Nancy-Université
Université Henri Poincaré

FACULTE DES SCIENCES & TECHNIQUES

U.F.R. Sciences et Techniques Mathématiques, Informatique et Automatique
Ecole Doctorale IAEM Lorraine
Département de Formation Doctorale Automatique

Thèse

Présentée pour l'obtention du titre de

Docteur de l'Université Henri Poincaré, Nancy I

En Automatique, Traitement du Signal, Génie Informatique

Par **Belynda BRAHIMI**

Proposition d'une approche intégrée basée sur les réseaux de Petri de Haut Niveau pour simuler et évaluer les systèmes contrôlés en réseau

Soutenance publique prévue le 05 Décembre 2007

Membres du jury :

Rapporteurs :	Mme Sylviane GENTIL	Professeur, INPG, Université de Grenoble
	M. Guy JUANOLE	Professeur, LAAS CNRS, Toulouse
Examinateurs :	M. Ye-Qiong SONG	Professeur, LORIA-U.H.P., Nancy
	M. Janan ZAYTOON	Professeur, CReSTIC, Université de Reims
	M. Eric RONDEAU	Professeur, U.H.P, Nancy (Directeur de thèse)
	M. Christophe AUBRUN	Professeur, U.H.P, Nancy (Directeur de thèse)

Centre de Recherche en Automatique de Nancy
UMR 7039, Nancy-Université, CNRS
Faculté des Sciences et Techniques – 54500 Vandoeuvre lès Nancy

« Ce travail de thèse est
dédié à ma mère,
ma sœur, ma grand-mère
et à toutes les belles
rencontres que j'ai pu faire
durant ce parcours… »

"On ne va jamais aussi loin que
lorsqu'on ne sait pas où l'on va"
-Christophe Colomb-

Table des matières

Introduction Générale

Depuis plusieurs décennies, les systèmes de contrôle/commande reposant initialement sur des architectures centralisées évoluent vers des organisations de plus en plus distribuées. La distribution des organes de commande permet notamment une meilleure flexibilité des installations qui doivent pouvoir évoluer en fonction des exigences du marché. Mais aussi, elle permet d'améliorer la réactivité du système en proposant une instrumentation dite « intelligente » intégrant des mécanismes de reconfiguration autonome qui doivent pouvoir faire face localement à des dysfonctionnements du process. Pour favoriser l'émergence de telles architectures, les communications point à point ont été remplacées par des réseaux dont les intérêts sont la réduction des coûts de câblage et de faciliter le partage de l'information du système réparti.

Cependant, les réseaux de communication génèrent des perturbations sur le système à commander en termes de retard, de pertes de données, … qui doivent être prises en compte dans la boucle de régulation du système asservi. De plus, les informations circulant dans le réseau sont de plus en plus volumineuses puisqu'elles ne reposent plus uniquement sur de l'échange de variables binaires, voire de quelques octets mais sur de l'information plus riche pouvant associer des données multimédia. Finalement, avec l'avènement de l'internet, les réseaux industriels se décloisonnent progressivement pour offrir de nouvelles fonctionnalités (télé-opération, e-maintenance,…) et pour intégrer les données de production dans des outils de gestion d'entreprises (ERP : Enterprise Resource Planning) multi-sites, multinationales.
Cette ouverture des réseaux peut rendre difficile la maîtrise des flux que véhicule le réseau industriel qui doit garantir une Qualité de Service (QdS) requise par l'application (QdC : Qualité de Contrôle). Cette problématique de recherche reposant sur l'adéquation de la QdS et de la QdC s'est fédérée au début des années 2000 sous la terminologie « Systèmes Contrôlés en Réseau : SCR », de la traduction anglaise « Networked Control Systems : NCS ». [Zhang, et *al*. 2001, Branicky et *al*. 2000, Walsh et *al*. 1999, Nilsson 1998, Lian et *al*. 2001, Georges 2005, Richard and Divoux 2007]. Mais on peut trouver leur fondement dans les années 80-90, sous les termes : « Communication Intégrée et Systèmes Commandés » (ICCS : Integrated Communication and Control Systems) [Ray 1989, Wittenmark et *al*. 1995].

Contexte

Le contexte général de cette thèse est donc celui des Systèmes Contrôlés en Réseau. Les SCRs sont des systèmes où les contrôleurs, les actionneurs, les capteurs et d'autres applications communiquent via un réseau de communication (figure I).

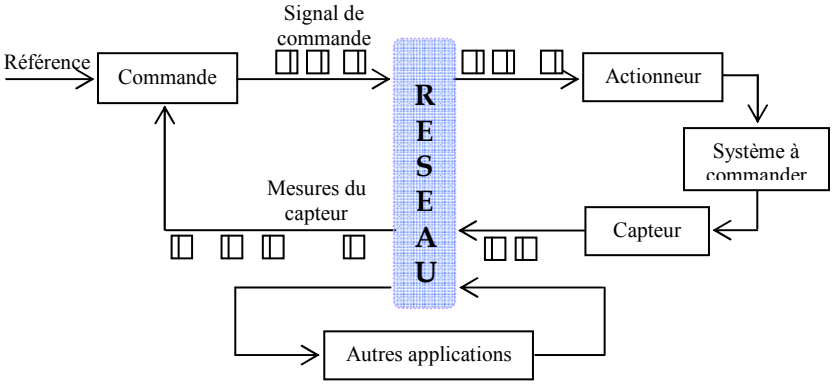


Figure 1. Système contrôlé en réseau.

On trouve les NCSs typiquement dans l'industrie aéronautique, automobile, ferroviaire, maritime [Ben Gaid et *al*. 2006, Grieu 2004] etc…

On distingue en général pour étudier les SCRs, deux types de réseaux [Lian et *al*. 2001] : les réseaux de données et les réseaux de contrôle. La différence entre ces deux réseaux réside dans le type d'information échangé. En effet, les réseaux de données sont caractérisés par des paquets de données de grande taille et sporadiques. En général, ces réseaux n'ont pas de contraintes temps réel dures à respecter. Par contre, les réseaux de contrôle traitent les paquets de petites tailles, qui sont envoyés fréquemment et doivent respecter des contraintes temps réel dures.

Les réseaux utilisés dans les SCRs sont souvent les réseaux de terrain tels que DeviceNet [Dev, 1997, Tindell et *al*. 95], ControlNet [Repère 1997] et FIP [Bergé 1996]…Ces réseaux ont une caractéristique commune, celle de satisfaire les contraintes temporelles des applications industrielles. Cependant, le coût, le manque d'interopérabilité et de flexibilité de ces équipements constituent une barrière réelle dans l'étude et le développement des systèmes industriels [Ji et Kim 05]. Pour pallier ces manques, le réseau Ethernet (norme 802.3 [IEEE 802.3]) est de plus en plus adopté et consensuel pour une utilisation dans un cadre industriel. Sachant qu'il n'était pas préalablement dédié pour les applications industrielles, puisque le réseau Ethernet est basé sur une méthode d'accès au médium CSMA/CD (Carrier Sense Multiple Access/ Collision Detection) non déterministe. Cependant, les nouveaux standards de l'Ethernet commuté comme la norme 802.1 D [IEEE 802.1 D] limitent le problème de l'indéterminisme [Ruping et *al*. 1999, Jasperneite and Neuman 2001, Rondeau et Divoux 2001, Lee et Lee 02, Ji et Kim 05]. Dans ce travail le réseau de communication choisie pour l'étude des systèmes contrôlés en réseau est l'Ethernet commuté.

Les SCRs présentent un aspect pluridisciplinaire qui requiert des connaissances en automatique, en réseau et en informatique. Les SCRs sont donc étudiés selon tous ces axes de recherche [Zhang, et *al*. 2001, Branicky et *al*. 2000, Walsh et *al*. 1999, Nilsson 1998, Lian et *al*. 2001, Georges 2005, Richard and Divoux 2007]. De même, des projets Européens (NeCST : Networked Control Systems Tolerant to fault), des comités scientifiques (IFAC TC 1.5 : Networked Systems), des projets nationaux (Safe-NeCS : Safe-Networked Control Systems, SACSS : Systèmes Automatisés Contraints en Sûreté de fonctionnement et Sécurité) rassemblent des groupes de recherche de compétences complémentaires.

La problématique des NCSs en général vise à contrôler et/ou adapter soit (a) le système de contrôle de l'application (qualité de contrôle) soit (b) le système de communication (qualité de service):

- Dans (a), il s'agit d'adapter l'application aux performances du réseau. Cette approche est référencée sous la terminologie anglaise « control over network ». Elle est en général abordée par les spécialistes de l'automatique et consiste à modéliser des stratégies de commande en fonction des délais et /ou la pertes des paquets causés par l'introduction d'un réseau dans la boucle de commande.

- Dans (b), il s'agit d'agir sur le réseau (control of network) pour fournir une QdS requise par l'application. Cette approche est abordée par les spécialistes en réseau et repose notamment sur des méthodes de contrôle de congestion et des algorithmes à évitement de congestion [Altman et *al*. 99, Mascolo 99].

Les deux approches sont complémentaires et la tendance actuelle est de combiner ces deux approches. On parle alors de modélisation intégrée ou de co-design. Dans le cadre de la modélisation intégrée, la qualité de service du réseau (*i.e*. la bande passante, la gigue, le retard et la perte des paquets et la fiabilité du réseau) est considérée simultanément avec la

performance du système commandé (*i.e.* stabilité, adaptabilité, robustesse et tolérance aux fautes).

Cadre: NeCST (Networked Control Systems Tolerant to fault)

Cette thèse s'inscrit dans le cadre d'un projet européen STREP n° IST – 2004-004303, s'intitulant: systèmes contrôlés en réseau tolérants aux fautes (NeCST : Networked Control Systems Tolerant to fault)[1]. Ce projet réuni plusieurs partenaires universitaires et industriels (Université Henri Poincaré, Université de Duisburg-Essen, Université de Technologie d'Helsinki, Université de Hull, Entreprise NESTE JACOBS OY, Entreprise PREDICT, Entreprise SAE AUTOMATION). L'objectif de ce projet est d'optimiser les performances des systèmes de communication en prenant notamment en compte les besoins applicatifs, les défaillances du réseau, et l'évolution de sa charge. Un autre objectif de ce projet est de proposer différents moyens pour améliorer le fonctionnement des composants embarqués, et développer des algorithmes et des méthodes pour détecter les fautes afin de commuter vers une stratégie de commande tolérante aux fautes. Enfin, la conception de tels systèmes doit se faire d'une manière intégrée et parallèle (co-design).

Ainsi, ce projet est basé sur le principe illustré sur la figure 2 complétée par un niveau de diagnostic. Ce niveau permet d'apporter une réponse globale pour compenser les fautes pouvant survenir dans le réseau et/ou sur le système commandé. Il s'appuie sur un ensemble d'actions coordonnées. Ces actions sont classées selon l'importance de l'intervention, et nous citons :

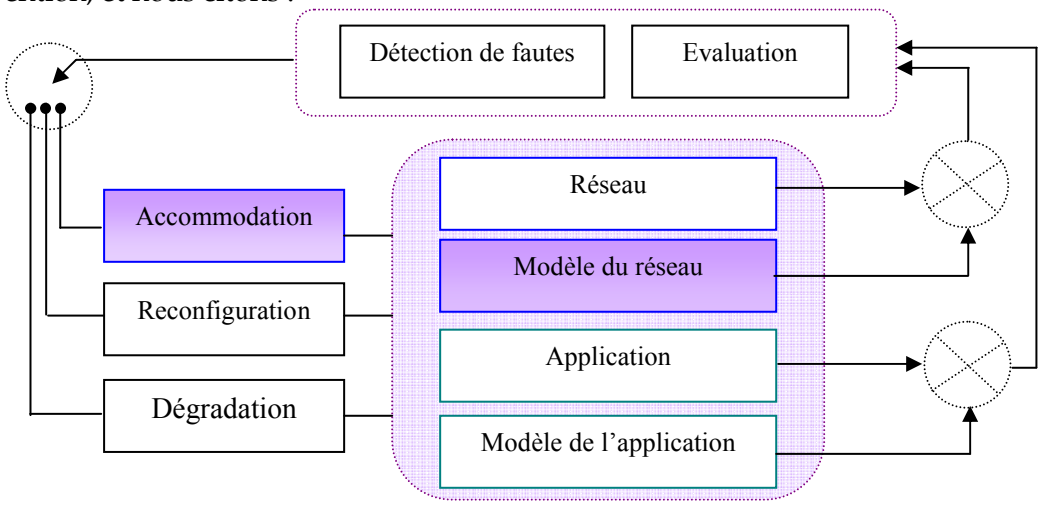


Figure 2. Principes de bases du projet NeCST.

Accommodation : consiste à poursuivre de façon continue ou à reprendre la mission sans remettre en cause les objectifs, cela suppose qu'il est possible de corriger, ou d'annuler les effets des défauts soit :

- par compensation des erreurs,
- par une procédure de reprise à partir d'un état initial connu (réinitialisation),
- par ajustement du régulateur du système ou du sous-système contenant l'élément défaillant (réadaptation des lois de commande par exemple – *i.e.* Même loi de commande avec des paramètres différents-).

[1] http://www.strep-necst.org

Reconfiguration : repose sur la détermination d'une nouvelle méthode (loi de commande par exemple) ou structure qui permet de prendre en compte l'influence des défauts, notamment par réactualisation des paramètres ou par modification de la structure.

Adaptation applicative : (dégradation ou retour à un fonctionnement normal) quand la reconfiguration ne permet plus de compenser l'effet des défauts. Une autre stratégie consiste à redéfinir les objectifs fixés pour atteindre des performances dégradées aussi proche que possible de celles visées en fonctionnement nominal (adaptation).

Différentes études ont été déjà menées au CRAN autour de cette problématique. Michaut et Lepage dans [Michaut et Lepage 2003] ont contribué à l'adaptation de l'application à la qualité de service du réseau, les travaux de [Rondeau et al. 2001] et [Krommenacker 2002] ont porté sur la reconfiguration de la topologie des réseaux Ethernet afin de respecter les contraintes temporelles prédéfinies. Les travaux de [Georges 2005, Georges et *al*. 2007] portent sur l'évaluation des performances des réseaux basés sur l'Ethernet commuté dans le contexte des systèmes contrôlés en réseau. Un modèle fonctionnel et analytique du réseau Ethernet basé sur la théorie du calcul réseau en vue de majorer les temps de traversée du réseau a été proposé.

Dans notre travail de thèse, nous proposons une approche intégrée pour l'étude des systèmes contrôlés en réseau, dont le but est de réduire les retards induits par le réseau Ethernet par des méthodes d'accommodation du réseau. Cette approche se veut être générique puisqu'elle relie la performance du système de commande et les services offerts par le réseau de communication.

Problématiques abordées

1- La première problématique concerne la modélisation et l'analyse de performance d'une architecture réseau Ethernet commuté où il faut pouvoir représenter des opérations de mémorisation, d'aiguillage, de classification et d'ordonnancement de paquets par lesquelles passe le flux dans un commutateur Ethernet. Plus précisément, il s'agit d'utiliser les réseaux de Pétri colorés (RDPC) [Jensen 92, 94] pour modéliser le réseau Ethernet commuté. Le formalisme des réseaux de Petri permet [Juanole et *al*. 04]:

- l'expression du parallélisme, de la synchronisation, du partage des ressources, des interactions,
- l'expression des caractéristiques temporelles (et stochastiques) associées aux mécanismes,
- une analyse qualitative (vérification de la logique des mécanismes non temporels et/ou temporels),
- une analyse quantitative (évaluation des performances fonctionnelles et/ ou de sûreté de fonctionnement).

Le choix de ce formalisme nous donne de plus, la possibilité d'intégrer dans le modèle du système de communication, la partie contrôle/commande applicative des SCRs.

Dans l'approche proposée, nous nous sommes contraints à une approche générique et modulaire qui requiert l'utilisation de coloration et la hiérarchie du modèle. L'analyse de performances du modèle est réalisée par simulation avec le logiciel CPNTools [Jensen 1997]. Ceci se justifie par le fait que les études analytiques sont difficilement envisageables à cause de la complexité des mécanismes d'ordonnancement de paquets.

2- La deuxième problématique concerne la mise en œuvre d'une méthode qui s'occupe de définir un seuil temporel qui permet d'évaluer la performance d'un système contrôlé. Il s'agit d'une méthode basée sur un test itératif de la stabilité asymptotique du système contrôlé, le seuil évalué constitue le retard toléré par le système de commande.

3- La troisième problématique concerne l'étude d'une méthode conjointe s'appuyant sur le seuil calculé comme indicateur de performance du système contrôlé et sur des mécanismes d'ordonnancement de paquets implantée dans le commutateur Ethernet. L'objectif de cette méthode est de réduire le retard induit par le réseau afin d'assurer la stabilité du système.

Organisation de la thèse

Cette thèse est composée de quatre chapitres.

Le premier chapitre décrit le contexte de notre travail à savoir les systèmes contrôlés en réseau. Ensuite l'influence du type d'ordonnancement de tâches, d'ordonnancement de paquets et de protocole du réseau sur la performance du système commandé est montrée sur un exemple modélisé sous Truetime. Enfin, les solutions proposées par la communauté automaticienne afin d'accommoder la commande au retard induit par le réseau sont exposées (approche (a)).

Dans le deuxième chapitre, les solutions proposées par les spécialistes en réseau (approche (b)) pour assurer la qualité de service imposée par l'application sont présentées. Ensuite des approches intégrées (co-design) qui prennent en compte de façon simultanée la qualité de service du réseau (c'est-à-dire la bande passante, la gigue, le retard et la perte des paquets et la fiabilité du réseau) et la performance du système commandé (c'est-à-dire la stabilité, la robustesse et la tolérance aux fautes) sont décrites.

Le troisième chapitre présente le réseau choisi pour l'étude des systèmes contrôlés en réseau, à savoir les réseaux Ethernet commutés. Un modèle en Réseaux de Petri colorés temporels et hiérarchiques (RDPCTH) des principales opérations par lesquelles passe le flux dans un commutateur Ethernet est présenté. Ensuite nous montrons comment exploiter ce modèle pour mesurer les paramètres de la qualité de service fournie par le réseau en fonction de la charge du nœud, et du type de mécanisme d'ordonnancement de paquets implanté. Deux mécanismes d'ordonnancement de paquets sont étudiés : le Weighted Round Robin (WRR) et l'ordonnancement à Priorité Stricte (PS). Deux propositions d'actions sur ces mécanismes d'ordonnancement sont présentées et analysées en simulant le modèle avec le logiciel CPNTools. L'évaluation de la qualité de service fournie par le réseau est effectuée dans un environnement producteur/consommateur périodique.

Le quatrième chapitre étend le modèle du commutateur Ethernet en y ajoutant l'environnement applicatif : la commande du système, l'actionneur, le système commandé et le capteur, afin de modéliser un système contrôlé en réseau. Le modèle RDPCTH du SCR est évalué et les résultats de simulation obtenus sont confrontés aux résultats du même modèle construit avec Matlab. Ensuite une méthode de calcul d'un seuil qui définit le délai toléré par le système commandé est présentée. Enfin, une méthode est proposée afin d'utiliser cette notion de seuil pour agir sur le réseau dans le but de réduire le délai et par conséquent de préserver la performance du système commandé. Les performances de cette méthode sont évaluées par simulation du modèle avec le logiciel CPNTools.

Chapitre I :
Systèmes contrôlés en réseau : de l'approche 'automatique'…

1. Introduction

L'objectif de ce chapitre est de développer la problématique générale des Systèmes Contrôlés en Réseau. Dans une première partie, l'architecture d'un SCR est introduite montrant les différents éléments qui le composent. Ensuite l'impact des performances du réseau sur les performances du système à commander. Enfin, les travaux sur les SCR portant sur l'adaptation de la commande du système pour compenser les perturbations générées par le réseau sont expliqués.

2. Les systèmes contrôlés en réseau

2.1 Introduction

Les SCRs peuvent se décomposer en trois parties (figure 1) : le contrôleur, le réseau et le procédé. Les échanges de données entre le contrôleur et le procédé se font en connectant les capteurs et les actionneurs sur le réseau. Le cheminement pour contrôler le système est donc le suivant : le contrôleur digital lit les valeurs de mesures de sortie du système, les compare aux valeurs désirées, et calcule l'entrée de commande à chaque intervalle de temps suivant la loi de commande choisie. La commande est encapsulée dans une trame puis elle est envoyée vers l'actionneur à travers un réseau de communication. L'actionneur récupère la trame puis en extrait la consigne. Celle-ci est maintenue constante durant chaque intervalle d'échantillonnage par un bloqueur d'ordre Zéro (BOZ) et est appliquée à l'entrée de commande du système commandé. Les informations d'état du système générées par les capteurs sont elles aussi encapsulées pour être transmises sur le réseau jusqu'au contrôleur.

Les retards dans les SCR ne sont donc pas dus uniquement au support de transmission mais correspondent à une somme de retards provenant des temps de conversion (A/D et D/A), des temps de traitement de la commande, des temps d'encapsulation/désencapsulation des messages, du temps d'acheminement des messages à travers le réseau et des temps de mesure et de réalisation de l'action. La contrainte pour pouvoir contrôler un procédé est que la somme de ces temps doit être inférieure à la période d'échantillonnage du système. Si certains temps sont facilement maîtrisables comme le temps de traitement de la commande, ou considérés comme négligeable, il est plus difficile d'estimer les retards induits par le réseau. Ces retards peuvent varier considérablement en fonction de l'évolution de la charge du réseau (problème de congestion) et même être infinis dans le cas d'une perte de trames.

Il faut donc pour étudier les Systèmes Contrôlés en Réseau :

- Réévaluer les théories de la commande basées sur des hypothèses telles que la synchronisation de la commande et de la sortie observée, et la capacité de réaction non-retardée du système etc…
- Prendre en compte le délai induit par le réseau (délais d'acheminent du message du capteur à la commande et de la commande à l'actionneur). Ce délai, qu'il soit constant ou variable peut dégrader les performances d'un SCR, voire même le rendre instable.
- Prendre en compte aussi les pertes, la duplication des paquets que peut induire le réseau de communication.

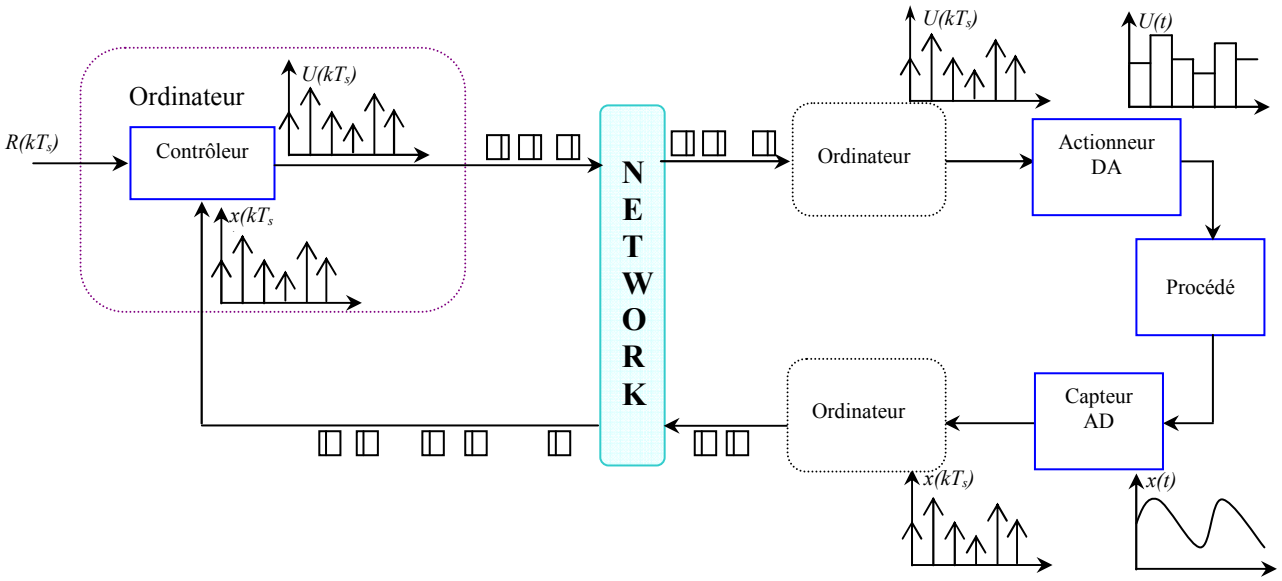


Figure 1. Représentation générale des systèmes contrôlés en réseau (sans retard induit par le réseau) avec T_s la période d'échantillonnage.

2. 2 Structures des systèmes contrôlés en réseau

Dans les Systèmes Contrôlés en Réseau, on distingue deux types de structure : la structure directe et la structure hiérarchique [Tipsuwan and Chow 2003] qui doivent être étudiées différemment.

2. 2. 1 Structure directe

La structure directe est composée d'une commande et d'un système distant contenant le système physique à commander, les capteurs et les actionneurs. La commande et le système commandé sont physiquement distants et sont reliés par un réseau de communication dans le but de réaliser une commande en boucle fermée telle qu'elle est illustrée dans la figure 2.
Un des exemples classiques rencontrés souvent dans la littérature est le système de commande d'un moteur [Tipsuwan and Chow 2001] et les systèmes d'apprentissage à distance [Overstreet and Tzes 1999].

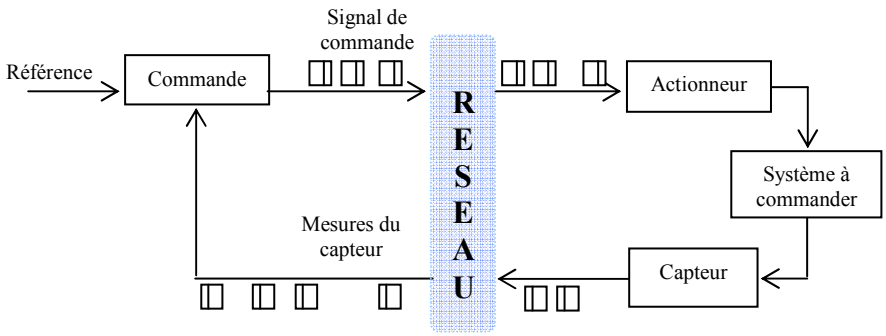


Figure 2. Structure directe d'un SCR.

2. 2. 2 Structure hiérarchique

La structure hiérarchique est constituée d'une commande principale et d'un système en boucle fermée distant (figure 3). La commande principale calcule et envoie périodiquement le signal de référence dans une trame via un réseau au système distant. Ce système distant traite le signal de référence pour exécuter la boucle de commande locale et envoie les mesures du capteur à la commande principale du système de commande en réseau. La boucle de contrôle en réseau a en général une période d'échantillonnage plus grande que la boucle de commande locale puisque la commande distante est supposée satisfaire le signal de référence avant de traiter les nouveaux signaux de références.

Similaire à la structure directe, la commande principale peut être configurée pour commander de multiples boucles de commande en réseau de différents systèmes distants.

Cette structure est utilisée dans beaucoup d'applications, principalement, dans les robots mobiles [Tipsuwan and Chow, 2002] et la télé-opération [Tarn and Xi, 1998].

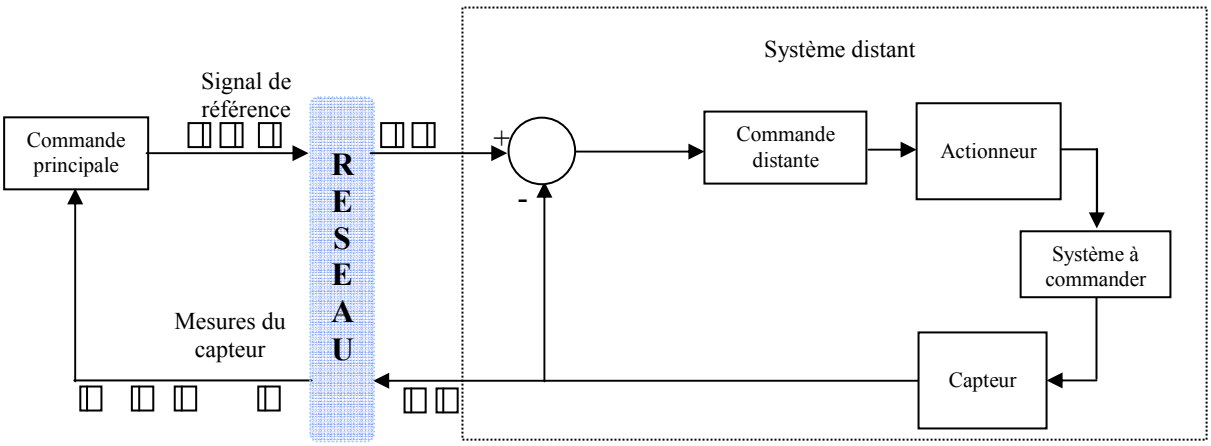


Figure 3. Structure hiérarchique d'un SCR

L'utilisation de l'une de ces deux structures dépend de l'application et de sa conception. Par exemple commander un robot, ou un bras en robotique exige l'utilisation de beaucoup de moteurs qui doivent fonctionner ensemble et simultanément. Donc, il est plus approprié d'utiliser une commande principale du robot et de formuler le système contrôlé en réseau en utilisant une structure hiérarchique.

Par opposition, lorsque le système est plus simple ou lorsque les temps de réponse exigés par le contrôleur principal doivent être plus rapide tel qu'un système de commande d'un DC moteur en réseau, la structure directe est préférable.

Dans le contexte de nos travaux de recherche, nous nous focaliserons sur la structure directe.

2. 3 Les réseaux dans les systèmes contrôlés en réseau

On distingue en général deux types de réseaux [Lian et *al*. 2001] : les réseaux de données et les réseaux de contrôle. La différence entre ces deux réseaux réside dans le type d'information échangé. En effet, les réseaux de données sont caractérisés par des paquets de données de grande taille et sporadiques. En général, ces réseaux n'ont pas de contraintes

temps réel dures à respecter. Quant aux réseaux de contrôle, ils traitent traditionnellement des paquets de petites tailles, qui sont envoyés fréquemment et ils doivent respecter des contraintes temps réel.

Les réseaux utilisés dans les SCRs sont souvent les réseaux de terrain tels que DeviceNet [Dev, 1997, Tindell et *al*. 95], ControlNet [Repère 1997] et FIP [Bergé 1996]…Ces réseaux ont une caractéristique commune, celle de satisfaire les contraintes temporelles des applications industrielles. Cependant, le coût, le manque d'interopérabilité et de flexibilité de ces équipements constituent une barrière réelle dans l'étude et le développement des systèmes industriels [Ji et Kim 05]. Pour pallier ces manques, le réseau Ethernet (norme 802.3 [IEEE 802.3]) est de plus en plus adopté. Cependant, le réseau Ethernet est basé sur une méthode d'accès au médium non déterministe, appelée CSMA/CD (Carrier Sense Multiple Access/ Collision Detection). Aussi les architectures Ethernet évoluent et sont passées d'une organisation en bus à une organisation en étoile en se basant sur la norme 802.1 D [IEEE 802.1 D] , plus connue sous le terme d'Ethernet commuté. L'Ethernet commuté associé au mode full-duplex (norme IEEE 802.1x) peut sous certaines contraintes assurer le déterminisme dans l'acheminement des messages [Ruping et *al*. 1999, Jasperneite and Neuman 2001, Rondeau et Divoux 2001, Lee et Lee 02, Ji et Kim 05] et peut donc être viable pour le transport d'information temps réel.

Dans cette thèse, le réseau de communication choisi pour étudier les Systèmes Contrôlés en Réseau est donc l'Ethernet commuté. Le premier avantage de ce type de réseau est le débit offert aux stations, l'Ethernet offre classiquement comme débit 10 Mbit/s ou 100 Mbit/s, et le gigabit est d'ores et déjà disponible. L'autre avantage est lié au fait qu'il soit un standard IEEE bien connu. Ceci implique une réduction des coûts, puisque les investissements peuvent être amortis sur de plus grands volumes. La flexibilité de ce réseau permet une réutilisation des outils de maintenance déjà développés pour d'autres applications. L'Ethernet commuté permet aussi l'utilisation des priorités (norme IEEE 802.1 p) pour pouvoir faire la classification de service. On pourra donc par cette norme différencier le trafic en fonction de ses contraintes applications et lui associer plus ou moins de bande passante. De plus, l'Ethernet est toujours en continuelle évolution et offre de plus en plus de possibilités pour l'industrie, comme des commutateurs Ethernet dits industriels résistants mieux aux environnements hostiles des sites de production. Ces avantages ont conduit à l'utilisation de l'Ethernet commuté par Airbus dans le système avionique de l'A380 [Grieu 2004].

3. Délais induits dans les systèmes contrôlés en réseau

Le délai est un paramètre inhérent aux réseaux, il peut être constant ou variable, négligeable ou important et peut même être infini en cas de pertes de trames. Les caractéristiques du délai dépendent du type du réseau utilisé et du type de protocole d'accès au média, de ce fait il peut être déterministe ou aléatoire. Mais à ce délai, il faut aussi ajouter les délais induits par le système.

3. 1 Délais induits par le système

Lorsque le contrôleur ou les cartes d'entrées/sorties reposent sur des équipements industriels (automates), ces délais sont constants et en général, connus. Il s'agit du délai de calcul de la commande τ_C , des délais de conversions des signaux (A/D et D/A), $\tau_{A/D}$, $\tau_{D/A}$ et du délai de mesure τ_S. Cependant, les architectures s'appuyant sur les nouveaux concepts SOA (Service Oriented Architecture), de Web service, OPC/Microsoft permettant

d'améliorer l'interopérabilité des systèmes tendent à complexifier la maîtrise temporelle des délais induits par le système. Dans cette thèse uniquement les délais induits par le réseau seront considérés. La figure 4 résume ces différents délais induits par les différents éléments de la boucle de commande (sans délai induit par le réseau).

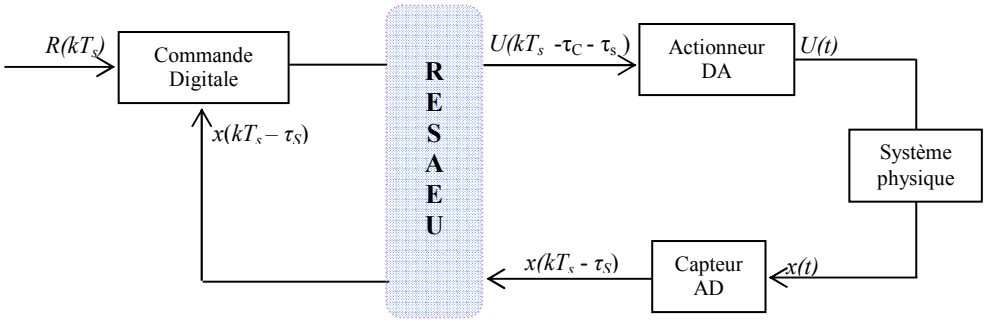


Figure 4. Modèle d'un système contrôlé en réseau et représentation des délais induits par le système (sans délai induit par le réseau).

3.2. Délais induits par le réseau

Les délais induits par le réseau peuvent être divisés selon le sens des transferts des données. On trouve le délai induit dans le sens de la commande vers l'actionneur τ_{CA} , et du capteur vers la commande τ_{SC}. Ces délais sont calculés comme suit :

$\tau_{CA} = t_{RA} - t_{EC}$, $\tau_{SC} = t_{CS} - t_{ES}$.

Où :

t_{EC} est l'instant où la commande encapsule le signal de commande dans un paquet pour l'envoyer à travers le réseau,

t_{RA} est l'instant où le signal de commande est reçu par l'actionneur,

t_{ES} est l'instant où le capteur encapsule la mesure du système commandé,

et, t_{CS} est l'instant où la commande traite la mesure reçue.

Ces deux délais induits par le réseau peuvent être inférieurs ou supérieurs à la période d'échantillonnage du système. Les différents délais système sont en général additionnés aux délais induits par le réseau pour simplifier leur analyse. Aussi, même si les délais systèmes existent toujours, ils sont souvent négligés comparés aux délais induits par le réseau et sont donc limités aux délais τ_{CA} et τ_{SC}. La Figure 5 montre les délais induits par le réseau.

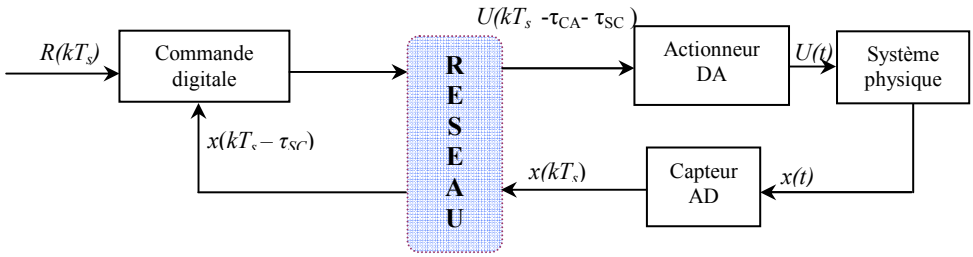


Figure 5. Modèle d'un système contrôlé en réseau et représentation des délais induits par le réseau.

La figure 6 montre le diagramme des temps d'un système contrôlé en réseau et le délai induit par le réseau (cas du délai inférieur à la période d'échantillonnage).

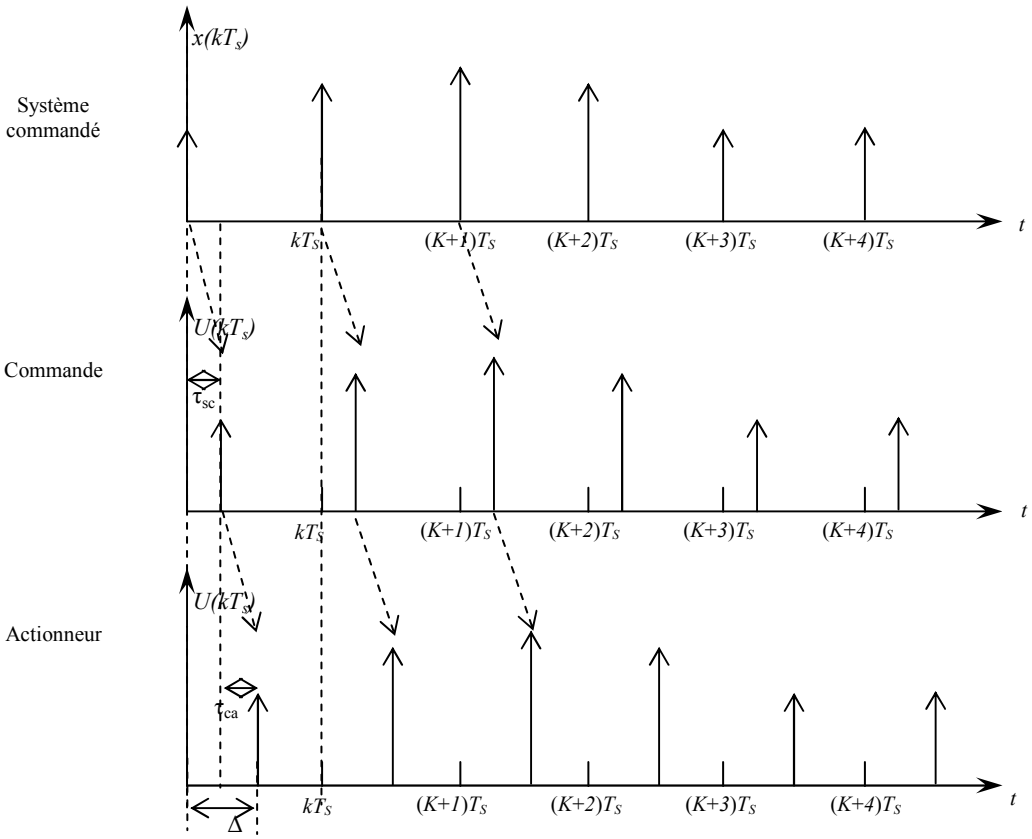


Figure 6. Représentation des délais induits par le réseau: Cas où le délai est inférieur à la période d'échantillonnage Ts.

3. 3 Les effets du délai sur les performances du système commandé

Les délais dans la boucle de commande d'un système de régulation tendent à dégrader ses performances. Pour illustrer cette dégradation, nous présentons une étude de cas.

Soit un système, représenté par la fonction de transfert suivante :

$$G_s(s) = \frac{1.5}{(s+0.5)(s+1.5)} \tag{1}$$

Ce système est commandé par un contrôleur PID (Proportionnel/Intégral/Dérivé), représenté par les paramètres suivants :

$K_p = 8$, $T_D = 0.2$, $T_I = 3.2$, et la fonction de transfert du contrôleur est comme suit :

$$G_C(s) = K_p\left(1 + \frac{1}{sT_i} + sT_D\right) \tag{2}$$

Le système de la figure 7 est utilisé pour montrer l'impact des délais sur la stabilité du système. La figure 8 montre les effets du délai sur la performance du système. Plus le délai est important plus les dégradations de la performance du système sont importants. Les délais dans la boucle de commande déstabilisent le système en réduisant sa marge de stabilité.

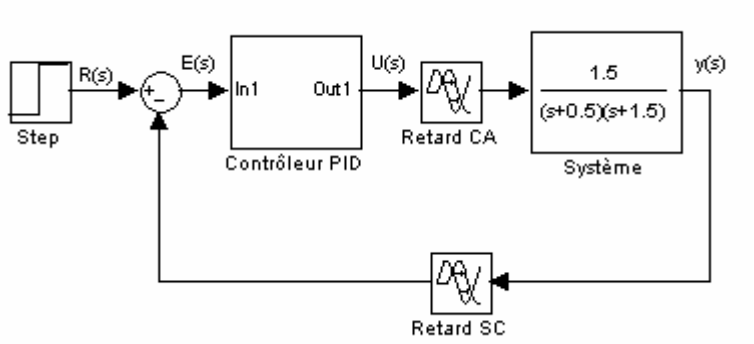


Figure 7. Système de commande en boucle fermée avec les retards

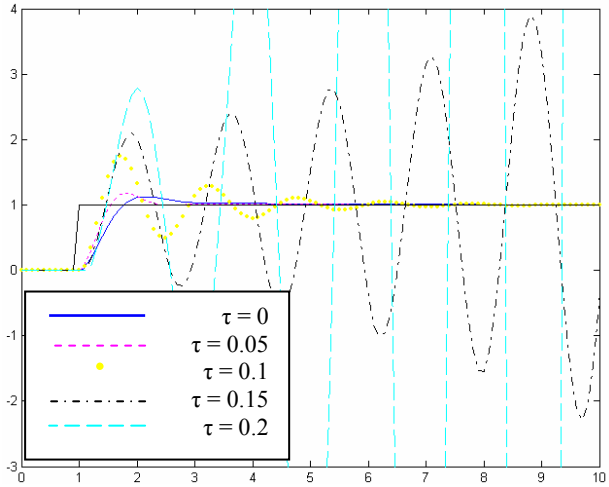


Figure 8. Signaux de sorties du système de commande en boucle fermée suivants les différents retards $\tau_{CA} = \tau_{SC} = \tau$.

Le calcul du diagramme de Bode du système et de sa marge de phase montre l'influence que peut avoir le délai sur la stabilité du système.

La représentation de Bode consiste à tracer séparément :

- le gain $A(\omega) = |G(j\omega)|$ en décibels ($[dB]$) : $A(\omega)\,|_{dB} = 20\log(|G(j\omega)|)$,
- la phase $\varphi(\omega) = \arg\{G(j\omega)\}$ en degrés ($[°]$) ou radians

en fonction des pulsations ω représentées sur une échelle logarithmique.

Nous avons vu que le retard induit par le réseau se décompose selon le sens du flux de l'information. On distingue alors le retard allant du contrôleur vers l'actionneur τ_{CA} et le retard du capteur vers le contrôleur τ_{SC}. Pour faciliter les calculs, ces deux délais vont être représentés par un seul retard $\tau_T = \tau_{CA} + \tau_{SC}$ [Zhang et *al*. 2001, Zhang 2001].

Ce retard donc va être considéré comme un retard pur de valeur τ_T de la fonction de transfert du système $G_{bo}(s) = G_c(s)Gs(s)$, et se distingue par la présence d'un terme $e^{-s\tau_T}$.
On a donc :

$$G(s) = G_{bo}(s)e^{-s\tau_T} \qquad (3)$$

La réponse harmonique $G(j\omega)$ de $G(s)$ se compose d'une partie rationnelle en $j\omega$:

$\frac{b_m(j\omega)^m + b_{m-1}(j\omega)^{m-1} + \cdots + b_1 j\omega + b_0}{a_n(j\omega)^n + a_{n-1}(j\omega)^{n-1} + \cdots + a_1 j\omega + a_0}$, ainsi que la contribution $e^{-j\omega\tau_T}$.

Concernant cette dernière, on a :

$$\begin{cases} \left| e^{-j\omega\tau_T} \right| = 1 = 0[dB] \\ \arg\left\{ e^{-j\omega\tau_T} \right\} = -\omega\tau_T \end{cases}$$

Le retard pur n'influence pas donc pas le gain du système $G(s)$. En revanche, avec la contribution de $\arg\left\{ e^{-j\omega\tau_T} \right\} = -\omega\tau_T$, il modifie la phase de manière linéaire[2], *i.e.* les harmoniques du signal d'entrée $u(t)$ sont déphasées d'un angle proportionnel à leur pulsation ω.

La figure 9 montre que le déphasage amené par un retard ne tend pas vers une valeur asymptotique (par exemple -90 °, -270°), mais croit indéfiniment avec la pulsation ω.
L'indicateur de stabilité représenté par la marge de phase est calculé comme suit :

$$\varphi m(\omega) = 180° - \arg\{G(j\omega)\} \qquad (4)$$

Pour ω tel que $A(\omega) = |G(j\omega)| = 1$.
Les résultats représentés par les figures 9 & 10 montrent bien que l'influence du délai induit par le réseau dégrade les performances du système et voire même le déstabilise. Cependant dans un système contrôlé en réseau on ne peut pas parler du réseau comme une seule entité qui influe sur la stabilité du système et surtout on ne peut pas réduire le réseau de communication à un élément (par exemple buffer) qui retarde ou élimine l'information. Cette abstraction faite du réseau est souvent rencontrée dans des travaux effectués par les spécialistes en automatique [Richard 2003, Lelevé 2000].

Sachant qu'un système contrôlé en réseau est un système de commande distribué, d'autres facteurs influent sur les performances d'un système commandé. D'ailleurs, comme le soulignent Juanole et Mouney dans leurs travaux [Juanole and Mouney 2005, 2006, 2007], les mécanismes de base pour la fourniture de la qualité de service fournie par le système informatique distribué appliqué à un système de commande en boucle fermée (voir chapitre 1 section 1.1) influent sur les performances des applications de contrôle-commande. Les mécanismes de la qualité de service considérés sont : l'ordonnancement de tâches, de trames, et les protocoles utilisés pour accéder au médium partagé (réseau).

Dans les paragraphes qui vont suivre nous allons montrer à l'aide d'un exemple l'influence de ces différents mécanismes sur la performance du système commandé. Les indices de performance du système seront la marge de phase et une fonction coût ISE (Integrated Square Error) ou encore appelée critère de Hall-Sartorius. Ce critère est utilisé habituellement pour informer le concepteur de la qualité du régulateur utilisé. Dans cette section nous l'utiliserons comme indice de performance du système pour montrer l'influence des mécanismes de la qualité de service sur le système commandé.
L'ISE est calculé comme suit :

[2] Notons que l'échelle logarithmique employée pour représenter la pulsation sur les diagrammes de Bode tend à masquer la linéarité du déphasage

$$ISE = \int_0^{Treg} e^2(t)dt \tag{5}$$

Avec T_{reg} appelé durée de réglage qui est la durée mesurée entre l'instant d'application de la consigne $u(t)$ et l'instant où la grandeur réglée $y(t)$ ne s'écarte plus de la bande de tolérance de ±5% de sa valeur finale y_∞.

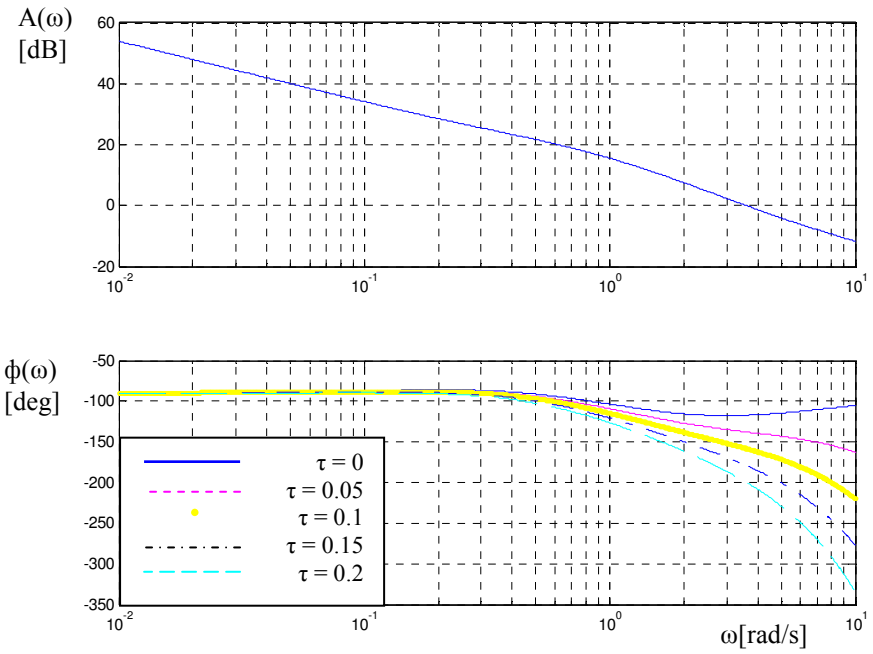


Figure 9. Représentation de Bode du système et l'influence des retards sur la phase avec $\tau_T = \tau_{CA} + \tau_{SC}$

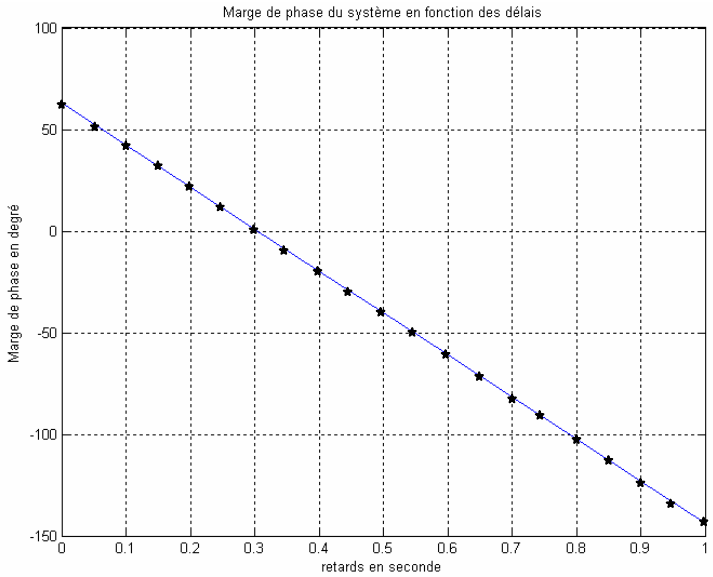


Figure 10. Marge de phase en fonction des délais induits par le réseau

4. Influence de l'ordonnancement de tâches

Avant d'aborder cette section la définition de quelques notions de base pour l'ordonnancement de tâches est nécessaire.

4.1 Caractéristique d'une tâche

La tâche est l'entité de base de l'ordonnancement temps réel et elle est périodique ou apériodique, à contrainte stricte ou relative. Aussi, un modèle canonique défini, dans Cottet et *al*. [Cottet et *al*. 2000, Albertos 2006] regroupe les principaux paramètres temporels qui peuvent caractériser les tâches temps réel et guider l'ordonnancement temps réel.
Dans ce modèle, une tâche est définie par des paramètres chronologiques qui dénotent des délais et des paramètres chronométriques qui indiquent des dates. Le modèle comprend les paramètres de base suivants:
- *r*, sa date de réveil, c'est-à-dire le moment de déclenchement de la première requête d'exécution appelé aussi phase,
- *C*, sa durée d'exécution maximale quand elle dispose du processeur pour elle seule, appelée aussi WCET (Worst Case Execution Time)
- *D*, son délai critique (deadline), c'est-à-dire le délai maximum acceptable pour son exécution,
- *P*, sa période lorsqu'il s'agit d'une tâche périodique,
- pour une tâche à contraintes strictes, le délai permet de calculer l'échéance $d = r + D$, c'est-à-dire la date dont le dépassement entraîne une faute temporelle.

Quand la tâche est apériodique, le paramètre *P* n'est pas donné. Lorsque la tâche est périodique, les quatre paramètres sont présents. Chaque réveil déclenche une requête de la tâche périodique. La qualité de l'ordonnancement sera d'autant meilleure que les valeurs de ces paramètres seront exactes. Aussi la détermination de ces paramètres est souvent un aspect important de l'analyse et de la conception d'une application temps réel.

En particulier il faut savoir si on néglige ou non les temps de commutation des tâches, les durées d'utilisation des primitives du système d'exploitation, les durées de prises en comptes des interruptions et les durées d'exécution de l'ordonnanceur.

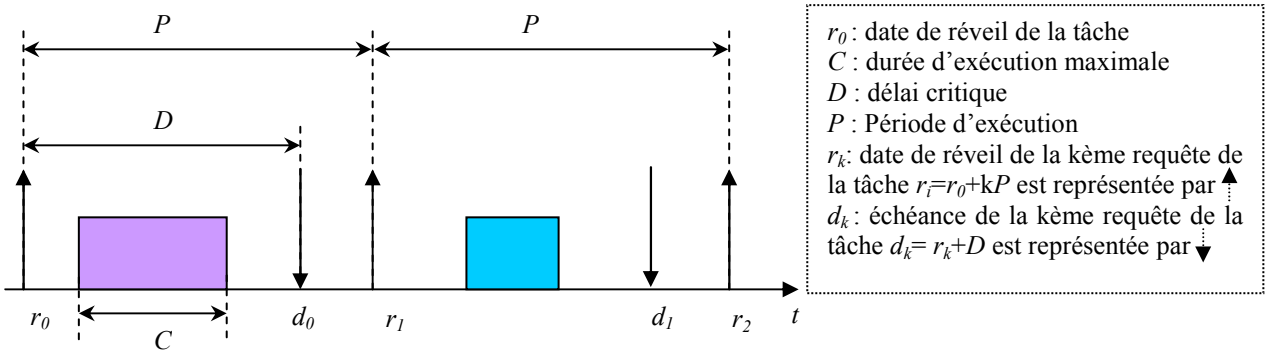


Figure 11. Modèle canonique des tâches [Cottet et *al*. 2000]

4. 2 Influence de l'ordonnancement de tâches

A présent, pour montrer l'influence de l'ordonnancement de tâches sur les performances d'un système commandé en boucle fermée, un processeur représenté par un ordinateur sur lequel trois tâches sont implémentées est simulé avec le simulateur TrueTime [Henriksson et *al*. 2002, 2003]. La marge de phase et l'ISE sont calculées pour chaque système. Les tâches

implémentées ont pour objectif de calculer la commande de trois systèmes (figure 12). Nous avons choisi volontairement de commander trois systèmes identiques pour faciliter la comparaison entre les signaux de sortie suivant la priorité des tâches. Bien entendu, la tâche de commande est calculée pour chaque système. Chaque système est représenté par la fonction de transfert de l'équation 1.

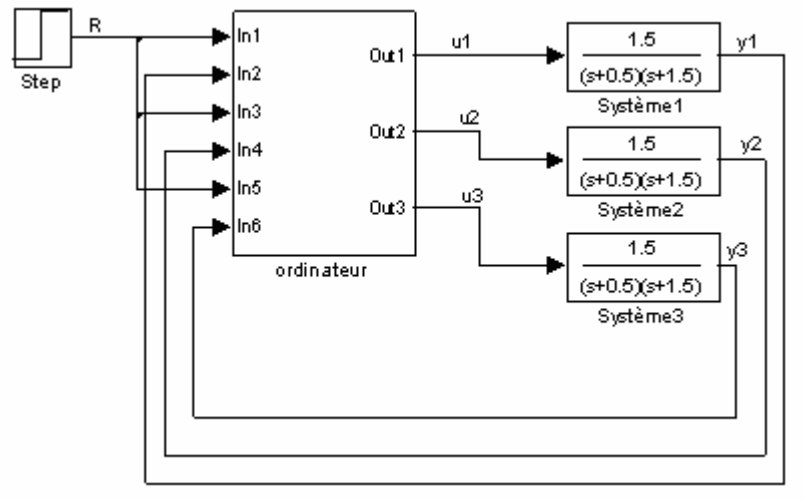


Figure 12. Commande de trois systèmes en boucles fermées

Le type d'ordonnanceur utilisé est l'ordonnanceur à priorité fixe, dont le fonctionnement se base sur des priorités fixées a priori et le processeur exécute les tâches selon l'ordre de priorité affecté. Les tâches sont périodiques et leurs paramètres de bases sont résumés dans le tableau ci-dessous :

Paramètres	Système 1 (sys1)	Système 2 (sys2)	Système 3 (sys3)
r_0 (offset)	0	0	0
C	0.035 s	0.035 s	0.035 s
$P = D$	0.1s	0.1s	0.1s

Tableau 1. Paramètres des tâches de commande

Le système de la figure 12 sera simulé pour deux ordres de priorité présentés dans le tableau suivant:

Ordre de priorité	Sys1	Sys2	Sys3
Scenario 1	1 (haute priorité)	2(moyenne priorité)	3(basse priorité)
Scenario 2	3	2	1

Tableau 2. Ordre des priorités affecté pour l'ordonnancement à priorité fixe.

Sachant que la tâche de commande à laquelle est affectée la plus grande priorité est représentée par le plus petit nombre, et celle de priorité inférieure par le plus grand nombre.

La figure 13, montre bien que plus le système possède une priorité faible plus la performance du système se détériore jusqu'à l'instabilité. La marge de phase, quant à elle diminue suivant la diminution de la priorité (figure 14. a). Une autre constatation qui découle de la figure 14.b est que plus la priorité affectée à *la tâche de commande* est petite plus le délai augmente entre

l'instant où la commande serait reçue (en théorie), et l'instant où elle est reçue par le système. De ce fait, la marge de phase diminue.

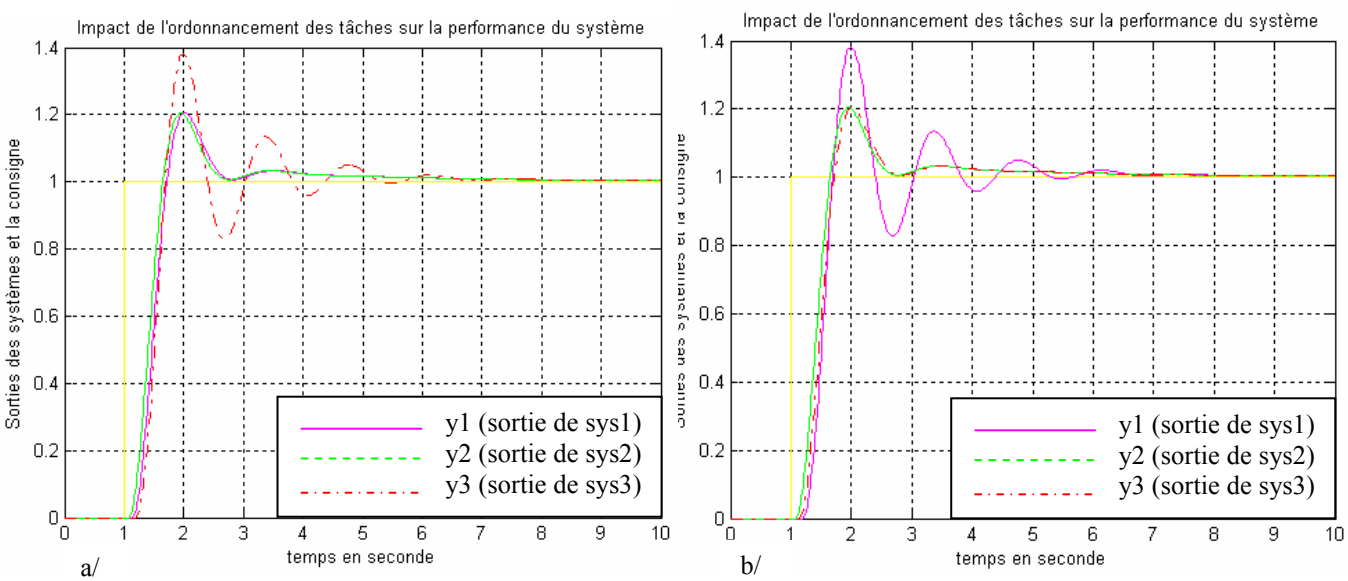


Figure 13. Sortie des différents systèmes et l'influence de la priorité des tâches de commande sur la sortie du système : a/scenario 1, b/scenario 2.

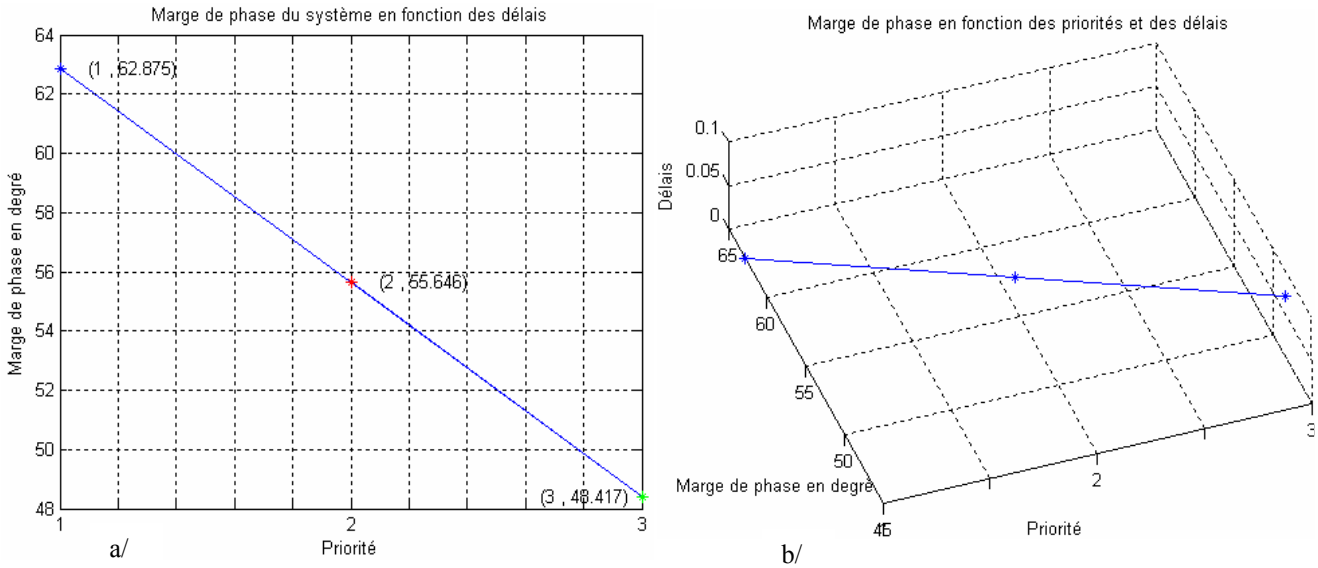


Figure 14. Influence des priorités sur la marge de phase pour les trois systèmes : a/ marge de phase Vs priorité, b/ marge de phase Vs priorité Vs délais

5. Influence de l'ordonnancement de trame

L'ordonnancement de trames dans les réseaux fait appel aux besoins en qualité de service. Comme pour l'ordonnancement de tâches, les trames possèdent leurs propres caractéristiques. Et suivant le réseau utilisé, elles possèdent une longueur constante (exp : ATM) ou variable (exp : Ethernet).

Dans cette section nous nous aiderons d'un exemple de système contrôlé en réseau, où le réseau utilisé est CAN (Controller Area Network). Avant d'aborder cette section, quelques définitions et règles de fonctionnement du réseau CAN s'imposent.

5.1 Le réseau CAN[3] [Paret 1999]

Le bus CAN (Controller Area Network) est né du besoin de trouver une solution de communication série dans les véhicules automobiles, qui ont tendance à intégrer de plus en plus de commandes électroniques. Jusqu'à maintenant, tous les organes de commandes des véhicules échangeaient les données par l'intermédiaire de lignes dédiées. L'augmentation du nombre d'organe embarqué a contraint les équipementiers automobiles à développer une nouvelle architecture à base de bus réseaux. Le GIE Renault-PSA avec les partenaires comme Sagem, Valeo et d'autres ont développé le bus VAN (Vehicule Area Network), les puces contrôleurs sont fabriquées par MHS, SGS, TI .... En Allemangne, Bosch a développé, au milieu des années 80, le bus CAN ou "Controller Area Network" qui a fait l'objet d'une normalisation ISO 11898.

Le protocole CAN est basé sur le principe de diffusion générale : lors d'une transmission, aucune station n'est adressée en particulier. Mais, le contenu de chaque message est explicité par une identification reçue de façon univoque par tous les abonnés. Grâce à cet identificateur, les stations, qui sont en permanence à l'écoute du réseau, reconnaissent et traitent les messages qui les concernent et ignorent les autres.

L'identificateur indique aussi la priorité du message, qui détermine l'assignation du bus lorsque plusieurs stations émettrices sont en concurrences. Par ailleurs, il existe deux types de format de trames (trame standard, trame étendue), les deux formats diffèrent l'un de l'autre par l'identificateur : identificateur de 11 bits pour les trames standards, et identificateur 29 bits pour les trames étendues. Chaque trame peut contenir jusqu'à 8 octets de données, ce qui correspond par exemple à l'état de 64 capteurs.

Le bus CAN est basé sur le mode de fonctionnement multi maîtres. Lorsque le bus est libre, chaque nœud peut décider d'envoyer un message. Aussi un problème d'arbitrage résulte de ce mode de fonctionnement. Si deux nœud ou plus tentent d'émettre en même temps un message sur le bus libre, ceci donne lieu à un conflit d'accès.
Pour palier ce problème, le procédé d'attribution du bus est basé sur le principe de l'"arbitrage bit à bit", selon lequel les noeuds en compétition, comparent bit à bit l'identificateur de leur message avec celui des messages concurrents. Les stations de priorité moins élevée perdront la compétition face à celle qui a la priorité la plus élevée.

Les stations sont câblées sur le bus par le principe du "OU câblé". En cas de conflit (c'est à dire d'émission simultanée), la valeur 0 écrase la valeur 1. On nomme donc l'"état dominant" l'état logique 0, et l'"état récessif" l'état logique 1. Lors de l'arbitrage bit à bit, dès qu'une station émettrice se trouve en état récessif et détecte un état dominant, elle perd la compétition et arrête d'émettre. Tous les perdants deviennent automatiquement des récepteurs du message, et ne tentent à nouveau d'émettre que lorsque le bus se libère. Cet arbitrage est qualifié de CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance)[4], il est aussi appelé (CSMA/AMP) CSMA/Arbitration on Message Priority [Hartman 2004, Andersson et *al*. 2005, Ohlin et *al*. 2006, Ohlin et *al*. 2007, Lian et *al*. 2001]. D'autres auteurs [Juanole et *al*. 2005] l'appellent CSMA/CD (collision Detection). Mais nous préférons réserver cette appellation au protocole Ethernet, nous choisirons CSMA/AMP qui à notre sens est explicite quand à la façon de résoudre les conflits dans CAN.

---

[3] http://www.can-cia.de
[4] http://uuu.enseirb.fr/~kadionik/formation/canbus/canbus_enseirb.pdf

5.2 Système étudié

Dans cette section le système représenté par l'équation 1 est distribué sur un bus CAN (figure 15.). Ce réseau possède un débit de 1Mbit/s. L'entête de la trame CAN compte 47 bits et le champ de données varie de 0 à 64 bits. Un autre ordinateur partage le même réseau que le système commandé dont le but principal est de générer du trafic supplémentaire externe à l'application considérée. L'application principale est composée de trois processeurs représentés par des ordinateurs :

- l'ordinateur 1 (commande) : implante la tâche de commande, qu'il calcule via un algorithme de commande (ici un contrôleur PID) tous les 1 ms et envoie les valeurs calculées à l'actionneur.
- l'ordinateur 2(actionneur) : implante la tâche de l'actionneur,
- l'ordinateur 3(capteur) : implante la tâche de capture de la sortie du système c'est-à-dire qu'il lit la sortie toutes les 0.001s.

La tâche du contrôleur et de l'actionneur sont guidées par les événements (Event-triggered) : le contrôleur et l'actionneur attendent la réception des échantillons avant de calculer et d'envoyer un message pour l'un et avant d'activer le système commandé pour l'autre.
La tâche du capteur est guidée par le temps (Time-triggered) : Le capteur lit périodiquement (1 ms) la sortie, basée sur une horloge.
Puisque le réseau a un débit de 1 Mbit/s, la durée d'un bit est de 1µs. La donnée incluse dans la trame de la commande et celle du capteur est de 3 octets (24 bits).

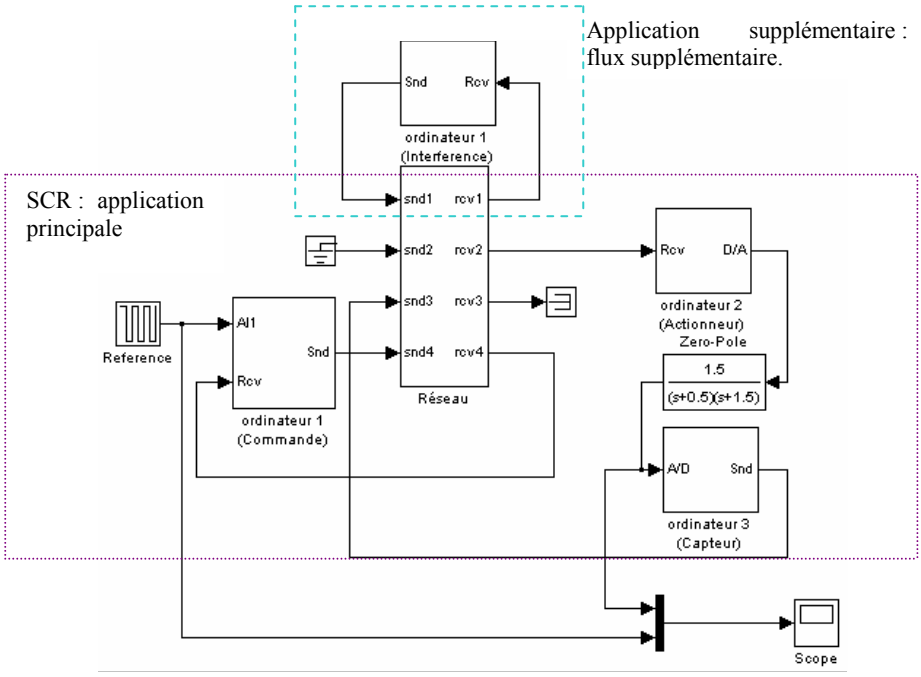


Figure 15. Système commandé distribué sur un réseau CAN

Cette valeur est calculée à partir de la longueur de la trame CAN : $L = 47 + 8n + \left\lfloor \frac{34 + 8n - 1}{4} \right\rfloor$[5][Navet 1999].
Aussi, cette valeur tient compte du fait que TrueTime ne considère que des longueurs de trames multiples de 8 bits. Donc la durée de la trame de commande (et du capteur) est de 80 µs.
Pour la trame de données de l'application externe nous choisirons une longueur de 128 bits (8 octets pour le champ de données). Cela donne une durée qui vaut 128 µs. La période de production de ces trames est 1 ms. L'application principale de la figure 15 sera d'abord simulée sans l'application supplémentaire dans deux cas de figure (tableau 3) :

Scenarii	Contrôleur	Capteur
Scenario 1	1 (haute priorité)	2 (basse priorité)
Scenario 2	2 (basse priorité)	1 (haute priorité)

Tableau 3. Scenarii étudiés

L'intérêt de cette simulation est de montrer l'influence du réseau sur le système sur sa marge de phase et son ISE.
Dans ce système, le retard induit par le réseau est lié à la transmission série de trames, donc le retard est dû à la transmission des trames du contrôleur vers l'actionneur τ_{ca}, et le retard dû à la transmission des trames du capteur au contrôleur τ_{sc}. Dans notre étude nous considérons que l'influence du bloqueur d'ordre zéro est négligeable[6] .

Les retards τ_{ca}, τ_{sc} représentent la durée des trames dont la valeur est de l'ordre de 64 µs pour chaque retard. Le retard total vaut 128 µs ($\tau_{ca} + \tau_{sc}$ =64 µs +64 µs).
Ce retard montré sur la figure 17 influe sur la marge de phase même si dans ce cas elle est infime puisque le système est simulé dans le cas idéal.
On remarque aussi que, quelle que soit la priorité associée au contrôleur ou au capteur selon le scénario 1 ou 2, l'influence du réseau sur la performance du système et donc sur sa marge de phase est la même. Ceci s'explique par le faible taux de requêtes utilisateur sur le réseau CAN. Il faut noter que le taux de requêtes utilisateur est la somme de la longueur de trame de chaque flux divisé par sa période de production :

$\sum_{k=1}^{i} \frac{L_k}{T_k}$ sachant que L_k et T_k sont respectivement la longueur de la trame et la période associées au flux *i*. Dans le cas ou le système de la figure 15 est considéré sans l'influence de l'application externe, le taux de requête utilisateur est égal 128. 10^{-5} donc très inférieure à 1, ce qui traduit une influence minime du réseau sur la performance du système puisqu'il a la possibilité de répondre aux requêtes des utilisateurs.

[5] 47bit est l'entête de la trame, 8n : est le champ de donnée, $\left\lfloor \frac{34 + 8n - 1}{4} \right\rfloor$: est dû au bit de transparence (bit stuffing).
[6] D'autres auteurs l'ont pris en compte dans leur étude [Juanole and Mouney 2007]

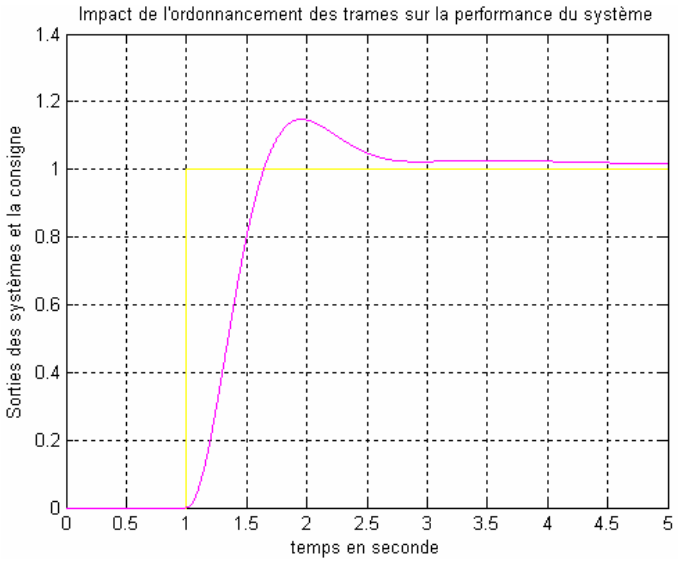


Figure 16. Sortie de l'application distribuée sur le réseau CAN quelle que soit la priorité du contrôleur et du capteur.

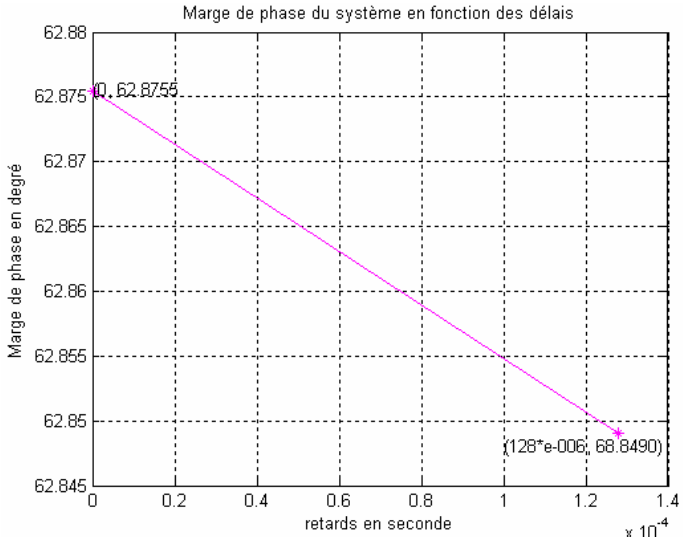


Figure 17. Marge de phase du système Vs délais avec et sans réseau.

L'ISE calculé dans le système contrôlé en réseau suivant les deux scénarii est égal dans les deux cas à 0.2048. Cela traduit que le signal de sortie converge vers la valeur désirée (la référence).

Dans une deuxième phase, le système de la figure 15 va être évalué en ajoutant l'application supplémentaire générant du flux dit externe qui vient surcharger le réseau. Ce système sera donc étudié pour les différentes configurations de priorité, résumées dans le tableau ci-contre :

Priorités	Contrôleur	Capteur	Interférence
Scénario 1	1 (priorité haute)	2	3
Scénario 2	2 (priorité intermédiaire)	1	3
Scénario 3	3 (priorité basse)	2	1
Scénario 4	2	3	1
Scénario 5	1	3	2
Scénario 6	3	1	2

Tableau 4. Scenarii étudiés

Rappelons que le flux externe est périodique et la taille de ces trames est 128 bits donc leur durée est 128 µs. Rappelons aussi que nous utilisons le simulateur TrueTime pour les simulations et que la gestion des messages dans un réseau modélisé dans la version 1.3 est basée sur une file d'attente FIFO (First In First Out). La technique de la mémoire FIFO avec écrasement a été ajoutée sur l'initiative de [Juanole et *al*. 2005], qui a donné lieu à une nouvelle version amélioré (1-4). Bien évidemment, il y a eu d'autres améliorations et au moment où cette thèse est écrite la version 1.5 est disponible. C'est cette dernière que nous utilisons pour nos simulations. Pour mener à bien cette étude, deux cas de figures ont été étudiés. Il faut noter que beaucoup de travaux dans le domaine des SCRs utilisent pour leur modélisation TrueTime, nous citons par exemple [Tanwani et *al*. 2006]. Les auteurs ont modélisé un drone implanté sur un réseau CAN.

5. 2. 1 Cas n°1

Dans ce cas le flux externe possède une période de 128 µs, le taux d'utilisation moyen du réseau CAN est de 1 signifiant que le réseau est très sollicité.

Scenarii 1 & 2 :

Nous remarquons que dans ce cas, le système fonctionne parfaitement bien puisque la commande et le capteur ont des priorités supérieures à celle de l'interférence.

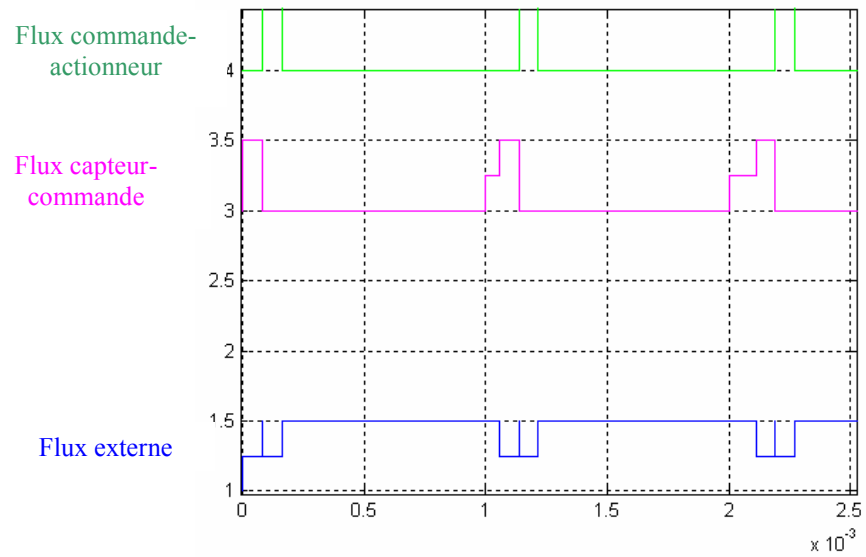


Figure 18. Ordonnancement des trames dans le réseau avec les différents niveaux représentant l'état du réseau. Niveau haut : envoie du message, niveau moyen : attente, niveau bas : inoccupé.

Le diagramme des flux représenté dans la figure 18, montre qu'à l'instant 0 l'application externe et le capteur sollicitent en même temps le bus CAN. Cependant, c'est le capteur qui envoie une trame de durée 80 µs puisque sa priorité est la plus haute.
L'application est mise en attente durant ce moment jusqu'à ce que le bus soit libre (niveau médian). Á 80 µs la commande[7] reçoit la trame envoyée par le capteur et calcule aussitôt la

[7] Rappel : la commande est guidée par les événements i.e. dès que le capteur envoie une trame la commande qui la reçois calcule aussitôt la nouvelle valeur de la commande.

valeur de commande[8]. A cet instant, la commande et l'application externe (qui était en attente) sollicitent le bus et puisque le contrôleur a une priorité supérieure, il envoie sa trame. La trame de l'application externe est donc toujours en attente jusqu'à 160 µs (80 µs+80 µs).
Á 160 µs l'application externe peut enfin envoyer sa trame dont la durée est de 128 µs. Á 1ms, le capteur émet sa trame. Mais il est mis en attente puisque l'application externe n'a pas fini d'émettre. Á 288 µs (80 µs+ 80 µs + 128 µs) le capteur envoie sa trame à 368 µs (288 µs + 80 µs). Le contrôleur récupère la donnée aussitôt et calcule la nouvelle valeur de la commande et l'envoie vers l'actionneur.
Notons bien qu'à 128 µs l'application externe produit une nouvelle donnée mais elle a été mise en attente puisque le bus était occupé dans un premier temps par la trame venant du capteur puis dans un second temps par une trame venant du contrôleur.

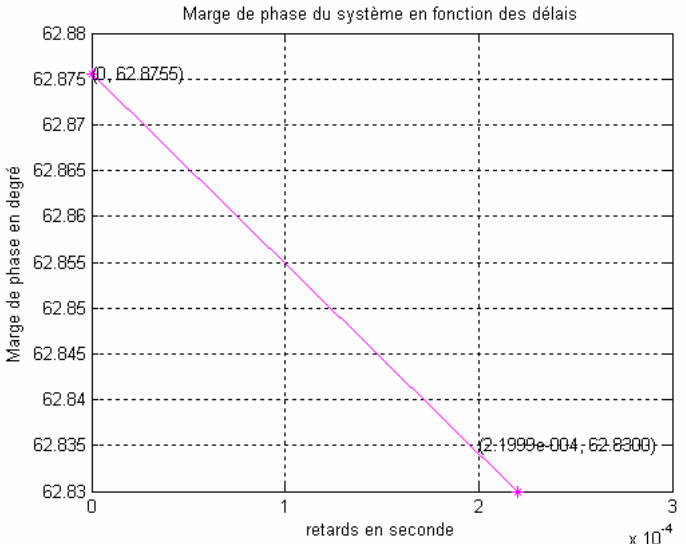


Figure 19. Marge de phase en fonction des délais induits par le réseau.

Dans le cas des deux scenarii 1 & 2 la marge de phase diminue preuve que le réseau a une influence sur la performance du système. Cependant, cette influence n'a pas de conséquence importante puisque le délai induit est de l'ordre de 2.1999 10^{-4} s.

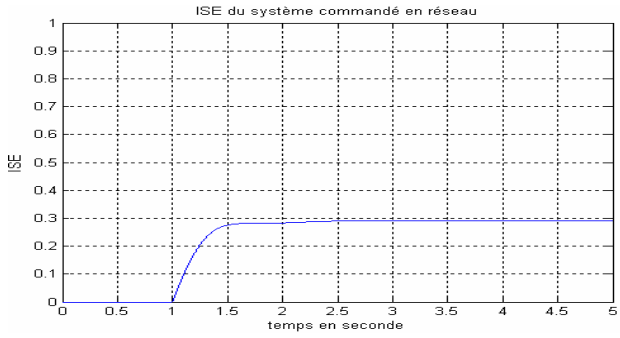


Figure 20. ISE Vs Scenarii 1 & 2.

L'ISE calculée montre que la sortie converge bien vers la consigne et il est de l'ordre 0.2964.

[8] Ici le temps de calcul est considéré négligeable.

Scenarii 3, 4, 5 & 6 :

Dans le cas contraire c'est-à-dire le cas où le flux externe possède une priorité supérieure (scenarii 3 & 4), le système est complètement instable car le bus est toujours occupé par le flux externe. Donc, il n'est pas possible d'implanter un système de commande dans ce cas de figure, et ceci est valable pour tous les autres scenarii (5 & 6).
Comme le montre la figure 21, le trafic extérieur ayant une priorité supérieure, il accapare le réseau, laissant le capteur en état d'attente permanent. En effet, à l'instant 0 le capteur et l'application externe envoient leur trame, puisque l'application externe possède une priorité supérieure il envoie sa trame d'une durée de 128 µs, à cet instant il envoie une autre trame (la période est de 128 µs) et ainsi de suite.

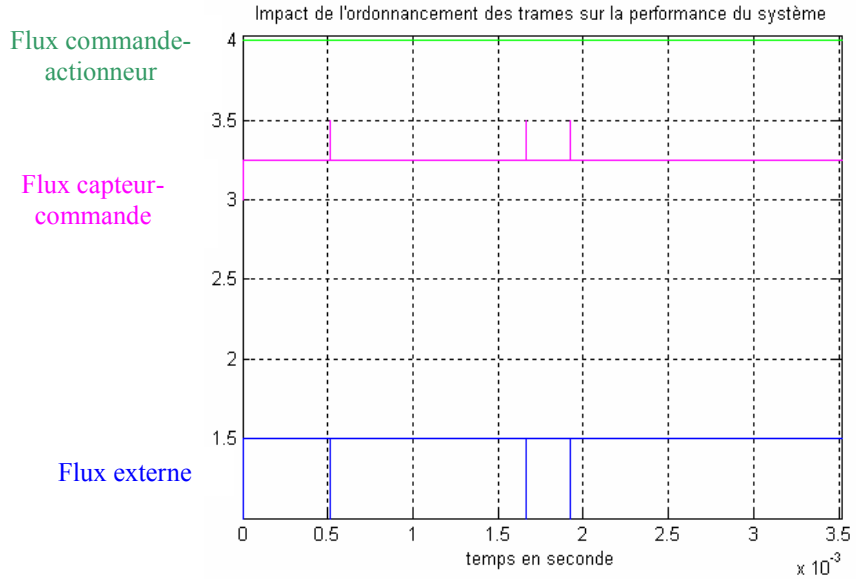


Figure 21. Ordonnancement des trames dans le réseau : scenario 3.

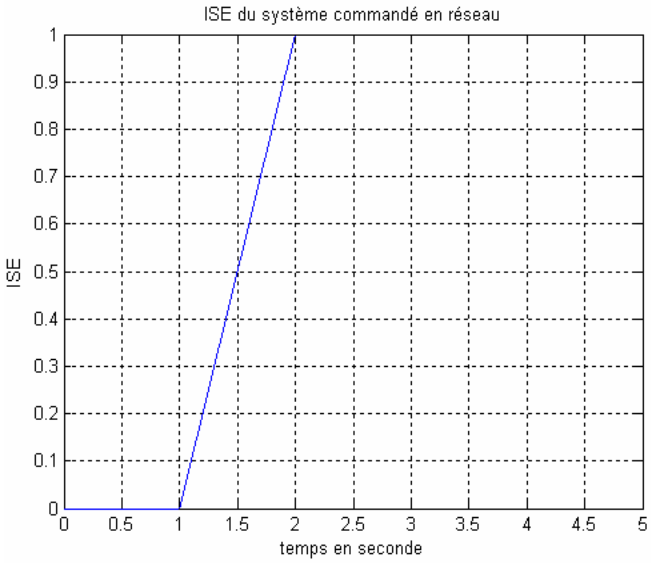


Figure 22. ISE du système contrôlé en réseau: scenario 3.

5. 2. 2 Cas n °2

Dans ce cas le flux externe possède une période de 150 µs, le taux d'utilisation moyen du bus CAN est de 0.85 < 1.

Scenario 1 & 2 :

Les résultats de simulation obtenus dans le scenario 1 sont semblables à ceux du scenario 2. A l'instant 0, l'application externe et le capteur envoient leur trame simultanément mais seule la trame venant du capteur est transmise puisque de priorité supérieure (durée de la trame 80 µs) et la trame de l'application externe est mise en attente.
A 80 µs la trame du contrôleur est en compétition avec celle de l'application externe, cependant c'est la trame envoyée par le contrôleur qui est transmise puisque le contrôleur est de priorité supérieure. A 160 µs l'application externe peut enfin envoyer sa trame (160 µs + 128 µs = 288 µs). On remarquera qu'entre 288 µs et 1 ms (période de production du capteur), l'application extérieure enverra six trames. De ce fait, le capteur qui devait envoyer à 1 ms une trame attendra que le réseau finisse de transmettre la trame de l'application extérieure (1.056 ms). Ensuite à 1.056 ms le capteur envoie une trame de nouveau.

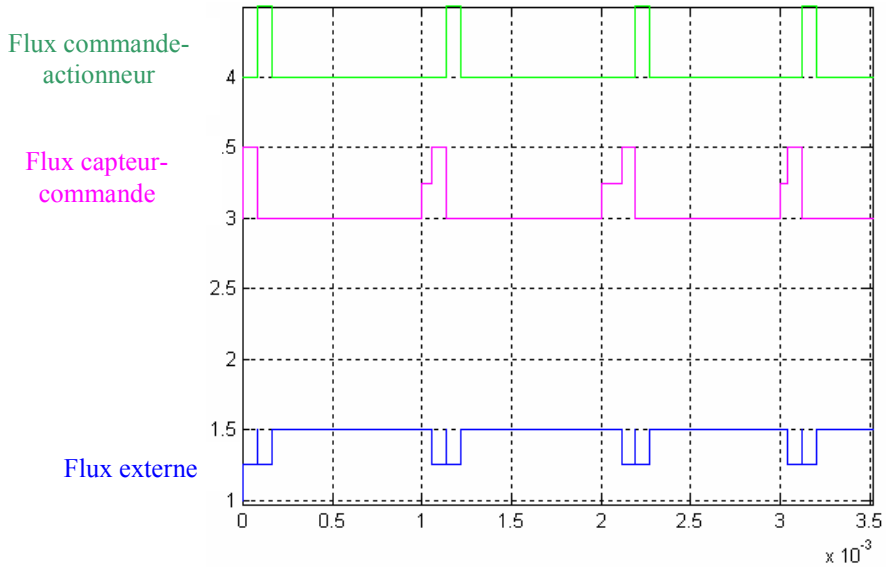


Figure 23. Ordonnancement des trames dans le réseau : scenario 1 & 2.

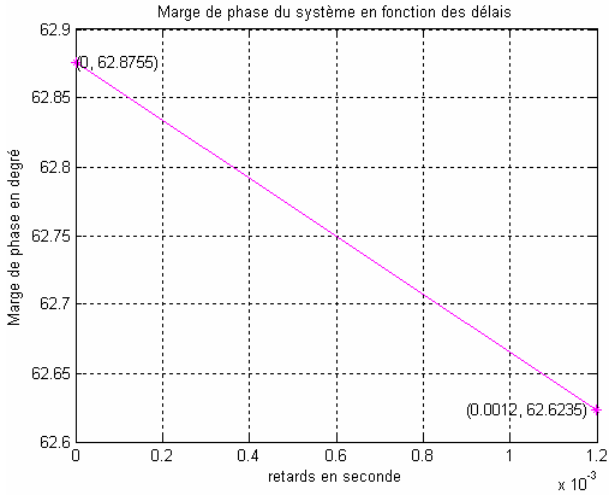


Figure 24. Marge de phase en fonction des délais induits par le réseau

Le retard induit par le réseau est de l'ordre de 0.0012 s ce qui a pour conséquence de réduire la marge de phase (voir figure 24), cependant cela n'a aucune conséquence sur la stabilité du système.

Scenario 3 & 6:

Ces deux scenarii sont rassemblés dans cette section car leur comportement est relativement le même, le capteur possède une priorité supérieure à celle du contrôleur.
A l'instant 0 (figure 25), l'application externe et le capteur envoient au même moment des trames. C'est donc l'application externe qui envoie sa trame de durée 128 µs puisque de priorité supérieure. A 128 µs, le capteur envoie sa trame sachant qu'à 150 µs la trame de l'application externe est mise en attente le temps que le réseau termine la transmission de la trame envoyée par le capteur. Une fois la trame venant du capteur transmise, le contrôleur calcule la commande et l'envoie aussitôt. Cependant il est en conflit avec l'application externe qui envoie sa trame (à 128 µs + 80 µs = 208 µs) en premier car de priorité supérieure (période 150 µs). Puis à (208 µs + 128 µs = 336 µs), c'est toujours l'application externe qui émet pour les mêmes raisons (envoie de trame à 200 µs qui était en attente le temps que le réseau termine d'envoyer la trame de la même application). Ainsi, la trame de la commande reste toujours en attente. Cette situation perdure jusqu'à 592 µs. A cet instant le réseau envoie la trame du contrôleur puisqu'il est libre entre 592 µs et 600 µs. Ce comportement se répète jusqu'à 5.616 ms (voir figure 26). Le contrôleur est mis en attente à cause des trames de l'application externe qui doivent être envoyées en premier. A 6 ms, le capteur produit une trame cependant le réseau n'a pas encore envoyé la trame de commande qui est le résultat d'un calcul basé sur la valeur de la trame du capteur reçue à 5.616 ms. Donc, la trame venant du capteur est émise à 6.128 ms et est reçue à 6.208 ms (6.128 ms + 0.080 ms).
La commande n'est envoyée qu'à 5.616 ms. Notons qu'à partir de 5.616 ms, une trame sur deux du contrôleur est transmise.
Cette combinatoire de priorité a pour conséquence un retard de 0.4181 s qui rétrécit la marge de phase la rendant négative -23.4741 °.
Une marge de phase négative témoigne de l'instabilité du système et un ISE (figure 28) qui montre la divergence de la sortie par rapport à la référence puisque d'une valeur de 3.5906.

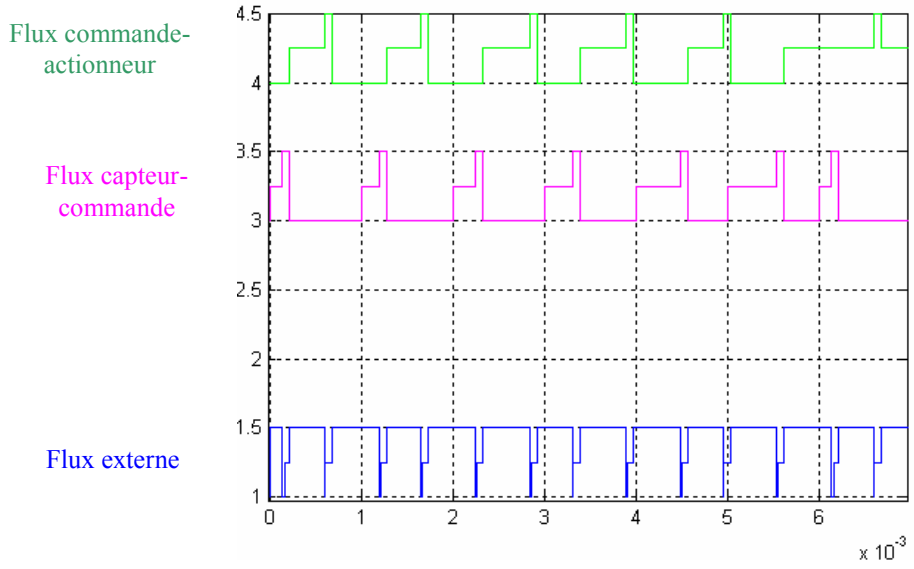


Figure 25. Ordonnancement des trames dans le réseau : scenario 3.

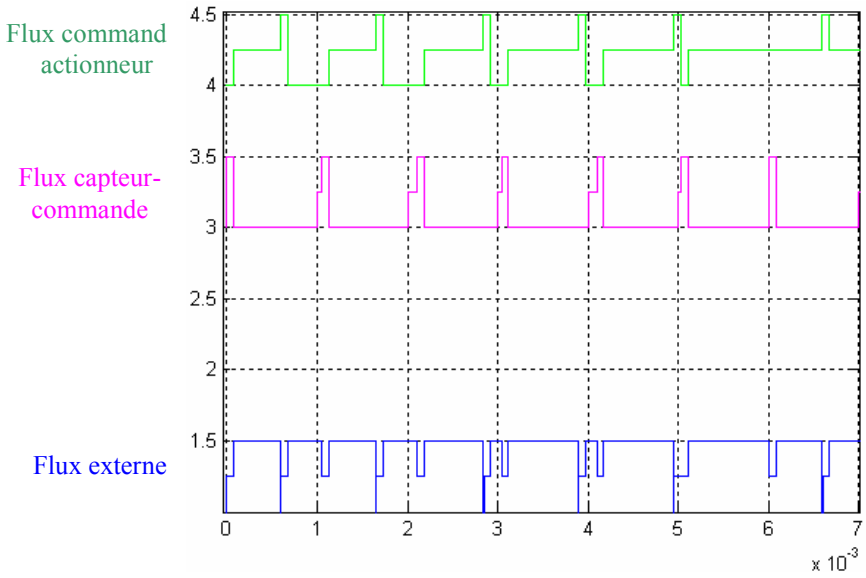


Figure 26. Ordonnancement des trames dans le réseau : scenario 6.

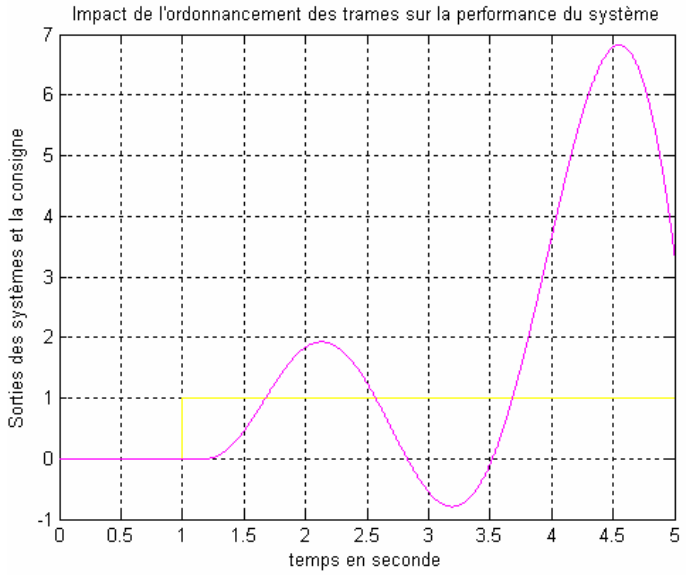


Figure 27. Sortie du système en fonction du délai induit par l'ordonnancement des trames selon le scenario 3 & 6

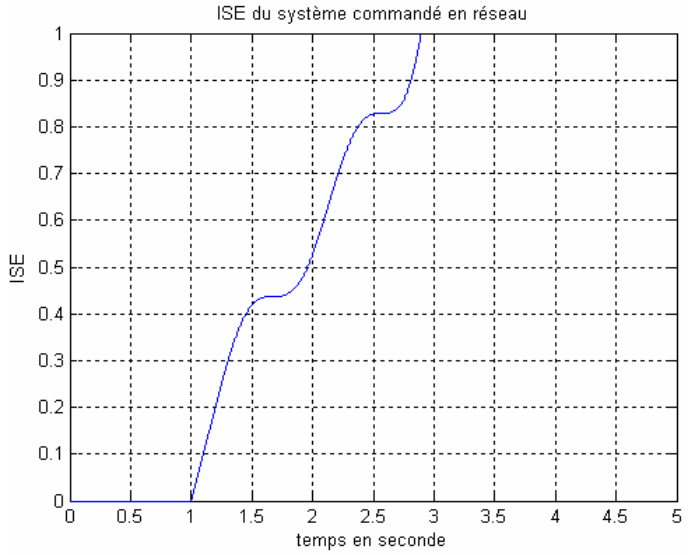


Figure 28. ISE du système : Scenarii 3 & 6

Scenarii 4 & 5 :

Dans ces scenarii le contrôleur possède une priorité supérieure à celle du capteur. Leur comportement est relativement le même. Notez qu'on retrouve les mêmes explications que précédemment jusqu'à 5.616 ms. A cet instant, puisque le contrôleur possède une priorité supérieure il ne subit pas le retard rencontré dans le scenario 3.

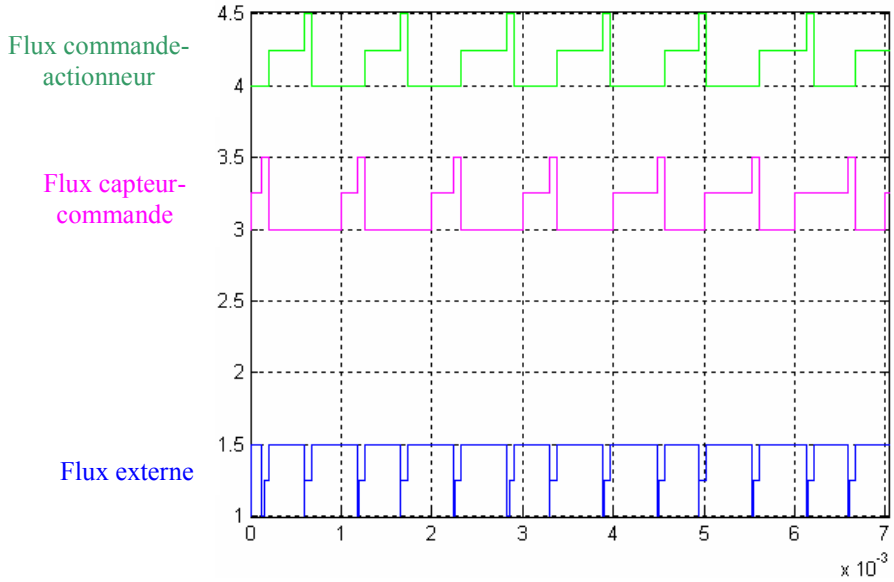


Figure 29. Ordonnancement de trame dans le réseau : Scenario 4.

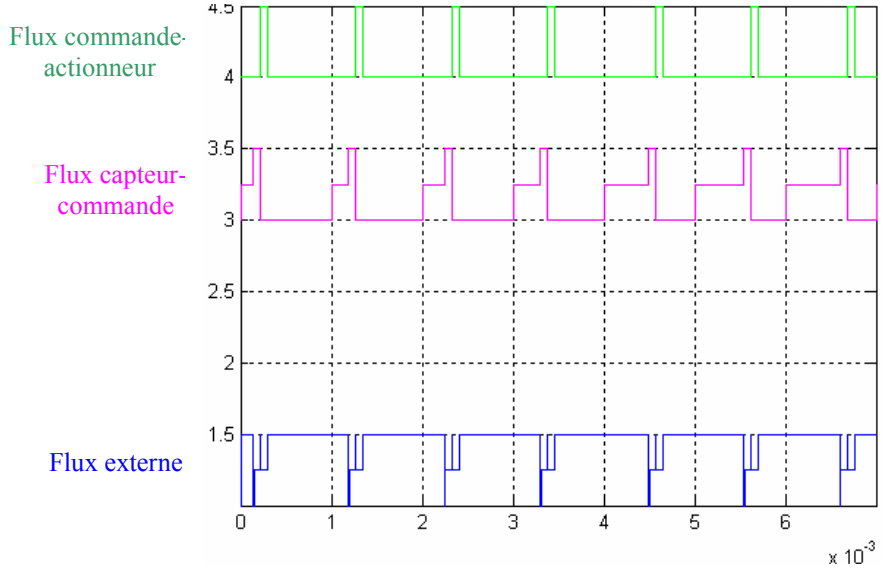


Figure 30. Ordonnancement de trame dans le réseau : Scenario 5.

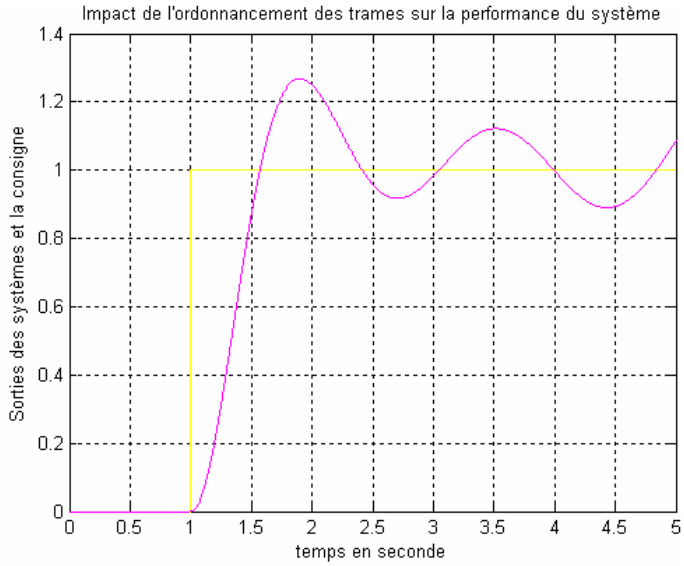


Figure 31. Sortie du système en fonction du délai induit par l'ordonnancement des trames selon le scenario 4 & 5

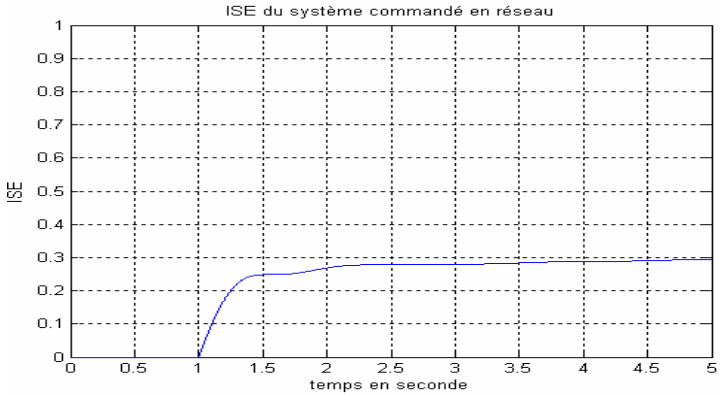


Figure 32. ISE du système: Scenarii 4 & 5.

La configuration de priorité du scenario 4 a pour conséquence un retard de 0.1930 s. Ce retard rétrécit la marge de phase 23.0109 ° rendant le système instable.
Pour le scenario 5 le retard est 0.2005 s et la marge de phase de 21.4720 °.

Dans ces quatre derniers scenarii nous remarquons l'importance d'avoir la priorité de la commande supérieure à celle du capteur. Ce même comportement pourrait être rencontré dans les scenarii 1 & 2, avec un système à temps de réponse plus court (voir l'exemple dans [Juanole and Mouney 2007]).

5.3 Comparaison de deux méthodes d'accès au médium basées sur différents procédés d'arbitrage :

Dans cette section nous choisirons deux types de méthodes d'accès au médium : CAN (*c.f.* paragraphe 1.2.1 page 12) et Ethernet (CSMA/CD) (*c.f.* le paragraphe 2.3 page 11). Les paramètres des réseaux sont présentés dans le tableau ci-dessous :

Paramètres	Ethernet	CAN
Débit (Mbit/s)	10	1
Durée d'un bit (bit time) (µs)	0.1	1
Taille max. du champ de donnée (Octet)	1500	8
Taille min. des messages (Octet)	72[9]	47/8[10]

Tableau 5. Paramètres des réseaux Ethernet et CAN.

Le comportement de CAN a été étudié précédemment et nous avons vu que le protocole CAN résout le conflit grâce au procédé d'arbitrage sur la priorité des messages.
Cette partie sera donc principalement, dédiée à la simulation du système précédent en utilisant le réseau Ethernet dont la méthode d'accès au médium est basée sur le CSMA/CD.
Les paramètres de simulations sont résumés dans le tableau suivant :

Cas de simulation	Période du flux d'interférence (µs)	Instant de début (offset) (s)	Taille de la donnée de l'interférence (Octet)	taille de la donnée du contrôleur et du capteur (Octet)
1ème cas	150	0	126	46
2er cas	150	0	1500	46

Tableau 6. Paramètres de simulations du réseau Ethernet

Comme pour le réseau CAN, la taille du champ de données pour les trames du contrôleur et du capteur est courte (46 Octets, donc la taille de la trame est de 64 Octets avec l'entête sans le préambule). Et pour la taille de données, le système est d'abord simulé dans le cas où la taille du champ de donnée est de 126 Octets, puis pour une taille maximale (1500 Octets).

5. 3. 1 Cas n°1

L'algorithme d'accès au médium d'Ethernet est le CSMA/CD. Dans ce cas de figure quand deux stations émettent en même temps, une collision est détectée puis elle attend un temps aléatoire au bout duquel elle tentera à nouveau d'émettre sa trame.
De ce fait le retard induit par le réseau est complètement aléatoire. On peut remarquer par exemple qu'à l'instant 0, le capteur et l'application externe ont tenté d'émettre en même temps, donc il y a eu collision alors les deux stations ont été mises en attente. A la deuxième tentative, c'est le capteur qui a gagné le réseau (figure 33).

Le retard induit par le réseau avec les paramètres du 2ème cas (voire tableau 6) est de l'ordre de 0.0895 s, ce qui rétrécit la marge de phase 44,3865°. Ceci cause un dépassement de 1.25 par rapport à 1 la valeur désirée (figure 34). L'ISE représentée par la figure 35, montre que la sortie du système finit par converger vers la valeur désirée.

[9] Le préambule et le bit délimiteur inclus
[10] L'entête d'une trame CAN est de 47 bit.

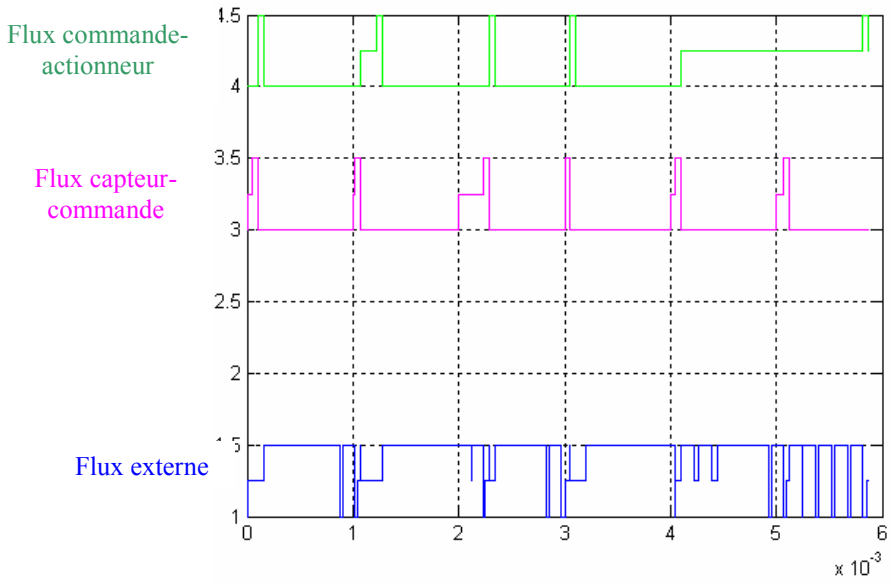


Figure 33. Ordonnancement de trames dans le réseau suivant le protocole CSMA/CD.

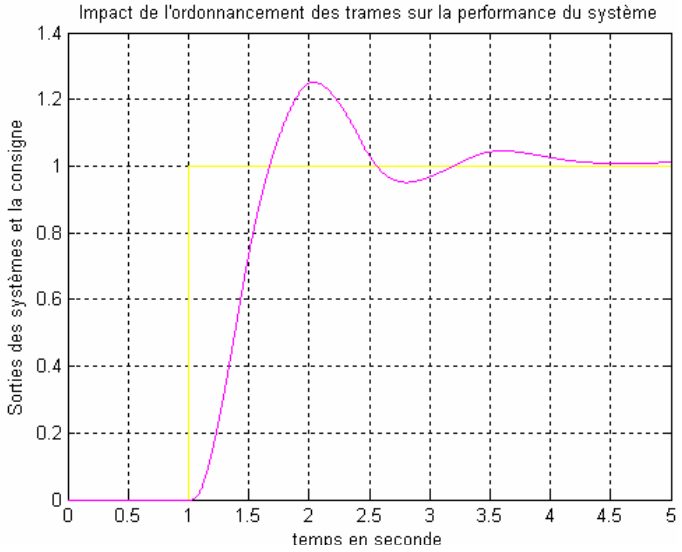


Figure 34. Sortie du NCS

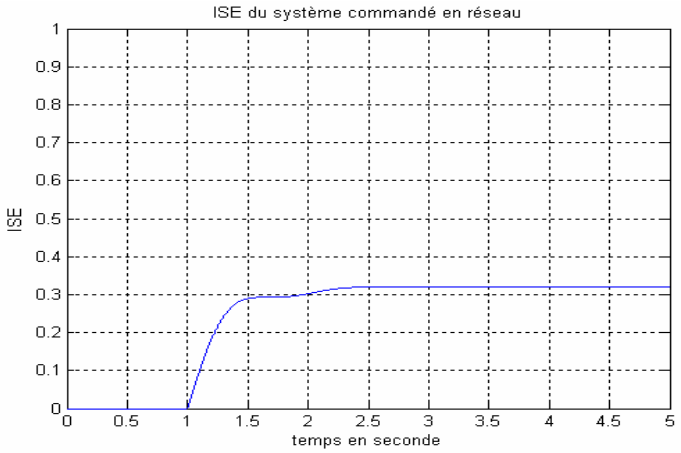


Figure 35. ISE du système.

Remarque
Ces mêmes simulations ont été effectuées avec TrueTime -1.3. Pour quelques scenarii notamment dans le cas où le réseau est saturé, il est presque impossible d'obtenir des résultats tant le temps de simulation est trop important. La cause d'un tel comportement est due au retard induit par la file FIFO utilisée dans la version -1.3. A partir de la version -1.4 il est possible d'utiliser la file d'attente FIFO avec écrasement. De ce fait, dans certain cas, il est préférable d'avoir des pertes (FIFO avec écrasements) que des retards (FIFO).

5.3. 2 Cas n°2

Dans le premier cas le système est complètement instable. Donc un système de commande en boucle fermée ne peut être implanté dans ces conditions.

6. Influence du type de protocole

On distingue deux types de protocoles : les protocoles sans contrôle de perte des messages, et les protocoles avec contrôle de perte des messages. L'utilisation de l'un ou l'autre protocole peut avoir un impact sur la stabilité du système. Un exemple intéressant a été étudié dans [Juanole and Mouney 2007, page 31-33] qui traite de ce paramètre.

7. Synthèse

L'influence de l'ordonnancement de tâches et de trames ainsi que du type de réseau sur la stabilité d'un NCS a été présentée. Comme le montre la figure 14 b. (page 14), plus la priorité de la tâche de commande diminue, plus le retard induit par ce phénomène augmente, et par conséquent la marge de phase rétrécit. Une forte diminution de la marge de phase cause l'instabilité du système commandé.

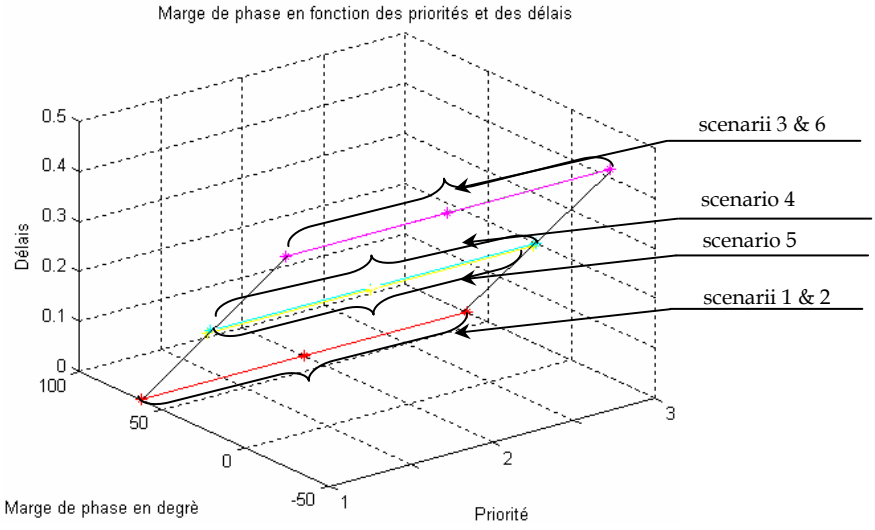


Figure 36. Marge de phase Vs priorité Vs délais

La figure 36 montre le comportement d'un système commandé suivant les différentes combinatoires de priorité. On peut remarquer que la marge de phase est plus étroite quand le capteur possède une plus grande priorité sur la commande rendant ainsi instable le système. Aussi, d'après le scenario 1 & 2 il est préférable d'associer une faible priorité au flux de l'application externe, pour avoir une bonne performance du système commandé.

Nous avons vu que différents paramètres de la qualité de service influent sur la performance d'un système commandé. La question posée à présent est comment compenser les retards de communication pour contrôler un système en réseau. Dans ce chapitre, nous allons exposer l'approche dite « control over network » qui propose d'adapter la commande relativement à la qualité de service offerte par le réseau.

8. Adaptation de la commande (Control Over Network)

Cette approche consiste à adapter la commande relativement aux performances du réseau. La plupart des chercheurs spécialistes de l'automatique ont beaucoup contribué dans ce domaine, le plus souvent, lors de la conception de la commande du système de délai induit par le réseau est pris en compte. Cependant, il faut noter que longtemps les SCRs ont été considérés par cette communauté comme des systèmes à retard [Loiseau and Rabah 1997, Richard 2003]. Mais, le réseau n'est plus limité à un simple retard : les phénomènes, tels que : la perte de paquets, la duplication de paquets (réémission), commencent à être pris en compte.

Dans ce qui va suivre, un état de l'art sur les différentes méthodes de commande sur le réseau sera présenté. Ces méthodes sont assez variées et classées en plusieurs catégories.

8. 1 Méthode basée sur le modèle à temps discret étendu (augmenté)

Halevi et Ray [Halevi and Ray 1988] ont proposé une méthode basée sur un modèle déterministe étendu pour commander un système linéaire avec des délais réseau périodiques. La structure du modèle discret étendu est relativement simple et facile. De plus cette méthode peut être modifiée pour intégrer les périodes d'échantillonnage qui peuvent être différentes entre le capteur et la commande [Liou and Ray 1990]. Le procédé physique utilisé dans cette méthode est de la forme suivante :

$$x(k+1) = Ax(k) + Bu(k) \tag{6}$$

$$y(k) = Cx(k) \tag{7}$$

Où :

$A = \exp^{(A_c T)}$, $B = \int_0^T \exp^{(A_c S)} ds B_c$, avec : {A, B,C} est la réalisation du système ou encore la matrice dynamique, la matrice de commande (ou d'entrées) et la matrice de mesure (ou de sorties). Les dynamiques du contrôleur linéaire utilisées ici peuvent être décrites comme suit :

$$\xi(k+1) = F\xi(k) - Gz(k) \tag{8}$$

$$u(k) = H\xi(k) - Jz(k) \tag{9}$$

Où, ξ est le vecteur d'état du contrôleur, $z(k) = y(k-i)$, $i = \{1,\ldots,j\}$ représente la mesure passée à l'instant où $u(k)$ est calculé par le contrôleur, et *F*, *G*, *H* et *J* sont des matrices constantes représentant les dynamiques du contrôleur. La commande *u* dans l'équation 9 représente la sortie du contrôleur.
L'idée principale est de manipuler les délais induits par le réseau en combinant les équations 6 & 9, dans l'équation d'espace d'état étendu (augmenté) comme suit :

$$X(k+1) = \omega(k)X(k) \tag{10}$$

Où, $X(k)=\left[x^T(k), y^T(k-1),\ldots, y^T(k-j), \xi^T(k), u^T(k-1), u^T(k-l)\right]^T$ est le vecteur d'état étendu, $\omega(k+1)$ est la matrice de transition étendue et est calculée à partir de *A*, *B*, *C*, *F*, *G*, *H* et *J*.
En posant l'hypothèse que pour les délais périodiques, il existe un entier positif *M* tel que $\tau_{k+M}^{SC}=\tau_k^{SC}$, Halevi et Ray ont prouvé que le système de l'équation 10 est asymptotiquement stable si toutes les valeurs propres de $\Xi_k^M=\prod_{j=1}^{M}\omega(k+M-j)$ sont incluses dans le cercle unité.

8. 2 Méthode basée sur les files d'attente

Le mécanisme basé sur les files d'attentes peut être utilisé pour rendre les délais variables induits par le réseau dans un SCR fixes. De cette façon, le SCR devient à temps invariant. Cette méthode utilise quelques informations déterministes et probabilistes d'un SCR pour la formulation de l'algorithme de commande.
Il faut noter qu'une autre méthode basée sur le même principe a été développée par [Luck and Ray 1990, 1994], et est appelée méthode de compensation de délai basée sur un prédicteur déterministe. Cette méthode utilise un observateur pour estimer l'état du système à commander et un prédicteur pour calculer la commande prédictive basée sur les mesures précédentes de sortie.
La commande et les mesures de sortie passées sont stockées dans une file FIFO (First In First Out) et un registre à décalage définit *Q*1 et *Q*2, où les tailles de *Q*1 et *Q*2 sont μ et θ respectivement (Figure 37).

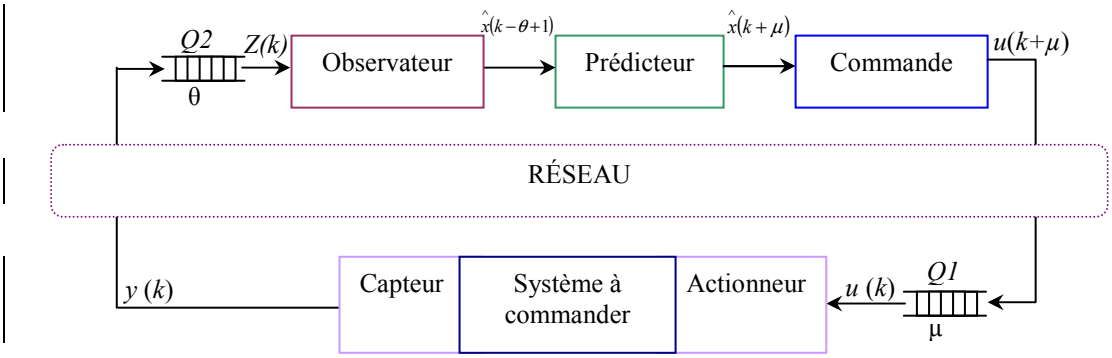


Figure 37. Compensation des délais induits par le réseau basée sur un prédicteur déterministe

Pour appliquer cette méthode, différentes étapes doivent être suivies :
1- utiliser l'ensemble des mesures passées $Z(k)=\{y(k-\phi), y(k-\phi-1),\ldots\}$ dans *Q*2, où ϕ est le nombre de paquets dans *Q*2 ; l'observateur estime l'état du système $\hat{x}(k-\theta+1)$.

2- le prédicteur utilise $\hat{x}(k-\theta+1)$ pour prédire le futur état $\hat{x}(k+\mu)$.

3- le contrôleur calcule la commande prédictive $u(k+\mu)$ à partir de $\hat{x}(k+\mu)$, et ensuite envoie $u(k+\mu)$ qui va être stocké dans *Q*1.

Puisque les performances de l'observateur et du prédicteur dépendent fortement de la précision du modèle, le modèle dynamique doit être très précis.
Chan et Özgüner (1995) ont développé un autre modèle de files d'attente pour commander des SCRs implantés sur des réseaux à retards aléatoires (Ethernet). C'est une méthode de

compensation de délai basée sur un prédicteur probabiliste qui utilise des informations probabilistes, ainsi que le nombre de paquets dans une file pour améliorer l'état de prédiction.
Comme le montre la figure 38 la file d'attente $Q1$ à la sortie du capteur possède une capacité μ, et la file d'attente à l'entrée du prédicteur ne peut stocker qu'un seul paquet. La sortie $y(k)$ est stockée dans $Q1$ en attendant d'être envoyée vers $Q2$ lorsque le réseau est disponible pour la transmission. Pour décrire cette méthode de compensation, supposons qu'un nombre de paquets soit stocké dans $Q1$ et que la sortie de $Q2$ soit définie en fonction de i et de $\omega(k)$, respectivement. A la période d'échantillonnage k, si le capteur ne peut envoyer $y(k)$ dans $Q2$ avant que $Q2$ ne soit lue, $\omega(k)$ est mise à $\omega(k\text{-}1)$. Sinon, $\omega(k)$ peut être identique à n'importe quelle valeur de l'ensemble $\{y(k), y(k-1), \ldots, y(k-\mu)\}$. Cependant, il y a une possibilité de réduire le choix de $\omega(k)$. Ainsi, $\omega(k)$ peut donc prendre soit la valeur de $y(k-i)$ ou de $y(k-i+1)$ si $i = 1, \ldots, \mu$ défini comme indice des délais et doit être connu.

Ensuite, le prédicteur estime l'état courant $\hat{x}$, puis la commande $u(k)$ peut être calculée suivant la loi de commande choisie.
Lelevé dans sa thèse [Lelevé 2000] a aussi proposé une solution basée sur les files d'attentes. Inspirée par les travaux de l'équipe de Kosgue [Kosgue et *al*. 1996], les variations de retard sont compensées grâce d'abord à une phase d'audit du réseau pour déterminer le retard mesuré le plus élevé. Ensuite les trames de données sont émises à période constante multiple de la période d'échantillonnage du maître (la commande) et de l'esclave (l'ensemble : actionneur, système commandé et capteur).

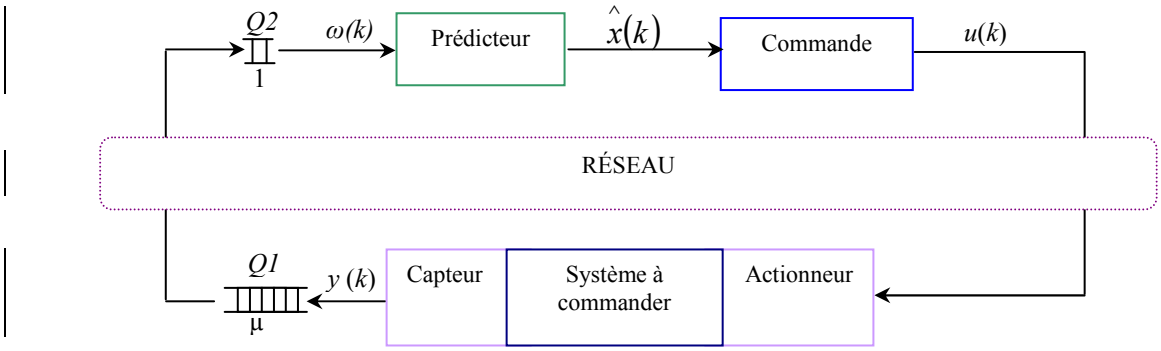


Figure 38. Compensation des délais induits par le réseau basée sur un prédicteur probabiliste.

8. 3 Méthode basée sur la commande stochastique optimale

Nilsson [Nilsson 1998] a proposé une commande stochastique optimale pour commander un NCS soumis à des délais aléatoires induits par le réseau. Cette commande traite les effets du délai induit par le réseau dans un SCR comme un problème linéaire quadratique gaussien (LQG). Dans cette méthode l'auteur suppose que le délai induit par le réseau τ : $\tau < T$.
Il faut noter que la commande quadratique gaussienne présente l'intérêt de s'appliquer à des systèmes dont l'état n'est pas mesuré. Les dynamiques du système distant dans cette méthode est décrite par :

$$x(k+1) = Ax(k) + A_0(\tau_k)u(k) + A_1(\tau_k)u(k-1) + v(k) \quad (11)$$

$$y(k) = Cx(k) + w(k) \quad (12)$$

Où, $\tau_k = \left[\tau_k^{sc}, \tau_k^{ca}\right]^T$, indique les délais à l'instant d'échantillonnage k. Avec, $A = \exp^{(A_c T)}$, $A_0 = \int_0^{T-\tau_k^{sc}-\tau_k^{ca}} \exp^{(A_c s)}\, dsB_c$, $A_1 = \int_{T-\tau_k^{sc}-\tau_k^{ca}}^{T} \exp^{(A_c s)}\, dsB_c$. $v(k)$, $w(k)$ sont respectivement le bruit d'état et le bruit de mesure, et sont des bruits blancs centrés avec une moyenne de zéro. Le problème LQG consiste à minimiser le critère suivant.

$$J(k) = E\left[x^T(N) Q_N x(N)\right] + E\sum_{k=0}^{N-1} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T Q \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \tag{13}$$

Où, $E[\cdot]$ est la valeur moyenne attendue, Q_N et Q sont les matrices de pondération. La loi de commande optimale du retour d'état est dérivée en utilisant la programmation dynamique.

$$u(k) = -L(k, \tau_k) \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \tag{14}$$

Où L est la matrice de gain optimale, le délai réseau τ_k est supposé indépendant, les délais passés sont des données supposées connues et exigées. Une autre loi de commande est aussi utilisée dans les mêmes travaux avec des délais modélisés par une chaîne de Markov. Une comparaison des résultats obtenus avec les deux modélisations a montré que la méthode basée sur l'utilisation de la commande LQG montre une meilleure performance.

Dans [Azimi-Sadjadi 2003] Azimi-Sadjadi utilise la commande linéaire gaussienne pour modéliser l'impact de la perte de paquets sur le SCR, et il utilise la méthode dite sous-optimale pour simplifier le calcul de l'estimateur et du contrôleur. Il montre que sous certaines conditions il existe un taux de paquets perdus pour lequel le SCR est stable (mean square stable).

8. 4 Méthode basée sur l'ordonnancement du temps d'échantillonnage

Une idée originale de Hong [Hong 1995] consiste à ordonnancer les périodes d'échantillonnage et à sélectionner la période d'échantillonnage appropriée pour un SCR, de telle façon que les délais induits par le réseau n'affecte pas significativement la performance du système commandé.

Cette méthode, est initialement utilisée dans des systèmes possédant plusieurs SCR, dont le délai induit est supposé périodique. Ensuite, l'équipe de Hong et Kim dans [Hong and Kim 2000] ont modifié cette méthode pour l'appliquer au cas des délais aléatoires induits par le réseau. Le réseau utilisé dans leur étude est CAN avec pour condition un délai induit $\tau<T$. Il faut noter que cette méthode n'est applicable que sur un seul SCR.

Pour décrire brièvement la méthode de Hong, soit M le nombre de SCR implantés sur un réseau, les temps d'échantillonnage des SCR sont calculés à partir du temps d'échantillonnage du SCR le plus sensible au délai induit par le réseau. Le choix de ce dernier se base sur une analyse dans le domaine fréquentiel du délai que peut supporter le SCR. Le SCR le plus sensible (SCR1) est celui qui possède le plus petit délai (φ_1). L'algorithme d'ordonnancement est basé sur le concept de fenêtre où L et σ sont respectivement les durées du champ de données et de l'entête des messages. T_1 est la période d'échantillonnage de SCR1 et r est le nombre de message qui peut être envoyé par le réseau surchargé.

Le temps d'échantillonnage T_1 est calculé comme suit :

$$T_1 = \frac{\varphi_1 + L}{3} \tag{15}$$

Pour trouver les périodes d'échantillonnage des autres SCR sur le même réseau, ces derniers sont d'abord rangés dans un ordre ascendant de leur délai pire cas. Par exemple SCR2 possède un délai pire cas supérieur à celui de SCR1 mais plus petit que celui de SCR3.
Les temps d'échantillonnage sont donc calculés à partir de T1 en respectant les conditions du réseau. Dans le cas général le temps d'échantillonnage est un multiple de T1 exprimé comme suit :
$T_i = k_i T_1,\ i = 2,3,\ldots,M.$

$$k_i = \wedge\left(\frac{\varphi_i - (T_i - L)}{2T_1}\right) \tag{16}$$

Où T_i est la période d'échantillonnage du SCRi, et $a = \wedge(b)$ indique que $a = 2^{v_i}$, $v_i \in \{0,1,2,\ldots\}$ et doit avoir une valeur très proche de b à condition qu'elle ne dépasse pas b.
En plus, l'utilisation optimale du réseau peut être accomplie avec la condition suivante :

$$2\sum_{i=1}^{M}\frac{T_1}{T_i} = r \tag{17}$$

Kim, Kwon et Park [Kim et *al*. 1998] ont amélioré cette méthode en développant un autre algorithme qui prend en compte le délai maximum de chaque SCR. Ce délai maximum est obtenu à partir des critères de stabilité du SCR concerné.

8. 5 Méthode basée sur le prédicteur de Smith

Comme nous l'avons souligné, en automatique classique, un retard pur τ peut être modélisé par le terme suivant :

$$e^{-s\tau} \tag{18}$$

On a donc :

$$G(s) = G_{bo}(s)e^{-s\tau} \tag{19}$$

Ceci entraîne un déphasage qui a pour conséquence la diminution rapide de la marge de phase. Donc, si nous souhaitons compenser parfaitement ce retard pur, il faudrait un prédicteur pur $c(t) = e^{s\tau}$, ce qui n'existe malheureusement pas (car non causal).
Une solution proposée dans [De Larminat 1993] utilise un prédicteur de Smith. Le principe du prédicteur est qu'il doit connaître les retards allers retours supposés constants (ou au moins lentement variable) τ, ainsi qu'un modèle (ne serait-ce qu'approché) $G_p^*(s)$ du processus à asservir $G_p(s)$. Le contrôleur doit posséder un correcteur rapide capable d'asservir le processus en l'absence de retard.

$y(s)$ est le signal utilisé par le contrôleur pour asservir le processus,

$$y(s) = \left[G_p^*(s) + e^{-2\varpi}\left(G_p(s) - G_p^*(s)\right)\right]u(s) \tag{20}$$

Si $G_p^*(s)$ modélise parfaitement le processus $G_p(s)$, alors :

$$y(s) = G_p^*(s)u(s) = G_p(s)u(s) \tag{21}$$

Le contrôleur peut ainsi agir comme s'il n'y avait aucun délai de transmission. Un simple correcteur de type PID peut suffire à contrôler parfaitement le processus.

Cependant, $G_p^*(s)$ n'est dans la réalité jamais parfait. D'autre part le prédicteur doit connaître parfaitement le retard τ, ce qui augmente nettement la marge d'imprécision de l'estimation.

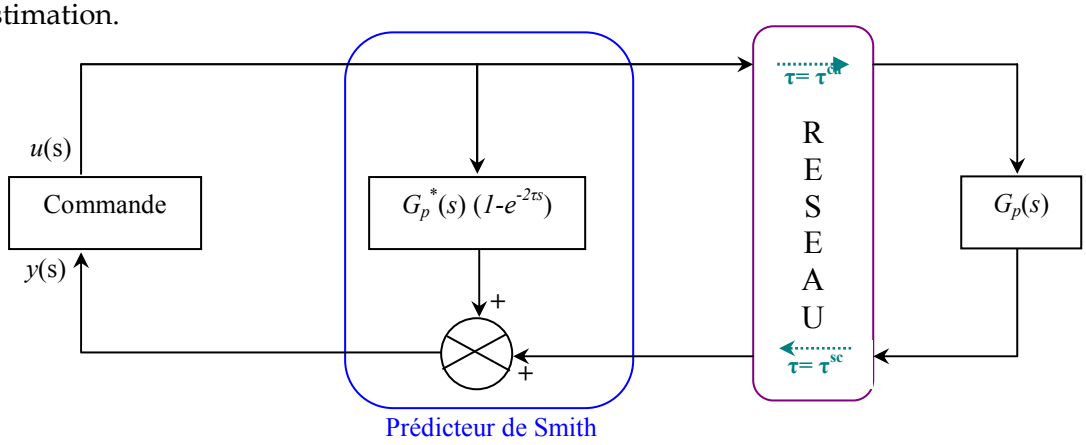


Figure 39. Prédicteur de Smith

Beaucoup d'autres travaux allant dans ce sens ont suivi comme l'équipe de Bauer et *al*. et Georges et *al*. dans [Bauer et *al*. 2001, Georges et *al*. 2005] qui ont utilisé le prédicteur de Smith pour compenser le délai induit par le réseau Ethernet commuté. L'originalité de ces travaux réside dans le fait que le délai est calculé en utilisant une horloge de synchronisation basée sur le protocole IEEE 1588. Il faut noter que le prédicteur de Smith est simple mais n'est plus utilisé à cause de son manque de robustesse [Cheong So 2003]. L'autre inconvénient est qu'il se base sur des hypothèses fortes (exemple retard constant).

8. 6 Méthode basée sur la commande robuste

Watanabe et son équipe, Göktas [Watanabe et *al*. 1996, Göktas 2000] ont développé une commande dans un SCR dans le domaine des fréquences en utilisant la théorie de la commande robuste. L'avantage de cette méthode est qu'elle n'exige pas la connaissance de la distribution de probabilité des délais réseau. Les délais τ^{sc} et τ^{ca} sont supposés avoir une limite et peuvent être approximés.
Le délai du réseau peut être calculé selon la théorie des fluides [Filipiak 1988] ou avec la théorie du calcul réseau [Georges et *al*. 2006 b].
Une fois que les délais maximum et minimum induits par le réseau sont estimés ils sont ensuite exprimés comme suit :

$$\tau^n = \frac{1}{2}(\tau_{\max} + \tau_{\min}) + \frac{1}{2}(\tau_{\max} - \tau_{\min})\delta,\ -1 \le \delta \le 1, \tag{22}$$

$$\tau^n = (1-\alpha)\tau_{\max} + \alpha\tau_{\max}\delta, 0 \le \alpha \le \frac{1}{2} \tag{23}$$

Où, τ^n peut être τ^{sc} ou τ^c, $\tau_{\max}$ est la borne maximale de τ^n et $\tau_{\min}$ est la borne minimale, α et δ sont des nombres réels déterminés en se basant sur l'application à étudier. Le premier terme $(1-\alpha)\tau_{\max}$ représente un délai constant, alors que $\alpha\tau_{\max}\delta$ représente l'incertitude du délai liée au premier terme. Ce délai est alors converti pour qu'il soit utilisable dans le domaine fréquentiel, il est donc approximé en utilisant l'approximant de Padé du premier ordre.

$$e^{-\tau^n s} = e^{-s(1-\alpha)\tau_{\max}} e^{-s\alpha\tau_{\max}\delta} \approx \frac{1 - s\tau^n/2}{1 + s\tau^n/2} \tag{24}$$

$$e^{-\tau^n s} \approx \left(\frac{1 - s(1-\alpha)\tau_{\max}/2}{1 + s(1-\alpha)\tau_{\max}/2} \right) \times \left(\frac{1 - s\alpha\tau_{\max}\delta/2}{1 + s\alpha\tau_{\max}\delta/2} \right)$$

Cette incertitude du délai est ainsi considérée comme une perturbation exprimée comme suit :

$\left(\frac{1 - s\alpha\tau_{\max}\delta/2}{1 + s\alpha\tau_{\max}\delta/2} \right) = 1 + W_m(s)\Delta$ où Δ est la fonction perturbation, avec :

$W_m(s) = \frac{\alpha\tau_{\max} s}{1 + \alpha\tau_{\max} s/k}$ est un poids qui couvre l'incertitude du délai, le facteur k se base sur la préférence du concepteur (ici il est égale à 3.465). La figure 40 résume brièvement une boucle de contrôle avec la commande robuste expliqué ci-dessous.

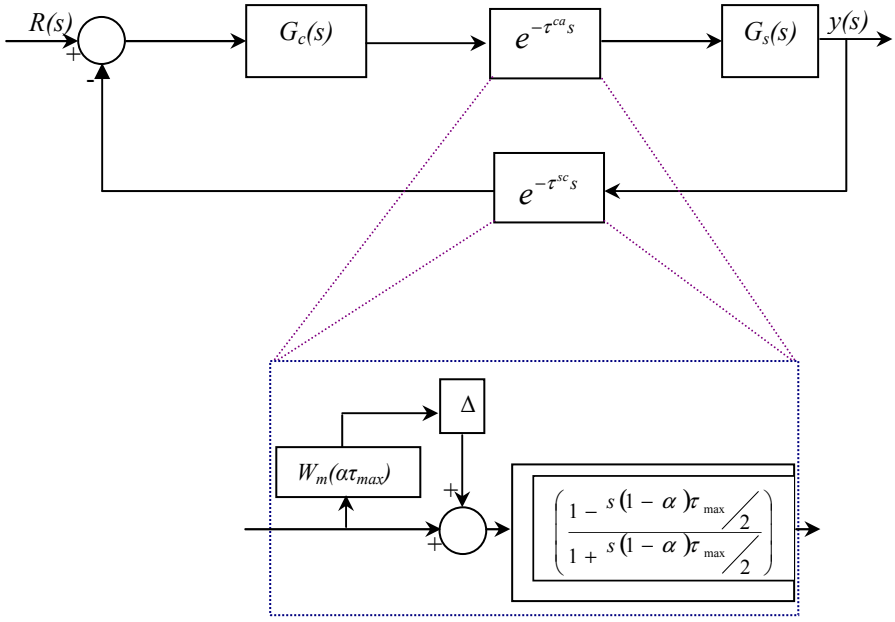


Figure 40. Compensation du retard basée sur la commande robuste [Tipsuwan and Chow 2001]

8. 7 Méthode basée sur un modulateur à logique floue

Pour compenser le retard induit par le réseau dans un SCR [Almutairi et *al*. 2001] ont proposé une commande PI modulable basée sur la théorie de la logique floue [Zadeh 1973]. Les gains de la commande PI (K_p, K_I) sont mis à jour à la sortie du contrôleur en fonction de l'erreur de la sortie du système causée par le délai réseau. Le problème de commande d'un DC moteur est utilisé pour illustrer cette méthode.

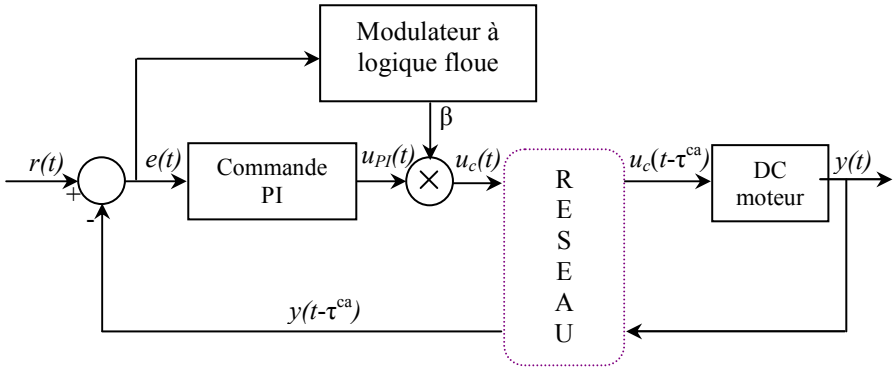


Figure 41. Compensation d'un retard induit par le réseau basée sur la logique floue.

Le système basé sur la logique floue est représenté par la figure 41 où r(t), e(t) et y(t) sont respectivement la référence, l'erreur et la sortie du système. La sortie de la commande PI est définie comme $u_{PI}(t)$ et la commande PI modifiée par le modulateur à logique floue est appelée $u_c(t)$. Cette unité modifie la commande $u_{PI}(t)$ comme suit :

$$u_c(t) = \beta u_{PI}(t) = \beta K_P e(t) + \beta K_I \int_{t_0}^{t} e(t)dt \tag{25}$$

Le facteur β est utilisé pour ajuster les gains de la commande PI sans l'interrompre. La valeur β est sélectionnée à partir de deux règles de logique floue basée sur les effets du délai réseau :
Si $e(t)$ est petite, alors $\beta = \beta_1$,
Si $e(t)$ est grande, alors $\beta = \beta_2$.

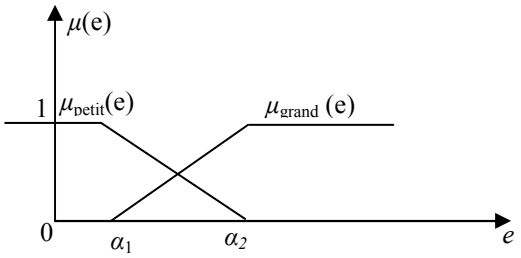


Figure 42. Fonctions d'appartenance de $e(t)$

Où, $0 < \beta_1 < \beta_2 < 1$. Les fonctions d'appartenance sont représentées par la figure 42 où μ_{petit}, μ_{grand} sont des fonctions d'appartenance; α_1, α_2 sont des facteurs qui ajustent la forme des fonctions d'appartenance. La forme des fonctions d'appartenance et les valeurs β_1, β_2 sont finement calculées grâce à une optimisation en ligne et hors-ligne. Celle-ci utilise l'algorithme du gradient basé sur des fonctions coût. Les fonctions coût pour l'optimisation en ligne sont :

$$J(k) = e^2(k) \tag{26}$$

$$J(k) = \sum_{i=k-m}^{k} e^2(i) \tag{27}$$

Ces deux coûts indiquent l'erreur quadratique instantanée et la somme des erreurs évaluées sur une fenêtre m. En d'autres termes, l'optimisation hors-ligne donne une fonction de coût différente :
$J = \lambda J_1 + (1-\lambda) J_2$,

Avec : $J_1(p)=\frac{\sum_{k=0}^{N} e(k)^2}{|J_1(P)|_\infty}$, $J_2(p)=\frac{\sum_{i=1}^{M} \Delta e_b(k)^2}{|J_2(P)|_\infty}$. Et $\{\Delta e_b(i)\}=\{\Delta e(k) \ | e(k)\Delta e(k)>0\}$, $\Delta e = e(k)-e(k-1)$.

Où λ est un facteur de pondération. Le coût J_1 place la pénalité sur le temps de réponse du système, le coût J_2 place une pénalité plus forte sur le dépassement et le comportement oscillatoire du système. P est un vecteur de paramètres de la fonction d'appartenance β_1 et β_2, p = [β_1 , β_2], [11].

8. 8 Quelques travaux *en bref*…

Dans la section précédente nous avons exposé quelques travaux menés dans le contexte des SCR. Cependant, d'autres études bien évidemment ont été menées telles que la méthode basée sur la perturbation de [Walsh et *al*. 1999]. Dans cette méthode les effets du délai sur les SCR ont été formulés suivant la théorie des perturbations qui s'annulent (vanishing perturbation) sous l'hypothèse qu'il n'y a pas de bruits d'observation. Cette méthode peut être appliquée à des SCR implantés sur des réseaux dont le retard induit est périodique, ou aléatoire, et se limite au cas où le réseau est implanté juste entre le capteur et la commande. Le réseau utilisé est basé sur les priorités (exp : CAN). Ces priorités peuvent être affectées en utilisant l'algorithme d'ordonnancement proposé par la même équipe [Walsh et *al*. 1999].
De plus, la méthode proposée exige un temps d'échantillonnage très court de telle manière que le SCR puisse être approximé par un système continu.

Tipsuwan et Chow dans [Tipsuwan and Chow 2001] ont proposé une méthode d'adaptation des paramètres de la commande suivant les conditions du trafic en cours. Dans ces travaux il est supposé que la commande et le système commandé peuvent mesurer les paramètres du réseau en utilisant un middleware. [Seuret et *al*. 2007] considère un SCR de type maître esclave communicant à distance par le moyen d'un réseau de type Internet. La commande utilisée correspond à un retour d'état basé sur un observateur et devra calculer la partie 'maître'. Le système global devra assurer une performance de rapidité garantie malgré les effets du retard induit par le réseau. Cette performance est obtenue en assurant une propriété de stabilisation exponentielle. Les gains du contrôleur et de l'observateur sont obtenus par la résolution d'inégalités matricielles linéaires (LMI). Avec cette méthode les retards induits par le réseau doivent être connus. Donc des hypothèses sur le retard ont été prises telle que : les bornes maximales du retard (aller-retour) sont supposées connues, un paquet de données perdu n'est pas réémis (protocole UDP) et les paquets doivent être envoyé dans l'ordre chronologique. Ce dernier point implique d'associer à chaque paquet son instant d'émission (par l'ajout d'une datation ou d'un time stamp). Pour réaliser ceci, cette équipe propose une solution technique qui consiste en une utilisation d'un système GPS. Ce système permet aux parties maître et esclave de synchroniser leurs horloges. Ainsi les paquets contenant les informations de contrôle et de mesure, contiennent aussi un time stamp de l'instant auquel ils sont envoyés.

[Tarn and Xi 1998] ont proposé une méthode de compensation des retards basée sur les événements pour un SCR implanté sur un réseau Internet : il s'agit d'un robot commandé via Internet. Ce concept est basé sur une carte de mouvement utilisée comme une référence du système (cette carte peut contenir les distances parcourues par le robot-l'historique-). Le mouvement de référence doit être une fonction non-décroissante du temps pour garantir la

[11] Pour plus de détail, le lecteur intéressé peut se référer à la thèse de [Al-Mutairi 2002]

stabilité du système. La mesure de la sortie du système est envoyée à travers le réseau et est appliquée à l'entrée de la carte de mouvement de référence. Cette dernière convertit la mesure en un mouvement référence qui est appliquée à un planificateur qui calcule la référence qui va être utilisé par le contrôleur. Cette méthode bien que inutilisable dans des structures directes et dans d'autres types de SCR à cause de son manque de généralité (puisque dédiée), compense les événements inattendus incluant le retard induit par le réseau. Notons que les recherches dans ce domaine sont abondantes [Montestruque et Antsaklis 2005, Nešić et Teel 2004, Yamé et Sauter 2007, Fraisse et *al*. 2007, Witrant et *al*. 2007]. Nous avons essayé d'être le plus synthétique possible et aussi nous avons essayé de dresser un large éventail de ce qui se fait dans ce domaine.

9. Conclusion

Dans ce chapitre les différents paramètres qui influent sur les systèmes contrôlés en réseau ont été présentés et montrés à travers un exemple simulé avec TrueTime. Ensuite, différents travaux de la communauté automaticienne qui traitent de l'accommodation des retards induits par le réseau dans les SCRs ont été exposés.
Rappelons que vu le caractère pluridisciplinaire des SCRs, les travaux de recherche sont pléthoriques dans ce domaine. Reste alors les solution proposées par les spécialistes en réseau et les nouvelles solutions sur lesquelles la communauté des SCRs se penche actuellement à savoir l'approche intégrée. Ces deux approches seront abordées dans le prochain chapitre.

Chapitre II :
Systèmes contrôlés en réseau : …De l'approche 'réseau' vers l'approche intégrée…

1. Introduction

Le chapitre précédent a montré l'incidence du réseau sur les SCR, puis a énuméré des solutions proposées par les automaticiens pour compenser les perturbations de communication dans la commande du système. Pour compléter l'état de l'art sur les recherches dans les SCR, ce chapitre sera dans un premier temps dédié aux solutions proposées par les spécialistes en réseau pour optimiser les performances du réseau. Puis, dans une deuxième partie, de nouvelles approches dites « intégrées » (ou co-design) associant plus finement le couple Qualité de Service et Qualité de contrôle seront présentées.

2. Méthodes de contrôle du réseau (Control of Network)

Appelé communément contrôle du réseau (control of Network). Cette méthode consiste à modifier les paramètres du réseau (protocole, bande passante, trafic,…) pour offrir une qualité de service adaptée aux besoins de l'application. Pour cela, les échanges entre le contrôleur et la partie instrumentation du process doivent être modélisés en prenant en compte la taille des messages, les temps d'inter-arrivée, etc… qui sont des informations d'entrées pour étudier le comportement du réseau.
La plupart des solutions de compensation du réseau pour les SCR reposent sur des méthodes algorithmiques, déterministes, probabilistes, ou combinées (graphiques et formelles) et proposent soit :

- de nouveaux protocoles d'accès au médium (exp: Try-Once-Discard (TOD) [Walsh et *al*. 1999, Mascolo 2007]) ;
- d'améliorer des protocoles existants
- d'ajouter des sous-couches protocolaires sur le protocole d'accès au médium (exp: TDMA, PCSMA, P-CSMA, CSMA/DCR [Wang et *al*. 2002, Lo Bello and Mirabella 2004, Alves et *al*. 2000]).
- de nouveaux algorithmes d'ordonnancement,
- d'améliorer des algorithmes d'optimisation de réseau existants.

2.1 Méthodes algorithmiques

2.1.1 Méthode basée sur des algorithmes d'accès au médium

Dans [Walsh et *al*. 1999] Walsh, Ye et Bushnell ont proposé un nouveau protocole d'accès au médium appelé Try-Once-Discard (TOD). Ce protocole prenant en compte les caractéristiques du SCR utilise un algorithme d'ordonnancement dynamique, faisant appel au réseau selon les besoins le l'application.
Dans le TOD le nœud qui possède la plus grande erreur pondérée à partir de la dernière valeur reportée (au contrôleur) va gagner le médium. Cette technique d'ordonnancement est appelée « Maximum-Error-First » (MEF) et protocole TOD. Si un paquet de donnée ne gagne pas la compétition pour accéder au médium, il est éliminé, et la prochaine nouvelle donnée est utilisée. Dans le but de caractériser le comportement de cet algorithme, les auteurs introduisent le concept de la croissance maximale permise β de l'erreur en τ secondes. Pour n'importe quel instant t, la limite β est définie comme suit : $\beta > \|e_i(t+\tau) - e_i(t)\|$ s'il n'y a pas de transmission de données. Plus clairement, β dépend de l'état interne du système à commander et des erreurs en cours. Cependant, ce protocole ne garantit pas que chaque nœud transmette un message une fois toutes les p transmissions. Sachant que p est le nombre de nœuds (capteurs) partageant le réseau.

D'autres auteurs ont tenté d'améliorer les protocoles existants afin qu'ils répondent à des contraintes temps réels. Le réseau qui a suscité le plus d'intérêt par rapport à ces approches est le réseau Ethernet. En effet, puisque Ethernet est un réseau non-déterministe dû à son protocole d'accès au médium (CSMA/CD), beaucoup d'améliorations ont été proposées. La solution la plus simple qui a été proposée est le TDMA [Kopetz et *al*. 1989] (Time Division Multiple Access) où un intervalle de temps est alloué à chaque station pour transmettre ses données. Cependant si une station n'a rien à transmettre elle utilise quand même le temps qui lui est alloué. Le protocole PCSMA (Predictable CSMA) [Yavatkar et *al*. 1992] est un algorithme orienté-donnée dans l'allocation du médium. C'est-à-dire que ce protocole effectue un ordonnancement hors-ligne en considérant que toutes les données temps réel qu'il manipule sont périodiques. Cependant, le phénomène de surcharge est inhérent à cette méthode d'accès au médium. Un autre protocole P-CSMA (Prioritized-CSMA) [Tobagi 1982] est aussi proposé, il est comme le TDMA mais orienté priorité des messages, avec ce protocole le média est divisé en *n* slots et une station possédant la plus haute priorité peut transmettre ses données durant le slot qui lui est alloué. Mais le même problème d'indéterminisme persiste quand deux stations possédant la même priorité émettent en même temps.

Toutes les approches précédentes sont des techniques d'évitement de collisions, cependant il y a d'autres approches qui se focalisent sur la technique de résolution de collision. Nous citons par exemple le CSMA/DCR (CSMA with Deterministic Collision Resolution) [Lann and Rivière 1993] qui fournit un délai d'accès au médium limité.

2.1.2 Méthode basée sur les algorithmes d'évitement et de contrôle de congestion (algorithmes d'ordonnancement)

[Mascolo 2003] a proposé un algorithme de contrôle de congestion utilisant un prédicteur de Smith avec lissage des entrées. L'auteur s'est intéressé au problème de congestion dans Internet. Dans son étude la théorie de contrôle est utilisée pour le contrôle de congestion dans Internet et considère le système comme un système à retard.
Mascolo a implémenté un simple contrôleur P (proportionnel) en plus du prédicteur de Smith. Le principe d'auto-synchronisation est utilisé pour connaître le retard induit par le réseau Internet. Cependant, comme nous l'avons montré précédemment cette méthode n'est pas robuste.

[Altman et *al*. 1999] ont formulé le problème de contrôle de congestion comme un problème de commande stochastique (LQG) pour les sources ABR (Available Bit Rate) implantés sur les réseaux ATM (Asynchronous Transfer Mode). Les auteurs considèrent le cas où un seul goulot d'étranglement (une seule file d'attente) détermine la performance d'un ensemble de sources. Dans ce cas un mécanisme de contrôle assez simple est appelé rate matching [Kalyanaraman 1997]. Dans ce type de mécanisme le nœud calcule à des intervalles périodiques le taux moyen disponible pour les sources et divise la fraction de cette capacité qui reste entre les différents utilisateurs. Cet algorithme a l'avantage d'être simple mais son inconvénient réside dans son incapacité à contrôler d'une façon optimale la longueur de la file d'attente et ainsi d'éviter le débordement du buffer. D'autres approches [Mascolo et *al*. 1996, Kolarov and Ramamurthy 1997, Benmohamed and Wang 1998] utilisant le même principe ont été développées. Leur but est la stabilité de la longueur de la file d'attente et non l'optimalité.

Il faut noter qu'il y a aussi d'autres techniques d'ordonnancement des files d'attente dont le but principal est de respecter la qualité de service et donc de réduire le retard et la perte des

paquets. Les techniques d'ordonnancements sont nombreuses. Elles sont pour la plupart algorithmiques (simulation) et elles sont choisies parmi les alternatives suivantes :
FIFO (first in first out), PS (Priorité statique, messages servis selon leur priorité), WFQ (weighted round robin), CBQ (class based queueing, EDF (Earliest deadline First).
Aussi, d'autres algorithmes ont vu le jour, nous citons par exemple le nouvel algorithme d'ordonnancement proposés par [Koubaâ 2004] appelé (m, k)-WFQ qui intègre les contraintes temporelles (m,k)-firm [Hamdaoui and Ramanathan 1995] dans l'algorithme de WFQ (weighted fair queueing) standard [Wang et *al*. 2002] afin d'améliorer la qualité de service temporelle des flux temps réel ayant de telles contraintes. L'ordonnanceur WFQ est déployé dans les réseaux à intégration de services grâce à ses propriétés de garantie de bande passante et de délai borné pour les applications temps- réels. Ce type d'ordonnanceur est efficace mais très complexe à mettre en œuvre.

2.1.3 Méthode basée sur les algorithmes génétiques (lissage de trafic)

Dans [Lo Bello et *al*. 2005] Lo Bello et son équipe ont proposé une méthode basée sur le lissage flou (fuzzy smoothing) du trafic pour améliorer le comportement temps réel d'Ethernet. Cette technique réalise un lissage adaptatif du trafic permettant de fournir une limite statistique maximum à laquelle le message arrive à destination. Aussi une optimisation du lisseur flou est effectuée en utilisant les algorithmes génétiques.
Le lisseur flou du trafic est un lisseur adaptatif basé sur une commande flou possédant deux entrées et une sortie. Les entrées envoient le nombre de collisions et le débit général observé durant un intervalle de temps, la sortie représente la variation de la période de rafraîchissement.
Les auteurs expliquent que si un modèle dynamique d'un système est complexe la logique floue semble être la méthode la plus adéquate dans ce cas de figure. Cette commande utilise la connaissance experte [Travé-massuyès et *al*. 1997] du système. Trois fonctions d'appartenance sont définies pour chaque entrée du contrôleur flou correspondant respectivement aux valeurs (faible, moyen, fort) pour chaque variable. Ces valeurs ont été choisies heuristiquement pour représenter tous les modes opératoires du système suivant les valeurs prises par les entrées du système sans introduire un nombre excessif de combinatoire. Pour compléter le contrôleur, des règles d'inférence ont été définies indiquant l'action de contrôle à appliquer suivant toutes les combinatoires possibles des variables d'entrées.
Exemple:
If collisions is < low> and throughput is <low> then VarRP is < - 0.5 RP min>
Avec VarRP la variation de la période de rafraîchissement et RP min est la période de rafraîchissement minimale.

L'optimisation du lisseur flou est basée sur les algorithmes génétiques (AG), plusieurs auteurs [Bodenhofer and Herrera 1997, Seng et *al*. 1999] ont confirmé l'efficacité des AG dans la conception des contrôleurs flous pour obtenir le comportement désiré. Les AG ont été utilisé pour raccorder les paramètres des fonctions d'appartenance du lisseur.

2.2 Méthodes déterministes

2. 2. 1 Méthode basée sur le calcul réseau

[Grieu 2004, Georges 2005] se sont basés sur la méthode du calcul réseau (Network calculus) pour modéliser le réseau Ethernet commuté. Les contextes de chacun sont différents mais les contraintes temporelles sont les mêmes. Pour Grieu cette méthode est utilisée dans le

contexte de l'aéronautique et pour Georges dans le contexte des systèmes de commande. Dans les deux travaux un modèle fonctionnel et analytique du réseau Ethernet basé sur la théorie du calcul réseau en vue de majorer les temps de traversée du réseau a été proposé.
Nous pouvons citer d'autres travaux [Jasperneite et *al*. 2002, Watson et Jasperneite 2003] qui utilisent la même théorie.
Le calcul réseau est une théorie développée par Cruz [Cruz 1991 a, b]. Le contexte général de ces travaux est celui des réseaux de communication avec commutation de paquets selon un routage statique. C'est à dire que toute donnée émise est sous forme de paquet et non sous forme de bits individuels. Ces paquets circulent dans le réseau suivant un itinéraire prédéfini jusqu'à atteindre leur destination. D'ailleurs comme l'a fait remarquer Cruz ces travaux peuvent être appliqués sur les chaînes d'assemblage de produits manufacturiers [Gallager and Parekh 1994].
L'intérêt principal de ces travaux est la notion d'enveloppe qui permet de caractériser un trafic, et Grieu dans ses travaux a pu constater que la méthode proposée par Cruz reste rudimentaire et manque de généralité. Cependant, Le boudec et Thirand dans [Le Boudec and Thirand 2001] ont présenté un formalisme basé sur l'algèbre min-plus qui permet de définir les notions de courbes de service et d'enveloppes.
L'idée qui a conduit à ce formalisme est la volonté de se rapprocher de la théorie des systèmes qu'on utilise par exemple, pour les réseaux électriques ou le traitement du signal. Par analogie, nous pouvons voir la courbe de service comme la réponse impulsionnelle du système, qui permet par un opérateur de convolution, de calculer la sortie d'un système dont les entrées sont connues. Nous avons souligné précédemment que le formalisme de Le Boudec fait appel à l'algèbre min-plus. Par analogie à l'algèbre (R,+,*), l'addition est remplacée par le minimum, et la multiplication est remplacée par l'addition .
Pour avoir une idée sur ce formalisme nous vous proposons un exemple de caractérisation des éléments par la notion de courbe de service :

Exemple :
Définition : Soit un système S et un flux passant par S de fonction d'entrée $R(t)$ et de sortie $R^*(t)$. On dit que S offre au flux une courbe de service β si et seulement si β est une fonction positive croissante telle que pour tout $t \geq 0$:

$$R^*(t) \geq \inf_{s \leq t} \{R(s) + \beta(t-s)\} \quad (1)$$

En pratique si β est continue et R continue à gauche, on peut remplacer cette définition par :
Pour tout t, il existe un t_0 tel que :

$$R^*(t) \geq R(t_0) + \beta(t - t_0) \quad (2)$$

Cette définition signifie qu'on peut toujours trouver un instant à partir duquel l'élément a servi le flux au moins à la vitesse représentée par β.
Pour expliciter cette notion de courbe de service on peut citer l'exemple d'un élément à délai borné T, c'est-à-dire que la seule garantie de service offerte au flux est qu'aucun de ses bits ne subira un retard de plus de T unités de temps dans cet élément. La fonction δ_T suivante permet de caractériser un tel élément :

$$\begin{cases} \delta_T(t) = 0 & pour\ t < T \\ \delta_T(t) = \infty & pour\ t \geq T \end{cases} \quad (3)$$

On peut en effet réécrire la définition d'une courbe de service avec cette fonction δ_T en : $R^*(t) \geq R(t - T)$, ce qui est précisément le comportement recherché.

Pour évaluer les performances de ce type de modélisation, la simulation est utilisée. L'inconvénient de cette approche est la lourdeur des calculs de cette méthode ce qui empêche son implantation dans les systèmes temps réels. Cependant son utilisation reste utile pour un calcul hors ligne (off-line) voir les travaux [Georges et *al*. 2006 a, b].

2. 3 Méthodes probabilistes

La méthode que nous allons présenter peut être considéré comme combinée car en plus des éléments probabilistes qu'elle modélise, elle décrit aussi le comportement du système.

2.3.1 Méthode basée sur les chaînes de Markov et la théorie des files d'attente

Nous avons vu dans les sections précédentes que [Nilsson 1998] avait utilisé les chaînes de Markov pour modéliser les retards induits par le réseau, puis il a proposé une commande pour les réduire. Dans cette section, cette méthode est utilisée pour modéliser le comportement du réseau et en évaluer ses performances. Une chaîne de Markov [Fdida et Hébuterne 2004, Baynat and Dallery 2004 a, Baynat 2000] décrit l'évolution d'un système à partir de ses états. Cette méthode analyse le comportement d'un système d'après les spécifications suivantes:

- la quantité d'information utilisée pour décrire les états de la chaîne.
- le calcul des taux de transition entre états.

Le choix de la définition des états de la chaîne est très important, car il intervient dans la complexité de la résolution du modèle. Le nombre d'états de la chaîne de Markov restreint l'utilisation de cette technique, dans la plupart des cas, à des petits systèmes.
Une alternative de modélisation est fournie par les réseaux de files d'attentes. Cette méthode est certainement la plus utilisée pour estimer les performances d'un système informatique, et de systèmes distribués sur un réseau de communication.
L'utilisateur de cette méthode dispose de langages spécialisés permettant la description et l'étude quantitative de modèles ayant la structure générale de réseaux de files d'attente. Cependant la classe des files d'attentes que l'on sait résoudre est réduite et ne permet pas de prendre en considération certains comportements très répandus dans les réseaux informatiques. On peut citer, par exemple, la possession simultanée de plusieurs ressources, le blocage d'un client, la concurrence pour l'accès à des ressources partagées ou des comportements de dépendance entre clients (synchronisation…etc). Le fait d'ignorer ces comportements a pour conséquence de restreindre les domaines d'application des réseaux de files d'attente [Baynat and Dallery 2004 b] ou conduit à des erreurs d'évaluation de leurs performances.
La représentation élémentaire d'un réseau de file d'attente est donnée par une file à un seul serveur. Cette file élémentaire est caractérisée par :

- la suite, $0 \le t_1 \le t_2 \le \cdots \le t_j \le \cdots$ des instants d'arrivée des clients dans la file. Notons $I_j = t_{j+1} - t_j$, $j \ge 1$ les durées séparant ces entrées successives ;
- la suite $S_1, S_2, \cdots, S_j, \cdots$ des temps de service des clients 1, 2, …, *j*, … ;
- la discipline de service qui donne l'ordre dans lequel des clients, arrivant dans la file, sont servis (FIFO, RANDOM, etc…) ;

Si la discipline de service est FIFO (premier arrivé premier servi) le temps d'attente w_j , du *j*-ème client est la durée de temps qui sépare son arrivée dans la file de l'instant t_j de son début de service. Pour une file à serveur unique, on a :

$$w_{j+1} = \begin{cases} 0 & si\ t_{j+1} < t_j + w_j + S_j \\ w_j + S_j - I_j & Sinon \end{cases} \qquad (32)$$

Le temps de résidence ou temps de réponse du *j*-ème client correspond au temps de séjour total dans le système, c'est-à-dire (w_j + S_j). La résolution analytique de la file donne, à l'état stationnaire, la probabilité $p(n)$ que la file contienne n clients, $n \geq 0$.

$$w_{j+1} = \begin{cases} 0 & si\ t_{j+1} < t_j + w_j + S_j \\ w_j + S_j - I_j & Sinon \end{cases} \quad (4)$$

Le temps de résidence ou temps de réponse du *j*-ème client correspond au temps de séjour total dans le système, c'est-à-dire (w_j + S_j). La résolution analytique de la file donne, à l'état stationnaire, la probabilité $p(n)$ que la file contienne n clients, $n \geq 0$.
Une file d'attente est généralement représentée suivant la notation de Kendall[12].
Un réseau de file d'attente est composé d'un ensemble de stations de service (modélisant les ressources : par exemple un nœud de commutation, une unité centrale, etc..) et d'un ensemble de clients. Les files d'attente devant chaque station contiennent les activités qui ont besoin d'accéder ou qui ont déjà accédé à la ressource. Ce système est caractérisé par les processus représentant l'arrivée des clients au réseau, les temps de service des clients aux stations, le cheminement des clients d'une station à l'autre et les disciplines de service des clients à chaque station.
Les principaux réseaux de files d'attente dont on connaît la solution sous forme explicite sont essentiellement les réseaux de Jackson et les réseaux BCMP. Cependant ces solutions sont limitées. Donc, étant donné les difficultés, voire l'impossibilité d'aboutir à des solutions exactes dans de nombreux cas, il est inévitable d'utiliser un certain nombre de méthodes approximative (Méthode par valeur moyenne, méthode de décomposabilité, méthode d'isolation) permettant d'approcher la solution.
Cette méthode peut être utilisée pour des systèmes simples mais elle montre de plus en plus de limites pour des systèmes complexes.
Il faut noter que cette méthode basée sur les chaînes de Markov est appelée aussi approche *probabiliste*. Par ailleurs, une étude d'évaluation d'Ethernet commuté a été effectuée par [Song 2001] sous l'hypothèse que toutes les trames Ethernet ont la même taille. Cette hypothèse de simplification ne reflète pas la réalité du réseau Ethernet où la taille de la trame est variable. [Koubâa and Song 2004] ont étendu leur étude en prenant en compte le cas général où la taille de la trame est aléatoire et en tenant compte des priorités des trames dans un réseau Ethernet commuté.

2. 4 Méthodes formelles et graphiques (comportementales)

2. 4. 1 Méthode basée sur les réseaux de Petri

Les réseaux de Pétri ont largement été utilisés dans le domaine des réseaux de communication. Un point crucial qui est inhérent au RDP est qu'il peut être étendu selon les besoins. Aussi en plus de ses caractéristiques analytiques, il offre une possibilité d'analyse qualitative. Cet avantage nous ne le rencontrons pas dans les approches déterministes et algorithmiques évoquées plus haut.

[12] La notation de Kendall est définie comme suit : A/S/C(DS/K/L) ou A/S/C/K/L/(DS) ou A :S/C/K/L/DS. Avec A : processus d'entrée, S processus de sortie, C : nombre de serveurs, K : capacité maximale de la file, L population des usagers, DS : discipline de service.

[Abdellatif 2002, Abdellatif and Juanole 1999] ont proposé un cadre basé sur les réseaux de Petri stochastiques pour l'analyse de la qualité de service. Les auteurs ont mis en œuvre une procédure de marquage des paquets qui a été proposée par le RFC 2698 de IETF, sur un ordonnancement à priorités statiques, et sur un algorithme de gestion de buffer dit à seuil par classe, c'est-à-dire qui prend les décisions d'élimination de paquets par simple comparaison de l'occupation des classes à des seuils prédéfinis. Aussi, ils proposent une amélioration de l'algorithme de contrôle d'admission de flux avec garanties de la qualité de service déterministes dans les réseaux à disciplines de service à débit contrôlés avec l'algorithme EDF pour ordonnancer les paquets.
Le choix de ce formalisme est basé sur la capacité des RDPS d'exprimer les mécanismes de la qualité de service tels que : l'ordonnancement des paquets, la gestion de buffer, la régulation de trafic et le contrôle d'admission. De plus, les RDPs peuvent exprimer le temps et certains aspects probabilistes pour l'analyse des performances.
Les auteurs se sont intéressés à l'analyse de la qualité de service dans les réseaux à commutation de paquets et plus spécifiquement dans les réseaux ATM et IP. Pour ce faire, ils ont modélisé les mécanismes qui agissent directement sur les paquets. Ensuite une méthode d'analyse et des mesures de performance sur le modèle RDPS d'un réseau ont été effectuées. Les métriques mesurées sont : le délai de transfert moyen, le taux de perte de paquets, des bornes sur le délai de transfert, l'utilisation du réseau. Ils ont aussi entrepris de modéliser un commutateur ATM sous un environnement producteur/consommateur périodique. Cet environnement est largement utilisé dans l'industrie [Thomesse et *al*. 1995, Décotignie 2005].
Les auteurs se sont contraints à une modélisation modulaire du système. La figure 41 représente le système global avec un rectangle pour indiquer une transition temporisée et une barre pour indiquer une transition immédiate (c'est-à-dire une transition qui est franchie dès qu'elle est sensibilisée). Le modèle du système est composé de quatre modules :
-Le module de production représente un modèle de trafic périodique généré lors d'une connexion C par le producteur.
-Le module du trafic de l'environnement qui représente une agrégation de (N-1) sources de Bernoulli. Ceci est effectué grâce à une distribution associée à deux transitions en conflit : arrivée des cellules et pas de cellules à l'entrée. La probabilité de tir (ou de franchissement) est fixée pour les deux transitions.
-Le module du commutateur qui est constitué de trois sous-modules : la concentration, la mémorisation et l'ordonnancement. Le concept de priorité des tâches est résolu en utilisant un module de séquencement des opérations existantes dans le commutateur.
L'idée principale de ce module est de diviser le slot time (intervalle de temps) en quatre phases: l'arrivée, la concentration, l'ordonnancement et la transmission. Durant chaque phase toutes les transitions exécutant cette phase sont sensibilisées, pendant que les autres ne sont pas actives.
-Le module de consommation n'est pas représenté ici, puisque le consommateur considéré est guidé par l'événement de l'arrivée d'une cellule. Donc le franchissement de la transition « transmission cellule de C » coïncide avec la consommation.

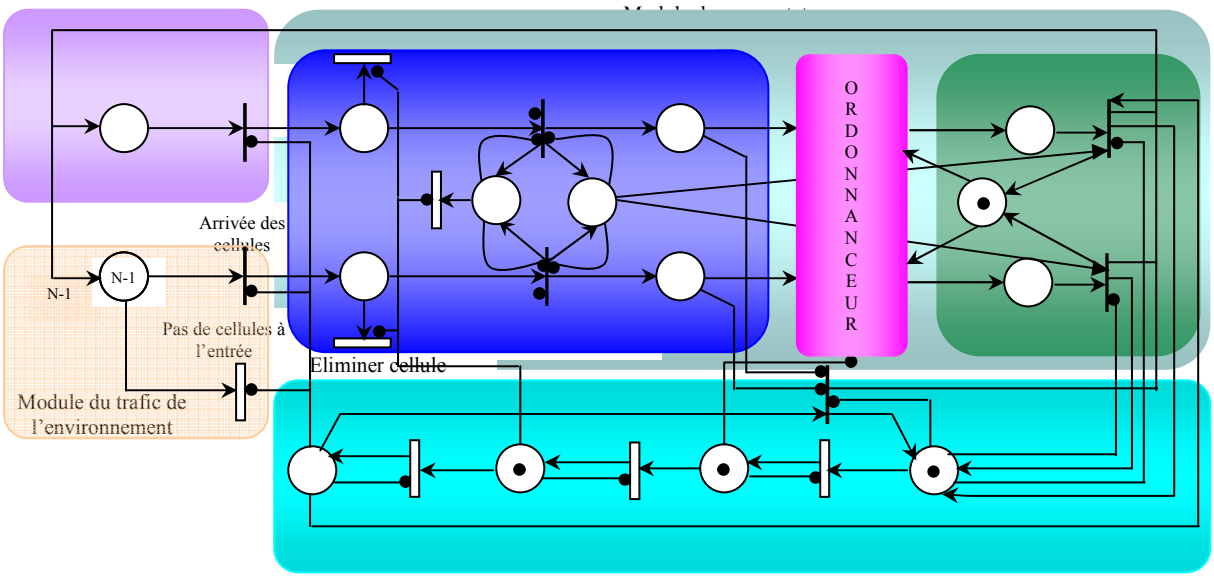


Figure 1. Modèle RDPS du commutateur ATM [Abdellatif and Juanole 1999]

Le but de ce modèle est l'évaluation de performance du modèle producteur/consommateur périodique dans le cas où la charge du commutateur et les politiques d'ordonnancement varient. Les performances auxquelles les auteurs se sont intéressées sont :
-La probabilité de perte des cellules (fraction des cellules produites perdues dans le commutateur),
-Le délai de la cellule, est défini comme le temps moyen perdu par la cellule produite entre le moment où elle entre dans le commutateur et l'instant de son départ du commutateur,
-Le temps d'inter-consommation représente le temps moyen passé entre deux consommations.
D'autres auteurs comme [Marsal et *al*. 2006, Marsal et *al*. 2005] utilisent les réseaux de Petri colorés pour évaluer le temps de réponse des architectures des systèmes automatiques utilisant les réseaux Ethernet commutés dans un contexte client serveur. Cependant la relation producteur /consommateur est plus souvent utilisée comme modèle de coopération pour la conception des systèmes distribués industriels [Thomesse et *al*. 1995, Décotignie 2005, Cisco].
L'inconvénient principal de ce type de modélisation est l'explosion combinatoire. En effet plus le système est de grande taille plus il nécessite un graphe d'état de grande taille ce qui le rend non analysable à l'aide de machines conventionnelles.

2. 4. 2 Méthode basée sur les automates

[Krȧkora and Hanzalek 2004, Krȧkora et *al*. 2004, Dolejš et *al*. 2004] ont utilisé le concept des automates temporisés pour modéliser un système distribué (d'une voiture) sur un réseau CAN. Les auteurs ont effectué une vérification de ces modèles en utilisant la logique temporelle. Ils ont souligné le problème crucial que soulevait la vérification de tels systèmes complexes, c'est à dire vérifier les propriétés temporelles (par exemple : le temps de réponse des messages) et la logique temporelle (par exemple : le deadlock).
L'approche proposée se base sur la modélisation d'un système à événement discret par les automates temporisés et sur la vérification du modèle par un outil de model checking (vérification) appelé UPPAAL [Pettersson and Larsen 2000].
Le comportement du réseau CAN est modélisé en prenant en compte les aspects de résolution de conflits basés sur les priorités dans le réseau CAN. Le modèle est appelé modèle d'arbitrage. Ensuite, un modèle du transceiver dont le modèle d'arbitrage est partie,

intégrante est réalisé. Pour représenter le comportement du bus physique, un modèle qui renseigne sur son état (libre ou occupé) est représenté.

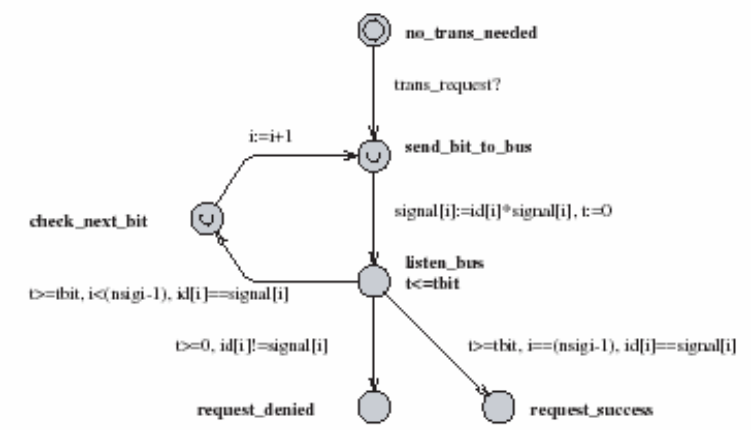


Figure 2. Exemple du modèle d'arbitrage réalisé avec UPPAAL [Krȧkora and Hanzalek 2004]

Pour que le modèle soit complet quatre applications sont représentées par quatre CPU, trois applications sont périodiques et la quatrième est sporadique.
Pour vérifier le modèle, des propriétés sont formulées avec la logique temporelle en utilisant UPPAAL. Cette étape permet de vérifier les propriétés logiques et temporelles qui peuvent être exprimées comme suit :

- Le système a-t-il un état bloquant ?
- Existe-t-il un état pour lequel un message possédant la plus haute priorité ne gagne pas le bus ?
- Est ce que tous les messages périodiques transmis sont reçus avant leur échéance ?
- Quels sont les temps de réponse pire cas pour les messages périodiques ?

Par ailleurs, dans [Gu and Shin 2003] les auteurs ont modélisé un système temps réel embarqué avec les réseaux de Pétri temporels pour la modélisation physique et comportementale d'un système de croisement des chemins de fer.
En adoptant cette approche, les modèles permettent d'avoir une vue intégrée et globale du système. Ceci peut aider le concepteur du système à voir l'influence d'un changement du système logique embarqué sur le système physique. Cette modélisation permet notamment d'analyser les contraintes de vivacité et de sûreté.

Les méthodes adoptées dans cette section ont une limite liée à l'explosion combinatoire : plus le système est complexe plus sa représentation le devient aussi, rendant sa vérification ardue et même dans quelques cas impossible. Aussi les outils de vérification (UPPAAL, KRONOS…) montrent une limite par rapport à l'expression de quelques propriétés logiques complexes.

3. Vers une approche intégrée…le co-design

Dans les sections précédentes, les approches présentées proposent soit des modèles pour évaluer les performances du réseau qui seront utilisées comme des données d'entrée pour analyser les performances de l'application, soit des mécanismes d'optimisation du réseau pour répondre aux contraintes applicatives.

Dans la modélisation intégrée la qualité de service du réseau (*i.e.* la bande passante, la gigue, le retard et la perte des paquets et la fiabilité du réseau) doit être considérée simultanément

avec la performance du système commandé (*i.e.* stabilité, l'adaptabilité, robustesse etc…) lors de la conception d'un SCR (figure 3).

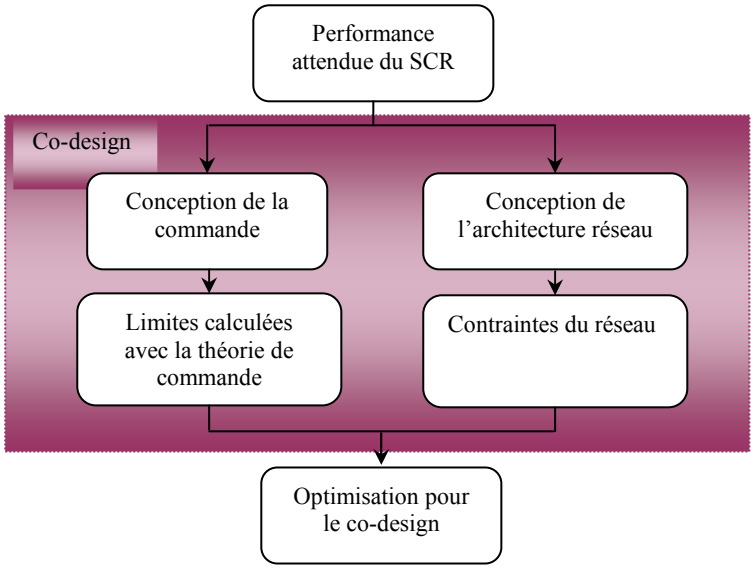


Figure 3. Principe du Co-design (la co-conception) [Branicky et *al*. 2003]

Un autre point important souligné par [Branicky et *al*. 2003] est de pouvoir proposer des outils de co-simulation pour aider à la co-conception de SCR.
Le but de la co-simulation est de simuler le système de commande en boucle fermée tout en prenant en compte l'ordonnancement de tâches et le comportement du réseau, de façon à pouvoir évaluer la performance du SCR globalement. (voir la figure 4).

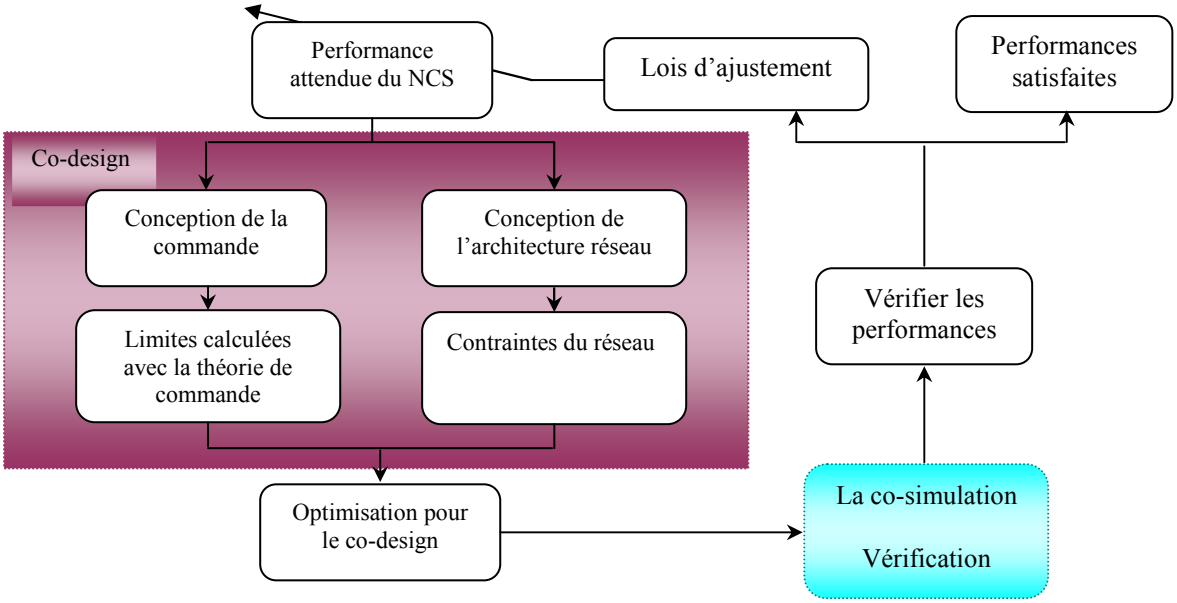


Figure 4. Boucle de conception d'un SCR [Branicky et *al*. 2003]

Beaucoup de travaux ont été réalisés prenant en compte de façon simultanée la performance du système commandé et l'ordonnancement de la CPU dans les systèmes temps réel, ou encore, la performance du système commandé et l'ordonnancement des messages dans le

réseau. Il faut noter, que d'autres travaux par exemple [Berbra et *al*. 2007] utilise la notion de co-design pour la tolérance aux fautes des systèmes contrôlés en réseau. Le but ici est d'évaluer l'influence du réseau sur la génération de résidus.

3. 1 Ordonnancement régulé des tâches de commande

Dans cette section deux grandes approches sont distinguées, à savoir : l'ordonnancement régulé basé sur l'état de la tâche de commande et l'ordonnancement régulé basé sur l'état de l'application (c'est-à-dire la performance du système commandé)

3. 1. 1 Ordonnancement régulé basé sur l'état de la tâche de commande

[Årzén et *al*. 1999, Cervin and Eker 2000, Årzén et *al*. 2000, Cervin et *al*. 2002, Cervin 2003] ont proposé un ordonnanceur à boucle de retour. L'ordonnanceur recueille les instants d'exécution courants des tâches et les informations de changement de mode des tâches de commande. La figure 6 résume cet ordonnanceur.

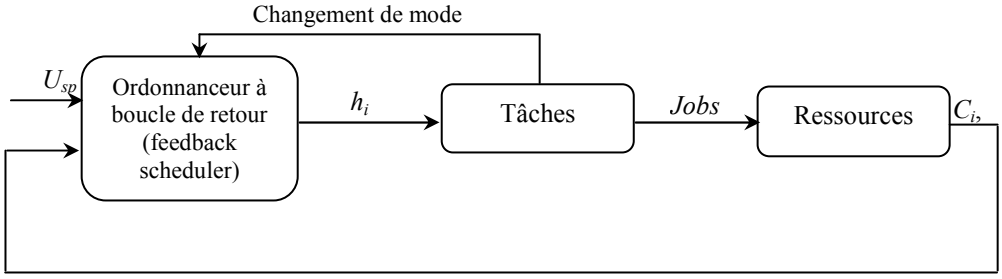


Figure 6. Architecture du feedback scheduler [Cervin et *al*. 2002]

Par analogie à un système de commande simple, nous retrouvons :

- la référence (U_{sp}),
- les signaux de mesures : temps d'exécution des tâches de commande,
- le signal de commande : période d'échantillonnage des tâches de commande (h_i),
- retour d'informations à propos du changement de mode de la commande.

Pour implanter cet ordonnanceur des considérations de conception sont prises en compte :

- le choix de U_{sp} dépend de la politique d'ordonnancement utilisée par les ressources et de la sensibilité du système aux dépassements des délais. Un U_{sp} trop petit donne de très faibles, utilisation des ressources et de performance de commande. Un U_{sp} trop grand provoque un dépassement de sa date critique d'exécution[13].
- Le feedback scheduler sera exécuté comme un système à tâches périodiques. Sa période doit être bien choisie puisque une période trop petite provoque un bon contrôle d'utilisation mais consomme beaucoup de ressources. Par contre une période trop grande provoque une réponse lente aux perturbations,
- Si les paramètres du contrôleur dépendent de la période d'échantillonnage, le contrôleur doit connaître sa période en cours et ajuter ses paramètres.

Le feeddback scheduler contrôle le taux d'utilisation du processeur en assignant les périodes des tâches qui optimisent la performance de tout le système. Les auteurs estiment le temps d'exécution des tâches comme suit :

[13] Notez que les limites garanties d'utilisation connues : 100% pour l'ordonnancement guidé par les dates de fin d'exécution critiques et 69% pour un ordonnancement guidé par la priorité [Liu et Layland 1973] ne sont pas applicables dans ce contexte puisque les dates de fin d'exécution et les périodes fixées sont violées.

$$\hat{C}_i = \lambda \hat{C}_i(k-1) + (1-\lambda)c_i \tag{5}$$

λ est le facteur dit d'absence. Un λ proche de 1, a pour conséquence une estimation lissée mais lente. Un λ proche de 0 provoque une estimation rapide mais bruitée.
L'utilisation optimale nominale est calculée avec :

$$\hat{U}_0 = \sum_{i=1}^{n} \frac{\hat{C}_i}{h_{0i}} \tag{6}$$

h_{0i} peut être en fonction du mode du contrôleur, ensuite les nouvelles périodes sont calculées comme suit :

$$h_i = h_{0i} \frac{U_{sp}}{\hat{U}_0} \tag{7}$$

Par la suite, grâce à l'information feedforward, l'ordonnanceur peut réagir aux changements soudains de la charge de travail.
Il faut noter aussi que beaucoup d'algorithmes d'ordonnancement basés sur l'idée d'une boucle de retour informant l'ordonnanceur (scheduler) de 'l'état' des tâches sont développés voir par exemple [Stankovic et *al*. 2001, Lu 2001], cependant ces auteurs n'ont pas considérés les tâches de commande.
[Sha et *al*. 2000] ont développé un algorithme d'ordonnancement guidé par la charge, c'est un algorithme d'ordonnancement en ligne dont le but est d'optimiser la performance de commande. L'algorithme ajuste la fréquence des tâches pour produire une performance optimale du système commandé soumis à une contrainte garantissant que quand le pire cas se produit aucune échéance ne doit être dépassée. L'approche proposée permet à chaque tâche de manipuler son dépassement localement, sans affecter les fréquences des autres tâches.
Quand le dépassement se produit un nouveau deadline et une nouvelle fréquence optimale de la tâche sont calculés. Les auteurs ont intégré l'algorithme proposé à la conception digitale directe (direct digital design (DDD)) et à la conception continue avec numérisation (CDD).
Dans leur travail Sha et *al*. utilisent l'algorithme EDF (Earliest Deadline First) pour ordonnancer les tâches. Pour chaque tâche un temps d'exécution pire cas et un temps de calcul normal $c_i^n \le WCET_i$ sont calculés en analysant la distribution de probabilité de la durée d'exécution de chaque tâche.
Aussi chaque tâche est décrite comme suit :

$$\tau_i\left(c_i^n, WCET_i, f_i^{\min}, \Delta J_i(f_i)\right) \tag{8}$$

Où $WCET_i$ est le temps d'exécution pire cas (durée d'exécution maximale), $f_i^{\min}$ est la fréquence minimum à laquelle τ_i peut être exécuté et enfin, $\Delta J_i(f_i)$ est l'indice de performance de perte de chaque tâche, elle peut être approximée comme suit :
$\Delta J_i(f_i) = \alpha_i e^{-\beta_i f_i}$, où f_i est la fréquence de la tâche de commande, a_i est le coefficient de magnitude, β_i est le taux de déclinaison ou d'affaiblissement (decay rate).
Chaque tâche doit être exécuté avant ou à l'instant où sa durée critique est écoulée $D_i^{hd} = 1/f_i^{\min}$.

Le but de cette approche est de calculer la fréquence optimale f_i^{opt}, en utilisant le temps d'exécution normal c_i^n, au lieu d'utiliser son WCET (sa durée d'exécution maximale), se basant sur l'hypothèse que chaque tâche ne dépasse jamais sa durée critique 'dure' D_i^{hd}.
Dans [Buttazzo 2000, Buttazzo et *al*. 2004], une application d'un ordonnancement élastique basé sur une estimation en ligne des temps d'exécution est développée. La période des tâches peut être ajustée d'une manière significative en se basant sur la charge de travail en cours. Cette charge est estimée en contrôlant le temps de calcul de chaque job. L'algorithme d'adaptation proposé utilise les temps d'exécution des tâches en ligne dans la boucle de retour pour accomplir l'adaptation de la charge, en ajustant la période de chaque tâche.
Dans son travail Buttazzo a caractérisé chaque tâche par quatre paramètres : une durée d'exécution C_i, une période nominale T_{i0}, une période minimum $T_{i\min}$, une période maximum $T_{i\max}$ et un cœfficient d'élasticité $e_i \geq 0$ qui spécifie la flexibilité de la tâche pour varier son utilisation, ceci dans le but d'adapter le système à une nouvelle configuration de taux d'utilisation faisable. Un grand e_i implique une tache très élastique. Ainsi une tâche élastique est notée comme suit :

$$\tau_i\left(C_i, T_{i\min}, T_{i\max}, e_i\right) \tag{9}$$

Avec T_i la période actuelle de la tâche τ_i est contenue dans l'intervalle $[T_{i\min}, T_{i\max}]$, chaque tâche peut varier sa période selon ses besoins, spécifiés par l'intervalle $[T_{i\min}, T_{i\max}]$. N'importe quelle variation dépend du coefficient d'élasticité et est acceptée seulement s'il existe un ordonnancement faisable (taux d'utilisation $U_p \leq 1$, avec $U_p = \sum_{i)1}^{n} \frac{C_i}{T_{i0}}$, n : nombre de tâche) pour lequel toutes les autres périodes évoluent dans leurs intervalles.

[Sename et *al*. 2003, Simon et *al*. 2005] se sont intéressés à la commande et l'ordonnancement intégré dans le cas des systèmes temps réel avec des délais induits par le réseau. La commande des ressources de calcul est abordée pour palier les variations des ressources et la charge de travail indéterminée.

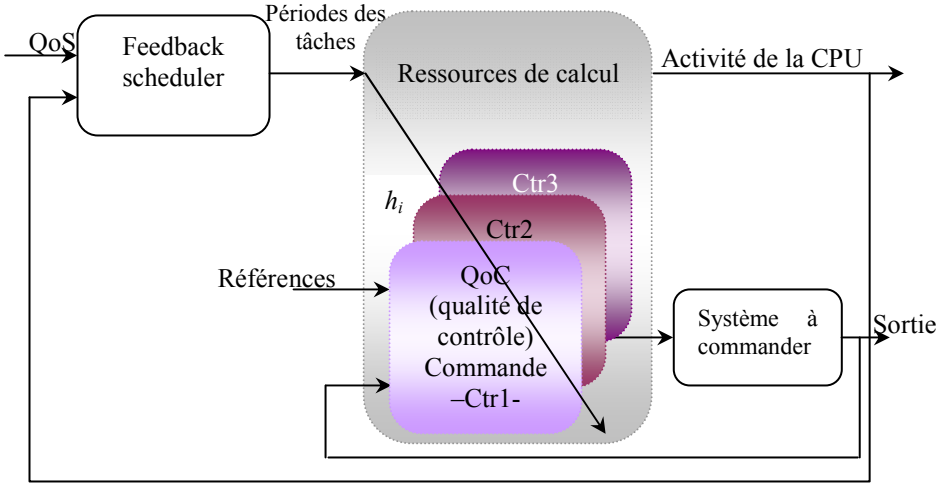


Figure 7. Structure de commande hiérarchique [Simon et *al*. 2005]

Les auteurs ont proposé une structure hiérarchique pour contrôler l'utilisation des ressources de calcul en ajustant les périodes des tâches (voir figure 7).

L'idée principale de ce travail consiste à ajouter à la commande du système une boucle appelée 'régulateur d'ordonnancement' aux paramètres d'ordonnancement de la commande. Cette action est considérée comme une mesure de la qualité de contrôle (QoC). Le critère QoC capture les exigences de performance de commande et le problème peut être considéré comme un problème d'optimisation sous les contraintes des ressources de calcul disponibles. Durant chaque expérimentation, une estimation de l'utilisation en cours est calculée :

$$U_{req} = \sum_{i=1}^{n} \frac{c_i}{h_i} \tag{10}$$

Où, c_i est le temps d'exécution estimé de la tâche i, et h_i est la période d'échantillonnage associée à la tâche i.
La nouvelle période de tâche est calculée comme suit :

$h_i^{new} = h_{ino\min al} \frac{u_{req}}{u_{sp}}$, avec u_{sp} le taux d'utilisation désiré.

Le Feedback Scheduler contrôle ainsi l'utilisation du processeur en affectant les périodes des tâches qui optimisent la performance de commande de tout le système.

3. 1. 2 Ordonnancement régulé basé sur l'état du système

[Henrikson and Cervin 2005] considèrent le problème de l'affectation optimale des périodes d'échantillonnage pour un ensemble de contrôleurs basé sur la commande linéaire quadratique (LQ). Cette idée basée sur l'allocation dynamique des ressources suivant l'état du système commandé n'est pas récente; [Marti et *al.* 2004] l'ont aussi développée. Cependant, leur travail montre un manque théorique par rapport au développement des fonctions coût dépendant des états.

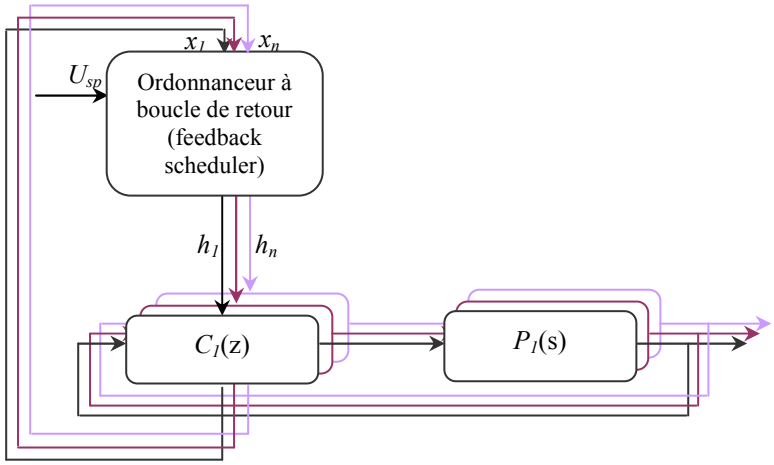


Figure 8. Allocation dynamique de ressources dans de multiples boucles de commande utilisant un Feedback Scheduler : l'ordonnanceur affecte des périodes d'échantillonnage h_1, …, h_n basées sur les états des systèmes x_1, …, x_n pour atteindre la valeur désirée U_{sp}

L'ordonnanceur développé par Henrikson et Cervin, est basé sur la performance en cours du système commandé (figure 8), c'est-à-dire sur l'état courant du système plus que sur la consommation de ressources. Ceci est fait en développant les fonctions de coût qui dépendent de l'état du système au début de la période d'ordonnancement du feedback

scheduler. Ils considèrent aussi que les ressources peuvent être distribuées d'une manière qui reflète la performance en cours du système commandé.
L'assignation de la période à la tâche est basée sur une fonction coût qui a un lien avec la période d'échantillonnage et l'état courant du système :

$$\min_{h_1,\ldots,h_n} \sum_{i=1}^{n} j_i\left(x_i(t), h_i, T_{fbs}\right) \tag{11}$$

Sous la contrainte suivante :

$$\sum_{i=1}^{n} \frac{C_i}{h_i} \le U_{sp} \tag{12}$$

Sachant que chaque tâche $i = 1,\ldots,n$ est caractérisée par une durée d'exécution C_i, une période ajustable h_i et une fonction coût $J_i\left(x_0, h, T_{fbs}\right)$. U_{sp} est l'utilisation désirée.

3. 2 Ordonnancement régulé des trames (messages)

Nous distinguons deux approches principales à savoir : l'approche qui s'intéresse à l'ordonnancement régulé en se focalisant sur un seul SCR et l'ordonnancement régulé de trames d'une structure implantant plusieurs SCRs.

3. 2. 1 Ordonnancement régulé des trames sur une structure simple (un seul SCR)

[El Mongi Ben Gaid et *al*. 2006] ont traité le problème d'ordonnancement du réseau et de la commande optimale d'un système commandé. Le réseau choisi est un réseau déterministe. L'approche proposée est basée sur une fonction coût quadratique, utilisée pour évaluer la performance de la commande et un ordonnancement adaptatif pour améliorer la performance du système commandé. Le réseau de communication est supposé limité en bande passante, dans le sens où à chaque intervalle d'échantillonnage il ne peut transmettre que b messages de commande sachant qu'il y a au total m signaux de commande à transmettre ($b \le m$).
La modélisation adoptée permet de spécifier d'une façon séparée les paramètres temporels du système de commande (période d'échantillonnage) et les paramètres correspondant à la capacité du réseau (la bande passante). La description des contraintes de la transmission des signaux de commande allant vers les actionneurs est formalisée par une fonction d'ordonnancement. Elle est définie comme suit :

$$\begin{cases} \delta_i(k) = 1, & si\ u_i(k)\ est\ mis\ à\ jour\ à\ l'instant\ k \\ \delta_i(k) = 0, & Sinon. \end{cases} \tag{13}$$

Avec k est l'instant d'échantillonnage.
La fonction décrivant la limitation en bande passante est représentée par ce qui suit :

$$\sum_{i=1}^{m} \delta_i(k) = b \tag{14}$$

Soit le vecteur de commande $v(k) \in \Re^b$ envoyé aux actionneurs via le réseau de communication à la $k^{ème}$ période d'échantillonnage et soit $u^f(k)$ le vecteur contenant b éléments de $u(k)$ (c'est-à-dire les éléments de $u(k)$ pour lesquels les indices i satisfont $\delta(k) = 1$) arrangés suivant l'ordre croissant de leurs indices. Avec :

$$u^f(k) = v(k) \tag{15}$$

Sous les contraintes suivantes :

$$\begin{cases} u_i(k) = v_j(k), & Si\ \ \delta_i(k) = 1\ \ et\ \ \sum_{l=1}^{i} \delta_l(k) = j \\ u_i(k) = u_i(k-1), & Sinon. \end{cases} \tag{16}$$

La formalisation du problème d'ordonnancement se traduit comme suit :
Soit un état initial $x(0)$, et N l'instant final, il faut trouver une séquence de commande $v^{N-1} = (v(0), \ldots, v(N-1))$ et une séquence d'ordonnancement optimale $\delta^{N-1} = (\delta(0), \ldots, \delta(N-1))$ qui minimise l'indice de performance suivant :

$$J(x,u,0,N) = \sum_{k=0}^{N-1} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T Q \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + x^T(N) Q_0 x(N) \tag{17}$$

Sous les contraintes suivantes :

$$x(k+1) = Ax(k) + Bu(k) \tag{18}$$

$$\sum_{i=1}^{m} \delta_i(k) = b \tag{19}$$

$$u(k) = D_\delta(k) v(k) + E_\delta(k) u(k-1) \tag{20}$$

$$L_i \le v_i(k) \le U_i \tag{21}$$

Avec Q est la matrice définie positive, D_δ et E_δ résultent de l'équation 16 :

$$\begin{cases} [D_\delta(k)]_{ij} = 1, & Si\ \ \delta_i(k) = 1\ \ et\ \ \sum_{l=1}^{i} \delta_l(k) = j \\ [D_\delta(k)]_{ij} = 0, & Sinon. \end{cases} \tag{22}$$

Et $E_\delta(k) = Diag(1 - \delta_1(k), \ldots, 1 - \delta_m(k))$, ce qui donne: $u(k) = D_\delta(k) v(k) + E_\delta(k) u(k-1)$.
La résolution de ce problème est réduite à l'exploration des séquences d'ordonnancements faisables et la résolution du problème de programmation quadratique (QP) de chaque séquence d'ordonnancement. Bien évidemment en pratique les séquences d'ordonnancement augmentent exponentiellement. Le problème peut être simplifié en remplaçant la contrainte $u(k) = D_\delta(k) v(k) + E_\delta(k) u(k-1)$ par $\delta_i(k) = 0 \Rightarrow u_i(k) = u_i(k-1)$.
Ceci n'enlève rien à la difficulté de la résolution, car dans ce cas il faut aussi transformer l'équation précédente en inégalité linéaire.
Les auteurs ont transformé le problème de commande optimale et d'ordonnancement intégrés en un problème d'optimisation. Cependant, ils se sont bornés à l'utilisation d'un réseau déterministe.

3. 2. 2 Ordonnancement régulé des trames sur une structure à plusieurs SCRs

[Branicky et *al*. 2002] ont quant à eux considéré un autre problème et il se pose en ces termes : Comment ordonnancer un *ensemble* de SCRs partageant le même médium de communication tout en maintenant les bonnes performances des systèmes à commander. Par ailleurs, un

SCR est dit 'ordonnançable' par un algorithme d'ordonnancement si toutes les transmissions peuvent être effectuées avant leur durée d'exécution critique (deadlines).
Dans leur travail les auteurs ont utilisés l'ordonnancement Rate Monotonic (RM) qui est un algorithme d'ordonnancement statique dans lequel la plus grande priorité est assignée à la tâche qui a la plus petite période d'exécution. Le RM peut être pré-emptif, c'est-à-dire qu'une tâche qui est en cours d'exécution, peut être interrompue par une nouvelle tâche qui possède une plus petite période. D'ailleurs Liu et Layland [Liu and Layland 1973] ont montré que le RM est optimal par rapport aux autres algorithmes statiques dans le sens où aucun autre algorithme ne peut ordonnancer les tâches comme le RM.

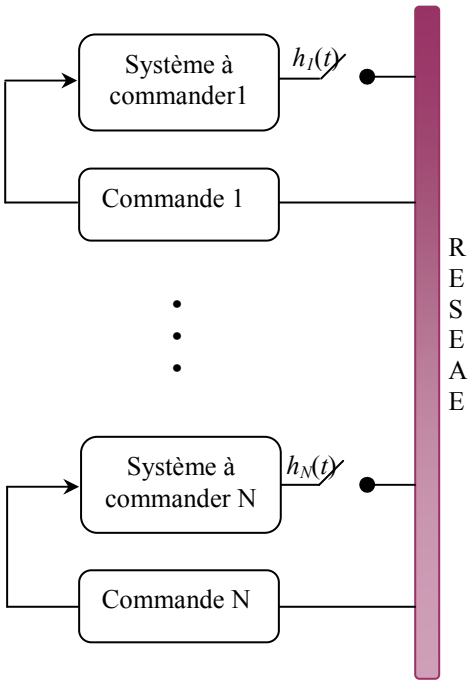


Figure 9. Ordonnancement d'un ensemble de SCRs

Le réseau utilisé est un réseau dont la résolution de conflit se base sur la priorité des messages tel que le réseau CAN.
Une extension naturelle au problème d'ordonnancement a été effectuée et qui se pose en ces termes :
Comment peut-on sélectionner un ordonnancement optimal faisable qui peut maximiser (minimiser) la mesure de quelques performances.
Le problème d'optimisation est formulé comme suit :

$$\text{Maximiser (minimiser) } J(hi) = \sum_{i=1}^{N} J_i(h_i) \tag{23}$$

Supposons que chaque SCR soit associé à une fonction de mesure de performance $J_i(h_i)$ qui donne un coût de contrôle en fonction de la période de transmission h_i.
La fonction objective est sujette aux contraintes suivantes :

- les contraintes d'ordonnancement RM ($i = 1, \ldots, N$) :

$$h_1 \le \ldots \le h_N \tag{24}$$

$$\frac{c_1}{h_1} + \ldots + \frac{c_i}{h_i} + \frac{\overline{b_{l,i}}}{h_i} \le i\left(2^{1/i} - 1\right) \tag{25}$$

- les contraintes de stabilité du SCR :

$$h_i \le h_{suff,i} - \overline{b_i}\ ,\ i = 1,\ldots,N \tag{26}$$

Avec $\overline{b_{l,i}}$ est le temps de blocage pire cas de la $i^{ème}$ tâche possédant la plus petite priorité, c'est-à-dire :

$$\overline{b_{l,i}} = \max_{j=i+1,\ldots,N} c_j \tag{27}$$

Le temps de blocage pire cas de chaque SCR est pris en compte quand on place une limite supérieure sur h_i.
La sélection de la fonction de mesure $J_i(h_i)$ est cruciale dans le problème d'optimisation. Ici les auteurs ont choisis le coût exponentiel ou quadratique déjà traité par [Bar-Noy et *al.* 1998].

4. Simulation des systèmes contrôlés en réseau

La simulation des systèmes aide à la vérification de ses performances à différents niveaux, dans le cas des SCRs cela aide à la vérification des performances du système de commande et la qualité de service offerte par les réseaux.
Deux équipes ont mis au point des simulateurs aidant à la conception, l'analyse et la simulation des SCRs. L'équipe du Michigan [Lian 2001] a développé le simulateur appelé Network & Control simulator (NCsimulator) basé sur le logiciel Matlab/Simulink. Ce simulateur permet de spécifier le protocole réseau utilisé, la vitesse du réseau le nombre de réseau utilisé. Une commande optimale peut être conçue avec ce simulateur prenant en compte les retards induits par le réseau pour améliorer les performances du système commandé.
L'équipe de Suède [Henrikson et *al.* 2002] a développé un simulateur appelé TrueTime, basé aussi sur le logiciel Matlab/Simulink. Ce simulateur permet :

- la modélisation des ordonnanceurs de tâches dans les processeurs s'occupant du calcul de la commande,
- le choix du réseau à utiliser (six protocoles MAC disponibles) et l'ordonnancement des trames.

Ce simulateur est toujours en évolution, de ce fait beaucoup de travaux de recherche se basent sur ce simulateur à des fins d'analyses.
L'outil ns-2 dédié à la modélisation des réseaux a été étendu par Branicky, Liberator et Phillips [Branicky et *al.* 2003] pour prendre en compte les contrôleurs, les actionneurs et les capteurs composant le SCRs. Cette idée a été reprise par Hartman [Hartman 2004] qui réutilise les résultats de simulation sous ns-2 dans une simulation Matlab/Simulink du système commandé.

Cette notion de simulation du réseau et du système commandé s'appelle : la co-simulation.

5. Synthèse

Les travaux qui intègrent la notion de co-conception présentés dans les sections précédentes s'intéressent, soit à :

- l'ordonnancement régulé des tâches des calculateurs basé soit sur l'état du système ou sur l'état de la tâche de commande.
- l'ordonnancement régulé des trames,

Dans le premier cas l'influence du réseau n'est pas considérée et dans le second cas l'influence de l'état de la tâche de commande ou de l'état du système n'est pas prise en

compte. Par conséquent, il serait intéressant de prendre en compte les deux types d'ordonnancement afin d'améliorer la performance des SCRs.
D'ailleurs, dans la présentation de Mouney et Juanole [Mouney and Juanole 2006] dans le cadre de l'ANR Safe-Necs, les auteurs ont énuméré les éléments contribuant à la conception conjointe (ou encore le co-design).
Les auteurs ont fait ressortir plusieurs points essentiels qui portent sur des différentes problématiques que posent les SCRs et leur supervision comme le problème de partage de ressources. Ils proposent alors d'étudier l'ordonnancement des tâches de commande implantées dans un calculateur, l'ordonnancement des messages envoyés dans le réseau avec comme objectif principal d'atteindre un ordonnancement régulé des tâches et des messages en fonction des performances des applications. Une autre problématique est de pouvoir proposer des outils de supervision des SCRs fournissant globalement l'état des applications, des calculateurs, réseaux,…) (figure 5). Dans le projet Européen NeCST, des logiciels de supervision de SCR ont été développés par la société SAE-Automation permettant de collecter des informations réseau basées sur le protocole SNMP et des informations applicatives via le standard OPC. Ce logiciel permet notamment d'interfacer OPC et SNMP pour faciliter le couplage entre le process (et sa commande) et le réseau.

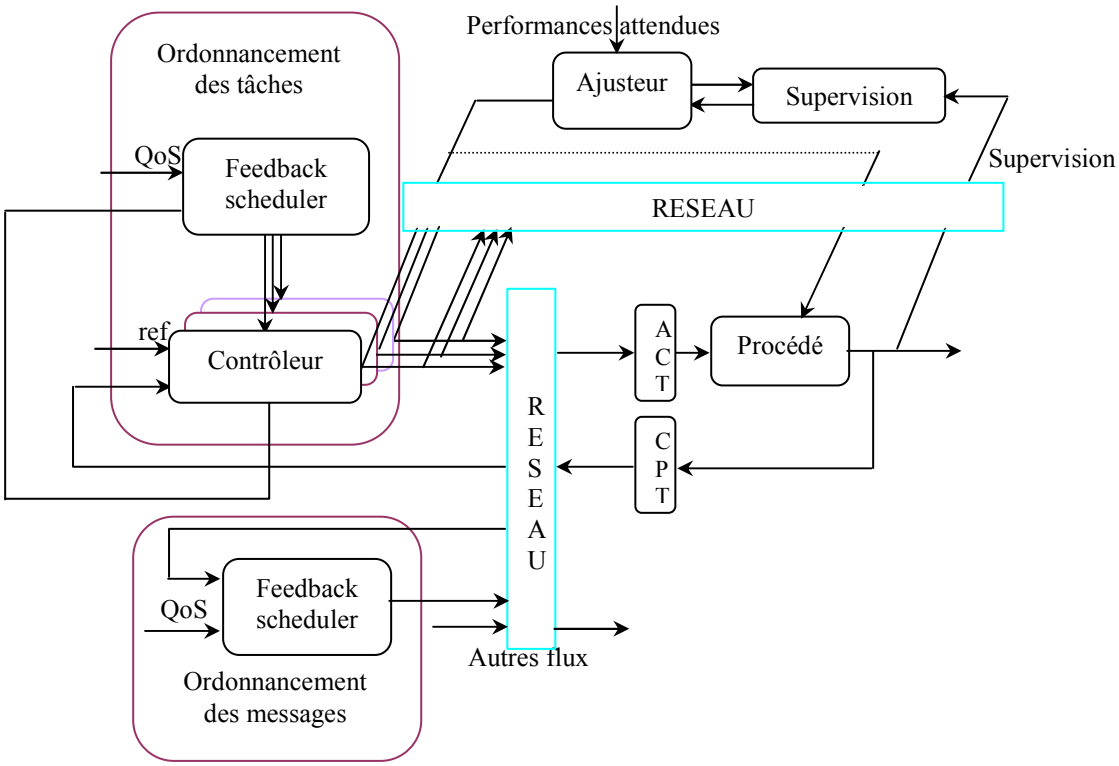


Figure 5. Les différentes boucles intervenant dans un système de commande en réseau tolérant aux défauts.

6. Conclusion

Dans ce chapitre, nous avons vu que pour aborder les SCRs, il fallait prendre en compte beaucoup de paramètres : des paramètres de communication et des paramètres du système à contrôler. Et, il n'existe pas encore de solutions globales qui prennent en compte simultanément une interaction entre modélisation du réseau et de la commande du système. De plus, les travaux dans le domaine des SCRs, reposent la plupart du temps sur des réseaux déterministes.

L'objectif de nos travaux est donc de proposer une approche de modélisation intégrée associant à la fois le comportement du réseau et le comportement du système pour permettre une évaluation globale du SCR. Pour faciliter, la représentation de ces deux comportements, nous n'utiliserons qu'un seul type de modèle : les Réseaux de Petri pour éviter les problèmes classiques mais complexes d'approches multi-modèles, ou de modélisations hybrides. Notre approche reposera sur une modélisation modulaire, hiérarchisée favorisant l'évolutivité et l'intégration de nouveaux composants : protocoles, contrôleurs, process,… au sein de notre modèle initial. L'originalité de notre recherche sera non seulement de pouvoir intégrer dans un même environnement l'aspect réseau et système pour pouvoir évaluer un SCR globalement, mais surtout de pouvoir commander le réseau. Le but est de pouvoir ensuite étudier la coordination d'actions de compensation à la fois sur le réseau et sur le process. En effet, actuellement les co-simulateurs permettent d'intégrer les données réseau dans l'étude des SCRs, mais le plus souvent, le réseau est considéré comme passif et ne peut pas être contrôlé.

Chapitre III :
Modélisation du réseau Ethernet commuté

1. Introduction

Nous avons vu dans le chapitre précédent que les travaux de recherche, dont le but est de traiter le retard (*i.e.* le réduire) ou la perte d'information dans les systèmes contrôlés en réseau sont nombreux. Cependant, d'après une constatation générale nous avons observé qu'il n'y a pas de méthodologies intégrées qui traitent les problèmes inhérents aux SCRs, d'abords en terme de modélisation, et ensuite, en terme d'analyse et d'évaluation de performances.
L'objectif de ce chapitre est donc de proposer un modèle permettant de représenter des comportements de communication pour pouvoir les évaluer en fonction de paramètres de commande du réseau basés sur des politiques d'ordonnancement. Le but sera ensuite dans le chapitre suivant de pouvoir y associer les comportements du système contrôle/commande de façon à proposer une démarche intégrée d'évaluation des Systèmes Contrôlés en Réseau.
Notre étude se focalise sur le réseau Ethernet commuté car son utilisation pour les SCRs va grandissante. De plus, ce choix réside dans la continuité des travaux effectués au CRAN [Rondeau et *al*. 2001, Adoud 2002, Krommenaker 2002, Georges 2005]. La modélisation du réseau Ethernet commuté repose sur les réseaux de Petri colorés temporisés et hiérarchique (RDPCTH).
Dans un premier temps, nous présentons les arguments qui font de l'Ethernet commuté un bon candidat pour les SCRs. Ensuite la modélisation d'un commutateur Ethernet est décrite. Des évaluations temporelles de l'Ethernet commuté sont finalement montrées en regard des politiques d'ordonnancement

2. Ethernet Commuté

2. 1 Introduction

Le contexte de notre travail est les systèmes contrôlés en réseaux. Nous avons souligné dans le chapitre 2 que peu de travaux de recherche portent sur utilisation d'Ethernet, et d'Ethernet commuté dans les SCR. Et encore moins d'études [Georges et *al*. 2007, Georges et *al*. 2006] portent sur l'analyse des retards induits par le réseau Ethernet dans une approche de co-conception des SCR.
Dans [Lian et *al*. 2001, Lian 2001] Lian, Moyne et Tilbury ont fait une analyse de performances des réseaux : Ethernet, ControlNet et DeviceNet dans le contexte des SCR. Dans leur travail, ils ont divisés le retard généré dans les SCR en trois parties : le délai induit par la source, le délai induit par le réseau et celui induit par la destination. Le premier et le dernier dépendent des caractéristiques des ordinateurs utilisés, et sont considérés constants. L'étude s'est alors concentrée sur le délai induit par le réseau qui est lui-même subdivisé en trois parties à savoir : le délai de blocage (blocking time), le délai de la trame et le délai de propagation : chaque délai est spécifique au réseau utilisé. Ensuite, deux cas de simulations sont étudiés : un système contrôlé en réseau avec 10 nœuds et des données de 8 octets émises périodiquement sur le réseau, puis une application embarquée dans un véhicule SAE avec 53 nœuds. Les travaux de [Tindell et *al*. 1995] sont utilisés pour calculer le temps de réponse des messages dans un réseau CAN. Ces systèmes sont simulés en utilisant le logiciel Matlab. Les résultats obtenus montrent que bien qu'Ethernet possède l'avantage d'avoir un débit allant jusqu'à 1 Gb/s, son utilisation n'est pas appropriée comparé à d'autres réseaux déterministes. DeviceNet et controlNet semblent mieux convenir pour les systèmes de commande temps réels. [Ji and Kim 2005] ont montré en se basant sur des résultats expérimentaux qu'Ethernet peut être utilisé dans des systèmes de commande temps réels. Cependant des tests n'ont pas été effectués dans un cas de congestion ou lorsque le réseau

est partagé par d'autres applications. [Lee and Lee 2002] ont évalué les performances du réseau Ethernet commuté dans le contexte des SCR en se basant sur des tests expérimentaux. Les auteurs concluent que la performance du système commandé est très peu affectée par le délai induit par le réseau et que l'Ethernet commuté représente un bon compromis en terme de qualité/prix pour l'industrie.
Les résultats de ces différentes recherches donnent des conclusions contradictoires et sont fonction du point de vue des auteurs. Par exemple, si on considère les performances du réseau Ethernet dans le cadre d'une utilisation industrielle où les données à échanger sont très petites, de l'ordre de quelques octets, alors qu'une trame Ethernet doit faire au moins 64 octets, on peut en conclure qu'Ethernet est mal adapté car il va consommer beaucoup de bande passante pour transporter très peu d'information utile. Dans ce cas, il est préférable de choisir des réseaux industriels avec des entêtes plus petites. Maintenant, si on considère les performances des SCR basées sur de l'Ethernet dont la bande passante est plus importante que celle des réseaux industriels classiques, alors les conclusions seront contraires.
Notre choix sur Ethernet est plutôt guidé par les offres commerciales actuelles sur cette technologie qui sont de plus en plus fournies en regard des autres protocoles de communication.

2. 2 Un peu d'histoire…

Ethernet, à ses débuts était réservé aux communications entre ordinateurs et imprimantes (bureautiques). Actuellement, il s'impose de plus en plus comme un standard de communication pour les instruments de mesures et les équipements d'automatismes, car Ethernet a gagné en vitesse et en robustesse, et il s'intéresse de plus en plus au temps réel [Peyrucat 2005].
A la fin des années *70* les ingénieurs de **Xerox** mirent au point le réseau Ethernet. Le protocole utilisé se distingue d'emblée des protocoles classiques de type maître esclave ou à jeton, où une station ne peut émettre que quand elle reçoit un signal d'autorisation. Avec Ethernet, il n'y a pas d'autorisation à attendre. Avant d'émettre une station doit écouter le réseau pour s'assurer qu'il n'y a pas de communication en cours, c'est le protocole CSMA/CD. Le point faible de ce protocole est son indéterminisme, car dans le cas où plusieurs stations veulent émettre en même temps, un mécanisme de collision rentre en jeu faisant avorter la tentative de communication. Chaque station fait une nouvelle tentative après un temps d'attente aléatoire. Cependant le risque de collision persiste toujours. En septembre 1980, le premier standard Ethernet est publié. Les sociétés Intel et Digital Equipment Corporation (DEC) ont rejoint en 1979 Xerox pour produire un standard utilisable par tous. Ce standard est baptisé DIX standard (il correspond à la version 10 Base 5 'Ethernet épais'). Les premières cartes Ethernet sont apparues avec la version 2.0 du standard DIX en novembre 1982 : le standard Ethernet II.
En *1983*, la première norme Ethernet est publiée par l'*Institut of Electrical and Electronic Engineer* (IEEE)[14], plus précisément par le sous-comité IEEE 802.3[15] . En 1985, l'IEEE publia la norme IEEE 802.3a correspondant à l'Ethernet 'fin'. En 1987, l'utilisation des fibres optiques deviennent effectives avec la norme 802.3 d.
Par la suite Ethernet connu deux tournants majeurs, le premier survient vers les années 1990 avec l'apparition des commutateurs (ou Switchs). C'est une avancée majeure par rapport aux concentrateurs (ou hubs) utilisés jusque-là. Le hub renvoyait les paquets entrants vers toutes les branches qui lui sont associées. Lorsque le réseau est fortement sollicité, les collisions sont multiples. Le commutateur évite cela en s'appuyant sur le principe de commutation des

[14] http://www.ieee.org/
[15] http://grouper.ieee.org/groups/802.3/index.html

standards téléphoniques. Les informations transitent directement du port source vers celui de la destination. Par conséquent, les risques de collision diminuent. Ensuite dans la même année, la norme IEEE 802.3i est publiée. Cette norme utilise les câbles de paires torsadées cuivre sur une topologie en étoile. En 1993, la norme IEEE 802.3j est venue étendre l'application de la topologie étoile sur fibre optique.
Le second tournant majeur dans l'évolution d'Ethernet est son passage au 100 Mbps. C'est la norme IEEE 802.3 u datant de 1995, plus connue sous le nom de *FastEthernet*.
En 1997 le mode *full-duplex* a été défini sous la norme IEEE 802.3 x. De 1998 à présent, les améliorations apportées à Ethernet se sont concentrées sur l'augmentation de son débit. Donc on a vu la sortie en 1998 du *Gigabit Ethernet* sous la norme de IEEE 802.3 z. Ensuite, en 1999, l'utilisation du Gigabit Ethernet a été définie sur les câbles en paires torsadées : la norme 802.3 ab.
Par la suite, les débits ont été une nouvelle fois multipliés par 10 pour atteindre les 10 Gbps avec la publication de la norme 802.3 ae en 2002. Entre 2005 et 2006 le 10 Gbps est défini sur les paires torsadées cuivre : c'est la norme 802.3 an. Et le Térabit est à venir [Latu 2007].
Il est vrai qu'en général les applications temps-réel critiques utilisent des réseaux de communication tels que PROFIBUS [Profibus 1992], FIP (Factory Instrumentation Protocol) [FIP 1989] et CAN (Controller Area Network) [CAN 1993]. Ces réseaux ont une spécificité commune qu'est le déterminisme de leur mécanisme d'accès au médium,. Cependant, les débits important dont bénéficie Ethernet et la micro-segmentation avec les commutateurs full-duplex [Seifert 1995, Rüping et *al*. 1999] combinés avec des politiques d'ordonnancement temps réels, permet désormais l'utilisation d'Ethernet dans des applications temps réel dures.
De plus, toutes les autres technologies de transmission qui ont cherché à augmenter les débits en faisant évoluer leur matériel, a eu pour conséquence des réinvestissements lourds, l'exemple de la technologie ATM (Asynchronous Transfer Mode) et de FDDI (Fiber Distributed Data Interface) sont caractéristiques. Or, il est évident qu'Ethernet est très attractif de ce point de vue, puisqu'il est peu coûteux, stable, mature, et en perpétuelle évolution. Ceci a fortement contribué à son utilisation dans les applications industrielles.

2. 3 Quelques solutions pour renforcer le déterminisme d'Ethernet

Avant d'aller plus en avant dans la définition et les généralités du réseau Ethernet commuté, nous allons évoquer les contributions de quelques équipes de recherche dans l'amélioration du réseau Ethernet et de son utilisation dans les SCRs.
L'équipe de Alves dans [Alves et *al*. 2000], l'équipe de Pedreiras [Pedreiras and Almeida 2005] et de Wang [Wang et *al*. 2002] ont dressé un panorama intéressant sur les améliorations effectuées sur Ethernet. Quelques travaux portent sur les améliorations de l'algorithme d'accès au médium (CSMA/CD), et ont été déjà cités dans le chapitre 2. Dans cette section, nous allons en présenter d'autres. Puis, nous nous attarderons sur les travaux qui traitent des réseaux Ethernet commuté.
Les deux premières propositions analysées sont RETHER (Realtime ETHERnet) [Venkatramani and Chiueh 1994], et TEMPRA (TimEd Mechanism Packet Release Access) [Pritty et *al*. 1995]. Dans RETHER, le réseau Ethernet se trouve dans un mode CSMA/CD quand il traite un trafic non temps réel (NRT). Si un nœud a besoin d'envoyer un message temps réel (RT), il envoie une requête pour demander un changement de mode, c'est-à-dire passer du mode CSMA/CD au mode temps réel et attend un accusé de réception de tous les nœuds. Cependant, ce protocole est inefficace lorsque les nœuds de réception possèdent des messages dans la phase de réémission –*i.e*. phase *back-off*- (dans le cas où auparavant il y avait collision).

Comme RETHER, TEMPRA possède deux phases de fonctionnement (RT et NRT), le mode temps réel est basé sur le mécanisme d'accès synchronisé sur l'instant d'envoie du paquet, qui commande l'envoie du paquet en utilisant le slot (une référence de temps commune) envoyé par un nœud moniteur. Cette méthode manque d'équité dans le sens où les délais d'accès au réseau sont fonction de la distance du nœud moniteur sachant que le nœud le plus proche possède la plus grande priorité. Plus encore, ces protocoles nécessitent un matériel et un logiciel dédiés non-compatibles avec Ethernet.

D'autres façons d'améliorer la communication temps-réel sur un bus partagé sont basées sur l'utilisation de l'approche maître/esclave, dans lequel un nœud 'le maître', contrôle l'accès au médium de tous les autres nœuds 'les esclaves'. Ainsi le délai du trafic se limite à un problème d'ordonnancement qui est localisé au niveau du maître. Cependant, cette approche mène inexorablement à une sous-exploitation de la bande passante du réseau par le trafic utile. L'un des protocoles qui utilise ce principe est le protocole FTT-Ethernet (Flexible Time Triggered- Ethernet) [Pedreiras and Almeida 2002, Pedreiras et *al*. 2002]. Il combine la technique de transmission maître/multi-esclaves avec un ordonnancement centralisé, maintenant ainsi les exigences de communication et la politique d'ordonnancement localisées dans un seul nœud, 'le maître'. Des durées correspondant à des intervalles de temps fixes appelées cycles élémentaires sont allouées au trafic. Ces durées élémentaires sont décomposées en deux phases : les fenêtres synchrones (synchronous) et asynchrones (asynchronous) qui ont des caractéristiques différentes. La fenêtre synchrone transporte un trafic périodique guidé par le temps (time-triggered) – *i.e*. trafic temps réel- qui est ordonnancé par le maître. La fenêtre asynchrone transporte un trafic sporadique, c'est-à-dire qu'il s'agit des messages de contrôle du protocole, ou les messages guidés par les événements (event-triggered). Ce trafic sporadique correspond au trafic non temps réel. Il y a une isolation temporelle stricte entre les deux phases. Ainsi, le trafic sporadique n'a pas d'impact sur le trafic temps réel. Ce protocole montre une flexibilité et un respect des propriétés temporelles du trafic. Néanmoins, il demeure des inconvénients liés à la charge de calcul exigée pour que le maître puisse ordonnancer les messages et effectuer l'analyse en ligne de l'ordonnancement. Aussi, une redondance du maître s'impose, pour assurer le fonctionnement du système dans le cas où le nœud 'maître' principal tombe en panne.

Une amélioration du protocole CSMA/CD a été proposée dans [Lann and Rivière 1993] et elle a pour but de modifier l'algorithme de résolution des collisions. L'idée est de fusionner le P-CSMA et le CSMA/DCR (c.*f*. chapitre II) donnant lieu au CSMA/PDCR (CSMA with Priority Deterministic Collisions Resolution). Cependant contrairement au P-CSMA, le CSMA/PDCR intervient lorsque la collision se produit, ensuite sa résolution est similaire au DCR –*i.e*. basée sur un arbre de recherche binaire[16]-. D'autres méthodes sont largement expliquées dans le chapitre II.

Par ailleurs, il y a d'autres méthodes qui sont développées dans le cadre des protocoles Ethernet, dont le but est de pallier l'effet de famine et donc de renforcer l'équité de la méthode d'accès au médium. Cette classe de méthode convient aux applications temps réel à contraintes molles. Les méthodes les plus simples sont : le CABEB (Capture Avoidance BEB[17]) proposé par Ramakrishan et Yang dans [Ramakrishan et Yang 1994], le BLAM (Binary Logarithmic Arbitration Method) proposé par Molle dans [Molle 1994] et enfin le BEB2 proposé par Christensen dans [Christensen 1998].

[16] Le lecteur intéressé pourra se référer à l'article [Pedreiras and Almeida 2005] dans lequel la méthode est expliquée.
[17] BEB: Binary Exponential Backoff.

La différence majeure entre les deux premières propositions réside dans le fait que le CABEB utilise un compteur de collision distinct pour chaque station et que le BLAM utilise un compteur global et il définit un canal qui maintient un temps limite pour chaque station. Le BEB2 est basé sur l'algorithme BEB. Les changements ont été effectués dans le gap inter-trames (le gap). En effet à la détection de la première collision, les transmissions de paquets sont dotées d'un double gap jusqu' à l'apparition de la seconde collision. De plus, une autre amélioration de l'algorithme réside dans la sélection adaptative du temps maximum backoff-limit: temps au delà duquel les trames ne sont plus réémises.

Il est aussi intéressant de mentionner trois autres méthodes qui traitent du problème de l'équité dans Ethernet sans introduire de changements notables dans le protocole. On peut citer : les protocoles à temps virtuels, les protocoles basés sur la notion de fenêtre, et le lissage de trafic (déjà mentionné et expliqué au chapitre 2). Ces méthodes exécutent une fonction d'ordonnancement locale suivant une connaissance globale de l'état du réseau, augmentant ainsi l'équité dans l'accès au réseau et réduisant le nombre de collision en utilisant des critères de priorité.
Les protocoles à temps virtuels (Virtual Time Protocol) [Zhao and Ramamritham 1986, El-Derini and El-Sakka 1990] implémentent différentes politiques d'ordonnancement en affectant des temps d'attentes aux messages qui seront transmis. Le trafic sur le bus est ainsi coordonné suivant des temps d'attente qui respectent un ordre. Cet ordre est approximé par une politique d'ordonnancement choisie. Ce mécanisme est flexible dans le sens où toutes les politiques d'ordonnancement temps réel peuvent être implantées, à savoir, les politiques d'ordonnancement statique tels que le Rate Monotonic (RM) et le Deadline Monotonic (DM), ou les politiques d'ordonnancement dynamique, tels que le Earliest Deadline First (EDF).
Les protocoles à fenêtres [Kurose et *al*. 1984, Malcolm and Zhao 1995] sont basés sur les principes suivants : les nœuds sur le réseau s'accordent sur un intervalle de temps, appelé : fenêtre. La taille de la fenêtre change dynamiquement suivant l'état du canal – *i.e*. libre, occupé ou en état de collision. L'approche la plus basique arbitre l'accès au canal suivant la politique FIFO (First In First Out). D'autres protocoles du même type implantent d'autres politiques d'ordonnancement basées sur la priorité ou la deadline.
Enfin, le lissage de trafic [Kweon et *al*. 1997] dont le principe est de calculer en utilisant une méthode statistique un seuil du temps d'accès au médium limitant ainsi le taux d'arrivée des paquets à la couche MAC. [Kweon et *al*. 1999] utilise un régulateur de trafic (traffic smoother) placé entre la couche de transport et la couche liaison de données d'Ethernet. La régulation se fait à partir d'une analyse probabiliste de la transmission réussie d'un paquet à chaque tentative. Les seuils du trafic sont calculés en utilisant une modélisation semi-markovienne du CSMA/CD et de l'algorithme BEB. D'autres méthodes découlant des travaux de l'équipe de Kweon tels que [Lobello et *al*. 2005, Caponetto et *al*. 2002] utilisent la commande de lissage (fuzzy controller) (*c.f*. chapitre II).

2. 4 Réseau Ethernet commuté : comment ça marche ?

L'intérêt porté au réseau Ethernet commuté va en grandissant. Et les améliorations apportées ont pour but d'augmenter la vitesse du réseau, l'isolation de trafic et réduire l'impact de l'indéterminisme dont l'origine vient du CSMA/CD. Dans un réseau Ethernet commuté, la configuration qui est adoptée est celle d'une topologie en étoile, avec un commutateur central dont les ports sont reliés à un seul équipement, ce qui constitue un LAN différent pour chaque port. L'interconnexion avec les autres éléments est réalisée en reliant les commutateurs entre eux. Cette topologie a l'avantage de réduire par segmentation les

domaines de collision au seul lien point à point entre un équipement et son commutateur, ou bien entre deux commutateurs.
Dans un réseau Ethernet commuté l'équipement le plus important est bien sûr le commutateur. La norme qui définit les commutateurs Ethernet est le 802.1 D [802.1 D]. Trois fonctions régissent le commutateur :

- le relai et le filtrage des trames,
- la mise à jour des informations permettant de remplir le rôle précédent,
- une surveillance de son fonctionnement interne.

Quand un message arrive à un port du commutateur, il est mis en mémoire dans un buffer, puis analysé et classifié. Ensuite, le message est mis dans le buffer correspondant à son port de destination (figure 1). Le bloc 'manipulation de paquet' dans la figure, appelé fabrique de commutateur (switch fabric) transfère les messages du port d'entrée au port de sortie. Quand le taux d'arrivée des messages de chaque port qu'il soit d'entrée ou de sortie est supérieur au taux de départ, les messages sont mis dans une file d'attente. Les messages mis dans une file d'attente sont transmis d'une façon séquentielle suivant la politique FIFO (First In First Out). La file d'attente FIFO peut mener à un délai induit par le réseau puisque les messages temps réel peuvent être bloqués durant la transmission de messages non prioritaires. Pour remédier à cela, l'utilisation de files d'attente parallèle en sortie de chaque port du commutateur a été proposée dans la norme IEEE 802.1p, qui est actuellement intégrée dans IEEE 802.1D. Le nombre de niveau de priorité est limité à 8, permettant l'utilisation de politiques d'ordonnancement.
En somme, lors d'une réception de trame, le commutateur accomplit les opérations suivantes :

- Réception de la trame,
- Destruction des trames corrompues, ou de celles qui ne sont pas du bon type (Data_User),
- Filtrage de la trame suivant les informations de filtrage : il y a destruction si nécessaire,
- Aiguillage vers les autres ports du commutateur,
- Choix de la classe de la trame, puis mise en place dans la file d'attente correspondante,
- Destruction des trames ayant attendu trop longtemps,
- Sélection de la trame à transmettre,
- Emission de la trame.

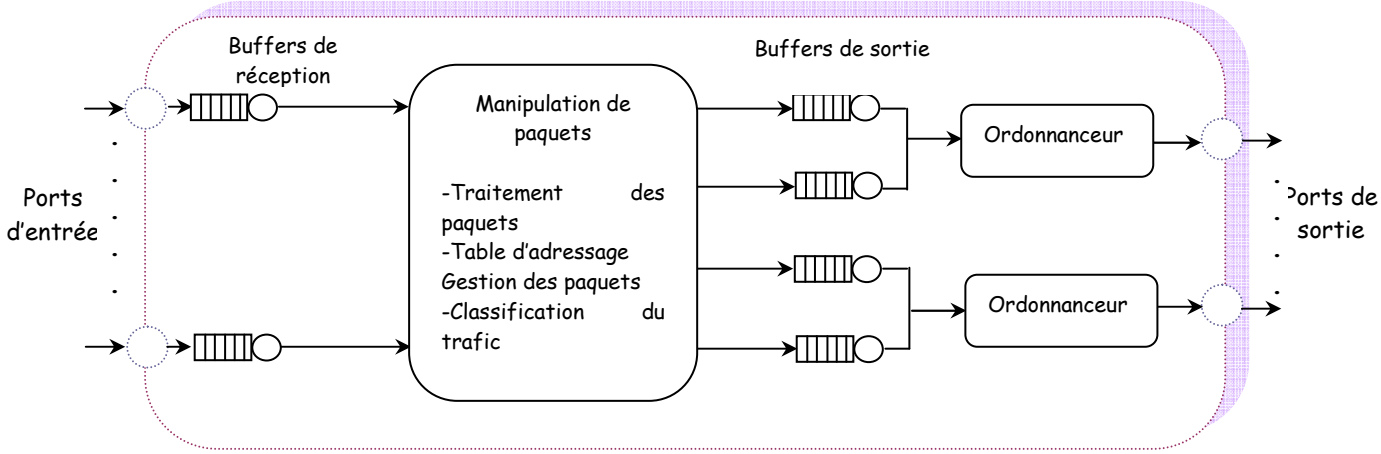


Figure 1. Architecture interne d'un commutateur [Pedreiras and Almeida 2005, Marau et *al.* 2006]

2. 5 Réseau Ethernet commuté full-duplex

Nous avons vu que dans une architecture Ethernet commuté, les seules collisions possibles se situent au niveau des liens point à point. La solution qui a été développée consiste à utiliser des liens bidirectionnels, qui opèrent selon le mode full duplex prévu dans la norme IEEE 802.3. Le mode full duplex permet une communication simultanée entre deux stations reliées en point à point par un médium physique dédié. Ce mode d'opération ne nécessite pas de retarder l'émission d'un message, ni d'écouter ou de réagir à l'activité sur le médium physique, puisqu'il n'y a pas de collision possible sur ce médium. On appelle ce type de réseau, un réseau Ethernet commuté Full-Duplex. Ces réseaux ont l'avantage de ne plus avoir de perte de trames à cause des collisions.
Cependant le fait qu'il n'y ait plus de collision dans un réseau Ethernet commuté full duplex, ne le rend pas pour autant déterministe. En effet, le problème de collision est décalé en un problème de congestion dans les commutateurs. Si trop de trafic se dirige à un instant donné vers un seul port de sortie, une congestion est possible ce qui peut impliquer des pertes de trames. On pourrait également voir apparaître des rafales de trafic, dues à un encombrement passager d'un port du commutateur. Ces rafales risquent à leur tour d'encombrer le commutateur suivant, entraînant également des retards et des pertes de trames.

2. 6 Quelques solutions pour renforcer le déterminisme dans Ethernet commuté full-duplex

Des solutions techniques, ont été proposées par exemple la norme IEEE 802.1 Q [IEEEvlan 98] qui définit la notion de réseau local virtuel (VLAN). Le VLAN offre la possibilité de créer de multiples sous-réseaux dans le commutateur Ethernet. Ainsi, il est possible d'isoler virtuellement un trafic de commande d'autres trafics de données des autres utilisateurs. Le VLAN peut aussi améliorer les performances du réseau en séparant les éléments possédant une haute priorité des éléments de priorité inférieure. Par exemple, dans le cas d'une trame de diffusion, elle est retransmise à tous les ports de sortie de commutateur. L'utilisation des VLANs pour segmenter virtuellement le réseau, permet de diffuser une trame seulement vers les ports du même VLANS.
D'autres solutions techniques qui consistent à réduire la charge des commutateurs ont été aussi proposées. Ces méthodes filtrent le trafic multicast (IGMP snooping). En effet, beaucoup d'applications industrielles utilisent le trafic multicast, ce qui peut réduire la performance du réseau. Pour pallier ce problème, l'IGMP Snooping peut être déployé limitant ainsi le problème d'inondation (flooding) causé par le flux multicast. Dans ce cas, quand un message multicast est envoyé au commutateur, celui-ci transfere le message seulement aux interfaces qui sont intéressées par le trafic. Par conséquent cela réduit la charge du réseau et soulage les récepteurs qui n'ont pas besoin de ces trames.
Une autre solution technique consiste à utiliser le mécanisme de la classification de service (CoS) pour améliorer les performances du commutateur Ethernet, en contrôlant et en gérant le trafic dans l'environnement industriel. Dans le réseau Ethernet industriel, différent types de trafic peuvent être transmis : du trafic de données, du trafic de commande critique, et du trafic vidéo/voix. Le réseau doit pouvoir distinguer ces différents trafics en leur assignant une priorité. Pour réaliser ceci différentes techniques de CoS peuvent être implémentées. Ces techniques sont capables de différencier les trafics en utilisant des méthodes de classification, ou d'écarter des paquets dans les buffers pour prévenir le problème de congestion, ou d'ordonnancer les paquets à la sortie des buffers. La norme IEEE 802.1p [IEEE prio 93] a été définie pour associer à chaque paquet une priorité donnée et permet de filtrer

dynamiquement le trafic multicast. Huit classes de priorité sont disponibles, et exprimées dans le champ user_priority de l'entête IEEE 802.1Q ajoutée à la trame Ethernet. L'IEEE 802.1p est utilisé dans les standards IEEE 802.1D et IEEE 802.1Q. Si un commutateur implante un mécanisme de qualité de service alors le flux de donnée dans le cas présent passe par plusieurs phases :

1- à la réception de la trame, le commutateur peut déterminer la priorité de cette trame, soit de manière explicite c'est à dire à partir de l'information fournit par la trame elle-même (par exemple user_priority), ou implicitement à partir du contenu de la trame et une mise en place de règles d'administration.
2- Si la priorité de la trame est inconnue le commutateur doit déterminer la priorité de chaque trame, en associant une classe de service disponible à chaque port de sortie vers lequel la trame est envoyée.
3- Pour un ensemble de buffers de sortie, le commutateur peut appliquer quelques algorithmes d'ordonnancement pour transmettre les trames de ces buffers suivant les besoins des classes de service qu'elles représentent

Remarque

Nous avons mentionné dans les paragraphes précédents la notion de classes de service. Il est important de souligner qu'il y a une différence fondamentale entre le concept de classe de service (CdS/CoS) et celui de qualité de service (QdS/QoS). Le terme qualité de service intervient lorsqu'une application particulière répond à un certain niveau de service avant d'utiliser le réseau. Le réseau doit réserver alors les ressources appropriées (par exemple : buffers, la bande passante du canal etc…) pour la durée de session de cette application. Si ces ressources ne sont pas disponibles, l'application doit être informée et elle peut choisir de ne pas fonctionner ou de fonctionner avec une garantie de qualité de service faible. Donc, la QoS implique une connexion-orientation et une réservation des ressources du réseau afin de fournir une garantie de niveau de service minimum. La classe de service est simple, dans le sens où le réseau fournit un niveau de service supérieur aux applications prioritaires mais ne garantit pas un niveau de service spécifique. La mémoire et la capacité du canal n'ont pas besoin d'être réservées, et aucun protocole additionnel n'est exigé pour réserver les ressources ou ouvrir la session de l'application. La classe de service suppose que la capacité moyenne du réseau est suffisante pour toutes les applications. Puisque dans les LANs actuels, on rencontre des capacités de plus en plus importantes et à des prix relativement bas, l'utilisation de la CoS (en respectant l'hypothèse précédente) plus simple à implanter permet d'améliorer les surcharges du réseau à moindre coût par rapport aux mécanismes de QoS [Seifert 2000].

D'autres équipes de recherches telles que l'équipe de Hoang dans [Hoang 2004, Hoang et *al*. 2002] ont développé une technique qui traite le trafic temps réel et le trafic non temps réel dans un commutateur Ethernet. Le trafic temps réel est ordonnancé suivant la politique EDF (Earliest Deadline First) et les contraintes temps réel sont garanties suivant le mécanisme en ligne de contrôle d'admission adéquat. Les auteurs proposent une architecture qui exige l'ajout d'une couche temps réel au composant réseau placée entre les couches 2 et 3. Cette nouvelle couche s'occupe d'établir une connexion temps réel, de réaliser un contrôle d'admission, de fournir un temps de synchronisation, une gestion des messages et la réception des deux classes de trafic (temps réel et non temps réel).

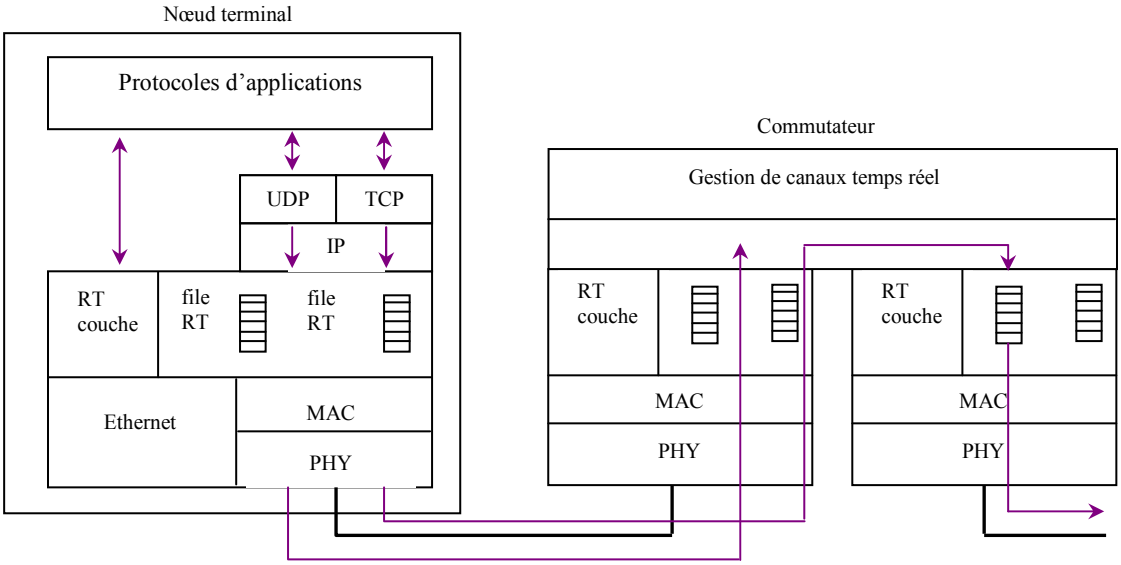


Figure 2. Architecture implémentant la couche temps réel de [Hoang et *al*. 2002]

Une analyse de faisabilité est proposée. Celle-ci est dérivée de l'analyse d'ordonnancement de tâches (EDF) de [Liu et Layland 1973], mais avec quelques adaptations qui tiennent compte des spécificités du système à savoir : les entêtes dues aux messages de contrôle et de l'impact de la transmission des messages non préemptifs. Cette méthode est intéressante cependant très complexe à mettre en œuvre. Ces travaux s'inscrivent aussi dans le contexte de la qualité de service offerte dans les systèmes temps réel distribués.

Dans le même contexte on retrouve les travaux de Koubaâ et Song dans [Koubâa and Song 2004] qui proposent la politique (m,k)-*firm* qui permet de garantir la transmission de m messages parmi k messages consécutifs. Autrement dit, cette politique garantit un seuil minimal de ressources ; ces travaux sont destinés au transfert des images vidéo. Cependant cette méthode peut être utilisée dans des applications temps réel à contraintes molles. En considérant la politique de qualité de service (m, k)-firm pour les trames dans le commutateur Ethernet, les auteurs ont modélisé la transmission des trames dans le commutateur Ethernet comme l'illustre la figure 3.

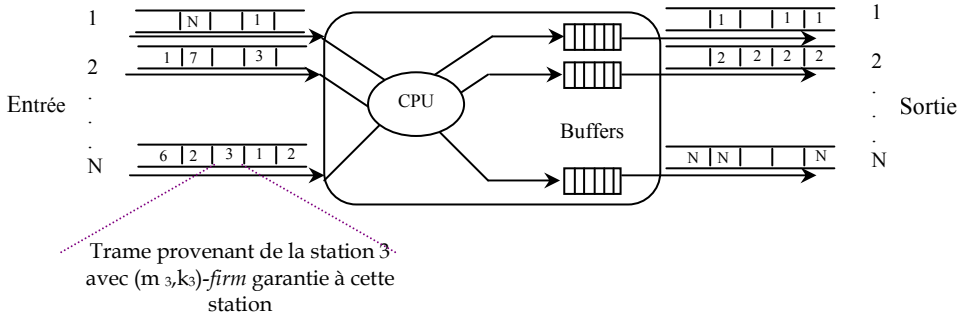


Figure 3. Modèle de transmission des trames dans le commutateur Ethernet

Dans [Grieu 2004], pour garantir la qualité de service, l'auteur a développé une méthode formelle basé sur le calcul réseau afin de prouver que le réseau Ethernet commuté offre bien la qualité de service requise dans le contexte avionique. Cette méthode calcule le délai de traversée borné des flux. L'approche utilisée impose des règles dès l'arrivée des trames. C'est-à-dire qu'un régulateur de trafic appelé seau percé est implanté dans chaque port

d'entrée des commutateurs. Cela permet de fixer une limite sur le trafic maximal reçu. Si cette limite est violée, les trames supplémentaires sont détruites. Dans [Georges 2005] basé sur la méthode du calcul réseau, l'auteur propose un modèle formel d'un commutateur Ethernet dans le but de majorer les temps de traversée du réseau. Des hypothèses sont prises au niveau des entrées du réseau pour appliquer le modèle proposé dans le contexte des systèmes contrôlés en réseau.

D'autres auteurs se basent sur une optimisation de la topologie du réseau Ethernet commuté pour améliorer les performances du réseau Ethernet commuté en termes de délai, et de perte de paquets. Nous citerons d'abords les travaux de [Ruping et *al*. 1999]. Les auteurs ont effectué une comparaison entre la topologie en bus ou linéaire, en boucle, et arborescente, appliquée dans le contexte des systèmes automatiques. En s'appuyant sur un modèle de communication basé sur le maître/esclave, le temps de cycle minimum est calculé d'abord théoriquement pour les différentes topologies. Les paramètres pris en compte sont le délai dans le commutateur, le type de protocole, le nombre d'élément raccordé au commutateur, et la taille de la trame. Les résultats obtenus montrent que la topologie arborescente outre l'avantage du nombre des commutateurs utilisés (i.e. peu de commutateurs), donne un temps de cycle plus court que les autres topologies.
Aussi [Jasperneite and Neumann 2001a, Jasperneite and Neumann 2001b] ont réalisé une étude comparative des différentes topologies du réseau Ethernet commuté en utilisant l'outil de simulation réseau OPNET. Les auteurs dans leur étude ont pris en compte le volume des données, la priorité…etc. Ils ont conclu que les performances de la topologie hiérarchique ou arborescente sont meilleures que celles de la topologie linéaire ou en bus. Les mêmes auteurs se sont intéressés à l'influence de la topologie du réseau combinée à l'utilisation des algorithmes d'ordonnancement FIFO et de la priorité stricte. Les auteurs affirment que la topologie en étoile est bien meilleure que la topologie en bus. Cependant l'utilisation d'une autre politique d'ordonnancement autre que FIFO, c'est à dire la priorité stricte améliore les performances des deux topologies dans le cas où la charge du réseau est supérieure à 50%. [Krommenaker 2002] dans son étude, présente une méthode de conception pour améliorer la performance du réseau. Le but est de minimiser le délai et augmenter la fiabilité du réseau. Cette méthode consiste en une topologie hybride utilisant la topologie arborescente et linéaire, et basé sur deux critères : le nombre de commutateurs traversé par le flux et l'équilibrage de charge dans chaque commutateur. L'auteur propose d'utiliser les algorithmes génétiques afin d'optimiser l'architecture du réseau pour de meilleures performances. D'autres travaux [Rondeau 2001, Rondeau et *al*. 2001, Adoud 2002] allant dans le même sens utilisent les algorithmes spectraux.

2. 7 Conclusion

Dans cette section, nous avons exposé un large panorama des études et améliorations apportées au réseau Ethernet commuté qui en font un réseau intéressant pour les SCR. De même, nous souhaitons dans notre travail de thèse proposer une méthode de modélisation intégrée basée sur Ethernet pour les SCR. De plus, un autre argument dans le choix du réseau Ethernet réside dans la continuité des travaux effectués au CRAN [Rondeau 2001, Rondeau et *al*. 2001, Adoud 2002, Krommenaker 2002, Georges 2005] sur ce type d'architecture.

3. Modélisation proposée du commutateur Ethernet

Avant de modéliser le réseau Ethernet commuté, il faut être capable de représenter son principal élément constitutif qui est le commutateur. Pour ce faire, la compréhension de son architecture interne s'impose.

3. 1 Architecture d'un commutateur Ethernet

Trois composants principaux constituent l'architecture des commutateurs :

1- les files d'attentes (queuing model) : se rapportent à la bufferisation (stockage temporaire des informations dans une mémoire tampon) et aux mécanismes de gestion de congestion associés. Il existe trois sortes de modèles de files d'attentes : les modèles de buffers dynamiques, les modèles de buffers fixes et les modèles basés sur la mémoire partagée,

2- l'implémentation de la commutation (switching implementation) : se dit d'un procédé de prise de décision dans le commutateur, *i.e.* où et comment la prise de décision dans un commutateur est effectuée. Deux méthodes peuvent répondre à cette question : la commutation centrale, et la commutation distribuée.

3- la fabrique de commutation (switching fabric) : se dit du chemin que la donnée prend pour passer d'un port à un autre. Il existe trois types de fabrique de commutation : l'architecture basée sur un bus partagé, l'architecture basée sur le concept de la grille (crossbar), et l'architecture basée sur la mémoire partagée.

Il existe donc plusieurs façons de construire une architecture de commutation dans les commutateurs avec les trois composants suscités. Une définition plus exhaustive des différents composants est présentée à l'annexe 1, et des exemples de technologies utilisant ces architectures sont aussi présentés.

Intéressons-nous de plus près à la fabrique de commutation basée sur la mémoire partagée, qui est la plus utilisée et la moins chère. Elle est basée sur un accès très rapide, multiple et simultané, à une mémoire partagée par tous les ports. Lorsqu'un paquet entre dans le commutateur, il est d'abord stocké en mémoire. Ensuite, il est redirigé par un ASIC vers la sortie appropriée en regardant son adresse MAC de destination dans la table de transfert. Puis, il est envoyé vers le port de sortie approprié. Des buffers de sorties sont utilisés à la place des buffers d'entrées pour éviter le problème de blocage de tête de ligne (voir Annexe 1).

Le délai total dans le commutateur est composé de :

- la latence dans le commutateur : il s'agit du temps mis par le commutateur pour classer le trafic suivant la norme 802.1p, le processus d'aiguillage vers le port de destination et le temps de la fabrique de commutation. C'est une valeur fixe qui dépend de la performance du commutateur et est souvent fourni par le vendeur. Par exemple : il peut être de 11 µs entre les ports de 100 Mbps et 70 µs entre les ports de 10 Mbps pour les Cisco Catalyst 1900, 2820. Maintenant avec l'évolution technologique ce temps de latence est négligeable [Décotignie 2005],
- le temps de transfert des trames : une trame une fois reçue dans le commutateur est mise dans une file d'attente avant d'être retransmise. cela peut être pénalisant dans le sens où un retard peut être induit par la longueur de la trame. Il existe plusieurs modes de transfert (ou de retransmission) (voir annexe 2) :

- le mode *Store & forward* : le commutateur attend la réception complète de la trame. Son avantage est de pouvoir détecter les trames erronées. Cependant, il induit un retard surtout lorsque la trame est longue,
- le mode *Cut-through* : dans ce mode le commutateur attend jusqu'à la réception de l'adresse de destination, en stockant le message éventuellement dans un buffer. Ensuite, il le transmet au port de sortie. L'avantage de ce mode est qu'il induit un délai minimal et est indépendant de la longueur de la trame, mais il risque de transmettre des trames erronées. Donc, le commutateur peut transmettre de petits fragments dus à des collisions qui n'ont pas été détectées,
- le mode *fragment-free* : équivalent au cut-through. Ce mode attend le 64ème octet de la trame avant d'être transmis. Son avantage est qu'il élimine les petits fragments.
- le mode *auto-select* : ce mode est capable de s'adapter à la qualité de transmission dans le réseau. Ce mode commence par le mode *Store & forward.* Si aucune erreur n'est produite, le commutateur passe au mode *Cut-through.* Si des petits fragments dus à des collisions sont détectés, il passe au mode *fragment-free*.

- le délai de bufferisation : apparaît quand la trame est mise dans une file d'attente, et se produit quand le port de sortie ne peut transmettre tous les messages d'entrée à temps. Ce délai dépend de la connaissance du trafic d'entrée. Pour le trafic périodique une analyse classique ou une analyse basée sur le calcul réseau [Georges et *al.* 2007] peut donner un délai pire cas de bufferisation, fournissant ainsi des garanties temps-réel dures. Pour les trafics apériodiques dont seulement quelques caractéristiques sont connues, une analyse stochastique s'impose.

Dans notre travail nous utiliserons le mode *Store & forward*, au lieu et de sélectionner le *Cut-through*, puisqu'il répond à l'exigence que nous avons posée, à savoir, que les trames ne doivent pas être erronées. De plus, selon [Seifert 2000] : [*…Let's cut through all the non sens ans hype-the bottom line is there is really no significant benefit* (*or detriment*) *to cut-through switching over the store and forward approach. Virtually all of the controversy was artificially created by the people marketing cut-through switches, as a way to differentiate their products…Cut-Through has become a data sheet "checklist" item, with no practical significance.*]. Le modèle de coopération utilisé est 'le modèle producteur/consommateur ', largement utilisé dans l'industrie temps-réel.

3. 2 Spécifications du commutateur à modéliser

Le commutateur modélisé dans notre travail est un commutateur Cisco de la série Catalyst 2950XL. Il est basé sur la mémoire partagée de 8 MB. Dès qu'une trame est reçue en entrée, elle est directement recopiée dans une mémoire partagée globale[18], et ce pour éviter le problème de blocage de tête de ligne (HOL). Aussi, le taux de transfert des paquets en entrée est égale à 1,6 Gbps, ce qui équivaut à 3 millions de paquets par seconde. Par conséquent, tant que ce taux n'est pas dépassé la congestion en entrée n'est pas critique. L'implémentation de la commutation est centralisée et la fabrique de commutation est basée sur une matrice de commutation qui peut aller jusqu'à 3.2 Gbps.
Le modèle du commutateur est construit suivant les composants cités plus haut. D'autres éléments vont se rajouter tels que les mécanismes de la qualité de service. Car en effet, le type de commutateurs que l'on veut modéliser propose deux modes de classification de

[18] Il faut noter qu'il n'y a pas de buffer d'entrée dans le cas ou le commutateur est basé sur une fabrique de commutation qui a une mémoire partagée ; le lecteur intéressé pourra se référer à [Seifert 2000, page 639-640].

service, auxquels s'ajoutent des mécanismes d'ordonnancement de trames. Le premier mode, fondé sur le standard IEEE 802.1 p, consiste à reconnaître les flux déjà étiquetés (802.1 p) et à les diriger vers les différentes files d'attentes suivant leur niveau de service.
Dans le second mode, l'administrateur réseau peut re-classifier les flux sur le port d'entrée avec une valeur par défaut (CdS/CoS). Dans le cas où une trame arrive sans que le champ CoS ne soit renseigné (trame dite 'non-taggée'), l'administrateur réseau peut classifier cette trame avec la valeur de la CoS par défaut définie sur ce port. Cette méthode permet d'accepter tout client qui n'est pas capable d'affecter lui-même une classe de service à ses paquets. Une fois que ces trames ont été classifiées, elles sont dirigées vers la file d'attente appropriée en sortie. Ce commutateur possède 4 files d'attentes par port en sortie. Une file d'attente possède une priorité 'absolue' (la priorité supérieure) et permet de traiter les applications temps réel, et les trois autres permettent de prendre en compte les paquets de basses priorités. Il faut noter, qu'il y a autant de buffers de sortie que de priorité. Les trames sont, ensuite, transmises suivant le choix de la politique d'ordonnancement implantée : soit l'ordonnancement Weighted Round Robin (WRR), soit l'ordonnancement à Priorité Stricte (PS) pour un Catalyst 2950XL.

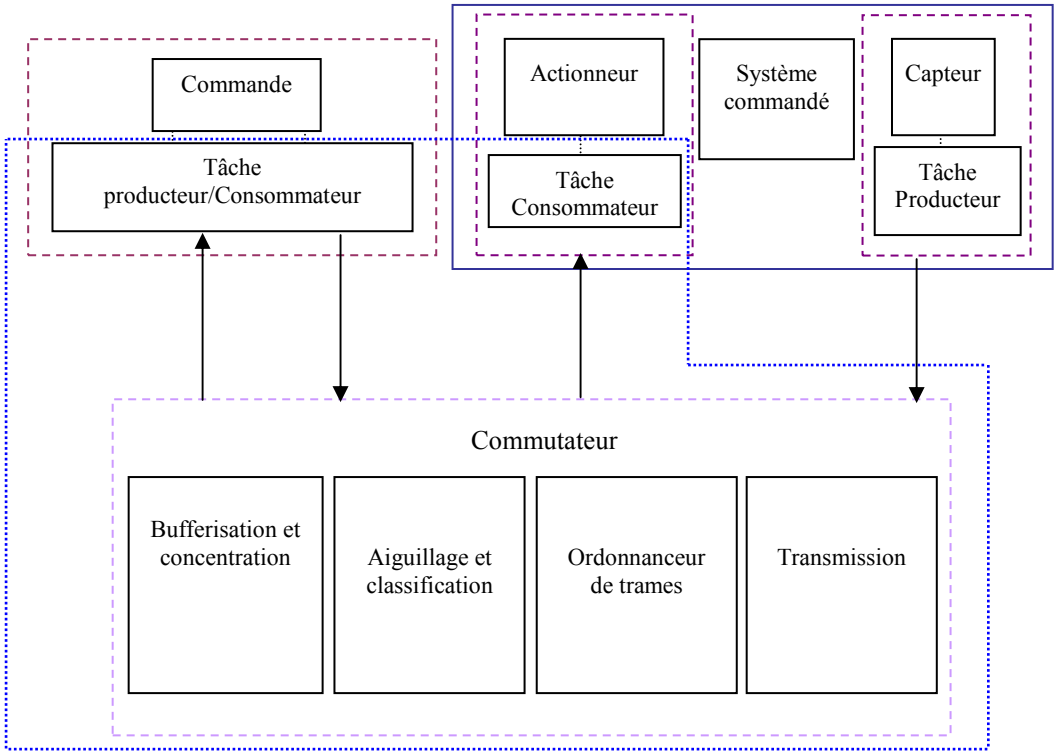


Figure 4. Le système modélisé : le commutateur à modéliser dans un environnement producteur/consommateur périodique.

Nous avons donc limité les fonctions du commutateur aux opérations suivantes et cela pour réduire la complexité et améliorer la lisibilité du modèle :

- Réception de la trame et mise dans une mémoire partagée,
- Aiguillage vers les autres ports du commutateur,
- Choix de la classe de la trame, puis mise en place dans la file d'attente correspondante,
- Ordonnancement des trames suivants deux politiques d'ordonnancement : priorité stricte (PS), et le Weighted Round Robin (WRR)
- Destruction des trames ayant attendu trop longtemps,
- Sélection de la trame à transmettre, et émission de la trame.

Nous avons choisi le formalisme de réseau de Petri coloré pour modéliser de système le commutateur fonctionnant dans un environnement producteur/consommateur périodique (figure 4). Cet environnement est idéal pour représenter un environnement de contrôle-commande [Blum and Juanole 1999, Abdellatif and Juanole 2001], qui représente le contexte de notre travail. D'autres auteurs ont utilisé ce formalisme pour modéliser le commutateur Ethernet. Nous citerons [Marsal 2006, Marsal et *al*. 2005, Zaitsev 2003, Zaitsev 2004] et bien entendu ce formalisme a été utilisé dans d'autres domaines tels que : les systèmes mécatroniques [Moncelet et *al*. 1998], dans quelques applications militaires [Kristensen et *al*. 2006], dans les systèmes workflow [Rozinat et *al*. 2006]…etc.

3. 3 Le formalisme des Réseaux de Petri colorés

Les réseaux de Petri (RDP) offre un cadre bien adapté et progressif pour la représentation et l'analyse des systèmes de communications. En effet là où échouent d'autres méthodes de modélisation (les chaînes de Markov, les réseaux de files d'attentes etc…), les RDP semblent bien appropriés pour décrire les systèmes dans lesquels interviennent les problèmes de concurrences, de synchronisation, de partage de ressources et de parallélisme [Juanole et *al*. 2004, David and Alla 1989, Jensen 1992]. Il arrive souvent que dans un système, on rencontre plusieurs parties qui ont une description identique. Elles sont alors modélisées par le même RDP autant de fois qu'il y a de parties identiques. Cela a pour conséquence d'aboutir à des modèles de taille trop importante et donc inexploitable. D'où la nécessité de réduire la taille des modèles.
De plus, dans un RDP l'information est portée par la place. Par exemple, la présence d'un jeton (ou marque) dans une place peut signifier une ressource disponible. Si elle est vide, cela signifie que la machine est occupée. Si on veut enrichir l'information apportée par une place d'un RDP, il faut être en mesure de distinguer deux jetons entre eux au sein de la même place. On associe alors un identificateur ou une couleur à chaque jeton de la place. L'information est représentée par un ensemble place-couleur. Ainsi est défini un nouvel outil de modélisation : les réseaux de Petri colorés. Bien entendu il existe plusieurs extensions et nous utiliserons ici les réseaux de Petri Colorés Temporels et Hiérarchiques (RDPCTH) autrement dit les RDPs de haut niveau.
Le formalisme présenté ici est celui défini par Kurt Jensen dans [Jensen 92, 94] et utilisé dans l'outil CPN Tools de l'université de Aarhus[19] [Jensen 1997].

3. 3. 1 Présentation intuitive des RDPC [Juanole et *al*. 2004, David and Alla 1989] :

Le comportement d'un système peut être vu en termes de conditions et d'actions (ou événements) : une action est réalisée quand un certain nombre de conditions (conditions d'entrée de l'action) existent. La réalisation d'une action induit la génération de nouvelles conditions (conditions de sortie de l'action) qui elles mêmes vont être les conditions d'entrée d'une autre action etc…
Le modèle RDP places-transitions est très bien adapté pour représenter cette vue '*comportementale*' d'un système.
Imaginons maintenant que deux systèmes identiques soient modélisés chacun par un RDP, on peut représenter le fonctionnement de ces deux systèmes par un seul RDP en attribuant une couleur à chaque jeton associé au RDP représentant chaque système. Pour réaliser une action un jeton passe d'une place à une autre en franchissant une transition. Les places et les transitions sont reliées par des arcs valués. La valuation des arcs en entrée permet d'indiquer

[19] http://wiki.daimi.au.dk/cpntools/cpntools.wiki

les conditions d'activation d'une transition alors que la valuation des arcs en sortie détermine l'effet de l'action représentée par la transition. Cette valuation représente la relation entre les couleurs de franchissement et le marquage coloré concerné.
La couleur associée à un jeton peut être complexe (i.e. composée de plusieurs couleurs) et peut donc porter une information complexe telle que la longueur d'une trame, son entête, le type de données transportée par une trame.
Notons que quelques éléments de base aidant à la compréhension des RDP sont présentés à l'annexe 2.

3. 3. 2 Présentation formelle [Marsal 2006, Jensen 1980, Jensen 1992, David and Alla 1989, David and Alla 1994] :

Les réseaux de Petri colorés sont standardisés par le IEC comme des réseaux de Petri à haut niveau (ISO Standard 2004), et ils sont définis comme suit :
Un réseau de Petri coloré est un n-uplet (P, T, D, $Type$, Pre, $Post$, M_0) :
Où,

- P est un ensemble fini de places,
- T est un ensemble fini de transitions disjoint de P, i.e. ($P \cap T = \phi$),
- D est un ensemble fini non – vide de domaines où chaque élément de D est appelé $Type$ qui est ici une couleur,
- $Type$: $P \cup T \rightarrow D$ est une fonction qui associe des types (couleurs) aux places et détermine les modes de franchissement,
- Pre, $Post$: $Trans \rightarrow \mu Place$ où Pre, $Post$ sont des fonctions d'incidence avant, et des fonctions d'incidence arrière, respectivement. Ces fonctions relatives aux couleurs de franchissement associées aux arcs, définissent les règles de franchissements, avec :

$$Trans = \{(t,m) \mid t \in T, m \in Type(t)\},$$
$$Place = \{(p,g) \mid p \in P, g \in Type(p)\}$$

- $M_0 \in \mu Place$ est un ensemble multiple appelé marquage initial du réseau avec $\mu Place$ qui est un ensemble de l'ensemble multiple appartenant à l'ensemble $Place$.

Pour mieux illustrer ces définitions, deux exemples sont présentés. Le premier est introductif utilisant les définitions présentées dans [David and Alla 1989]. Le second exemple est un complément du premier utilisant les définitions de [Jensen 1997], plus générales et complètes :
Exemple 1 :

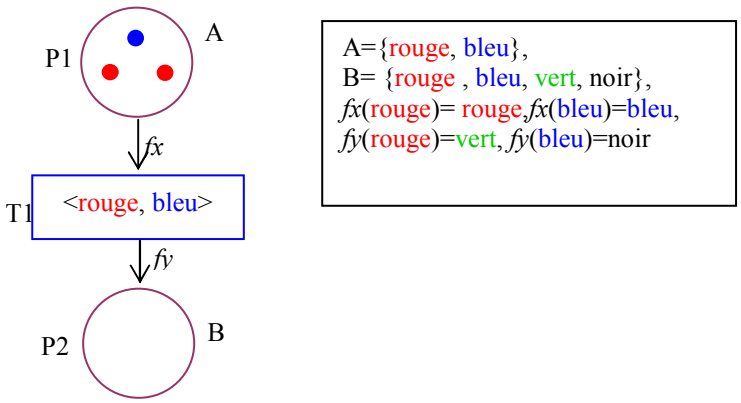


Figure 5. Exemple d'un RDP coloré

Comme dans tous les RDP, le graphe de la figure 5 comporte deux types de nœuds, les places et les transitions, reliés par des arcs orientés.
Une place est représentée par un cercle. La place P1 de la figure 5 contient trois jetons, un jeton de couleur bleu, et deux jetons de couleur rouge. Ces couleurs représentent les types. Une transition est représentée par un rectangle. A chaque transition est associée un ensemble de couleurs. Chacune de ces couleurs indique une possibilité distincte de franchissement. La transition T1 peut être franchie par rapport à la couleur <rouge> et à la couleur <bleu>.
Un arc orienté relie une place à une transition, ou une transition à une place. Le poids d'un arc est une fonction *Pre* (appelée ici *fx*) ou *Post* (appelée ici *fy*) qui établit une correspondance entre chaque couleur de la transition et les couleurs de la place. Ainsi le franchissement de T1 par rapport à la couleur <rouge> retire le couleur <rouge> qui se traduit par la fonction [*fx*(rouge)= rouge]. Elle ajoute un jeton de couleur <vert> qui se traduit par la fonction [*fy*(rouge)=vert]. Donc les fonctions *Pre* et *Post* s'écrivent comme suit :

Pre (P1, T1/<rouge>)= *fx*[20](rouge)= rouge,
Pre (P1, T1/<bleu>)= *fx*(bleu)= bleu.

Post(P2, T1/<rouge>)= *fy*(rouge)= vert,
Post(P2, T1/<bleu>)= *fy*(bleu)= noir.

Exemple 2 :

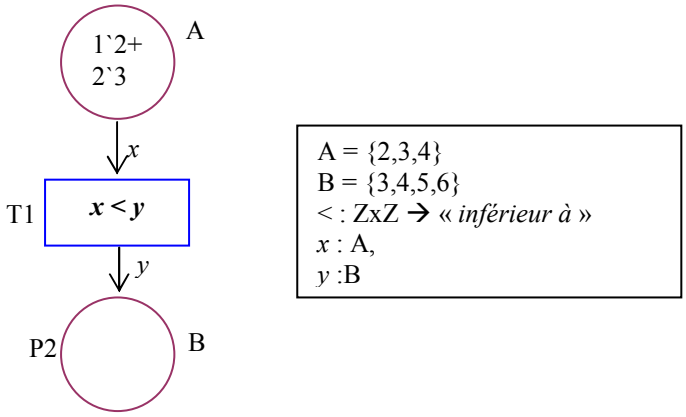


Figure 6. Exemple d'un RDP coloré

Dans cet exemple (figure 6) le RDP contient deux places P1 et P2, auxquelles sont associées deux différents types ou couleurs A et B. Ces types sont définis dans la déclaration du RDP, à savoir : A = {1,2,3,4}, B = {3,4,5,6}. Ici les couleurs de franchissement de la transition sont définies implicitement. La place P1 contient 3 jetons : un jeton dont la valeur est 2 (1`2) et deux jetons dont la valeur est 3 (2`3). Quand la transition T1 est franchie, un jeton appelé '*x* 'associée à l'arc amont de T1, est retiré de la place P1, et un jeton appelé '*y*' associée à l'arc aval de T1, est ajouté à P2. Le garde '*x* < *y*' associée à T1 définit la condition de franchissement de la transition T1, à savoir pour que T1 soit franchie il faut que le jeton x soit inférieur à y.
En faisant une analogie aux définitions de [David and Alla 1989] cela peut être assimilé à la fonction suivante (*x* : *x*(2) = 2, *x* :*x*(3) =3, *x* : *x*(4)=4.) associée à l'arc amont de T1. Le garde et 'y' peuvent être définis par la fonction suivante :

y : y(2)= 3 ou 4 ou 5 ou 6.

[20] Cette fonction est appelée fonction identité, le lecteur intéressé peut se référer au livre de David et Alla [David and Alla 1989].

Y : y(3)= 4 ou 5 ou 6.
Y : y(4)= 5 ou 6.

3. 3. 3 Les réseaux de Petri hiérarchiques

Le réseau de Petri Hiérarchique est un modèle dans lequel une partie peut être représentée par une transition de substitution qui est une abstraction d'un autre modèle. C'est à dire, que cette partie est détaillée à part. La hiérarchie sert à subdiviser un modèle en différentes parties ce qui autorise une modélisation modulaire.
Dans ce qui va suivre, nous allons définir la sémantique utilisée par K. Jensen. Donc un réseau de Petri coloré hiérarchique (RDPCH) est un n-tuplet :

$$RDPCH = (S, SN, SA, PN, PT, PA, FS)$$

Où,

- *S* est un ensemble fini non-vide de pages où chaque page représente un réseau de Petri coloré non-hiérarchique,
- *SN* est un ensemble de nœuds de substitution appartenant à l'ensemble *T* (ensemble de transitions),
- *SA* est une fonction d'assignation des pages définies à partir de *SN* dans *S*. Cela veut dire que le nom (ou l'assignation) du nœud de substitution correspond au nom (ou l'assignation) de la sous-page qui le représente par un RDPC non-hérarchique.
- *PN* est un ensemble de nœuds appelés *Port* appartenant à l'ensemble *P*.
- *PT* est le type de fonctions associé au noeud 'Port ' des éléments *PN*. Ce type de fonctions peut être : *in*, *out*, *i/o* (*in/out*) (figure 7).
- *PA* est une fonction d'assignation du port qui définit la relation entre les nœuds, et entre les pages et les sous-pages. Deux types d'interconnexion sont possibles : les transitions de substitution et les places fusionnées. Une transition de substitution remplace une page complète. Un ensemble de places fusionnées est fonctionnellement identique : elles ont le même marquage à tout instant et peuvent être utilisées à différents endroits du modèle comme s'il s'agissait d'une même et unique place.
- *FS* est un ensemble qui regroupe tous les ensembles appelés fusion appartenant à P où tous les éléments ont la même couleur, et la même fonction d'initialisation (nombre de jeton).

Pour mieux illustrer ces définitions un exemple introduisant ces notions est représenté ci-dessus :

Exemple 3 :

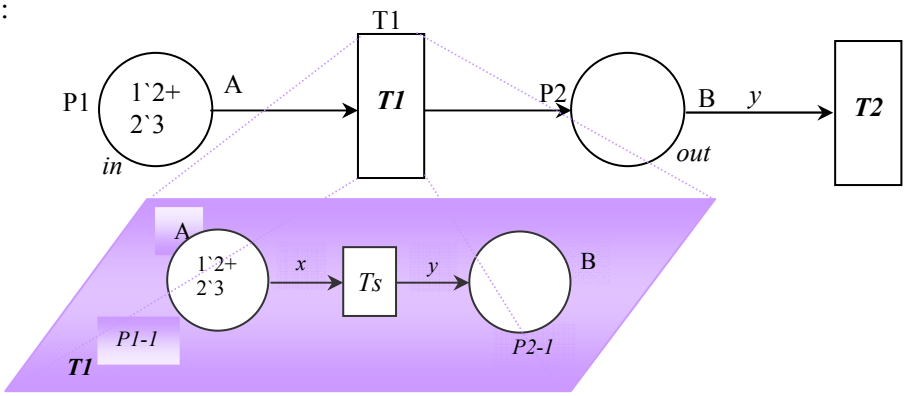


Figure 7. Exemple d'un RDP coloré hiérarchique.

Dans cet exemple (figure 7) nous avons les mêmes déclarations que dans l'exemple 2, seulement ici nous avons ajouté une transition T2 qui est une transition puits. Elle retire un jeton *y* de P2. La transition T1 est une transition de substitution à laquelle est associée une sous-page du même nom. Cette sous-page représente un RDP coloré (*i.e.* la transition T1 est une abstraction du réseau de Petri coloré représenté par la sous-page T1) possédant deux places : P1-1, P2-1. P1-1 et P2-1 doivent être du même type que P1 et P2 respectivement et doivent avoir le même nombre de jetons. Les places P1 et P1-1, (respectivement P2 et P2-1) sont appelées des places fusionnées.
Les places P1 et P2 représentent les ports d'entrées et de sorties représentées par les étiquettes *in* et *out* respectivement de la transition de substitution T1 donc du RDPC représenté par la sous-page T1.
S est la sous-page de la transition de substitution T1. SN est la transition de substitution elle-même. PN représentent les places ports P1 et P2, PT représente les fonctions *in* et *out* associées à P1 et P2. *PA* représente la fonction d'interconnexion entre les places fusionnées et la transition de substitution ainsi que la page qu'elle représente. *FS* représente les places concernées par la fusion (P1, P1-1) et (P2, P2-1).

3. 3. 4 Le temps et les réseaux de Petri colorés

Nous allons ici introduire la notion du temps selon la sémantique de K.Jensen qui est aussi utilisé par l'outil de simulation CPN Tools. Donc un RDPCT est défini comme un n-tuplet :
(*CPN*, *R*, r_0)

- *CPN* est le réseau de Petri coloré,
- *R* est l'ensemble des valeurs inclus dans $\Re^{+0}$,
- r_0 est la valeur initiale dans *R*.

Exemple 4 :

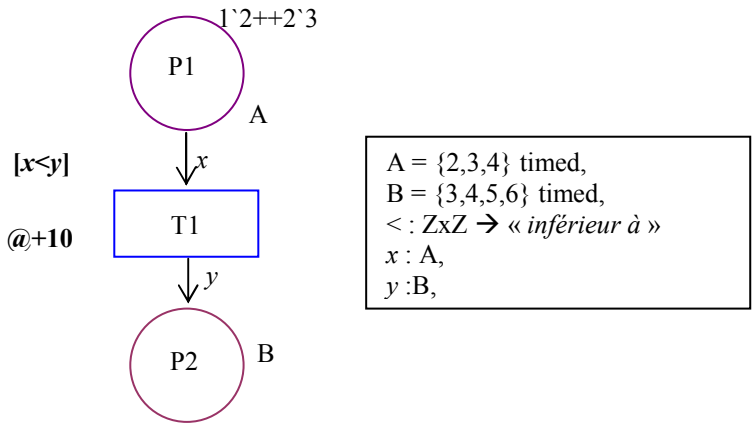


Figure 8. Exemple d'un RDP coloré temporisé.

Reprenons l'exemple 2 dans lequel on ajoute la composante temporelle (figure 8), elle s'exprime par l'ajout de l'attribut 'timed'. Quand la transition t1 est tirée un temps de 10 (@+10) va être ajouté au jeton produit par P1. Le marquage de la place P2 avec le jeton temporisé est noté par un opérateur spécifique '@+10 ' (par exemple 1`2@+10). Ce délai détermine la durée pendant laquelle le jeton va rester indisponible. C'est-à-dire que ce jeton ne peut être disponible dans P2 que après 10 unités de temps, (une fois que la transition eut été franchie).
Ce fonctionnement ressemble beaucoup au RDP P-temporisé de Sifakis [Sifakis 1977] Cependant ici sa représentation diffère de ce qu'on a vu dans la littérature. Le temps n'est

pas seulement associé aux transitions, mais il est aussi associé aux jetons (i.e. chaque jeton peut porter une étiquette représentant le temps –time stamp-).

3. 3. 5 CPN Tools : Spécificités liées à la modélisation

CPNTools est un outil de modélisation et de simulation pour le formalisme des réseaux de Petri colorés. Il permet d'éditer des réseaux, de vérifier la syntaxe, de simuler (en pas à pas ou automatiquement avec pour critère d'arrêt une date ou un nombre de tirs de transitions), de construire le graphe d'occurrence et de l'exploiter à l'aide d'un ensemble de primitives. Nous présentons ici quelques caractéristiques de cet outil liées à la modélisation.

Utilisation des couleurs complexes et des fonctions

Couleurs complexes :

La déclaration des couleurs et des variables est effectuée à l'aide du langage CPN ML basé sur le langage de programmation ML [Milner et *al*. 1997, Ullman 1998].
Grâce au logiciel CPNTools, il est possible d'utiliser des couleurs complexes et des fonctions. Les couleurs simples existantes dans le CPN ML, sont : unité (couleur noir –sans couleur-), entier, booléen, string (caractère), énuméré, indice. Basés sur ces types, des couleurs complexes peuvent être construites en faisant par exemple le produit de ces différentes couleurs, ou alors de construire une liste d'un ensemble ordonné de variables du même type (même couleur). Dans cette section nous allons présenter un exemple d'utilisation de la liste qui sera très employé dans nos modèles.
Dans une liste des éléments peuvent être enlevés ou ajoutés durant la simulation. C'est-à-dire que des couleurs du même type peuvent être empilées et gardées dans leur ordre d'arrivée dans un seul jeton de type liste. Ce comportement représente celui d'une file[21] FIFO (First In First Out).
Pour contrôler les listes deux opérateurs spécifiques sont utilisés : ' ^^' et ' ::'. Le premier sert à concaténer deux listes. Le second sert à enlever le premier élément de la liste (i.e. l'élément qui se trouve au début de la liste). La figure 9 représente le RDPC de la file FIFO. La place P2 représente le buffer contenant la liste des valeurs de couleur A qui est alimentée par la transition T1 avec des éléments e1 venant de P1. Ensuite cette liste (couleur B) est vidée en retirant l'élément e2 de l'entête de cette même liste.

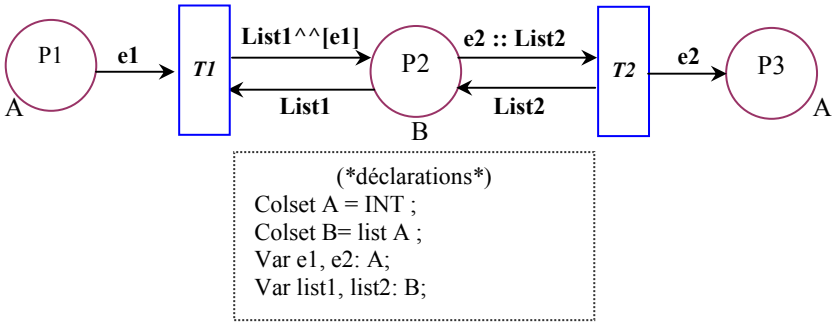


Figure 9. Modèle RDPC d'une file FIFO

Fonctions :

[21] Bien entendu d'autre type de file d'attente peuvent être modélisé tel que la LIFO (Last In First Out)

Le logiciel CPN Tools fournit un ensemble de structures et de syntaxes permettant de construire des fonctions. Un tel avantage donne la possibilité de construire un RDP paramétrable dont on peut obtenir une instanciation en changeant seulement les déclarations.

Exemple 5:

Fun exemple1(*x*) = *if* x<3
then "x is less than three"
else "x is greater than or equal three"

Cette expression vérifie si la valeur de x est inférieure à 3 et retourne le caractère approprié à la réponse.

Exemple 6:

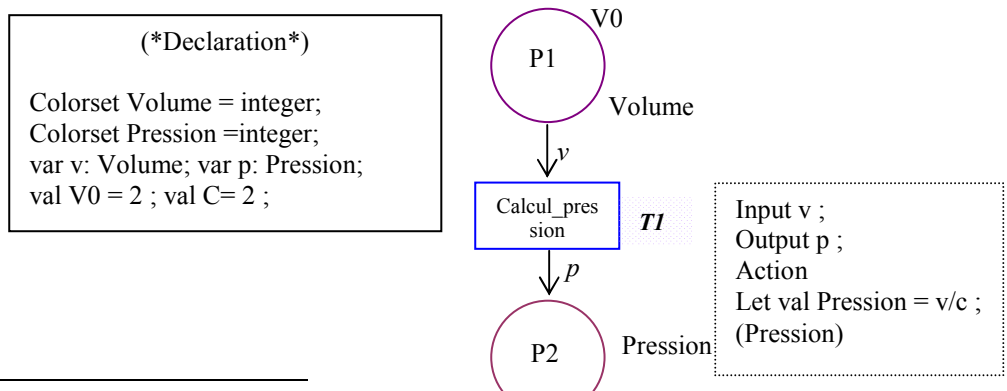
Fun exemple2(day) = *case* day of
Mon => "Today is Monday"
| Tues => "Today is Tuesday"
| Wed => "Today is Wednesday"
| Thurs => "Today is Thursday"
| Fri => "Today is Friday"
| _ => "It is a weekend day"

Cette expression retourne le caractère qui dépend de la valeur de 'day', où day est de type Day qui est un exemple de la couleur ''énuméré''[22]. Le sous-tiret '_' indique que l'expression ne retourne le caractère « It is a weekend day » que si la variable 'day' est égale à un caractère autre que Mon, Tues, Wed, Thurs, or Fri.

Le segment de code :

De plus CPNTools permet de réaliser une tâche au moment du franchissement d'une transition. Par exemple un calcul peut être effectué pour instancier une variable dans l'expression de l'arc de sortie, ou le résultat d'un calcul sur les attributs des jetons amont peut être écrit dans un fichier. Ces tâches sont réalisées par une procédure écrite en langage ML. Le code de cette procédure est appelé *segment de code* et est toujours associé à une transition. La figure 10 ci-dessous illustre un exemple d'utilisation du segment de code [Moncelet 1998].

Exemple 7 :



[22] La déclaration du type énuméré se fait comme suit : colset Day= with Mon | Tue| Wed | Thurs | Fri | Sat | Sun ;
Var day : Day ;

Figure 10. Exemple de segment de code associé à une transition

Dans cet exemple lorsque la transition *Calcul_pression* est franchie la pression *p* est calculée à partir d'une information 'volume' portée par le jeton amont.

3. 4 Modélisation du commutateur Ethernet

Dans cette section, le but est de modéliser un commutateur Ethernet dont les fonctions élémentaires sont représentées sur la figure 11. L'intérêt du RDPCTH est qu'il permet une représentation modulaire, et hiérarchique. Dans les sections qui vont suivre nous procédons d'une manière descendante. C'est-à-dire que nous partons du modèle global du système, puis de fil en aiguille, chaque module le constituant sera expliqué et son modèle représenté.

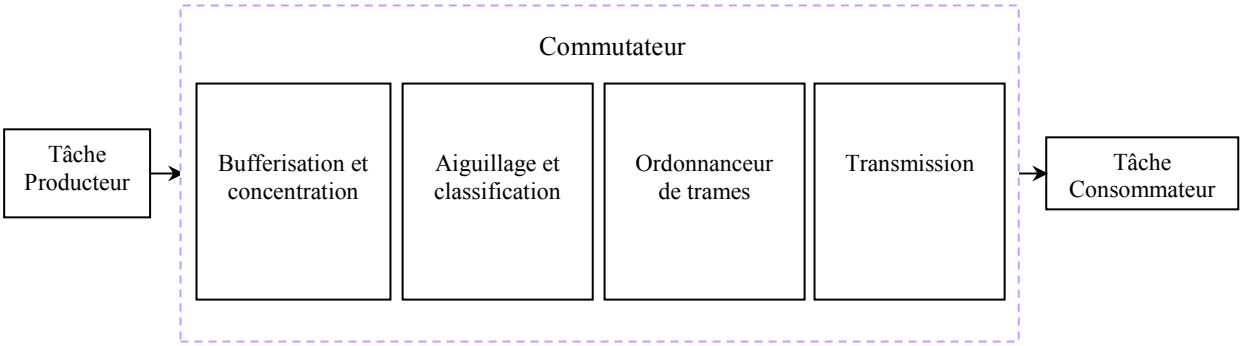


Figure 11. Opérations traversée par le flux venant du producteur

3. 4. 1 Représentation hiérarchique

Le modèle du système global représente l'abstraction du système de la figure 11. On voit que le système englobe le commutateur et le modèle coopératif sous lequel il fonctionne.
Le commutateur est divisé en cinq sous modules qui représentent son fonctionnement et son architecture interne.
Le module multiplexage et bufferisation exprime l'opération de réception et de mise en mémoire (mémoire partagée) de la trame d'entrée. L'opération de multiplexage résout le phénomène de compétition qu'il peut y avoir à l'entrée du commutateur.
Ensuite les trames subissent l'opération d'aiguillage, qui correspond à une lecture de l'adresse de destination afin qu'elles soient transmises vers la file d'attente de sortie appropriée. Juste après ceci une classification des trames suivant leur priorité est réalisée. Ces trames sont mises dans des files d'attentes FIFO afin de veiller à ce qu'elles soient transmises selon l'ordre de leur arrivée. Il faut savoir qu'il y a autant de files d'attente que de priorités. Ensuite, un ordonnancement suivant soit l'algorithme de priorité stricte ou l'algorithme du Weighted Round Robin est construit. Enfin, un mécanisme de transmission des trames est mise en place : il transmet les trames si le port est libre. Cette configuration respecte le fonctionnement du commutateur 2950 XL exposé au paragraphe (§ 3.2, page. 80).
Au final, les trames une fois transmises sont consommées par des consommateurs périodiques.

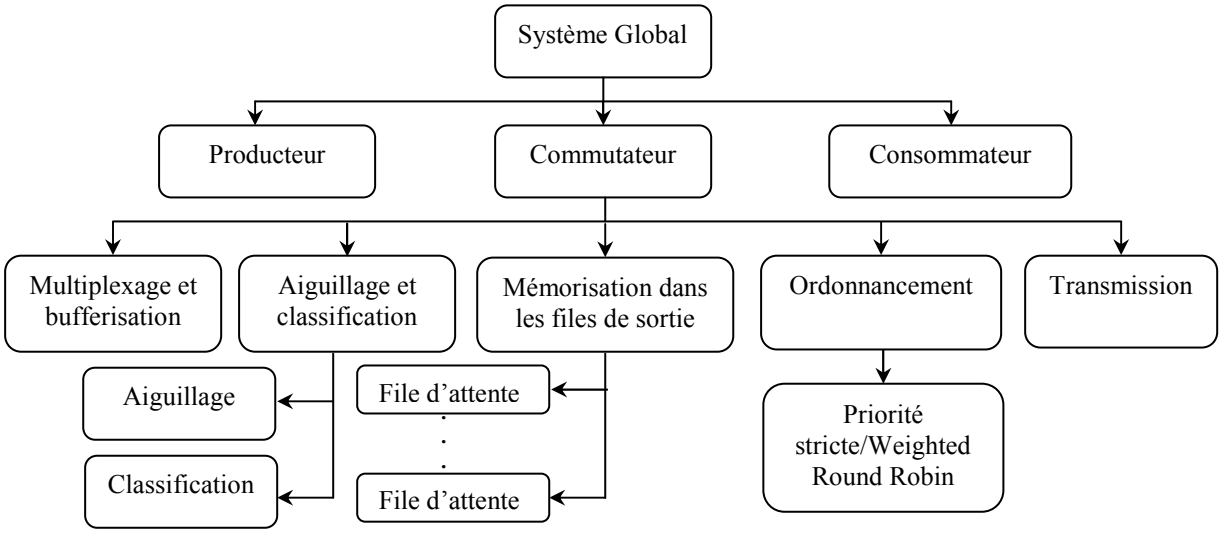


Figure 12. Modèle hiérarchique du RDPCTH représentant de système étudié.

3. 4. 2 Définition des couleurs :

Les couleurs définies représentent la trame Ethernet avec quelques abstractions, suivant les besoins et les hypothèses posées.
Les éléments que nous avons modélisés sont choisis selon nos besoins et leurs pertinences. Ces éléments sont : l'adresse de destination, l'adresse source, le tag représentant le niveau de priorité de la trame, la donnée à transmettre.
Le mode de transmission sélectionné est le *store & forward,* ce qui laisse entendre, que les trames ne sont pas erronées. La taille de la trame est supposée constante et égale à 64 octets (sans le préambule et SFD).
Une trame est exprimée suivant les couleurs ci-dessous :

```
Colset inp = with I1 | I2;
Colset outp = with O1 | O2;
Colset prio = int with B…H;
Colset data = INT;
Val H = 2;
Val M = 1;
Val B = 0;
Colset paquet = product inp * outp * prio timed;
Var  i: paquet;
```

L'adresse destination est définie par la couleur 'outp', l'adresse source par la couleur 'inp', et le tag exprimant le niveau de priorité de la trame par la couleur 'prio'. Ici trois niveaux de priorité sont définis : H = 2 (haute priorité associée aux trames de commande, *i.e.* à temps critique), M= 1 (priorité moyenne appelée aussi 'excellent effort'), et B = 0 (faible priorité appelée aussi 'best effort')

Le paquet est composé d'une couleur complexe réalisée grâce à la fonction *Product* (Produit de plusieurs couleurs). La couleur *data* n'a pas été ajoutée au produit, puisqu'elle n'est pas encore sollicitée dans cette section.

3. 4. 3 Module producteur et consommateur périodique :

3. 4. 3. 1 Le producteur périodique

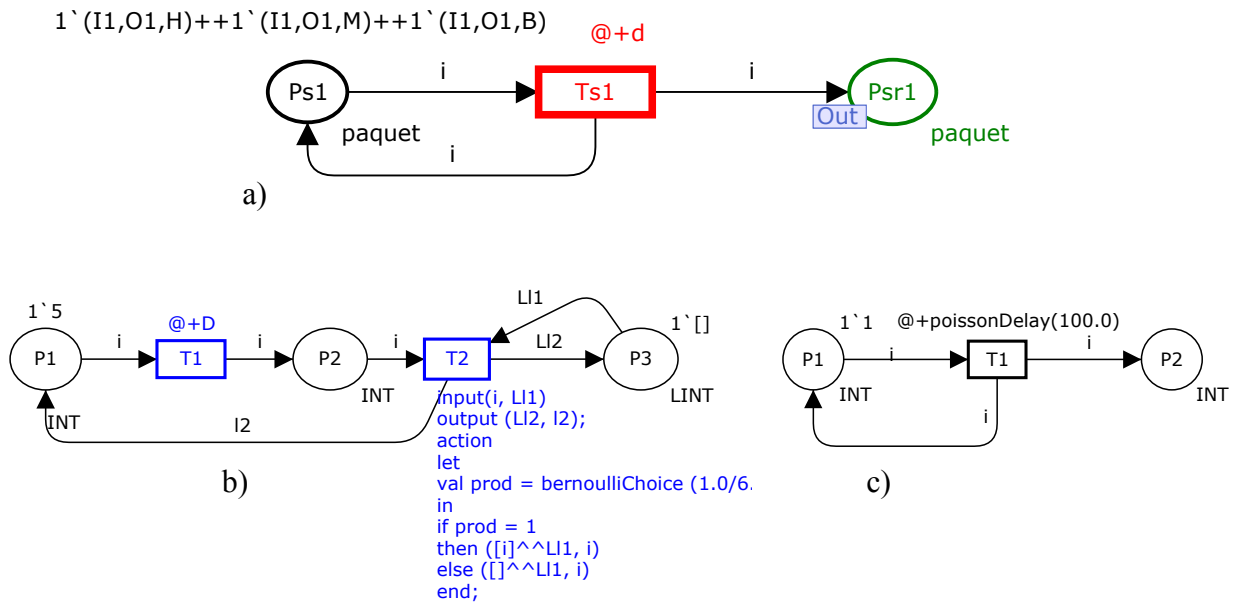


Figure 13. Modèle RDPCT de : a) la source périodique, b) la source de Bernoulli, c) la source de Poisson

Le modèle RDPCT d'une source périodique est représenté par la figure 13. a). La durée qui sépare deux générations successives de paquets est une constante égale à la période 'd', et est associée à la transition 'Ts1' grâce à la primitive '@+d'. La place 'Psr1' représente le paquet du flux généré.

D'autres sources de trafic sont utilisées pour décrire un trafic de type aléatoire, nous citons par exemple la source de Poisson, ou de Bernoulli.

Source de Poisson **:**
Le modèle RDPCT de la source de Poisson (figure 13. c) représente la durée séparant deux générations successives de paquets par une distribution de probabilité exponentielle. Cette durée est modélisée par la fonction 'poissonDelay 'associée à T1. Cette fonction exploite le fait que CPNTools offre la possibilité d'utiliser plusieurs fonctions de probabilité[23].

Source de Bernoulli :
La source de Bernoulli (figure 13. b) utilise deux paramètres : une durée déterministe 'D' et une probabilité 'P'. La durée 'D', est associée à la transition 'T1'. Chaque fois que la transition T2 est tirée, un choix probabiliste est effectué grâce au segment de code associé à la transition T2. Ce choix dépend de la probabilité de Bernoulli représentée par la fonction 'bernoulliChoice' . Si celle-ci est un succès alors le paquet est généré à la fin de la durée D avec une probabilité P dans le cas contraire le paquet n'est pas généré.

[23] http://wiki.daimi.au.dk/cpntools-help/

3. 4. 3. 2 Le consommateur périodique

Le modèle RDPCT d'un consommateur périodique est présenté sur la figure .14. a. Les paquets sont consommés périodiquement toutes les 'd' périodes. La place 'PC1o' reçoit les paquets qu'il va consommer à condition que la mémoire de la taille de la trame soit vide, c'est-à-dire que la place PC1v est marquée. La place 'Ppc1' représente l'horloge du consommateur. Bien évidemment d'autres représentations prennent en compte le déphasage [Abdellatif 2002, page : 104-105] qu'il y a entre le producteur et le consommateur au lieu de représenter explicitement l'horloge. Un autre modèle de consommateur appelé consommateur guidé par les événements (figure 14.b) peut être utilisé. Il peut être aussi représenté par une transition puits puisque dès l'arrivée d'un paquet il est consommé.

A la place 'Ptr2' est associée un tag 'In' exprimant le port d'entrée (input) du module consommateur. Sur la place 'Pbp1' est associée un tag 'Out' qui représente la place port de sortie (output) du même module, et représente en même temps une place port d'entrée du module commutateur (figure 15). Cette place autorise (ou pas) la transmission, et par conséquent, la consommation (ou pas) des trames transmises par le commutateur, et cela en fonction de la disponibilité du consommateur. Ce fonctionnement traduit implicitement 'le module de transmission' en charge de transmettre les paquets qui ont été choisis par le mécanisme d'ordonnancement de paquets.

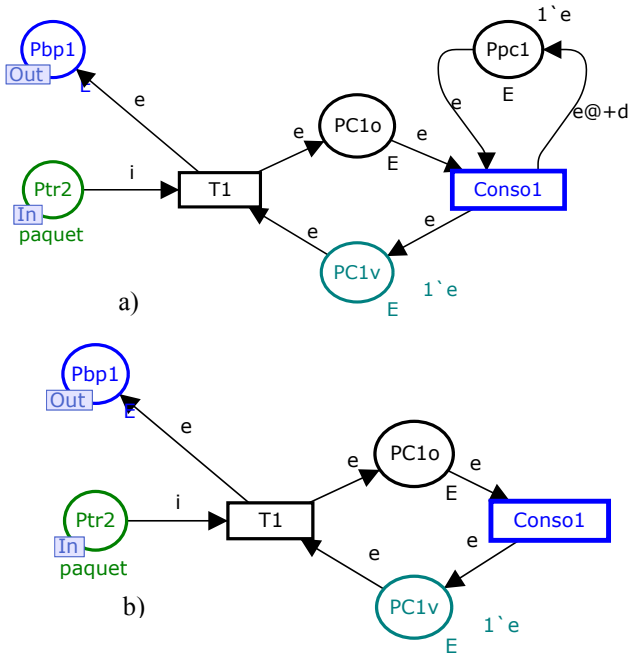


Figure 14. Modèle RDPCT d'un consommateur : a) guidé par le temps (périodique), b) Guidé par les événements.

3. 4. 4 Le module commutateur

Dans le modèle de la figure 15 toutes les opérations exposées dans la section 3. 2 (page 80) par lesquelles passe le flux traversant le commutateur, sont modélisées. A savoir la mémorisation dans une mémoire d'entrée de type FIFO dès la réception de la trame. Ensuite, le flux est confronté à l'opération d'aiguillage qui permet d'envoyer les paquets vers la file de sortie appropriée. Après cela vient l'opération de classification qui permet de sélectionner les paquets suivant leur priorité dans la file de sortie adéquate. Enfin, ces paquets sont

transmis vers la sortie suivant l'algorithme d'ordonnancement de paquets implanté dans le commutateur.

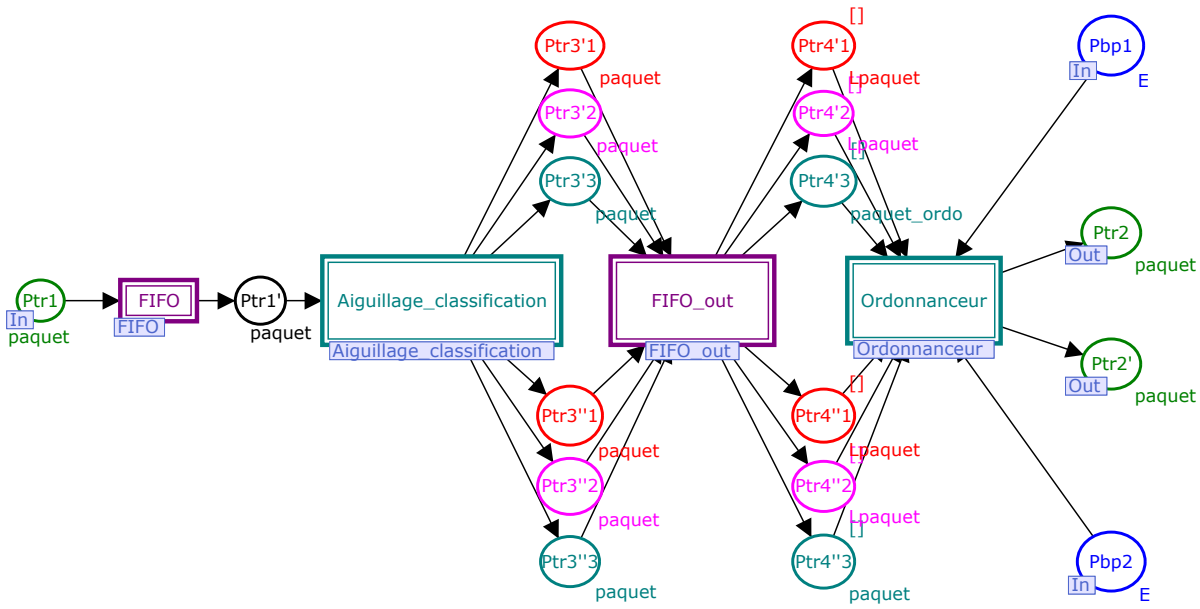


Figure 15. Modèle RDPCTH du commutateur.

3. 4. 4. 1 Le module concentration et bufferisation :

Ce module est représenté par la transition de substitution FIFO, les trames générées par le producteur périodique sont dirigées vers la place port d'entrées Ptr1 (qui porte le tag In) et sont immédiatement mémorisées dans la mémoire partagée FIFO. Ce module FIFO décrit une file d'attente qui fonctionne selon la politique du premier arrivé, premier servi. Son fonctionnement et son modèle sont expliqués et représentés dans la section 3. 3. 5. 1 page 87.

3. 4. 4. 2 Le module Aiguillage et classification :

Les trames reçues par la place 'Ptr1' ' qui correspond au port d'entrée du module 'aiguillage et classification' sont d'abord redirigées suivant l'adresse de destination contenue dans les trames. Cet aiguillage est réalisé grâce aux conditions associées aux arcs en aval de la transition 'Aiguillage' (figure 16). Cette condition repose sur la primitive '*case of* ' qui exprime que toutes les trames ayant une adresse de destination égale à 'O1' (ou 'O2' respectivement) –c'est-à-dire le port de sortie '1' (ou '2' respectivement) – sont dirigées vers les buffers du port de sortie '1' (ou '2' respectivement). La liste de la primitive '*case of* ' peut s'allonger proportionnellement au nombre de port de sortie et d'entrée. Dans cet exemple nous avons limité la liste par souci de lisibilité, en modélisant un commutateur avec deux ports d'entrée et deux ports de sortie.
Outre l'opération d'aiguillage, l'opération 'classification' est représentée par les transitions 'Classification_1' et 'Classification_2' (classification pour les trames à destination du port de sortie '1' respectivement '2'). Des conditions de classification du même type que pour l'aiguillage sont associées aux arcs en aval des transitions 'Classification_1' et

'Classification_2'. Les trames reçues par les places PM1 et PM2 sont alors orientées suivant leur niveau de priorité vers l'entrée de la file d'attente de sortie appropriée. Les places auxquelles, des tags 'Out' sont associés sont les ports de sortie du module 'Aiguillages et classification et les ports d'entrées du module FIFO_OUT. Ce dernier représente le module de la mise en mémoire des trames classifiées.

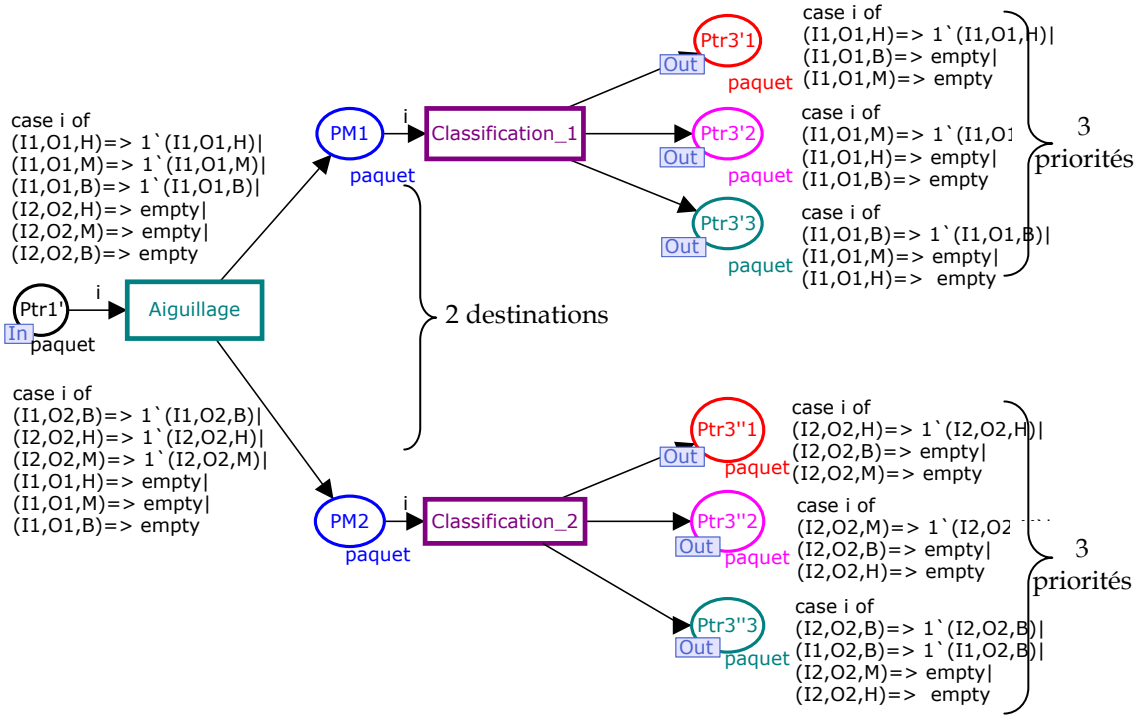


Figure 16. RDPCT du module aiguillage et classification.

3. 4. 4. 3 Module de mémorisation dans des buffers de sortie

Il y a autant de priorité que de buffers de sorties. Par conséquent, ici puisque trois niveaux de priorité sont définis trois files d'attentes FIFO[24] par sortie sont modélisées (puisque nous avons deux ports de sortie donc six FIFO sont modélisées : 3 niveaux de priorité x 2 ports de sortie).

3. 4. 4. 3 Module ordonnanceur

L'opération d'ordonnancement intervient à la sortie des files d'attentes de sortie FIFO. Les trames sont transmises suivant un algorithme d'ordonnancement. Ici deux politiques d'ordonnancement sont modélisées : l'ordonnancement à priorité stricte (ou statique), et l'ordonnancement Weighted Round Robin, car elles ont implantées dans les équipements réseau étudiés.

3. 4. 4. 3. 1 Modèle de l'ordonnanceur à priorité statique[25] :

Pour modéliser cette politique d'ordonnancement, des arcs dits inhibiteurs sont utilisés. Les arcs inhibiteurs modélisent la priorité entre les flux. Les arcs inhibiteurs, habituellement, sont représentés par un arc orienté dont l'extrémité est marquée par un petit cercle [David

[24] Le fonctionnement ainsi que la modélisation de la file d'attente FIFO est disponible au paragraphe x de la section x, page x.

[25] Le lecteur intéressé par d'autres types d'algorithmes d'ordonnancement, un panorama intéressant est pérsenté dans [Abdellatif 2002].

and Alla 1994]. Sur la figure 17. a, l'arc inhibiteur entre P1 et T2 signifie que la transition n'est validée que si la place P1 ne contient aucun jeton[26] . Sa représentation dans les réseaux de Petri colorés, spécifiquement dans l'outil CPNTools, est illustrée par un double arc orienté (fléché à ses deux extrémités) en gras auquel est associé une liste vide '[]'[27][28] (figure 17. b).

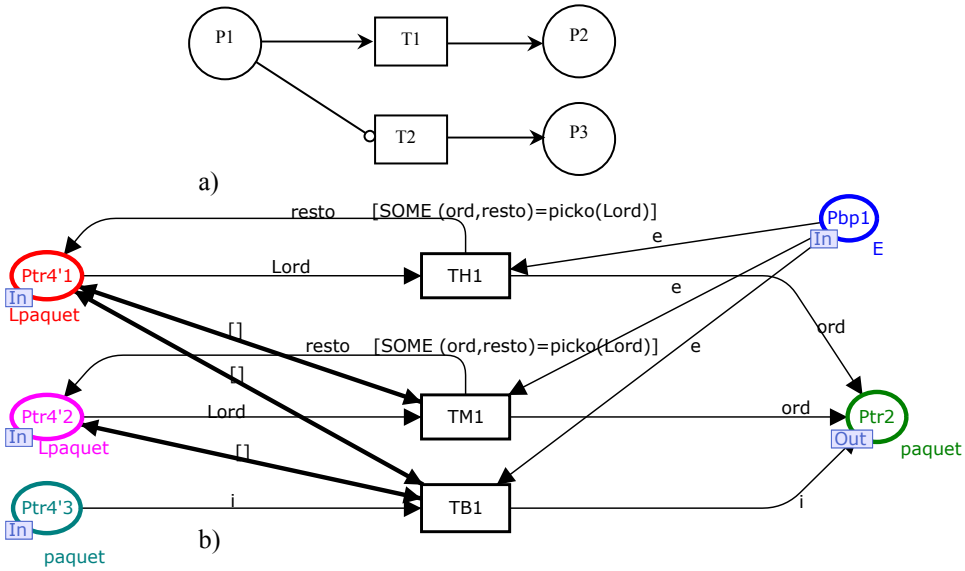


Figure 17. Modèle RDPCT de l'ordonnanceur à priorité statique

Les places ports d'entrée (Ptr4'1, Ptr4'2, Ptr4'3) du module ordonnanceur recueillent les trames suivant leur ordre d'arrivée et leur niveau de priorité, c'est-à-dire que les places Ptr4'1 Ptr4'2, et Ptr4'3 reçoivent les trames de priorité H, M, et B, respectivement.
Les transitions TH1, TM1, et TB1 ne sont franchies que par les trames de priorité H, M, et B respectivement. La transition TM1 n'est active (franchissable) que lorsque les paquets de priorités supérieures sont transmis, et la transition TB1 n'est franchie que lorsque qu'il n'y a plus de paquets de priorités supérieures à transmettre (dans ce cas les paquets de priorité H et M).
De plus, la transmission ne peut s'effectuer que lorsque la ligne est libre (port de sortie du commutateur). Celle-ci est modélisée par la place Pbp1.
Un garde est associé aux transitions TH1 et TM1 dont la fonction 'Picko' retourne tous les éléments jusqu'à ce qu'aucun élément ne reste dans la place d'entrée de la transition

```
fun picko(ord::lord)=
 let
 val c = discrete (0, (List.length (ord::lord))-1)
 in
 SOME (List.nth ((ord::lord), c), List.take ((ord::lord),c)^^List.drop ((ord::lord), c+1))
 end
 |picko [] = NONE;
```

[26] On retrouve souvent l'expression test à zéro dans l'utilisation des arcs inhibiteurs.
[27] On peut voir le petit cercle (zéro) à l'extrémité de l'arc orienté du RDP autonome comme une liste vide de l'arc représenté dans l'outil CPN Tools.
[28] http://wiki.daimi.au.dk/cpntools-help/inhibitor_arcs.wiki

L'algorithme d'ordonnancement à priorité statique possède néanmoins un inconvénient majeur dû à son caractère non préemptif. En effet, dans le cas où un paquet de haute priorité arrive durant la transmission d'un paquet de basse priorité, (c'est-à-dire que la ligne n'est pas libre à ce moment là) le paquet de priorité supérieure doit attendre la fin de transmission du paquet de priorité inférieure pour qu'il soit à son tour transmis. L'autre inconvénient concerne son caractère non équitable. En effet, ce type d'ordonnanceur avantage les paquets de haute priorité. De ce fait, les consommateurs qui attendent les paquets de priorité inférieure souffrent du phénomène de famine.
Ce type de politique d'ordonnancement est généralement utilisé pour faire face à des besoins temps réel durs (en considérant, bien entendu qu'une haute priorité est affectée à l'application temps réel).

3. 4. 4. 3. 2 Modèle d'ordonnancement Weighted Round Robin :

Nous avons évoqué dans la section précédente que le caractère non équitable de l'ordonnanceur à priorité statique peut poser un problème de famine. Pour pallier ce problème des algorithmes 'équitables' ont été proposés (exp : WFQ). Parmi eux se distinguent les algorithmes Round Robin (RR). En effet, en plus de leur faible complexité et de leur simplicité d'implantation, les algorithmes RR assurent un partage équitable de la bande passante entre les flux. Les algorithmes RR les plus connus en pratique sont le Deficit Round Robin (DRR) [Shreedhar and Varghes 1996] et le Weighted Round Robin (WRR) [Wang and Levy 2000].

Le Deficit Round Robin

Une quantité *qi* exprimée en bits est affectée à chaque file *i*, cette quantité permet de déterminer la portion de la bande passante allouée à chaque flux contenu dans chaque file. Lorsque le serveur visite une file *i*, une variable *Si* (en bits) est incrémentée de *qi*. Le serveur permet la transmission des paquets contenus dans cette file. A chaque transmission de paquet *Si* est décrémentée de la taille du paquet. Lorsque le serveur visite une file vide, la variable d'état *Si* associée à cette file est réinitialisée à zéro.
Il est à noter que le serveur visite les files à tour de rôle (cycliquement).

Le Weighted Round Robin

Dans WRR un poids *wi* est affecté à chaque file *i*. Ce poids *wi* fixe la quantité de service souhaitée par *i*. A chaque visite du serveur, les paquets stockés dans la file *i* sont transmis jusqu'à ce qu'elle soit vide ou que le nombre de paquet transmis (ou servi) a atteint la quantité *wi*. A cet instant le serveur bascule vers la file suivante dans le cycle, et la même procédure recommence. Dans notre travail le WRR est modélisé car son comportement se confond avec celui du DRR lorsque les paquets sont de la même taille, hypothèse que nous avons posée à la section 3. 4. 2.
Les places d'entrée Ptr4'1, Ptr4'2 et Ptr4'3 (figure 18) reçoivent les paquets aiguillés et classifiés. Chaque place contient respectivement les trames de haute, moyenne, et basse priorité. Le *WRR* sert les buffers suivant le cycle suivant : Ptr4'1 → Ptr4'2→ Ptr4'3.

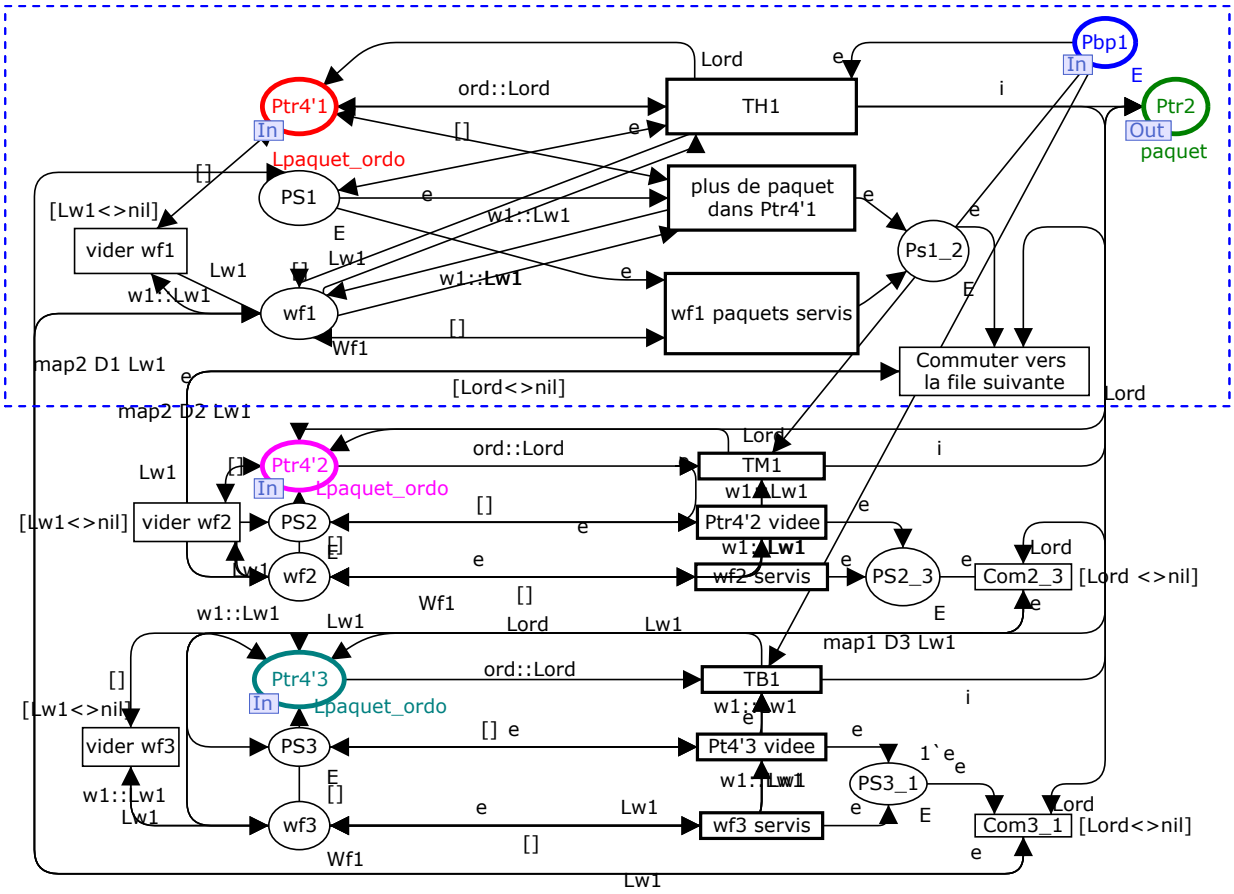


Figure 18. Modèle RDPCT de l'ordonnanceur WRR.

Lorsque le serveur *WRR* visite le buffer représenté par la place Ptr4'1, la place PS1 devient marquée limitant ainsi le nombre de transmission à la quantité *wf1*. *wf1* représente le pourcentage de bande passante allouée à une file de sortie. La place 'wf1' est donc mise à *wf1* (à l'aide de la fonction map2) et est décrémentée chaque fois que des paquets sont transmis en franchissant la transition 'TH1'. Cette même place est mise à zéro lorsque le serveur quitte la file grâce à la transition 'vider wf1'.

Le serveur passe à une autre file dans le cas où les *wf1* transmissions sont atteintes. Puis la transition 'wf1 paquets servis' est tirée, marquant la place 'PS2_1' qui représente l'état de départ du serveur de la file. Un autre cas intervient quand il n'y a plus de paquets à servir. Dans ce cas, la place wf1 est vidée puis la transition 'Ptr4'2 vidée' est franchie permettant au serveur de passer de 'PS1' (i.e. serveur sur la file Ptr4'1) à 'PS1_2' (i.e. départ de la file 1 vers la file 2). La dernière opération marque la fin de service de la file d'attente 'Ptr4'1', et le début de service de la file suivante définie dans le cycle. Le serveur commute vers la file suivante en franchissant la transition 'commuter vers file suivante' (figure 18 : voir encadré en pointillé). Si le buffer est vide (*i.e.* le marquage de Ptr4'2 est nul), le serveur est retenu dans la place PS1_2 jusqu'à ce qu'à ce qu'un nouveau paquet soit mémorisé dans 'Ptr4'2'. Cette opération est nécessaire pour éviter un graphe de marquage non bornée. En effet, puisque les paquets générés par le module source sont temporisés, et les jetons représentant la quantité de paquet à transmettre (wfi), et la situation du serveur (Psi), ne sont pas temporisés, les transitions 'plus de paquets dans 'Ptr4'x[29]' sont franchies en priorité. Le risque est que ces

[29] x représente la file 1, 2, ou 3 (*i.e.* Ptr4'1, Ptr4'2, et Ptr4'3)

transitions soient franchies successivement et indéfiniment (non borné), si le serveur n'est pas retenu jusqu'à ce qu'un nouveau paquet soit mémorisé.

3. 4. 4. 4 Module de transmission :

Le module transmission est modélisé d'une manière implicite, et est représenté par la place PB1 (figure 14, 15, 17, et 18). Grâce à ce mécanisme les paquets sélectionnés par le module ordonnanceur sont transmis dans le cas où la ligne de transmission est libre. Un délai de transmission égal à la durée de transmission d'un paquet (51 ,2 µs) est ajouté à la transition T1 (figure 14 : module consommateur).

4. Evaluation de performances :

Dans cette section, il s'agit d'évaluer les performances d'une relation producteur/consommateur distribuée au travers d'un commutateur Ethernet. L'objectif visé est de montrer une approche permettant d'effectuer des mesures de performances sur des applications distribuées qui utilisent la relation producteur/ consommateur comme modèle de coopération [Vega and Thomesse 1995, Thomesse et *al*. 1995]. Les applications de contrôle/commande reposent souvent sur ce type de coopération. Ainsi la relation producteur/consommateur peut modéliser les interactions entre un contrôleur (qui joue le rôle de producteur) et un actionneur (consommateur), ou encore les interactions entre le capteur (producteur) et le contrôleur (qui joue le rôle cette fois du consommateur) (figures 4 & 11). Les évaluations de performances effectuées concernent des paramètres de qualité de service qu'offre le réseau à la relation producteur/consommateur. Par exemple : le délai moyen, et le taux de perte des trames. Ces évaluations sont réalisées en faisant varier : (1) – la stratégie d'ordonnancement utilisée par le commutateur, qui a un impact direct sur le service qu'offre le réseau, et (2) la charge du réseau, en ajoutant un trafic supplémentaire, pour perturber fortement la qualité de service offerte par le réseau.

4. 1. Hypothèses

Dans notre travail nous avons modélisé un commutateur à deux ports d'entrée et deux ports de sortie (figure 19). Le modèle du système en entier est représenté à l'annexe 3.
Le commutateur modélisé possède les paramètres résumés dans le tableau 1. C'est un commutateur full-duplex, avec un débit de 10 Mbit/s. Le temps minimal pour transmettre 64 octets est donc de 51,2 µs. Son temps de latence est fourni par le constructeur et est égal à 500ns pour un commutateur Cisco catalyst 2950-12-*SI* et de 101 ns pour le 2950-48-*EI*[30]. Puisqu'il est de l'ordre de la nanoseconde, celui-ci est négligé.
Nous avons posé pour hypothèse que les paquets échangés entre les producteurs et les consommateurs ont la même taille c'est-à-dire : 64 octets sans le préambule. De ce fait on prend cette taille comme unité de mesure pour modéliser le commutateur (taille d'une rafale, taille d'un buffer etc…). Nous rappelons que le mode de transmission utilisé est le ' *store & forward* ', par conséquent les paquets consommés ne sont pas erronés. Aussi puisque ce mode de transmission attend la réception totale de la trame avant son transfert vers sa destination, un délai supplémentaire dû à la longueur du paquet est considéré (51,2 µs). La transmission d'un paquet commence au moment où l'entité d'information qu'elle transporte a été produite.

[30] 2950 : code du commutateur, 12/ 48 : le nombre de ports, *SI*/*EI* : le logiciel et IOS utilisables.

Paramètres	Ethernet commuté
Débit (Mbit/s)	10
Durée d'un bit (bit time) (µs)	0.1
Taille max. du champ de donnée (Octet)	1500
Taille min. des messages (Octet)	64[31]

Tableau 1. Paramètres du commutateur modélisé

4. 2 Application à un exemple

Comme le montre la figure 19, chaque port d'entrée (I1, I2) est relié à un concentrateur (Hub) raccordé à trois producteurs périodiques. Aux paquets produits par le producteur P1, P2, et P3 sont associés respectivement une haute (H), une moyenne (M), et une basse priorité. A chaque port de sortie est relié un hub raccordé à 3 consommateurs périodiques de même période que les producteurs avec un déphasage 'ph'.
Les paramètres du modèle sont résumés dans le tableau 2.

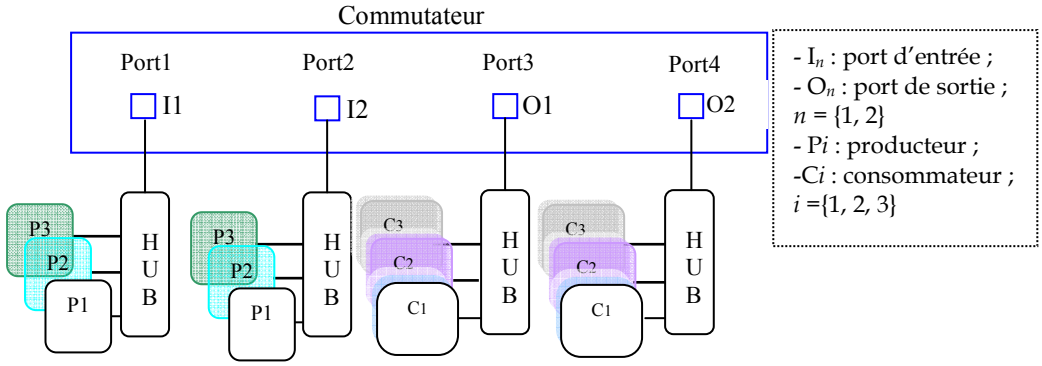


Figure 19. Structure du système modélisé : un exemple de LAN basé sur un commutateur Ethernet.

Paramètres	Ethernet commuté : valeurs réelles	Valeurs du modèle (ns)
Latence	500ns	500
Période du producteur	1ms	10^6
Période du consommateur	1ms	10^6
Délai dû au mode *store & forward*	51,2µs	$512 .10^2$
Trafic additionnel	4 paquets/1ms	4 paquets/10^6
	8 paquets/1ms	8 paquets/10^6

Tableau 2. Paramètres du modèle RDPCTH.

L'ensemble des mesures a été réalisé à l'aide de l'outil CPNTools, le but de l'analyse est d'évaluer la qualité de service fournie par le réseau (en l'occurrence ici c'est le commutateur) à la relation producteur/consommateur. Ces évaluations sont effectuées en faisant varier la charge du commutateur, et cela en ajoutant un trafic additionnel, et en faisant varier la politique d'ordonnancement utilisée par le commutateur (PS, WRR). Nous calculons alors :
Le délai moyen de transfert : il s'agit du temps moyen que met un paquet pour parvenir à sa destination à partir de l'instant où il est produit jusqu'au moment où il est consommé. En somme c'est le temps moyen de séjour dans le commutateur.

[31] Le préambule et le bit délimiteur non-inclus.

4. 3 Evaluation de la QdS fournie par le réseau :

Le modèle est simulé d'abord en fonction de la charge de l'environnement sans mécanisme d'ordonnancement. Ensuite, il est simulé avec le mécanisme d'ordonnancement à priorité statique, et enfin avec le mécanisme d'ordonnancement Weighted Round Robin. Deux scénarii de charges sont étudiés :

- Dans le premier scenario, seuls les buffers de sortie associés aux paquets de basses priorités sont chargés (trafic de fond),
- Dans le second scenario, les buffers de sortie associés à tous les types de paquets c'est-à-dire de haute, moyenne et basse priorité sont chargés de la même façon.

4. 3. 1 Modèle sans ordonnancement

Les résultats de simulation (figure 20) du modèle obtenus avec le logiciel CPNTools sont recueillis et traités pour les producteurs/consommateurs du port 1 (I1 : port d'entrée 1) et du port 3 (O1 : port de sortie 1)[32] (figure 19).

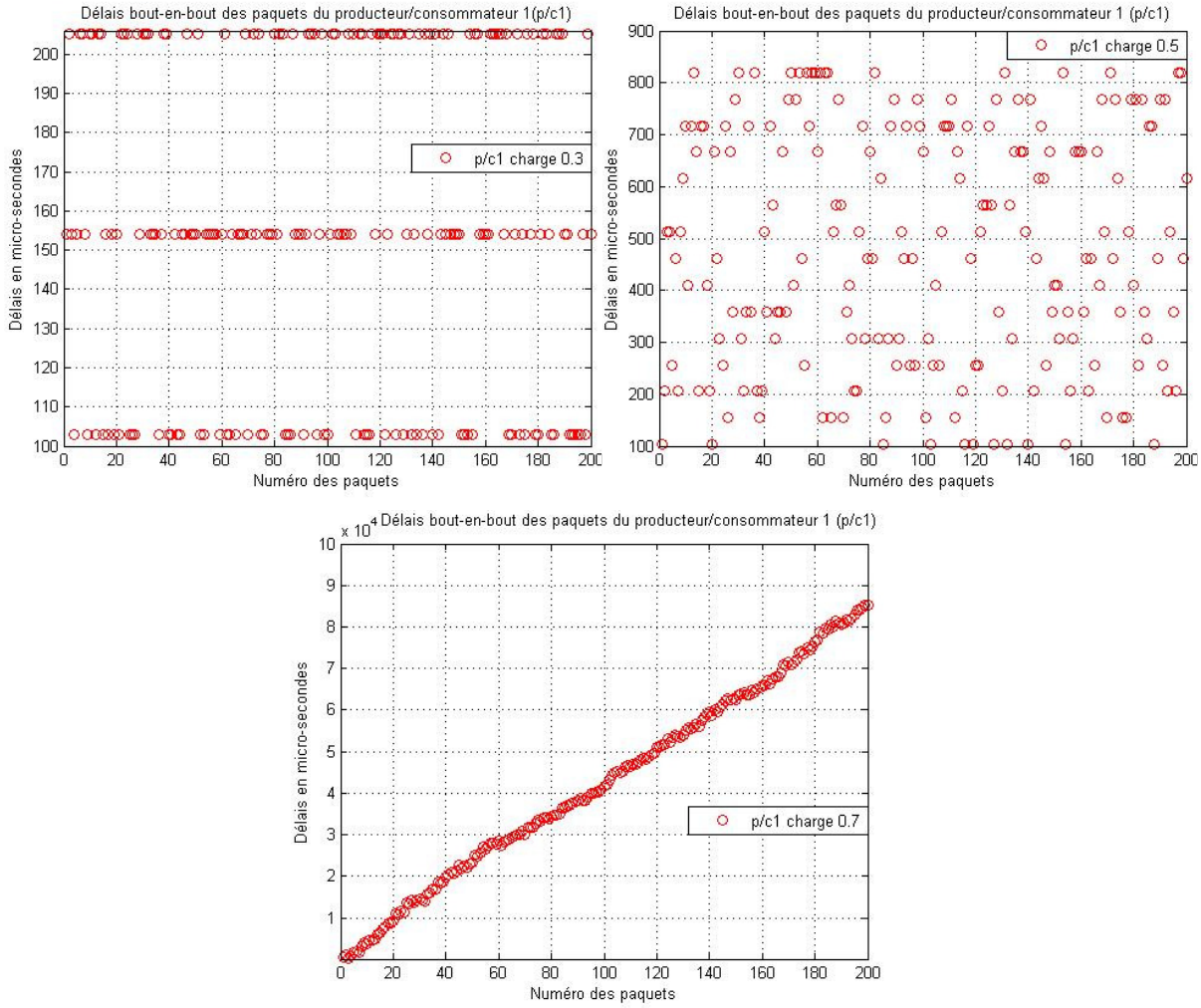


Figure 20. Modèle sans ordonnancement : Délais bout-en bout des paquets pour les différents producteurs/consommateurs (p/c1) pour une charge de 0.3, 0.5 et 0.7 du nœud.

[32] Il faut noter que nous obtenons les mêmes valeurs moyennes avec les autres producteurs/consommateurs des autres ports.

Les délais moyens en fonction de la charge illustrés sur la figure 21 montrent comme attendu une augmentation du délai en fonction de la charge du nœud. Les deux scenarii ne sont pas étudiés ici puisque le modèle étudié est sans ordonnancement.

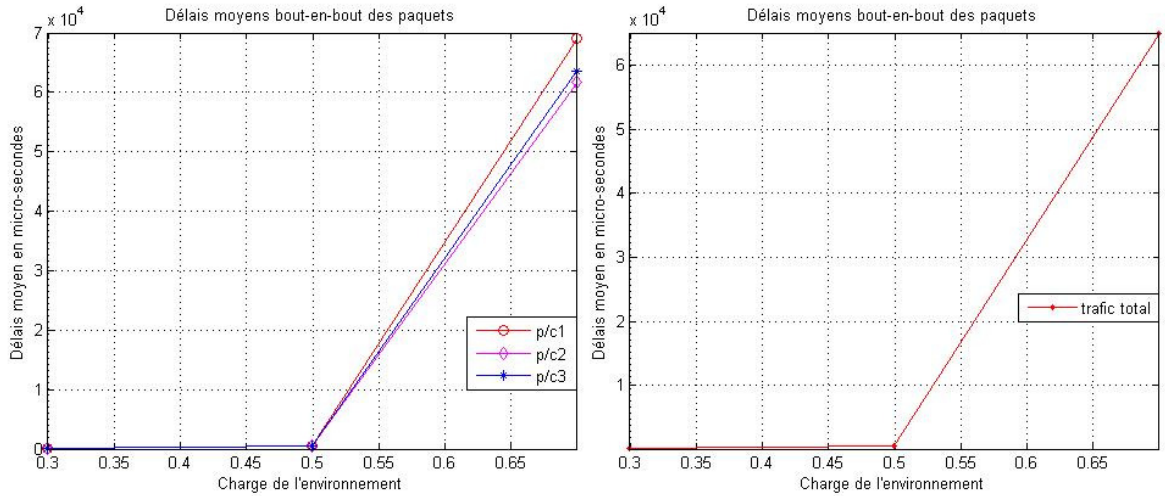


Figure 21. Modèle sans ordonnancement : Délais moyens bout-en-bout en fonction de la charge du noeud

4. 3. 2 Modèle avec un ordonnancement à priorité statique (PS) :

Dans le modèle RDPCTH, trois niveaux de priorité sont affectés aux messages venant des différents producteurs. Aux paquets générés par le producteur p1 sont affectés une haute priorité (HP), aux paquets générés par le producteur p2 sont affectés une moyenne priorité (MP) et aux paquets générés par le producteur p3 sont affectés la plus faible priorité (BP).
La figure 22 illustre les délais de bout-en-bout des paquets en fonction de la charge de l'environnement. Comme prévu les délais dépendent fortement de la charge du nœud et du scenario de charge appliqué.

Scenario 1 & 2:

Quel que soit le scenario appliqué pour le modèle du commutateur avec un ordonnancement à priorité statique son comportement est le même. C'est-à-dire qu'il privilégie la transmission des paquets de haute priorité. Lorsque le trafic de charge est de 0.3, le délai moyen des paquets HP est de 127.88 µs . Le plus petit délai noté pour ce type de paquet est évalué à 102.9 µs (51.2+0.5+51.2)[33] et le plus grand délai est égal à 154.1 µs. Ce délai est dû à l'attente de la transmission d'un paquet de priorité inférieure (51,2 µs) avant que le paquet de haute priorité soit transmis. Ce délai supplémentaire illustre le caractère non-préemptif de l'ordonnanceur PS. Ainsi, le délai bout-en bout des paquets de basse priorité augmente en fonction de la charge. Ceci est dû à la nature de l'ordonnanceur PS qui crée la famine dans les autres consommateurs, et qui classe cet ordonnanceur dans la catégorie des politiques d'ordonnancement 'non-équitable'.

[33] Délai minimal que peut avoir le paquet, le mode de transfert (mode *store & forward*)(51,2 µs), la latence (500ns) et le temps de transmission à partir de l'ordonnanceur vers le nœud de réception.

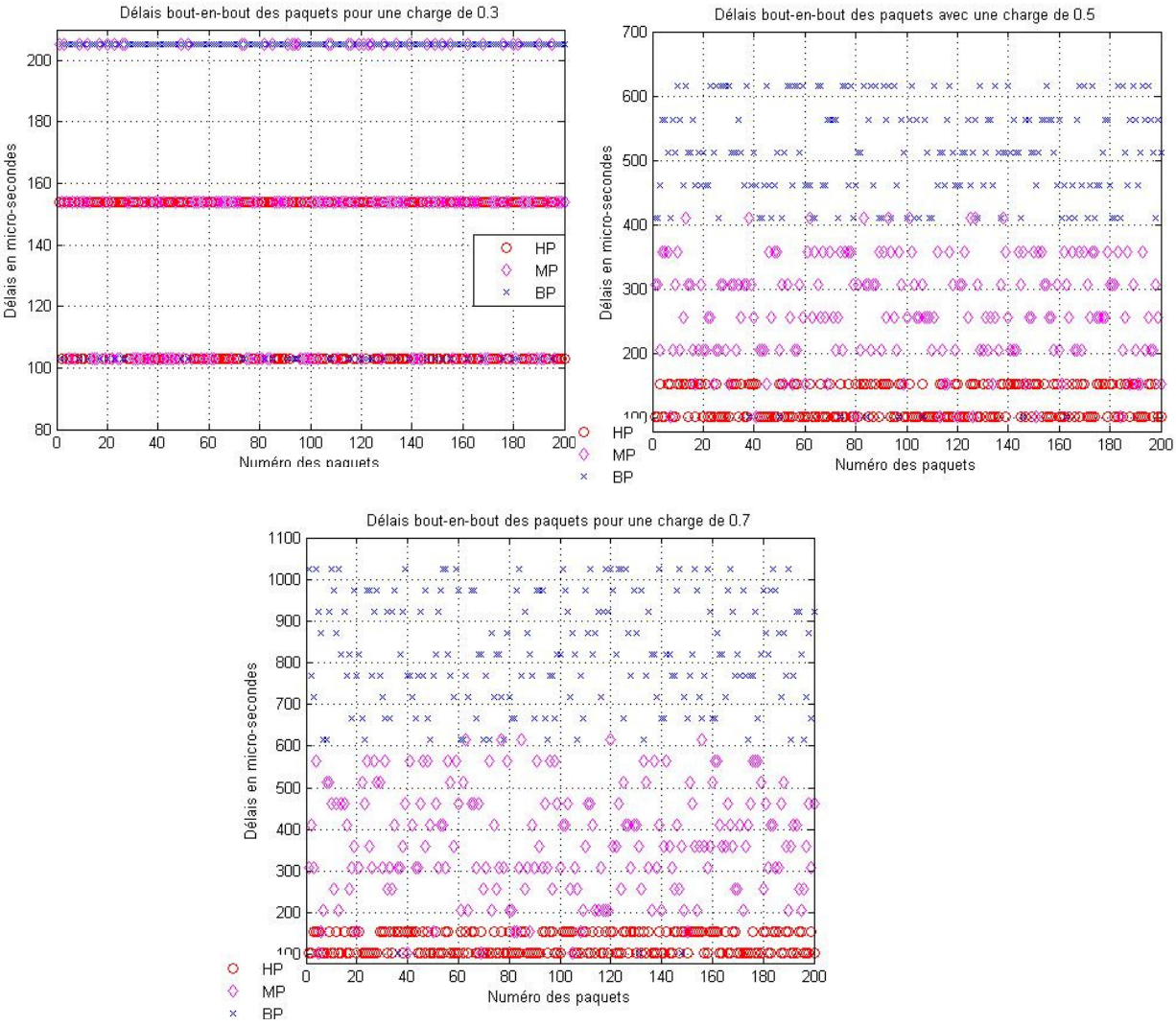


Figure 22. Modèle avec ordonnancement PS : Délais bout-en bout des paquets pour les différents producteurs/consommateurs (p/c1, p/c2, et p/c3) pour une charge de 0.3, 0.5 et 0.7 du nœud.

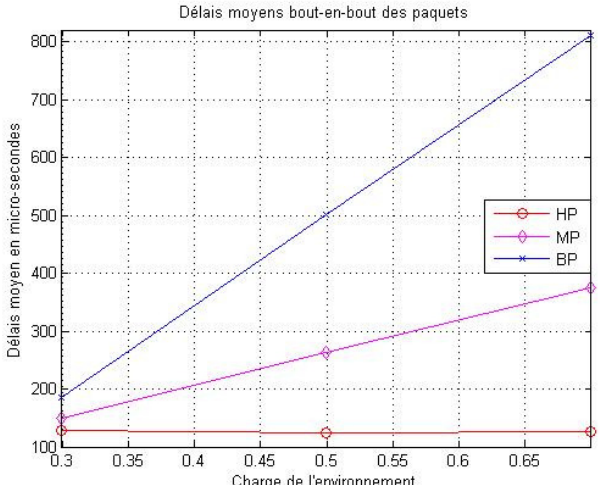


Figure 23. Modèle avec ordonnancement (PS): Délais moyens bout-en-bout en fonction de la charge du noeud

4. 3. 3 Modèle avec un ordonnancement Weighted Round Robin (WRR) :

Pour mener à bien la simulation de ce modèle, différents poids sont affectés au buffer contenant des paquets générés par le producteur p1 (w_1), au buffer contenant des paquets générés par le producteur p2 (w_2), et au buffer contenant des paquets générés par le producteur p3 (w_3).

Scenario 1 :

Nous rappelons que dans ce scenario, seuls les buffers associés aux paquets de basses priorités sont chargés. Les poids w_1 , w_2 , et w_3 permettent la transmission de 10%[34], 45% et 45% respectivement, des paquets mis dans les différents buffers. Les paquets générés par les différents producteurs p1, p2, et p3 sont toujours étiquetés HP, MP, et BP respectivement.
Les résultats de la figure 24 montrent avec un poids de 10% de la bande passante associée à la transmission des paquets de haute priorité, que plus la charge augmente plus le délai de bout en bout est important (figure 24 a, b).
Ce comportement peut potentiellement causer un non respect de la qualité de service que doit offrir le réseau spécialement lorsqu'il s'agit de paquets de haute priorité. Ce constat nous a amené à augmenter le poids associé aux paquets de haute priorité. Les poids w_1 , w_2 , et w_3 choisis à présent permettent la transmission de 99%, 0.99% et 0,01% respectivement. Ceci a pour résultat de diminuer le délai bout-en bout des paquets de haute priorité et notamment dans le cas où le réseau est fortement chargé 0 ,7 à 0,9 (figure 25).

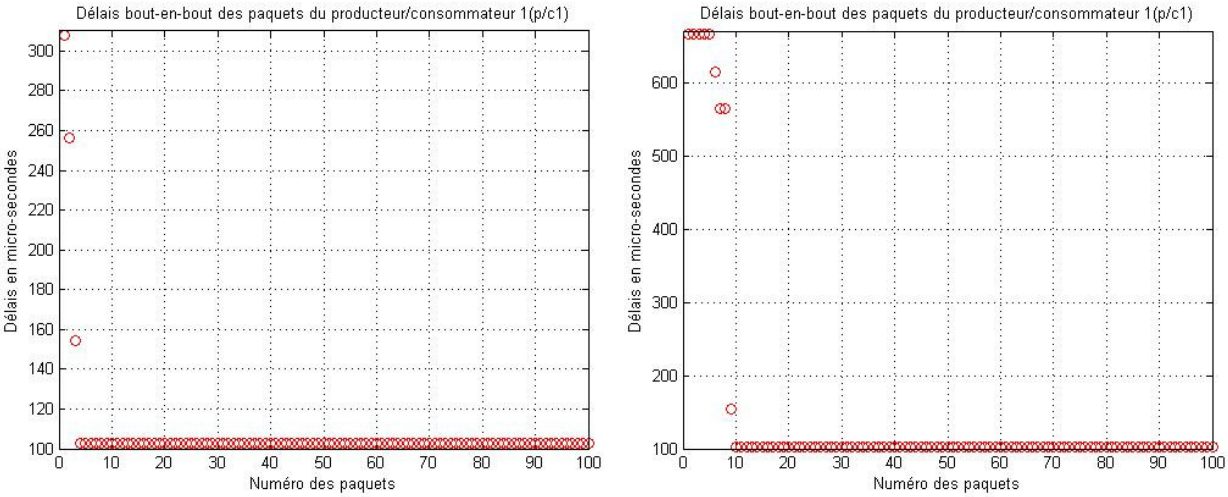


Figure 24. a. Modèle avec ordonnancement WR : Délais bout-en bout des paquets pour le producteur/consommateur (p/c1 : paquet HP) pour une charge de 0.3 et 0.5 du nœud.

[34] Le poids associés aux paquets de haute priorité est de l'ordre de 10% parce qu'en général ces paquets transportent des informations de commande qui ne nécessite pas un fort pourcentage de la bande passante.

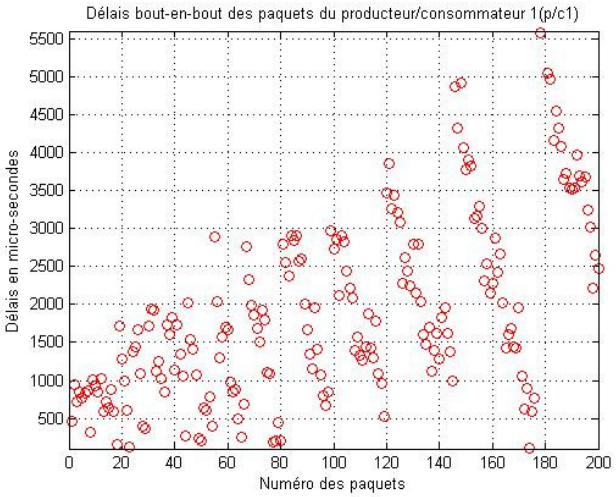


Figure 24. b. Modèle avec ordonnancement WR : Délais bout-en bout des paquets pour le producteur/consommateur (p/c1 : paquet HP) pour une charge de 0.7 du nœud.

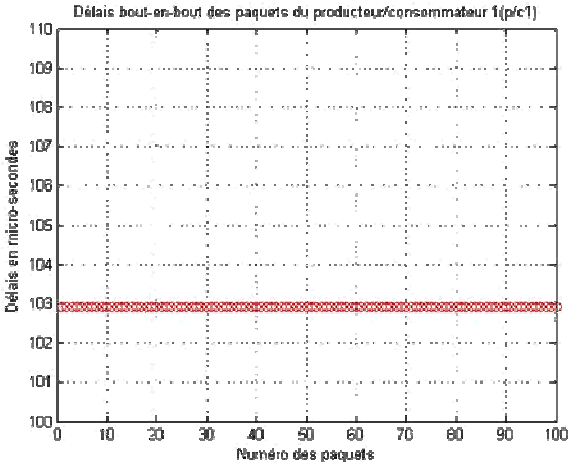


Figure 25. Modèle avec ordonnancement WRR : Délais bout-en bout des paquets pour le producteur/consommateur (p/c1 :paquet HP) pour une charge de 0.3, 0.5 et 0.7 du nœud.

La figure 26, résume le comportement du modèle RDPCTH avec l'ordonnanceur WRR en fonction de la charge et du poids associés à la file de sortie de haute priorité. Nous constatons qu'à un poids de 99% le comportement du commutateur avec un ordonnancement WRR est proche du comportement noté avec le commutateur dont l'ordonnancement est le PS.

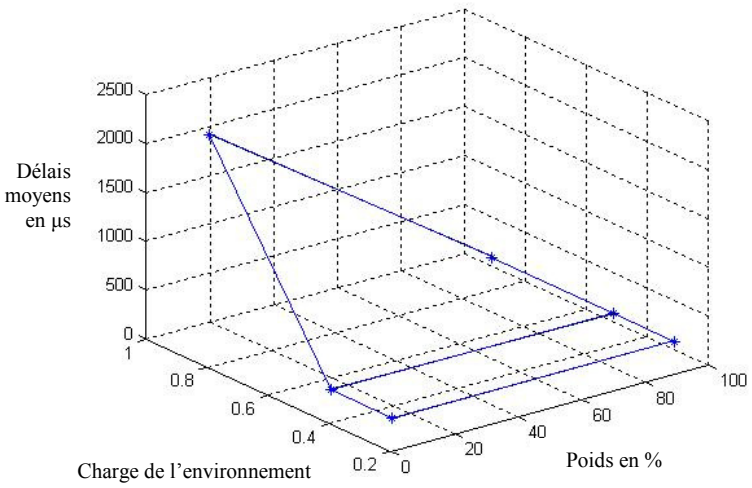


Figure 26. Modèle avec ordonnancement WRR : Délais moyens bout-en bout des paquets pour le producteur/consommateur (p/c1 : Paquet HP) en fonction de la charge (0.3, 0.5 et 0.7) et des poids associés à la transmission des paquets de haute priorité.

Scenario 2

Dans ce scenario, nous voulons voir le comportement du modèle dans le cas où la file de sortie de haute priorité a pour mission de transmettre plusieurs paquets de haute priorité venant de différentes sources. Son comportement est étudié en faisant varier la charge du nœud et cette fois ci les trois files de haute, moyenne et basse priorité sont chargés uniformément.
Par exemple dans le cas où la charge du nœud est égale à 0.5 : nous avons besoin de 9 producteurs périodiques qui génèrent des paquets d'une taille de 64 octets (51.2 µs) avec une période de 1ms[35]. Donc il y aura 3 producteurs périodiques de haute priorité, 3 producteurs de moyenne priorité, et enfin 3 producteurs de basse priorité.
Les poids w_1 , w_2 , et w_3 permettent la transmission de 60%, 30% et 10% respectivement, des paquets mis dans les différents buffers. Les paquets générés par les différents producteurs p1, p2, et p3 sont toujours étiquetés HP, MP, et BP respectivement.

[35] Pour le calcul de la charge du nœud il faut utiliser l'opération suivante : $\sum_{k=1}^{i} \frac{L_k}{T_k}$ sachant que L_k et T_k sont respectivement la longueur de la trame et la période associées au flux i.

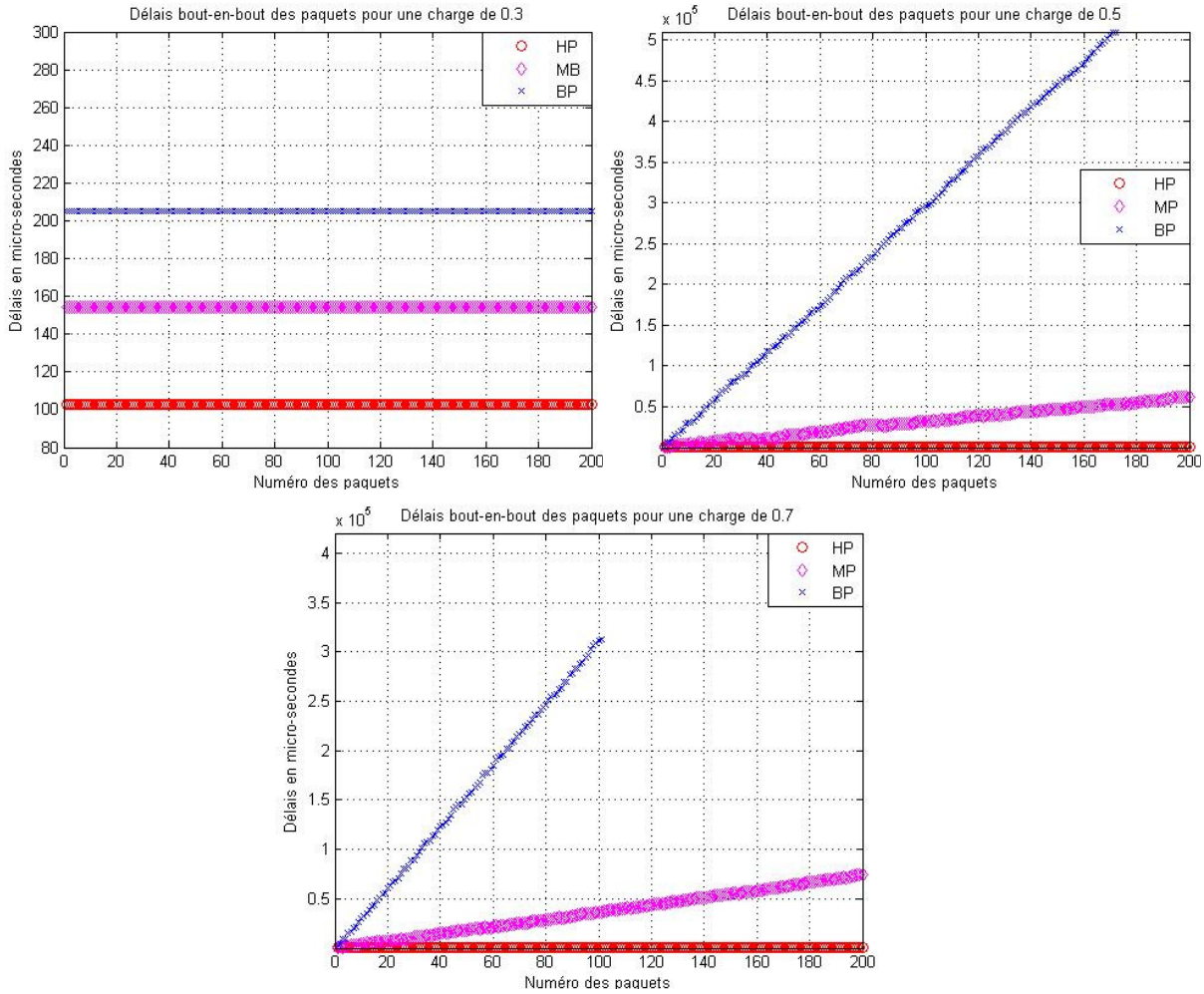
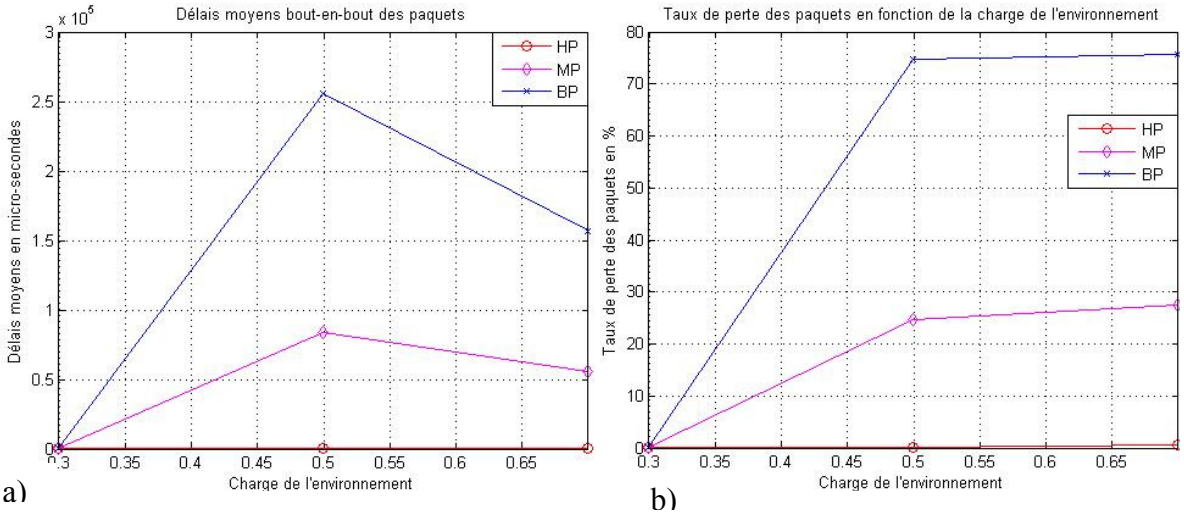


Figure 27. Modèle avec ordonnancement WRR : Délais bout-en bout des paquets pour les différents producteurs/consommateurs (p/c1, p/c2, et p/c3) pour une charge de 0.3, 0.5 et 0.7 du nœud.



a) b)

Figure 28. Modèle avec ordonnancement WRR : Délais moyens bout-en-bout en fonction de la charge du nœud (a), et taux de perte des paquets en fonction de la charge de l'environnement (b).

Les délais bout en bout des différents paquets sont illustrés à la figure 27. On remarque à l'évidence que les délais bout-en-bout dépendent de la charge du réseau. Cependant lorsque la charge est égale à 0.5, les délais moyens (figure 28. a) sont supérieurs aux délais moyens lorsque la charge du noeud est à 0.7 pour les paquets de priorité inférieure (MP et BP). Ceci semble inattendu. Cependant cela s'explique par le fait qu'il y a un taux de perte des paquets important lorsque la charge est à 0.7 (figure 28. b). De ce fait le nombre de paquets consommés à 0.5 est supérieur au nombre de paquets consommé à 0.7.

4. 3. 4 Comparaison

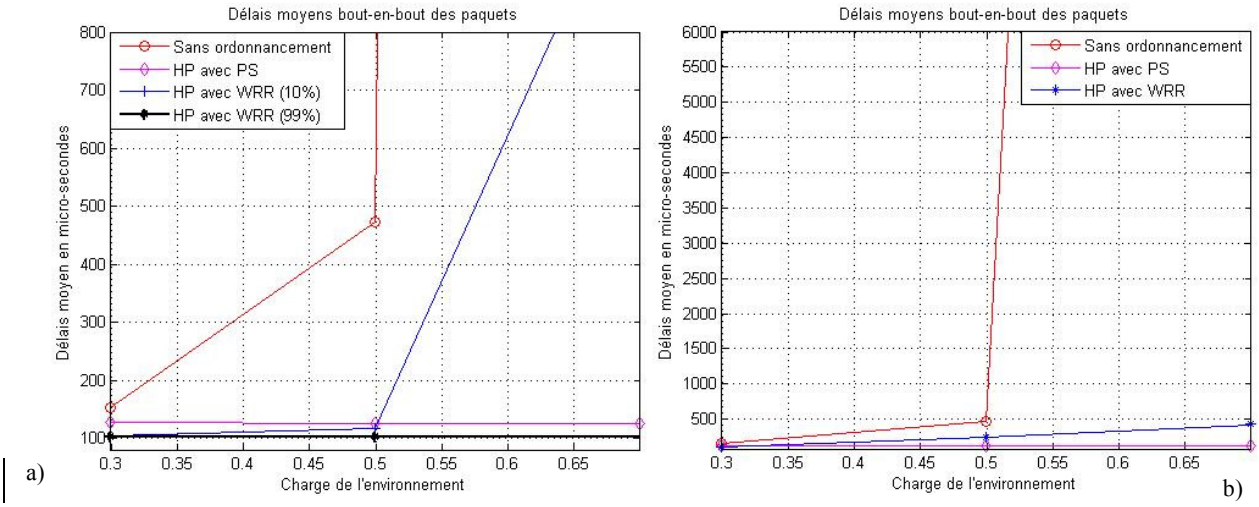


Figure 29. Délais moyens des paquets de haute priorité pour le modèle simulé sans ordonnanceur, avec un ordonnancement PS, et avec un ordonnancement WRR : a) scenario 1, b) scenario 2.

La figure 29 illustre les délais moyens des paquets en fonction de la charge du nœud pour différents modèles de commutateur : commutateur sans ordonnanceur, commutateur avec un ordonnancement PS, et commutateur avec un ordonnancement WRR. Dans le cas où une politique d'ordonnancement est utilisée, seuls les paquets de HP sont représentés.

Dans le cas du scénario 1, quand la charge est à 0.3 le délai moyen évalué avec la politique WRR est plus petit et est égal à 102.9 µs. Lorsque la politique PS est utilisée le délai moyen est égal à 127,88 µs. Ceci est dû à son caractère non-préemptif déjà expliqué plus haut. Enfin lorsque aucun ordonnanceur n'est utilisé le délai est de 152,97 µs. Nous pouvons dire qu'à faible charge l'utilisation du WRR quelque soit le poids associé à la transmission des paquets de haute priorité, semble intéressante pour deux raisons : d'abords pour le faible délai qu'il induit ensuite pour sa qualité d'équité entre les différents paquets de différentes priorités. Ce comportement change complètement lorsque la charge augmente et que le poids associé aux paquets HP est de 10%. Le délai comme attendu augmente et l'utilisation de l'ordonnanceur PS semble dans ce cas présent plus pratique puisque la qualité de service offerte par celui-ci est meilleure : délai est de 126.21 µs.
Cependant, nous avons remarqué qu'en changeant de poids aux paquets de HP dans le WRR son comportement s'apparente à celui de l'ordonnancement à PS et offre même de meilleurs résultats (voir tableau 4).

Dans le cas du scenario 2, plus la charge augmente plus le délai moyen augmente pour un commutateur sans ordonnanceur, et avec un ordonnanceur WRR. A l'évidence dans le cas où

la charge du nœud est importante l'ordonnanceur PS est plus performant (tableau 5) puisqu'il ne sert que les paquets de haute priorité.

		Charge		
		0.3	0.5	0.7…0.9
Délai Moyen (µs)	Sans ordonnanceur	152.97	472.05	69108.15
	Avec PS	127.88	125.18	126.21
	Avec WRR (w_1 : 10%)	103.31	116.25	2140.27
	Avec WRR (w_1 : 99%)	102.9	102.9	102.9

Tableau 4. Délai moyen en fonction de la charge du nœud des paquets HP pour différentes politiques d'ordonnancement de trames : scenario 1.

		Charge		
		0.3	0.5	0.7…0.9
Délai Moyen (µs)	Sans ordonnanceur	152.97	472.05	69108.15
	Avec PS	127.88	125.18	126.21
	Avec WRR (w_1 : 60%...90%)	102.9	247.177	426.559

Tableau 5. Délai moyen en fonction de la charge du nœud des paquets HP pour différentes politiques d'ordonnancement de trames : scenario 2.

5. Conclusion

Dans cette partie un modèle de commutateur est réalisé et ses performances sont estimées en simulant le modèle avec le logiciel CPNTools. Les délais moyens de bout-en-bout fournis par le modèle du commutateur sont évalués en fonction de la charge de l'environnement et de la politique d'ordonnancement de trames implantée. Comme attendu plus la charge augmente plus le délai moyen bout en bout offert par le réseau (le commutateur) à la relation producteur/consommateur augmente.
Deux scenarii ont été étudiés. Dans le premier, lorsque la charge du nœud est égale à environ 0.3 l'utilisation de l'ordonnanceur WRR semble plus intéressante. Quand la charge augmente, l'ordonnanceur à PS est plus efficace. Cependant nous avons constaté qu'en augmentant le poids associé à la transmission des paquets de HP, l'utilisation du WRR comme politique d'ordonnancement de trames dans le commutateur Ethernet offre de meilleures performances.
Dans le second cas, lorsque le réseau a pour mission de transmettre plusieurs informations de haute priorité, l'ordonnanceur WRR est efficace pour de faibles charges, et l'ordonnanceur PS offre de meilleures performances pour des charges importantes.
De ce constat il serait intéressant d'utiliser les performances de chaque ordonnanceur pour rendre le commutateur plus performant, cependant la problématique qui s'impose à présent est la façon de les utiliser.
La manière la plus évidente consiste en un réglage des poids du WRR suivant le retard induit par le réseau pour le premier scenario.
L'autre façon consiste en une commutation d'un ordonnancement WRR à un ordonnancement PS dès que la charge du nœud devient importante (voir second scenario).

Chapitre IV : Modélisation Intégrée d'un SCR avec les RDPC et Evaluation de performances sur une étude de cas

1. Introduction

Ce chapitre est dédié à la présentation, dans un premier temps, du modèle RDPCTH du système contrôlé en réseau. Son évaluation de performances sera effectuée en confrontant les résultats de simulations obtenus avec CPNTools aux résultats du même modèle construit avec Matlab. Ensuite une méthode basée sur un ordonnancement de trames sera proposée afin de réduire le délai induit par le réseau (commande du réseau). L'ensemble sera modélisé avec les RDPCTH et évalué par simulation du modèle avec le logiciel CPNTools.
Nous nous appuierons sur une modélisation modulaire et hiérarchique pour représenter le système contrôlé en réseau. Les systèmes de commande représentés sont des systèmes à temps discrets.

2. Modélisation du système contrôlé en réseau par les RDPCTH

2. 1 Représentation hiérarchique

Le modèle hiérarchique du RDPCTH représentant le système contrôlé en réseau est illustré par la figure 1. La hiérarchie du modèle englobe les éléments constituant un SCR : la commande, le réseau représenté par le commutateur Ethernet, l'actionneur, le système à commander et le capteur. A chaque élément correspond un module. Et chaque module est représenté dans les RDPC par une transition d'abstraction qui détaille la modélisation de chaque élément. Le niveau hiérarchique le plus élevé permet de comprendre la composition générale du système. Cette modélisation a été réalisée en faisant une extension du modèle du commutateur déjà effectué dans le chapitre 3.

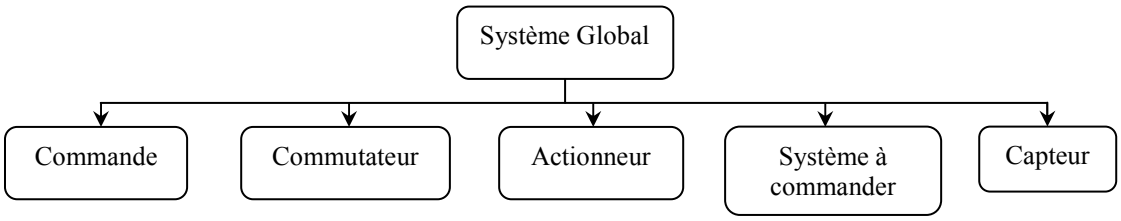


Figure 1. Représentation hiérarchique du système contrôlé en réseau.

La première modification concerne la définition de la couleur 'paquet'. En effet, en plus des éléments précédemment définis tels que : l'adresse de destination, l'adresse source, le tag représentant le niveau de priorité de la trame; nous avons ajouté un champ 'data' dans le produit de la couleur 'paquet' correspondant à la donnée à transmettre. Cette donnée permet de transporter la consigne calculée par le contrôleur qui est envoyée à l'actionneur à travers le réseau, ainsi que la valeur mesurée de l'état du système envoyée au contrôleur.

```
Colset paquet = product inp * outp * prio*data timed;
Var  i: paquet;
```

2. 1. 1 Le système global

Le système contrôlé par le réseau dans sa globalité est illustré par le RDPCTH de la figure 2. Chaque transition d'abstraction (contrôleur, commutateur Ethernet, actionneur, process et capteur) représente un RDPCT (ou RDPCTH pour la transition d'abstraction commutateur Ethernet), qui sont expliqués dans les sections suivantes. Les places illustrées dans ce modèle représentent les ports d'entrée et de sortie entre chaque module.

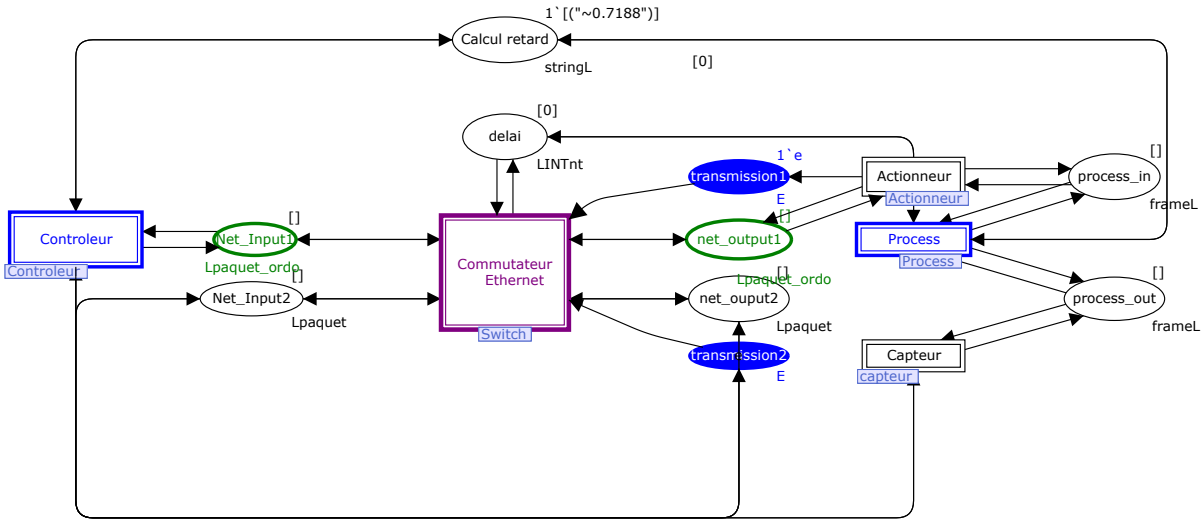


Figure 2. Module RDPCTH d'un système contrôlé en réseau.

2. 1. 2 Le modèle du système à commander

Le type de système à commander (process) modélisé dans notre travail est un système linéaire à temps discret décrit par l'équation suivante :

$$x(k+1) = Ax(k) + Bu(k) \qquad (1)$$
$$y(k) = Cx(k)$$

Où k est l'indice de temps. Une unité de temps représente la période d'échantillonnage T_s, *et* $x \in \Re^n$, $y \in \Re^m$ et $u \in \Re^l$ sont respectivement, les vecteurs d'états, de sorties et d'entrées du système. A : la matrice d'évolution, B : la matrice de commande, C : la matrice d'observation.
Nous supposons que le système est stable pour : $u(k) = -Fx(k)$, F représente la commande optimale du système.
Dans le modèle illustré sur la figure 3, le segment de code est utilisé afin de calculer la sortie et l'état du système. Les valeurs de l'état sont lues et écrites dans une mémoire locale afin d'être utilisées par le segment de code pour le calcul des prochaines valeurs.

Sur la figure 3, l'équation xk1 = (A1_1*xk1_1)+(B1_1*uk_1) qui se trouve dans le segment de code associé *plant* représente l'équation de récurrence suivante $x(k) = Ax(k-1) + Bu(k-1)$. xk_1, uk_1 dans le graphe représentent respectivement : x(k-1) et u(k-1). L'état xk (x(k)) du système est calculé à chaque période d'échantillonnage Te en fonction de l'état précédent xk_1 et de la commande précédente uk_1 qui vient de la place *process_in*. La nouvelle valeur de xk est calculée en exécutant le segment de code associé à la transition *plant*. La valeur résultante est mise dans la place *process_out* , et dans la mémoire locale du système. La place *process_out* est le port de sortie du process (voir le tag associé –Out-) et la place d'entrée du capteur. La place *process_in* représente le port d'entrée (tag associé –In-) et le port de sortie de l'actionneur.

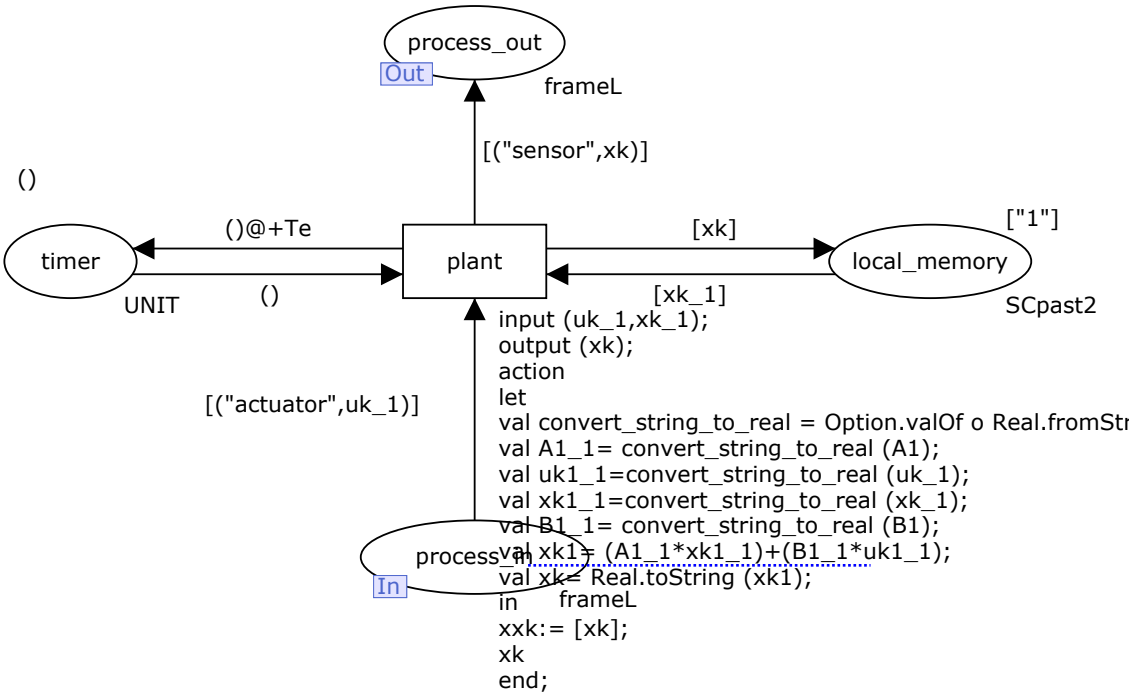


Figure 3. Modèle RDPCT du système à commander (le process)

2. 1. 3 Le modèle de l'actionneur

La figure 4 représente le modèle de l'actionneur guidé par les évènements. En effet, dès qu'il reçoit la valeur de commande calculée venant du réseau, représentée par la variable *uk* , la transition *actuator* est franchie immédiatement, et la valeur de uk est mise dans la place *process_in*. La place *process_in* correspond au port de sortie du module actionneur et au port d'entrée du module précédent.

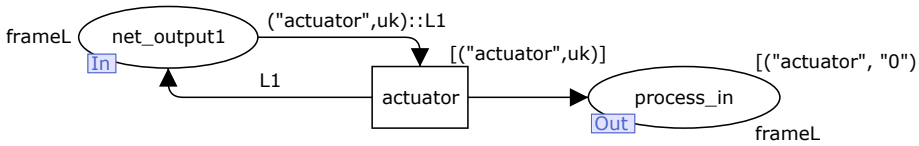


Figure 4. Modèle RDPCT de l'actionneur

2. 1. 4 Le modèle du capteur

La figure 5 représente un capteur événementiel. Dès que la valeur xk est calculée par le process, celle-ci est mise dans la place port d'entrée process_out (port de sortie du module système à commander) permettant le franchissement immédiat de la transition *sensor*. La valeur xk est envoyée alors vers le réseau. Le réseau est représenté par le modèle commutateur Ethernet exposé dans le chapitre 3.

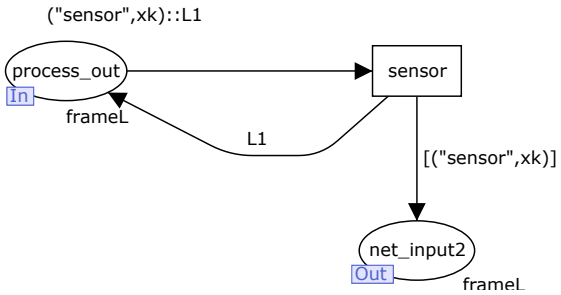


Figure 5. Modèle RDPCT du capteur.

2. 1. 5 Le modèle de la commande

La place *net_output2* reçoit la valeur d'état xk calculée précédemment par $x(k) = Ax(k-1) + Bu(k-1)$ (figure 3), et envoyée à travers le réseau. Cette valeur xk permet à la transition *controller* d'être active et d'exécuter le segment de code qui lui est associée. L'équation uk1= k1_1*(cons1-xk1) dans ce segment de code (figure 6) représente la valeur de commande qui va être calculée. Elle est exprimée en fonction de *k1_1* (*F* : équation 1) qui est la commande optimale pré-calculée sous Matlab, la valeur de l'entrée de référence *cons1* et de l'état *xk*.

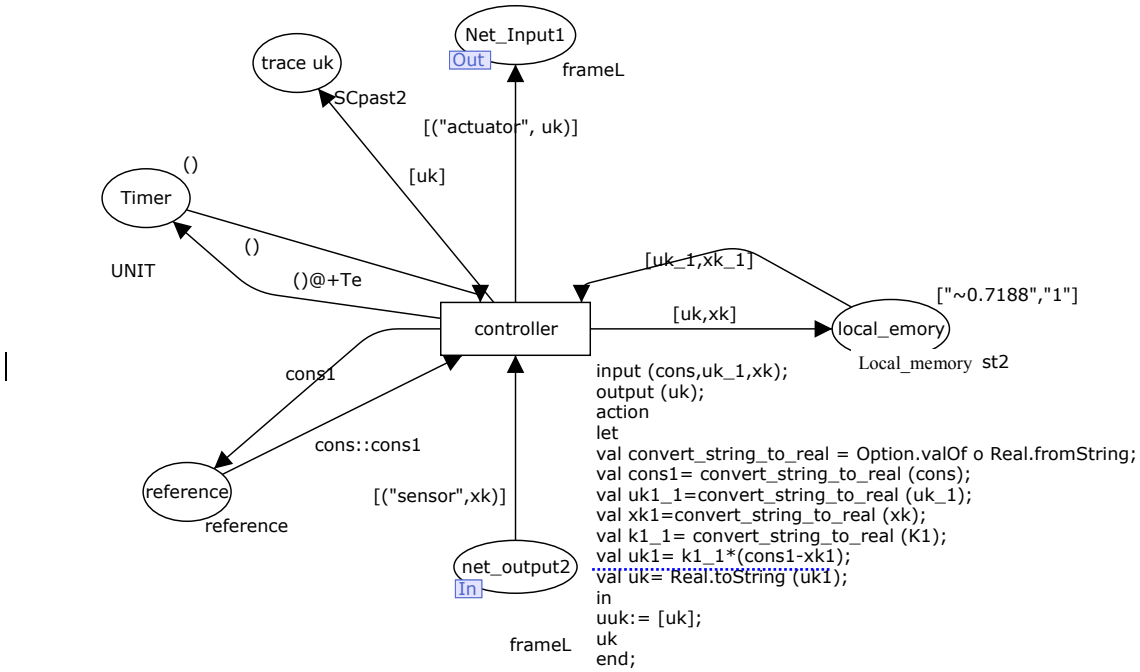


Figure 6. Modèle RDPCT du contrôleur

Remarque

Les éléments de base constituant un système en boucle fermée ont été exposés précédemment. Cependant les opérations d'encapsulation et de désencapsulation n'ont pas été présentées. L'opération d'encapsulation et de désencapsulation consistent respectivement à mettre l'information de commande (ou d'état) dans une trame pour pouvoir l'envoyer dans le réseau, et à extraire l'information de la trame utile pour le process (ou la commande). Ceci peut être réalisé grâce à l'opération de transformation de couleur que permet les RDPC.

Nous avons ajouté dans le module contrôleur une source de trafic pour évaluer les performances du système dans le cas où le réseau est chargé. Des consommateurs guidés par les événements sont également ajoutés au niveau de l'actionneur pour que le trafic supplémentaire généré soit consommé.

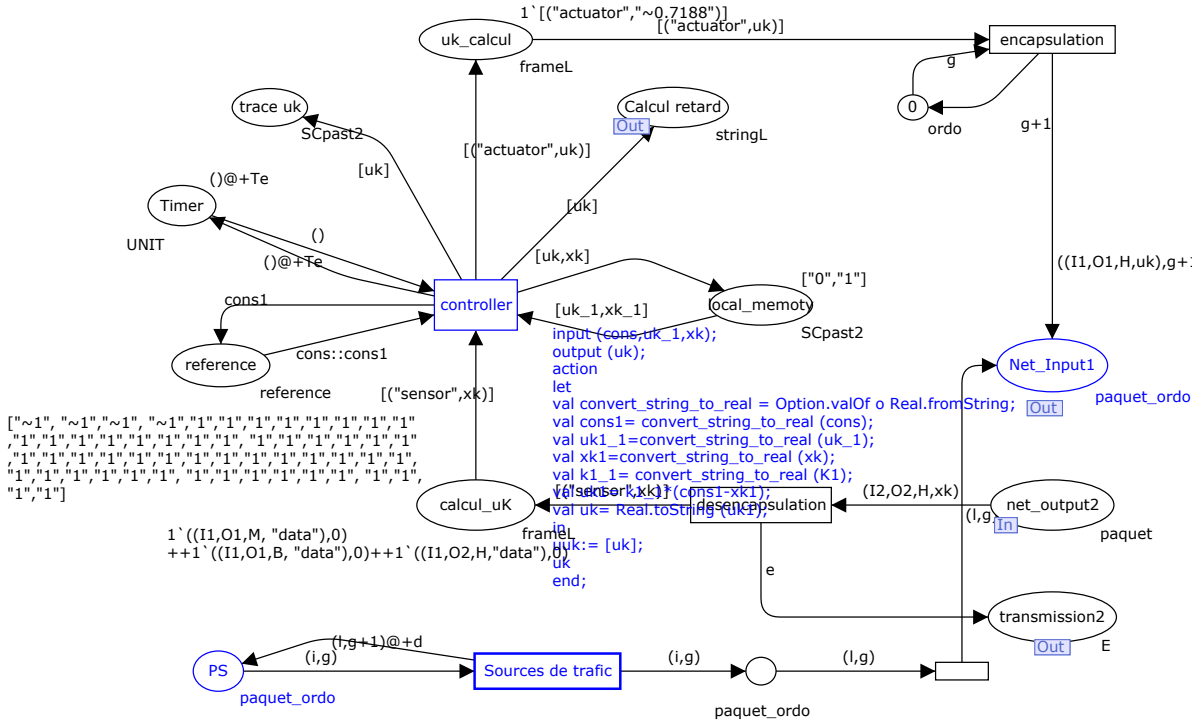


Figure 7. Modèle RDPCT du contrôleur avec la source de trafic supplémentaire

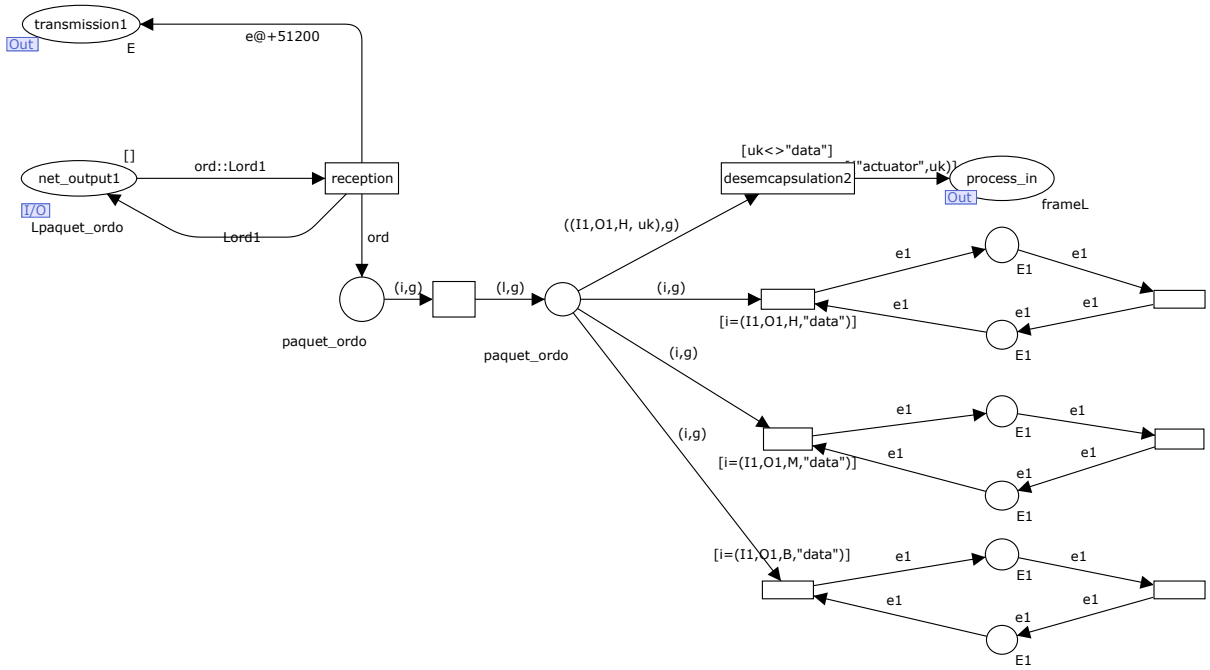


Figure 8. Modèle RDPCT de l'actionneur avec les consommateurs événementiels

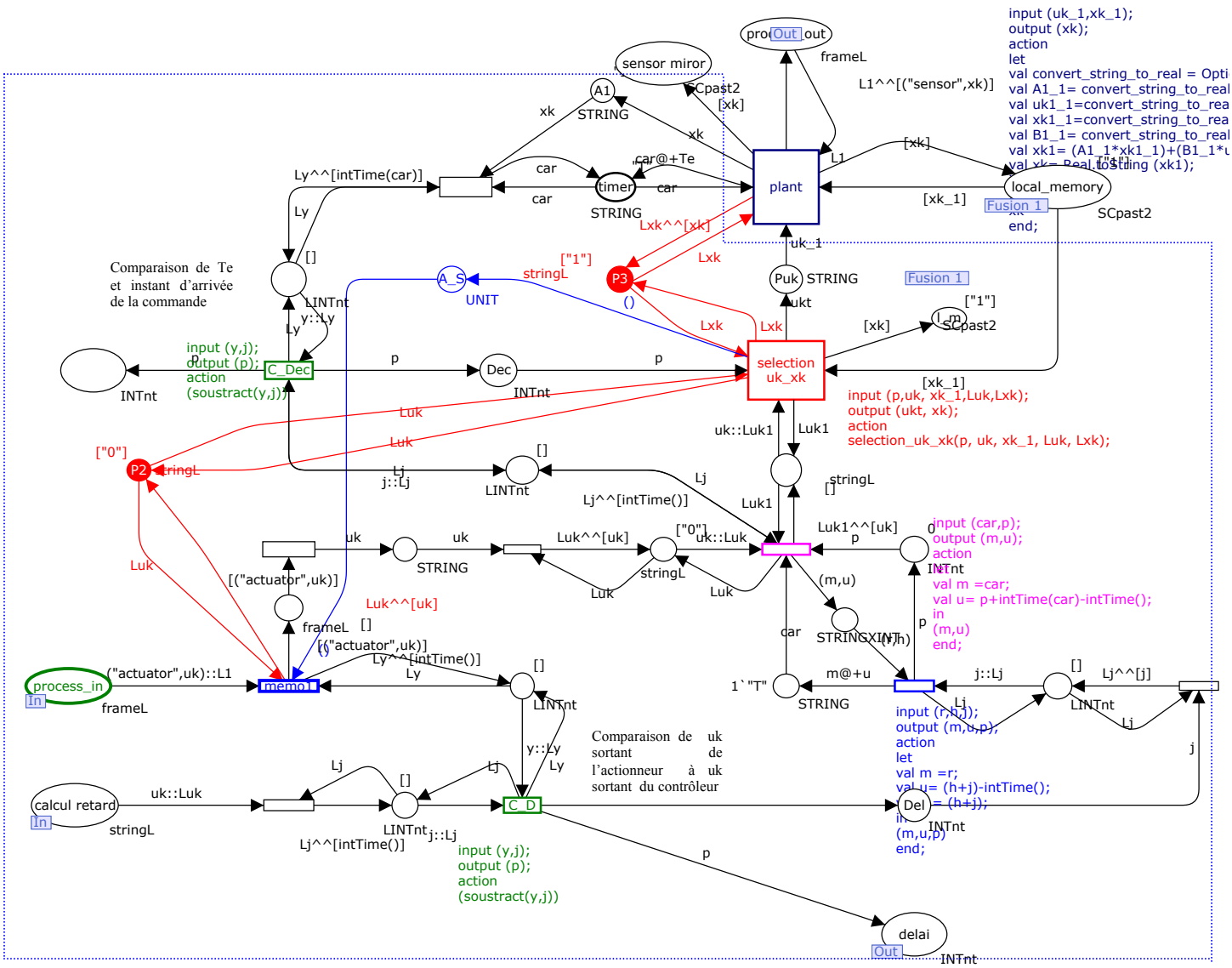


Figure 9. Modèle RDPCT du module process avec le dispositif de calcul de décalage et calcul de délai

2. 1. 6 Calcul du décalage entre la période d'échantillonnage du process et l'instant de réception de la commande

La comparaison de la période d'échantillonnage du système à l'étiquette temporelle de la valeur de la commande reçue permet de :

- calculer le décalage entre l'instant de calcul de l'état précédent et de l'instant de réception de la commande. Cela va servir à calculer le prochain état du système (segment de code associé à la transition *C_Dec*) (figure 9). Pour que cette opération soit séquentielle, le calcul du décalage est effectué après chaque calcul du nouvel état xk. Pour mieux comprendre la notion de décalage[36] la figure 10 illustre un exemple de SCR où le délai induit par le réseau est inférieur à une période d'échantillonnage. Pour calculer le prochain état à l'instant (k+1)*Te, le système commandé doit connaître l'état passé du système à k*Te ainsi que la commande calculée et envoyée au réseau à k*Te. Cette commande arrive à l'actionneur avec un retard Δ que nous

[36] Nous aurions pu l'appeler décalage 'délai', ce qui revient à la même chose. Mais puisque ce terme est utilisé plus bas à d'autres fins, nous avons pris la liberté de les appeler différemment pour les distinguer.

avons appelé décalage. L'opération est réalisée en fonction de l'axe temporel du système commandé.

- choisir les valeurs de commande et d'état, passées enregistrées dans deux mémoires tampons, représentées par les places *P2* et *P3* respectivement (figure 9). Les valeurs de l'état du système et de la commande appropriés sont sélectionnées suivant le décalage calculé à l'aide de la fonction selection_uk_xk(i, uk, xk_1, Luk, Lxk). Cette fonction est représentée dans le segment de code associé à la transition *selection uk_xk* (figure 9). Les valeurs sélectionnées servent à calculer le prochain état du système.

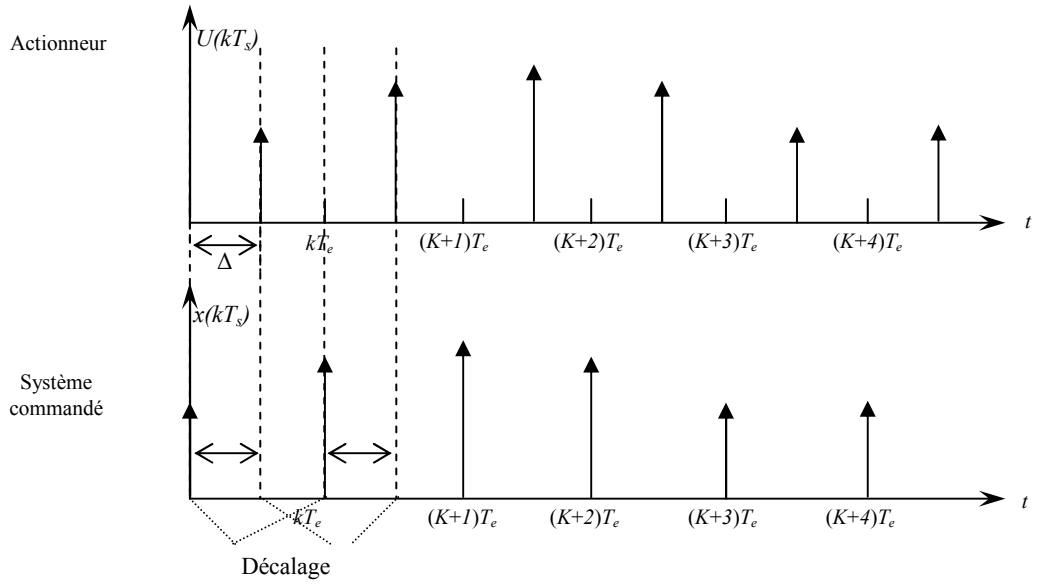


Figure 10. Exemple de calcul de décalage suivant l'axe temporel du système commandé.

Fonction *selection_uk_xk* a donc pour tâche de sélectionner, la commande et l'état passé mémorisées dans les mémoires représentées par *P1* et *P2* (figure 9). Elle s'exprime comme suit :

```
fun selection_uk_xk(i:int, uk:string, xk_1:string, Luk:stringL, Lxk:stringL)=
let
val n=length Luk;
val m=length Lxk;
in
if i=0 then (if n=1 then (uk, xk_1)
  else if n=2
  then (List.nth(Luk, (n-2)), List.nth(Lxk,(m-1)))
  else if n>=3 then (List.nth(Luk, (n-1)), List.nth(Lxk, (m-1)))
  else (uk,xk_1))
else if i<0 andalso i>(~Te) then
  (if n=1 then (uk,xk_1) else if n>=2 then
  (List.nth(Luk, (n-2)), List.nth(Lxk, (m-1)))
  else  (List.nth(Luk, (n-2)), List.nth(Lxk, (m-1))))
else if i=(~Te) then (if n= 1 then(uk,xk_1)
  else if n =2 then
  (List.nth(Luk, (n-2)), List.nth(Lxk, (m-1)))
  else if n=3 then
  (List.nth(Luk, (n-3)), List.nth(Lxk, (m-1)))
  else if n>=4 then
```

```
  (List.nth(Luk, (n-2)), List.nth(Lxk, (m-1)))
  else (uk,xk_1))
else if i>(~2*Te) andalso i<(~Te) then
  (if n=1 then (uk, xk_1) else if n=2 then
  (List.nth(Luk,(n-2)), List.nth(Lxk, (m-1)))
  else if n>=3 then
  (List.nth(Luk, (n-3)), List.nth(Lxk, (m-1)))
    else (uk,xk_1))
else if i = (~2Te) then (if n=1 then (uk, xk_1)
  else if n=2 then
  (List.nth(Luk,(n-2)), List.nth(Lxk, (m-1)))
  else if n=3 then
  (List.nth(Luk, (n-3)), List.nth(Lxk, (m-1)))
  else if n=4 then
  (List.nth(Luk, (n-4)), List.nth(Lxk, (m-1)))
  else if n>=5 then
  (List.nth(Luk, (n-3)), List.nth(Lxk, (m-1)))
  else (List.nth(Luk, (n-4)), List.nth(Lxk, (m-1))))
else (uk, xk_1)
end
```

Pour que ces opérations soient séquentielles, il faut que la mémorisation de uk dépende de xk. C'est-à-dire qu'on autorise la mémorisation d'une nouvelle valeur de uk, que si la fonction selection _uk _xk a été exécutée. Cela est réalisé grâce à la place *A_S* (figure 9) qui relie la transition *selection uk_xk*(figure 9) à la transition *memo* (figure 9).

2. 1. 7 Calcul du délai

Pour prendre en compte le délai induit par le commutateur Ethernet, une procédure de calcul de délai a été élaborée. La valeur de la commande calculée dans le module de commande est envoyée vers le module commutateur par la place port *net_input1* et dans la place *calcul retard* qui est une place port d'entrée du process (figure 7). Dès que l'actionneur reçoit la trame, celui-ci les envoie au process par la place port process_in. Les étiquettes temporelles associées aux jetons qui arrivent à la place *process_in* et à la place *calcul retard* sont extraites grâce à la fonction intTime() : fun intTime(i)=IntInf.toInt(time()).
Cette fonction traduit la conversion de l'étiquette temporelle qui est de la forme @+T (exp : @+5) en un entier. Une opération de soustraction est appliquée à ces valeurs grâce à la fonction soustract(i,y) (transition *C_D*) qui est associée à une transition. La différence est obtenue dès que celle-ci est exécutée[37]. Cette valeur peut être ensuite réutilisée, de deux façons :

- Premier cas : Les délais sont supérieurs à une ou plusieurs périodes d'échantillonnage. Comme les RDPCT dans CPNTools sont P-temporisés, lorsque la transition *plant* est active le jeton attend *Te* unité de temps pour envoyer l'état calculée xk. Cependant, imaginons que le jeton porteur de l'information uk possède une étiquette temporelle (t1) dont la valeur est supérieure à l'étiquette temporelle (t2) associée au jeton dans *timer,* le jeton dans *timer* doit alors attendre (t1-t2) unité de

[37] Il faut noter que nous avons utilisé la notion de list expliquée dans le chapitre 2 avec les opérations de concaténation et d'extraction du premier élément de la list pour éviter un mélange entre les différentes étiquettes temporelles. Avec ce procédé, les étiquettes de la même trame immédiatement calculée par la commande et reçue par l'actionneur sont comparées et leur différence calculée.

temps avant que la transition plant ne soit franchissable. Ensuite la nouvelle valeur calculée xk est envoyée au capteur après Te unité de temps. En somme, le jeton dans *timer* aura à présent une étiquette de (t1-t2)+Te risquant de causer un décalage erroné dans le traitement de données (Transition *selection uk_xk*). La fonction selection _uk _xk associée à la transition *selection uk_xk* choisit les états à prendre en compte pour le calcul du prochain état du système.

- Deuxièmement, la connaissance de cette valeur est utile pour commander le réseau afin minimiser les délais. Cette partie sera exposée dans les prochaines sections.

3. Evaluation de performances du modèle RDPCHT d'un système contrôlé en réseau

Dans cette section, un système contrôlé en réseau est simulé. Les résultats obtenus seront analysés et comparées avec des modèles Matlab. Pour ce faire, les délais de communication obtenus par notre modèle seront utilisés dans le modèle Matlab.

L'exemple suivant est considéré dans notre analyse :

$$x(k+1) = 1.05x(k) + 1.8u(k) \qquad (2)$$

Où $Q = 2$, $R = 7$, la commande optimale $F = 0.3594$. Te = 1ms

3. 1 Système de commande en boucle fermée

La simulation du système de commande en boucle fermée (sans réseau) est effectuée dans un premier temps en utilisant notre modèle RDPTH puis en utilisant Matlab (figure 11). La figure 12 montre que les résultats obtenus par ces deux outils sont semblables.

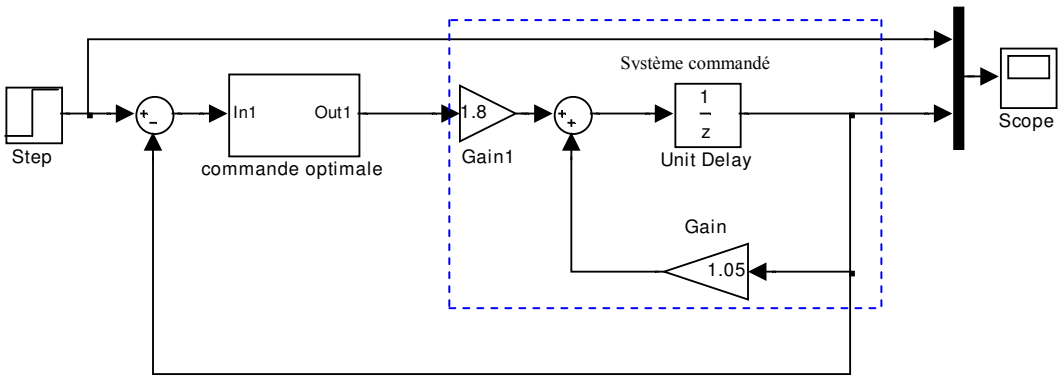


Figure 11. Système de commande en boucle fermée représenté avec Matlab.

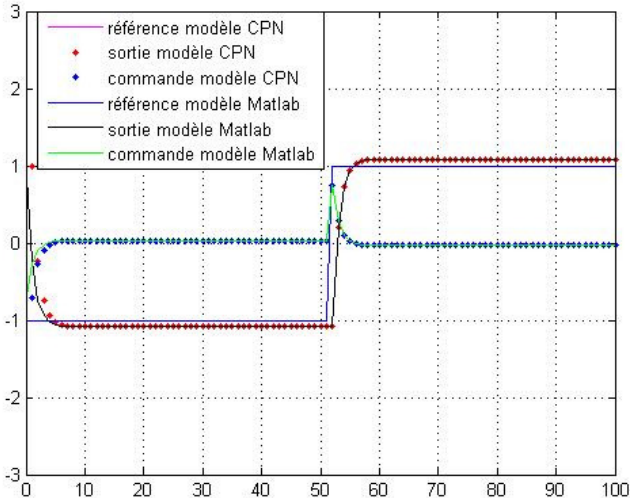


Figure 12. Comparaison des résultats du modèle RDPCTH obtenus sous simulation de CPNTools et du système sous Matlab.

3. 2 Système contrôlé en réseau

Le système est maintenant simulé en considérant un réseau passif (sans ordonnanceur) et supportant une faible charge (0.3). Pour vérifier le bon fonctionnement du système, le délai de bout-en bout des paquets transportant l'information de commande est recueilli et une moyenne est calculée. Ce délai moyen est injecté à l'entrée du système commandé de la figure 11 afin de comparer les résultats de simulation du modèle Matlab et les résultats obtenus avec notre modèle RDPCTH.

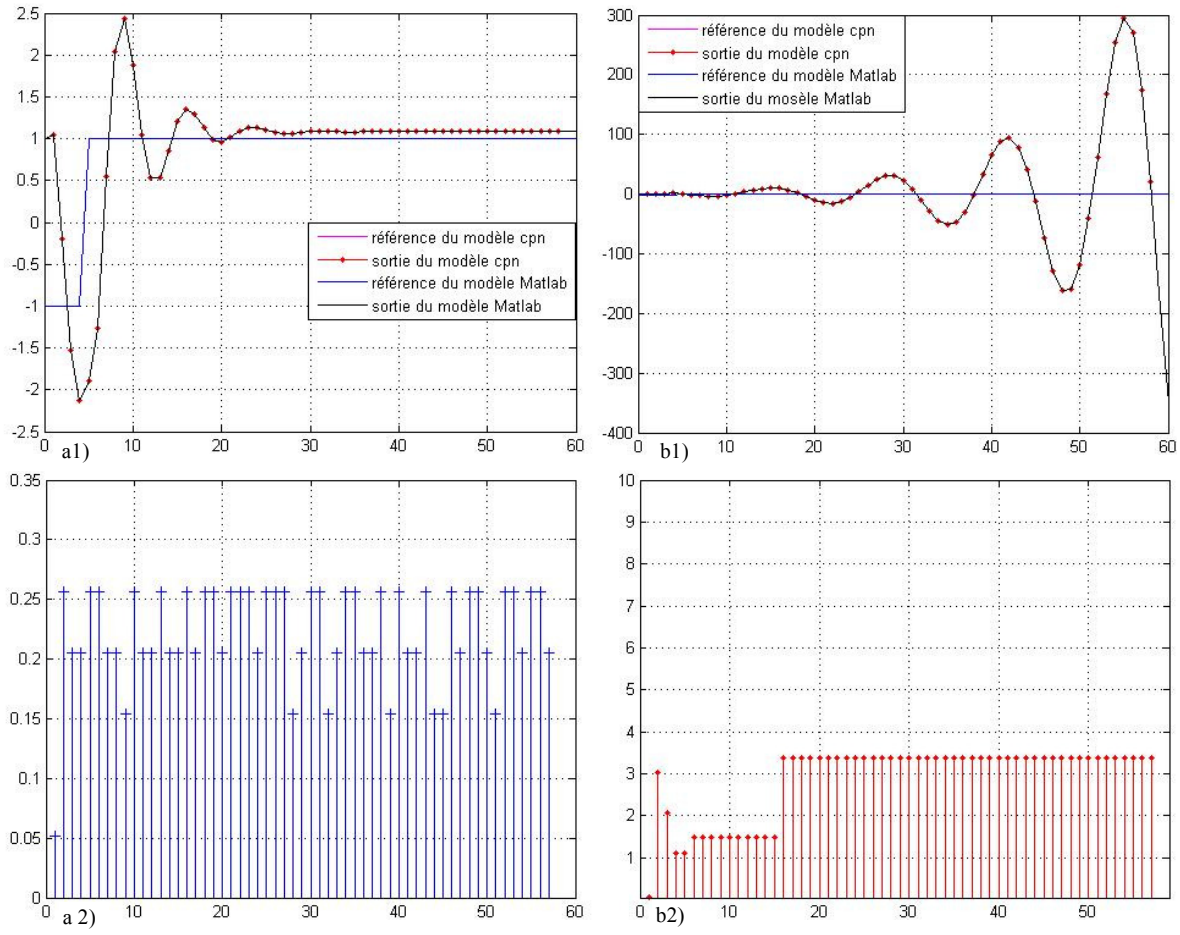


Figure 13. Sortie (a1, b1) et délai (a2, b2) du modèle RDPCTH et modèle Matlab, pour une charge réseau de 0.3 (a) et une charge de 0.7 (b).

Le délai moyen de bout en bout obtenu par la simulation du modèle RDPCTH pour une faible charge (0.3) est de l'ordre de 0.2215 ms (figure 13. a2). Ce délai est utilisé dans le modèle Matlab. Le système est stable (figure 13. a1) et ses valeurs propres traduisant son indice de performance ne dépassent pas le cercle unité ($0 < \lambda 1 < 0.2237$). Les mêmes opérations sont effectuées dans le cas où le réseau est chargé à 0.7 le système devient instable (figure 13. b1) et le délai moyen induit (figure 13. b2) est de l'ordre de 4.71 ms. La figure 13 montre que les résultats obtenus par simulation avec notre modèle RDPCTH sont conformes aux résultats générés sous Matlab.

Remarque
Le fait que la comparaison du modèle RDPCTH soit réalisée par rapport au modèle Matlab et non par rapport au modèle TrueTime, est dû à l'architecture interne du commutateur Ethernet dans TrueTime qui ne correspond pas à celle du réseau Ethernet que nous avons modélisé.

4. Conclusion :

Dans les sections précédentes, nous avons proposé un modèle RDPCTH d'un SCR. Ses performances sont évaluées en confrontant les résultats de simulation du modèle RDPCTH aux résultats obtenus avec Matlab. Cette comparaison est intéressante puisque Matlab fournit des résultats cohérents avec ceux obtenus par le modèle RDPCTH.
La seconde phase de notre étude va donc maintenant consister à commander le réseau de façon à pouvoir maintenir une Qualité de Service suffisante pour l'application et ainsi éviter que le système soit instable (cf figure 13 avec une charge de 0,7).
Pour cela, nous avons besoin de trouver un indice de performances qui permettra de :

- contrôler les poids de l'ordonnanceur de trames Weighted Round Robin,
- ou de commuter d'un type d'ordonnancement (WRR) à un autre.

De plus, cet indice de performances doit prendre en compte la performance du système commandé (Qualité de Contrôle). La méthode de calcul de cet indice de performance est exposée dans la partie suivante.

5. Indice de performances : Méthode de calcul

Dans le chapitre III, les résultats de simulation du modèle montrent que le délai moyen de bout-en-bout fourni par le réseau dépend des mécanismes d'ordonnancement de trames et de la charge du noeud. Pour que le réseau fournisse une performance optimale de la qualité de service, nous avons proposé d'abord :

- d'agir sur les poids de l'ordonnanceur WRR, dès que le délai devient trop important.
- Et de commuter de l'ordonnanceur WRR vers l'ordonnanceur à PS, si l'action sur le WRR ne fait pas baisser suffisamment le délai prioritaire.

Le problème est donc de savoir à quel moment on doit agir sur le réseau pour réduire le délai. L'idée principale de ce travail est donc de mettre au point une méthode qui calcule un temps appelé temps réel de décision (TRD) d'un système contrôlé en réseau. Ce temps réel de décision est considéré comme un seuil à partir duquel les conditions de stabilité du système ne sont plus respectées et implique alors qu'il faut commander le réseau pour redescendre en dessous de ce seuil.
Cette méthode consiste à évaluer la stabilité du système contrôlé en réseau, en calculant ses valeurs propres. Des travaux utilisant le concept de seuil pour réduire le retard que peut induire un réseau ont déjà été menés. Cependant, les méthodes pour le réduire diffèrent. Nous citons par exemple les travaux de [Lelevé 2000] qui a utilisé le délai maximum induit par le réseau obtenu par une phase d'audit du réseau afin de compenser les variations du

retard. Cette compensation est réalisée en utilisant des buffers. D'autres travaux [Zhang et *al.* 2001, Zhang 2000] parlent de seuils obtenus en utilisant les régions de stabilité du système contrôlé en réseau. Cette méthode permet d'établir une relation entre le délai induit par le réseau et la période d'échantillonnage du système commandé et d'évaluer les seuils de stabilité minimum et maximum du SCR. Cependant aucune mesure n'est prise pour réduire le délai.
Notre méthode se distingue des autres dans le sens où en plus de prendre en compte la stabilité du système (représenté par le TDR), le délai induit par le réseau est considéré. C'est la connaissance de l'ensemble qui permet d'agir sur le réseau pour maintenir la stabilité du système. La méthode s'inspire des travaux de Kim et Shin [Kim et Shin 92]. Les auteurs ont établi une méthode qui permet de calculer un 'temps critique appelé dur' d'un système de commande en boucle fermée temps réel. Les auteurs considèrent le retard généré par le temps de calcul de la commande d'un système de commande en boucle fermée. Ce retard peut être dû à une défaillance du calculateur. Cette défaillance empêche alors la non mise à jour de la valeur de commande durant une ou plusieurs périodes d'échantillonnage. Si ce retard dépasse une certaine limite appelée 'temps critique dur' cela implique que : - soit les conditions de stabilité du système sont violées ou - soit le système a quitté son espace d'état 'permis'.

5. 1 Hypothèse et méthode

Avant d'exposer la méthode de calcul du temps réel de décision, les hypothèses suivantes sont posées:

1. La commande du système est conçue sans prendre en compte les contraintes du réseau. Cela signifie que si la commande est directement connectée au système commandé, le système reste asymptotiquement stable.
2. La commande, l'actionneur, le capteur et le système commandé ne sont pas défaillants.
3. Les informations de la commande et du capteur sont transmises dans des paquets différents,
4. Quel que soit le type d'ordonnanceur de tâches choisi dans le nœud de contrôle, la tâche de commande est prioritaire, ce qui ne génère pas de retard d'ordonnancement de tâches.
5. Le retard induit par le réseau peut être inférieur à une période d'échantillonnage ou supérieur à une ou plusieurs périodes d'échantillonnage ou infini (*i.e.* information perdue). Si on faisait une abstraction du comportement du retard dans un réseau, on peut supposer qu'au final le message est reçu ou non. Ce comportement peut être modélisé par une loi binomiale.

5. 2 Représentation du retard induit par le réseau sur le système de commande en boucle fermée

Soit le système contrôlé en réseau suivant :

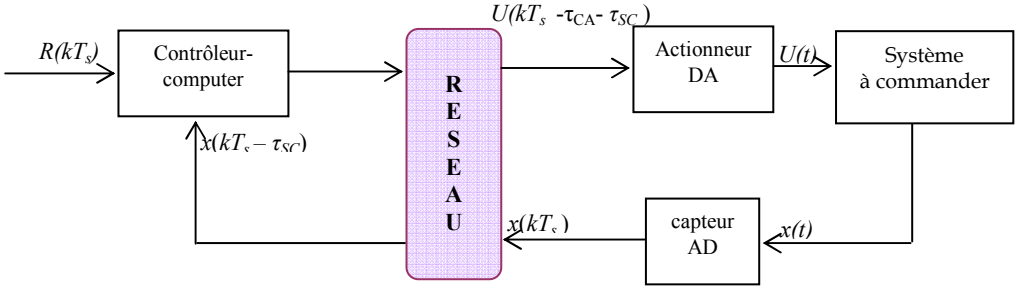


Figure 14. Modèle d'un système contrôlé en réseau et représentation des délais.

Dans le système de la figure 14, la commande est calculée à chaque intervalle d'échantillonnage et envoyée au système commandé distant via un réseau de communication. Le système commandé est un système linéaire à temps invariant décrit par l'équation suivante :

$$x(k+1) = Ax(k) + Bu(k) \tag{1}$$

Où k est l'indice de temps. Une unité de temps représente la période d'échantillonnage T_s, *et* $x \in \Re^n$ et $u \in \Re^l$ sont respectivement, les vecteurs d'état et d'entrée du système.
Les coefficients de la matrice, $A \in \Re^{n\times n}$ et $B \in \Re^{n\times l}$, sont obtenus à partir du modèle à temps continu :

$$A = e^{AcTs}, B = \int_0^{Ts} e^{Ac(Ts-S)} ds B_c \tag{2}$$

Où A_c et B_c sont les matrices du modèle continu. Le contrôleur digital lit les valeurs de mesures du système de sortie, les compare aux valeurs désirées, et calcule l'entrée de commande à chaque intervalle de temps suivant la loi de commande choisie.
La commande est envoyée vers l'actionneur à travers un réseau de communication une fois que celle-ci est encapsulée dans une trame. L'information contenue dans cette trame est reçue par l'actionneur. Elle est maintenue constante durant chaque intervalle d'échantillonnage par un bloqueur d'ordre Zéro (BOZ). Ensuite, cette valeur est appliquée à l'entrée de commande du process (voir figure 2).

L'équation (1) doit inclure les retards dus aux temps de conversion (A/D et D/A), de calcul de la commande (temps d'exécution du programme du contrôleur) τ_c, le temps de propagation à travers le réseau τ_p et le temps de mesure τ_s. La somme de ces délais est inférieure à la période d'échantillonnage T_s, en l'absence de congestion ou de rafale qui peut causer une congestion[38] du réseau ajoutant ainsi un délai supplémentaire. Comme il a été déjà expliqué dans le chapitre II, il y a deux sources de délai : le délai du capteur au contrôleur et le délai du contrôleur à l'actionneur.
$\tau_{sc} = \tau_s + \tau_p$, $\tau_{ca} = \tau_c + \tau_p + \tau_a$.
Ces deux délais sont additionnés pour faciliter l'analyse du système [Lian 2000 et Zhang 2001] :
$\Delta = \tau_{sc} + \tau_{ca}$.

En présence de congestion dans le réseau, ce temps Δ peut être soit inférieur à une période d'échantillonnage, soit supérieur à une ou plusieurs périodes d'échantillonnage.

5. 2. 1 Cas où le délai est inférieur à une période d'échantillonnage

Dans le cas où le délai Δ est inférieur à la période d'échantillonnage T_s l'équation (1) s'écrit comme suit :

$$x(k+1) = Ax(k) + B_1 u(k) + B_2 u(k-1) \tag{3}$$

Où :

[38] Cette congestion peut être dû à une défaillance d'un élément réseau (commutateur par exemple) qui réoriente le flux vers le commutateur qui s'occupe de la transmission des informations du système commandé.

$$B_1 = \int_0^{T_s-\Delta} e^{Ac(S)} ds B_c \tag{4}$$

Et,

$$B_2 = \int_{T_s-\Delta}^{Ts} e^{Ac(S)} ds B_c \tag{5}$$

5. 2. 2 Cas où le délai est supérieur à une période d'échantillonnage

Dans le cas où le retard est supérieur à *Ts*, on suppose que la valeur de commande n'est pas transmise à l'actionneur durant '*n*' intervalles de temps à partir de l'instant k_0. La commande appliquée à l'entrée du système commandé sera donc maintenue à $u(k_0)$ par un convertisseur D/A et un BOZ (circuit latch).
Le retard induit se produit d'une manière aléatoire, à cause de la nature du réseau utilisé. Par conséquent, ce retard est considéré comme une perturbation stochastique du système commandé. Donc, le système peut être représenté par un modèle dépendant des caractéristiques stochastiques de ce retard.
Par ailleurs, Le délai induit par le réseau dégrade les performances du système commandé et peut mener à une instabilité si le délai excède un certain seuil. L'instabilité se produit quand les conditions de stabilité du système ne sont plus respectées. Comme l'environnement est supposé stochastique stationnaire, l'occurrence du retard des messages peut être représentée par une fonction de distribution de probabilité.
Le délai induit par le réseau change la position des pôles du système contrôlé[39]. Par conséquent, les conditions nécessaires de stabilité peuvent être obtenues.
Soient XA et UA l'espace d'état défini et l'espace d'entrée admissible, respectivement. Supposons que l'état a évolué en présence d'un délai N induit par le réseau, suivant :

$$x(k) = \phi(k, k_0, x(k_0), u(k-N)) \tag{6}$$

Où ϕ est la fonction de transition d'état. Donc le temps réel de décision d'une tâche de contrôle commençant à k_0, peut être représenté par :

$$D(x(k_0)) = \sup_{u(k-N)\in U_A} \{N : \phi(k, k_0, x(k_0), u(k-N)) \in X_A\} \tag{7}$$

L'équation (7) représente le temps réel de décision de cette tâche commençant à l'instant k_0. Il est défini comme le délai maximum que le système commandé peut tolérer à cet instant.

[39]Cela a été démontré dans le travail de [Lian 2000].

5. 3 Calcul du temps réel de décision

Le temps réel de décision est calculé pour un système commandé représenté par (1). Les conditions de stabilité du système peuvent être utilisées pour obtenir le temps réel de décision.
Soient NT_s et DT_s le délai maximum et le délai maximum courant, respectivement. Le temps réel de décision est obtenu par des tests itératifs des conditions nécessaires de stabilité pour N variant de 1 à D.

5. 3. 1 Les effets de l'occurrence stationnaire des délais sur la stabilité du système

Dans cette section, nous voulons obtenir les conditions nécessaires sous lesquelles le système restera stable même en présence de retards aléatoires.
Ces conditions exigent la connaissance de l'équation dynamique du système et de l'algorithme de contrôle utilisé pour commander le système.
Considérons maintenant un système commandé, représenté par une équation différentielle linéaire à temps invariant, qui peut être convertie en une équation à temps discret en utilisant (2) et un indice de performance quadratique :

$$J = \frac{1}{2}\left(\sum_{k=0}^{k_f-1} \left[x^T(k) Q x(k) + u^T(k) R u(k) \right] + x^T(k_f) Q x(k_f) \right) \tag{9}$$

Où :
$Q \in \Re^{n \times n}$ est une matrice semi-définie positive,
$R \in \Re^{l \times l}$ est une matrice définie positive. Ces matrices sont déterminées par l'objectif de commande.
La synthèse linéaire quadratique dénommée LQ ou LQR (Linear Quadratique Regulator) consiste en la recherche d'une matrice de gain F telle que la commande par retour d'état $u(t) = -Fx(t)$ stabilise le système et minimise le critère quadratique J.
Cette matrice de gain F est obtenue en résolvant l'équation de Riccati discrète [Brogan 1985].
Supposons que l'entrée de commande a été émise à $t=mNT_s$. Si la valeur de la commande du système n'a pas été mise à jour pour les i périodes d'échantillonnage à cause d'un retard induit par le réseau, où $0 \le i \le N$, les équations d'état correspondantes pour le groupe d'intervalles durant lequel la commande n'est pas mise à jour, sont définies comme suit :

$$\begin{aligned}
x(mN+1) &= Ax(mN) + Bu(mN) \\
x(mN+2) &= Ax(mN+1) + Bu(mN) \\
x(mN+2) &= A^2 x(mN) + (A+I)Bu(mN) \\
&\vdots \\
x(mN+i) &= A^i x(mN) + \left(\sum_{j=0}^{i-1} A^j B \right) u(mN) \\
x(mN+i+1) &= A^{i+1} x(mN) + \left(\sum_{j=1}^{i} A^j B \right) u(mN) + Bu(mN+i) \\
&\vdots \\
x((m+1)N) &= A^N x(mN) + \left(\sum_{j=N-i}^{N-1} A^j B \right) u(mN) + \sum_{j=0}^{N-i-1} A^j B u(mN+N-j-1)
\end{aligned}$$

Où m est l'indice de temps pour les groupes de chaque intervalle d'échantillonnage N. Soit $X(m) = [x_1, x_2, \ldots, x_N]^T \equiv [x(mN+1), x(mN+2), \ldots, x((m+1)N)]^T$ et
$U(m) = [u_1, u_2, \ldots, u_N]^T \equiv [u(mN+1), u(mN+2), \ldots, u((m+1)N)]^T$
$X(m)$ et $U(m)$ sont respectivement les vecteurs d'état et de commande augmentés à l'intervalle de temps durant lequel la commande n'a pas été mise à jour ou transmise.
Quand le délai est égal à i périodes d'échantillonnage, il en résulte l'équation d'état augmentée suivante :

$$X(m+1) = A_D X(m) + B_{Di}^1 U(m) + B_{Di}^2 U(m+1) \tag{10}$$

$$U(m) = -F_D X(m) \tag{11}$$

Où :

$$A_D = \begin{bmatrix} 0 & \cdots & 0 & A \\ 0 & \cdots & 0 & A^2 \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & A^N \end{bmatrix},\ F_D = \begin{bmatrix} F & 0 & \cdots & 0 \\ 0 & F & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \ldots & F \end{bmatrix},\ B_{Di}^1 = \begin{bmatrix} 0 & \ldots & 0 & B \\ 0 & \ldots & 0 & (AB+B) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & \sum_{j=0}^{i-1} A^j B \\ 0 & \ldots & 0 & \sum_{j=1}^{i} A^j B \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & \sum_{j=N-i}^{N-1} A^j B \end{bmatrix} \text{ et}$$

$$B_{Di}^2 = \begin{bmatrix} 0 & & \ldots & & 0 \\ \vdots & \ddots & \ldots & & \vdots \\ 0 & \ldots & 0 & \ldots & 0 \\ 0 & \ldots & B & \ldots & 0 \\ 0 & \ldots & AB & \ldots & 0 \\ \vdots & \ldots & \vdots & & \vdots \\ 0 & \ldots & A^{N-i-1}B & \ldots & B \quad 0 \end{bmatrix}^N$$

Le retard induit par le réseau suit une distribution binomiale (hypothèse 6), avec un paramètre P. En effet, pour calculer le temps critique de décision du système on a besoin de n'avoir que deux cas de figures, à savoir : soit le message est transmis, soit il ne l'est pas.
Soient $q_0, q_1, \ldots, q_N$ les probabilités associées aux délais $0, T_S, \ldots, NT_S$, respectivement, tel que : $\sum_{i=1}^{N} q_i = P$. Le délai maximum est supposé égal à NT_S.
En combinant l'équation d'état (10) avec la distribution du délai, l'équation d'état incluant les effets des délais $\leq NT_S$ devient :

$$X(m+1) = A_D X(m) + \sum_{i=0}^{N} \xi_i \left(B_{Di}^1 U(m) + B_{Di}^2 U(m+1) \right) \tag{12}$$

Où $\xi_i \in \{0,1\}$ est une variable aléatoire distribuée suivant une loi binomiale avec les paramètres q_i, *i.e.* $\Pr[\xi_i = 1] = q_i$, ainsi le premier moment est égal à:

$$\overline{X}(m+1) = A_D \overline{X}(m) + \sum_{i=0}^{N} q_i \left(B_{Di}^1 U(m) + B_{Di}^2 U(m+1) \right) \tag{13}$$

Puisque la période de l'indice *m* est *N*, la variable complexe de la transformée en *Z* de (10), correspond à Z_k^N, où Z_k est la variable complexe de la transformée en *Z* des équations avec l'indice *k*. En utilisant (3) et (13), l'équation caractéristique du système de commande en présence d'occurrence stationnaire des retards, est représentée par :

$$\det\left[\left(I + \sum_{i=0}^{N} q_i B_{Di}^2 F_D \right) z^N - A_D + \sum_{i=0}^{N} q_i B_{Di}^1 F_D \right] = 0 \tag{14}$$

La stabilité asymptotique peut être testée en étudiant 'la position des pôles' à partir de l'équation (14). L'équation caractéristique (14) est sous une forme simple due à la structure simple de A_D malgré sa dimension augmentée.

Dans le cas où il n'y a pas de retard induit par le réseau (i.e. $q_0 = 1$ et $P = \sum_{i=1}^{N} q_i = 0$). L'équation caractéristique est :

$$\det\left[z^N I - (A - BF)^N \right] = 0 \tag{15}$$

Dans le cas où le retard est maximal (i.e. le pire cas) dans lequel $q_N = 1$, la commande est mise à jour seulement après *N* intervalles de temps NT_S. L'équation caractéristique est, donc, sous la forme suivante :

$$\det\left[z^N I - A^N + \sum_{i=0}^{N-1} A^i BF \right] = \det\left[z^N I - \sum_{i=0}^{N-1} A^i (A - BF) + \sum_{i=1}^{N-1} A^i \right] = 0 \tag{16}$$

Où :

$A = QDQ^{-1}$ par transformation de similarité si $D = diag[d_1, \ldots, d_n]$ et d_i sont les valeurs propres de *A*.

Si le système représenté par la matrice de transition *A* sans boucle de régulation (sans retour) est instable, c'est-à-dire qu'il existe au moins une valeur propre $|d_j| \geq 1$ [Brogan 1985], la boucle de retour est utilisée pour stabiliser le système en changeant la matrice de transition en $A - BF$, qui peut aussi être diagonalisée par la transformation de similarité R, i.e., $A - BF = R\Lambda R^{-1}$, où $\Lambda = diag[\lambda_1, \ldots, \lambda_n]$ et λ_i sont les valeurs propres de $A - BF$.

Par exemple, si N=2 :

$$Eig\left[A^2 - (A + I)BF \right] = Eig\left[(A + I)(A - BF) - A \right] = Eig\left[QD_1 Q^{-1} R\Lambda R^{-1} - QDQ^{-1} \right] \tag{17}$$

Où :

$D_1 = diag[d_1 + 1, \ldots, d_n + 1]$, le $ij^{ème}$ élément de la matrice $T = A^2 - (A + I)BF$

Si le délai n'est pas un entier multiple de la période d'échantillonnage, i.e., $\Delta = T_d = (N-1)T_S + \delta$, $0 < \delta < T_S$, toutes les procédures restent les mêmes que pour le délai maximum supposé NT_s sauf pour *i*= *N* :

$$x((m+1)N) = A^N x(mN) + \left(\sum_{i=1}^{N-1} A^j B + B_1 \right) u(mN) + B_2 u((m+1)N-1) \tag{18}$$

Où :

$B_1 = \int_0^{T_s-\delta} e^{Ac(S)} ds B_c$, $B_2 = \int_{T_s-\delta}^{Ts} e^{Ac(S)} ds B_c$. $B_{D_N}^1$ et $B_{D_N}^2$ sont décrits comme suit :

$$B_{D_N}^1 = \begin{bmatrix} 0 & \cdots & 0 & B \\ 0 & \cdots & 0 & AB+B \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \sum_{j=0}^{N-2} A^j B \\ 0 & \cdots & 0 & \sum_{j=1}^{N-1} A^j B + B_1 \end{bmatrix}, \quad B_{D_N}^2 = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & B_2 & 0 \end{bmatrix}.$$

5. 3 .2 Exemple illustratif

Considérons un simple système commandé décrit par l'équation aux différences suivante :

$$x(k+1) = 1.05x(k) + 1.8u(k) \tag{19}$$

Où $Q = 2$, $R = 7$. Ce système est instable sans aucun retour de commande mais le système est commandable.

La commande optimale est donnée par :

$u(k) = -Fx(k)$

La matrice de gain[40], $F = 0.3594$, la commande par retour d'état $u(k) = -Fx(k)$ stabilise le système et minimise le critère quadratique J (équation 9). Puisque la position du pôle λ a changé de *1.05* à *0.4031*.

La relation entre la position des pôles et N pour le pire cas $P=q_N=1$ (c'est-à-dire dans le cas où le délai est maximal (*c.f.* section 5. 3. 1, page 125)) est déterminé par l'équation suivante :

$$Eig\left[A^N - \sum_{i=0}^{N-1} A^i BF \right] = 0 \tag{20}$$

En appliquant l'équation 19, on obtient les résultats résumés dans le tableau 1 :

N	Equation	$\lvert\lambda\rvert$
2	$Eig(A^2 - ABF - BF)$	0.2237
3	$Eig(A^3 - A^2BF - ABF - BF)$	0.8818
4	$Eig(A^4 - A^3BF - A^2BF - ABF - BF)$	1.5728

Tableau 1**.** Relation entre la position du pôle et N quand $P = q_N=1$

[40] Les résultats sont obtenus avec le logiciel Matlab.

D'après le tableau 1. le temps réel de décision du système (19) est $D = 3T_s$, puisqu'à $N = 4$ la valeur du pôle est à l'extérieur du cercle unité.
Le système est simulé avec le logiciel Matlab pour 100 pas d'échantillonnage avec une régulation autour d'une consigne non nulle (figure 11).

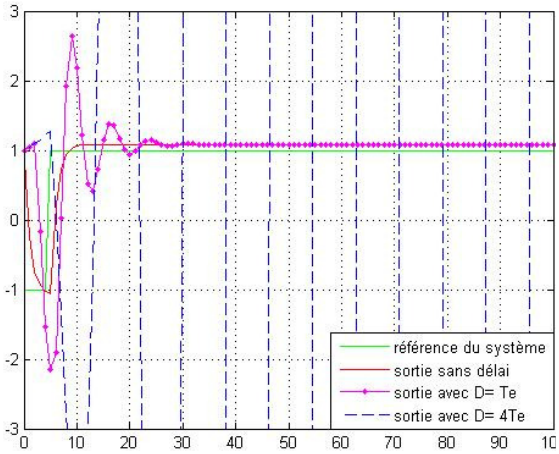


Figure 15. Résultats de simulation du système pour différents délais : D= 0, D = T_e et D= $4T_e$.

6. Conclusion

Les résultats de simulation obtenus montrent d'abord que le retard induit par le réseau influe sur la stabilité d'un système contrôlé en réseau.
La connaissance d'un seuil que nous avons appelé 'temps réel de décision' du système est importante, pour évaluer le seuil de tolérance d'un système contrôlé en réseau, par rapport, au retard induit par le réseau.
L'utilisation de ce temps réel de décision permettra d'agir sur le réseau afin d'augmenter la qualité de service fournie par le réseau.
L'objectif de la section suivante est, donc, d'intégrer la méthode de calcul du seuil de décision dans l'utilisation optimale des mécanismes d'ordonnancement de trames (vus au chapitre III). Cela suggère, en premier lieu, d'utiliser le même outil de modélisation. Dans notre cas, nous avons choisi d'utiliser les RDPCHT. Cet outil offre de nombreuses possibilités pour modéliser les systèmes de commande, en ajoutant notamment, des actions aux transitions en forme de segment de code. Aussi, le fait que le commutateur Ethernet soit déjà réalisé avec les RDPCHT, les ajouter consiste à étendre le modèle existant (Ceci a été réalisé dans la première partie). Donc, un autre modèle va être ajouté pour étendre le modèle RDPCHT du SCR. Ce modèle a pour objectif de comparer le délai mesuré induit par le réseau au temps réel de décision. Celui-ci permet d'agir sur le mécanisme d'ordonnancement de trames du commutateur Ethernet.

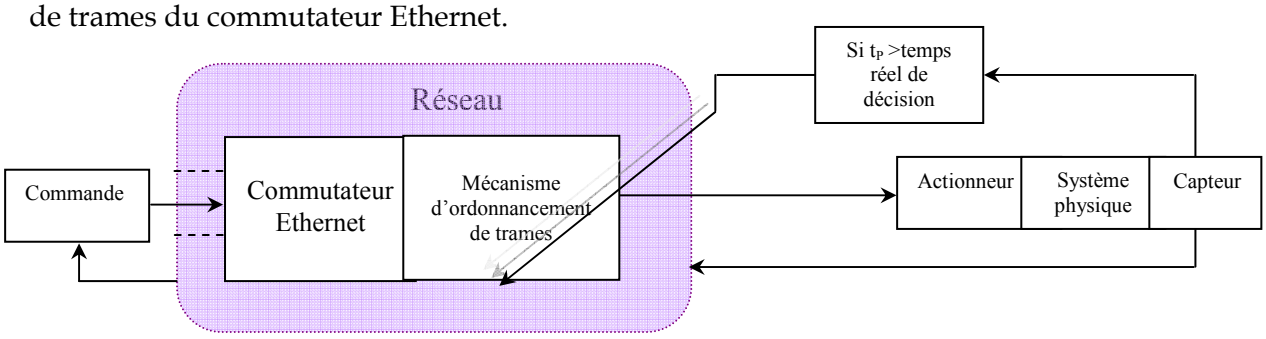


Figure 16. Modélisation intégrée et méthode d'action sur le mécanisme d'ordonnancement de trames

L'évaluation de la méthode proposée, est effectuée en simulant le modèle du système commandé en réseau dans sa globalité avec le logiciel CPNTools.

7. Intégration de l'indice de performances au modèle RDPCTH du SCR

Dans cette section le calcul du délai modélisé à la section 2. 1. 7 page 115 est utilisée afin de comparer sa valeur au seuil de prise de décision TRD calculé dans section 5. 3. 1 page 124. Cette opération permet d'agir sur le commutateur afin de réduire le délai.

7. 1 Le seuil de prise de décision

Le dispositif utilisant le seuil de prise décision qui représente l'indice de performances de notre système, est illustré dans la figure 17. Ce dispositif est placé au niveau du module buffers de sortie (FIFO_OUT) du commutateur. La prise de décision permet d'agir sur les poids du mécanisme WRR pour le scénario 1, ou de passer d'un mécanisme d'ordonnancement à un autre pour le scenario 2. Par conséquent, il doit être capable :
Scenario 1 :

- de changer les poids du WRR qui permettent la transmission de paquets suivant le pourcentage de la bande passante qui leur est attribuée. Pour ce faire nous avons choisi de modéliser deux mécanismes WRR l'un avec les poids w_1, w_2 et w_3 qui correspondent à la transmission des paquets HP, MP et BP et portent les valeurs suivantes : 10%, 45%, 45% respectivement. L'autre WRR portent les poids suivants 99%, 0,99% et 0.01%. Le délai calculé et comparé au TRD, permet de cette façon d'agir sur les poids[41] en passant du premier WRR au second WRR.

Scenario 2 :

- de bloquer le passage des paquets vers le premier mécanisme d'ordonnancement (WRR) qui n'est plus efficace,
- d'autoriser leur passage vers le second ordonnanceur (PS), et vice-versa.

Le délai calculé est transmis au commutateur par la place port *Delai*, cette valeur est comparée au TDR préalablement calculé. Des conditions sont associées aux arcs entrants des places *place 1* et *place 2*, exemple :

```
 if p < TDR then List.drop(le1, length(le1))^^[e1]
else  List.drop(le1, length(le1))^^[]
```

Elle est traduite comme suit : Dans le cas où le seuil de décision n'est pas atteint la *place1* est marquée d'un élément [e1] et dans le cas contraire elle n'est pas marquée [].
Ces conditions permettent d'informer le module FIFO_Out si le délai est supérieur (jeton dans la place 2), ou inférieur à TDR (jeton dans la place 1).
Par ailleurs, dans cette modélisation, il est nécessaire de réaliser une exclusion mutuelle entre ces deux places. C'est dire que si la place *place 1* est marquée, la *place 2* ne doit pas l'être (et si la *place2* est marquée la *place 1* ne doit pas l'être). Ceci est également réalisé grâce aux mêmes conditions associées aux arcs. La validation d'une des deux conditions annule l'autre et vice versa.

Une fois marquée, une de ces places (*place 1* ou *place 2*) doit autoriser le passage des paquets des buffers au module *ordo* qui contient les deux mécanismes d'ordonnancement. Dans le cas où le seuil TRD n'est pas atteint, les buffers de sortie sont directement reliés au WRR

[41] C'est une façon comme une autre de modéliser l'action entreprise, sur le poids du WRR, d'autres façons de modélisation existent, nous avons choisis la plus simple.

autorisant le passage des paquets (figure 17). Le passage des paquets se fait sous deux conditions :

- la place 1 doit être marquée et,
- la place 2 ne doit pas l'être (arc avec [], traduit l'arc inhibiteur).

Lorsque le TDR est atteint, la place 2 est marquée. Elle bloque le passage des paquets vers WRR et autorise leur transfert vers soit l'autre WRR si on est dans le scénario 1, ou soit vers l'ordonnanceur PS si on est dans le scénario 2.

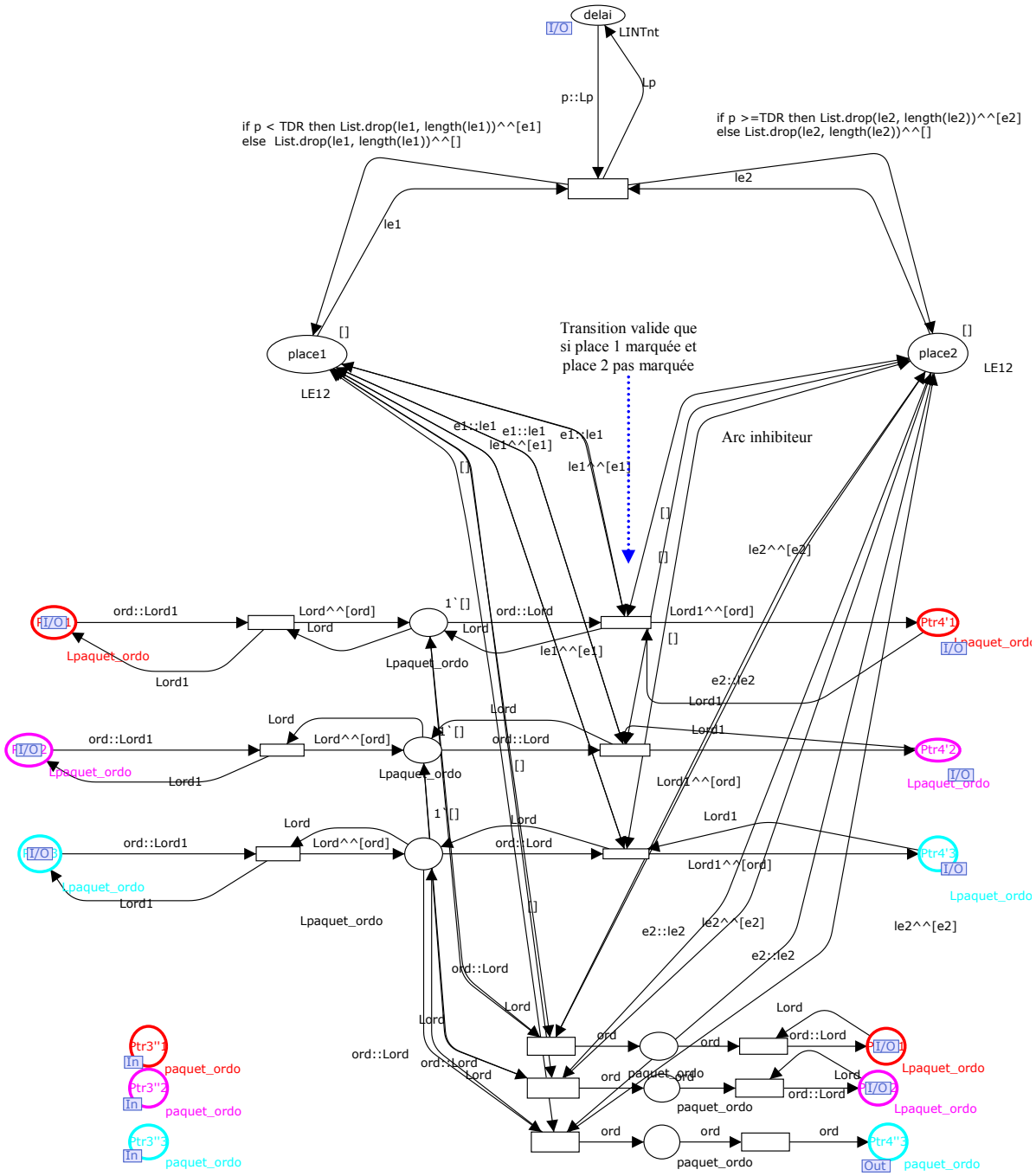


Figure 17 . Modèle RDPC du mécanisme de prise de décision.

7. 2 Evaluation de performances sur une étude de cas

Dans cette section l'exemple modélisé et simulé précédemment, est repris. Il a été simulé suivant les deux scenarii.
Dans le premier scenario le SCR est simulé pour les différents cas suivants :

1- le SCR est simulé dans le cas où WRR implanté dans le commutateur Ethernet possède les poids suivants 10%, 45% et 45% dans un environnement faiblement chargé 0.3.
2- le SCR est simulé avec les mêmes poids associés au WRR dans un environnement fortement chargé 0.7 sans que le mécanisme de prise de décision n'intervienne.
3- le SCR est simulé avec les mêmes poids associés au WRR dans un environnement fortement chargé 0.7 avec le mécanisme de prise de décision (changement de poids du WRR).

Dans le second scenario le SCR est simulé pour les différents cas suivants :

1- le SCR est simulé dans le cas où WRR implanté dans le commutateur Ethernet possède les poids suivants 60%, 30% et 10% dans un environnement faiblement chargé 0.3,
2- le SCR est simulé avec les mêmes poids associés au WRR dans un environnement fortement chargé 0.7 sans que le mécanisme de prise de décision n'intervienne.
3- le SCR est simulé avec les mêmes poids associés au WRR dans un environnement fortement chargé 0.7 avec le mécanisme de prise de décision (passage de PS à WRR).

7. 2. 1 Etude de cas

Reprenons l'exemple précédent :

$$x(k+1) = 1.05x(k) + 1.8u(k)$$

Où $Q = 2$, $R = 7$, la commande optimale $F = 0.3594$. Te = 1ms

Rappelons que le seuil de décision (TRD) est obtenu grâce à un calcul itératif des pôles du système pour différents N dans le cas ou q_N=1, et le TDR = 3Te.

N	Equation	$\lvert\lambda\rvert$
2	$Eig\left(A^2 - ABF - BF\right)$	0.2237
3	$Eig\left(A^3 - A^2BF - ABF - BF\right)$	0.8818
4	$Eig\left(A^4 - A^3BF - A^2BF - ABF - BF\right)$	1.5728

Tableau 2**.** Relation entre la position du pôle et N quand $P = q_N = 1$

7. 2. 1. 1 Scenario 1

L'exemple est modélisé et simulé, pour les différents cas évoqués plus haut. Le délai moyen induit par le réseau est utilisé dans le modèle Matlab (figure 11), afin de confronter les résultats obtenus sous CPNTools et ceux obtenus sous Matlab.

Environnement à faible charge

Le système est simulé et ses performances évaluées pour une charge de 0.3. L'objectif de cette simulation est d'observer le comportement du SCR avec notre modèle RDPCTH et celui de Matlab (figure 18).

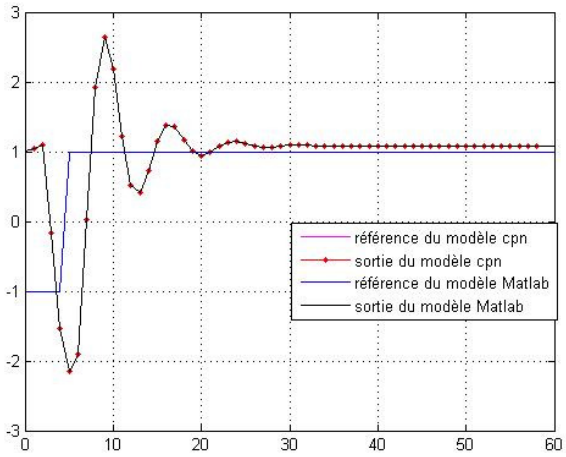


Figure 18. Sortie du modèle RDPCTH et modèle Matlab, pour une charge réseau de 0.3.

Le délai moyen est 0.2242 ms et les valeurs propres du système indique que ce délai reste tolérable ($0 < \lambda 2 < 0.2237$).

Environnement à forte charge

Dans le cas où le réseau est chargé à 0.7 le système est simulé sans que le dispositif de prise de décision n'intervienne.

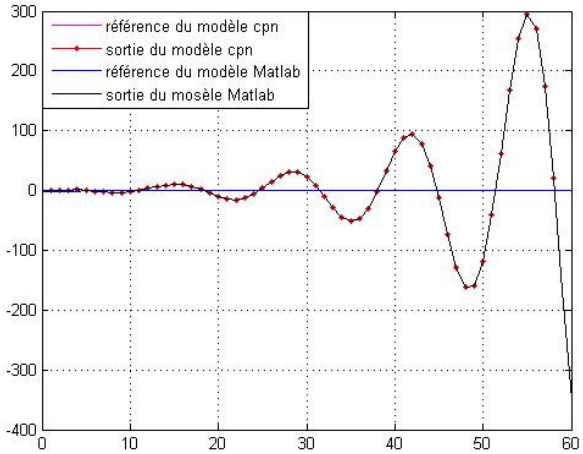


Figure 19. Sortie du modèle RDPCTH et modèle Matlab, pour une charge réseau de 0.7

D'après la figure 19, le système est instable et le délai moyen calculé est égal à 3,22 ms. Comme attendu, le délai augmente suivant la charge du réseau, et cela cause l'instabilité du système.

Environnement à forte charge avec mécanisme de prise de décision

Maintenant pour la même charge le système est simulé avec le procédé de prise de décision et le résultat obtenu est illustré dans la figure 20.

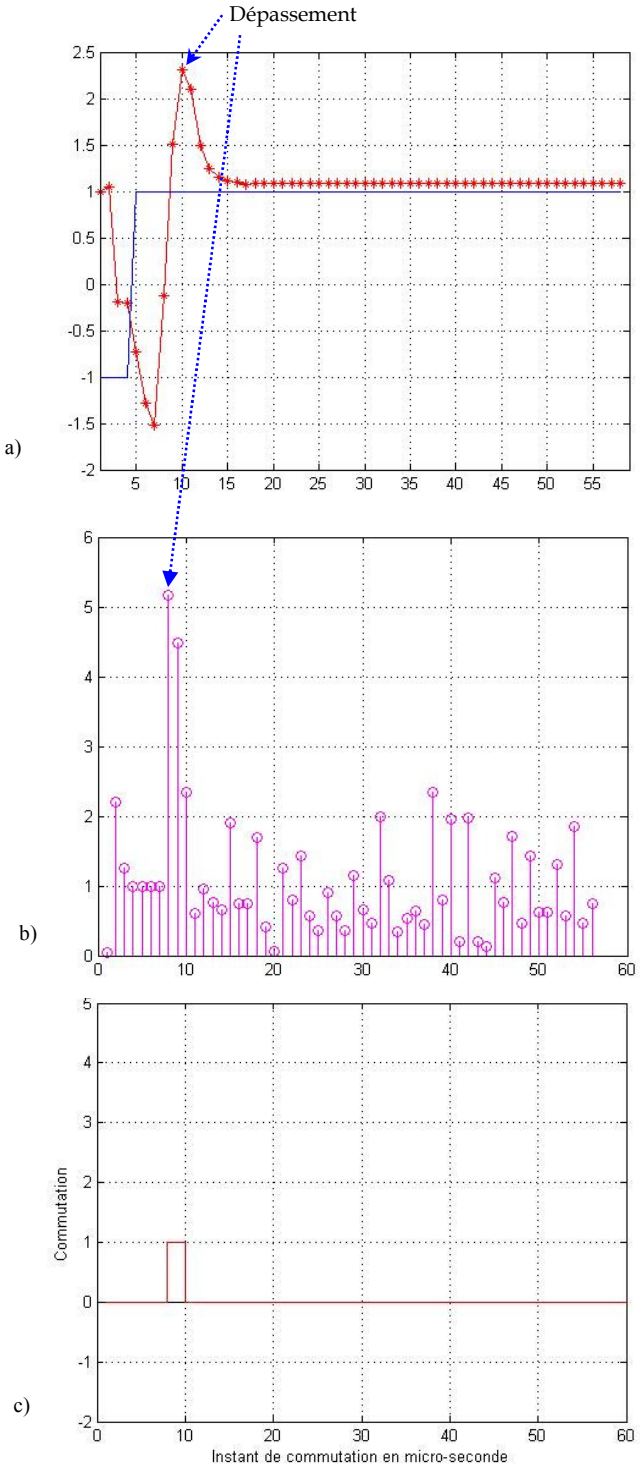


Figure 20. Sortie a), délai en (µs) (b) et instants de prise de décision (c) du modèle RDPCTH et modèle Matlab, pour une charge réseau de 0.7 avec le dispositif prise de décision.

D'après le résultat obtenu, le système montre certes un dépassement important (figure 20. a)dû à un retard (figure 20. b) de 5,1744 ms. Cependant le dispositif de prise de décision a commuté vers l'ordonnanceur WRR avec les poids 99%, 0.99% et 0.01%, cela se traduit par le passage de 0 à 1 traduisant la commutation dans la figure 20. c. Cela a pour effet de réduire le retard à 2,3344 ms jusqu'à atteindre 0,4608 ms. Le retard moyen après l'intervention du

dispositif de prise de décision est égal à 1,397 ms (passage du dispositif de prise de décision vers le poids initial). Donc la valeur propre du système est maintenue entre 0 et 0,2237.

7. 2. 1. 2 Scenario 2

Dans le cas où le système est simulé à faible charge et à forte charge, les résultats obtenus sont sensiblement les mêmes que dans le scénario 1 pour les faibles charges, sauf qu'à forte charge le délai est de l'ordre de 5.71 ms. Par conséquent, la valeur propre du système commandé est supérieure à 1.5728 ce qui traduit l'instabilité du système.

Environnement à forte charge avec mécanisme de prise de décision

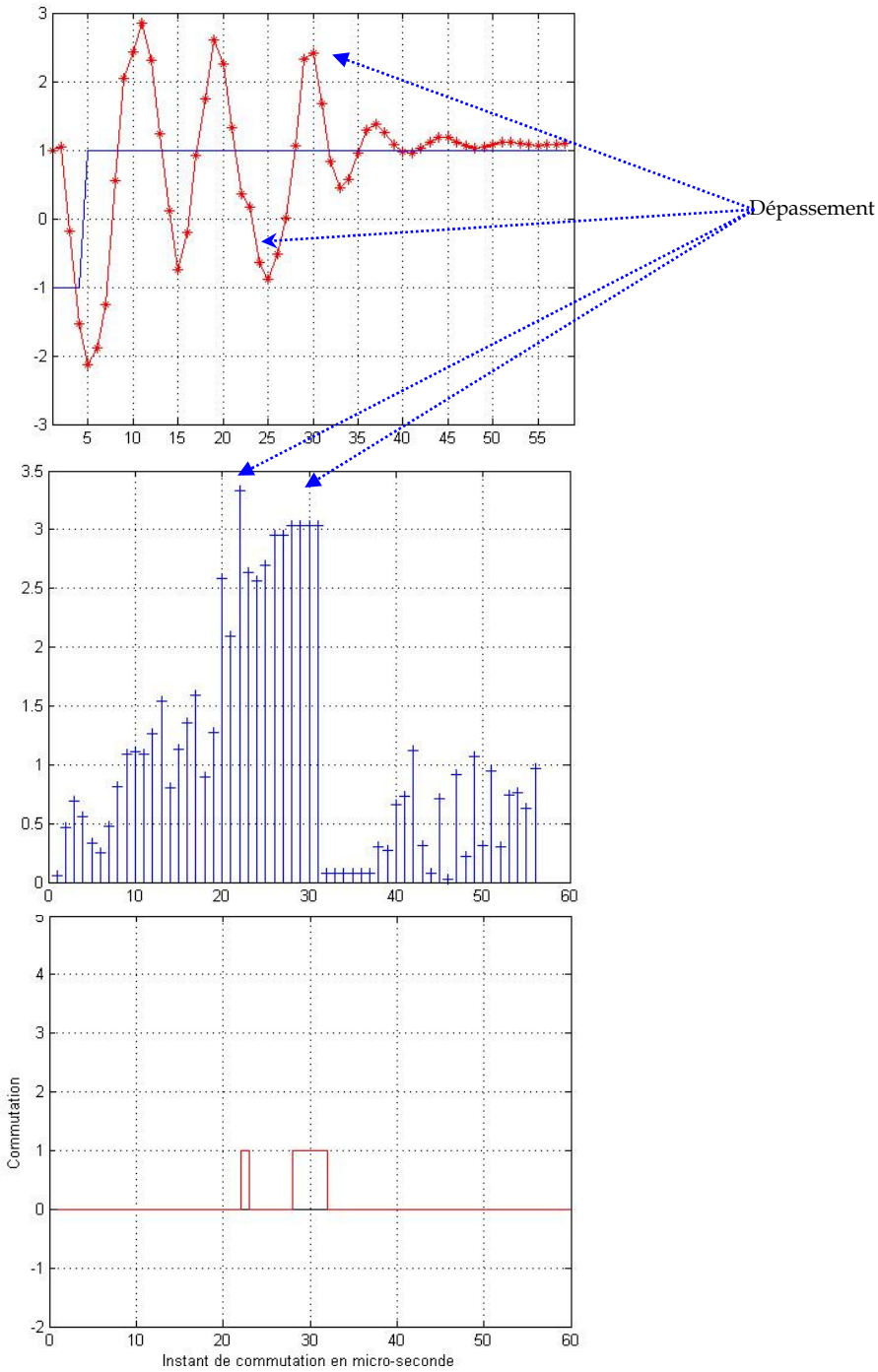


Figure 21. Sortie (a), délai (en µs) (b) et instants de prise de décision (c) du modèle RDPCTH et modèle Matlab, pour une charge réseau de 0.9 avec le dispositif prise de décision

Dans cette partie, nous avons exagéré la charge du nœud (0,9) dans le but de voir l'efficacité de notre méthode.
D'après la figure 21 .b), nous notons un délai de 3,328 ms qui a pour conséquence un dépassement important du signal de sortie (figure 21. a). Ce délai va donc, déclencher le dispositif de prise de décision, qui fera passer le commutateur du mécanisme d'ordonnancement WRR au mécanisme PS (premier passage de 0 à 1 dans le graphe 21. c). Ceci va réduire le délai à une valeur de 2,6352 ms. Cette réduction aura pour effet de faire revenir le commutateur à son mécanisme d'ordonnancement initial WRR avec les paramètres suivants : w_1= 60%, w_2= 30%, w_3=10% (premier passage de 1 à 0 dans le graphe 21.c). Ensuite, nous notons que le système revient à un délai important qui est de l'ordre de 3,0328 ms (Second passage de 0 à 1 dans le graphe 21. c). Cela est dû à la forte charge de l'environnement du nœud.
Donc, le réseau commute de nouveau sur le mécanisme d'ordonnancement PS. Cette prise de décision réduit le délai à 0,751 ms. La valeur propre du système passe à une valeur $0.818 < \lambda < 1.5728$ au 22ème pas d'échantillonnage et est maintenue entre 0 et 0,22 à partir du 31ème pas d'échantillonnage.
Cet exemple est intéressant car il met en évidence des problèmes d'instabilité sur la commande du réseau. Le passage entre les différentes configurations réseau est basé uniquement sur le délai, et a aussi pour conséquence de faire varier ce même délai. Cela montre que la stratégie pour commander le réseau doit reposer non seulement sur le délai mais aussi sur d'autres informations de communication, comme le taux d'utilisation du réseau, la quantité du trafic, l'état du process…de façon à éviter tout problème d'instabilité.
Cette remarque montre l'intérêt d'avoir des outils de supervision de SCRs efficaces pour avoir une vue globale de l'état du SCR avant toute prise de décision (c.f. chapitre II).

7. 2. 2 Avantages et limites

Les RDPC offrent la possibilité d'étendre le modèle du commutateur Ethernet pour traiter des paquets de différentes tailles. Aussi, d'autres solutions liées à la réduction du délai dans un SCR peuvent être incluses dans le modèle proposé sans modifications majeures du modèle existant.
Ce travail de modélisation permet d'observer le comportement détaillé d'un SCR implanté sur un commutateur Ethernet. Aussi, le modèle proposé représente un prototype qui est certes incomplet mais reste améliorable : l'outil de modélisation utilisé permet de le faire.
Les limites principales auxquelles nous nous sommes confrontés demeurent le temps de calcul lié à la simulation du modèle. Ce temps devient important en augmentant le nombre de commutateurs.
De plus, l'outil de simulation ne permet pas l'utilisation de nombres réels. Pour pallier ce manque, nous avons dû trouver une alternative à savoir :
- transformer les nombres réels en caractères,
- ensuite les utiliser dans les segments de code (transition *controller* et transition *plant*), les transformer en réel dans les segments de code puisqu'il possible de manipuler des réels dans le segment de code,
- faire l'opération de calcul,
- ensuite, transformer le résultat obtenu en caractère puis le transmettre.

8. Conclusion

Dans ce chapitre, un modèle RDPCTH d'un système contrôlé en réseau est proposé. L'évaluation de ses performances est réalisée en confrontant les résultats de simulation obtenus avec CPNTools aux résultats du même modèle construit avec Matlab. Ensuite une méthode basée sur un ordonnancement de trames (commande du réseau) dont le but est de réduire le délai induit par le réseau a été proposée. L'ensemble a été modélisé avec les RDPCTH et évalué par simulation du modèle avec le logiciel CPNTools.
Afin de modéliser le système contrôlé en réseau, nous avons procédé à une représentation modulaire, et hiérarchique, comme cela a été effectué pour le commutateur Ethernet dans le chapitre précédent. Les systèmes de commande modélisés sont les systèmes à temps discrets. Ensuite, une approche qui permet de calculer un seuil de tolérance du retard induit par le réseau, sur un système contrôlé a été établie dans la section 3. Ce seuil de tolérance est appelée 'temps critique de décision' du système et est utilisé comme critère pour agir sur le réseau afin d'augmenter sa qualité de service.

Conclusion Générale

Les travaux de recherche sur les SCRs sont relativement récents et la communauté scientifique commence juste à s'organiser autour cette nouvelle thématique de recherche. On peut citer notamment la création du TC 1.5 sur les « networked systems » lors du 16ème congrès World IFAC à Prague en 2005. De plus, l'étude des SCR est fortement appuyée par la Commission Européenne qui propose une action dédiée au SCR (Objective ICT 3.7 : Networked Embedded and Control Systems) lors de son dernier appel d'offre dans le contexte du 7ème PCRD.

L'étude des SCR, reposant sur des compétences en automatique, en informatique et en réseau propose naturellement des solutions propres à chaque domaine. On peut ainsi dégager les contributions en deux familles : l'approche « control over network » qui consiste à adapter la commande du process relativement aux performances de communication, et l'approche « control of network » dont le but est d'optimiser les performances du réseau en fonction des exigences applicatives.

Cependant, la conception évoluée d'un SCR passe par une coordination des actions de commande à la fois sur le réseau et sur le process en analysant le couple Qualité de Service offert par le réseau et Qualité de Contrôle exigée par l'application. Cette troisième approche dite « intégrée » ou appelée « co-design » nécessite des outils d'analyse dédiés au SCR permettant de modéliser globalement le SCR. C'est-à-dire, ces outils doivent regrouper la partie contrôle/process et la partie réseau. Des propositions ont été faites dans ce sens avec des outils comme TrueTime qui propose des blocs de comportements de communication dans l'environnement Simulink/Matlab. Ce simulateur est très intéressant pour les automaticiens qui souhaitent tester leurs algorithmes de contrôle en considérant le réseau. Mais, il ne permet pas réellement de faire de la commande de réseau.

L'objectif de nos travaux de thèse est donc de proposer un environnement de modélisation intégré permettant de représenter le comportement de SCRs. Nous avons choisi les Réseaux de Petri de haut niveau qui possède un fort pouvoir d'expression, de formalisation et dont la modularité permet de faire évoluer et compléter les modèles qui sont développés dans cette thèse.

Dans cette thèse, nous avons proposé un modèle Ethernet Commuté gérant des mécanismes d'ordonnancement. Le choix de ce réseau a été guidé par le fait qu'il est de plus en plus utilisé dans les SCRs. Cette approche pragmatique se retrouve dans la sélection des mécanismes d'ordonnancement. Le but n'était pas de modéliser des ordonnanceurs évolués mais de considérer des ordonnanceurs réellement implantés dans les commutateurs Ethernet du marché. Finalement, et contrairement à de nombreuses propositions faites dans l'approche « control of network », notre volonté a été d'éviter de développer de nouveaux protocoles de communication qui seraient mieux adaptés aux SCRs, mais dont l'issue commerciale serait aléatoire. L'un des objectifs de cette thèse a donc été de chercher à modéliser une architecture réseau normalisé et réellement utilisé dans les SCRs.

Enfin, un SCR a été modélisé par des Réseaux de Petri de haut de niveau, en intégrant au modèle Ethernet Commuté, l'environnement applicatif : Contrôleur, Process,.. Des scénarii identiques et modélisés par notre approche et sous Simulink/Matlab montrent que l'on obtient des résultats semblables. Ensuite, nous avons montré, des stratégies pour commander le réseau de façon à adapter sa Qualité de Service en regard de la Qualité de Contrôle requise par l'application. Pour cela, des ordonnanceurs à priorité stricte et de type WRR sont utilisés. Les résultats obtenus dans les simulations où on commande le réseau dynamiquement sont intéressants, car ils montrent clairement que des dispositifs de

compensation sur le réseau pour améliorer les performances du système de communication, permet aussi d'améliorer les performances du système à commander. Mais, il montre aussi, que la problématique d'instabilité inhérente au contrôle de process peut s'étendre à la commande du réseau. Il est important dans ce cadre, de pouvoir agir sur le réseau en évitant d'osciller entre deux commandes contradictoires.

Perspectives

Ce problème d'instabilité dans les réseaux correspond à un nouveau sujet de thèse développé au CRAN par I. Diouri. Le constat actuel est qu'il faut dans un premier avoir à sa disposition un superviseur de SCR qui permet d'apprécier son état relativement à des informations provenant du réseau mais aussi du système à commander. Ensuite des stratégies de commande du réseau efficace pourront être développées.

L'apport de notre travail de thèse est donc de proposer un cadre de modélisation de SCR pour faciliter sa co-conception. A partir de ce modèle, il est possible de l'enrichir en y ajoutant d'autres protocoles de communication. Notre idée serait maintenant d'élaborer des modèles de protocoles sans-fil qui commencent à être de plus en plus utilisés dans les SCR. Des contrôleurs de process plus complexes pourraient aussi être développés en Réseaux de Petri de Haut Niveau pour étudier la coordination des deux types de contrôleurs, l'un commandant le réseau et l'autre le process.

Finalement, une autre perspective est d'étendre les blocs de communication de Truetime de façon à pouvoir faire de la commande de réseau. Plus concrètement, il s'agit de doter le bloc commutateur Ethernet existant de mécanismes d'ordonnancement de trames.

PUBLICATIONS

Conférences internationales

B.Brahimi, C.Aubrun and E.Rondeau (2006). Comparison between Networked Control System behaviour based on CAN and Switched Ethernet networks, 2nd NeCST: Networked control system tolerant to fault Workshop, Calabria. October 2006.

B.Brahimi, C.Aubrun and E.Rondeau (2006). Modelling of scheduling policies implemented in Ethernet switch by using Coloured Petri Nets. Soumis à l'ETFA, 11th IEEE International Conference on Emerging Technologies and Factory Automation Prague, Czech Republic, 2006.

B.Brahimi, C.Aubrun and E.Rondeau (2006). Network calculus based FDI approach for switched Ethernet architecture. 6 th IFAC Symposium on Fault Detection, Supervision and safety of technical processes (Safeprocess), Beijing, PR. China, 2006.

E. Rondeau, J.-P. Georges, B. Brahimi, N. Vatanski and S.-L. Jämsä-Jounela, Simulation Platform for Behavioural Analysis of Networked Control Systems, 13th Nordic Process Control Workshop, Lyngby, Denmark. Lyngby 2006.

B. Brahimi, H. Demmou, A. Hélias and J.P. Steyer.Risk assessment for sake restarts of anaerobic digestion processes. 16th IFAC World Congress, Prague (République Tchèque), 3-8 Juillet 2005, 6p.

P. Boula de Mareuil, B. Brahimi and C. Gendrot (2004). Role of segmental and suprasegmental cues in the perception of maghrebian-accented . French INTERSPEECH 2004-ICSLP, 8th International Conference on Spoken Language Processing, Jeju Island, Korea Octiber 4-8, 2004

P. Boula de Mareuil et B. Brahimi (2004). Rôle du segmental et du suprasegmental dans la perception de l'accent maghrébin en Français. JEP 2004. Journées d'Etude sur la Parole 2004. Fès, Maroc, 19-22 avril 2004.

Conférences nationales

B. Brahimi, JP. Georges, N. Vatanski, E. Rondeau et C. Aubrun (2006). Plate-forme expérimentale pour analyser le comportement des systèmes contrôlés en réseau. Journées de la Section Automatique, Démonstrateurs en Automatique à vocation de recherche 28-29 mars 2006 à Angers.

Rapport de contrat européen NeCST

B.Brahimi, C.Aubrun et E.Rondeau (2006). Delivrable 5-4 : Study on embedded components. Projet STREP NeCST, mai 2006.

C.Aubrun, E.Rondeau, JP. Georges, N. krommenaker, V.Lecuire, B.Brahimi et F.Michaut. Delivrable 5-1: Simulation of optimized network architectures. Projet STREP NeCST, mai 2005.

Références Bibliographiques

[Abdellatif and Juanole 1999] S. Abdellatif and G. Juanole, Evaluation of quality of service of Periodic Producer/Consumer Relationship implemented on an ATM LAN, In Emerging Technologies and Factory Automation, 1999. Proceedings. ETFA '99. 1999.

[Abdellatif 2002] S. Abdellatif, Contribution à la modélisation et à l'analyse de la qualité de service dans les réseaux à commutation de paquets. Thèse de doctorat préparée au Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, Toulouse Janvier 2002.

[Albertos 2006] P. Albertos, Embedded Control Systems: Control Issues, in Graduate Course on Embedded Control Systems Department of Control Engineering, FEE, Prague, R-C, Avril 3-7th , 2006.

[Almutairi et *al*. 2001] N.B. Almutairi, M-Y. Chow and Y. Tipsuwan, Network-based controlled DC motor with fuzzy compensation. In proc. Of 27 th annual conference of the IEEE industrial electronic society (IECON'01), Vol. 3, pp. 1844-1849, Denver, CO, 2001.

[Altman et *al*. 1999] E. Altman, T. Bas,ar, and R. Srikant, "Congestion control as a stochastic control problem with action delays," Automatica, vol. 35, pp. 1937-1950, Dec. 1999.

[Alves et *al*. 2000] M. Alves, E. Tovar and F. Vasques, Ethernet goes real-time: a survey on research and Technological developments. Rapport de l'institut polytechnique de Porto (Ecole d'ingénieurs (ISEP-IPP), 2000.

[Andersson et *al*. 2005] M.Andersson, D.Henriksson and A.Cervin, Truetime -1.3:reference manual. http://www.control.lth.se/database/publications/article.pike?artkey=and%2b05tt

[Årzén et *al*. 1999] K-E. Årzén, B. Bernhardsson, J.Eker, A. Cervin, P. Persson, K. Nilsson and L.Sha, Integrated Control and Scheduling, Dept. of Automatic Control, Lund Institute of Technology, August, 1999.

[Årzén et *al*.2000] K-E. Årzén, A. Cervin, J. Eker and L. Sha, An Introduction to Control and Scheduling Co-Design, In Pro. of the 39th IEEE Conference on Decision and Control, Sydney, Australia, December 2000.

[Aubrun et *al*. 2005] C.Aubrun, E.Rondeau, JP. Georges, N. krommenaker, V.Lecuire, B.Brahimi et F.Michaut. Delivrable 5-1: Simulation of optimized network architectures. Projet STREP NeCST, mai 2005.

[Azimi-Sadjadi 2003] B.Azimi-Sadjadi, Stability of Networked Control System in the Presence of Packet Losses, In proc. Of 42nd IEEE conference on Decision and Control, December 2003.

[Bar-Noy et *al*. 1998] A.Bar-Noy, R.Bhatia, J.Naor, and B.Schieber. Minimizing Service and Operation Costs of Periodic Scheduling. In proc. Of Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 98), 11--20, 1998. http://citeseer.ist.psu.edu/bar-noy98minimizing.html.

[Bauer et *al*. 2001] P.H Bauer, M. Sichitiu, C.Lorand and K. Premaratne, Total Delay Compensation in LAN Control Systems and Implications for Scheduling, In LAN Control Systems and Implications for Scheduling of the American Control Conference, pp.4300-4305, June 2001.

[Baynat 2000] B. Baynat, Théorie des files d'attente, Hermès, Paris, 2000.

[Baynat and Dallery 2004 a] B. Baynat and Y. Dallery, Les méthodes markoviennes, la théorie des files d'attentes, Traité IC2 Information-Commande-Communication, Hermès Science, Lavoisier, N°ISBN 2-7462-0880-6, 2004, Chapitre 2, pp.51-120.

[Baynat and Dallery 2004 b] B. Baynat and Y. Dallery, Les réseaux de files d'attente, Traité IC2 Information-Commande-Communication, Hermès Science, Lavoisier, N°ISBN 2-7462-0880-6, 2004, Chapitre 3, pp.121-187.

[Ben Gaid et *al*. 2006] M. E.M. Ben Gaid, A. Çela et Y. Hamam, Optimal Integrated Control and Scheduling of Networked Control Systems with Communication Constraints:

Application to a Car Suspension System, In pro. Of IEEE Transactions On Control Systems Technology, Vol. 14 N° 4, July 2006.
[Benmohamed and Wang1998] L. Benmohamed and Y.T. Wang, A control-theoric ABR explicit rate algorithm for ATM swithes with per-vc queueing, in proc. Of IEEE infocom San-francisco, CA, mars 1998.
[Berbra et *al*. 2007] C. Berbera, S. Gentil, S. Lesceq, et J-M. Thiriet, Co-design for a safe network control DC motor, In proc. of 3rd Workshop of Européan Project, NeCST: Networked Control Systems Tolerant to fault, Nancy, France, Juin 2007.
[Bergé 1996] N. Bergé, Modélisation au moyen des réseaux de Petri stochastique d'une application de contrôle –commande de poste de transformation d'énergie électrique répartie sur le réseau de terrain FIP. PhD Dissertation. Laboratoire d'Analyse et d'Archiecture des Systèmes du CNRS, Université Paul Sabatier de Toulouse, Mai 1996.
[Bodenhofer and Herrera 1997] U. Bodenhofer and F. Herrera, Ten lectures on genetic fuzzy systems, in Reprints of the International suumer period school : advanced control fuzzy, neural, genetic, R. mesiar, ed. Bratislava, slovakia: Slovak technical Univ. 1997, pp. 1-69.
[Brahimi et *al*. 2006] B. Brahimi, E. Rondeau and C. Aubrun, Comparison between Networked Control System behaviour based on CAN and Switched Ethernet networks, in proc. Of 2nd Whorkshop of Networked control systems tolerant to fault (NeCST), Calabre 23-24th Novembre 2006.
[Branicky et *al*. 2000] M.S. Branicky, S. M. Phillips and W. Zhang, Stability of networked Control systems: Explicit Analysis of Delay, In Proc. of the American Control Conference (ACC'2000), Chicago, Illinois, June 2000.
[Branicky et *al*. 2002] M. S. Branicky, S. M. Phillips and W. Zhang, Scheduling and feedback co-design for networked control systems, in Proc. of the IEEE Conference on Decision and Control, December 2002.
[Branicky et *al*. 2003] M. S. Branicky, V. Liberatore, and S. M. Phillips, Networked Control System Co-Simulation for Co-Design, in Proc. American Control Conference, Denver, USA, vol. 4, pp. 3341--3346, June 2003.
[Brogan 1985] W. L. Brogan, Modern Control Theory, Second Edition, Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1985.
[Buttazzo 2000] G. Buttazzo, Adaptative rate control through elastic scheduling, in proc. Of IEEE conference on decision and control, pp. 4883-4888, Sydney, Australie, Décembre 2000.
[Buttazzo et *al*. 2004] G. Buttazzo, M. Velaski, P. Marti and G. Fohler, Managing Quality of control performance under Overload Conditions. In pro. Of IEEE 16th Euromicri Conference on real-time systems (ERTS'04), Juillet, 2004.
[CAN 1993] Road Vehicle-Interchange of Digital Information – Controller Area Network (CAN) for High Speed Communication, ISO 11898, ISO, 1993.
[Caponetto et *al*. 2002] R. Caponetto, L. Lo Bello, et O. Mirabella, Fuzzy Traffic Soothing : Another Step towards Statistical Real-Time Communication over Ethernet Networks, In proc. Of 1st Inerbational Workshop on Real-Time LANs in the Internet Age (RTLIA'2002), Vienne Autiche, pp- 45-48.
[Cervin 2003] A. Cervin, Integrated Control and Real-Time Scheduling, Ph.D. Dissertation, Dept. of Automatic Control Lund Institute of Technology, Lund, April, 2003.
[Cervin and Eker] A. Cervin and J. Eker, Feedback Scheduling of control tasks, in proc. Of IEEE conference on decision and control, pp. 4877-4882, Sydney, Australie, Décembre 2000.
[Cervin et *al.* 2002] A. Cervin , J. Eker, B. Bernhardsson and K-E. Årzén, Feedback-Feedforward Scheduling of control tasks, in Real time systems special Issue on control-theoretical approaches to real-time computing. 2002.
[Chan and Özgüner 1995] H. Chan and Ü. Özgüner, Closed-loop control of systems over communications network with queues, in proc. Of International Journal of Control, 63(3), pp. 493-510, 1995.

[Cheong So 2003] J.K. Cheong So, Delay Modelling and Controller Design for Networked Control Systems, Master of Applied Science Thesis, Dept. of Electrical and Computer Engineering in the University of Toronto, September 2003.
[Christensen 1998] K-J. Christensen, A simulation study of enhanced arbitration methods for improving Ethernet performance. In proc. Of Computer communications 21, pp. 24-36, 1998.
[Cisco]Cisco systems, Industrial Ethernet: A Control Engineer's Guide, In Cisco systems, http://www.cisco.com/application/pdf/en/us/guest/products/ps628/c1244/cdccont_0900aecd8013313e.pdf.
[Cottet et *al*. 2000] F. Cottet, J. Delacroix, C. Kaiser et Z. Mammeri, « Ordonnancement temps réel », Edition Hermes, 2000.
[Cruz 1999 a] R.L. Cruz, A calculus for network delay, part I: Network element in isolation, in proc. Of IEEE Transaction on information theory, 37:114-131, Janvier 1999.
[Cruz 1999 b] R.L. Cruz, A calculus for network delay, part II: Network element in isolation, in proc. Of IEEE Transaction on information theory, 37:114-131, Janvier 1999.
[David and Alla 1989] R. David and H. Alla, Du grafcet aux réseaux de Petri, Edition Hermes, Paris, 1989.
[David and Alla 1994] R. David and H. Alla, Petri Nets for Modelling of Dynamic Systems – A Survey. In proc. Of Automatica, Vol. 30, N° 2, pp. 175-202, 1994.
[De Larminat 1993] P. De Larminat, Automatique - commande de systèmes linéaires, Editions Hermes, Paris, 1993.
[Decotignie 2005] J-D. Decotignie, Ethernet Based Real-Time and Industrial Communications, In proc. Of IEEE, Vol. 93, N°.6, June 2005.
[Dev. 1997] DeviceNet spécifications. Boca raton, Florida, Open DeviceNet Vendors Association, 2.0 edition, 1997.
[Dolejš et *al*. 2004] O. Dolejš, P. Smolík, Z. Hanzálek, On the Ethernet Use for Real-Time Applications, 5th IEEE International Workshop on Factory Communication Systems, WFCS, Vienna, September 22-24, 2004.
[El Mongi Ben Gaid et *al*. 2006] M. E.M. Ben Gaid, A. Çela et Y. Hamam, Optimal Integrated Control and Scheduling of Networked Control Systems with Communication Constraints: Application to a Car Suspension System, In pro. Of IEEE Transactions On Control Systems Technology, Vol. 14 N° 4, July 2006.
[El-Derini and El-Sakka 1990] M. El-Derini et M. El-Sakka, A CSMA protocol under a priority time constrained for real-time communication, In proc. Of Future Trends on Distributed Computing Systems, pp. 128-134, Caire, Egypte, 1990.
[Fdida and Hébuterne 2004] S. Fdida and G. Hébuterne, Introduction la modélisation des outils, Traité IC2 Information-Commande-Communication, Hermès Science, Lavoisier, N°ISBN 2-7462-0880-6, 2004, Chapitre 1, pp.21-50.
[Feather et *al*. 93] Feather, F., D. Slewiorek and R.Maxion .(1993). Fault detection in an Ethernet network using anomaly signature matching. SIGCOMM'93- Ithaca, N.Y., USA September 1993.
[Filipiak 1988] J. Filipiak, Modelling and control of dynamic flows in communication networks, Berlin: Springer, 1988.
[FIP 1989] FIP Bus for Exchange of Information between Transmetters, Actuators and Process controllers, Standard Français NF c 46, 1989.
[Fraisse et *al*. 2007] P. Fraisse, A. Lelevé and W. Perruquetti, Robot en réseau, Traité IC2 Information-Commande-Communication, Hermès Science, Lavoisier, N°ISBN 978-2-7462-1513-9, 2007, Chapitre 4, pp.125-188.
[Gallager and Parekh 1994] R. G. Gallager and A. K. J Parekh. A generalized processor sharing Approach to flow control in integrated services networks: the multiple-node case. In proc. Of IEEE/ACM Transaction on Networking, volume 2, pages 137–150,Avril 1994.

[Georges 2005] J-P. Georges, Systèmes contrôlés en réseau : Evaluation de performance d'architectures Ethernet commutées, PhD dissertation. Centre de Recherche en Automatique de Nancy, Université Henri Poincaré, Novembre 2005.
[Georges et *al*. 2005] J-P.Georges, T.Divoux, et E.Rondeau. Conforting the performances of switched Ethernet network with industrial constraints by using the network calculus. International Journal of Communication systems 2005, 18:877-903.
[Georges et *al*. 2006 a] J-P.Georges, N. Vatanski E.Rondeau and C. auburn, Network instrumentation, In deliverable WP5.2 of NeCST European project (Networked Control System Tolerant to Fault, May 2006.
[Georges et *al*. 2006 b] J-P. Georges, N. Vatanski, E. Rondeau, SL. Jämsä-Jounela, Use of upper bound delay estimate in stability analysis and robust control compensation in networked control systems, In 12th IFAC Symposium on Information Control Problems in Manufacturing, St-Etienne, May 2006.
[Georges et *al*. 2006] J.P. Georges, N. Vatanski, E. Rondeau, C. Aubrun et SL. Jamsa-Jounela, Control compensation based on upper bound delay in networked control systems, In proc. Of 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto, Juillet, 2006.
[Georges et *al*. 2007] J-P. Georges, T. Divoux et E. Rondeau, Evaluation d majorants des délais de transmission pour les systèmes contrôlés en réseau, Traité IC2 Information-Commande-Communication, Hermès Science, Lavoisier, N°ISBN 978-2-7462-1513-9, 2007, Chapitre 5, pp.189-230.
[Göktas 2000] F. Göktas, Distributed control systems over a communication networks, PhD. Dissertation, université de Pennsilvanie, 2000.
[Grieu 2004] J. Grieu. Analyse et évaluation des techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques. PhD. Dissertation. Institut National Polytechnique INP de Toulouse, Septembre 2004.
[Gu and Shin 2003] Z. Gu and K.G. Shin, An Integrated Approach to Modeling and Analysis of Embedded Real-Time Systems Based on Timed Petri Nets, In ICDCS archive Proceedings of the 23rd International Conference on Distributed Computing Systems, Page: 350, 2003.
[Gupta 2001] V.Gupta, A distributed backoff algorithm to support real-time traffic on Ethernet, In ACM Oper.Syst. Rv., Vol.35, n°.3, pp.43-66, July 2001.
[Halevi and Ray 1988] Y. Halevi and A. Ray, Integrated communication and control systems: part I- Analysis. In Journal of Dynamic systems, Measurment and Control, 110, pp. 367-373. 1988.
[Hamdaoui and Ramanathan 1995] M. Hamdaoui and P. Ramanathan, A dynamic priority assignment technique for streams with (m,k)-firm deadlines, in proc. Of IEEE transaction on computers, 44(4), pp-1443-1451, Décembre 1995.
[Hartman 2004] J. Hartman, Networked Control System Co-simulation for Co-design: Theory and Experiments, Master Science Thesis, Department of Electrical Engineering and Computer Science Case Western Reserve University, June, 2004.
[Henrikson et *al*. 2002] D. Henriksson, A. Cervin and K-E. Årzén, TrueTime : Simulation of Control Loops under Shared Computer Resources,15th Triennial World Congress, IFAC, Barcelona, Spain, 2002.
[Henriksson and Cervin 2005] D. Henriksson and A. Cervin, Optimal on line sampling period assignement for real time control based on state information, in proc. Of 44 th IEEE Conference on decision and control, and the European control conference, Seville espagne, décembre 12-15, 2005.
[Henriksson et *al*. 2003] D. Henriksson, A. Cervin and K-E. Årzén, TrueTime: Real-Time Control System Simulation with Matlab/Simulink, In Proc. Of the Nordic MATLAB Conference, Copenhagen, Denmark, October 2003.

[Hoang 2004] H. Hoang, Real-Time Communication for Industrial Embedded Systems Using Switched Ethernet, in proc. Of the 18th International Parallel and Distributed Processing Sympsium (IPDPS'04), 2004.
[Hoang et *al*. 2002] H. Hoang, M. Jonsson, A. Larsson, R. Olsson, et C. Bergenhem. Deadline First Scheduling in Switched Real-Time Ethernet- Deadline Partitionning Issues and Software Implementation Experiments. In proc. Of 1st Intl. Workshop on Real-Time LANs in the Internet Age Satellite Event to 14th Euromicro Conference on Real- Time Systems Technical University of Vienna, Austria, 18 juin 2002.
[Hong 1995] S.H. Hong, Scheduling algorithm of data sampling times in the integrated communication and control systems, in proc. Of IEEE Transactions On Control Systems Technology, 3(2), pp. 225-230, 1995.
[Hong and Kim 2000] S.H. Hong and W.H. Kim, Bandwidth allocation scheme in CAN protocol, In proc. Of IEE of the control theory and Applications, pp. 37-44, 2000.
[IEEE 802.1 D] IEEE 802.1 D, 1998 Edition: Local and Metropolitan Area Networks: Media Access Level Bridging.
[IEEE 802.3] IEEE Std 802.3, 1998 Edition: Information technology- Telecommunication and information exchange between systems- Local and metropolitan area networks- specific requirements- Part 3: Carrier senses multiple access with collision detection (CSMA/CD) access method and physical layer specifications.
[IEEE prio 93] IEEE 802.1F, Common Definitions and Procedures for IEEE 802 Management Information, 1993.
[IEEE VLAN 98] IEEE standard 802.1Q? Virtual Bridged Local Area Networks, 1998.
[Jasperneite and Neuman 2001] E. Jasperneite and P. Neumann, Switched Ethernet for Factory Communication, In proc. IEEE Int. Conf. Emerging Technologies and Factory Automation, vol. 1, pp. 205-212, 2001.
[Jasperneite et *al*. 2002] J. Jasperneite, P. Neumann, M. Theis and K.Watson, Deterministic real-time communication with switched Ethernet, in Proc. Of 4th IEEE International Workshop on Factory Communication Systems, 2002, pp.11- 18.
[Jensen 1992] K. Jensen, Coloured Petri Nets: Basic Concepts, Analysis Methods and Practice Use, Volume 1, Basic concepts, Springer-Verlag, monographs in heoretical computer science edition, 1993.
[Jensens 1980] K. Jensens, How to find invariants for Coloured Petri Nets. Aarhus Univ., Comput Science Dep., DAIMI PB-120, 1980.
[Ji and Kim 05] K. Ji and W-J. Kim, Real-Time of Networked Control Systems via Ethernet, In International Journal of Control, Automation, and Systems, vol.3, n°4, pp. 591-600, December 2005.
[Juanole 2000] G. Juanole, Quality of service of communication networks and distributed automation: models and performances, In proc. of 15th Triennial World Congress of the IFAC, Barcelone (Espagne), 21-26 Juillet 2002, 6p.
[Juanole and Blum 1999] G. Juanole and I. Blum, Influence de fonctions de base (communication-ordonnancement) des systèmes distribués temps –réel sur les performances d'applications de contrôle-commnade. Dans les actes de Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'99), Nancy, France 26-29 avril 1999.
[Juanole and Mouney 2005] G. Juanole and G. mouney, QOS in Real Time Distributed Systems and Process Control Applications, In Proc. Of Workshop on Networked Control Systems (NeCST), European Project, Corsica, France, October 2005.
[Juanole and Mouney 2006] G. Juanole, G. Mouney, Real time distributed systems: QoS and impact on the performances of process control applications, in Proc. Of 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS'2006), Kyoto (Japon), 24-28 Juillet 2006, pp.1739-1746.

[Juanole and Mouney 2007] G. Juanole, G. Mouney, Impact du réseau sur les performances des applications, Systèmes Commandés en réseau, Traité IC2 Information-Commande-Communication, Hermès Science, Lavoisier, N°ISBN 978-2-7462-1513-9, 2007, Chapitre 1, pp.21-52 .
[Juanole et *al*. 2004] G.Juanole, M.Diaz, et F.Vernadat. Les réseaux de Petri étendus et méthodologie pour l'analyse de performances. In the book 'Méthodes exactes d'analyse de performance des réseaux'. IC2 Réseaux et Télécoms (Information –Commande – Communication), Edition Hermès science, 2004.
[Juanole et *al*. 2005] G. Juanole, C. Calmettes, G. mouney and M. Peca, On the Implementation of a Process Control System on a CAN Network : Linking the process Control Parameters to the Network Parameters, In Proc.of the IEEE International conference on Emerging Technologies and Factory Automation, ETFA'05, Catania, Italy, September 2005.
[Kalyanaraman et *al*. 1997] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal and B. Vandalore, The ERICA switch algorithm for ABR traffic management en ATM network, 1997.
[Kim et *al*. 1998] Y.H. Kim, H.S. Park and W.H. Kwon, A scheduling Method of network-based control systems, in proc. Of the American Control Conference, Vol.2, pp. 718-722, Philadelphia, PA, 1998.
[Kolarov and Ramamurthy 1997] A. Kolarov and G.Ramamurthy, A control theoretic approach to the design of closed loop rate based flow control for high speed ATM networks. In proc. Of IEEE Infocom, 1997.
[Kopetz et *al*. 1989] H. Kopetz, A. Damm, C. Kosa, M. Mulazzani, W. Schwalb, C. Senft and R. Zailinger, Distributed Fault-Tolerant Real-Time Systems: The MARS approach, in proc. Of IEEE Micro, ç(1), pp. 25-40, Février, 1989.
[Kosgue et *al*. 1996] K. Kosuge, H.Murayama and K.Takeo, Bilateral Feedback Control of Telemanipulators via Computer Network, in Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS'96), pp. 1380-1385, Osaka, Japon, November 4-8, 1996.
[Koubâa 2004] A. Koubaâ, Gestion de la Qualité de Service temporelle selon la contrainte (m,k)-firm dans les réseaux à commutation de paquets, LORIA-TRIO,thèse octobre 2004.
[Koubâa and Song 2004] A. KOUBAA, Y-Q. SONG, (m,k)-WFQ : Integrating (m,k)-Firm Real-Time Constraints into Guaranteed-Rate Networks To Appear in the Proceedings Real Time Systems Conference RTS'2004 Paris (France) 30 Mars-1 Avril 2004.
[Koubâa and Song 2004] A. KOUBAA, Y-Q. SONG, (m,k)-WFQ : Integrating (m,k)-Firm Real-Time Constraints into Guaranteed-Rate Networks To Appear in the Proceedings Real Time Systems Conference RTS'2004 Paris (France) 30 Mars-1 Avril 2004.
[Krádora and Hanzalek 2004] J. Krádora and Z. Hanzalek, Timed Automata approach to CAN verification. In proc. of 11th IFAC symposium on information control problems in manufacturing (INCOM'04), Salvador Bahia, Bresil, 2004.
[Krákora and Hanzálek 2004] J. Krákora, and Z. Hanzálek, Timed Automata Approach for CAN Verification , 11th IFAC Symposium on Information Control Problems in Manufacturing, INCOM, Salvador, Elsevier, April 2004.
[Krákora et *al*. 2004] J. Krákora, L. Waszniowski, P. Píša, Z. Hanzálek, Timed Automata Approach to Real Time Distributed System Verification, 5th IEEE International Workshop on Factory Communication Systems, WFCS, Vienna, September 22-24, 2004.
[Kristensen et *al*. 2006] L.M. Kristensen, P. Mechlenborg, L. Zhang, B. Mitchell, G.E. Gallash: Model-based Development of a Course of Action Scheduling Tool In K. Jensen (ed.): Proceedings of the Seventh Worskhop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, October 2006, Department of Computer Science, University of Aarhus, PB-579, 1-16.

[Krommenaker 2002] N. Krommenaker, Heuristiques de conception de topologies réseaux : application aux réseaux locaux industriels, PhD thesis. Centre de Recherche en Automatique de Nancy (CRAN), Université Henri Poincaré, 2002.
[Kurose et *al*. 1984] J-F. Kurose, M. Schwartz et T. Yemini, Multiple access protocols and time-constrained communication, in Computing Surveys, Vol. 16, n°1, Mars 1984.
[Kweon et *al*. 1997] S-K. Kweon, K-G. Shin et Q. Zheng, Statistical Real-Time communication over Ethernet/Fast Ethernet, Technical Report, Mitsubishi Electric Research Lab. Cambridge, MA, Août 1997.
[Kweon et *al*. 1999] S-K. Kweon, K-G. Shin and Q. Zheng, Statistical Real-time communication over Ethernet for manufacturing automation systems, in proc. Of IEEE real-time Technology and Applications Symposium (RTAS'1999), pp. 192-202, Juin 1999.
[Lann and Rivière 1993] G. Lann and N. Rivière, Real-time communications over broadcast networks: the CSMA/DCR and the DOD CSMA/CD protocols, Technical report 1863, INRIA, Mars 1993.
[Latu 2007] P. Latu, Technologie Ethernet, Version révisée, http ://www.linux-france.org/prj/inetdoc/articles/ethernet/ .
[Le Boudec and Thiran, 2001] J.Y. Le Boudec and P. Thiran. Network calculus: a theory of deterministic queuing systems for the Internet. In Lecture Notes in computer science, volume 2050. Springer Verlag, 2001.
[Lee and Lee 2002] K.C.Lee and S.Lee, Performance evaluation of switched Ethernet for real-time industrial communications. Computer Standards & Interfaces 24 (2002) 411-423.
[Lelevé 2000] Lelevé A. : «Contribution à la téléopération de robots en présence de délais de transmission variable », thèse préparée à l'Université de Montpellier II, Science et Technique du Languedoc, décembre 2000.
[Lian 2001] F-L.Lian, Analysis, Design, Modelling and Control of Networked Control Systems. PhD dissertation, Mechanical Engineering University of Michigan, 2001.
[Lian et *al*. 2001] F-L Lian, J.R. Moyne and D.M. Tilbury, Performance Evaluation of Control Networks: Ethernet, ControlNet, and DeviceNet, In Proc. of Control Systems Magazine, IEEE Volume 21, Issue 1, Feb 2001 Page(s):66 – 83.
[Lian et *al*. 2001] F-L. Lian, J. Moyne and D. Tilbury, Analysis and Modelling of Networked Control Systems: MIMO Case with Multiple Time Delays, In Proc. of the American Control Conference Arlington, Virginia, June 2001.
[Liou and Ray 1990] L.W. Liou and A. Ray, Integrated communication and control systems: part III- Nondynamical sensor and controller sampling. In Journal of Dynamic Systems, Measurement and Control, 112, 357-364, 1990.
[Liu and Layland 1973] C. L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, 20(1):46–61, 1973.
[Lo Bello and Mirabella 2004] L. Lo Bello, O. Mirabella, An Approach to Reduce Application-to-Application Latency in a Real-Time Distributed Architecture. In proc. Of RTN 2004 - 3rd Int. Workshop on Real-Time Networks 2004.
[Lo Bello et *al*. 2005] L.Lo Bello, G.A. Kaczyński and O. Mirabella, Improving the Real-Time Behaviour of Ethernet Networks Using Traffic Smoothing, In IEEE Transactions On Industrial Informatics, Vol.1, N° 3, August 2005.
[Loiseau and Rabah 1997] J-J. Loiseau and R. Rabah, Analysis and control of time-delay systems, in JESA, European Journal of Automatic Systems, 31(6) (Special Issue of JESA), 1997.
[Lu 2001] C.Lu, feedback real-time scheduling, in PhD. Dissertation presented to the Faculty of the School of Engineering and Applied Science, University of Virginia, Mai 2001.
[Luck and Ray 1990] R. Luck and A. Ray, An observer-based compensator for distributed delays, Automatica, 26(5), pp. 903-908, 1990.

[Luck and Ray 1994] R. Luck and A. Ray, Experimental verification of delay compensation algorithm for integrated communication and control systems, in proc. Of international journal of control, 59(6), pp. 1357-1372.
[Malcolm and Zhao 1995] N. Malcolm et W. Zhao, Hard real-time communication in Multiple access networks, in Journal of Real-Time Systems, Vol. 8, pp. 35-37, Kluwer Academic Publishers, 1995.
[Marau et *al*. 2006] R. Marau, P. Pedreiras, et L. Almeida, Enhanced Ethernet Switching for Flexible Hard Real-Time Communication, In proc. Of Real-Time Network, RTN'06, Dresden, 04 Juillet, 2006.
[Marsal 2006] G. Marsal, Evaluation of time performances of Ethernet-based Automation System by simulation of High Level Petri Nets, PhD dissertation de l'Ecole Normale Supérieure de Cachan et de l'Universté de Kaiserslautern, Décembre 2006.
[Marsal et *al*. 2005] G.Marsal, D.Witsch, B.Denis, J.-M.Faure et G.Frey. Evaluation of real-time capabilities of Ethernet-based automation systems using formal verification and simulation. Ecole d'été Temps Réel , Nancy, Septembre. ETR'2005.
[Marsal et *al*. 2006] G.Marsal, B.Denis, J.-M.Faure et G.Frey. Evaluation of Response Time in Ethernet-based Automation Systems. In Emerging Technologies and Factory Automation proceeding. ETFA'2006.
[Marti et *al*. 2004] P. Marti, C. Lin, S.A.Brandt, J.M. Fuertes, Optimal state feedback based resource allocation for resource –constrained control task, in proc. Of 25th IEEE International Real-Time Systems Symposium, pp-161- 172, 5-8 Dec. 2004.
[Mascolo 1999] S. Mascolo, "Congestion control in high –speed communication networks using the smith predictor principle," in Automatica, vol. 35, n° 12, dec. 1999.
[Mascolo 1999] S. Mascolo, Classical control theory for congestion avoidance in high-speed Internet, in pro. IEEE conf. Decision and control Phoenix, pp. 2709-2714. December 1999.
[Mascolo 2003] S. Mascolo, Modeling the internet congestion control using a smith controller with input shaping, in proc. Of IFAC'03 workshop on time delay systems, 2003.
[Mascolo 2007] S. Mascolo, Quality of Service Control: Quality of Service Control: The case of IEEE 802.11e WLANs The case of IEEE 802.11e WLANs, in proc. Of 3rd Whorkshop of Networked control systems tolerant to fault (NeCST), Nancy 20-21 juin 2007.
[Mascolo et *al*. 1996] S. Mascolo, D. Cavendish, and M. Gela, ATM rate based congestion control using smith predictor : An EPRCA implementation, In proc. Of IEEE Infocom san fransisco, CA, mars 1996.
[Michaut and Lepage 2003] F. Michaut and F. Lepage, Un cadre pour la prise en compte du retard dans la téléopération d'un robot mobile. In JDA'03 Journée Doctorales d'Automatique, Valenciennes, France, 2003.
[Milner et *al*. 1997] R. Milner, M. Tofte, R. Harper and D. MacQueen, The definition of standard ML- Revised, May 1997.
[Moncelet 1998] G. Moncelet, Application des réseaux de Petri à l'évaluation du la sûreté de fonctionnement des systèmes mécatroniques du monde automobiles, Thèse préparée au Laboratoire d'Analyse et l'Architecture des Systèmes du CNRS, (LAAS), Octobre 1998.
[Moncelet et *al*. 1998] G. Moncelet, S. Christensen, H. Demmou, M. Paludetto, J. Porras: Analysing a Mechatronic System with Coloured Petri Nets. International Journal on Software Tools for Technology Transfer, 2 (1998), Springer-Verlag, 160-167.
[Moukrim and Picouleau 2003] A. Moukrim et C. Picouleau, « ordonnancement pour l'informatique parallèle », Edition Lavoisier, Hermès sciences publications, 2003.
[Navet 1999] N. Navet, Evaluation des performances temporelles et optimisation de l'ordonnancement de tâche et messages. Thèse de doctorat, Institut National Polytechnique de Lorraine, 1999.
[Nešić and Teel 2004] D. Nešić, A.R. Teel, Input-output stability properties of networked control systems, in poc. of, IEEE Transactions on Automatic Control, Volume: 49, Issue: 10

[Nilsson 1998] J. Nilsson, Real Time Control Systems with Delays, Ph.D. dissertation, Dept. Automatic Control, Lund Institut of Technology, Lund, Sweden, January 1998.
[Nilsson 1998] J. Nilsson, Real Time Control Systems with Delays, Ph.D. dissertation, Dept. Automatic Control, Lund Institut of Technology, Lund, Sweden, January 1998.
[Ohlin et *al*. 2006] M.Ohlin, D.Henriksson and A.Cervin, Truetime -1.4:Reference manual. http://www.control.lth.se/database/publications/article.pike?artkey=ohl%2b06tt
[Ohlin et *al*. 2007] M.Ohlin, D.Henriksson and A.Cervin, Truetime -1.5:Reference manual. http://www.control.lth.se/database/publications/article.pike?artkey=ohl%2b07tt.
[Overstreet and Tzes 1999] J-W. Overstreet and A. Tzes, An Internet-based real-time control engineering laboratory, in IEEE Control systems magazine, 1999, pp.19-34.
[Paret 1999] D. Paret, Le bus CAN applications CAL, CANOpen, DeviceNet, OSEK, SDS…, Editions DONUD, Paris, 1999.
[Pedreiras and Almeida 2002] P. Pedreiras, et L. Almeida. Flexibility, Timeliness and Efficiency in Ethernet. In proc. Of RTLIA'02, 1st Workshop on Real-Time LANs in the Internet Age, (satellite of ECRTS'02), Vienna, Austria, Juin 2002.
[Pedreiras and Almeida 2005] P. Pedreiras, et L. Almaida, in The industrial Communication Technology Handbook, Zurawski R. Edition, CRC Press, Boca Raton, FL 2005.
[Pedreiras et *al*. 2002] P. Pedreiras, L. Almeida, et P. Gai, The FTT-Ethernet protocol: merging flexibility, timeliness and efficiency, in proc. Of 14th Euromicro Conference on Real-Time Systems, pp. 134-142, 2002.
[Pedreiras et *al*. 2003] P. Pedreiras, R. Leite and L. Almaida, characterizing the real time behaviour of prioritized switched Ethernet, in proc. On 2nd Int. Workshop Real-time LAN's in the internet age, Vienna, Austria, 2003.
[Pedreiras et *al*. 2003] P. Pedreiras, R. Leite and L. Almaida, characterizing the real time behaviour of prioritized switched Ethernet, in proc. On 2nd Int. Workshop Real-time LAN's in the internet age, Vienna, Austria, 2003.
[Pettersson and Larsen 2000] P. Pettersson and K. G. Larsen, UPPAL2K, 2000.
[Peyrucat 2005] J-F. Peyrucat, Ethernet se rapproche des bus de terrain, Dossier 'Ethernet' dans MESURES n°772, pp. 21-39, Février 2005.
[Phipps 1999] M. Phipps, Understanding: The basics. A guide to LAN Switch Architectures and Performance. CISCO SYSTEMS, 1999.
[Profibus 1992] Profibus Standard DIN 19245 Part I and II, traduit de l'allemand , Nutzer organisation 1992.
[Repère 1997] Repères, Edition Terrain, ControlNet ouvert, N°12, Février, Mars avril 1997.
[Richard 2003] J-P. Richard, Time-delay Systems : an overview of some recent advances and open problems, In Automatica 2003, pp. 1667-1694.
[Richard 2003] J-P. Richard, Time-delay systems: an overview of some recent advances and open problems, in Automatica 39 pp. 1667-1694. 2003.
[Richard and Divoux 2007] J-P. Richard et T. Divoux, « systèmes commandés en réseau », Edition Lavoisier, Hermès sciences publications, 2007.
[Richard et *al*. 2003] P. Richard, M. Richard et F. Cottet, Analyse holistique des systèmes temps réel distribués : Principes et algorithmes. Cité dans « ordonnancement pour l'informatique parallèle » sous la direction de A. Moukrim et C. Picouleau parue dans IC2 Informatique et systèmes d'information Edition Lavoisier, Hermès sciences publications, 2003.
[Rondeau et *al*. 2001] E. Rondeau, T. Divoux and H. Adoud, Study and method of Ethernet architecture networks with differents topologies used in automation systems, in 3rd IFAC International Conference on Fielbus Systems and their Applications (FET'2001), Nancy, France, 2001, pp. 351-358.
[Rondeau et Divoux 2001] E. rondeau et T. Divoux, '' Conception d'architecture de réseaux Ethernet industriels'', dans j'automatise N°18, pp. 52-58, Novembre- Décembre 2001.

[Rozinat et *al*. 2006] A. Rozinat, R.S. Mans, and W.M.P. van der Aalst: Mining CPN Models: Discovering Process Models with Data from Event Logs In K. Jensen (ed.): Proceedings of the Seventh Worskhop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, October 2006, Department of Computer Science, University of Aarhus, PB-579, 57-76.
[Ruping et *al*. 1999] Rüping S., Vonnahme E., Jasperneite J. "Analysis of Switched Ethernet Networks with different topologies used in automation systems", In Proc. Fieldbus Conference (FeT'99), Magde-burg, Allemagne, pages 351-358,septembre 1999.
[Rüping et *al*. 1999] S. Ruping, E. Vonnhame, and J. Jasperneite, Analysis of Switched Ethernet Networks with Diffetrent Topologies Used in Automation Systems, In proc. Of Fieldbus Conference (FeT'99), Magdeburg, Germany, pp. 351-358, springer Verlag, Septembre 1999.
[Rüping et *al*. 1999] S. Rüping, E. Vonnhame, and J.Jasperneite. Analyse of switched Ethernet networks with different topologies used in automation systems. Proc. Of fieldbus technology conference (FeT'99), 351-358, Magdeburg, Springer-Verlag, 1999.
[Seifert 1995] R. Seifert, Issues in LAN Switching and Migration from Shared LAN Environment, Technical Report, Networks ans Communications Consulting, Novembre 1995.
[Seifert 2000] R. Seifert, The Switch Book: The complete Guide to LAN Switching Technology, Published by John Wiley & Sons, Inc, 2000.
[Sename et *al*. 2003] O. Sename, D. Simon and D. Robert, Feedback Scheduling for Real-Time Control of Systems with Communication Delays, In IEEE Conference ETFA'03, Lisbonne, Portugal, 2003.
[Seng et *al*. 1999] T.L.Seng, M.Khalid and R. Yusof, Tunning of a neuro-fuzzy controller by genetic algorithm, IEEE Transaction systems, Man, Cybern.B, cybern.,vol.29, n°.2, pp.226-236, Avril 1999.
[Seuret et *al*. 2007] A. Seuret, J-P. Richard , F. Michaut and F. Lepage, Commande à travers Internet : prise en compte des retards variables, Traité IC2 Information-Commande-Communication, Hermès Science, Lavoisier, N°ISBN 978-2-7462-1513-9, 2007, Chapitre 2, pp.55-81.
[Sha et *al*. 2000] L; Sha, X. Liu, M. Caccamo and G. Buttazzo, On line control Optimization using load driven scheduling, in proc. Of IEEE conference on decision and control, pp. 4877-4882, Sydney, Australie, Décembre 2000.
[Shreedhar and Varghes 1996] M. Shreedhar and G. Varghes, Efficient fair queuing using deficit round robin. In proc. Of IEEE ACM Transations On Networking, Juin 1996.
[Sifakis 1977] J. Sifakis, Etude du comportement permanent des réseaux de Petri temporisés, Journées AFCET sur les RDPs , Paris 1977.
[Simon et *al*. 2005] D. Simon, D. Robert, O. Sename. Robust control/scheduling co-design: application to robot control. In RTAS'05 IEEE Real-Time and Embedded Technology and Applications Symposium, Pages 118-127, San Francisco, Mars 2005.
[Song 2001] Y-Q. Song, Time Constrained Communication Over Switched Ethernet, in proc. Of 4th IFAC International Conference on Fieldbus Systems and their Applications - FeT'2001.
[Stankovic et *al*. 2001J. Stankovic, T. He, T. Abdelzaher, M. Marley, G. Tao, and S. H. Son. Feedback Control Scheduling in Distributed Real-Time Systems. In Real-Time Systems Symposium, London, England, December 2001.
[Tanwan et *al*. 2006] A. Tanwan, J. Galdun, J-M. Thiriet, S. Lesceq et S. Gentil, Modeling of an Experimental Networked Drone with consideration of faults, In proc. Of ACD -2006. http://www.acd-2006.cran.uhp-nancy.fr/Files/ACD/p24.pdf
[Tarn and Xi 1998] T-J. Tarn and N. Xi, planning and control of internet-based teleoperation, in proc. of SPIE, Telemanipulator and Telepresence technologies, Vol. 3524, Bostion 1998, pp. 189-193.

[Tarn and Xi 1998] T-J. Tarn and N. Xi, planning and control of internet-based teleoperation, in proc. of SPIE, Telemanipulator and Telepresence technologies, Vol. 3524, Bostion 1998, pp. 189-193.
[Thomesse et *al*. 1995] J-P. Thomesse, Z. Mammeri and L. Vega, Time in distributed Systems cooperation and communication models, in poc. Of 5 th Workshop on future trends of distributed systems, 1995.
[Tindell et *al*. 1995] K. Tindell, A. Burns, and A.J.Wellings, calculating controller area network (CAN) message response time, Contr. Eng. Practice, vol.3, n°8. pp. 1063-1169.
[Tindell et *al*. 1995] K.Tindell, A.Burns, and A.J.Wellings, "Calculating Controller Area Network (CAN) message response time", Control engineering Practice, vol. 3, n°8, pp. 1163-1169, Aug. 1995.
[Tipsuwan and Chow 2001] Y. Tipsuwan and M-Y. Chow, Network-based controller adaptation based on QoS negotiation and deterioration. In proc. of the 27th annual conference of the IEEE industrial electronics society (ICON 01), Vol 3, Denver ,2001, pp. 1794-1799.
[Tipsuwan and Chow 2002] Y. Tipsuwan and M-Y. Chow, Gain adaptation of Networked mobile robot to compensate QoS deterioration., in proc. of the 28th annual conference of the IEEE industrial electronics society (ICON 01), Vol 3, Denver ,2001, pp. 1794-1799.
[Tipsuwan and Chow 2003] Y. Tipsuwan and M-Y. Chow, Control Methodologies en Networked Control systems, In Proc. of Control engineering Practice 2003, pp. 1099-1111.
[Tobagi 1982], F. Tobagi, Carrier Sense Multiple Access with Message-Based Priority Functions, In proc. Of IEEE Transactions on Communcations, Vol. Com-30, n°1, Janvier 1982.
[Travé-Massuyès et *al*. 1997] L. Travé-Massuyès, P. Dague and F. Guerrin, Le raisonnement qualitatif, Edition Hermès, 1997.
[Ullman 1998] J. D. Ulman, Elements of ML programming, 2nd Edition (ML97), 1998.
[Vega and Thomesse 1995] L. Vega and J.P. Thomesse. Temporal properties in distributed real-time applications. In 13th Workshop on Distributed Computer Control Systems, DCCS'95, IFAC. Toulouse (France), September 1995.
[Walsh et *al*. 1999] G.C. Walsh, H. Ye and L. Bushnell, Stability Analysis of Networked Control Systems, In Proc. American Control Conference, Chicago, pp. 2876-2880, June 1999.
[Walsh et *al*. 1999] G.C. Walsh, H. Ye, and L. Bushnell, Stability analysis of networked control systems, in Proc. Amer. Control Conf., San Diego, CA, June 1999, pp. 2876-2880.
[Wang and Levy 2000] J. Wang and Y. Levy, Managing performance using Weighted Round Robin, In proc. of Fifth IEEE Symposium on Computers and Communications (ISCC 2000), 2000.
[Wang et *al*. 2002] Z.Wang, Y-Q. Song, J-M. Chen and Y-X. Sun, Real time characteristics of Ethernet and its improvement, In proc. 4ème World congress on intelligent control and automation, Shanghai, Chine, June 10-14, 2002.
[Wang et *al*. 2002] Z.Wang, Y-Q. Song, J-M. Chen and Y-X. Sun, Real time characteristics of Ethernet and its improvement, In proc. 4ème World congress on intelligent control and automation, Shanghai, Chine, June 10-14, 2002.
[Watanabe et *al*. 1996] K.Watanabe, E. Nobuyama and A. Kojima, Recent advances in control of time delay systems- a tutorial review, in Proc. Of the IEEE Conference on Decision and Control, pp.2083-2088, December 1996.
[Watson and Jasperneite 2003] K. Watson, J. Jasperneite, Determining end-to-end delays using network calculus, In proc.of International Federation of Automatic Control -IFAC-:FET 5th IFAC International Conference on Fieldbus Systems and their Applications, July 7-8, Aveiro, Portugal 2003 pp.255-260.
[Witrant et *al*. 2007] E. Witrant, D. Georges, C. Canudas de Wit and O. Sename, Stabilisation des systèmes commandés par le réseau : une approche prédictive, Traité IC2 Information-Commande-Communication, Hermès Science, Lavoisier, N°ISBN 978-2-7462-1513-9, 2007, Chapitre 3, pp.83-123.

[Yamé and Sauter 2007] J.J. Yamé and D. Sauter, On control performance specifications in network-based control systems, in proc. Of 3ème workshop on Networked Control systems tolerant to fault. Juin 2007.
[Yavatakar et *al*. 1992] R. Yavatakar, P. Pai and R. Finkel, A reservation based CSMA Protocol for Integrated Manufacturing Networks, Technical report 216-92, Department of Computer Science, University of Kentucky, Octobre 1992.
[Zadeh 1973] L.A. Zadeh, Outline of a new approach to the analysis complex systems and decision processes. In Proc. Of IEEETransaction On Systems Man and Cybernatics (SMC-3), pp. 28-44, 1973.
[Zhang 2001] W. Zhang, Stability analysis of networked control systems, PhD. Dissertation, Department of Electrical Engineering and Computer Science, case Western Reserve University, Août 2001.
[Zhang et *al*. 2001] W. Zhang, M.S. Branicky and S.M. Phillips, Stability of networked control systems. IEEE Control Systems Magazine, February 2001.
[Zhao and Ramamritham 1986] W. Zhao et K. Ramamritham, A virtual time CSMA/CD protocol for hard real-time communication, In proc. Of the IEEE Real-Tim Systems Symposium, pp. 120-127, 1986.

Annexe 1

1. Le modèle de files d'attente

Dans ce modèle les mécanismes de gestion de congestion sont exigés, quand plusieurs ports « concourent » vers le même port dans le commutateur. Deux questions doivent être posées :
1- Les buffers utilisés dans le commutateur sont-ils dynamiques ou fixes?,
2- Où est ce que les buffers sont physiquement placés dans le commutateur?
L'architecture basée sur les buffers dynamiques est composée de buffers individuels divisés en petites parties. Cette structure permet une utilisation efficace des buffers car elle permet de mettre à disposition beaucoup plus de mémoires physiques, comparée à l'architecture basée sur des buffers fixes.
L'architecture basée sur les buffers fixes fige la taille de chaque buffer à la taille maximum de l'unité de transfert MTU (maximum transmission unit). Par exemple dans une telle architecture où la taille des buffers est fixée à 2000 octets, n'importe quelle trame de taille significativement inférieure à la taille du buffer en occupe un seul, ce qui résulte une perte en mémoire. Malgré que la construction d'un buffer fixe soit moins coûteuse qu'un buffer dynamique, l'architecture basée sur ce dernier reste plus efficace.

Les buffers sont généralement placés aux ports d'entrée du commutateur, c'est un mécanisme relativement simple mais inefficace à cause du phénomène du blocage de tête de ligne HOL (Head of Line Blocking). Le blocage de tête de ligne se produit lors du scénario suivant : Quand un port de sortie est congestionné, et qu'en même temps deux paquets arrivent à au même buffer d'entrée, le premier paquet doit être transféré au port congestionné et le deuxième doit être dirigé vers un autre port. Dans ce cas de figure le second paquet est dans l'obligation d'attendre la disponibilité du port de sortie du premier paquet afin qu'il soit transmis à son tour. De ce fait ce Phénomène réduit significativement le débit dans le commutateur.
Pour pallier ce problème une architecture de bufferisation alternative qui consiste à placer des buffers aux ports de sortie en prenant en compte le mécanisme de gestion de congestion.
Cette méthode est plus efficace que la première car elle permet d'éliminer le HOL cependant, elle peut causer un problème de débordement dans les buffers durant le pic de trafic.
Une troisième solution appelée files d'attentes basées sur la mémoire partagée place une mémoire centrale ce qui la rend accessible par tous les ports du commutateur.
Le mécanisme de la mémoire partagée permet d'éliminer le HOL et fournit un débit important avec peu de buffers que cela peut être nécessaire dans une structure à files d'attente qui associe chaque buffer à un port.
La dernière solution traite le problème de l'architecture à files d'attente multiple. Elles ont été conçues pour répondre problème de qualité de service.
Elle est basée sur le concept suivant : plusieurs files d'attente peuvent être associés à un port pour différencier entre les données critiques et non-critiques. Cette structure peut être implémentée soit dans les modèles à files d'attente de sortie ou dans les modèles à mémoire partagée, cependant ceci exige un mécanisme de gestion de congestion.

2. Implémentation de commutation

Un point important doit être considéré : Où et comment la décision de commutation doit être prise ?

La première méthode consiste en une commutation centralisée, basée sur une table d'adressage centrale contrôlé par un circuit ASIC (Application-Specific Integrated Circuit). Ce circuit doit regarder dans cette table d'adressage rapidement pour éviter le problème de goulot d'étranglement.
La seconde méthode est appelée commutation distribuée, chaque table d'adressage est localement située sur un port. Ces tables d'adressage doivent être synchronisées pour suivre les changements qui peuvent se produire.
Dans le cas ou la synchronisation avec une table « maître » n'est pas considérée, une des tables peut contenir une donnée « âgée » dont le port de sortie n'existe plus.

3. Fabrique de commutation

Il existe trois types de fabrique de commutation :
La première architecture est basée sur un bus qui représente une fabrique centrale utilisée par tous les ports pour communiquer. L'accès à ce bus est basé sur l'algorithme TDMA, cette architecture supporte le trafic de diffusion. Par contre le problème principal de cette architecture est que les buffers de sorties peuvent être congestionné lorsque beaucoup de paquets d'entrée doivent être transférés vers le même buffer de sortie.
En pratique quelques commutateur dans le commerce utilise une architecture mixte : l'architecture en bus et en grille (ou matrice), tel que les commutateur CISCO Catalyst 5500, et 6000 series (www.cisco.com) et les Allayer ROX series (www.allayer.com).
La seconde architecture est basée sur une grille (crossbar), elle consiste en une maille dans la fabrique de commutation qui fait la connexion entre les ports. L'intérêt principal de cette architecture est le grand nombre de port qu'elle supporte que ni le bus ou la mémoire partagée ne peut offrir. Un nombre important de ports signifie un débit et une bande passante importantes. Par contre son inconvénient survient quand la diffusion (le broadcast) ou le multicast et l'unicast se produisent simultanément. En effet, un trafic unicast ne peut être transmis quand un trafic multicast ou un trafic de diffusion est traité.
La dernière architecture est appelée mémoire partagée c'est la plus répondu car moins chère et efficace. La mémoire partagée est un moyen de commutation de paquets en utilisant les high-speed ASIC.
Les paquets arrivent dans l'architecture, ils sont placés en mémoire et mis dans les files d'attente des ports de sortie qui correspondent à leur destination. Cette bufferisation est interne de la fabrique de commutation dans ce type d'architecture et ces buffers peuvent être fixes ou dynamiques.

Quelques exemples de commutateur implémentant ces composants [Phipps 1999]

La série Catalyst 4000 de CISCO :
Cette série est un parfait exemple d'une architecture de commutation entièrement centralisée, basée sur une fabrique de commutation non bloquante de 24 Gb, l'implémentation de commutation et la bufferisation est centralisé par un ASIC.
Dans le Catalyst 4000, lorsqu'un paquet entre dans le commutateur, il est stocké en mémoire, le circuit ASIC cherche l'adresse MAC du paquet dans la table d'adressage, la trouve et envoie le paquet au port de sortie approprié.
En terme de bufferisation et de gestion de congestion, le Catalyst 4000 utilise une mémoire partagé de 8 MB et possède une table d'adressage centralisée, cette mémoire garde la trace des adresses Mac de la couche 2, cherche et met à jour tous les changements et est responsable de prise de décision dans le commutateur.
La série Catalyst 4003 possède 10/100 MB pour 96 ports et traite 14,3 millions de paquets par seconde (pps).

Annexe 2

La trame Ethernet

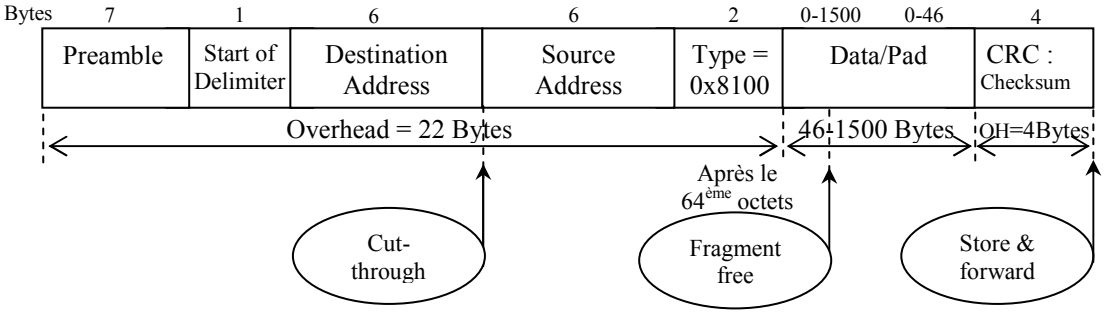


Figure 1. Format d'une trame Ethernet et les différents modes de transmission dans un commutateur.

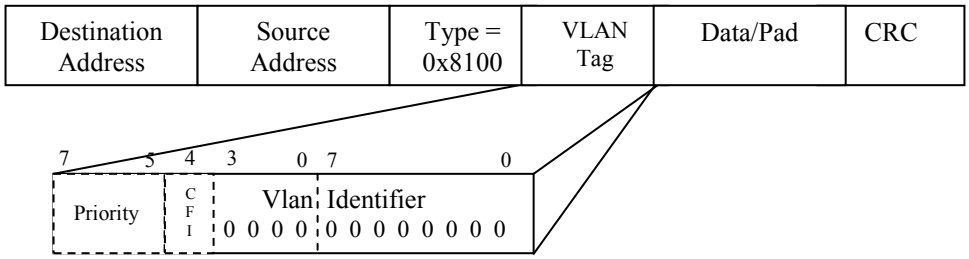


Figure 2. Format d'une trame Ethernet 802.1 P/Q (trame taggée)

Annexe 3

1. Eléments de bases des RDP :

Les réseaux de Petri présentent deux caractéristiques intéressantes. D'abord, ils permettent de modéliser et de visualiser les systèmes à événements discrets. Ensuite, en raison de leur simplicité mathématique, il reste un des outils privilégié de ces dernières décennies.

1. 1 Définition II.6: un RDP est un quintuplet $R = (P, T, \Pr e, Post, M_0)$ tel que :

- $P = \{P_1, P_2, ..., P_n\}$ est un ensemble fini et non vide de places ;
- $T = \{T_1, T_2, ..., T_m\}$ est un ensemble fini et non vide de transitions ;
- $P \cap T = \emptyset$, c'est à dire que les ensembles P et T sont disjoints ;
- $\Pr e$: $P \times T \to \mathrm{N}$ est l'application d'incidence avant (*i.e.* elle indique le nombre de jetons prélevés par les transitions dans les places amonts).
- $Post$: $P \times T \to \mathrm{N}$ est l'application d'incidence arrière (*i.e.* elle indique le nombre de jetons déposés par les transitions dans les places en aval).
- M_0 : le marquage initial.

Exemple :

La figure II.3 représente le cycle des saisons. A chaque saison est associée une place, et à chaque transition le passage d'une saison à la suivante.

Ce RDP n'a qu'une transition validée **T_1**, le franchissement de celle-ci fera correspondre le marquage suivant à l'été ce qui sensibilise la transition **T_2**, son franchissement fait évoluer le marquage du système.

- $P = \{P_1, P_2, P_3, P_4\}$;
- $T = \{T_1, T_2, T_3, T_4\}$;

$$- Post = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{cccc} T_1 & T_2 & T_3 & T_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

$$- \Pr e = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{cccc} T_1 & T_2 & T_3 & T_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$$- M_0 = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \\ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}.$$

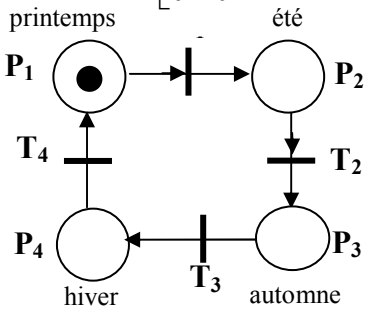


Figure 1. RDP représentant le cycle des saisons

On appelle matrice d'incidence, la matrice :

$$W = Post - \Pr e$$

Dans notre exemple la matrice d'incidence est égale :

$$W = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{cccc} T_1 & T_2 & T_3 & T_4 \end{array} \\ \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \end{array}.$$

Une colonne de cette matrice correspond à la modification du marquage apportée par le franchissement de la transition correspondante, par exemple la première colonne de *W*

indique que le franchissement de la transition $\mathbf{T_1}$ consiste à retirer une marque de $\mathbf{P_1}$ et à ajouter une marque dans la place $\mathbf{P_2}$.

Remarque
Un RDP peut être représenté par des automates et ceci en représentant son graphe de marquage qui peut être considéré comme l'automate correspondant.
Considérons le cas de notre exemple (*Cf.* figure II.4.) :

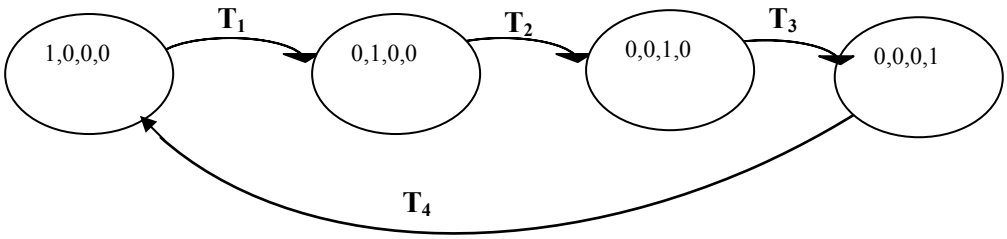


Figure 2. Automate du RDP représentant le cycle des saisons

1. 2 Réseaux de Petri temporisés, temporels :
Un réseau de Petri temporisé, temporel permet de décrire un système dont le fonctionnement dépend du temps.

Définition[42] II.7: Un RDP T- temporisé [Ramchandani 1974] est un couple $R_T = (R, \Theta)$ avec :
- R un RDP ;
- Θ : application qui associe à chaque transition une durée de franchissement.

Définition II.8: Un RDP T- temporel (Merlin 1974) est un couple $R_T = (R, I)$ avec :
- R un RDP ;
- I : application intervalle de tir qui associe à chaque transition son intervalle de tir.

L'analyse des RDP temporels se fait par le passage par la transformation des RDP temporels en automates temporisés.

[42] Il existe aussi des RDP P- temporisés dans lesquels les durées sont associées aux places.

Annexe 4

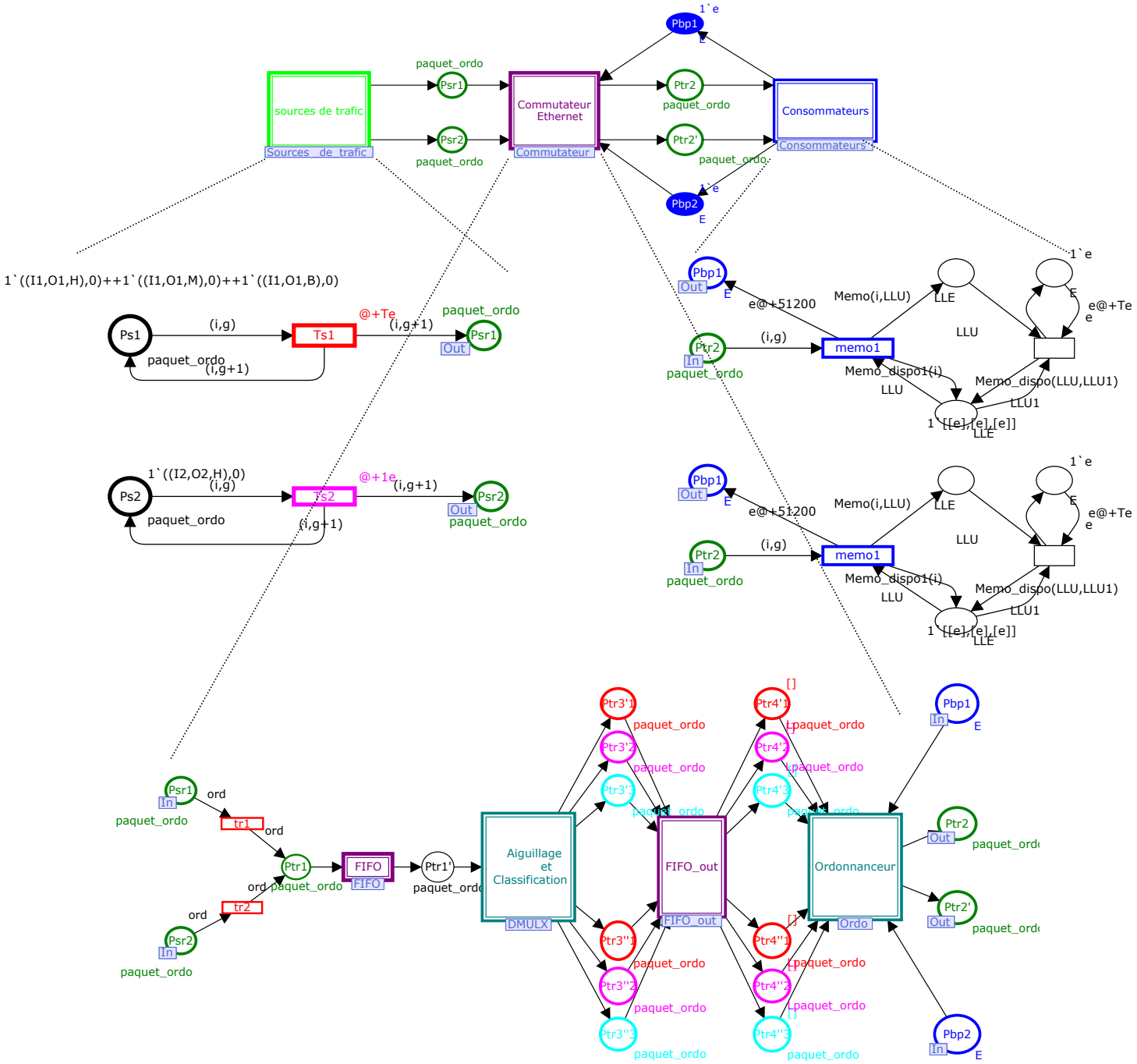


Figure 1. Modèle global du commutateur Ethernet sous un environnement producteur/consommateur périodique

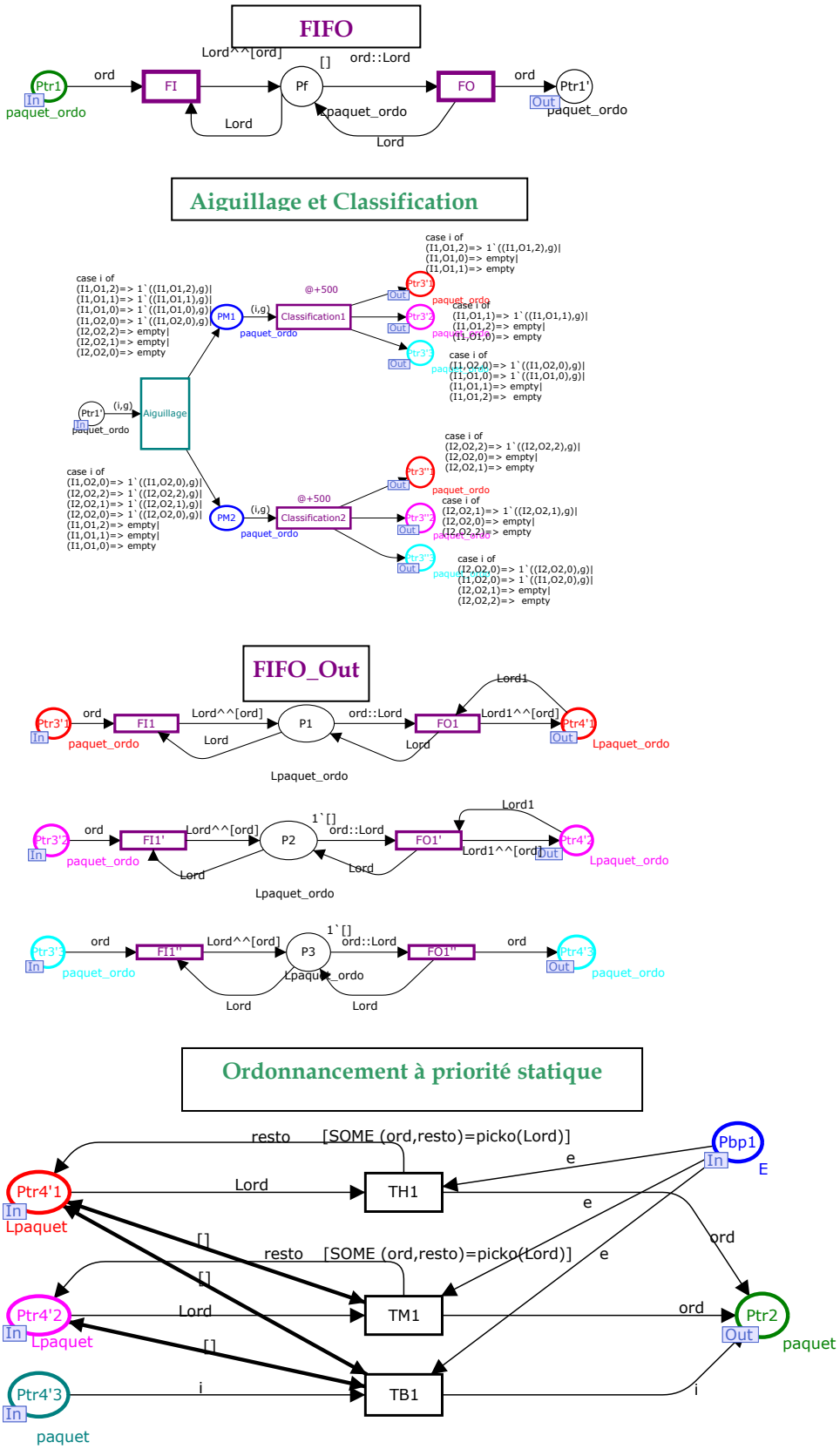


Figure 1. (suite…) Modèle global du commutateur Ethernet sous un environnement producteur/consommateur périodique

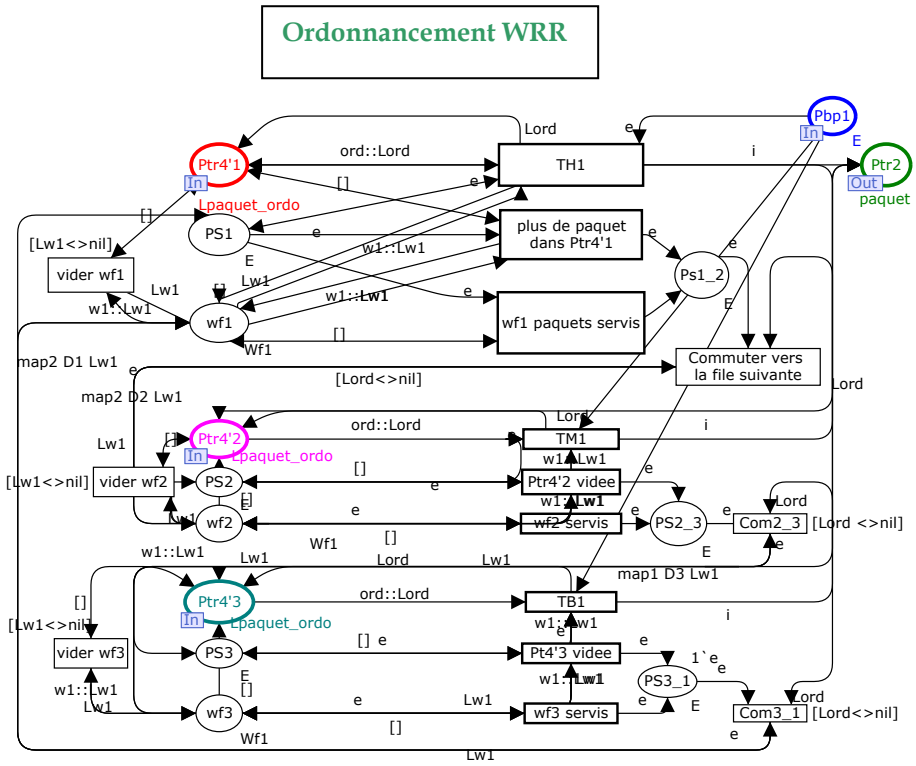
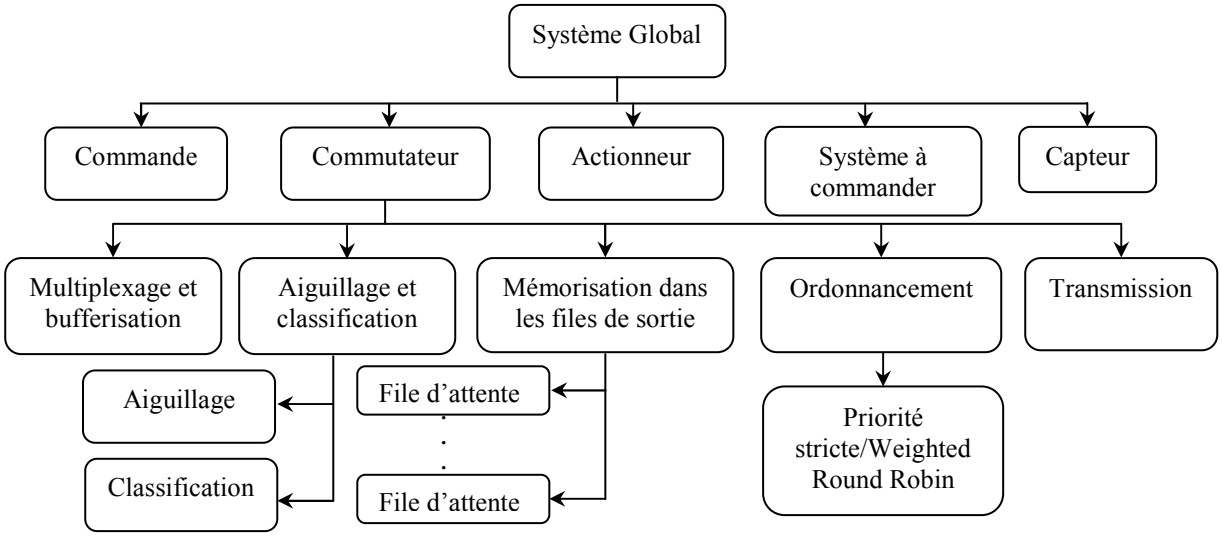


Figure 1. (…suite et fin) Modèle global du commutateur Ethernet sous un environnement producteur/consommateur périodique

Annexe 5


Système Global
Commande
Commutateur
Actionneur
Système à commander
Capteur
Multiplexage et bufferisation
Aiguillage et classification
Mémorisation dans les files de sortie
Ordonnancement
Transmission
Aiguillage
Classification
File d'attente
File d'attente
Priorité stricte/Weighted Round Robin

Résumé

L'étude des systèmes en réseau supports d'applications collaboratives, distribuées et interconnectées par un réseau repose sur l'identification des exigences de fonctionnement de l'application appelées Qualité de Contrôle (QdC), et sur l'évaluation des performances du réseau pour obtenir son niveau de Qualité de Service (QdS). « Cette thématique comporte d'importants verrous de nature fondamentale relevant du domaine de l'automatique, de la robotique, des capteurs, de la théorie de l'information, des réseaux. Par souci de simplification, les travaux sur les systèmes en réseau se repartissent selon deux approches: la première compense les perturbations générées par les communications au niveau de l'application (« control over network »). La seconde adapte les performances du réseau en fonction des besoins applicatifs (« control of network »). L'objectif de nos travaux de thèse est donc de proposer un environnement de modélisation intégré permettant de représenter le comportement des SCRs. Nous avons choisi les Réseaux de Petri de haut niveau qui possède un fort pouvoir d'expression, de formalisation et dont la modularité permet d'ajouter et/ou de faire évoluer les modèles qui sont développés dans ce travail. Dans un premier temps, nous avons proposé un modèle Ethernet Commuté gérant des mécanismes d'ordonnancement. Le choix de ce réseau a été guidé par le fait qu'il est de plus en plus utilisé dans les SCRs. Ensuite, le modèle d'un SCR a été proposé, et modélisé par des Réseaux de Petri de haut de niveau, en intégrant au modèle Ethernet Commuté, l'environnement applicatif : Contrôleur, Process,.. Enfin, des stratégies pour commander le réseau de façon à adapter sa Qualité de Service en regard de la Qualité de Contrôle requise par l'application, ont été mises en oeuvre. Pour cela, des ordonnanceurs à priorité stricte et de type WRR sont utilisés. Les résultats de simulation montrent clairement que des dispositifs de compensation du réseau pour améliorer les performances du système de communication, permettent aussi d'améliorer les performances du système à commander.

Mots clés : Système contrôlé en réseau (SCR), réseau de Petri de haut niveau (RDPHN), qualité de service (QdS), qualité de contrôle (QdC), contrôle du réseau, modélisation intégrée, ordonnanceur Weighted Round Robin, ordonnanceur à Priorité stricte.

Abstract

The Networked Control Systems (NCS) used in collaborative, and distributed applications are based both on identification of application requirements named Quality of Control (QoC), and on performance evaluation of network to obtain the required Quality of Service (QoS). The pluridisciplinary aspect of NCS needs the knowledge of many domains such as information theory, robotics, sensors and networks. In general, two approaches are investigated in NCS: the first one deals with the compensation of the network perturbations by using control theory (control over network). The second one adapts the network performances according to the application needs (control of network). The objective of this thesis is to propose an integrated modelling environment to represent globally the NCS behaviour by using the High Level Petri Nets (HLPN) formalism. This work considers NCS based on switched Ethernet architectures. Such communication architecture is modelled with HLPN and is evaluated according to different scheduling mechanisms and traffic load. After that, NCS model based on HLPN is proposed. This model integrates the Ethernet switch model and applicative environment of NCS: controller, process… Finally, the strategies to control the network in order to adapt the QoS according to the QoC required by the application are proposed. These strategies are achieved using priority static and Weighted Round Robin policies. The obtained results show that the scheduling mechanisms enable to improve the performance of communication system and then to improve the application performances.

Keywords: Networked control systems (NCS), high level Petri nets, quality of service (QoS), quality of control (QoC), control of network, co-design, Weighted Round Robin policy, static priority policy.