrait alors attendre la prochaine requête du contrôleur, ce qui constitue un délai long. Faisons remarquer que ce fait a été relevé aussi expérimentalement dans (Denis *et al.*, 2007) où il a été observé que si le contrôleur est interrogé (ce qui reviendrait à retarder l'envoi des requêtes), le temps de réponse minimal diminue. Gardons cependant à l'esprit que retarder l'envoi de la requête vers la source augmenterait le temps d'aller retour T_{A/R_S} et donc α . Or, si cette valeur dépasse un certain seuil, la condition de calcul de q_l peut basculer et augmenter brusquement le temps de réponse.

Par ailleurs, le terme positif $\sum_{i=1}^D T_{EM_i}$ suggère que pour diminuer le temps de réponse, il faut diminuer ce terme en interrogeant le MES destination le plus tôt possible.

En conclusion, pour diminuer le temps de réponse, on peut retarder la scrutation de la source tout en restant en deçà d'un certain seuil et scruter la destination au plus tôt.

3.2 Evaluation du temps de réponse des SCR : analyse stochastique

3.2.1 Formulation du problème

Dans la section précédente, nous avons proposé une méthode déterministe de calcul du temps de réponse en se focalisant sur les bornes. Nous avons également proposé une méthode itérative de calcul de l'allure de la distribution du temps de réponse. L'idéal est bien entendu de trouver la fonction de distribution du temps de réponse analytiquement en considérant les distributions des délais élémentaires. C'est l'objet de cette section.

La formule de calcul du temps de réponse du $l^{\text{ème}}$ cycle est :

$$D_r(l) = \theta_{8_D}(l + q_l) - [\theta_{6_S}(l) - \tau_l] \quad (3.46)$$

Nous pouvons réécrire cette formule comme suit :

$$D_r(l) = \theta_{8_D}(l + q_l) - [\theta_{6_S}(l - 1) + \bar{\tau}_l] \quad (3.47)$$

où $\bar{\tau}$ est le délai entre l'arrivée de la requête du $(l - 1)^{\text{ème}}$ cycle et la date d'occurrence de l'événement. Le pire cas est donc atteint quand $\bar{\tau} = 0^+$.

En remplaçant les dateurs par les solutions dans (3.47), nous trouvons :

$$D_r(l) = (q_l + 1) \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + \bar{\Delta}(l - 1, q_l + 2) + T_{E/S_D} \quad (3.48)$$

où $\bar{\Delta}(l - 1, q_l + 2) = T_{Req_D}(l + q_l) - T_{Req_S}(l - 1) - \bar{\tau}_l$

Dans la formule (3.48), seuls les deux paramètres q_l et $\bar{\Delta}(l - 1, q_l + 2)$ sont variables. Le calcul de la fonction de distribution du temps de réponse dépend alors seulement de ces deux paramètres.

En fait, les valeurs de la fonction de distribution qui nous intéressent sont surtout celles qui sont au voisinage de la borne maximale du temps de réponse. L'objectif est de trouver la probabilité que le temps de réponse dépasse une certaine valeur limite D_r^{lim} , cette dernière étant au voisinage de D_r^{max} . Pour ce faire, nous posons :

$$D_r^{\text{lim}} = (q_{\text{max}} + 1) \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + \bar{\Delta}_{\text{lim}} + T_{E/S_D} \quad (3.49)$$

Nous formulons alors le lemme suivant :

Lemme 3.1 : *le temps de réponse $D_r(l)$ dépasse la limite D_r^{lim} si et seulement si :*

$q_l = q_{\max}$ et $\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\lim}$, soit :

$$D_r(l) \geq D_r^{\lim} \Leftrightarrow q_l = q_{\max} \text{ et } \bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\lim}$$

Preuve :

L'implication : $q_l = q_{\max}$ et $\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\lim} \Rightarrow D_r(l) \geq D_r^{\lim}$ est évidente. Nous allons prouver l'autre implication.

$D_r(l) \geq D_r^{\lim}$ implique que :

$$\begin{aligned} (q_l + 1) \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + \bar{\Delta}(l-1, q_l + 2) + T_{E/S_D} \\ \geq (q_{\max} + 1) \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + \bar{\Delta}_{\lim} + T_{E/S_D} \end{aligned} \quad (3.50)$$

Ceci est équivalent à :

$$(q_l + 1) \cdot T_{COM} \geq (q_{\max} + 1) \cdot T_{COM} + \bar{\Delta}_{\lim} - \bar{\Delta}(l-1, q_l + 2) \quad (3.51)$$

Nous savons que les délais de bout-en-bout sont très inférieurs à la période de scrutation T_{COM} et $\bar{\tau}_l$ est proche de zéro (puisque nous nous intéressons au voisinage du pire cas). Nous pouvons alors écrire : $T_{COM} \gg \bar{\Delta}(l-1, q_l + 2)$ ou encore $-\bar{\Delta}(l-1, q_l + 2) \gg -T_{COM}$.

Ce qui implique que :

$$\bar{\Delta}_{\lim} - \bar{\Delta}(l-1, q_l + 2) \gg -T_{COM}$$

En injectant cette inéquation dans l'inégalité (3.51), nous trouvons :

$$(q_l + 1) \cdot T_{COM} > (q_{\max} + 1) \cdot T_{COM} - T_{COM},$$

ce qui donne : $(q_l + 1) > q_{\max}$.

Comme q_{\max} est par définition la borne maximale de q_l , nous avons forcément $q_l = q_{\max}$.

En remplaçant q_l par q_{\max} dans (3.51), nous obtenons finalement : $\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\lim}$ \square

Corollaire 3.1 : le calcul de la probabilité $P(D_r(l) \geq D_r^{\lim})$ est donné par :

$$P(D_r(l) \geq D_r^{\lim}) = P(q_l = q_{\max}) \cdot P(\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\lim})$$

Preuve :

D'après le lemme 3.1, le corollaire est vrai si la proposition $(q_l = q_{\max})$ et la proposition $\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\lim}$ sont aléatoirement indépendantes.

Rappelons que le calcul de q_l passe par la condition : $r \cdot q_l > \Gamma(i, l) + \lfloor \alpha \rfloor + \beta$. Or, cette condition implique des délais qui sont subis lors du $l^{\text{ème}}$ cycle de scrutation. La condition

$\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\text{lim}}$ implique quant à elle les délais du $(l-1)^{\text{ème}}$ cycle et du $(l+q_l)^{\text{ème}}$ cycle. Comme il n'y a pas d'interaction entre les requêtes envoyées durant des cycles différents (conséquence de l'hypothèse H6), les délais qu'elles subissent aussi sont indépendants. \square

Au final, le calcul de la probabilité $P(D_r(l) \geq D_r^{\text{lim}})$ revient à calculer $P(q_l = q_{\text{max}})$ et $P(\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\text{lim}})$.

3.2.2 Calcul analytique de la densité de probabilité du temps de réponse

Comme évoqué plus haut, la probabilité $P(D_r(l) \geq D_r^{\text{lim}})$ est simplement le produit de deux probabilités : $P(q_l = q_{\text{max}})$ et $P(\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\text{lim}})$. Nous allons les calculer en utilisant les résultats de la section 3.1.

i) $P(q_l = q_{\text{max}})$?

Cas $r \in \mathbb{N}^+$: Pour des raisons de complexité évoquée précédemment, nous commençons par $r \in \mathbb{N}^+$.

Le fait que $q_l = q_{\text{max}}$ est équivalent à : $r \cdot q_{\text{max}} > 1 + \lfloor \alpha \rfloor + \beta \geq r \cdot (q_{\text{max}} - 1)$. Or, comme $\lfloor \alpha \rfloor$ et r sont des nombres entiers et $\beta < 1$, ceci est équivalent à : $1 + \lfloor \alpha \rfloor = r \cdot (q_{\text{max}} - 1)$ ou encore $\lfloor \alpha \rfloor = r \cdot (q_{\text{max}} - 1) - 1$. Ceci conduit finalement à : $r \cdot (q_{\text{max}} - 1) > \alpha \geq r \cdot (q_{\text{max}} - 1) - 1$.

En posant $\alpha_0 = r \cdot (q_{\text{max}} - 1) - 1$ et sachant, d'après (3.40), que : $\alpha = \max[T_{A/R}, \sum_{i=1}^N T_{EM_i}] / T_{CPU}$,

$$\text{nous aboutissons à : } (\alpha_0 + 1) \cdot T_{CPU} > \max[T_{A/R_S}, \sum_{i=1}^N T_{EM_i}] \geq \alpha_0 \cdot T_{CPU} \quad (3.52)$$

On peut vérifier aisément que cette inégalité peut aussi s'écrire :

$$\begin{cases} (\alpha_0 + 1) \cdot T_{CPU} > T_{A/R_S} \geq \alpha_0 \cdot T_{CPU} & \text{si } \sum_{i=1}^N T_{EM_i} < \alpha_0 \\ (\alpha_0 + 1) \cdot T_{CPU} > T_{A/R_S} & \text{si } \alpha_0 < \sum_{i=1}^N T_{EM_i} < \alpha_0 + 1 \end{cases} \quad (3.53)$$

En supposant que la densité de probabilité du temps d'aller-retour T_{A/R_S} est $f_{A/R}$, la probabilité recherchée est donnée par :

$$P(q_l = q_{\text{max}}) = \begin{cases} \int_{\alpha_0 \cdot T_{CPU}}^{(\alpha_0 + 1) \cdot T_{CPU}} f_{A/R}(t) \cdot dt & \text{si } \sum_{i=1}^N T_{EM_i} < \alpha_0 \\ \int_0^{(\alpha_0 + 1) \cdot T_{CPU}} f_{A/R}(t) \cdot dt & \text{si } \alpha_0 < \sum_{i=1}^N T_{EM_i} < \alpha_0 + 1 \end{cases} \quad (3.54)$$

- ii) $P(\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\text{lim}})$?

Le calcul de cette probabilité est bien plus facile. Rappelons que la densité de probabilité d'une somme $z = x + y$ de deux variables aléatoires indépendantes x, y dont les densités de probabilité respectives sont f_x, f_y est donnée par la convolution :

$$f_z(t) = (f_x * f_y)(t) = \int_{-\infty}^{+\infty} f_x(t-u)f_y(u) \cdot du \quad (3.55)$$

En posant : $x = T_{\text{Req}_D}(l + q_l)$, $y = -T_{\text{Req}_S}(l-1)$, $z = -\bar{r}_l$, nous avons :

$$\bar{\Delta}(l-1, q_l + 2) = x + y + z \quad (3.56)$$

Le calcul de la densité de probabilité de cette somme est donné par :

$$f_{\bar{\Delta}}(t) = (f_x * f_y * f_z)(t) = \int_{-\infty}^{+\infty} f_z(t-w) \int_{-\infty}^{+\infty} f_x(w-u)f_y(u) \cdot du \cdot dw \quad (3.57)$$

La probabilité recherchée a finalement comme expression :

$$P(\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\text{lim}}) = \int_{\bar{\Delta}_{\text{lim}}}^{+\infty} f_{\bar{\Delta}}(t) \cdot dt \quad (3.58)$$

- $r \in \mathbb{Q}^+$: le seul changement résultant du fait de considérer un rapport rationnel, est la condition de calcul de q_l . Rappelons que d'après (3.31), celle-ci est donnée par :

$$r \cdot q_l > \Gamma(i, l) + \lfloor \alpha \rfloor + \beta \text{ où } \Gamma(i, l) = i + 1 - \lceil r \rceil \cdot (l-1) \text{ et}$$

$$\lceil \alpha \rceil < \Gamma(i, l) \leq 1 + \lceil \alpha \rceil \quad (3.59)$$

Le calcul de la probabilité $P(q_l = q_{\text{max}})$ revient à calculer la probabilité d'avoir :

$$\Gamma(i, l) + \lfloor \alpha \rfloor + \beta > r \cdot (q_{\text{max}} - 1) \quad (3.60)$$

Il est clair que cela n'est pas évident vu la complexité de l'expression $\Gamma(i, l) + \lfloor \alpha \rfloor + \beta$. Nous proposons de calculer un majorant $P(q_l = q_{\text{max}})^+$ de la probabilité recherchée en acceptant un peu de pessimisme en prenant le minorant de $\Gamma(i, l)$ dans (3.60), soit $\lceil \alpha \rceil$. La condition (3.60) devient alors :

$$\lceil \alpha \rceil + \lfloor \alpha \rfloor + \beta > r \cdot (q_{\text{max}} - 1),$$

ou encore :

$$\alpha + \beta > r \cdot (q_{\text{max}} - 1) \quad (3.61)$$

Sachant que le CPU est indépendant de la COM, les paramètres α et β sont aussi indépendants. La densité de probabilité de la somme $(\alpha + \beta)$ est donnée par :

$$f_{(\alpha+\beta)}(t) = (f_\alpha * f_\beta)(t) = \int_{-\infty}^{+\infty} f_\alpha(t-u)f_\beta(u) \cdot du \quad (3.62)$$

où f_α et f_β sont les densités respectives de α et β .

Sachant que $\alpha_0 = r \cdot (q_{\max} - 1) - 1$, la probabilité recherchée est donnée par :

$$P(q_l = q_{\max})^+ = (f_\alpha * f_\beta)(t) = \int_{\alpha_0}^{+\infty} f_{(\alpha+\beta)}(t) \cdot dt \quad (3.63)$$

- **Remarque 3.3** : la probabilité $P(q_l = q_{\max})$ est recherchée uniquement dans le cas où $q_{\max} \geq 2$. Dans le cas où $q_{\max} = 1$, $q_l = 1$ est toujours vrai et la probabilité $P(q_l = q_{\max})$ est égale à 1. Par conséquent, la formule de calcul du temps de réponse devient :

$$D_r(l) = 2 \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + \bar{\Delta}(l-1, q_l + 2) + T_{E/S_D} \quad (3.64)$$

Cette formule implique un seul terme variable (la gigue $\bar{\Delta}$) et une expression constante C :

$$D_r(l) = C + \bar{\Delta}(l-1, q_l + 2) \quad (3.65)$$

$$\text{où } C = 2 \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + T_{E/S_D}$$

Finalement, la fonction de densité de probabilité du temps de réponse est égale à celle de la gigue, décalée de la constante C , soit :

$$f_{D_r}(t) = f_{\bar{\Delta}}(t - C) \quad (3.66)$$

3.3 Analyse de sensibilité du temps de réponse

Dans cette section, nous allons analyser la sensibilité du temps de réponse maximal aux différents délais élémentaires. Dans un premier temps, nous nous penchons sur la sensibilité aux délais dus à la non synchronisation des composants du SCR, puis dans un deuxième temps à la sensibilité aux délais réseau de bout en bout.

3.3.1 Sensibilité aux délais de non synchronisation

Pour analyser la sensibilité aux délais de non synchronisation, nous considérons les pires cas de non synchronisation dans tous les niveaux du SCR en repassant par les étapes 1 à 8 :

- Lorsqu'un événement apparaît, il attend l'arrivée d'une requête en provenance du contrôleur. Le pire délai d'attente est obtenu lorsque l'événement apparaît un instant après l'arrivée d'une requête. De ce fait, il doit attendre l'arrivée de la prochaine

requête, soit un délai moyen d'attente égal à la période de scrutation T_{COM} (plus la gigue du réseau dans le pire cas).

- L'événement traverse le réseau et y subit un délai maximal T_{Rep_S} .
- Une fois dans la COM puis dans la mémoire tampon, l'événement attend le début d'un nouveau cycle CPU. Le pire cas est obtenu lorsque l'événement arrive un instant après le début d'un cycle CPU. Il doit donc attendre un délai maximal égal à T_{CPU} .
- Une fois pris en compte, il est utilisé dans le programme de commande pour fabriquer l'événement de réaction et le mettre en mémoire tampon. Tout cela prend un délai total égal à T_{CAL} .
- Une fois le signal de réaction dans la mémoire, il doit attendre le début d'un nouveau cycle de scrutation. Le pire cas est quand un cycle débute un moment avant la fabrication de la réaction, soit un pire délai de non synchronisation égal à T_{COM} .
- La réaction étant prise en compte, elle est envoyée dans une requête vers la destination après un temps $\sum_{i=1}^D T_{EM_i}$. Pour traverser le réseau, il faut un délai maximal égal T_{Req_D} .
- Enfin, la requête est traitée par le MES de destination durant un temps égal à T_{E/S_D} .

En récapitulant les étapes 1-8 et les pires délais correspondants, nous aboutissons à la formule des pires cas suivante :

$$D_r^{pire} = T_{COM} + \Delta_{S_max} + T_{Rep_S} + T_{CPU} + T_{CAL} + T_{COM} + \sum_{i=1}^D T_{EM_i} + T_{Req_D} + T_{E/S_D} \quad (3.67)$$

Cette formule peut être réécrite :

$$D_r^{pire} = [2 + (1 + \beta) / r] \cdot T_{COM} + \Delta_{S_max} + T_{Rep_S} + \sum_{i=1}^D T_{EM_i} + T_{Req_D} + T_{E/S_D} \quad (3.68)$$

En considérant l'approximation :

$$\Delta_{S_max} + T_{Rep_S} \approx T_{Req_S_max} - T_{Req_S_min} + T_{Rep_S_max} \approx T_{A/R} - T_{Req_S_min} = \alpha / r \cdot T_{COM} - T_{Req_S_min}$$

et sachant que : $T_{A/R} = \alpha / r \cdot T_{COM}$, nous aboutissons à la formule de pires cas :

$$D_r^{pire} = [2 + (1 + \beta + \alpha) / r] \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} + \Delta_{max} + T_{E/S_D} \quad (3.69)$$

Proposition 3.3 : la formule des pires cas (3.69) donne un majorant pessimiste de la borne maximale du temps de réponse i.e. $D_r^{pire} > D_r^{MAX}$.

Preuve :

Nous savons que la borne maximale du temps de réponse est donnée par :

$$D_r^{MAX} = (q_{\max} + 1) \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + \Delta_{\max} + T_{E/S_D} \quad (3.70)$$

où $r \cdot q_{\max} > 1 + \lfloor \alpha \rfloor + \beta > r \cdot (q_{\max} - 1)$ dans le cas $r \in \mathbb{N}^+$ et $r \cdot q_{\max} > 1 + \alpha + \beta$ (condition simplificatrice pouvant être pessimiste) dans le cas $r \in \mathbb{Q}^+$.

Le temps de réponse des pires cas peut être écrit en fonction de la borne maximale :

$$D_r^{pire} = D_r^{MAX} + [1 + (1 + \beta + \alpha) / r - q_{\max}] \cdot T_{COM} + \sum_{i=1}^S T_{EM_i} \quad (3.71)$$

La preuve recherchée est trouvée si le terme $[1 + (1 + \beta + \alpha) / r - q_{\max}] \cdot T_{COM} + \sum_{i=1}^S T_{EM_i}$ est positif.

La condition précédente dans le cas $r \in \mathbb{N}^+$ peut s'écrire :

$$q_{\max} > [1 + \lfloor \alpha \rfloor + \beta] / r > q_{\max} - 1$$

Ou encore :

$$q_{\max} > [1 + \lfloor \alpha \rfloor + \beta] / r > q_{\max} - 1 ,$$

ou bien :

$$1 + q_{\max} > 1 + [1 + \lfloor \alpha \rfloor + \beta] / r > q_{\max} \quad (3.72)$$

Le nombre α étant positif et donc $\alpha \geq \lfloor \alpha \rfloor$, la condition (3.72) implique que :

$$1 + [1 + \alpha + \beta] / r > q_{\max} \quad (3.73)$$

Il est clair que cela implique que le terme précédent est positif et donc $D_r^{pire} > D_r^{MAX}$.

La preuve dans le cas $r \in \mathbb{Q}^+$ est encore plus évidente en utilisant la condition simplificatrice, pouvant être pessimiste par rapport au cas exact. Ce qui prouve bien que la formule des pires cas est plus pessimiste que la formule exacte dans tous les cas. \square

En conclusion, la méthode des pires cas de non synchronisation est pessimiste par rapport à la méthode analytique exacte que nous avons proposée. La démonstration ne donnant pas un ordre de grandeur de ce pessimisme, voyons cela sur un exemple concret.

Soit les valeurs numériques suivantes (données à titre explicatif) : $T_{COM} = 10ms$, $T_{CPU} = 5ms$, $r = 2$, $\alpha = 0.4$, $\beta = 0.6$, $q_{\max} = 1$, $S = D = 1$, $T_{EM_i} = 0.1ms$, $\Delta_{\max} = 1ms$, $T_{E/S_D} = 0.6ms$. Le calcul exact de la borne maximale donne $D_r^{MAX} = 21.6ms$ et le

majorant de la méthode des pires cas donne : $D_r^{pire} = 31.7ms$. La surestimation de la borne maximale exacte est d'environ 47%. Il est clair que cela est énorme et inacceptable. En effet, les conséquences que cela engendre sur la qualité de commande, comme nous le verrons par la suite dans la section 3.3.3, peuvent s'avérer très néfastes.

3.3.2 Sensibilité aux délais de bout-en-bout

Il est évident que les délais de bout-en-bout ont un impact direct sur la borne maximale du temps de réponse. La formule de calcul inclut en effet un terme (la gigue Δ qui inclut T_{Req_D}) qui les représente. Ce terme T_{Req_D} reste cependant largement inférieur au reste de la formule puisque les délais de bout-en-bout sont de loin inférieurs à la période de scrutation T_{COM} . La sensibilité du temps de réponse reste alors assez limitée par rapport au délai T_{Req_D} . Gardons à l'esprit cependant que la condition de calcul du nombre q_l inclut le terme α qui représente l'autre délai réseau $T_{A/R}$. Sachant que q_l est un entier, le temps de réponse y est très sensible puisqu'il peut subir des changements très brusques. Voyons cela sur un exemple :

Soit les valeurs numériques des paramètres : $T_{COM} = 10ms$, $T_{CPU} = 5ms$, $r = 2$, $T_{A/R} = 4.6ms$, $\alpha = 0.82$, $\beta = 0.6$, $S = D = 1$, $T_{EM_i} = 0.1ms$, $\Delta_{max} = 1ms$, $T_{E/S_D} = 0.6ms$.

Puisque $r \in \mathbb{N}^+$, le nombre q_{max} se calcule en utilisant la condition : $r \cdot q_{max} > 1 + \lfloor \alpha \rfloor + \beta > r \cdot (q_{max} - 1)$. On peut vérifier que pour les valeurs numériques considérées, $q_{max} = 1$ et le temps de réponse maximal est $D_r^{MAX} = 21.6ms$. Supposons cette fois que le délai d'aller-retour $T_{A/R}$ augmente de seulement 10%, soit atteignant la valeur de $T_{A/R} = 5.06ms$. Il s'ensuit que $\alpha = 1.012$. La partie entière $\lfloor \alpha \rfloor$ bascule alors de 0 à 1 et par conséquent le nombre entier q_{max} vérifiant la condition aussi bascule de 1 vers 2. La borne maximale du temps de réponse est donc égale cette fois à $D_r^{MAX} = 31.6ms$, soit une augmentation d'environ 46%.

En résumé, une augmentation d'un délai réseau de 10% peut impliquer une augmentation de la borne maximale du temps de réponse d'environ 46%.

Conséquence : une évaluation légèrement pessimiste des délais de bout-en-bout peut conduire à une évaluation très pessimiste de la borne maximale du temps de réponse. Par conséquent, les délais de bout-en-bout doivent être évalués de la manière la moins pessimiste possible.

3.3.3 Conséquence de la sensibilité du temps de réponse sur la qualité de la commande

Intuitivement, une bonne évaluation du temps de réponse signifie que les valeurs calculées soient proches des valeurs réelles. Mais quel est concrètement l'impact de cela sur les applications ?

- Si l'application concerne un système d'automatisation par exemple, il s'agit simplement de s'assurer que le temps de réponse ne dépasse jamais une valeur maximale critique. L'évaluation dans ce cas est en quelque sorte une vérification d'une propriété. Si elle est respectée, le SCR est validé et il peut être mis en marche avec confiance. Si ce n'est pas le cas, des mesures s'imposent pour réduire le temps de réponse (ex : reconfigurer le SCR, remplacer les composants par d'autres plus performants, ...). Les conséquences d'une évaluation pessimiste de la borne maximale du temps de réponse peuvent alors conduire à prendre des mesures qui peuvent coûter cher alors que celles-ci ne sont pas forcément nécessaires !

- Si l'application concerne du contrôle commande d'un système (ex : régulation), le raisonnement est un peu plus compliqué. Comme énoncé dans le Chapitre 1, il existe deux approches de commande des systèmes en réseau. La première approche consiste à synthétiser la loi de commande une bonne fois pour toute et le réseau est géré de manière à minimiser les délais afin de satisfaire tout le temps les exigences de la commande. On parle dans ce cas de *commande du réseau*. L'autre approche est la *commande en réseau*. Elle consiste à synthétiser la loi de commande en s'adaptant aux conditions du réseau sans toucher à ce dernier. La commande en réseau nécessite donc l'évaluation des délais subis dans le SCR. La question qui en découle est alors : quel est l'impact de la précision de cette évaluation sur la qualité de commande ? Nous allons montrer cela sur un exemple simple. Faisons remarquer cependant que l'objectif du paragraphe ci-après n'est pas d'apporter une contribution majeure quant à la synthèse de lois de commande des systèmes à retard mais juste de donner un aperçu sur les conséquences d'une mauvaise évaluation du temps de réponse sur la qualité de commande.

La Fig. 3.4 montre une représentation classique d'une boucle de commande où $C(s)$ est la fonction de transfert du contrôleur et $G(s)$ celle du système commandé. L'objectif de cette commande est de réguler la sortie y pour suivre une valeur de référence ref .

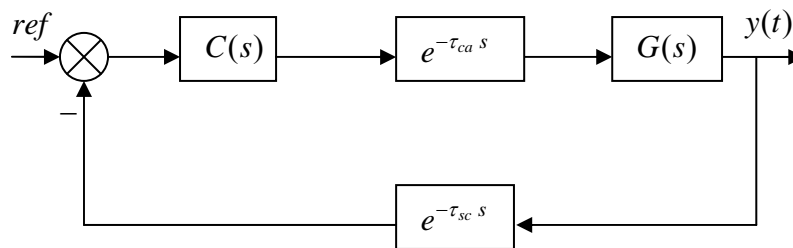


Fig. 3.4 : Boucle de commande classique

Comme nous pouvons le constater, il y a deux retards (ou délais) introduits dans la boucle de commande : le délai τ_{ca} contrôleur \rightarrow actionneur et le délai τ_{sc} capteur \rightarrow contrôleur. Faisons remarquer que contrairement à ce qui est considéré le plus souvent dans l'étude des SCR, ces deux délais ne contiennent pas uniquement les délais dus au réseau de communication mais bien plus comme nous l'avons montré tout au long de ce manuscrit. Par exemple, le délai τ_{ca} inclut aussi bien le délai du réseau que le délai de calcul dans le CPU ou même le délai de non synchronisation entre le CPU et la COM. Au final, la somme des deux délais τ_{ca} et τ_{sc} est égale au temps de réponse : $D_r = \tau_{ca} + \tau_{sc}$. La fonction de transfert de la boucle de commande entre ref et y est :

$$F(s) = \frac{G(s) \cdot e^{-\tau_{ca} s} \cdot C(s)}{1 + G(s) \cdot C(s) \cdot e^{-D_r s}} \quad (3.74)$$

Remarquons que le temps de réponse est introduit dans le dénominateur de la fonction de transfert. Nous pouvons d'ores et déjà en déduire intuitivement toute son importance dans la dynamique et la qualité de commande.

Pour ce genre de système (à retards), des lois de commande avec des stratégies de compensations de retards sont introduites pour contrer l'effet négatif des délais. La plus connue est ce qu'on appelle la commande avec un *prédicteur de Smith* (Smith, 1959). Une boucle de commande avec ce prédicteur est donnée sur la Fig. 3.5.

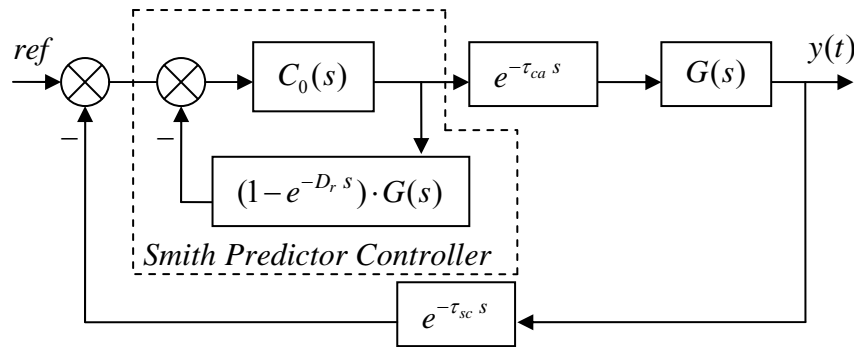


Fig. 3.5 : Boucle de commande avec Predictor de Smith

La commande avec un prédicteur de Smith classique contient une commande $C_0(s)$ qu'on synthétise en l'absence des retards et un retour contenant la valeur estimée de la somme des deux délais τ_{ca} et τ_{sc} , soit l'estimation du temps de réponse. La simplicité de cette commande a fortement contribué à son succès (Hägglund, 1996), (Astrom *et al*, 1994). Cependant, l'estimation en ligne du temps de réponse n'est pas aisée. On peut remédier à cela en considérant seulement une estimation hors ligne de la moyenne. Cela n'est malheureusement pas sans risques pour la stabilité du système comme nous le verrons. La solution la plus radicale est de prendre simplement la valeur maximale du temps de réponse afin d'assurer la stabilité. C'est à ce stade que la précision de l'évaluation de la borne maximale devient cruciale. Voyons cela encore une fois sur un exemple concret.

Soit un système à commander de premier ordre dont la fonction de transfert est $G(s) = \frac{1}{1+T_i s}$. Pour suivre exactement la référence en régime permanent sans déviation, nous proposons un correcteur PI (Proportionnel Intégral) en ignorant les retards dans un premier temps. La fonction de transfert est de la forme $C_0(s) = K \frac{1+T_i s}{s}$.

La loi de commande implémentée dans le contrôleur étant échantillonnée, nous devons utiliser un modèle échantillonné du système. Nous utilisons une période d'échantillonnage égale à la période de scrutation, soit $10ms$ pour cet exemple. Pour les valeurs numériques $K = 60$, $T = T_i = 20ms$, la fonction de transfert échantillonnée de $G(s)$ est égale à $G(z) = \frac{1+z}{5z-3}$. La

commande PI correspondante est $G(z) = 60 \frac{25-15}{z-1}$.

La dynamique de la commande PI pour une entrée de référence carrée, en l'absence de retards, est représentée sur la Fig. 3.6.

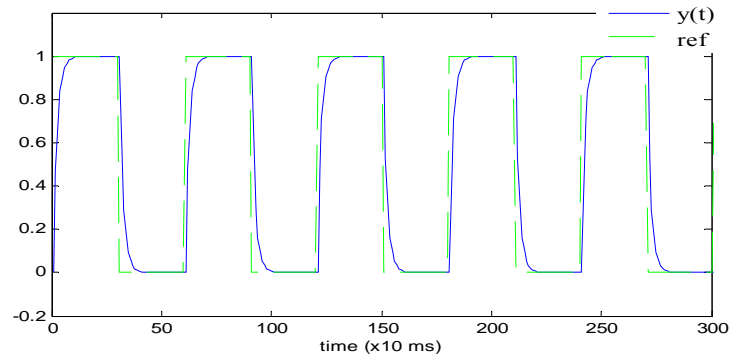


Fig. 3.6 : Dynamique de la commande PI en absence des délais

Les délais sont évidemment toujours présents en pratique. Voyons ce qui se passe en les prenant en compte. Pour un SCR avec les paramètres $T_{COM} = 10ms$, $T_{CPU} = 5ms$, $r = 2$, $\alpha = 0.4$, $\beta = 0.6$, $q_{max} = 1$, $S = D = 1$, $T_{EM_i} = 0.1ms$, $\Delta_{max} = 1.5ms$, $T_{E/S_D} = 0.6ms$, les temps de réponse sur une période de 300 x 10ms (limité à 300 périodes de scrutation pour la clarté des figures) sont représentés sur la Fig. 3.7.

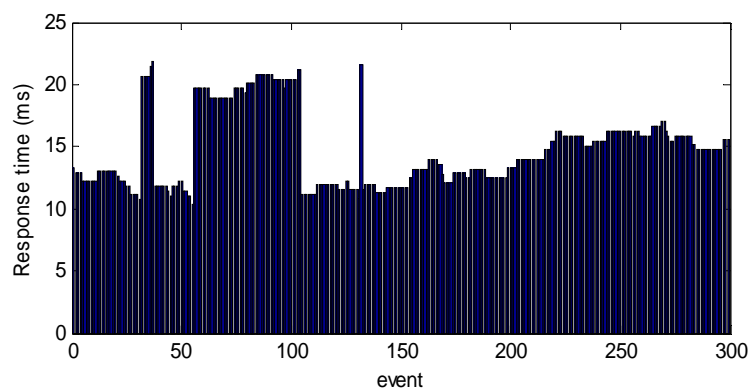


Fig. 3.7 : Temps de réponse du SCR

L'utilisation d'une estimation hors ligne de la moyenne du temps de réponse (environ 16ms) pour la synthèse du prédicteur de Smith donne le résultat de la Fig. 3.8.

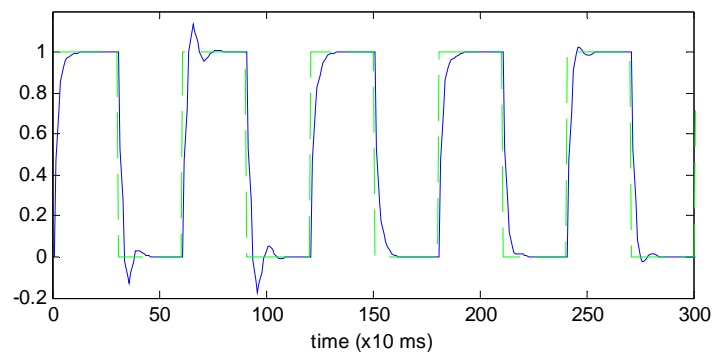


Fig. 3.8 : Dynamique de commande avec la moyenne du temps de réponse dans le predicteur de Smith

On peut remarquer que la dynamique de la commande PI est satisfaisante dans les périodes où le temps de réponse réel est inférieur à l'estimation (moyenne). Dès que ce n'est pas le cas en revanche, on remarque des dépassements pouvant aller jusqu'à 20% de la consigne. Ce phénomène est bien sûr inacceptable et il convient de l'éviter.

Pour y remédier, une estimation de la borne maximale est souvent utilisée dans le prédicteur de Smith pour privilégier la stabilité. Mais quelle valeur de la borne faut-il utiliser ? Le calcul exact de la borne maximale en utilisant les formules donne $D_r^{MAX} = 22.1ms$ et la méthode des pires cas conduit à $D_r^{pire} = 32.2ms$. Le résultat de la commande en utilisant ces deux estimations est donné sur la Fig. 3.9.

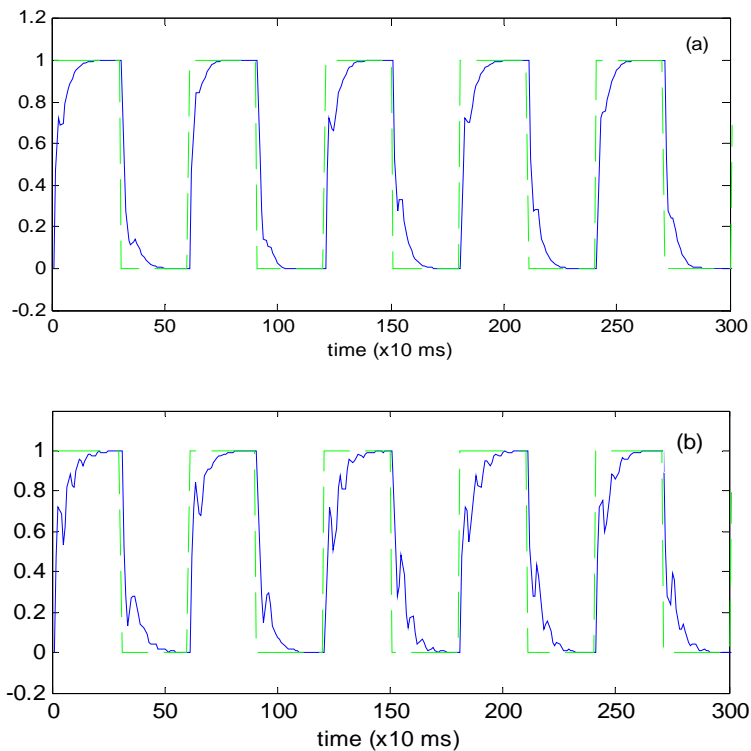


Fig. 3.9 : Dynamique de commande (a) borne maximale du temps de réponse donnée par les formules (b) borne maximale donnée par la méthode des pires cas

Comme nous pouvons le constater sur la Fig. 3.9, l'utilisation de la borne maximale donnée par les formules procure un résultat assez satisfaisant même si la dynamique de la commande est quelque peu ralentie (c'est le prix à payer pour assurer la stabilité). Le résultat de la borne des pires cas quant à lui n'est pas aussi satisfaisant. Non seulement la dynamique de la commande est fortement ralentie, mais de fortes oscillations indésirables sont apparues.

Conclusion du chapitre

En conclusion, qu'il s'agisse d'un système d'automatisation en réseau ou de contrôle commande en réseau, une bonne évaluation du temps de réponse, la moins pessimiste possible, est recherchée. Comme nous l'avons vu sur des exemples, les formules analytiques développées dans cette thèse donnent des évaluations meilleures que celles de la méthode des pires cas.

Dans l'absolu, l'efficacité des formules ne peut être démontrée qu'en comparant leurs évaluations à des mesures expérimentales réelles. C'est l'objet du Chapitre 5. Par ailleurs, comme nous l'avons vu, une utilisation efficace des formules de calcul du temps de réponse est conditionnée par une bonne évaluation des délais réseau de bout-en-bout. Le prochain chapitre est consacré aux délais de bout-en-bout dans le cas particulier des réseaux Ethernet commuté.

Evaluation des délais bout en bout dans les SCR

Cas d'Ethernet commuté

Sommaire

4.1 Introduction	85
4.1.1 Pourquoi Ethernet ?	85
4.1.2 La micro segmentation et le fonctionnement d'un commutateur Ethernet	86
4.2 Evaluation des délais de bout-en-bout dans un réseau Ethernet commuté	87
4.2.1 Etat de l'art	87
4.2.2 Analyse des scénarii d'interférence dans un SCR à base d'Ethernet	89
4.3 Déroulement itératif d'un scenario de traversée d'un réseau Ethernet	94
4.3.1 Modélisation d'un commutateur à l'aide de GETC	94
4.3.2 Représentation d'état (max, +) des GETC	97
4.3.3 Modèle en équations (max, +) récurrentes d'un réseau Ethernet commuté . . .	101
4.4 Calcul de la borne maximale d'un délai de bout-en-bout	105
4.4.1 Méthode de calcul combinatoire	105
4.4.2 Algorithmes génétiques comme alternative pour des SCR de grande taille . .	106

Dans ce chapitre, nous donnons suite à l'analyse de sensibilité réalisée dans le chapitre précédent pour proposer une méthode précise d'évaluation des pires délais de bout-en-bout dans le cas d'un réseau Ethernet commuté. Pour ce faire, nous proposons d'abord une analyse des interférences des paquets dans ce réseau pour identifier le pire scénario de traversée du réseau par un paquet donné. Ensuite, nous modélisons le réseau à l'aide d'une classe particulière de RdP que nous appelons les GETC et exprimons son évolution à l'aide d'équations (Max,+). En utilisant ces équations, nous pouvons dérouler le pire scénario et calculer le délai de bout-en-bout correspondant. Enfin, comme la procédure d'identification, mentionnée précédemment, du pire scénario est basée sur un calcul combinatoire, pouvant être coûteux en temps de calcul pour les SCR de grande taille, nous proposons les algorithmes génétiques comme alternative. Un exemple de SCR avec comparaison des deux méthodes est donné pour illustrer cet aspect.

~ You may delay, but time will not ~ Benjamin Franklin

4.1 Introduction

4.1.1 Pourquoi Ethernet commuté?

Si Ethernet s'était imposé très rapidement comme le protocole réseau LAN (Local Area Network) le plus utilisé dans les réseaux informatiques, son introduction dans le milieu industriel n'était pas aussi évidente malgré les très nombreux avantages qu'il procure. Par exemple, Decotignie (2001) présente sept bonnes raisons d'intégrer Ethernet comme réseau de terrain dans le milieu industriel :

- le coût et la multitude des circuits intégrés,
- la compatibilité avec Internet et TCP/IP,
- les applications et protocoles classiques (HTTP, FTP, ...) peuvent être réutilisés,
- l'évolution constante de la bande passante,
- l'universalité devient possible (une solution unique plutôt que plusieurs réseaux de terrain),
- les réseaux traditionnels ont atteint leurs limites,
- le marché en demande de plus en plus.

Pourtant, Ethernet a eu du mal à faire une percée dans le milieu industriel. Et pour cause, Ethernet présentait à l'origine un inconvénient de taille : le *non déterminisme*. Le mécanisme de contrôle d'accès au medium d'Ethernet classique se base en effet sur le CSMA/CD (Carrier Sense Multiple Access with Collision Detection) qui n'est pas prévisible à cause des collisions qu'il génère. Avec ce mécanisme, une station écoute le medium et si aucune autre station n'est en train d'émettre, elle peut commencer à envoyer ses messages. Le problème est qu'il peut arriver que plusieurs stations commencent à émettre en même temps et par conséquent provoquent des collisions. Pour limiter cela, l'algorithme de résolution de collisions BEB (Binary Exponential Backoff) est appliqué. Après une collision, chaque station attend un laps de temps aléatoire (*Backoff*) et réécoute le medium pour savoir si elle peut recommencer à émettre. Malheureusement, même avec cela, rien ne garantit que d'autres collisions ne se reproduisent pas indéfiniment. Pire encore, d'autres phénomènes comme l'effet de capture (*Capture Effect*) peuvent se produire (Alves *et al.*, 2000). Une station occupe exagérément le medium alors que d'autres stations veulent émettre. Ceci pose alors le problème d'équité de partage du medium. En conséquence de tous ces phénomènes, le temps d'attente pour une station n'est pas prévisible et peut même théoriquement être infini. On parle alors de *non déterminisme*. Ceci est évidemment inacceptable dès lors qu'il s'agit d'un système temps-réel comme les SCR où les délais doivent être bornés.

Des solutions alternatives devaient donc être trouvées. Par exemple, Alves *et al.* (2000) dressent les principales tentatives qui ont été faites pour remédier à ces problèmes de non déterminisme d'Ethernet pour qu'il puisse prendre place dans le milieu industriel. L'augmentation de la vitesse de transmission dans les composants Ethernet a été considérable (Giga Ethernet avec la norme IEEE 802.3z) mais cela n'a pas suffi. La révolution dans Ethernet est incontestablement l'introduction des commutateurs (norme IEEE 802.1D) avec le mécanisme de micro segmentation et du full-duplex.

4.1.2 La micro segmentation et le fonctionnement d'un commutateur

La micro segmentation consiste à segmenter le réseau en plusieurs domaines de diffusion (ou domaines de collision) en isolant chaque station sur son propre segment (un port du commutateur). Ainsi, le problème des collisions de messages provenant simultanément de plusieurs stations est limité à chaque port où chaque station peut, soit émettre, soit recevoir et non les deux à la fois. On parle alors de transmission en mode Half-duplex. Mieux encore avec le full-duplex, les collisions sont définitivement éliminées et le transfert des données peut se faire dans les deux directions simultanément (une paire de fils au lieu d'un), doublant théoriquement par la même occasion la bande passante pour chaque station. Un commutateur peut être vu comme un concentrateur intelligent où chaque message est mis dans une file d'attente dans un port d'entrée avant d'être routé vers le seul port de sortie de destination (Fig. 4.1).

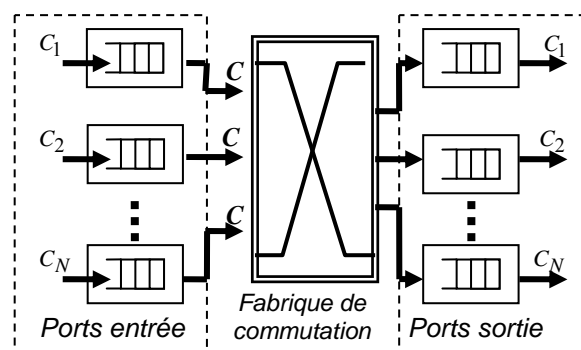


Fig. 4.1 : Schéma de fonctionnement d'un commutateur

Si le problème de collisions est définitivement résolu, un problème de congestion intervient lorsque plusieurs ports d'entrée veulent émettre simultanément des paquets vers les ports de sortie. Comme la vitesse de la fabrique de commutation est limitée, même si celle-ci est souvent de loin plus importante que la vitesse des ports, un paquet dans une file d'attente subit inévitablement un délai, en attendant son tour pour être expédié vers un port de sortie. De

même, le port de sortie peut contenir plusieurs paquets en attente avant leurs retransmissions en dehors du commutateur, entraînant des délais importants puisque la vitesse du port de sortie est relativement limitée.

Notons qu'il existe plusieurs types de commutateurs plus au moins performants et classés suivant plusieurs critères. Par exemple, les commutateurs sont classés suivant la localisation des buffers d'attente des paquets (bufferisation en entrée, en sortie, à mémoire partagée) (Georges *et al.*, 2005b). Ils sont aussi classés suivant le type de mode utilisé pour le routage des paquets : *store & forward*, *cut-through*, et *fragment free*. Contrairement aux deux derniers modes, dans le mode *store & forward*, le début de routage vers le port de sortie d'un paquet ne commence que lorsque celui-ci est entièrement reçu dans le port d'entrée. Ce mode présente l'avantage de pouvoir vérifier la validité du paquet et éventuellement l'écarter avant son expédition vers la sortie.

Ceci dit, quelque soit le type du commutateur, chaque message subit des délais d'attente aux différents niveaux du commutateur. Ceci nous ramène finalement au problème de déterminisme et à la capacité de prédire si un message peut atteindre sa destination avant une certaine échéance. Des méthodes d'évaluation des délais de bout-en-bout (de la source jusqu'à la destination) sont donc requises. Dans le contexte des SCR, contrairement aux réseaux complètement ouverts comme Internet, nombre d'informations sur le mode de génération du trafic (périodique, majoré, ...) sont connues et ceci a aidé considérablement au développement de nombreuses méthodes d'évaluation des délais de traversée de ces réseaux. Dans la section ci-après, nous revenons sur les principaux travaux concernant le cas d'Ethernet commuté.

4.2 Evaluation des délais de bout-en-bout dans un réseau Ethernet commuté

4.2.1 Etat de l'art

Rappelons que nous avons déjà cité dans le Chapitre 1 de nombreux travaux concernant l'évaluation des délais dans les SCR en général et les avons classés en quatre catégories : simulation, mesure, vérification et méthode analytique. Nous allons en rappeler certains ou en citer d'autres, concernant la détermination des délais de bout-en-bout dans le cas d'Ethernet commuté.

Nous rappelons tout d'abord le calcul réseau (Le Boudec *et al.*, 2004), (Georges *et al.*, 2005), (Grieu, 2004), (Scharbarg *et al.*, 2009), et le calcul réseau groupé (Bauer *et al.*, 2009). Rappelons également l'approche par trajectoire introduite dans (Martin *et al.*, 2006) puis améliorée dans (Bauer *et al.*, 2010). D'autres méthodes existent avec des analyses

diverses et variées. Dans (Fan *et al.*, 2008), des algorithmes permettant de simuler le fonctionnement d'un réseau Ethernet commuté en vue de calculer des délais de bout-en-bout sont proposés. Une approche de pire cas, se basant sur des conditions de stabilité du réseau Ethernet commuté, a été présentée dans (Lee *et al.*, 2002). Une méthode analytique et comparative entre différentes topologies de réseau Ethernet a été proposée dans (Rüping *et al.*, 1999). Il a été conclu que la topologie hiérarchique est souvent la meilleure. Dans (Jasperneite *et al.*, 2001) et (Diouri *et al.*, 2007), l'outil de simulation réseau OPNET est utilisé pour analyser le comportement des réseaux Ethernet et calculer des délais de bout-en-bout.

Il faut noter par ailleurs que beaucoup de travaux basés sur l'expérimentation ont été proposés, notamment dans (Ferrari *et al.*, 2006a), (Schafer *et al.*, 2007), et (Silvola *et al.*, 2007).

A côté de toutes ces méthodes, il existe également des approches probabilistes. Elles permettent de trouver la probabilité qu'un délai de traversée d'un réseau Ethernet soit supérieur à une certaine valeur ou même de déterminer sa distribution. Parmi les travaux adoptant cette approche, nous trouvons le formalisme des files d'attente stochastiques dans (John, 2005), et (Kamen *et al.*, 1999), des chaînes de Markov dans (Gunter *et al.*, 2006), du calcul réseau probabiliste dans (Scharbarg *et al.*, 2009), et (Starobinski *et al.*, 2000). Dans (Torab *et al.*, 1999), une autre méthode se basant sur l'analyse de la charge des commutateurs a été proposée. Dans ces travaux, la modélisation du réseau est faite à l'aide de graphes couplés avec une matrice de trafic. Un calcul matriciel itératif permet alors de calculer les charges des nœuds du réseau puis d'en déduire les délais de bout-en-bout. Song (2001) présente une approche analytique à point fixe pour calculer des délais de bout-en-bout dans le cas de messages cycliques. Le cas apériodique y est aussi traité en considérant que le flux d'entrée de chaque port suit un processus de Bernoulli.

Il faut noter que d'autres travaux abordant les réseaux Ethernet sous un autre angle ont été menés. Parmi ceux-la, nous trouvons des études se penchant sur l'optimisation des topologies des réseaux afin de minimiser les délais de bout-en-bout (Rondeau *et al.*, 2001), et (Krommenacker *et al.*, 2002a,b). Ces travaux ont été complétés par la suite en intégrant les bornes d'acheminement des paquets comme critère d'optimisation (Georges *et al.*, 2005b).

Dans la section suivante, nous introduisons une nouvelle approche basée sur l'analyse des interférences entre des paquets dans des SCR client/serveur. Il n'est pas sans utilité de rappeler les motivations d'introduction de cette approche alors que nous venons de mentionner l'existence de beaucoup de méthodes d'évaluation des délais de bout-en-bout :

- Premièrement, nous avons montré l'importance de l'enjeu de trouver des délais les moins pessimistes possibles. Or, cela n'est pas toujours vérifié dans les méthodes existantes.

- En considérant l'approche la plus répandue, c.-à-d. le calcul réseau, et en ignorant son pessimisme (la méthode n'est en effet pas toujours pessimiste), nous ne pouvons l'utiliser dans le cas des SCR de notre étude. En effet, les modules déportés E/S émettent des paquets mais uniquement pour répondre aux requêtes qu'ils reçoivent. Un modèle d'émission des paquets depuis les E/S n'est donc pas connu a priori. Par conséquent, trouver une courbe d'arrivée pour ces modules n'est pas chose aisée. De plus, même en supposant l'existence d'une telle courbe, le résultat d'évaluation sera forcément pessimiste puisque les flux émis par les contrôleurs et ceux émis par les modules E/S seront considérés en même temps alors que nous savons qu'une requête et sa réponse ne peuvent coexister dans le réseau.

Nous pouvons faire de même avec le reste des méthodes exposées dans l'état de l'art et pointer de nombreux verrous existant quant à l'évaluation des délais des SCR de nos travaux. Une étude dédiée semble donc nécessaire et c'est l'objet du prochain paragraphe.

4.2.2 Analyse des scénarii d'interférence des paquets dans un SCR à base d'Ethernet

Pour expliquer le principe de notre méthode, considérons le cas du SCR de l'exemple de la Fig. 4.2.

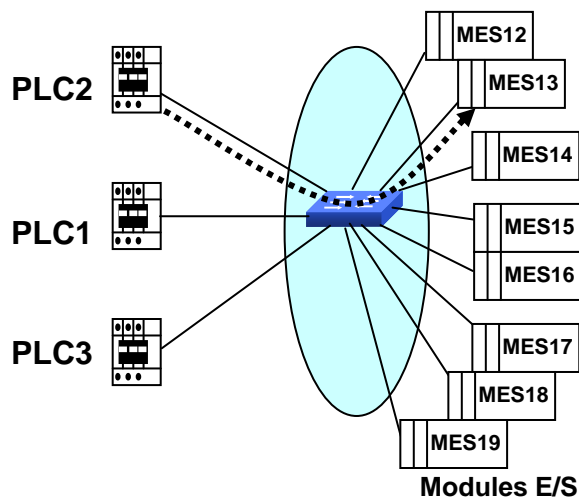


Fig. 4.2 : Exemple de SCR à base d'Ethernet

Dans cet exemple, les contrôleurs scrutent les modules E/S déportés comme suit :

- le PLC2 scrute périodiquement, avec une période T_2 , les modules MES12, MES13, MES14, MES15, et MES16.
- Le PLC1 scrute périodiquement, avec une période T_1 , les modules MES14, MES15, MES16, et MES17.

- Le PLC3 scrute périodiquement, avec une période T_3 , les modules MES16, MES17, MES18, et MES19.

Au début de son cycle de scrutation, chaque contrôleur envoie une rafale de paquets vers les modules qu'il scrute. Ce mécanisme se répète de manière périodique. Les contrôleurs n'étant pas synchronisés, chacun d'entre eux commence son cycle initial indépendamment des autres. Il y a donc des décalages entre les débuts d'envoi des rafales depuis les différents contrôleurs. Or, un décalage plutôt qu'un autre conduit à une interférence différente entre les rafales. Ceci induit alors des délais différents de traversée du commutateur par les paquets. La question qui se pose alors est : pour un paquet donné, quel est le scénario (ou décalages) qui conduit à son délai de traversée maximal ? Pour répondre à cette question, faut-il aussi pouvoir analyser les différents scénarii d'interférence.

Cette analyse est relativement simplifiée dans le contexte des SCR que nous étudions. L'hypothèse H6 posée dans le Chapitre 1 s'avère déterminante. En effet, en supposant que les périodes de scrutation soient de loin supérieures aux délais de bout-en-bout, nous pouvons affirmer qu'une rafale envoyée par un contrôleur n'interfère qu'avec une autre rafale au maximum, envoyée depuis un autre contrôleur. En effet, un paquet d'une rafale interférant avec un autre paquet d'une rafale parallèle aura toujours atteint sa destination avant l'arrivée de la rafale parallèle suivante. Lors de l'analyse des interférences, une rafale de chaque contrôleur suffit alors. Considérons le schéma de la Fig. 4.3 pour analyser le scénario conduisant au pire délai de traversée du second paquet envoyé par le contrôleur PLC2, soit le paquet hachuré f_2^2 .

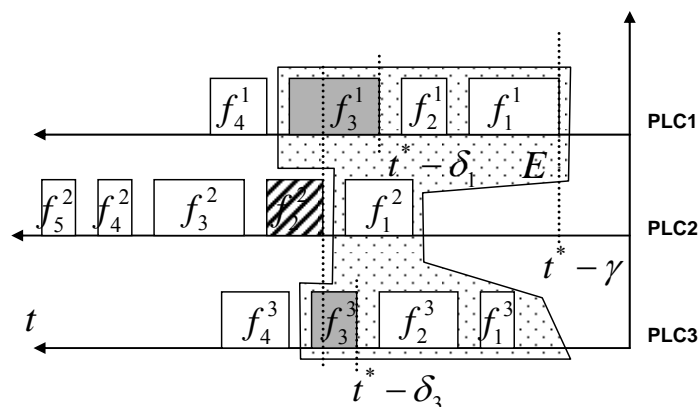


Fig. 4.3 : Analyse des interférence des paquets

En prenant comme origine du temps la date d'entrée dans le commutateur du paquet f_2^2 , notée t^* , le début d'envoi des rafales depuis le PLC1 se fait à l'instant $(t^* - \gamma_1)$ et depuis le PLC3 à l'instant $(t^* - \gamma_3)$. Sachant que nous considérons un ordonnancement de type FIFO

(First In First Out), le paquet f_2^2 subit un délai de traversée D qui dépend exclusivement des paquets qui sont arrivés avant lui dans le commutateur. Notons cet ensemble de paquets E , comme sur la Fig. 4.3. Comme cet ensemble dépend directement des décalages γ_1 et γ_3 , la recherche du pire scénario revient à rechercher les valeurs correspondantes de ces décalages. Le lemme 4.1, sans identifier complètement ce scénario, donne une bonne piste pour le rechercher.

Lemme 4.1

Soit E l'ensemble des paquets parallèles arrivant avant f_2^2 et soient f_3^1 et f_3^3 les dates d'entrée des deux paquets parallèles arrivant immédiatement avant f_2^2 (Fig. 4.3). Les dates d'entrée de ces deux paquets sont respectivement notées $(t^* - \delta_1)$ et $(t^* - \delta_3)$.

Si l'ensemble E correspond au pire scénario, alors le délai maximal D est subi quand les deux paquets parallèles arrivent quasi simultanément que f_2^2 (avant t^* tout de même), soit à l'instant $t^* - 0^+$.

Preuve (raisonnement par l'absurde) :

Supposons que le pire délai est subi alors que les deux paquets arrivent à des instants $(t^* - \delta_1)$ et $(t^* - \delta_3)$ où $\delta_1 > 0$ et $\delta_3 > 0$ (donc pas de quasi simultanéité). Notons ce délai D' .

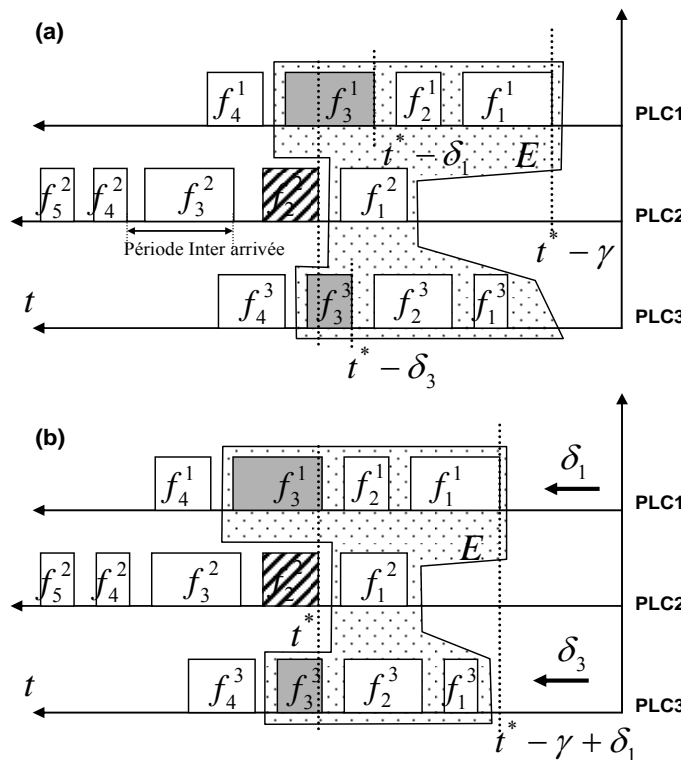


Fig. 4.4 : Analyse d'interférence dans le pire cas

Retardons maintenant les dates d'arrivée des deux rafales de PLC1 et PLC3 avec les décalages δ_1 et δ_3 respectivement, tout en gardant le même ensemble E (comme sur la Fig. 4.4b). Ceci est possible puisque les dates d'envoi depuis les différents contrôleurs sont indépendantes.

Naturellement, puisque l'ensemble des paquets entrant avant f_2^2 reste le même (c'est E) mais que celui-ci est retardé, la date de fin d'expédition de tous ses paquets vers les ports de sortie est elle aussi retardée. Or, la date d'arrivée du paquet f_2^2 n'a pas changé. Par conséquent, la date d'expédition de f_2^2 est forcément retardée (ou ne bouge pas dans le meilleur des cas). Au final le délai d'attente de f_2^2 est forcément supérieur à D' . Ceci est évidemment absurde puisque D' est supposé être le délai maximal depuis le début. \square

Le lemme précédent nous donne une importante indication sur le pire scénario concernant le délai subi par f_2^2 : *le délai maximal survient quand deux paquets parallèles arrivent quasi simultanément que le paquet f_2^2* . Deux paquets, mais lesquels ?

Pour résoudre complètement le problème, il reste à identifier les deux paquets parallèles. Une méthode simpliste consisterait à considérer toutes les combinaisons possibles entre un paquet de PLC1 f_i^1 et un autre de PLC3 f_j^1 . Le scénario de chaque combinaison est déroulé et le délai correspondant est calculé. Le délai maximal de tous les délais trouvés est alors le pire délai recherché. L'inconvénient de cette approche est qu'elle nécessite dans un premier temps de rechercher individuellement les dates d'arrivée de tous les paquets d'une rafale puis de calculer les dates d'entrée des autres paquets pour pouvoir dérouler le scénario. Cette procédure peut très vite devenir non viable pour des SCR de grande taille. Pour remédier à cela, nous proposons une autre approche qui consiste rechercher le pire délai indirectement en recherchant plutôt l'ensemble E en faisant varier les paramètres γ_1 et γ_3 suivant la condition du théorème suivant :

Théorème 4.1

Si un pas d'échantillonnage γ_e , plus petit que la plus petite période d'inter-arrivée des paquets, est utilisé pour faire varier les décalages γ_1 et γ_3 de façon combinatoire dans un domaine suffisamment large, alors :

- i) *Il est certain de trouver l'ensemble E correspondant au pire cas.*

- ii) Le couple des décalages γ_1 et γ_3 , correspondant à cet ensemble E , induit une sous-estimation du pire délai égale au plus à γ_e .

Preuve :

Rappelons d'abord que la période d'inter-arrivée est le temps entre les dates d'arrivée de deux paquets successifs (Fig. 4.4a).

i) Si le pas d'échantillonnage γ_e de γ_1 par exemple est inférieur à cette période d'inter-arrivée, alors nous sommes certains qu'en faisant varier γ_1 , la date d'arrivée du paquet f_2^2 (soit t^*) se retrouve à un moment donné insérée entre n'importe quel couple de deux paquets successif de la rafale de PLC1. Ceci implique donc que le sous ensemble de E contenant les paquets de cette rafale est garanti d'être trouvé. Le même raisonnement peut être fait concernant les paquets de la rafale de PLC3. Au final, l'ensemble E est assuré d'être trouvé.

ii) Nous sommes donc certains de trouver l'ensemble E mais il faut rappeler que cet ensemble à lui seul n'est pas suffisant puisque le même ensemble sur la Fig. 4.4 induit des délais différents. En faisant varier γ_1 et γ_3 , même avec un pas γ_e vérifiant la condition, le scénario de synchronisation expliqué dans le lemme n'est pas garanti. On se retrouve finalement avec un scénario du type Fig. 4.4a avec des décalages $\gamma_e \geq \delta_1 > 0$ et $\gamma_e \geq \delta_3 > 0$ entre les deux paquets parallèle et f_2^2 . Le décalage maximal de $(t^* - \delta_1)$ par rapport à t^* est dans le cas extrême égal γ_e . Il en est de même pour $(t^* - \delta_3)$. Autrement dit, dans ce cas extrême, les dates d'entrée de tous les paquets de E sont avancées de γ_e . En conséquence, la date d'expédition de f_2^2 est elle-même avancée de γ_e dans le meilleur des cas. Ceci revient à dire que le scénario avec des décalages δ_1 et δ_3 fait diminuer le pire délai de γ_e au plus. En d'autres mots, la sous estimation du pire délai est au maximum de γ_e . \square

Remarques 4.1 :

- Une surestimation du pire délai étant recherchée, il suffit d'ajouter γ_e au délai trouvé précédemment pour s'assurer de cela. La pire surestimation est dans ce cas égale à γ_e .
- La méthode d'analyse peut être appliquée dans le cas de génération de messages non périodiques, exactement de la même manière, à condition que la période minimale entre l'arrivée de deux messages sporadiques soit de loin supérieure aux délais de traversée. Dans ce cas en effet, un seul message de la rafale aperiodique est utilisé pour l'analyse, exactement comme dans le cas périodique.

- L'analyse précédente a été faite en considérant un seul commutateur mais peut être faite de manière similaire pour un nombre quelconques de N commutateurs traversés. La précision de l'évaluation est dans ce cas égale à $N \cdot \gamma_e$. Notons toutefois qu'il est possible d'améliorer cette précision progressivement en diminuant le pas d'échantillonnage γ_e une fois que l'ensemble E est trouvé. Cette stratégie a été adoptée sur un exemple d'application par la suite dans la section 4.4.
- Avec Ethernet le pas γ_e doit vérifier : $\gamma_e < \min_k [(72+12)/C_k]$ où 72 est la longueur minimale en octets d'un trame Ethernet (préambule inclus), 12 est la période d'inter trames en octets (96 bits), et C_k la vitesse de transmission du port k d'un commutateur du réseau. Pour des liaisons toutes configurées à 10 Mbps, le pas γ_e doit être inférieur à $67.2\mu s$.

4.3 Déroulement itératif d'un scénario de traversée d'un réseau Ethernet commuté

Le théorème précédent nous donne une méthode formelle pour évaluer le pire délai d'un paquet donné. Cela consiste à rechercher exhaustivement parmi un nombre fini de scénarii, celui qui conduit au délai maximal. Cependant, il faut calculer le délai correspondant à chaque scénario. Ceci implique donc la construction d'un modèle du réseau Ethernet. C'est l'objet de la section ci-après.

4.3.1 Modélisation d'un commutateur à l'aide de GETC

La difficulté principale de modélisation d'un commutateur est évidemment la présence de ressources partagées (ex : la fabrique de commutation). Avec ce handicap, il est en effet très difficile, voire impossible, de construire un modèle en GET et en déduire un modèle en équations (Max,+) linéaires. Toutefois, un autre modèle qui présente une structure tout aussi intéressante, est possible. Voyons cela sur un exemple simple.

L'exemple de la Fig. 4.5a représente un cas simple de communication à travers un commutateur. Pour simplifier l'analyse, nous supposons qu'un émetteur Em1 émet des messages vers un récepteur Rec1 et un émetteur Em2 émet des messages vers Rec2. Nous supposons aussi que les messages émis par chaque émetteur sont de taille constante. Le temps nécessaire pour commuter les paquets émis par Em1 est noté τ_{c1} et le temps pour leur retransmission par le port de sortie est τ_{s1} . Un modèle à base de réseaux de Petri (RdP) de ce système est représenté sur la Fig. 4.5b.

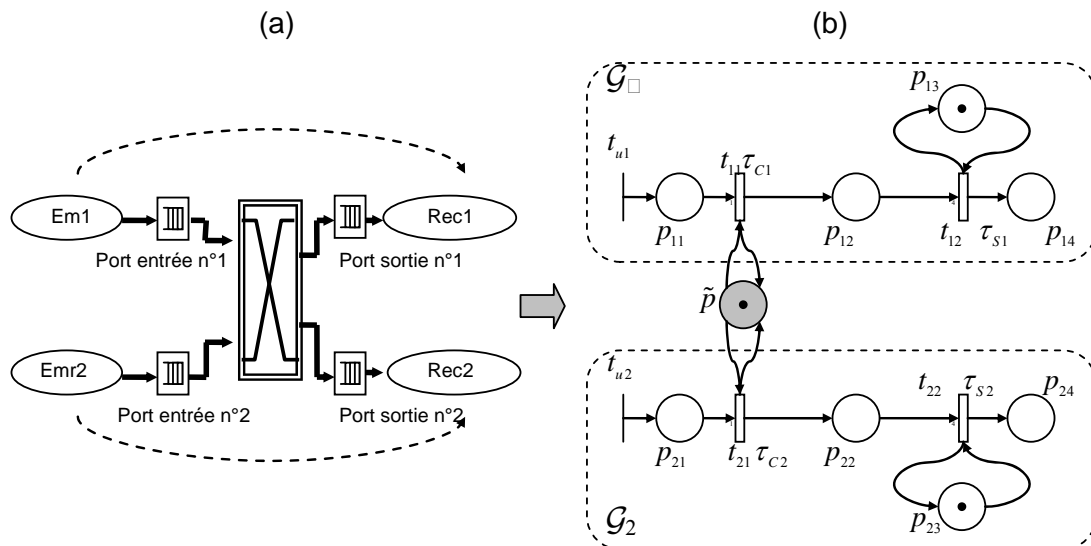


Fig. 4.5 : Modèle RdP d'un commutateur simple

Le modèle peut être décrit comme suit :

- Le franchissement de la transition d'entrée t_{u1} signifie l'entrée dans le commutateur d'un message émis par Em1. Celui-ci est matérialisé par un jeton dans la place p_{11} , représentant le port d'entrée n°1 du commutateur. Par la suite, ce jeton contribue au franchissement de la transition t_{11} . Ceci signifie la commutation du paquet vers le port de sortie n°1, représenté par la place p_{12} . Cependant, le franchissement de t_{11} est conditionné par la présence d'un jeton dans la place \tilde{p} . Ceci signifie naturellement la condition de disponibilité de la fabrique de commutation. Celle-ci est en effet une ressource partagée qui peut soit commuter des paquets du premier port d'entrée ou des paquets du second port, mais pas les deux à la fois.

Ensuite, le jeton de p_{12} contribue au franchissement de la transition t_{12} , signifiant ainsi la sortie du message vers le récepteur Rec1 (place p_{14}). Notons que ce franchissement est aussi conditionné par la présence d'un jeton dans la place p_{13} . Ceci représente simplement le fait que la transmission d'un nouveau message vers le récepteur ne commence que si la tête de file de la queue FIFO du port de sortie est vide. Autrement dit, à partir du moment où tous les messages précédents ont quitté le port.

Remarquons que le réseau de Petri de la Fig. 4.5b n'est pas un graphe d'événements temporisé car il contient un conflit structurel. La place \tilde{p} , représentant une ressource partagée, contient plus d'une transition en aval (et en amont d'ailleurs). Une modélisation du fonctionnement du système en utilisant la forme d'état $(\max,+)$ linéaire usuelle n'est donc pas possible.

Etat de l'art de l'algèbre (max,+) et les conflits

De nombreux travaux, abordant les réseaux de Petri contenant des conflits (ressources partagées), ont été entrepris pour tenter de les modéliser dans l'algèbre (Max,+).

Les auteurs Hillion *et al.* (1989) ont abordé le problème des systèmes répétitifs dont les ressources partagées sont allouées périodiquement. Ils ont réussi à transformer le RdP représentant un système avec des ressources partagées en un GET. Ce dernier étant facile à modéliser en équations (Max,+). Dans (Gaubert *et al.*, 1999), une approche basée sur la théorie des *tas de pièces* et des automates (Max,+) a été proposée pour modéliser des RdP contenant des conflits mais uniquement saufs. Une étude assez complète des RdP à choix libre a été développée dans (Baccelli *et al.*, 1992b). Le cas des systèmes pouvant commuter entre plusieurs modes de fonctionnement, ont été étudiés dans (Van Den Boom *et al.*, 2006). La modélisation est réalisée à l'aide de systèmes (Max,+) linéaires à commutation. Nous pouvons également citer les travaux de (Correia *et al.*, 2009) où des systèmes avec ressources partagées invariantes ont été abordés à l'aide d'équations (Max,+) linéaires sans tenir compte des conflits. Cependant, ces équations sont couplées avec une contrainte (inéquation représentant l'invariance des ressources) pour ne représenter que les comportements admissibles du système. Une autre étude introduisant le concept de transition virtuelle a été présentée dans (Naït *et al.*, 2006) pour modéliser en équations (Max,+) un système de transport. Plus récemment, (Boutin *et al.*, 2009) ont utilisé les diodes d'intervalles pour représenter les comportements extrêmes (majoration et minoration) de systèmes de production afin de calculer des bornes de leur taux de production.

Nous voyons effectivement que de nombreux travaux ont abordé le problème de modélisation des conflits. Cependant, des contraintes ou hypothèses souvent fortes ont été utilisées. Si, nous revenons à l'exemple simple du commutateur de la Fig. 4.5b, nous remarquons que le modèle RdP n'est pas cyclique, n'est pas sauf (ex : la place p_{11} peut contenir plusieurs jetons), n'est pas à choix libre (ex : la transition t_{11} contient plus d'une place en amont), ...

Néanmoins, nous pouvons noter que ce réseau de Petri présente une structure bien particulière. Il est constitué de deux GET, partageant la place \tilde{p} . C'est ce que nous appelons un réseau de *Graphes d'Événements Temporisés en Conflit* ou GETC. Le prochain paragraphe est consacré à la présentation d'une nouvelle approche de leur modélisation basée sur l'algèbre (Max,+).

4.3.2 Représentation d'état (max, +) des GETC

Nous reprenons l'exemple précédent du commutateur pour montrer la démarche à suivre pour aboutir à un modèle (Max,+) des GETC. Rappelons d'abord le fonctionnement des GET sans tenir compte de la place de conflit. Prenons l'exemple du premier GET \mathcal{G}_1 (Fig. 4.6).

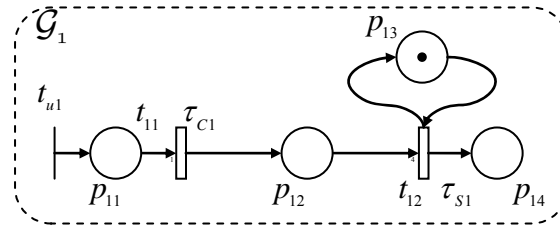


Fig. 4.6 : Le GET \mathcal{G}_1 sans la place de conflit

En associant un dateur x_{ij} à la transition t_{ij} de la Fig. 4.6, nous pouvons écrire les équations :

$$\begin{cases} x_{11}(k_1) = \tau_{c1} \otimes u_1(k_1) \\ x_{12}(k_1) = \tau_{s1} \otimes [x_{11}(k_1) \oplus x_{12}(k_1 - 1)] \end{cases} \quad (4.1)$$

Ces équations peuvent se mettre sous la forme linéaire standard suivante :

$$X_1(k_1) = A_1 \cdot X_1(k_1 - 1) \oplus B_1 \cdot u_1(k_1) \quad (4.2)$$

où $X_1(k_1) = (x_{11}(k_1) \quad x_{12}(k_1))^t$, $A_1 = \begin{pmatrix} \varepsilon & \varepsilon \\ \varepsilon & \tau_{s1} \end{pmatrix}$, et $B_1 = \begin{pmatrix} \tau_{c1} \\ \tau_{s1} \otimes \tau_{c1} \end{pmatrix}$.

Prenons maintenant en compte la place de conflit et examinons en les conséquences.

En introduisant la place \tilde{p} , seul le franchissement de la transition t_{11} est impacté puisqu'il est conditionné par la présence d'un jeton dans cette place. Supposons que quand la fabrique de commutation est disponible, après avoir été utilisée k fois pour la commutation des paquets des deux ports d'entrée, elle est effectivement utilisée pour la commutation d'un paquet émis par Em1. Notons cette date de disponibilité pour la $k^{\text{ème}}$ fois par $\tilde{\psi}(k)$. Si en plus nous savons qu'à ce moment là, $(k_1 - 1)$ paquets émis depuis Em1 ont déjà été commutés, nous pouvons écrire la date de franchissement de t_{11} pour la $k_1^{\text{ème}}$ fois comme suit :

$$x_{11}(k_1) = \tau_{c1} \otimes [u_1(k_1) \oplus \tilde{\psi}(k)] \quad (4.3)$$

La date de franchissement de t_{12} , ne dépendant pas directement de la place de conflit, reste quant à elle inchangée :

$$x_{12}(k_1) = \tau_{s1} \otimes [x_{11}(k_1) \oplus x_{12}(k_1 - 1)] \quad (4.4)$$

En regroupant les deux équations (4.3) et (4.4), nous aboutissons à la forme matricielle :

$$X_1(k_1) = \begin{pmatrix} \varepsilon & \varepsilon \\ \varepsilon & \tau_{s1} \end{pmatrix} \cdot X_1(k_1 - 1) \oplus \begin{pmatrix} \tau_{c1} \\ \tau_{s1} + \tau_{c1} \end{pmatrix} \cdot u_1(k_1) \oplus \begin{pmatrix} \tau_{c1} \\ \tau_{s1} + \tau_{c1} \end{pmatrix} \cdot \tilde{\psi}(k) \quad (4.5)$$

Ce système est donc de la forme :

$$X_1(k_1) = A_1 \cdot X_1(k_1 - 1) \oplus B_1 \cdot u_1(k_1) \oplus F_1 \cdot \tilde{\psi}(k) \quad (4.6)$$

où $F_1 = (e \ \tau_{s1})^t$. Les matrices A_1 et B_1 sont exactement les mêmes que celles utilisées en l'absence de la place de conflit. Elles caractérisent le GET \mathcal{G}_1 .

Par ailleurs, nous savons que le $k^{\text{ème}}$ jeton de \tilde{p} , ayant contribué au franchissement de la transition t_{11} pour la $k^{\text{ème}}$ fois, devient disponible pour la $(k+1)^{\text{ème}}$ fois juste après la fin de ce franchissement. Nous pouvons alors écrire :

$$\tilde{\psi}(k+1) = e \otimes x_{11}(k_1), \quad (4.7)$$

ou encore sous la forme matricielle :

$$\tilde{\psi}(k+1) = G_1 \otimes X_1(k_1)$$

où $G_1 = (e \ \varepsilon)$.

Pour résumer, si le $k^{\text{ème}}$ jeton de \tilde{p} est consommé par le GET \mathcal{G}_1 pour la $k_1^{\text{ème}}$ fois, les équations (Max,+) récurrentes suivantes sont vérifiées :

$$\begin{cases} X_1(k_1) = A_1 \cdot X_1(k_1 - 1) \oplus B_1 \cdot u_1(k_1) \oplus F_1 \cdot \tilde{\psi}(k) \\ \tilde{\psi}(k+1) = G_1 \otimes X_1(k_1) \end{cases} \quad (4.8)$$

Les équations relatives au GET \mathcal{G}_2 sont écrites de la manière similaire.

Les équations (Max,+) récurrentes dans le cas général sont écrites comme suit :

$$\begin{cases} X_i(k_i) = A_i \cdot X_i(k_i - 1) \oplus B_i \cdot u_i(k_i) \oplus F_i \cdot \tilde{\psi}(k) \\ \tilde{\psi}(k+1) = G_i \otimes X_i(k_i) \end{cases} \quad (4.9)$$

où les matrices A_i et B_i sont les matrices caractérisant le GET \mathcal{G}_i en l'absence de conflit.

Finalement, seules les nouvelles matrices F_i et G_i doivent être calculées pour identifier complètement le comportement du GETC. Il faut noter aussi que l'invariance de la ressource partagée implique que : $k = k_1 + k_2 - 1$.

En conséquence, si nous connaissons *a priori* la séquence d'allocation (politique d'ordonnancement) de la ressource partagée, quelle qu'elle soit, nous pouvons, grâce aux équations (4.9), trouver l'évolution du GETC.

Aussi, nous pouvons adopter des politiques dynamiques, connues a priori, mais dont la séquence d'allocation de la ressource est dynamique. C'est le cas de la politique FIFO (premier arrivé premier servi) que nous trouvons par exemple dans les commutateurs.

La question qui se pose alors est : comment intégrer cette politique avec les équations (4.9) ?

Le mode FIFO dans le commutateur signifie que le premier port d'entrée qui demande la commutation est le premier à être servi. Pour prendre cela en compte, nous introduisons la notion de *date de demande d'accès* à la ressource partagée. Désignons par $\eta_i(k_i)$ la date de demande d'accès à la ressource partagée \tilde{p} pour la $k_i^{\text{ème}}$ fois du GET \mathcal{G}_i . Ainsi, à un moment donné, le GET dont la date de demande d'accès est la minimale (qui demande en premier), obtient l'accès à la ressource. Voyons cela sur l'exemple précédent.

Pour le premier GET, la date $\eta_1(k_1)$ peut être formulée par la date d'entrée du $k_1^{\text{ème}}$ jeton dans la place p_{11} , soit la date de quasi-validation de la transition t_{11} (toutes les places en amont, en ne tenant pas compte de \tilde{p} , ont au moins un jeton disponible). L'expression de $\eta_1(k_1)$ est donc exactement la même que la date de franchissement de t_{11} pour la $k_1^{\text{ème}}$ fois mais sans tenir compte de la place de conflit. Elle s'écrit donc :

$$\eta_1(k_1) = \tau_{c1} \otimes u_1(k_1) \quad (4.10)$$

Cette date peut se réécrire en utilisant les matrices caractéristiques de \mathcal{G}_1 comme suit :

$$\eta_1(k_1) = A_1(1,:) \cdot X_1(k_1 - 1) \oplus B_1(1,:) \cdot u_1(k_1)$$

où $A_1(1,:)$ et $B_1(1,:)$ sont les première lignes des matrices A_1 et B_1 respectivement.

De la même manière, nous écrivons la date $\eta_2(k_2)$:

$$\eta_2(k_2) = A_2(1,:) \cdot X_2(k_2 - 1) \oplus B_2(1,:) \cdot u_2(k_2) \quad (4.11)$$

Finalement, en combinant les dates de demande d'accès $\eta_i(k_i)$ et les équations (4.9), nous pouvons exprimer l'évolution du GETC en mode FIFO comme suit : après k utilisation de la ressource partagée, $(k_1 - 1)$ fois par \mathcal{G}_1 et $(k_2 - 1)$ fois par \mathcal{G}_2 , l'égalité

$\eta_i(k_i) = \min\{\eta_1(k_1), \eta_2(k_2)\}$ implique que :

$$\begin{cases} X_i(k_i) = A_i \cdot X_i(k_i - 1) \oplus B_i \cdot u_i(k_i) \oplus F_i \cdot \tilde{\psi}(k) \\ \tilde{\psi}(k+1) = G_i \otimes X_i(k_i) \end{cases} \quad (4.12)$$

- Remarques 4.2 :

En cas d'égalité de deux ou plusieurs dates de demande d'accès, l'allocation de la ressource est faite au GET prioritaire si cela est défini a priori. Si ce n'est pas le cas, il convient de désavantager un GET par rapport aux autres pour privilégier un cas pessimiste. C'est le cas par exemple du paquet dont le délai de bout en bout est recherché. Celui-ci doit être désavantagé s'il demande l'accès à la fabrique de commutation en même temps qu'un autre paquet.

Quelques résultats génériques, au delà des réseaux de communication ...

Dans l'exemple précédent, nous avons considéré un GETC avec une seule ressource partagée. L'analyse peut être aisément généralisée pour un système avec plusieurs ressources.

Soit un GETC avec N GET et un ensemble $\tilde{R} = \{\tilde{p}_1, \tilde{p}_2, \dots, \tilde{p}_M\}$ de M ressources. Chaque GET \mathcal{G}_i utilise un sous ensemble de ressources $\tilde{R}_i \subseteq \tilde{R}$. Nous associons à chaque ressource \tilde{p}_j , la date de disponibilité de son jeton pour la $l_j^{\text{ème}}$ fois par $\psi_j(l_j)$.

Les équations d'état de ce GETC peuvent être exprimées par les équations (max,+) récurrentes suivantes (voir (Addad *et al.*, 2010a) pour plus de détails) :

$$\begin{cases} X_i(k_i) = A_i \cdot X_i(k_i - 1) \oplus B_i \cdot u_i(k_i) \oplus \bigoplus_{\tilde{p}_j \in \tilde{R}_i} F_{ij} \cdot \tilde{\psi}_j(l_j) \\ \tilde{\psi}_j(l_j + 1) = G_{ij} \otimes X_i(k_i) \quad \text{pour tout } j / \tilde{p}_j \in \tilde{R}_i \end{cases} \quad (4.12)$$

où les matrices A_i et B_i sont les matrices caractéristiques du GET \mathcal{G}_i sans prise en compte des conflits. Les matrices colonne F_{ij} et matrices ligne G_{ij} sont les seules à calculer.

- Remarque 4.3 :

- Si un GETC remplit un certain nombre de conditions (voir *Théorème 4.1* dans (Addad *et al.*, 2010a)) et évolue suivant une séquence arbitraire $\sigma = \mathcal{G}_i \mathcal{G}_i \dots \mathcal{G}_{i(p)}$ (les ressources sont attribuées pour les GET suivant l'ordre de cette séquence), une représentation (Max,+) linéaire à temps variant des équations (4.12) est possible. Elle est de la forme :

$$\bar{X}(k) = \bar{A}(k) \cdot \bar{X}(k-1) \oplus \bar{B}(k) \cdot \bar{U}(k) \quad (4.13)$$

où $\bar{X}(k) = (X_1(k_1-1) \quad X_2(k_2-1) \quad \dots \quad X_i(k_i) \quad \dots \quad X_N(k_N-1))^t$,

$\bar{U}(k-1) = (U_1(k_1-1) \quad U_2(k_2-1) \quad \dots \quad U_i(k_i-1) \quad \dots \quad U_N(k_N-1))^t$.

Les matrices $\bar{A}(k)$ et $\bar{B}(k)$ sont calculées en fonction des matrices caractéristiques des équations (4.12) et en fonction de la séquence σ .

- Si la séquence d'évolution d'un GETC est cyclique $\sigma = \sigma_0 \sigma_0 \dots$ où $\sigma_0 = \mathcal{G}_i \mathcal{G}_i \dots \mathcal{G}_{i(T)}$, la représentation d'état à temps variant (4.13) admet une forme d'état standard à temps invariant. Elle est retrouvée en utilisant la notion de monodromie, connue par ailleurs dans les systèmes linéaires classiques périodiques et appliquée de manière similaire pour les systèmes (max,+) linéaires (Lahaye *et al.*, 2004). Cette forme est donnée dans (Addad *al.*, 2010a) par :

$$\bar{\bar{X}}(l) = \bar{\bar{A}} \otimes \bar{\bar{X}}(l-1) \oplus \bar{\bar{B}} \otimes \bar{\bar{U}}(l) \quad (4.14)$$

$$\text{où } \bar{\bar{A}} = \Phi(1+T, 1), \quad \bar{\bar{B}}(p, q) = \begin{cases} \Phi(1+T, 1+q) \otimes \bar{B}(1+q) & \text{si } (p = q) \\ I_\varepsilon & \text{sinon} \end{cases},$$

$$\Phi(i, i-q) = \begin{cases} \bar{A}(i-1) \otimes \bar{A}(i-2) \otimes \dots \otimes \bar{A}(i-q) & \text{si } q \geq 1 \\ I_d & \text{si } q = 0 \end{cases}.$$

La matrice $\bar{\bar{A}}$ est appelée *matrice de monodromie* et Φ est appelée *matrice de transition*.

Rappelons que les deux formes (4.13), (4.14) sont très utiles et peuvent être utilisées pour la résolution de nombreux problèmes pratiques : évaluation de performance, optimisation, synthèse de commande, simulation, ... des SED.

4.3.3 Modèle en équations (Max,+) récurrentes d'un réseau Ethernet

L'exemple de communication à travers un commutateur que nous avons présenté précédemment était très simplifié. Regardons ce qui se passe dans le cas où les deux émetteurs Em1 et Em2 envoient tous les deux des messages de taille différente vers les deux récepteurs. Un modèle RdP de ce système est représenté sur la Fig. 4.6.

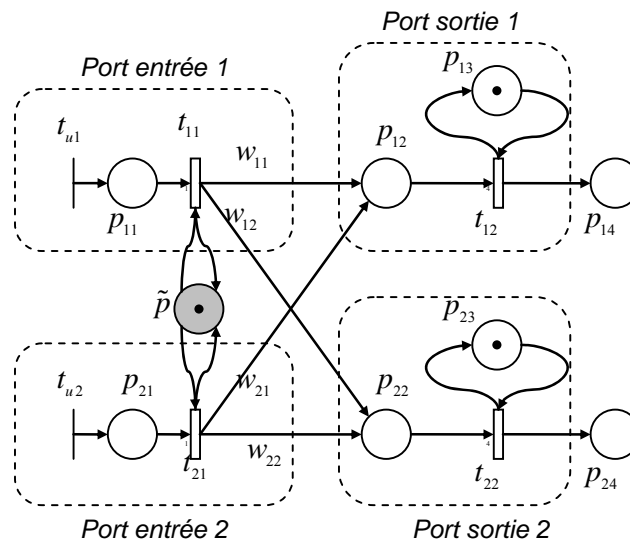


Fig. 4.6 : Modèle plus élaboré d'un commutateur

Le franchissement de la transition t_{i1} ($i \in \{1, 2\}$) modélise la commutation d'un paquet. Comme la taille du paquet peut varier, la temporisation associée à t_{i1} également. Elle est donc fonction de la longueur L_k du $k^{\text{ème}}$ paquet commuté. Elle est égale à $\tau_{i1}(k) = L_k \cdot C$ où C est la vitesse de la fabrique de commutation. De même, la temporisation associée à une transition du port de sortie i est $\tau_{i2}(k) = L_k \cdot C_i$ où C_i est la vitesse de transmission du port de sortie i .

Par ailleurs, lors du franchissement de t_{i1} , un jeton est mis dans p_{12} si le paquet est commuté vers le port de sortie n°1. Un jeton est mis dans la place p_{22} dans le cas contraire. Pour prendre cela en compte et bien router les jetons, nous associons à chaque jeton une propriété sur sa destination. Cette propriété est exploitée dans le modèle (Max,+), à l'instar des RdP colorés, à l'aide de fonctions notées w_{ij} , sur les arcs du RdP (Fig. 4.6). Nous associons à chaque jeton le couple $(L_k, Dest_k)$ où L_k la longueur du paquet qu'il représente et $Dest_k$ sa destination. Ainsi, nous définissons le nombre de jetons transmis par un arc en aval d'une transition d'un port d'entrée i comme suit :

$$w_{ij}(Dest_k) = \begin{cases} 1 & \text{si } Dest \equiv j \\ 0 & \text{sinon} \end{cases} \quad (4.15)$$

De manière similaire que dans l'exemple précédent, à la $k^{\text{ème}}$ étape, le port i dont la demande d'accès $\eta_i(k_i)$ est minimale est choisi pour expédier son paquet $(L_k, Dest_k)$ en attente. La date de routage est :

$$x_{i1}(k_i) = \tau_{i1}(k) \otimes [u_i(k_i) \oplus \tilde{\psi}(k)] \quad (4.16)$$

Ou encore en remplaçant la temporisation $\tau_{i1}(k)$ par sa valeur :

$$x_{i1}(k_i) = L_k \cdot C \otimes [u_i(k_i) \oplus \tilde{\psi}(k)] \quad (4.17)$$

La nouvelle date de disponibilité de la fabrique de commutation est :

$$\tilde{\psi}(k+1) = e \otimes x_{i1}(k_i) \quad (4.18)$$

Supposons que le paquet est commuté vers le port de sortie j qui a reçu k'_j paquets, des deux ports d'entrée confondus, depuis le début. La date de sortie de ce paquet est alors donnée par la date de franchissement de t_{i2} :

$$x_{i2}(k'_j) = \tau_{i2}(k) \otimes [x_{i1}(k_i) \oplus x_{i2}(k'_j - 1)] \quad (4.19)$$

Ou encore :

$$x_{i2}(k'_j) = L_k \cdot C_j \otimes [x_{i1}(k_i) \oplus x_{i2}(k'_j - 1)] \quad (4.20)$$

- Remarque 4.4 :

Faisons remarquer que la place p_{j4} en sortie peut représenter, dans le cas d'un réseau avec plusieurs commutateurs, le port d'entrée d'un autre commutateur en aval de celui de la Fig. 4.6. La transition t_{i2} est dans ce cas une transition d'entrée (du commutateur en aval) qu'on peut noter t'_{ij} . Le dateur associé est :

$$u'_j(k'_j) = L_k \cdot C_j \otimes [x_{i1}(k_i) \oplus x_{i2}(k'_j - 1)] \quad (4.21)$$

L'écriture de l'équation (4.21) signifie une introduction éventuelle d'une nouvelle date de demande d'accès $\eta'_j(k'_j)$ à la fabrique de commutation du commutateur en aval. Cela signifie que l'ensemble des dates de demande d'accès est mis à jour. Nous pouvons alors procéder à la commutation du prochain paquet en recherchant la date de demande d'accès minimale dans le nouvel ensemble. Une fois celle-ci trouvée, nous réutilisons exactement les mêmes équations (4.16) à (4.21), ces dernières étant génériques.

Finalement, nous avons abouti à des équations récurrentes qui permettent de dérouler itérativement le fonctionnement du réseau. Ces équations sont utilisées itérativement jusqu'à ce que tous les paquets soient commutés. De là, les délais de traversée peuvent être calculés en utilisant les dates de franchissement des transitions d'entrée et de sortie. Par exemple le $k_i^{\text{ème}}$ paquet du port d'entrée i , expédié à la $k^{\text{ème}}$ commutation vers le port de sortie j qui a reçu k'_j paquets à ce moment là, subit un délai donné par :

$$D_i(k_i) = x_{i2}(k'_j) - u_i(k_i) \quad (4.22)$$

- Dans le cas de notre SCR, nous avons également besoin d'intégrer des modèles des modules E/S déportés. Le modèle d'un module E/S connecté au $j^{\text{ème}}$ port du commutateur est représenté sur la Fig. 4.7.

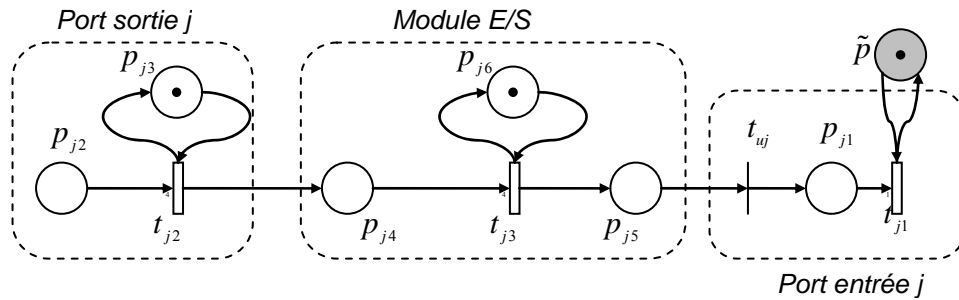


Fig. 4.7 : Modèle d'un module E/S connecté au $j^{\text{ème}}$ port d'un commutateur

L'arrivée des requêtes dans le module E/S est matérialisée par des jetons dans la place p_{j4} . La contribution de chaque jeton au franchissement de la transition t_{j3} est conditionnée par la présence d'un jeton dans p_{j6} (disponibilité du processeur du module qui ne peut traiter qu'une requête à la fois). Une fois qu'une requête est traitée, durant un temps τ_{j3} , une réponse est renvoyée vers le port d'entrée du commutateur avec le franchissement de t_{uj} (Fig. 4.7). Faisons remarquer que ceci revient à dire que le commutateur est en aval de lui-même, au module E/S près.

En intégrant le module E/S, l'équation (4.21) devient :

$$u'_j(k'_j) = e \otimes x_{j3}(k'_j) = \tau_{j3} \otimes [x_{j2}(k_i) \oplus x_{j3}(k'_j - 1)] \quad (4.22)$$

Ou encore en remplaçant $x_{j2}(k_i)$ par son expression (4.19) :

$$u'_j(k'_j) = \tau_{j3} \otimes L_k \cdot C_j \otimes [x_{i1}(k_i) \oplus x_{i2}(k'_j - 1)] \oplus \tau_{j3} \otimes x_{j3}(k'_j - 1) \quad (4.23)$$

Finalement, chaque scénario étant complètement identifié par les dates d'émission des paquets, soit des dates de franchissement de transitions d'entrée t_u , nous pouvons dérouler tout scénario en utilisant les équations (Max,+) récurrentes (4.16) à (4.23). Le délai correspondant est alors aisément calculé en utilisant les dates adéquates.

Cette méthode de calcul se basant sur des modèles événementiels, donc « event-driven », est naturellement plus rapide que l'utilisation de simulateurs classiques qui sont le plus souvent time-driven. En effet, au lieu de tester les transitions valides à chaque pas d'échantillonnage, qui est généralement trop petit pour éviter des pertes d'informations, seules les dates utiles des équations (4.16) à (4.23) sont utilisées pour dérouler l'évolution du système.

- Remarque 4.5 :

Les équations récurrentes (4.16) à (4.23) ont également été obtenues à l'aide d'un autre formalisme, introduit dans (Addad *et al.*, 2010e) appelé *files d'attente virtuelles*. Un algorithme regroupant des équations similaires y est donné. Ce formalisme permet une modélisation assez intuitive des systèmes à files d'attente comme les réseaux de communication. Il utilise aussi la notion de date de disponibilité ψ associée aux ressources partagées.

4.4 Calcul de la borne maximale d'un délai de bout-en-bout

4.4.1 Méthode de calcul combinatoire

Rappelons que d'après le *Théorème 4.1*, si un pas d'échantillonnage γ_e , inférieur à la période minimale d'inter-arrivée des paquets, est utilisé pour faire varier les décalages d'émission entre les contrôleurs, une recherche exhaustive du délai maximal est assurée. Une méthode combinatoire de calcul du délai est possible en faisant varier chaque décalage dans le domaine $[-T_{Dom}, +T_{Dom}]$, T_{Dom} pris suffisamment grand pour s'assurer de balayer le pire cas. Pour chaque combinaison de décalages (scénario), nous déroulons les équations (4.16) à (4.23) pour calculer le délai de bout-en-bout correspondant. On répète la procédure jusqu'à balayer toutes les combinaisons, soit au total un nombre de *scenarii* égal à $(2 \cdot \lfloor T_{Dom} / \gamma_e \rfloor)^n$ où n est le nombre total d'émetteurs.

Nous avons appliqué cette méthode combinatoire sur un réseau Ethernet commuté déjà considéré dans (Georges *et al.*, 2005b) et représenté sur la Fig. 4.8.

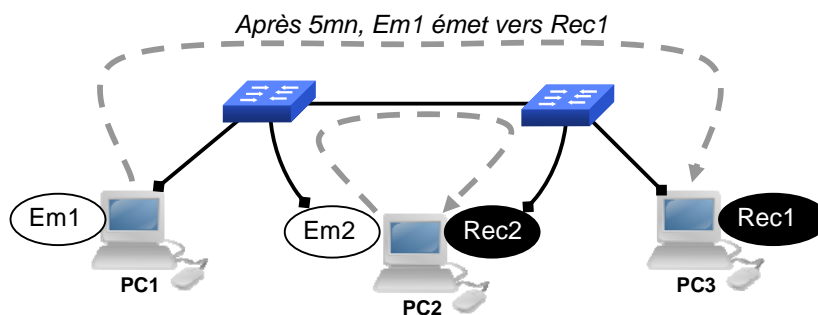


Fig. 4.8 : Exemple de réseau Ethernet commuté (Georges *et al.*, 2005b)

Ce système est composé de deux commutateurs, deux émetteurs Em1 et Em2 et deux récepteurs Rec1 et Rec2. La communication dure au total 10 mn. Elle est composée de deux phases : entre 0 et 5 mn, seul l'émetteur Em2 émet périodiquement des messages de 72 octets, toutes les secondes, vers le récepteur Rec2. Ensuite, entre 5 mn et 10 mn, le deuxième

émetteur Em1 envoie en parallèle, toutes les 10 ms, des messages de 1026 octets vers le récepteur Rec1.

Il s'agit de calculer le pire délai de bout-en-bout des messages envoyés de Em2 vers Rec2. Des mesures expérimentales réalisées par (Georges *et al.*, 2005b) ont montré un délai maximal de 312 μs durant la première phase puis 1119 μs pour la deuxième phase.

Le calcul réseau appliqué par (Georges *et al.*, 2005b) à ces deux scénarii donne une évaluation de 328 μs pour la première phase et 1173 μs pour la deuxième, soit une surestimation d'environ 5% dans les deux cas.

Nous avons appliqué la méthode combinatoire précédente et avons abouti à un délai de 315 μs pour la première phase et 1130 μs pour la deuxième, soit 1% de surestimation dans les deux cas. Au final, nous obtenons des résultats moins pessimistes qu'avec le calcul réseau tout en surestimant le pire délai observé expérimentalement.

Cette méthode combinatoire, comme énoncé dans le théorème, donne des résultats assez précis mais présente un inconvénient en temps de calcul lorsqu'il s'agit de calculer les pires délais dans des SCR de grande taille. Rappelons que le nombre de scénarii à considérer est de $(2 \cdot \lfloor T_{Dom} / \gamma_e \rfloor)^n$, soit une complexité exponentielle en fonction du nombre d'émetteurs. Pour remédier à ce problème, nous proposons une méthode alternative qui consiste à utiliser un algorithme génétique pour rechercher le pire des *scenarii*, chaque scénario étant noté suivant le délai correspondant, aisément calculé à l'aide des équations (4.16) à (4.23).

4.4.2 Méthode alternative : algorithmes génétiques

Les algorithmes génétiques (AG) appartiennent à la famille des algorithmes évolutionnistes, appliquant le principe de la sélection naturelle. Leur but est de résoudre en un temps raisonnable des problèmes d'optimisation difficilement abordables avec les approches classiques. Les AG ont été introduits pour la première fois par John Holland (1975) et ont eu un succès remarquable. Ils ont été appliqués dans pratiquement tous les domaines (Passino, 2005). Concernant les réseaux de communication, les AG ont été utilisés dans (Georges *et al.*, 2006), (Zhang *et al.*, 2007) et (Carro-Calvo *et al.*, 2010) pour optimiser le partitionnement d'un réseau Ethernet commuté afin de minimiser les délais de communication. Ils ont été utilisés dans (Lee *et al.*, 2004) pour aider les architectes réseau dans la sélection de certains paramètres (Timers) pour satisfaire d'une part des délais de bout-en-bout dans un réseau Profibus Token Passing et maximiser le transfert du trafic non temps réel d'autre part. Les AG ont servi également à l'optimisation de la qualité de service dans des réseaux mobiles ad hoc dans (Cheng *et al.*, 2010) et (Garcia-Nieto *et al.*, 2010). Enfin, ils ont été utilisés dans

(Abadeh *et al.*, 2007) pour améliorer la sécurité des réseaux informatiques en réduisant les fausses alarmes sur des activités intrusives.

Dans notre cas, il s'agit évidemment de déterminer le pire scénario (maximisation de délai) dans un SCR.

4.4.2.1 Principe de fonctionnement d'un algorithme génétique

Un AG est une méthode de recherche stochastique imitant le processus biologique de l'évolution pour trouver une solution optimale à un problème donné. Il consiste à créer une population d'individus (solutions potentielles) sur lesquels on effectue des opérations sur plusieurs générations jusqu'à trouver une solution satisfaisante.

Un AG est initialisé avec un ensemble de candidats (population de première génération) appelés *chromosomes*. Un chromosome consiste en un ensemble de paramètres variables appelés *gènes*. Chaque chromosome est noté en utilisant une *fonction objectif* (ou *fitness* en anglais) suivant sa capacité à résoudre le problème posé. Dans notre cas, la fonction objectif correspond au délai de bout-en-bout et un meilleur candidat est un scénario dont le délai correspondant est plus grand. Une fois que tous les chromosomes de la génération sont notés, seuls les plus forts survivent pour se reproduire et avoir des enfants, les autres meurent. Les enfants deviennent alors les parents de la prochaine génération. Les étapes précédentes sont répétées jusqu'à ce qu'une solution satisfaisante soit obtenue. Aussi, pour éviter de tomber dans un optimum local comme c'est le cas dans les méthodes classiques à gradient, un processus de *mutation* des enfants est introduit dans les AG.

Suivant le contexte et le problème d'optimisation posé, diverses approches existent pour aborder les différentes phases d'un AG. Concernant l'AG que nous avons utilisé dans notre problème, les étapes et leurs caractéristiques sont les suivantes (Fig. 4.9) :

- **Codage des chromosomes** : chaque chromosome étant un scénario dans notre cas, qui lui-même est complètement défini par les décalages des émissions des rafales, nous supposons que chaque gène est un décalage codifié avec un nombre réel (AG continu). Par conséquent, nous considérons que chaque gène g_k^i du chromosome i représente un décalage δ_i . Le $i^{\text{ème}}$ chromosome est alors noté $\Theta^i = \{g_1^i, g_2^i, \dots, g_n^i\}$ et sa fonction objectif est notée $J(\Theta^i)$. Bien entendu, $J(\Theta^i) = D_i$ où D_i est le délai de bout-en-bout recherché calculé en utilisant les équations (Max,+) (4.16) à (4.23).
- **Initialisation de la population** : dans certains problèmes, on peut avoir une idée de la situation de la solution et on peut dans ce cas initialiser la population des

chromosomes en fixant les paramètres de la sorte. Dans des cas plus complexes, l'initialisation se fait aléatoirement. Nous adoptons cette stratégie et générons aléatoirement les décalages de départ dans un domaine assez large $[-T_{Dom}, +T_{Dom}]$.

- **Sélection** : suivant le processus de la sélection naturelle, les meilleurs candidats dont les valeurs de la fonction objectif sont les plus grandes survivent et participent à la reproduction et les autres meurent. Beaucoup d'approches existent pour réaliser cette sélection. Un moyen efficace pour faire cela est d'utiliser ce qu'on appelle la *roue de la fortune*. Cette roue consiste en une roue classique sur laquelle chaque individu est représenté par une portion $P^i = J(\Theta^i) / \sum_{i=1,n} J(\Theta^i)$ proportionnelle à sa fonction objectif.

Pour chaque candidat de la génération, on lance la roue et l'individu dont la portion est pointée par le curseur, est sélectionné pour le croisement. Ce tirage au sort privilégie évidemment les meilleurs candidats.

- **Croisement** : les candidats sélectionnés dans l'étape précédente participent à la reproduction en subissant ce qu'on appelle le croisement. Diverses possibilités existent pour réaliser cette phase (Passino, 2005). L'une des plus utilisée est de prendre un couple de parents Θ^i et Θ^j puis de les croiser, avec une probabilité P_{Crs} , des couples de leurs gènes g_k^i , g_k^j pour former des enfants Θ^i , Θ^j . Les gènes de ces enfants sont formés comme suit :

$$\begin{cases} \underline{g}_k^i = \alpha \cdot g_k^i + (1 - \alpha) \cdot g_k^j \\ \underline{g}_k^j = \alpha \cdot g_k^j + (1 - \alpha) \cdot g_k^i \end{cases}$$

α étant un nombre aléatoire avec une distribution uniforme dans le domaine $[0,1]$.

Parmi les deux enfants obtenus après le croisement, nous gardons seulement le meilleur, qui devient un parent de la prochaine génération.

- **Mutation** : pour éviter une convergence rapide et le risque de rester bloqué dans un optimum local, nous forçons l'AG à explorer d'autres régions de l'espace en introduisant le mécanisme de mutation. Chaque gène \underline{g}_k^i du meilleur enfant, notons le Θ^{i*} , obtenu après le croisement, subit une mutation avec une probabilité P_{Mt} , comme suit :

$$\underline{g}_k^{i*} = (2\beta - 1) \cdot T_{Dom}$$

β étant un nombre aléatoire avec une distribution uniforme dans le domaine $[0,1]$.

Les étapes : sélection, croisement et mutation sont répétées jusqu'à ce qu'une solution, satisfaisant un critère d'arrêt soit trouvée (voir schéma de la Fig. 4.9). Par exemple, nous pouvons poser un critère consistant à arrêter l'AG après qu'un nombre maximal de générations soit atteint.

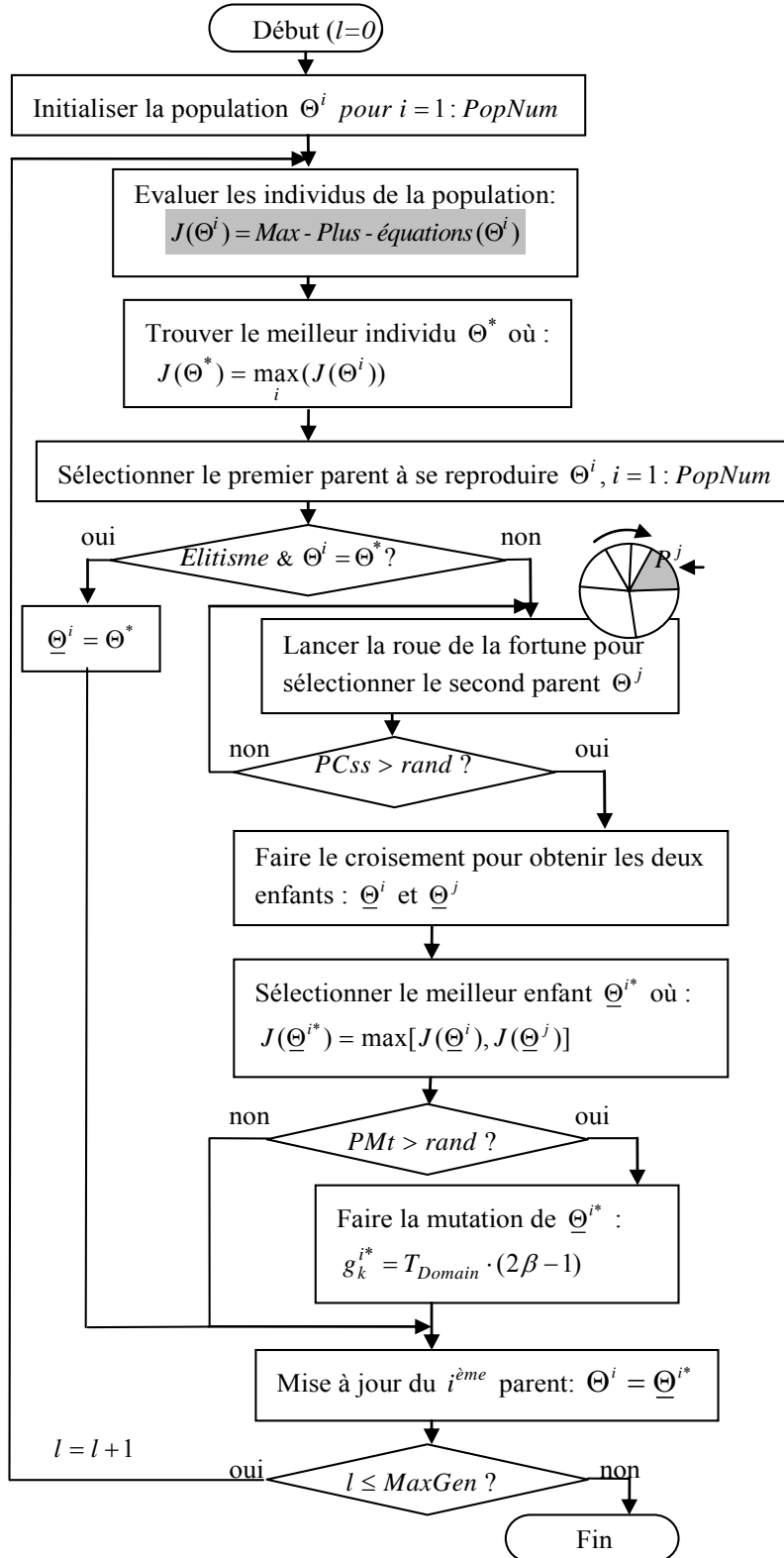


Fig. 4.9 : Algorithme de l'AG utilisé (*rand* est un nombre aléatoire appartenant au domaine [0,1])

- Remarque 4.6 :

Dans la résolution de problème d'optimisation à l'aide des AG, il est souvent considéré ce qu'on appelle *l'élitisme* (Fig. 4.9). Cela consiste à garder le meilleur candidat de chaque génération pour la génération suivante en participant uniquement dans le croisement et ne subissant pas de mutation. Cela a pour conséquence d'obtenir une fonction objectif monotone croissante (voir l'exemple de la section ci-après).

4.4.2.2 Exemple d'application des GA

Pour illustrer l'apport en temps de calcul de l'utilisation des AG par rapport à la méthode combinatoire pour la recherche du pire délai de bout-en-bout, nous considérons l'exemple du SCR de la Fig. 4.10. Ce SCR est composé de deux parties et fonctionne comme suit : la première partie *PART I* est dédiée à la commande. Les contrôleurs PLC1, PLC2, PLC3 scrutent respectivement les modules E/S : {MES12, MES13, MES14}, {MES14, MES15, MES16}, et {MES16, MES17} en utilisant des requêtes de taille minimale de 72octets. La deuxième partie *PART II* quant à elle sert pour la supervision de la première partie pour s'assurer de son bon fonctionnement. La station PC2 envoie périodiquement, toutes les 300ms, des requêtes à tous les modules déportés E/S. La station PC1 par contre ne fait que générer du trafic parallèle en envoyant des requêtes de 1008octets aux stations PC3 et PC4.

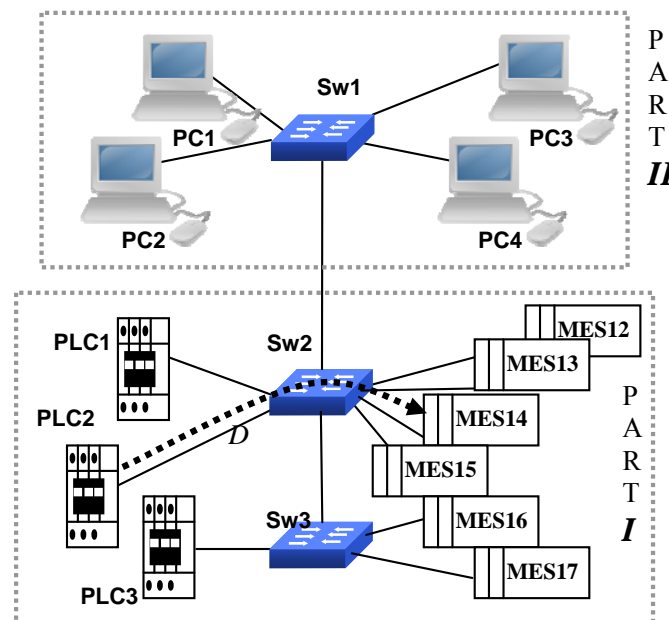


Fig. 4.10 : Exemple de SCR

L'objectif est de calculer le pire délai, que la requête générée par PLC2 et envoyée vers le MES14, met pour traverser le commutateur SW2. Nous avons choisi cette requête du fait que le module MES14 soit scruté par trois stations et donc le pire scénario pas évident à trouver. Pour évaluer le pire délai, nous considérons les deux méthodes : combinatoire et AG. Pour la méthode combinatoire, nous avons considéré deux pas de variation des décalages $\gamma_e = 20\mu s$ et $\gamma_e = 50\mu s$ pour illustrer l'impact sur le temps de calcul. Remarquons que ces valeurs vérifient les conditions du Théorème 4.1 puisque $\gamma_e < 67.2\mu s$. Nous avons aussi resserré progressivement les valeurs du pas γ_e jusqu'à atteindre $\gamma_e = 1\mu s$. Les résultats d'évaluation des délais ainsi que le temps de calcul sont donnés dans la Table 4.

Table 4. Résultats d'évaluation du pire délai bout en bout.

Méthode	étape	Pas γ_e (μs)	Pire délai trouvé en (μs)	Temps de calcul
Combinatoire I	Step1	20	319.96	~ 2h
	Step2	1	319.96	
Combinatoire II	Step1	50	283.52	~ 4min
	Step2	10	313.52	
	Step3	1	319.96	
AG	/		320.50	~ 29 s

Comme nous pouvons le constater, toutes les méthodes aboutissent quasiment au même pire délai de bout-en-bout. Notons toutefois que le temps de calcul est considérablement réduit et passe d'environ 2h à seulement 4mn en utilisant un pas initial de $\gamma_e = 50\mu s$ au lieu de $\gamma_e = 20\mu s$. Dans les deux cas, le délai obtenu est de $D = 319.96\mu s$ avec une précision de $1\mu s$. Le délai maximal réel est donc au pire égal à $(319.96 + 1)\mu s$, soit $D = 320.96\mu s$.

Nous avons appliqué la méthode des AG pour le calcul du même délai. Les propriétés de l'AG utilisé sont les suivantes :

- Le nombre d'individus d'une génération est : $PopNum = 20$
- Le nombre maximal de générations est : $MaxGen = 1000$
- La probabilité de croisement est : $P_{Crs} = 0.8$
- La probabilité de mutation est : $P_{Mt} = 0.08$

Le résultat d'évaluation à travers les générations en adoptant l'option d'élitisme est représenté sur la Fig. 4.11.

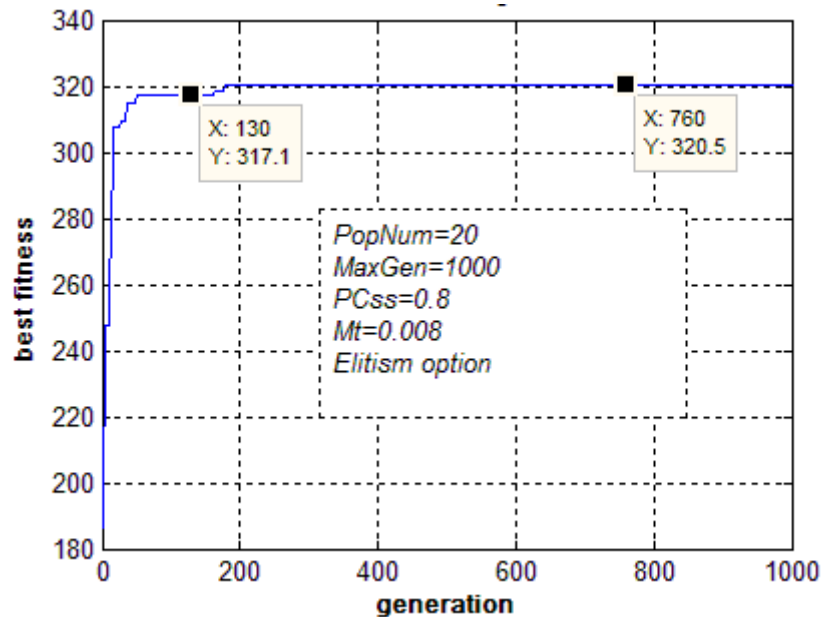


Fig. 4.11 : Résultat d'évolution de l'AG et du pire délai trouvé

Comme on peut le remarquer, le résultat final de l'évolution de l'AG sur 1000 générations est $D = 320.50 \mu s$. Ce résultat est supérieur au délai $D = 319.96 \mu s$ obtenu par la méthode combinatoire mais reste en dessous du pire délai réel prédit $D = 320.96 \mu s$. Le résultat est donc conforme aux prévisions. La fonction objectif de l'AG étant restée invariable à partir de la 200^{ème} génération jusqu'à la 1000^{ème}, nous pouvons accepter le délai obtenu comme étant effectivement le pire délai réel.

L'avantage de l'AG est évidemment dans le temps de calcul qui est réduit à seulement 29s.

En conclusion, l'AG s'avère plus efficace aussi bien sur la précision du calcul que sur le temps de calcul.

Conclusion du chapitre

Dans ce chapitre, nous avons complété le chapitre 3 où nous avons démontré qu'une évaluation complète et efficace du temps de réponse des SCR passe par une bonne évaluation des délais de bout-en-bout. Nous avons proposé une nouvelle approche de modélisation, basée sur des modèles en équations (Max,+), pour modéliser l'évolution d'un réseau Ethernet commuté suivant un scénario de départ donné. Ensuite, deux méthodes de recherche du pire scénario ont été proposées : une méthode combinatoire utilisable pour des SCR de taille moyenne et une méthode alternative à base des AG pour les SCR de grande taille.

Au final, nous disposons maintenant de tous les outils théoriques nécessaires pour accomplir l'évaluation du temps de réponse.

Néanmoins, ces résultats théoriques reposent sur de nombreux modèles et quelques hypothèses qu'il convient maintenant de valider par une comparaison avec des mesures expérimentales. Cette étape de validation est exposée dans le prochain chapitre.

Validation expérimentale

Sommaire

5.1 Présentation de la plateforme expérimentale	115
5.1.1 Architecture de commande considérée	115
5.1.2 Outils et procédure de mesure expérimentale du temps de réponse	116
5.2 Confrontation des résultats théoriques aux mesures sur un cas d'étude	118
5.2.1 Calcul des bornes du temps de réponse	118
5.2.2 Calcul de la fonction de distribution du temps de réponse	119
5.3 Validation expérimentale de la formule de la borne maximale	123
5.3.1 Impact de la période de scrutation sur la borne maximale	124
5.3.2 Impact de la charge du module E/S capteur sur la borne maximale	126
5.3.3 Impact de la charge du module E/S actionneur sur la borne maximale	127
5.3.4 Impact de la charge d'autres module E/S sur le la borne maximale	128

Dans ce dernier chapitre, nous exposons les résultats d'une campagne que nous avons conduite afin de valider les résultats théoriques présentés dans les chapitres précédents. Nous décrivons d'abord la plateforme expérimentale utilisée à cet effet puis la procédure adoptée pour la mesure du temps de réponse. Pour la confrontation des résultats théoriques aux mesures, nous considérons dans un premier temps un cas d'étude. Sur cette étude de cas, aussi bien les bornes que la distribution du temps de réponse sont évaluées. Dans un deuxième temps, nous nous penchons plus spécifiquement sur la borne maximale du temps de réponse, cette dernière étant la plus importante et difficile à évaluer. Pour ce faire, une série de mesures est réalisée, en ciblant un paramètre à la fois, de manière à analyser son impact spécifique sur la borne maximale. Ainsi, l'analyse de cet impact et sa confrontation aux prédictions théoriques nous permettront de vérifier la validité de nos modèles.

*~ It doesn't matter how beautiful your theory is,
it doesn't matter how smart you are,
if doesn't agree with experiment it's wrong. ~
- Richard Phillips Feynman*

5.1 Présentation de la plateforme expérimentale

5.1.1 Architecture de commande considérée

Pour réaliser des mesures de temps de réponse et les comparer aux évaluations analytiques présentées dans ce manuscrit, nous considérons le SCR de la Fig. 5.1. Il est composée de :

- Un contrôleur (PLC) TSX 57203 Premium (Schneider) dont le fonctionnement peut être configuré en mode cyclique ou en mode périodique.
- Deux PC pour générer du trafic parallèle à la boucle de commande dont le temps de réponse est recherché. Le PC1 génère du trafic temps réel du type Modbus TCP/IP en émulant un contrôleur grâce à un logiciel appelé Modbus Poll¹. PC2 quant à lui génère du flux non temps réel (du type UDP) grâce à un logiciel appelé Iperf² (plus de détails seront donnés à ce sujet par la suite).
- Un coupleur (la COM) Ethernet ETY 5102 (Schneider) relié au contrôleur. Il permet la scrutation périodique des modules déportés. Ce coupleur présente également l'avantage de contenir un serveur web qui permet d'accéder à diverses informations sur le contrôleur/coupleur et faire du diagnostic (paquets émis, paquets reçus, paquets perdus, modules E/S hors service, ...).
- Huit modules E/S déportés 170 ENT 11000 Momentum (Schneider). Chaque module possède 16 entrées et 16 sorties logiques.
- Trois commutateurs Ethernet 499 NES 18100 (Schneider) spécialement conçus pour les environnements industriels. Ils possèdent huit ports chacun et peuvent fonctionner en 10 Mbps ou en Fast Ethernet (100 Mbps).

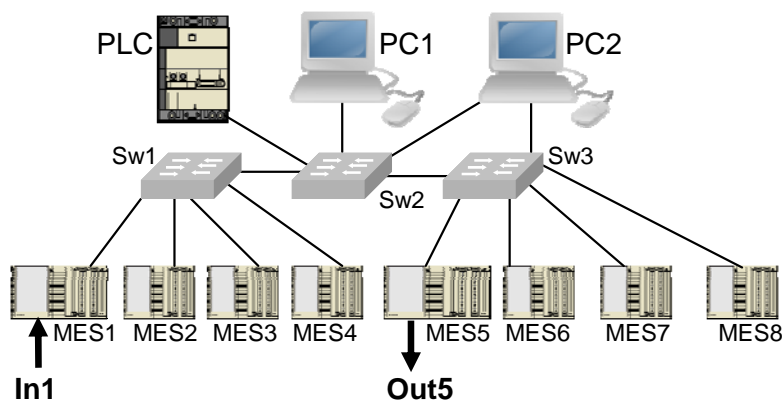


Fig. 5.1 : Architecture de commande en réseau expérimentale considérée

¹ Modbus Poll est un émulateur d'un maître Modbus. Il sert à tester et à diagnostiquer des stations esclaves Modbus. Modbus Poll supporte Modbus RTU/ASCII et Modbus TCP/IP. www.Modbustools.org

² www.iperf.org

Dans cette architecture de commande, les communications s'effectuent comme suit :

- Le PLC scrute les modules MES1, MES2, MES3, MES4 et MES6 en lecture (Read Holding Register) et scrute le module MES1 en écriture/lecture (R/W Register). Un programme de commande est implémenté dans le PLC. Il consiste à copier, à chaque cycle de calcul, la valeur de l'entrée In1 obtenue depuis le MES1, et la mettre dans la zone mémoire dédiée à la variable envoyée en écriture vers le module MES5. Le but de cette boucle est simple ; transmettre la valeur de l'entrée In1 vers la sortie Out5. Le temps de réponse de cette commande est le délai qui s'écoule entre un changement d'état de l'entrée In1 et l'occurrence du même changement à la sortie Out5. C'est ce que nous mesurons expérimentalement.
- Le PC1 scrute ou non, suivant la configuration choisie, le module MES5 en lecture de 10 mots mémoires (Holding registers) grâce à Modbus Poll.
- Le PC2 sert aussi à perturber la boucle de commande avec du trafic UDP avec des paquets de taille maximale de 1470 octets et avec des débits variables. Les différents modules E/S déportés sont scrutés, suivant la configuration choisie.

5.1.2 Outils et procédure de mesure

La plateforme de mesure est composée de trois parties principales (Fig. 5.2) :

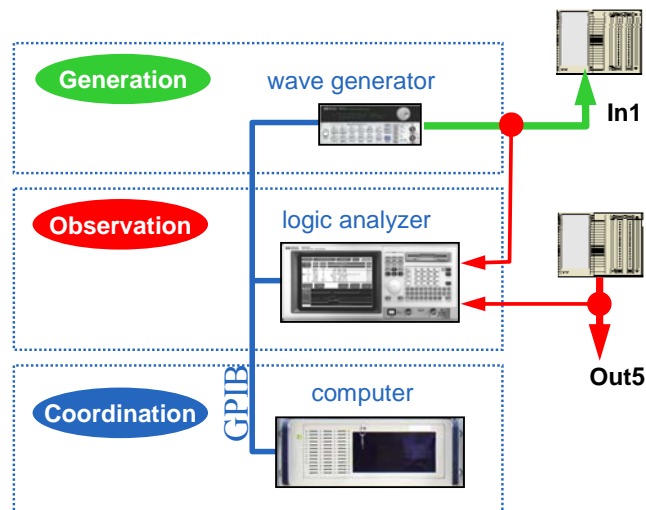


Fig. 5.2 : Composition de la plateforme de mesure du temps de réponse

- Un générateur de signal (HP30129A) : ce générateur permet de stimuler l'entrée In1 du module E/S déporté MES1. Pour ce faire, un signal logique carrée périodique est

généralisé avec une période T_{Gen} , suffisamment grande pour éviter de possibles ambiguïtés et erreurs dans la mesure du temps de réponse (voir la Fig. 5.3).

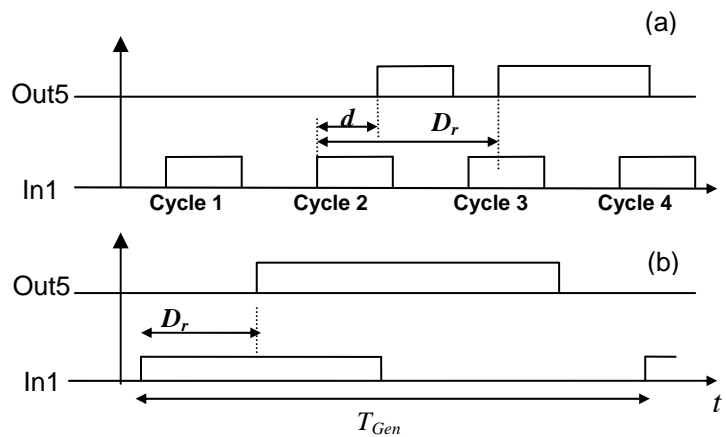


Fig. 5.3 : Mesure du temps de réponse (a) mesure erronée, (b) mesure correcte

Pour éviter des mesures erronées, il faut en effet laisser le temps au signal de l'entrée In1 de se propager et arriver à la sortie Out5. Dans le cas contraire, un front montant sur Out5 peut apparaître après plusieurs fronts montants de In1, desquels on ne peut choisir le bon puisque ces fronts ne sont pas distinguables. De ce fait, nous choisissons dans nos mesures une période T_{Gen} d'environ dix fois la période de scrutation T_{COM} .

- Un analyseur logique (HP301120A) : cet outil est très similaire à un oscilloscope classique mais est mieux adapté pour analyser des signaux logiques. De plus, pour la mémorisation des signaux, il présente l'avantage de ne pas conserver tous les instants d'échantillonnage des signaux et les amplitudes correspondantes. Au lieu de cela, il mémorise uniquement les instants d'apparition des changements sur les signaux logiques. Il en résulte naturellement une capacité de stockage relativement élevée. Pour mesurer le temps de réponse, cet analyseur est utilisé pour capter à la fois le signal de l'entrée In1 et le signal de la sortie Out5 puis les mémoriser. La période d'échantillonnage utilisée est de $50 \mu s$, ce qui permet une bonne précision de mesure.
- Un ordinateur de coordination : cet ordinateur est utilisé d'abord pour la configuration puis la coordination du générateur et de l'analyseur. Ensuite, à la fin de chaque série de mesure et de stockage des signaux de In1 et de Out5 dans l'analyseur, cet ordinateur les copie et les utilise pour calculer les temps de réponse. Vu le nombre de mesures de chaque série (environ 10.200), toutes les opérations effectuées par

l'ordinateur de coordination sont automatisées grâce un programme codé en Python³. Malgré cela, le temps nécessaire au traitement de chaque série de mesures est non négligeable (environ 30 mn).

5.2 Confrontation des résultats théoriques aux mesures sur un cas d'étude

Dans cette section, nous allons confronter nos différents résultats théoriques aux mesures expérimentales des temps de réponse (bornes et distribution). Pour ce faire, nous considérons le SCR de la Fig. 5.1 avec la configuration suivante :

- Le PLC fonctionne en mode périodique avec une période de 5 ms. Sa carte de communication COM fonctionne aussi périodiquement et scrute les modules E/S déportés MES1 jusqu'à MES6 toutes les 30 ms.
- Le PC1 scrute périodiquement le module MES5 toutes les 10 ms.
- Le PC2 est déconnecté (on ne considère que du trafic temps réel pour ce cas d'étude).

Nous allons confronter trois approches d'évaluation du temps de réponse de ce SCR :

- i) **Expérimentation** : une série d'environ 10.200 mesures expérimentales a été réalisée. La répartition des temps de réponse mesurés est représentée sur la Fig. 5.4a.

- ii) **Calcul itératif** des temps de réponse : rappelons que nous avons proposé dans la section 3.1 du Chapitre 3 une approche itérative pour le calcul des temps de réponse relatifs aux différents cycles de scrutation. La formule (5.1) suivante y est donnée :

$$D_r(l_m) = q_{l_m} \cdot T_{COM} + T_{Req}(l_m + q_l) - T_{Req}(l_m) + T_{E/S} - \tau_{l_m} \quad (5.1)$$

Pour calculer les temps de réponse, nous avons généré les valeurs des délais élémentaires relatifs à chaque cycle de scrutation en utilisant des distributions échantillonnées. Par exemple, la distribution des décalages τ_{l_m} est considérée uniforme pour imiter le caractère périodique du signal carré du générateur de notre plateforme. Ainsi, les valeurs de chaque délai relatif à un cycle étant connues, il suffit de les remplacer dans la formule (5.1) pour calculer le temps de réponse correspondant. Le processus est répété pour environ 10.200 cycles. La répartition des délais calculés est représentée sur la Fig. 5.4b.

- iii) **Calcul analytique** : rappelons que deux calculs analytiques sont développés dans la thèse : un calcul pour les bornes du temps de réponse et un autre pour le calcul de sa fonction de distribution :

5.2.1 Calcul des bornes du temps de réponse

Les bornes minimale et maximale du temps de réponse, calculées en utilisant les formules analytiques (3.44) et (3.45) du Chapitre 3 (p. 69), sont données comme suit :

³ Voir www.python.org

$$D_r^{\min}(l) = q_l \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + \Delta(l, q_l) + T_{E/S_D} \quad (5.2)$$

$$D_r^{\max}(l) = (q_l + 1) \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + \Delta(l-1, q_l + 2) + T_{E/S_D} \quad (5.3)$$

$$\text{où } \Delta(l, q_l) = T_{Req_D}(l + q_l) - T_{Req_S}(l)$$

Les valeurs numériques suivantes sont utilisées pour le calcul des bornes : $T_{COM} = 30ms$, $T_{CPU} = 5ms$, $T_{EM_i} = 0.25ms$, $T_{CAL} = 3ms$, $T_{E/S_D} = 0.7ms$, $T_{fil} = 0.06ms$. La valeur maximale du délai de bout-en-bout T_{Req_D} , calculée en utilisant les modèle (max,+) du Chapitre 4, est $T_{Req_D} = 1.5ms$. La valeur minimale du délai T_{Req_S} est négligée. La valeur du délai T_{A/R_S} étant très inférieure à T_{CPU} (de l'ordre de $2ms$), le nombre q_l est égal à 1 ($q_{\max} = 1$).

En remplaçant ces valeurs numériques dans les formules (3.44) et (3.45), nous aboutissons aux bornes :

$$D_r^{\min} = 29.31ms$$

$$D_r^{\max} = 62.51ms$$

5.2.2 Calcul de la fonction de distribution du temps de réponse

Le calcul de la fonction de distribution du temps de réponse se fait à travers le corollaire 3.1 du Chapitre 3 qui stipule que :

$$P(D_r(l) \geq D_r^{\lim}) = P(q_l = q_{\max}) \cdot P(\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\lim}) \quad (5.4)$$

Puisque $q_{\max} = 1$ (comme expliqué plus haut), nous avons : $P(q_l = q_{\max}) = 1$. Le calcul de la probabilité $P(D_r(l) \geq D_r^{\lim})$ se réduit à $P(\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\lim})$ et il s'ensuit que :

$$P(\bar{\Delta}(l-1, q_l + 2) \geq \bar{\Delta}_{\lim}) = \int_{\bar{\Delta}_{\lim}}^{+\infty} f_{\bar{\Delta}}(t) \cdot dt$$

Nous savons par ailleurs que :

$$f_{\bar{\Delta}}(t) = (f_x * f_y * f_z)(t) = \int_{-\infty}^{+\infty} f_z(t-w) \int_{-\infty}^{+\infty} f_x(w-u) f_y(u) \cdot du \cdot dw \quad (5.5)$$

où $\bar{\Delta}(l-1, q_l + 2) = x + y + z$ avec $x = T_{Req_D}(l + q_l)$, $y = -T_{Req_S}(l-1)$, et $z = -\bar{\tau}_l$.

Ainsi, pour calculer la densité de probabilité (5.5), nous avons uniquement besoin des densités f_x , f_y , et f_z . Pour le cas du SCR de notre cas d'étude, nous considérons les distributions suivantes :

- La distribution du décalage $\bar{\tau}_i$ est supposée uniforme dans le domaine $[0, T_{COM}]$. La fonction f_z est alors donnée par :

$$f_z(t) = \begin{cases} 1/T_{COM} & \text{si } t \in [-T_{COM}, 0] \\ 0 & \text{sinon} \end{cases} \quad (5.6)$$

- Les densités de distribution des délais de bout en bout T_{Req_D} et T_{Req_S} sont supposées être gaussiennes. Ceci est justifié par des observations expérimentales de la distribution des ces délais (Denis *et al.*, 2007). Les délais T_{Req_D} et T_{Req_S} ont donc pour fonctions de distribution respectives $f_{(\mu_1, \sigma_1)}$ et $f_{(\mu_2, \sigma_2)}$ où :

$$f_{(\mu, \sigma)}(t) = 1/(\sigma\sqrt{2\pi}) \cdot \exp[-(t - \mu)^2 / (2\sigma^2)] \quad (5.7)$$

Les paramètres μ et σ sont respectivement la moyenne et l'écart type.

Nous avons montré dans (Addad *et al.*, 2011) que pour les distributions considérées, la fonction $f_{\Delta}(t)$ a pour expression :

$$f_{\Delta}(t) = 1/T_{COM} \cdot \int_t^{t+T_{COM}} f_{(\bar{\mu}, \bar{\sigma})}(w) \cdot dw \quad (5.8)$$

où : $\bar{\mu} = \mu_1 - \mu_2$ et $\bar{\sigma}^2 = \sigma_1^2 + \sigma_2^2$.

Cette expression peut être réécrite :

$$f_{\Delta}(t) = 1/(2T_{COM}) \cdot \left[\text{erf} \left(\frac{t + T_{COM} - \bar{\mu}}{\bar{\sigma}\sqrt{2}} \right) - \text{erf} \left(\frac{t - \bar{\mu}}{\bar{\sigma}\sqrt{2}} \right) \right] \quad (5.9)$$

où $\text{erf}(t) = 2/\sqrt{\pi} \cdot \int_0^t \exp(-w^2) \cdot dw$ est ce qu'on appelle la "Gauss error function". Cette transformation est due au fait que la fonction $f_{\Delta}(t)$ n'est pas intégrable analytiquement. La fonction $\text{erf}(t)$ est quant à elle intégrée dans la majorité des logiciels de calcul mathématique (ex : Matlab pour notre étude).

Les valeurs retenues pour les paramètres μ_i et σ_i sont 0.9 ms et 0.25 ms . Ces valeurs ont été déterminées par identification et ajustées de sorte à avoir un délai de bout-en-bout maximal d'environ 1.5 ms et toujours positif.

Finalement, la fonction de distribution du temps de réponse de notre SCR est représentée sur la Fig. 5.4c.

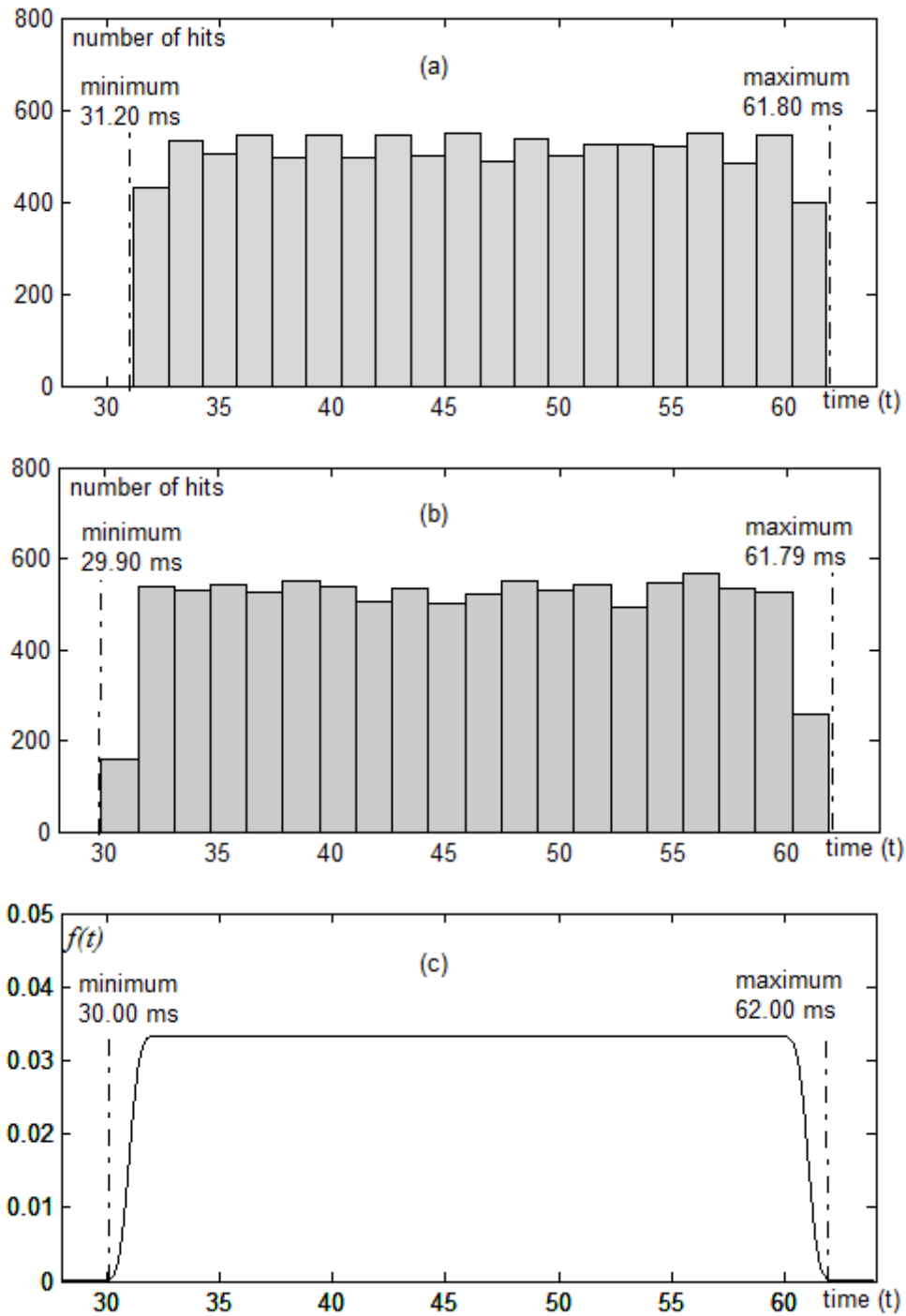


Fig. 5.4 : Temps de réponse du SCR de la Fig. 5.1 (a) mesures expérimentales (b) calcul itératif (c) calcul analytique de la distribution

Discussion des résultats

- Tout d'abord, nous remarquons que la borne maximale calculée analytiquement à l'aide de la formule (3.45), soit $D_r^{\max} = 62.51 \text{ ms}$, dépasse toutes les valeurs du temps de réponse évaluées avec les autres méthodes. Ce dépassement est de seulement 1%

par rapport à la valeur expérimentale. Par ailleurs, la borne minimale analytique $D_r^{\min} = 29.31 \text{ ms}$ est inférieure à toutes les autres valeurs évaluées. Ceci montre bien le caractère exhaustif des formules des bornes où même les cas les plus improbables, sont pris en compte.

- Contrairement aux formules de calcul des bornes, la méthode itérative n'est pas exhaustive. Pour preuve, nous pouvons noter que la borne maximale de tous les temps de réponse calculés itérativement est de 61.79 ms . Cette valeur est inférieure non seulement à la borne analytique mais aussi à la borne expérimentale. Ceci prouve que certains *scenarii* n'ont pas été balayés par la méthode itérative. Nous pouvons faire la même remarque concernant la borne minimale. Les écarts dans ces évaluations ne sont pas très importants mais révèlent le risque de passer à côté des pires cas en utilisant ce type de méthodes, à l'instar de la simulation.

Ceci dit, la méthode itérative donne quasiment la même valeur moyenne, autour de 45 ms , que les autres approches. Elle peut donc être utilisée sans problème pour des aspects qualitatifs (ex : qualité de commande).

- Nous pouvons remarquer également que les allures des distributions des temps de réponse sont très similaires pour les trois méthodes. La courbe de la fonction de distribution analytique (Fig. 5.4c) est clairement similaire à l'histogramme des mesures expérimentales. Le taux de recouvrement est d'ailleurs d'environ 94%. Concernant les bornes du temps de réponse données sur la courbe de la Fig. 5.4c, soient 62 ms pour la maximale et 30 ms pour la minimale, elles ne sont données que pour montrer visuellement que la probabilité d'avoir un temps de réponse en dehors de ces valeurs est extrêmement faible. Mais encore une fois, les formules des bornes (3.44) et (3.45) prouvent, même si cela n'est pas très probable, que cela est possible.

Finalement, cette étude nous montre que les évaluations obtenues par les approches développées dans ce manuscrit sont assez satisfaisantes, avec une couverture de tous les *scenarii* possibles d'une part et une bonne précision d'autre part.

Nous avons étudié d'autres cas de SCR et avons abouti aux mêmes conclusions (Addad et al., 2010c).

Pour viser une validation plus rigoureuse en revanche, il nous faut considérer davantage de cas. Pour aller dans ce sens, nous allons nous pencher dans le prochain paragraphe sur la

validité de la formule analytique de calcul de la borne maximale du temps de réponse à travers diverses mesures bien ciblées.

5.3 Validation de la formule de la borne maximale du temps de réponse

La borne maximale du temps de réponse étant souvent la plus importante et la plus difficile à évaluer, nous allons vérifier la validité de la formule analytique (3.45), développée pour son évaluation. Rappelons que cette formule est donnée par :

$$D_r^{\max}(l) = (q_{\max} + 1) \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + T_{Req_D}(l + q_{\max}) - T_{Req_S}(l - 1) + T_{E/S_D}$$

où $q_{\max} > (T_{A/R} + 2 \cdot T_{CAL}) / T_{COM}$ (**)

Rappelons que (**) est la version simplifiée de la condition de calcul du temps de réponse. C'est aussi la condition pour le cas d'un fonctionnement cyclique du contrôleur. L'analyse reste cependant la même si on est amené à considérer la version originale avec un fonctionnement périodique du contrôleur.

Pour étudier la validité de la formule (3.45), nous allons changer la configuration du SCR en ciblant à chaque fois un paramètre particulier de cette formule et en faisant une analyse de sensibilité. Pour cela, nous analysons l'impact de l'évolution de ce paramètre sur la borne maximale mesurée et comparons cela aux résultats calculés. Autrement dit, si la variation de la borne maximale est linéaire par rapport à un paramètre donné, suivant cette formule, nous allons observer la même variation dans les mesures expérimentales.

Parmi les paramètres que nous allons cibler, certains sont réglables par l'utilisateur par configuration du SCR. Par exemple, la période de scrutation T_{COM} du contrôleur est à choisir parmi un ensemble de valeurs proposées par le constructeur : 10 ms , 20 ms , 30 ms , ...

En revanche, certains paramètres ne peuvent être modifiés qu'indirectement. Par exemple, le délai de bout-en-bout T_{Req_D} . Pour le faire varier, nous allons charger le module E/S déporté MES5 avec plus ou moins de requêtes. Plus précisément, nous changeons le débit de scrutation du MES5. Ceci est possible grâce l'outil Iperf qui permet de générer du trafic UDP avec des débits variables.

Pour notre analyse, nous allons donc étudier la sensibilité de la formule (3.45) aux paramètres suivants :

- Période de scrutation T_{COM} : réglée par l'utilisateur
- Le délai T_{Req_D} : en chargeant le module MES5

- Le délai $T_{A/R}$: en chargeant le module MES1
- Des paramètres autres que ceux de la formule : en chargeant le module MES4, MES7, et MES8

- Remarque 5.1 :

- Certains paramètres de la formule ne sont pas évoqués précédemment. Par exemple, le terme $\sum_{i=1}^D T_{EM_i} + T_{E/S_D}$ est constant et il n'est pas évident de le faire varier.

Cependant, le fait de faire varier T_{Req_D} implique indirectement la variation de ce terme. En effet, retarder l'arrivée d'une requête vers le module MES5 ou bien retarder son traitement dans ce module revient à exactement la même chose. Que ce soit au niveau de la carte de communication, en augmentant l'ordre d'envoi de la requête (en augmentant le délai $\sum_{i=1}^D T_{EM_i}$), ou bien au niveau du réseau de communication (en variant le délai T_{Req_D}), ou encore au niveau du module MES5 (en variant le délai T_{E/S_D}).

- Rappelons que nous nous intéressons à la borne maximale du temps de réponse. Dans la formule (3.45), le terme T_{Req_S} est précédé d'un signe négatif. Ce terme doit donc être le plus petit possible. De ce fait, si charger le module MES1 provoque inévitablement l'augmentation de la borne maximale de T_{Req_S} , cela n'est pas forcément le cas pour la borne minimale.

Charger MES1 ne devrait donc a priori pas avoir un impact significatif sur le temps de réponse maximal. N'oublions cependant pas que charger le module MES1 fait augmenter la borne maximale du délai d'aller retour $T_{A/R}$ et au delà d'un certain seuil, l'effet se voit rapidement sur le temps de réponse maximal. Nous allons voir tout cela plus concrètement avec les mesures.

5.3.1 Analyse de l'impact de la variation de la période de scrutation T_{COM}

Nous avons mené plusieurs séries de mesures en faisant varier la période de scrutation du PLC de 10 ms à 60 ms. Son mode de fonctionnement est cette fois cyclique. Les deux PC sont déconnectés pour cette expérimentation. Pour chaque période de scrutation du PLC, environ 10.200 temps de réponse sont mesurés et la valeur maximale est prélevée. Les

résultats sont affichés dans la Fig. 5.5. Sur cette figure, chaque point représente la valeur maximale des 10.200 temps de réponse mesurés.

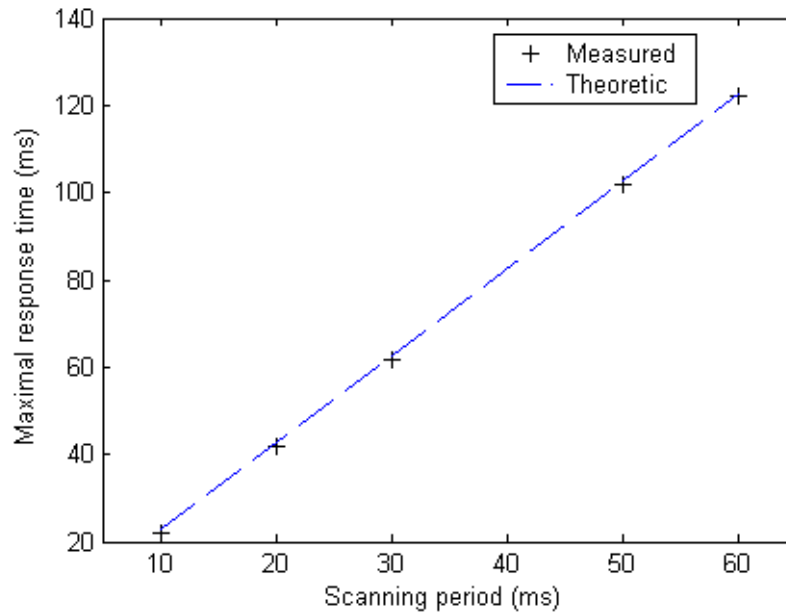


Fig. 5.5 : Impact de la variation de la période de scrutation sur le temps de réponse maximal

Nous remarquons que la variation de la borne maximale en fonction de la période de scrutation est quasiment linéaire. Le temps de réponse maximal est égal, dans tous les cas, à environ le double de la période de scrutation. Ceci est en accord avec la formule analytique (3.45) et cela peut s'expliquer comme suit :

- Les délais T_{CAL} et $T_{A/R}$, qui sont de l'ordre de 3 ms et 2 ms respectivement, sont relativement petit devant la période de scrutation la plus rapide qui est de $T_{COM} = 10\text{ ms}$ (c'est la plus petite possible avec les TSX 57203 dont nous disposons). Ceci implique que le nombre q_{\max} vérifiant la condition (***) est $q_{\max} = 1$. Or, l'augmentation de période de scrutation fait que le terme $(T_{A/R} + 2 \cdot T_{CAL}) / T_{COM}$ diminue et par conséquent renforce le fait que $q_{\max} = 1$ vérifie la condition (**). Au final, nous avons tout le temps $q_{\max} = 1$. La formule du temps de réponse devient donc :

$$D_r^{\max}(l) = 2 \cdot T_{COM} + \sum_{i=1}^D T_{EM_i} - \sum_{i=1}^S T_{EM_i} + T_{Req_D}(l + q_{\max}) - T_{Req_S}(l - 1) + T_{E/S_D} \quad (5.11)$$

Comme les délais de bout-en-bout sont petits devant la période de scrutation, la formule implique une borne maximale d'environ : $2 \cdot T_{COM} + \text{petit_terme}$. D'où l'allure quasi linéaire des mesures de la Fig. 5.5.

- Un autre facteur, allant dans le même sens, est le fait que l'augmentation de la période de scrutation fait diminuer l'encombrement dans le réseau est par conséquent diminuer le délai maximal aller retour $T_{A/R}$. Ceci renforce aussi la condition (**) pour $q_{\max} = 1$.

- Résultat :

Dans le cas où les délais de bout-en-bout sont inférieurs à 2 ms , comme c'est souvent le cas dans les SCR, une approximation de la formule analytique est possible. Elle est donné par :

$$\left(D_r^{\max} \approx 2 \cdot T_{COM} + 2 + T_{E/S_D}\right)ms$$

La courbe de cette formule approximée est représentée en pointillés sur la Fig. 5.5. Nous pouvons remarquer qu'elle est assez satisfaisante puisqu'elle passe au dessus, et très près, des points des mesures expérimentales.

Cela dit, l'hypothèse d'approximation n'est pas toujours vérifiée et il convient de l'utiliser avec attention. Par ailleurs, il faut s'assurer d'abord que T_{CAL} soit très inférieur à T_{COM} pour vérifier la condition (**), car ça n'est pas forcément le cas.

5.3.2 Analyse de l'impact de la charge d'un module E/S capteur (module MES1)

Pour charger le module MES1, nous avons utilisé Iperf. Le PC2 envoie des requêtes UDP de 1470 octets vers le module MES1 avec des débits allant de 0 Kps à 2000 Kps. Le résultat des mesures des temps de réponses maximaux est affiché sur la Fig. 5.6.

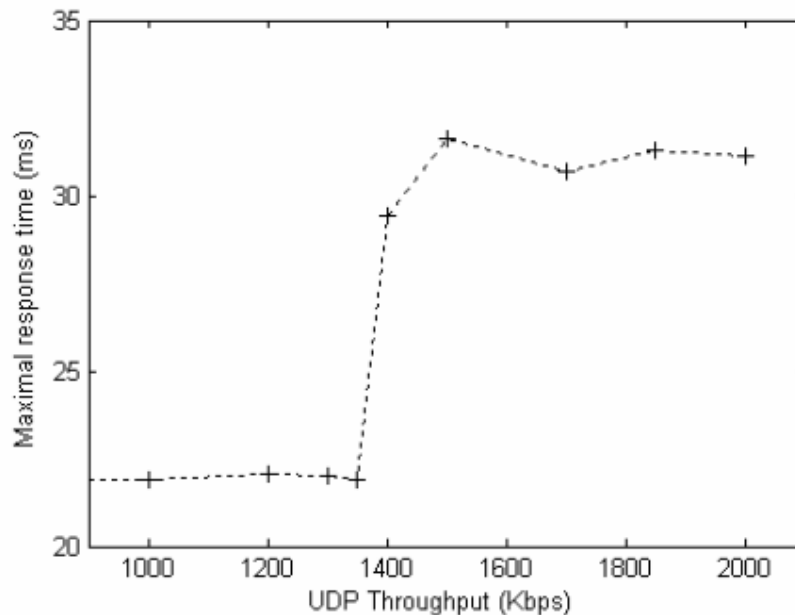


Fig. 5.6 : Impact de la charge du module MES1 sur le temps de réponse maximal

Comme nous pouvons le constater, cette courbe de mesures présente deux phases bien distinctes. Une première phase avec un temps de réponse aux alentours de 22 ms et une deuxième avec un temps de réponse autour de 32 ms. Ce résultat est aussi en accord avec la formule analytique et peut s'expliquer comme suit :

- Comme mentionné précédemment, charger MES1 revient à faire augmenter le délai maximal aller-retour $T_{A/R}$. Or, celui-ci n'intervient pas directement dans la formule mais plutôt dans la condition (**). Ceci implique que si $T_{A/R}$ augmente mais reste en deçà de la valeur du seuil qui fait basculer le nombre q_{\max} , alors aucun effet n'est aperçu sur la borne maximale du temps de réponse. C'est effectivement le cas dans la première phase dans la courbe. Mais si ce délai va au delà du seuil, un basculement intervient et le nombre q_{\max} passe de 1 à 2 puisque q_{\max} est un nombre entier. Ceci explique donc le basculement brusque, autour d'un débit UDP de 1350 Kbps, du temps de réponse maximal. Il passe brusquement de 22 ms vers la valeur d'environ 30 ms. Ce phénomène non trivial, observé expérimentalement, est finalement en accord avec la formule analytique qui inclut une composante non continue, le nombre entier q_{\max} .
- Dans chacune des deux phases de la courbe, également en accord avec la formule (3.45), il y a parfois des diminutions légères de la borne maximale en augmentant le débit UDP. Ceci est probablement dû au délai T_{Req_S} , qui rappelons le est précédé par un signe négatif dans la formule. Le fait de charger MES1 peut effectivement engendrer une augmentation de ce délai mais cela reste assez limité concernant le délai minimal.

5.3.3 Analyse de l'impact de la charge d'un module E/S actionneur (module MES5)

Pour charger le module MES5, nous avons procédé comme dans l'expérience précédente en envoyant des requêtes UDP de 1470 octets au module MES5 avec des débits variables. Le résultat des mesures est donné sur la Fig. 5.7.

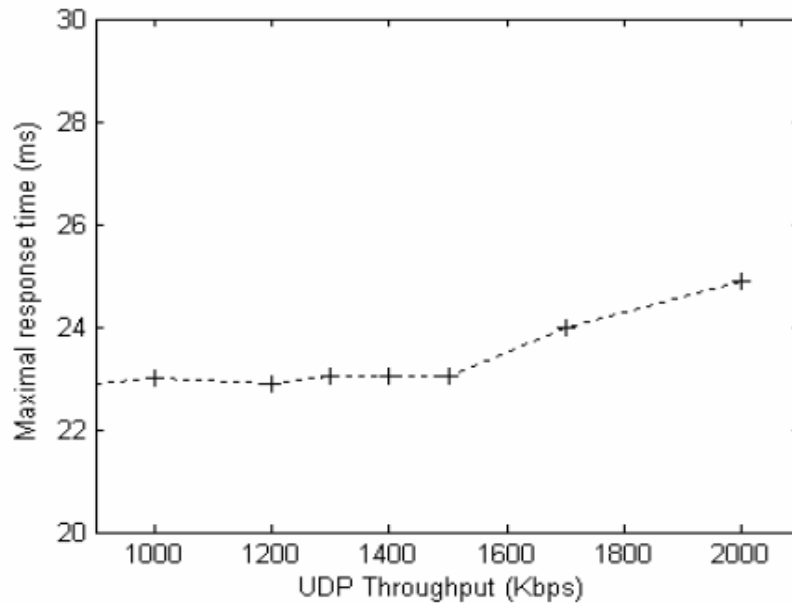


Fig. 5.7 : Impact de la charge du module MES5 sur le temps de réponse maximal

Comme nous pouvons remarquer, la variation du temps de réponse maximal est cette fois plus lisse que dans l'expérience précédente. Ceci est également en accord avec la formule qui prédit une variation presque linéaire avec le délai T_{Req_D} . Ainsi, l'augmentation du temps de réponse maximal est quasiment proportionnelle à celle du débit UDP comme nous le révèlent les mesures. La variation n'est cependant pas parfaitement linéaire et cela est dû à la variation collatérale, même si elle est légère, du délai T_{Req_S} .

5.3.3.1 Analyse de l'impact de la charge d'un autre module, ni capteur ni actionneur

Cette fois-ci, nous avons chargé le module MES4, qui n'est connecté ni au capteur ni à l'actionneur. Le résultat de l'impact de l'augmentation du débit sur le temps de réponse maximal est représenté sur la Fig. 4.8.

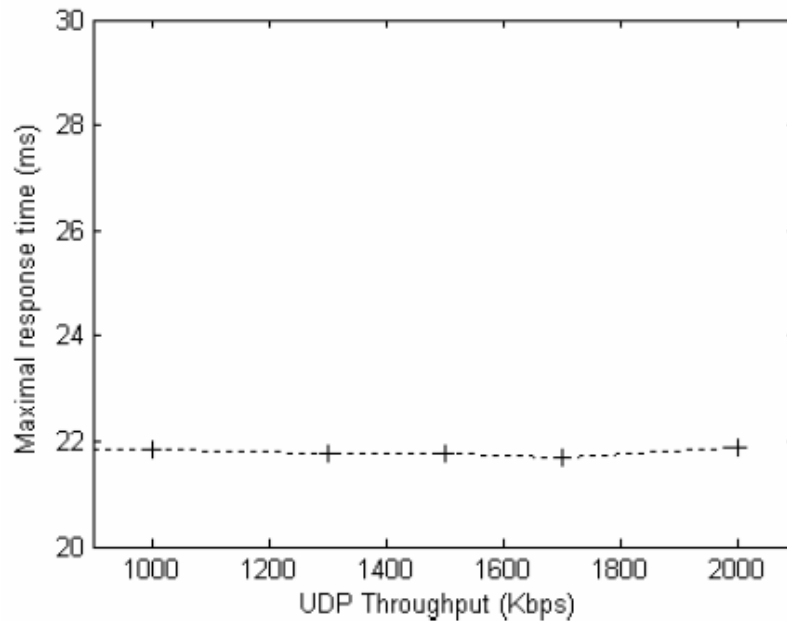


Fig. 5.8 : Impact de la charge du module MES4 sur le temps de réponse maximal

Contrairement aux précédentes expériences, cette fois nous ne constatons quasiment aucun effet suite au changement du débit de scrutation du module MES4. Le même constat a d'ailleurs été observé en chargeant d'autres modules comme le MES7 et le MES8. Ceci est également en accord avec la formule (3.45) puisque aucun de ses paramètres n'est ciblé. Le seul effet résultant de la charge de MES4 est au niveau des commutateurs. Or, la vitesse de transmission de ces derniers est assez rapide et le débit non temps réel UDP ne gêne pas remarquablement les paquets impliqués dans la boucle de commande. Autrement dit, les variations des délais T_{Req_D} , T_{Req_S} et $T_{A/R}$ sont négligeables et l'effet sur le temps de réponse également.

- Résultats

- En conclusion des expériences et des analyses précédentes, la formule analytique (3.45) est globalement en accord avec les mesures. Elle peut donc être utilisée avec suffisamment de confiance pour l'évaluation de la borne maximale du temps de réponse. Cela nous permet par la même occasion de valider les différentes hypothèses retenues concernant le fonctionnement des SCR étudiés.

- D'après la formule analytique (3.45), et en accord avec les observations expérimentales, il est recommandé de limiter le nombre de communications avec le module E/S capteur. Un nombre élevé d'échanges avec ce module peut en effet provoquer un changement brusque du temps de réponse maximal, ce qui peut avoir des conséquences dangereuses ...

- Par ailleurs, une charge du module E/S actionneur provoque une augmentation du temps de réponse même si celle-ci est relativement modérée, comparée à la précédente.
- Finalement, pour limiter l'impact d'un trafic parallèle sur une boucle de commande, il convient de ne pas charger les deux modules clés de la boucle : la source des événements (capteur) et la destination des conséquences (actionneur).

Conclusion & perspectives

Dans ces travaux de recherches, nous avons abordé le problème d'évaluation du temps de réponse des systèmes de commande en réseau. Dans le premier chapitre de ce manuscrit, nous avons rappelé les différents travaux ayant considéré des problématiques liées aux systèmes en réseau. Nous avons montré les avantages et les inconvénients des uns et des autres et un constat s'est dégagé rapidement. La majorité des études abordent l'évaluation des délais subis exclusivement dans le réseau de communication et ignorent souvent les délais liés aux composants de terrain. Or, comme nous l'avons montré tout le long de cette thèse, une évaluation complète passe par la prise en compte des délais subis dans tous les composants des SCR. Nous avons exposé des travaux qui prennent en compte tous les délais mais ils souffrent le plus souvent de limites importantes comme la non exhaustivité pour la simulation ou l'explosion combinatoire pour la vérification par model-checking. Nous avons tenté de lever ces difficultés en proposant une nouvelle approche analytique.

Dans le deuxième chapitre, nous avons abordé la modélisation des SCR client/serveur. Nous avons d'abord décrit le fonctionnement des différents composants technologiques utilisés en précisant par la même occasion nos hypothèses de travail. Ensuite, nous avons proposé des modèles des SCR à base de Graphes d'Événements Temporisés (GET). Pour faciliter la compréhension de la démarche, des modèles de complexité croissante ont été donnés.

Dans le troisième chapitre, les modèles GET obtenus précédemment ont été utilisés pour en extraire des équations (max,+) linéaires. A travers la résolution des équations, nous avons obtenu des solutions donnant les dates d'occurrence des événements remarquables dans les SCR. En traquant les messages circulant dans le SCR en utilisant ces dates, nous avons pu aboutir à des formules analytiques de calcul du temps de réponse pour chaque événement. En considérant des cas extrêmes par la suite, nous avons déduit les bornes minimale et maximale du temps de réponse. Nous avons également déduit une méthode de calcul itératif de l'allure de la distribution du temps de réponse. En poussant le raisonnement un peu plus loin, nous avons considéré une analyse probabiliste et trouvé la fonction caractérisant la densité de probabilité exacte du temps de réponse. Par la suite, nous avons abordé une étude de sensibilité du temps de réponse à différents paramètres des formules obtenues. Nous avons montré que les délais de non synchronisation dans les composants de terrain ont un impact prépondérant sur le temps de réponse. Nous avons démontré que les délais de bout-en-bout

sont tout aussi importants et qu'il convient de les évaluer pour calculer efficacement le temps de réponse. C'était l'objet du Chapitre 4 en considérant un réseau de communication de type Ethernet commuté.

Dans le quatrième chapitre, une approche d'évaluation de délais de bout en bout dans les réseaux Ethernet commuté a été développée. Pour ce faire, nous avons d'abord rappelé les travaux existants quant à l'évaluation de ces délais. Ensuite, nous avons décrit le fonctionnement d'un réseau Ethernet avant de proposer une stratégie de recherche du pire scénario, parmi un ensemble fini de scénarii, de traversée pour un paquet donné. Le déroulement de chaque scénario étant nécessaire pour calculer le délai correspondant, nous avons montré que cela était possible grâce à un ensemble d'équations (max,+) récurrentes. Deux stratégies de recherche du pire délai ont alors été proposées : une méthode combinatoire adéquate pour les SCR de moyenne taille et une méthode alternative à base d'algorithmes génétique pour les SCR de grande taille.

Enfin, le cinquième chapitre a consisté en la validation des résultats théoriques proposés. Nous avons exposé une série de mesures expérimentales et les avons comparées aux prédictions des formules analytiques obtenues. Nous avons ainsi pu constater une bonne concordance entre nos résultats théoriques et nos mesures. Ceci nous amené à conclure que les formules analytiques et les hypothèses sous-jacentes peuvent être valablement retenues pour l'évaluation des temps de réponse.

Dans cette thèse, nous avons abordé les seuls SCR fonctionnant suivant le protocole Client/serveur. Il serait intéressant maintenant de nous pencher sur les possibilités d'étendre nos résultats à des SCR fonctionnant suivant d'autres protocoles comme Producteur-consommateur. Nous pensons que cela devrait être possible sans trop de difficultés puisque les modèles des composants ne devraient pas être trop affectés et que la difficulté liée à la non synchronisation entre les contrôleurs et les modules E/S du protocole Client/serveur n'est pas présente dans le protocole Producteur/consommateur.

Nous avons supposé dans cette thèse que le SCR ne présente pas de pertes de paquets. Lever cette hypothèse, même si nous n'avons pas constaté de pertes durant nos expérimentations, pourrait ultérieurement se faire du point de vue théorique. Cela entraînerait cependant des modifications importantes sur les modèles, pour prendre en compte la retransmission des paquets perdus.

Concernant le réseau de communication, nous nous sommes penchés sur Ethernet commuté standard. Il serait également utile de considérer par exemple la classification de service,

implémentée de plus en plus dans les commutateurs afin d'améliorer les performances des applications temps-réel. Il est vrai que cela pourrait se révéler difficile puisque le principe même de la méthode proposée au Chapitre 4 pour l'évaluation des délais de bout-en-bout est basé sur la politique FIFO. Néanmoins, cela ne semble pas exclu en exploitant les modèles de commutateurs avec priorités proposés dans (Georges et al, 2005a).

Dans un contexte plus général de recherche fondamentale, l'extension de la méthode de représentation des systèmes impliquant des ressources partagées à l'aide d'équations $(\max,+)$ nous semble une perspective intéressante. En effet, plusieurs hypothèses plus au moins fortes ont été posées. Par exemple, relaxer l'hypothèse sur le fait que certains circuits dans les GETC soient saufs aurait des applications intéressantes pour des systèmes avec des ressources multiserveurs. Par ailleurs, la généralisation de la notion de séquence d'évolution d'un GETC, commune à toutes les ressources partagées, à une séquence propre à chaque ressource paraît une perspective intéressante. Trouver un modèle $(\max,+)$ linéaire pour tout GETC est une ambition de taille.

~ The only limit to our realization of tomorrow will be our doubts of today. ~

- Franklin D. Roosevelt

Bibliographie

- Abadeh M-S, J. Habibi, Z. Barzegar, M. Sergi, (2007). A parallel genetic local search algorithm for intrusion detection in computer networks, *Engineering Applications of Artificial Intelligence*, 20(8), 2007, pp. 1058-1069.
- Addad B., S. Amari (2008a). Modelling and response time evaluation of Ethernet based automation systems using Max-Plus algebra and timed events graphs, 4th IEEE Conf. Automation Science and Engineering, Washington DC, 2008, pp. 418-423.
- Addad B., S. Amari (2008b). Delays evaluation and compensation in Ethernet networked control systems, 16th Conf. on Real Time and Network Systems, Rennes, 2008, pp. 139-148.
- Addad B., S. Amari (2008c). Response time evaluation in Ethernet-based automation architectures, *2nd International Workshop on Verification and Evaluation of Computer and Communication Systems*, Leeds, 11 pp., July 2008
- Addad B., S. Amari (2009a). Evaluation de délais dans les systèmes de communication temps réel en utilisant des files d'attente virtuelles, *MSR - Journal Européen des Systèmes Automatisés (JESA)*, 43(7-9), pp. 985-1000, 2009.
- Addad B., S. Amari (2009b). Modélisation et évaluation temporelle des architectures de commande en réseau, *3èmes Journées Doctorales / Journées Nationales MACS (JD-JN-MACS'09)*, papier N°25, Angers, France, 17-18 mars, 2009.
- Addad B., S. Amari, et J-J Lesage, (2010a). Linear Time-Varying (Max,+) Representation of Conflicting Timed Event Graphs, *10th Int. Workshop on Discrete Event Systems*, Berlin, Germany, pp.310-315, 2010.
- Addad B., S. Amari, et J-J Lesage, (2010b). Modélisation de réseaux de graphes d'événements temporisés avec conflits dans l'algèbre (Max,+)", *IEEE Conférence Internationale Francophone d'Automatique*, papier N°22, Nancy, France, 2-4 Juin 2010.
- Addad B., S. Amari, et J-J Lesage, (2010c). Analytic calculus of response time in networked automation systems, *IEEE Trans. on Automation Science and Engineering*, 7(4), pp. 858-869, 2010.
- Addad B., S. Amari, et J-J Lesage, (2010d). Genetic algorithms for delays evaluation in networked automation systems, *Engineering Application of Artificial Intelligence*, 24(3), pp. 485-490, 2011.
- Addad B., S. Amari, et J-J Lesage, (2010e). A Virtual Queuing based Algorithm for Delays Evaluation in Networked Control Systems, *IEEE Transactions on Industrial Electronics*, DOI : [10.1109/TIE.2010.2098358](https://doi.org/10.1109/TIE.2010.2098358), 2010.
- Addad B., S. Amari, et J-J Lesage (2011). Client-Server Networked Automation Systems Reactivity: Deterministic and Probabilistic Analysis, *IEEE Trans. on Automation Science and Engineering*, 8(3), pp. 540-548, 2011.
- Alves M., E. Tovar et F. Vasques, (2000). Ethernet goes real-time: a survey on research and technological developments, Technical Report HURRAY-TR-2K01. Groupe de recherche IPP HURRAY, Polytechnic Institute of Porto (ISEP-IPP).

- Astrom K. J., Hang Lim (1994), A new Smith predictor for controlling a process with an integrator and long dead-time, *IEEE Trans. on Automatic Control*, Vol. 39, pp. 343-345.
- Baccelli F., G. Cohen et B. Gaujal, (1992a). Recursive equations and basic properties of timed Petri nets, *Discrete Event Dynamic Systems: Theory and Applications*, 1(4), 1992.
- Baccelli F., G. Cohen, G. J. Olsder, et J. P. Quadrat, (1992b). Synchronization and Linearity: An algebra for Discrete Event Systems, Wiley, 1992.
- Bauer H., J-L. Scharbarg, et C. Fraboul, (2009). Applying and optimizing Trajectory approach for performance evaluation of AFDX avionics network, *In IEEE Conf. On Emerging Technologies and Factory Automation*, Malloca, Spain, pp. 1-8, 2009.
- Bauer H., J-L. Scharbarg, et C. Fraboul, (2010). Improving the Worst-Case Delay Analysis of an AFDX Network Using an Optimized Trajectory Approach, *IEEE Trans on Industrial Informatics*, 6(4), pp. 521-533, 2010.
- Berge J., (2002). Introduction to Fieldbuses for Process Control, *The Instrumentation, Systems, and Automation Society*, 2002.
- Bolzern, P., P. Colaneri, et R. Scatolini, (1986). Zeros of discrete time linear periodic systems. *IEEE Transaction on Automatic Control*, vol. 31, pp. 1057-1058.
- Boutin, O., B. Cottenceau, A. L'Anton, et J-J. Loiseau, (2009). Modeling systems with periodic routing functions in dioid (Min,+), *13th IFAC Symposium on Information Control Problems in Manufacturing*, Moscow, Russia, pp. 1377-1382.
- Chang, C.S., R. Cruz, J-Y. Le Boudec et P. Thiran, (2002). A min-plus system theory for constrained traffic regulation and dynamic service guarantees, *In IEEE/ACM Transactions on Networking*. Vol. 10. pp. 805–817.
- Cheng H, et S. Yang, (2010). Genetic algorithms with immigrant schemes for dynamic multicast problems in mobile ad hoc networks, *Engineering Applications of Artificial Intelligence*, Available online February 2010.
- Cohen, G., S. Gaubert, and J. P. Quadrat, (1999). Max plus algebra and systems theory: where we are and where to go now, *Annual Reviews in Control*, vol. 23, pp. 207-219.
- Correia, A., A. Abbas-Turki, Bouyekhf, et A. El Moudni, (2009). A dioid model for invariant resource sharing problems, *IEEE Trans. on Systems, Man, and Cybernetics–Part A: Systems and Humans*, 39(4), pp. 770-781.
- Cruz R. L., (1991a). A calculus for network delay, Part I: network in isolation, *IEEE Trans. Information Theory*, Vol. 37, n° 1, 1991, p. 114-131.
- Cruz R. L., (1991b). A calculus for network delay, Part II: Network analysis, *IEEE Trans. Information Theory*, Vol. 37, n° 1, 1991, p. 132-141.
- Decotignie J.D., (2001). A perspective on Ethernet-TCP/IP as a fieldbus, *In 4th IFAC International Conference on Fieldbus Systems and their Applications*, Nancy, France. pp. 138–143.
- Denis B., S. Ruel, J-M. Faure, et G. Marsal, (2007). Measuring the impact of vertical integration on response times in Ethernet fieldbuses, *In IEEE Conf. Emerging Technologies and Factory Automation*, Patras, Greece, pp. 532-539, 2007.
- Diouri I. (2010). Proposition de méthodes pour adapter le réseau aux contraintes d'application temps-réel, PhD thesis. Université Henry Poincaré de Nancy, Ecole doctorale IAEM Lorraine.

- Fan X., M. Jonsson, et J. Jonsson, (2008). Guaranteed real time communication in packet switched network with FCFS queuing, *Computer and Networks*, Vol. 53, n° , 2008, p. 400-417.
- Ferrari P., A. Flammini, D. Marioli, et A. Taroni (2006a). An experimental approach to estimate real-time characteristic of PROFINET IO versus PROFIBUS DP V2, *Transactions on Circuits and Systems*, April, 2006, Vol. 5, N. 4, pp. 485-492, ISSN 1109-2734.
- Ferrari P., A. Flammini, et S. Vitturi (2006b). Performance analysis of PROFINET networks, *Computer Standards and Interfaces*, 28(4), pp. 369-385, 2006.
- García-Nieto J., J. Toutouh, et E. Alba, (2010). Automatic tuning of communication protocols for vehicular ad hoc networks using metaheuristics, *Engineering Applications of Artificial Intelligence*, Available online 5 February 2010.
- Gaubert, S., et Mairesse, J., (1999). Modeling and analysis of timed Petri nets using heaps of pieces. *IEEE Transactions on Automatic Control*, 44(4), pp. 683-697.
- Georges J. P., E. Rondeau, et T. Divoux, (2005a). Confronting the performances of switched Ethernet network with industrial constraints by using Network Calculus, *Communication Systems*, 18(9), pp. 877-903, 2005.
- Georges J-P., T. Divoux, et E. Rondeau (2005b). Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques, PhD thesis. Université Henry Poincaré de Nancy, Ecole doctorale informatique et télécommunications.
- Georges J-P, (2006). A design process of switched Ethernet architectures according to real-time application constraints, *Engineering Applications of artificial intelligence*, 19(3), pp. 335-344, 2006.
- Greifeneder J. and G. Frey, (2006). Optimizing Quality of Control in Networked Automation Systems using Probabilistic Models, In *11th IEEE Conf. on Emerging Technologies and Factory Automation*, Prague, Czech Republic, pp. 372-379, 2006.
- Greifeneder J. et G. Frey G., (2007). Probabilistic Timed Automata for Modelling Networked Automation Systems, In *Proc. of the 1st IFAC Workshop on Dependable Control of Discrete Systems DCDS*, pp. 143-148, Cachan, France, 2007.
- Greifeneder J. et Frey G., (2008a). Reactivity Analysis of different Networked Automation System Architectures, In *13th IEEE Int. Conf. Emerging Technologies and Factory Automation*, Hamburg, Germany, pp. 1031-1038, 2008.
- Greifeneder J. et Frey G., (2008b). Comparing Simulative and Formal Methods for the Analysis of Response Times in Networked Automation, In *IFAC world congress*, Seoul, South Korea, 2008.
- Grieu J. (2004). Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques, PhD thesis. Institut National Polytechnique de Toulouse, Ecole doctorale informatique et télécommunications.
- Gunter B., G. Stefan, de. M. Hermann, et S. T. Kishor, (2006). *Queuing networks and Markov chains*, Editions John Wiley & Sons, 2006.
- Hägglund T. (1996), An industrial dead-time compensating PI controller, *Control Engineering Practice*, 4(6), pp. 749-756.

- Hillion H. et J. M. Proth, (1989). Performance evaluation of jobshop systems using timed event graphs, *IEEE Transaction on Automatic Control*, 34(1), pp. 3-9.
- Holland J., *Adaptation in natural and artificial systems*, Ed: Ann Arbor university of Michigan press, 1975.
- Houssin L., S. Lahaye, J-L. Boimond, Just in time control of constrained (max,+) linear systems, *Discrete Event Dynamic Systems*, 17(2), pp. 159-178.
- IEEE (1998). IEEE standards for information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks – Common specifications - Part 3 : Media Access Control (MAC) bridges. ANSI/IEEE Std 802.1D, Edition 1998.
- Jasperneite. J, J. Imtiaz, M. Schumacher, et K. Weber, (2009). A Proposal for a Generic Real-Time Ethernet Systems, *IEEE Trans. on Industrial Informatics*, 5(2), pp. 75-85, 2009.
- Jasperneite J. et P. Neumann, (2001). Switched Ethernet for Factory Communication, *In 8th IEEE Int. Conf. Emerging Technologies and Factory Automation, ETFA* , Antibes, France, pp. 205 – 212, 2001.
- John N. D., (2005). *Queuing Theory with application to packet communication*, Boston, Editions Springer, 2005.
- Kamen, E.W., P. Torab, K. Cooper et G. Custodi (1999). Design and analysis of packet-switched networks for control systems. In : 38th IEEE Conference on Decision and Control (CDC'99). Vol. 5. Phoenix, Etats-Unis. pp. 4460–4465.
- Kjellson J, A. E. Vallestad, R. Steigmann, et D. Dzung, (2009). Integration of a Wireless I/O Interface for Profibus and Profinet for Factory Automation, *IEEE Trans. on Industrial Electronics*, 56(10), pp. 4279-4287, 2009.
- Komenda J., S. Lahaye, J-L. Boimond, Supervisory control of (max,+) automata: a behavioural approach, *Discrete Event Dynamic Systems*, 19(4), pp. 525-559.
- Krakora J., L. Waszniowski, P. Pisa et Z. Hanzalek (2004). Timed Automata Approach to Real Time Distributed System Verification, *In 5th IEEE International Workshop on Factory Communication Systems*, Vienne, Autriche, pp. 407–410.
- Krommenacker N., E. Rondeau et T. Divoux, (2002a). Genetic algorithms for industrial Ethernet network design, *In 4th IEEE International Workshop on Factory Communication Systems*. Vasteras, Suede. pp. 149–155.
- Krommenacker N., J.P. Georges, E. Rondeau et T. Divoux (2002b). Designing, modelling and evaluating switched Ethernet networks in factory communications systems, *In 1st International Workshop on Real-Time LANs in the Internet Age*. Vienne, Autriche, pp. 72–75.
- Lahaye, S., J-L. Boimond, et L. Hardouin, (2004) Linear periodic systems over dioids, *Journal of Discrete Event Dynamic Systems*, 14(2), pp 133-152.
- Le Boudec J. Y, et P. Thiran, (2004). *Network Calculus: a theory of deterministic queuing systems for Internet*, Editions Springer Verlag, 2004.
- Lee K. C. et S. Lee, (2002). Performance evaluation of switched Ethernet for real-time industrial communications”, *Computer standards & interfaces*, Vol. 24, pp. 411–423, 2002.

- Lee, K.C., H-K. Kim, S. Lee, M.H. Lee, (2004). Timer selection for satisfying the maximum allowable delay using performance model of Profibus Token passing protocol. *IEEE Transactions on Industrial Electronics* 51 (3), 701–710.
- Lee K. C., S. Lee, et M. H. Lee, (2006). Worst case communication delay of real time industrial switched Ethernet with multiple levels”, *IEEE Trans. on Industrial Electronics*, 53(5), pp. 1669-1676, 2006.
- Limal S. (2009). Architectures de controle-commande redondantes _a base d'Ethernet Industriel Modelisation et validation par model-checking temporise, PhD thesis, ENS Cachan (France), janvier 2009.
- Liu G. P., Y. Xia, J. Chen, D. Rees, et W. Hu (2007). Networked Predictive Control of Systems with Random Network Delays in Both Forward and Feedback channels, *IEEE Trans. on Industrial Electronics*, 54(3), pp. 1282-1297, 2007.
- Liu L., et G. Frey, (2006a). Simulation approach for evaluating response times in networked automation systems, In *IEEE Int. Conf. Emerging Technologies and Factory Automation*, Patras, Greece, pp. 1061-1068, 2007.
- Marsal G, B. Denis, J.-M. Faure, and G. Frey, (2006a). Evaluation of response time in Ethernet-based automation systems, In *IEEE Int. Conf. Emerging Technologies and Factory Automation*, Prague, Czech Republic, pp. 380-387, 2006.
- Marsal G., (2006a). Evaluation of time performances of Ethernet based automation systems by simulation of high level Petri nets, PhD thesis, ENS Cachan (France) and Kaiserslautern University (Germany), 2006.
- Martin S. et P. Minet, (2006). Schedulability analysis of flows scheduled with fifo: application to the expedited forwarding class, *Parallel and Distributed Processing Symposium*, pp. 8 pp.–, April 2006.
- Misra, P., (1996). Time-Invariant representation of discrete periodic systems, *Automatica*, 32(2), pp. 267-272.
- MODBUS Messaging on TCP/IP Implementation Guide V1.0b, (October 2006), [Online]. Available: www.modbus-ida.org/specs
- Murata T., (1989). Petri nets Properties analysis and applications, *Proceedings of the IEEE*, 77(4), pp. 541 – 580, 1989.
- Nait, S. M., M. A. Manier, A. El Moudni, et M. Wack, (2006). Petri net with conflicts and (max,plus) algebra for transportation systems, *11th IFAC Symposium on Transportation Systems*, Netherlands.
- Natori K et K. Ohnishi, (2008). A design Method of communication Disturbance Observer for Time-Delay Compensation, Taking the Dynamic Property of Network Disturbance Into Account, *IEEE Trans. on Industrial Electronics*, 55(5), pp. 2152-2168, 2008.
- Natori K, T. Tsuji, K. Ohnishi, A. Hace, et K. Jezernik, (2010). Time Delay Compensation by Communication Disturbance Observer for Bilateral teleoperation Under Time-Varying delay”, *IEEE Trans. on Industrial Electronics*, 57(3), pp. 1050-1062, 2010.
- Neumann P, “Communication in industrial automation: what is going on?”, *Control Engineering Practice*, 15(11), pp. 1332-1347, 2007.
- Passino K. M (2005), Biomimicry for optimization, control and automation, Chap IV: Evolution, Ed: Springer Verlag, 2005.

- Pereira N., E. Tovar, et L-M. Pinho, (2004a). From simulation to statistical analysis: Timeliness Assessment of Ethernet/IP based Distributed Systems, Technical Report HURRAY-TR-0416. Groupe de recherche IPP HURRAY, Polytechnic Institute of Porto (ISEP-IPP).
- Pereira N., E. Tovar, et L-M Pinho, (2004b). Timeliness in COTS factory-floor distributed systems : what role for simulation ?, *In IEEE International Workshop on Factory Communication Systems*, pp. 13-21, September 2004.
- Phipps M., (1999). A guide to LAN switch architectures and performance, In Packet Cisco Systems user magazine. Vol. 4. pp. 54–57. Cisco Systems.
- Rondeau, E., T. Divoux et H. Adoud (2001). Study and method of Ethernet architecture segmentation for Industrial applications. In : 4th IFAC Conference on Fieldbus Systems and Their Applications (FET'2001). Nancy, France. pp. 165–172.
- Rüping, S., E. Vonnahme et J. Jasperneite, (1999). Analysis of switched Ethernet networks with different topologies used in automation systems, *In 3rd IFAC International conference on Fieldbus Systems and Their Applications*. Magdeburg, Allemagne. pp. 351–358.
- Ruel S., O-de Smet, et J-M. Faure, (2008a). Building effective formal models to prove time properties of networked automation systems, In 9th International Workshop on Discrete Event Systems, pp. 334-339, 2008.
- Ruel S., O-de Smet, et J-M. Faure, (2008b). Efficient representation for formal verification of time performances of networked automation architectures, In 17th World Congress of The International Federation of Automatic Control, pages 5119{5124, Seoul, Korea, 2008.
- Ruel S., O. DE Smet, J.-M. Faure, (2009). Finding the bounds of response time of networked automation systems by iterative proofs, *In Proc. of 13th IFAC INCOM*, Moscow, Russia, pp. 1365-1370, 2009.
- Schafer I., M. Felser, (2007). Precision of Ethernet Measurements based on Software Tools, *In IEEE Conf. Emerging Technologies and Factory Automation*, Patras, Greece, pp. 510-515, 2007.
- Scharbag J-L et C. Fraboul, (2007). Simulation for end-to-end delays distribution on a switched Ethernet", *In Proc. of ETFA'07*, pp. 1092-1099, 2007.
- Scharbag J. L., F. Ridouard, et C. Fraboul, (2009). A probabilistic analysis of end to end delays on an AFDX avionics network, *IEEE Trans. Industrial Informatics Theory*, Vol. 5, n° 1, 2009, p. 38-49.
- Seno L, S. Vitturi, et C. Zunino, (2009). Analysis of Ethernet Powerlink Wireless Extensions Based on the IEEE 802.11 WLAN, *IEEE Trans. on Industrial Informatics*, 5(2), pp. 86-98, 2009.
- Schmidt K., et E.G. Schmidt, (2010). A Longest-Path Problem for Evaluating the Worst-Case Packet Delay of Switched Ethernet, *Symposium on Industrial Embedded Systems*, Trento, Italy.
- Silvola R., Hannila, J. Seppälä, J. et H. Koivisto, (2007). Experimental Performance Evaluation of Profinet IO Real-Time Version, *Automation 07 conference*, Helsinki, pp. 27-28, 2007.
- Smith O. J . M, A controller to overcome dead time, *ISA Journal*, 6(2), 1959.

- Song Y.Q. (2001). Time constrained communication over switched Ethernet, In 4th IFAC International conference on Fieldbus Systems and Their Applications (FET'01). Nancy, France. pp. 152–169.
- Starobinski D. et M. Sidi, (2000). Stochastically bounded burstiness for communication networks, *IEEE Transactions on Information Theory*, 46(1), 206–212.
- Tian X, X. Wang, et Y. Cheng, (2007). Self-tuning fuzzy controller for networked control system, *International Journal of Computer Science and Networks Security*, 7(1), pp. 97-102, 2007.
- Torab P. et E.W. Kamen, (1999). Load analysis of packet switched networks in control systems, In 25th Annual Conference of the IEEE International Electronics Society. Vol. 3. San Jose, California. pp. 1222–1227.
- Tovar E., et F. Vasques, (2001). Distributed computing for the factory-floor: a real time approach using WorldFIP networks, *Computers in Industry*, 44(1), pp. 11-31, 2001.
- Van Den Boom T-J, et B. De Shutter, , (2006). Modeling and control of discrete event systems using switching Max-Plus linear systems, *Control Engineering Practice*, 14(10), pp. 1199-1211.
- Vitturi S., (2001). On the use of Ethernet at low level of factory communication systems, *Computer Standards and Interfaces*, 2(4), pp. 267-277, 2001.
- Witsch D., B. Vogel-Heuser, J-M Faure, et G. Poulard-Marsal, (2006). Performance analysis of industrial Ethernet networks by means of timed model-checking, *12th IFAC Symposium on Information Control Problems in Manufacturing*, pp. 101–106, 2006.
- Zaitsev D.A., (2004a). Switched LAN simulation by colored Petri nets”, *Mathematics and Computers in Simulation*, 65(3), pp. 245–249, 2004.
- Zaitsev D.A., (2004b). An Evaluation of Network Response Time Using a Coloured Petri Net Model of Switched LAN, *5th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, Aarhus, 8-11 October 2004, pp. 157-167.
- Zhang Q., W. Zhang, Network partition for switched industrial Ethernet using genetic algorithm, *Engineering Applications of Artificial Intelligence*, 20(1), February 2007, pp. 79-88.

Résumé :

Les systèmes de commande en réseau (SCR) sont de plus en plus répandus dans le milieu industriel. Ils procurent en effet de nombreux avantages en termes de coût, de flexibilité, de maintenance, etc. Cependant, l'introduction d'un réseau, qui par nature est composé de ressources partagées, impacte considérablement les performances temporelles des systèmes de commande. Un signal de commande par exemple n'arrive à destination qu'après un certain délai. Pour s'assurer que ce délai soit inférieur à un certain seuil de sécurité ou du respect d'autres contraintes temps réels de ces systèmes, une évaluation au préalable, avant la mise en service d'un SCR, s'avère donc nécessaire. Dans nos travaux de recherche, nous nous intéressons à la réactivité des SCR client/serveur et évaluons leur temps de réponse.

Notre contribution dans ces travaux est d'adopter une approche analytique à base de l'algèbre $(Max,+)$ et remédier aux problèmes des méthodes existantes comme l'explosion combinatoire de la vérification formelle ou de la non exhaustivité des approches par simulation. Après modélisation des SCR client/serveur à l'aide de Graphe d'Événements Temporisés puis représentation de leurs dynamiques à l'aides d'équations $(Max,+)$ linéaires, nous obtenons des formules de calcul direct du temps de réponse. Plus précisément, nous adoptons une analyse déterministe pour calculer les bornes, minimale et maximale, du temps de réponse puis une analyse stochastique pour calculer la fonction de sa distribution. De plus, nous prenons en compte dans nos travaux tous les délais élémentaires qui composent le temps de réponse, y compris les délais de bout-en-bout, dus à la traversée du seul réseau de communication. Ce dernier étant naturellement composé de ressources partagées, rendant l'utilisation des modèles $(Max,+)$ classiques impossibles, nous introduisons une nouvelle approche de modélisation à base du formalisme $(Max,+)$ mais prenant en compte le concept de conflit ou ressource partagée. L'exemple d'un réseau de type Ethernet est considéré pour évaluer ces délais de bout-en-bout. Par ailleurs, cette nouvelle méthode $(Max,+)$ est assez générique et reste applicable à de nombreux systèmes impliquant des ressources partagées, au delà des seuls réseaux de communication.

Enfin, pour vérifier la validité des résultats obtenus dans nos travaux, notamment la formule de la borne maximale du temps de réponse, une campagne de mesures expérimentales sont menées sur une plateforme dédiée. Différentes configurations et conditions de trafic dans un réseau Ethernet sont considérées.

Mots clés : systèmes de commande, temps de réponse, réseau de communication, évaluation analytique, algèbre $(Max,+)$, graphe d'événements temporisés, ressource partagée, Ethernet.

Abstract:

Networked automation systems (NAS) are more and more used in industry, given the several advantages they provide like flexibility, low cost, ease of maintenance, etc. However, the use of a communication network in SCR means in essence sharing some resources and therefore strikingly impacts their time performances. For instance, a control signal does get to its destination (actuator) only after a non zero delay. So, to guarantee that such a delay is shorter than a given threshold or other time constraints well respected, an a priori evaluation is necessary before operating the SCR. In our research activities, we are interested in client/server SCR reactivity and the evaluation of their response time.

Our contribution in this investigation is the introduction of a $(Max,+)$ Algebra-based analytic approach to solve some problems, faced in the existing methods like state explosion of model checking or the non exhaustivity of simulation. So, after getting Timed Event Graphs based models of the SCR and their linear state $(Max,+)$ representation, we obtain formulae that enables to calculate straightforwardly the SCR response times. More precisely, we obtain formulae of the bounds of response time by adopting a deterministic analysis and other formulae to calculate the probability density of response time by considering a stochastic analysis. Moreover, in our investigation we take into account every single elementary delay involved in the response time, including the end-to-end delays, due exclusively to crossing the communication network. This latter being however constituted of shared resources, making by the way the use of TEG and $(Max,+)$ Algebra impossible, we introduce a novel approach to model the communication network. This approach brings to life a new class of Petri nets, called Conflicting Timed Event Graphs (CTEG), which enables us to solve the problem of the shared resources. We also manage to represent the CTEG dynamics using recurrent $(Max,+)$ equations and therefore calculate the end-to-end delays. An Ethernet-based network is studied as an example to apply this novel approach. Note by the way that the field of application of this approach borders largely communication networks and is quite possible when dealing with other systems.

Finally, to validate the different results of our research activities and the related hypotheses, especially the maximal bound of response time formula, we carry out lots of experimental measurements on a lab facility. We compare the measures to the formula predictions and check their agreement under different conditions.

Key words: automation systems, response time, communication networks, analytic evaluation, $(max,+)$ Algebra, Timed Event Graphs, shared resource, Ethernet.