

FACULTE DES SCIENCES & TECHNOLOGIES

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 Gilbert HABIB

# **Qualité de service et qualité de contrôle d'un Système Discret Contrôlé en Réseau Sans Fil : proposition d'une approche de co-conception appliquée au standard IEEE 802.11**

Membres du jury :

Rapporteurs : M. Philippe LE PARC Professeur, LISyC, Université de Bretagne Occidentale

M. Armand TOGUYENI Professeur, LAGIS, Ecole centrale de Lille

Examineurs : M. Bruno DENIS Mcf, Lurpa, ENS Cachan

Mme Pascale MARANGE Mcf, CRAN, U.H.P, Nancy

M. Thierry DIVOUX Professeur, CRAN, U.H.P, Nancy (Directeur de thèse)

M. Jean-François PÉTIN Professeur, INPL, Nancy (co-directeur de thèse)



## Remerciements

Je tiens à remercier en tout premier lieu la France, qui m'a accueilli sur sa terre pour continuer mes études de Master. Ce pays m'a permis de continuer mes études en finançant mes trois années de thèse.

Les travaux de recherche présentés dans ce mémoire ont été développés au Centre de Recherche en Automatique de Nancy (CRAN, UMR 7039 du CNRS) dirigé par le Professeur Alain RICHARD que je remercie vivement pour son accueil au sein du laboratoire. Plus précisément, ma thèse a été réalisée au sein du groupe thématique SYMPA animé par le Professeur Thierry DIVOUX que j'ai connu en Master lors de mon premier séjour en France et auquel je tiens à exprimer ma profonde gratitude.

Je remercie M. Philippe LE PARC et M. Armand TOGUYENI de m'avoir fait l'honneur d'être rapporteurs de ces travaux de thèse. Je remercie également M. Bruno DENIS pour avoir accepté d'examiner mes travaux et Melle Pascale MARANGE pour l'intérêt qu'elle a porté à ces travaux en participant au jury de cette thèse mais aussi en partageant son expérience lors de travaux communs

Je tiens à remercier mes directeurs de thèse Professeur Thierry DIVOUX et Professeur Jean-François PETIN qui ont su me conseiller et me recadrer pendant ces trois ans. J'adresse également tous mes remerciements à l'ensemble des personnels du CRAN, IUT brabois et l'ESIAL, personnes et amis que j'ai pu rencontrer durant ces années de thèse pour leur soutien et leur bonne humeur. Je pense notamment à Leila MAKKAOUI, Asma YEHA, Jean Philipe GEORGES, Jeremy ROBERT, Cheick kone TIDJANE, Farida LOUTIS, Carlos et Isabelle HERRERA, Roger NAKAD, Roland et Maria EL HAJJ, Claude et Florence MAHFOUZ, Père Sayed MARROUN, Nicolas SALLES,...

Mes remerciements surtout vont aussi à ma fiancée Suzanne ABBAS qui, à sa manière, m'a énormément aidé, je lui exprime mon admiration pour son courage et sa patience et ma gratitude pour son soutien indéfectible.

Je dédie ce doctorat à ma mère et à mon père, dont les bénédictions m'ont suivi tout au long, et sans lesquelles je n'en serais pas là. Je leur exprime toute mon admiration, mon affection et ma gratitude.

Finalement, je tiens à remercier mes frères, Georges, Jawad et Jad qui m'ont toujours soutenu et encouragé dans les moments les plus difficiles.



## Sommaire

Introduction .....	11
Chapitre 1 Systèmes Discrets contrôlés en réseau sans fil.....	15
1.1 Introduction .....	16
1.2 NCS ou SCR.....	16
1.2.1 Structure hiérarchique :.....	16
1.2.2 Structure directe :.....	17
1.2.3 Problématique :.....	17
1.3 NDSCS ou SDCR .....	18
1.3.1 Partie Commande.....	20
1.3.2 Partie Opérative .....	20
1.3.3 Partie Réseau.....	20
1.4 SDCR sans fil .....	20
1.4.1 ZigBee :.....	21
1.4.2 ISA 100 :.....	21
1.4.3 Bluetooth :.....	22
1.4.4 WirelessHart :.....	22
1.4.5 IEEE 802.11 :.....	22
1.5 Etude de la norme 802.11 .....	23
1.5.1 IEEE 802.11: Couche MAC .....	25
1.5.1.1 Mode DCF : Distributed Coordination Function .....	25
1.5.1.2 Mode PCF : Point Coordination Function.....	28
1.5.1.3 Limitations des modes DCF et PCF .....	29

1.5.2 IEEE 801.11e: Couche MAC.....	30
1.5.2.1 Enhanced Distributed Channel Access, EDCA.....	31
1.5.2.2 HCF-Controlled Channel Access, HCCA.....	34
1.5.3 Conclusion et hypothèses.....	34
1.5.3.1 Hypothèses .....	34
1.5.3.2 Attributs retenus pour l'étude.....	35
1.6 Influence des attributs réseau sur un système contrôlé en réseau.....	36
1.7 Conclusion du chapitre .....	39
Chapitre 2 Etat de l'art sur la modélisation, l'évaluation et l'amélioration d'un système contrôlé en réseau.....	41
2.1 Introduction .....	42
2.2 Modélisation .....	42
2.2.1 Première catégorie, <i>orientée commande</i> :.....	43
2.2.2 Deuxième catégorie, <i>orientée réseau</i> .....	45
2.2.3 Troisième catégorie, <i>orientée commande-réseau</i> , .....	46
2.2.4 Approche par simulation.....	48
2.2.4.1 Outils logiciels dédiés à la modélisation et l'évaluation la commande.....	48
2.2.4.2 Outils logiciels dédiés à la modélisation et l'évaluation du réseau.....	48
2.2.4.3 Outils logiciels pour le co-design.....	51
2.3 Amélioration d'un système contrôlé en réseau.....	53
2.3.1 Première approche, <i>orientée QdC</i> .....	53
2.3.2 Deuxième approche, <i>orientée QdS</i> .....	54
2.3.2.1 Etudes du comportement d'EDCA.....	55
2.3.2.2 Proposer de nouveaux algorithmes pour le 802.11e pour améliorer sa performance :.....	56

2.3.3 Troisième approche, <i>orientée QdS- QdC</i> ,co-design: .....	57
2.4 Conclusion du chapitre .....	59
Chapitre 3 Outils pour modéliser un SDCR sans fil .....	61
3.1 Introduction : .....	62
3.2 Outil orienté « commande » : Matlab-TrueTime .....	63
3.3 Outil orienté « réseau » : OPNET.....	68
3.3.1 Modélisation sur OPNET.....	68
3.3.2 Simulation sur OPNET .....	70
3.4 Comparaison OPNET et <i>TrueTime modifiée</i> .....	71
3.4.1 Première partie : Comparaison Truetime, <i>TrueTime modifiée</i> , OPNET.....	72
3.4.2 Deuxième partie : Comparaison <i>TrueTime modifiée</i> , OPNET sur un SDCR sans fil.....	75
3.4.2.1 Modèles génériques d'un SDCR sans fil.....	75
Modèle de commande .....	75
Modèle de la partie opérative .....	77
Synchronisation entre les modèles d'environnement de la PC/PO.....	78
Modèle du réseau .....	78
3.4.2.2 Cas d'étude.....	80
3.4.2.3 Instanciation des modèles génériques sur un cas d'étude .....	80
3.4.2.4 Implantation du cas d'étude sous OPNET et <i>TrueTime modifiée</i> .....	84
3.4.2.5 Comparaison des résultats sur OPNET et <i>TrueTime modifiée</i> .....	88
3.5 Conclusion du chapitre .....	90
Chapitre 4 Optimisation des attributs et proposition d'un algorithme afin d'améliorer la performance d'un SDCR sans fil .....	91
4.1 Introduction : .....	92

4.2 Optimisation des valeurs des attributs d'un réseau de communication sans fil dans un SDCR sans fil .....	93
4.2.1 Etudier/optimiser l'influence des attributs réseau sur le comportement du réseau sans fil, (Habib, et al., 2009).....	93
4.2.2 Estimation analytique des délais de bout en bout dans le pire cas.....	96
4.2.2.1 Estimation analytique des délais maximum et minimum de bout en bout dans un réseau sans fil, IEEE 802.11e mode EDCA .....	98
Durée de transmission d'un paquet au niveau couche physique.....	98
Calcul du délai minimal .....	100
Calcul du délai maximal.....	100
Application du calcul analytique sur un cas d'étude.....	101
4.2.2.2 Calcul de $ResT$ maximal et minimal .....	104
4.2.2.3 Application des résultats analytiques sur un cas d'étude .....	105
4.2.2.4 Validation des résultats par des simulations .....	106
4.2.3 Interpretation des résultats et optimisation des attributs.....	108
4.2.4 Conclusion de la première partie: .....	110
4.3 Proposition d'un algorithme afin d'améliorer la performance d'un SDCR sans fil.....	110
4.3.1 Algorithme proposé .....	111
4.3.2 Cas d'étude.....	115
4.3.2.1 Modélisation du système étudié sur OPNET .....	117
4.3.2.2 Modélisation de la Partie Opérative .....	117
4.3.2.3 Modélisation du réseau.....	118
4.3.2.4 Modélisation du commande .....	118
4.3.2.5 Implémentation de l'algorithme proposé .....	121
4.3.2.6 Précautions .....	123
4.3.2.7 Simulation et analyse des résultats.....	124

4.4 Conclusion du chapitre .....	128
Conclusion et perspectives .....	129
Bibliographie .....	133



# Introduction

Depuis quelques années, l'étude des systèmes de commande contrôlés en réseau montre que l'intégration du réseau entre les parties de ces systèmes génère des perturbations en termes de retard, de pertes de données, ... Ces effets peuvent influencer la performance du système mesurée en termes de stabilité, robustesse, adaptabilité, déterminisme. Notons bien que ces effets dépendent du type du réseau intégré. Nous pouvons citer des réseaux déterministes comme CAN, Profibus qui répondent exactement aux besoins des systèmes commandés en offrant une garantie en termes de temps de réponse. Par contre, le développement de ces réseaux est freiné par plusieurs handicaps comme le coût des équipements qui implémentent ces protocoles, leur flexibilité, ... Un autre type de réseau résout ces handicaps et commence à être de plus en plus répandu dans les systèmes contrôlés, c'est le réseau Ethernet. Au contraire des réseaux précédents, ce type de réseau est non déterministe mais des études montrent qu'il peut vérifier néanmoins les contraintes applicatives sous certaines conditions d'usage.

Ce travail s'inscrit dans un cas particulier de la famille des systèmes contrôlé en réseau: les systèmes discrets contrôlés en réseau sans fil ou SDCR sans fil. Ces systèmes sont caractérisés par une commande logique distribuée sur des automates programmables industriels (API) et/ou des boîtiers d'entrées/sorties déportés et par une partie opérative dont le comportement continu est décrit par des états discrets. Le réseau intégré dans ce système est un réseau sans fil, 802.11 (Wifi). Ce type de réseau est plus sensible au bruit et aux interférences, les retards éventuellement importants, et des pertes de paquets et de la gigue (variation du délai) sont possibles. La performance de ce type de système est évaluée en termes de temps de réponse, déterminisme et sûreté. Cette performance dépend des attributs manipulables des différentes parties d'un SDCR sans fil.

L'objectif de cette thèse est de contribuer à la résolution de deux grands problèmes :

- Le problème de modélisation : qui se pose puisqu'un SDCR sans fil combine deux parties de natures différentes : la partie réseau est de nature non déterministe et la partie commande/opérative est de nature déterministe. Donc le problème est de trouver le meilleur outil qui peut modéliser au mieux le comportement d'un SDCR sans fil en prenant en compte les différents effets et les attributs des parties de ce système.
- Le problème d'amélioration des performances du système : afin de résoudre ce problème, nous proposons un algorithme d'adaptation dynamique des attributs de la partie réseau en fonction de l'état/besoins de la partie commande et du comportement de la partie réseau. L'environnement de simulation résultant de notre contribution au premier problème nous permettra d'évaluer l'efficacité de cet algorithme.

Ce rapport de thèse comporte quatre chapitres dans lesquels nous présentons la problématique scientifique, puis nous exposons nos contributions avant de les appliquer sur un cas d'étude dans le but de valider nos propositions.

**Le premier chapitre** situe le contexte de la thèse et introduit les systèmes contrôlés en réseau. Puis, nous réduisons progressivement le périmètre de notre étude vers les systèmes discrets contrôlés en réseau et finalement, les systèmes discrets contrôlés en réseau sans fil. Dans chaque étape, nous exploitons leurs spécificités, leurs caractéristiques, l'avantage de l'utilisation du réseau et les différents types d'accès au medium. Nous décrivons également les différentes parties qui forment un SDCR sans fil en précisant les attributs gérables. En plus, un état de l'art est fait sur l'influence des différents attributs sur la performance du système contrôlé en réseau.

**Le deuxième chapitre** décrit le problème de la modélisation d'un SDCR sans fil et les choix possibles afin de représenter ce système. Le choix sera orienté commande-réseau c'est à dire que nous modéliserons les différentes parties de ce système sans négliger aucun attribut d'aucune de ses parties. Dans le but d'évaluer nos modèles, trois types d'approches sont disponibles : expérimentale, analytique ou par simulation. Finalement, notre choix dans ce chapitre sera l'approche par simulation. Dans cette approche, le problème se pose au niveau du choix de l'outil, d'autant que ce type de système combine des parties de nature différentes (déterministe/non déterministe). Le choix porte sur deux outils, Matlab (considéré comme outil orienté commande, avec modélisation grossière du réseau, Truetime) et OPNET (considéré comme outil orienté réseau, a priori non adapté pour la modélisation de la commande). Le chapitre 2 présente aussi les différentes solutions existantes pour améliorer les performances des systèmes contrôlés en réseau. On peut voir des solutions présentées par la communauté automatique afin d'accommoder la commande aux variations du comportement du réseau (*orientées Qualité de Commande, QdC*), les solutions proposées par les spécialistes en réseau pour assurer un niveau de QoS du réseau (*orientées Qualité de service, QoS*), et finalement, des solutions qui prennent en compte de façon simultanée la qualité de service du réseau et la performance du système commandé (*orientées Qualité de commande/service, QdC-QoS* ou co-conception).

**Dans le troisième chapitre**, nous exploitons ces deux outils et présentons les modifications apportées à la librairie Truetime de MATLAB pour qu'elle représente mieux la partie réseau. Dans ce chapitre, les modèles dans OPNET sont également étudiés dans le but de voir la possibilité d'implémenter la partie commande/opérative dans cet outil. Des comparaisons sont ensuite faites au niveau réseau et sur un cas d'étude qui représente un SDCR sans fil. La convergence des résultats nous a assuré du bon comportement de ces outils, ainsi que de l'exactitude des modifications faites sur MATLAB-Truetime ainsi que l'implémentation de des différentes parties d'un SDCR sans fil sur OPNET.

**Dans le quatrième chapitre**, nous allons étudier/améliorer la performance d'un SDCR sans fil. Ce chapitre est divisé en deux grandes parties :

- Dans la première partie, nous nous intéressons à optimiser le choix des attributs du réseau. Afin d'atteindre ce but, nous allons analyser le comportement du système dans les pires des cas puis déduire dans ces cas les meilleurs attributs qui garantissent un

niveau de performance acceptable. L'approche par simulation peut ne pas couvrir des situations où le système se trouve dans des pires cas. Par conséquent, nous focalisons sur l'approche analytique. Les calculs analytiques seront établis dans cette partie pour optimiser le choix des attributs du réseau. Ces calculs sont appliqués sur un cas d'étude.

- Dans la deuxième partie, nous nous situons dans des cas où le réseau est chargé. Dans ces cas, le choix des attributs ne sera pas une solution suffisante pour avoir une performance acceptable du système. D'où l'idée de proposer un algorithme d'amélioration dynamique de la performance. Cet algorithme est basé sur l'utilisation du standard IEEE 802.11e, extension de 802.11, qui permet une QoS différenciée entre les flux. L'algorithme alloue dynamiquement les priorités sur les trafics de communication en fonction des états de la commande et de la QoS courante offerte par le réseau. Dans le but de prouver l'efficacité de cet algorithme, une application académique est mise en place. Les simulations montrent une amélioration de la performance du système.

Enfin nous concluons et énonçons les perspectives que suggère notre travail.



tel-00544932, version 1 - 9 Dec 2010

# Chapitre 1

## **Systemes Discrets contrôlés en réseau sans fil**

## 1.1 Introduction

Une tendance importante dans les systèmes industriels et commerciaux modernes est d'intégrer un réseau de communication entre les différents niveaux du système contrôlé d'où son nom « système contrôlé en réseau » (SCR) ou NCS (*Networked Control System*). Cette intégration peut influencer sur la performance du système en termes de stabilité, adaptabilité,.... Dans notre étude, les variables du système de commande appartiennent à un espace d'état discret et évoluent dans des instants discrets du temps. Dans ce type de système nommé SDCR (système discret contrôlé en réseau), la performance est évaluée en termes de temps de réponse, déterminisme et sûreté. De plus, le réseau intégré dans notre étude est un réseau sans fil (IEEE 802.11) au vu des avantages qu'il fournit comme la facilité et le coût de câblage ... d'où son nom SDCR sans fil ou WNDCS (*Wireless Networked Discret Control System*). Par contre, ce type de réseau est plus sensible au bruit et aux effets externes d'où un comportement plus fragile par rapport à d'autres types de réseaux.

Aussi, nous exploiterons une extension d'IEEE 802.11, IEEE 802.11e, qui pourra être une bonne alternative afin d'améliorer la performance d'un SDCR sans fil. Dans ce chapitre, nous étudions également l'influence des attributs réseau sur le système global.

## 1.2 NCS ou SCR

Les systèmes commandés en réseau sont des systèmes automatiques formés de capteurs, contrôleurs, actionneurs. Ces équipements sont distribués autour d'un médium de communication. Les capteurs, contrôleurs et actionneurs constituent avec les systèmes de communication les éléments d'un SCR. Le médium de communication sert à établir un lien de communication pour échanger les données entre les équipements de ce système. De plus, il réduit les coûts de câblage par rapport à une connexion point à point (topologie maillée), et facilite la maintenance. Les capteurs, actionneurs et les contrôleurs peuvent être reliés entre eux de différentes manières. Par exemple, deux configurations sont identifiées par (Chow, et al., 2001): la structure directe et la structure hiérarchique (d'autres structures sont possibles):

### 1.2.1 Structure hiérarchique :

Cette structure est formée d'un contrôleur principal et de sous-systèmes en boucle fermée. Chaque boucle est formée d'un contrôleur local (C), de capteurs (S), d'actionneurs (A) et d'un système physique à réguler (process) (Figure 1.1, a)). Périodiquement, le contrôleur principal calcule et envoie la consigne dans une trame via le réseau au sous-système. Ce dernier traite la consigne pour effectuer localement la commande en boucle fermée et renvoie la mesure des capteurs au contrôleur principal. Le contrôleur local doit satisfaire les performances de la boucle fermée avant de recevoir le message du contrôleur principal.

La structure hiérarchique peut éventuellement se décliner sur plusieurs niveaux.

### 1.2.2 Structure directe :

Elle se compose d'un contrôleur principal et de capteurs et d'actionneurs (Figure 1.1, b)). Dans ce type de structure, pas de contrôleur local, les capteurs et les actionneurs sont reliés directement au contrôleur principal à travers le réseau. En d'autres termes, les données sont directement échangées entre les capteurs/actionneurs et le contrôleur principal sans un équipement intermédiaire.

Chaque structure présente des avantages par rapport à l'autre. La structure hiérarchique est plus simple à configurer puisque les fonctions sont distribuées sur les contrôleurs (principaux ou locaux). De plus, cette structure n'est pas trop sensible aux effets du réseau par rapport à la structure directe puisque les informations ne sont pas toutes envoyées via le réseau. Par contre, la structure directe a une meilleure interaction entre les équipements (capteurs, actionneurs) et le contrôleur principal puisque les données sont directement échangées sans passer par un contrôleur intermédiaire. L'interaction est nécessaire dans un SCR surtout pour certains types d'informations comme par exemple un signal d'alarme.

L'utilisation de l'une de ces deux structures dépend des conditions d'application et des souhaits du concepteur. Par exemple, un robot manipulateur utilise plusieurs moteurs qui fonctionnent simultanément ensemble. Il peut être plus commode et plus robuste d'employer un contrôleur existant et de formuler le problème dans une structure hiérarchique. Cependant, un concepteur peut avoir besoin d'un système de commande de vitesse de moteur pour avoir une réponse plus rapide de commande au-dessus du réseau. La structure directe peut être préférée dans ce cas-ci. Nous utilisons dans notre travail la structure directe, et nous nous limitons à une commande centralisée implantée sur un API unique couplé à des boîtiers E/S déportés sans traitement.

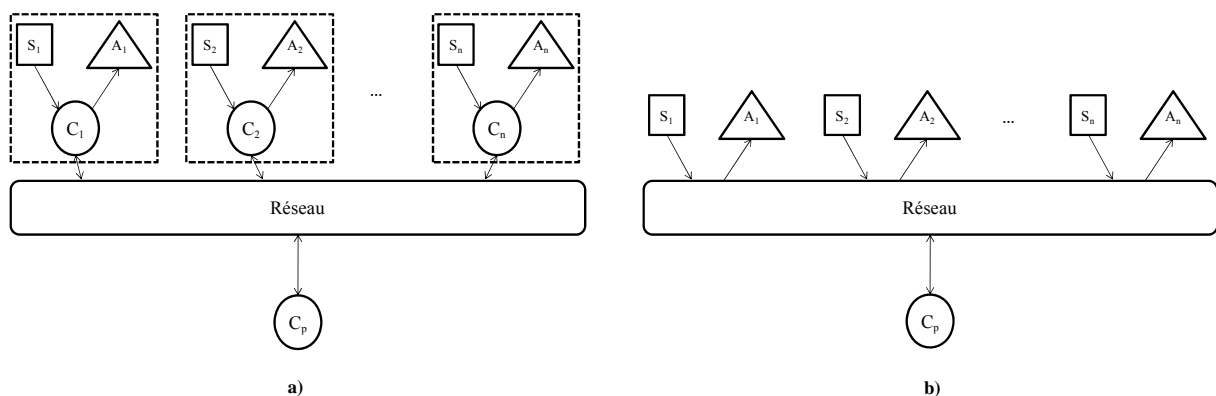


Figure 1.1 : Structures a) hiérarchique, b) directe

### 1.2.3 Problématique :

L'étude des SCR repose sur l'identification des exigences de fonctionnement de l'application appelées Qualité de Contrôle (QdC) définie en termes de stabilité, temps de réponse,

déterminisme et sûreté, et sur l'évaluation des performances du réseau pour obtenir son niveau de Qualité de Service (QoS) définie en termes de bande passante, débit, délais de transmission, gigue, taux de perte de paquets....

L'intégration du réseau introduit des effets comme les délais, les pertes de paquets et la variation de délai appelé la gigue. Ces effets peuvent perturber la performance du système évaluée par la stabilité, adaptabilité, robustesse,....

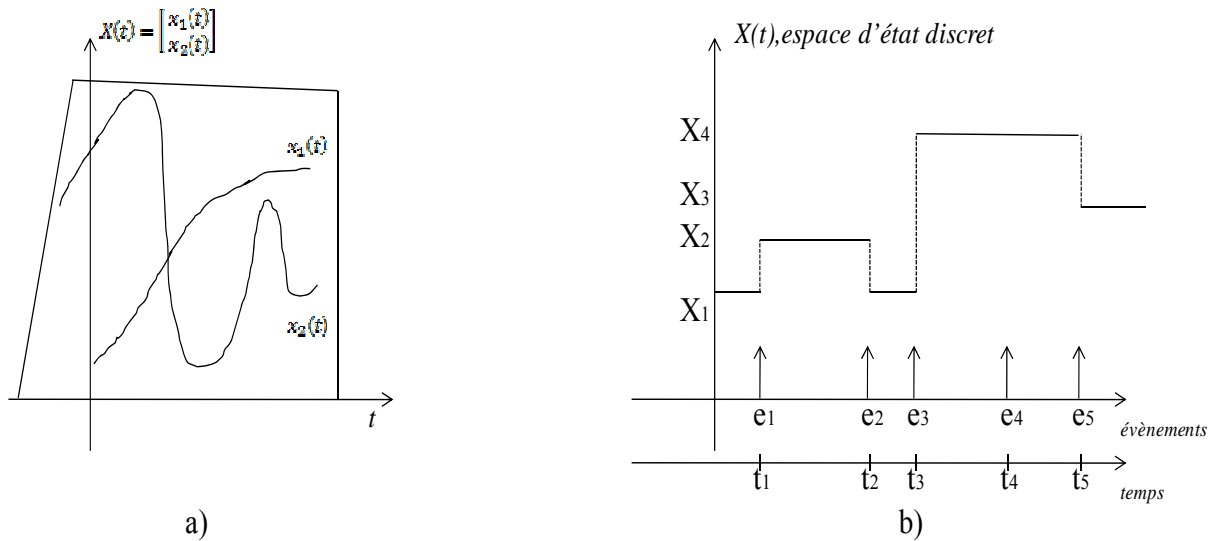
Les effets du réseau dépendent de plusieurs de ses caractéristiques comme par exemple le protocole d'accès au médium, la largeur de la bande passante, ... Plusieurs protocoles ont été proposés pour répondre aux besoins des systèmes automatiques commandés en réseau. Nous pouvons citer CAN, Profibus, WorldFIP. Ces protocoles fournissent une garantie en termes de temps de réponse, déterminisme. Cependant, le coût, le manque d'interopérabilité et de flexibilité des équipements qui implémentent ces protocoles constituent des inconvénients réels dans l'étude et le développement des systèmes industriels. Ces inconvénients sont absents dans le protocole Ethernet vu son coût réduit et sa facilité d'utilisation. Par conséquent, Ethernet est une solution adoptée de plus en plus souvent pour la communication d'un SCR. Ce protocole semble peu adapté aux applications industrielles, du fait de son non déterministe puisqu'il utilise l'algorithme CSMA/CD (*Carrier Sense Multiple Access/ Collision Detection*) pour accéder au réseau. Il convient de vérifier a priori qu'il satisfait néanmoins les contraintes applicatives.

### 1.3 NDCS ou SDCR

Dans un SCR, il existe des variables échangées entre les équipements de ce système. Ces variables représentent un comportement (état) significatif du système, par exemple la vitesse, la pression, la position. Les valeurs de ces variables sont de nature continue dans le sens qu'elles peuvent prendre n'importe quelle valeur dans l'ensemble  $\mathbb{R}$  lorsque le temps, de nature continue, évolue. Le comportement de ce système à l'instant  $t$  peut être résumé par son état qui représente la mémoire minimale du passé nécessaire à la détermination du futur. Par suite, l'espace d'état est un continuum composé de tous les vecteurs à  $n$  dimensions dans  $\mathbb{R}$  (Figure 1.2, a)). Notons finalement que les équations différentielles sont alors l'outil de base principal pour la modélisation, l'analyse et la commande de ces systèmes.

Par contre, d'autres types de système ont des variables d'état de nature discrète : elles prennent des valeurs discrètes spécifiées dans un ensemble bien défini. Par exemple : {On, Off} ou {Vert, Rouge, Bleu}. De plus, ces variables changent de valeurs seulement à des instants du temps. Ces instants correspondent à des événements asynchrones. D'où leur nom de systèmes à événements discrets. De façon informelle, les systèmes à événements discrets peuvent être définis comme des systèmes dans lesquels les variables d'état changent sous l'occurrence d'événements. L'espace d'état est un ensemble discret et l'état change seulement à certains instants du temps, d'une manière instantanée (Figure 1.2, b)). Ils ne peuvent généralement pas être décrits, à l'instar des systèmes continus classiques, par des équations

différentielles en raison de la nature des phénomènes qui entrent en jeu, notamment des phénomènes de synchronisation ou d'exclusion mutuelle.



**Figure 1.2 : Espaces d'état pour a) un système continu, b) système à événements discrets**

Ces systèmes sont alors souvent représentés par des modèles états-transitions. Les plus connus sont les automates à états finis qui servent pour représenter les systèmes déterministes les plus simples, les chaînes de Markov pour leurs analogues stochastiques et les réseaux de Petri pour des systèmes plus complexes qui comportent à la fois des phénomènes de synchronisation, de concurrence et de parallélisme. Les SCR dont le comportement est décrit par un système à événements discrets, seront nommés SDCR. La performance de ces systèmes sera évaluée par rapport à certaines propriétés comme le déterminisme, la réactivité. La réactivité, c'est la capacité pour qu'un système réagisse le plus vite aux événements venant de son environnement. Un système est supposé déterministe lorsqu'une même variation des événements d'entrées provoque une même variation des événements de sorties.

Dans les deux systèmes SCR et SDCR, les variables sont échantillonnées avant d'être envoyées au réseau. Un problème se pose donc au niveau du choix de la période d'échantillonnage. D'après Nyquist-Shannon, la fréquence d'échantillonnage doit être au moins deux fois plus grande que la fréquence maximale contenue dans le signal d'origine, afin de pouvoir le reconstruire au niveau récepteur. Pour les SDCR, le problème est plus compliqué. (Caspi, 2003) exploite un exemple d'un système à événements discrets échantillonné avec deux périodes différentes. Il montre que dans ses deux cas, les valeurs obtenues par échantillonnages peuvent être différentes ce qui nous amène à une situation indéterministe.

Un SDCR est formé de trois grandes parties: la Partie Commande (PC), Partie Opérative (PO) et le réseau. Dans la suite, nous allons détailler chaque partie.

### 1.3.1 Partie Commande

Dans les SCR et SDCR, cette partie est appelée partie contrôle ou contrôleur. Dans un SDCR, elle est un ensemble de lois de commande discrètes pilotant une partie opérative dont les entrées et les sorties sont constituées de signaux booléens. L'implantation des lois de commande est supportée par un (structure directe) ou plusieurs (structure hiérarchique) Automates Programmables Industriels (API). L'API est formé d'un processeur (s), de mémoires et de cartes d'entrées/sorties E/S. Les mémoires dans ce type de systèmes servent à sauvegarder le programme codé et les variables intermédiaires. L'API lit les entrées (lecture). Le(s) processeur(s) exécute (ent) les fonctions demandées (traitement) et finalement l'API met à jour les variables de sortie (écriture). Ces trois étapes seront répétées d'une manière périodique, avec une période appelé *Papi*.

L'attribut *Papi* joue un rôle essentiel sur la réactivité de l'API. Plus cette valeur est petite, plus l'API sera à l'écoute des événements produits par son environnement, et donc apte à y réagir promptement.

### 1.3.2 Partie Opérative

Elle est composée des capteurs et des actionneurs. Les capteurs envoient des informations pour la partie commande et les actionneurs reçoivent des ordres à travers le réseau. Les informations échangées sont de nature discrète.

### 1.3.3 Partie Réseau

Le réseau de communication sert à échanger les informations entre les différents équipements du système (capteurs, actionneurs ou PC). Les protocoles nécessaires à la gestion de la communication et à l'interconnexion en réseau des systèmes ouverts sont décrits par un modèle en couche proposé par l'ISO, d'où son nom, modèle OSI (*Open Systems Interconnection*). Ce modèle est constitué de sept couches : Physique, MAC, ... jusqu'à la couche Application. Ces couches permettent de gérer respectivement le comportement physique du médium, le protocole d'accès, ... les services applicatifs de communication. Selon les choix effectués aux niveaux de ces sept couches, plusieurs types de réseaux peuvent être employés. Dans le domaine des SDCR, nous rencontrons principalement deux types de réseaux : les réseaux de terrain limités à 3 couches et déterministes (exemple : Profibus), et de plus en plus des réseaux de type Ethernet.

## 1.4 SDCR sans fil

Dans un SDCR ou un SCR, le réseau entre les différents équipements est généralement un réseau filaire mais le coût des câbles, leur maintenance et l'installation sont des handicaps. Pour ces raisons, l'usage de la technologie radio est souvent envisagé. Deux types de structure peuvent être mis en œuvre : réseau en mode ad hoc dont les équipements communiquent directement entre eux sans passer par un intermédiaire, réseau en mode infrastructure pour

lequel les informations doivent nécessairement passer par un intermédiaire appelé point d'accès qui se charge de les acheminer. Notre étude portera sur un Système Discret Contrôlé en Réseau (SDCR) sans fil ou WNDCS en mode infrastructure.

L'intégration de ce type de technologie en milieu industriel a en réalité été envisagée de manière relativement ancienne puisqu'un des premiers articles (Lessard, et al., 1988) sur le sujet est daté de 1988. Cependant, comme nous l'avons déjà évoqué, les principales limites sont relatives aux interférences, au bruit induit et à la faible bande passante comparée à celle des technologies filaires. Pour palier ce problème, plusieurs auteurs (Ferreira, et al., 2002), (Alves, et al., 2002) ont proposé une architecture hybride entre réseau filaire et sans fil permettant de commuter de l'un à l'autre en cas de qualité de service non satisfaisante. Même si cette solution présente des avantages incontestables, elle reste néanmoins complexe à mettre en œuvre pour gérer l'interconnexion entre ces deux réseaux.

D'un point de vue industriel, plusieurs standards sont actuellement disponibles. Les paragraphes suivants les présentent brièvement et les comparent selon plusieurs critères : débit maximal, portée, disponibilité. Les résultats de cette comparaison sont présentés dans le Tableau 1.

#### **1.4.1 ZigBee :**

ZigBee est une technologie sans fil, caractérisée par sa consommation basse d'énergie et la communication de petites radios pour des réseaux à dimension personnelle (Wireless Personal Area Networks : WPANs). Cette technologie est basée sur la norme IEEE 802.15.4. Zigbee permet d'atteindre un débit maximal de 250 Kbit/s. Les travaux de (Salles, et al., 2007) évaluent les performances temporelles de cette technologie et la possibilité de l'adopter dans des applications à contraintes de temps réel fortes. Notons qu'une alliance existe entre les sociétés industrielles (ZigBee alliance<sup>1</sup>), dont le rôle est de développer cette technologie et fabriquer de nouveaux produits/applications basés sur cette technologie.

#### **1.4.2 ISA 100 :**

L'organisation ISA<sup>2</sup> "The Instrumentation, Systems and Automation Society" a pour but de développer des protocoles spécifiques au milieu industriel, et en particulier, de nouvelles solutions de réseaux sans fil dans l'industrie (principes, performances, compétitivité, sécurité, applications etc.). Nous pouvons citer le dernier standard ISA 100a apparu en 2009.

---

<sup>1</sup> [www.zigbee.org](http://www.zigbee.org)

<sup>2</sup> [www.isa.org](http://www.isa.org)

### 1.4.3 Bluetooth :

Le but initial de Bluetooth était de proposer une norme universelle pour les communications sans fil, plus performante et plus globale que les liaisons infrarouges (IrDA en particulier), déjà très répandues au début des années 90. Cette norme est caractérisée par sa courte portée et sa forte consommation d'énergie. De plus, elle impose une limitation du nombre maximal des stations dans un réseau Bluetooth. Le débit maximal autorisé est de 20 Mbits/s (dépend essentiellement des versions de ce type de réseau).

### 1.4.4 WirelessHart :

WirelessHART est un protocole de communication de réseau sans fil destiné aux applications dans l'automatisation de processus. Ce standard ajoute une capacité sans fil au protocole HART<sup>3</sup> compatible avec les appareils, commandes et outils HART existants. Le WirelessHART comprend différentes fonctions qui assurent une communication fiable dans des environnements d'entreprise présentant une infrastructure fortement occupée dans laquelle de grands véhicules ou appareils se déplacent, des conditions en perpétuel changement ainsi que différentes sources de radiofréquence et des interférences électromagnétiques pouvant générer des problèmes.

### 1.4.5 IEEE 802.11 :

La norme IEEE 802.11 est un standard international décrivant les caractéristiques d'un réseau local sans fil (WLAN). Cette technologie est connue sous le nom de Wifi. Parmi les divers types de réseaux sans fil cités avant, Wifi est choisi dans cette étude pour servir à la communication dans le SDCR sans fil pour ses avantages par rapport à d'autres technologies sans fil :

- **Le débit** : dans le Wifi, le débit maximal fourni peut atteindre 54 Mbit/s selon la couche physique utilisée (les différentes possibilités 802.11a, b, g seront présentées au paragraphe 1.5). Le débit important (par rapport aux technologies précédentes) du Wifi est un atout pour transmettre les données des applications multimédia ou temps réel, qui demandent un haut niveau de qualité de service même si cela n'est pas suffisant, en particulier en cas de gigue ou de latence.
- **La portée**, Dans le milieu industriel, les machines peuvent être éloignées l'une de l'autre. Donc la transmission des données depuis l'émetteur vers le récepteur peut passer une distance considérable. Pour cela, la portée peut être une contrainte essentielle de la technologie choisie. Les portées des technologies sans fil dépendent de la puissance d'émission. Plus la puissance est grande, plus la distance entre la source et le destinataire est potentiellement grande. Or dans le Wifi, la puissance

---

<sup>3</sup> [www.hartcomm.org](http://www.hartcomm.org)

maximale autorisée est plus grande que dans d'autres types de technologie, ce qui favorise une portée plus importante. Pour le Wifi, la transmission peut atteindre une distance de 500 mètres. Les autres technologies ont une portée de 100m mais avec une consommation d'énergie plus conséquente, c'est le cas du Bluetooth.

- **Populaire**, Wifi est devenu une technologie publique par rapport à d'autres technologies sans fil. Les *Hotspot* sont populaires dans tous les cafés, hôtels,... Presque tous les PC portables sont équipés de cartes Wifi. La demande pour cette technologie motive les chercheurs à développer des travaux autour de ce standard. De plus, Cette demande les encourage à améliorer cette technologie pour qu'elle s'adapte à tous les milieux où elle se trouve et aux types de données transmises. Plusieurs améliorations sont standardisées : exemple le 802.11e, qui est le 802.11 avec priorité entre les flux.
- **Coût**, l'augmentation de la demande, pousse la fabrication des équipements qui implémentent cette technologie. Cette augmentation diminue donc le coût de fabrication de ces équipements. Exemple : on peut trouver facilement dans le marché une carte Wifi à quelques euros.

Finalement, le Tableau 1 présente les différentes caractéristiques des standards cités ci-avant.

	ZigBee	ISA 100	Bluetooth	WirelessHart	IEEE 802.11
<b>Débit Maximal</b>	20 à 250 kbit/s	20 à 250 kbit/s	1 à 20 Mbit/s	250Kbit/s	54 Mbit/s
<b>Portée</b>	≤ 10 m	≤ 10 m	[10m, 100 m]	≤ 10 m	≤ 500 m
<b>Disponibilité</b>	mondiale	mondiale	mondiale	mondiale	mondiale

**Tableau 1 : Caractéristiques techniques des différents standards**

Par la suite, nous allons détailler le mode de fonctionnement de 802.11

## 1.5 Etude de la norme 802.11

De manière générale, un équipement réseau comporte sept couches, qui vont de la couche Physique à la couche Application. Dans notre cas, plusieurs couches ne sont pas prises en compte comme par exemple la couche réseau (3ème couche) qui sert au routage entre les réseaux car nous nous situons toujours dans un seul réseau de communication. Le modèle sera donc réduit en trois couches : *Physique*, *MAC* et *Application* (Figure 1.3).

*Couche Physique(PHY)*: c'est la première couche dans le modèle OSI. Elle est chargée de la transmission effective des signaux. Son service est typiquement limité à l'émission et la réception d'un bit ou d'un train de bits continu. Cette couche est chargée donc de la conversion entre bits et signaux électriques ou optiques. Il existe plusieurs technologies de modulation de signaux sur des bandes de fréquences ISM (*Industrial, Scientific and Medical*) dédiées aux réseaux WLANs (*Wireless-LAN*). Voici une brève description de différentes technologies sans fil existantes : 802.11 (IEEE 802.11, 1999) offre un débit de 2 Mbit/s. 802.11b (IEEE 802.11b, 1999) et 802.11a (IEEE 802.11a, 1999) sont basées sur la technologie *Direct*

*Sequence Spread Spectrum* (DSSS) et *Orthogonal Frequency Division Multiplexing* (OFDM), elles fournissent un débit maximal de 11 Mbit/s et 54 Mbit/s respectivement. 802.11g (IEEE 802.11g, 2003) utilise la même technologie que le 802.11a, son débit peut donc atteindre 54 Mbit/s et elle a une compatibilité avec la norme 802.11b.

L'évolution au niveau de la couche physique en termes de débit montre clairement les grandes avancées qui ont été faites afin de répondre aux besoins des applications multimédia exigeantes en termes de bande passante. Notons bien que chaque couche encapsule le paquet reçu par des entêtes. Ces entêtes servent à transmettre des informations concernant le protocole ou la technologie utilisée, l'adresse de la station émettrice, des informations pour détecter et corriger les éventuelles erreurs dans ce paquet. Notons finalement que la taille de ces entêtes dépend essentiellement de la technologie utilisée, par exemple, l'entête dans la couche physique pour l'IEEE 802.11b est de 192 bits.

*Couche MAC:* Cette couche fournit les moyens fonctionnels et procéduraux pour le transfert de données entre les nœuds adjacents d'un réseau. Pour Wifi, deux technologies majeures existent dans cette couche: IEEE 802.11 et 802.11e. Elles seront détaillées par la suite.

Les deux couches Physique et MAC sont réalisées par un circuit électronique spécifique. Ce circuit porte plusieurs noms, par exemple : coupleur de communication, interface réseau, carte E/S réseau ou carte réseau sans fil. En conclusion, dans un SDCR sans fil, chaque équipement qui envoie des informations à travers le réseau doit être équipé de ce type de carte.

*Couche Application:* dans cette couche, les paquets d'information sont créés. Dans un SDCR, cette création se fait d'une manière périodique (*Pcarte*) appelé *période d'échantillonnage* ou *de scrutation*. Après la création, ces paquets sont envoyés vers les couches de niveaux inférieurs jusqu'à la couche Physique qui les transmet sur le médium. Du côté récepteur, les paquets reçus sont immédiatement désencapsulés et les données sont extraites (pas de traitement périodique).

Application			
Liaison(MAC)	802.11	802.11e	
Physique	802.11a	802.11b	802.11g

**Figure 1.3 : l'architecture en trois couches : PHY, MAC et Application**

Les couches Application et Physique (avec leurs différentes technologies) peuvent fonctionner avec les deux standards IEEE 802.11 et 802.11e. IEEE 802.11e fournit la notion de priorité entre les flux du réseau. Il sera exploité pour améliorer la performance du SDCR

sans fil. Nous allons par la suite détailler le fonctionnement de la couche MAC dans ces deux standards.

### 1.5.1 IEEE 802.11: Couche MAC

La couche liaison, dans la norme IEEE 802.11, est constituée de deux sous-couches: LLC (*Logical Link Control*) et MAC (*Medium Access Control*). La première sous-couche consiste à délimiter les trames et à corriger d'éventuelles erreurs survenues lors de la transmission physique. Le rôle de la deuxième sous-couche est de gérer le partage du support physique entre plusieurs stations en mettant en œuvre des mécanismes d'accès au canal. Pour atteindre cet objectif, le standard IEEE 802.11 définit un protocole d'échange des trames. La séquence minimale d'échange contient deux trames: une trame de données envoyée de la source à la destination et une trame d'acquiescement (ACKnowledgment ou ACK) envoyée de la destination à la source une fois que la trame de données est reçue avec succès. Chaque trame émise contient une *Frame Check Sequence* (FCS : 32bit CRC) qui est vérifiée par le destinataire lors de sa réception. Si la FCS correspond au contenu attendu, le destinataire envoie un ACK, le cas échéant cet ACK n'est pas envoyé. Si la source n'a pas reçu l'ACK attendu, la trame est retransmise de nouveau. Ce mécanisme permet de pallier les problèmes d'erreurs causés par des interférences sur le canal radio. Cela garantit l'intégrité des données au niveau de la couche liaison.

Dans cette partie, nous cantonnerons notre étude à la sous-couche MAC et présenterons les différents modes d'accès, ainsi que leurs avantages et inconvénients. IEEE 802.11 propose deux modes d'accès au canal: *Distributed Coordination Function* (DCF) et *Point Coordination Function* (PCF).

La transmission dans le mode DCF s'effectue d'une manière asynchrone, c'est à dire que toutes les stations tentent d'envoyer des données, sans être gérées par un contrôleur. PCF est également un mode de communication asynchrone mais où les émissions des stations sont gérées par un contrôleur (point accès). Notons que le mode DCF peut être utilisé dans un réseau en mode infrastructure ainsi qu'en mode ad-hoc, tandis que PCF est utilisable seulement en mode infrastructure. Finalement, la méthode d'accès fondamentale est la DCF, alors que la méthode PCF est optionnelle.

#### 1.5.1.1 Mode DCF : Distributed Coordination Function

Le mode DCF est basé sur l'algorithme *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) une adaptation du CSMA/CD (*Collision Detection*) qui est utilisé par le protocole Ethernet dans les réseaux locaux filaires. Avec CSMA/CD, la station émettrice envoie et écoute le médium en même temps pour détecter d'éventuelles collisions avec d'autres signaux existants dans le canal. Ce n'est pas le cas dans les réseaux sans fil où les collisions peuvent ne pas être détectées par la station émettrice car la puissance du signal émis masque tous les autres signaux. Une autre raison qui peut causer des collisions est le problème

des stations cachées. Nous détaillerons après ce problème. Donc pour qu'une station sans fil envoie des données en évitant les collisions : elle écoute le medium, si ce dernier est libre, la station commence à transmettre les trames. L'algorithme CSMA/CA est basé sur ce mécanisme de transition. Cette méthode ne garantit pas la transmission correcte des trames surtout si la collision est due à des effets extérieurs comme ondes électromagnétiques venant des machines dans un milieu industriel. Un mécanisme d'acquiescement est implémenté pour assurer l'arrivée correcte des trames émises vers le récepteur. Pour cette raison, la communication sans fil est considérée comme connexion half duplex.

Pour assurer le fonctionnement d'accès au canal, le DCF se base sur un jeu d'intervalle de temps appelé IFS (*Inter Frame Spacing*). Lorsqu'une station souhaite émettre des trames, elle écoute le medium pendant un temps nommé DIFS (*Distributed IFS*). Si le canal est libre durant cette période, et si la file d'attente dans la couche MAC de l'émetteur est vide, la trame est directement envoyée. Si non, la station tire un temps aléatoire nommé *Backoff Time* (BT) durant lequel elle s'absente. Notons que, BT représente un nombre de slots aléatoirement choisi suivant une loi de distribution uniforme dans un intervalle.

$BT = rand[0, CW] * \text{slot time}$   
avec  $rand[ ]$  est une fonction  
qui suit une loi de distribution uniforme

Cet intervalle est appelé fenêtre de contention CW (*Contention Window*). Le slot est une unité de temps (slot time) défini par la couche physique (exemple : slot time pour 802.11a=9μs). La Figure 1.4 explique l'accès au canal en utilisant le mode DCF. La station reste bloquée tant que le canal est occupé. Chaque fois que le médium reste libre durant une durée égale à DIFS, la station recommence à décrémenter son ancien BT. Une fois que le BT atteint la valeur zéro, la station transmet la trame immédiatement. La Figure 1.5 illustre la procédure de Backoff pour différentes stations.

Dans le cas où la trame envoyée n'est pas erronée, la station réceptrice envoie une trame d'acquiescement à l'émetteur après un temps SIFS (*Short IFS*). Ce dernier est utilisé pour séparer les trames d'un dialogue. Sa valeur est inférieure à celle d'un DIFS. Notons que  $DIFS = SIFS + 2 * \text{slot time}$ .

Dans le cas où l'émetteur ne reçoit pas la trame d'acquiescement, c'est à dire lors d'un échec de transmission, la trame de donnée est supposée mal reçue par le récepteur. Cette situation est le plus souvent due à une collision entre les trames émises par différentes stations dans le canal. Pour sortir de cette situation, la station ralentira son rythme d'émission afin que le canal soit moins occupé. Pour cela, elle doublera sa fenêtre de contention CW dans le but d'avoir un BT (temps d'absence) plus grand, puis elle recommencera de nouveau la procédure.

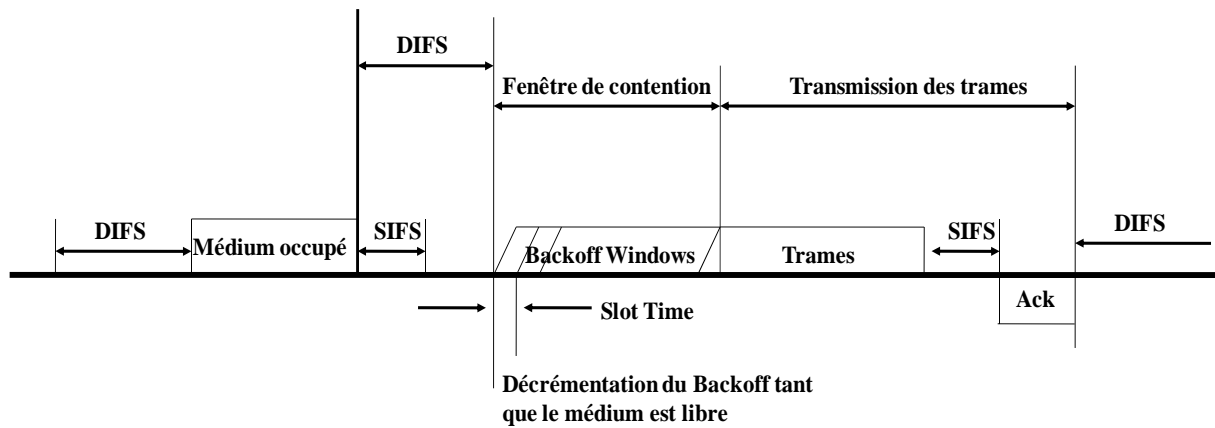


Figure 1.4 : la procédure du *Backoff time* (BT)

La taille de la fenêtre de contention est initialisée à une valeur  $CW=CW_{min}=31$ . Lors de chaque nouvelle tentative de transmission suite à un échec,  $CW$  augmente exponentiellement sa valeur selon la formule  $(CW+1)*2 - 1$ .  $CW$  augmentera jusqu'à atteindre  $CW_{max}$ . En cas de succès d'une transmission,  $CW$  est réinitialisée et reprendra la valeur  $CW_{min}$ . Notons qu'une station transmettra une même trame un nombre maximal de fois, équivalent à *Short Retry Limit* lorsque la taille de cette trame est plus petite qu'un seuil spécifié, ou bien *Long Retry limit* lorsque sa taille est plus grande qu'un seuil spécifié. Notons bien que les deux attributs *short Retry limit* et *Long Retry limit* sont des attributs variables. En d'autres termes l'utilisateur peut gérer ces attributs pour paramétrer son réseau.

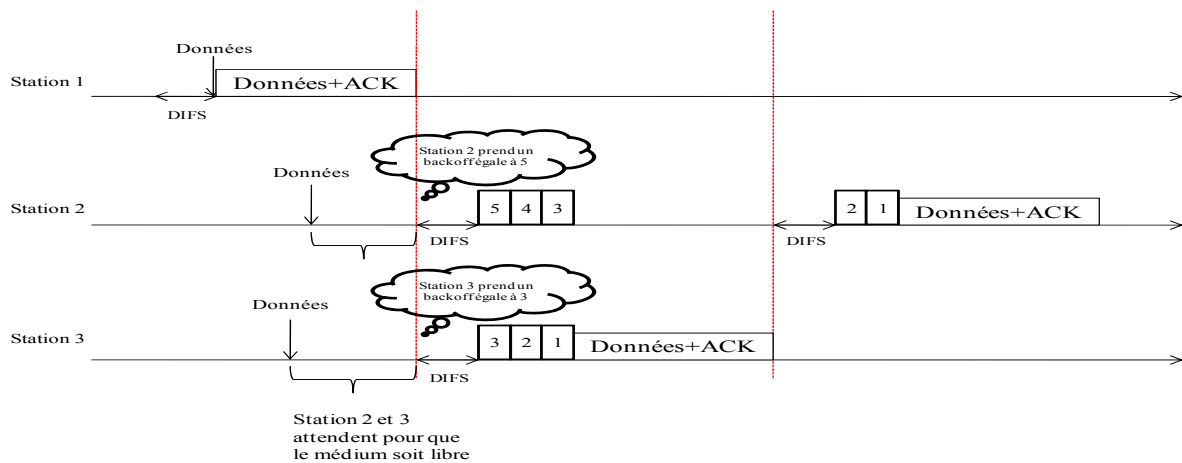


Figure 1.5 : Chronogramme de l'algorithme CSMA/CA

Un problème classique des réseaux sans fil est relatif aux stations cachées : deux stations émettrices se situent hors de portée radio l'une de l'autre, mais partagent une zone de couverture commune. Si ces stations veulent envoyer des trames vers une station réceptrice qui se trouve dans la zone commune, leurs trames peuvent entrer en collision puisque chaque station croit que le canal est libre. Pour palier ce problème, la station émettrice peut utiliser un protocole MAC optionnel: le RTS/CTS (*Request To Send / Clear To Send*). Lorsque cette

fonction est utilisée, la station émettrice envoie un RTS à la station réceptrice. Cette dernière répond par une trame CTS qui sera entendue par toutes les stations dans la portée de la station réceptrice. Les trames RTS et CTS contiennent des informations sur la durée d'occupation du réseau nécessaire pour la transmission de données entre la station émettrice et la station réceptrice. Par conséquent, toutes les stations émettrices qui se situent à la portée de la station réceptrice pourront donc différer de ce temps leur accès au médium.

### 1.5.1.2 Mode PCF : Point Coordination Function

Pour pouvoir répondre aux besoins des applications qui demandent des temps de service bornés, IEEE 802.11 définit le mode d'accès PCF pour permettre aux stations d'avoir une priorité d'accès au médium radio.

PCF est un mode de communication synchrone où chaque station émet ses trames après avoir reçu une autorisation. Donc l'accès au médium est géré par un contrôleur (*Point Coordinator, PC*), qui est relié au point accès. C'est pour cette raison que le PCF n'est utilisé que dans le mode infrastructure. Le PC appelle les stations d'une manière cyclique. Ce cycle est nommé *Beacon interval* et la longueur de ce cycle est défini par *Target Beacon Transition Time (TBTT)*.

Ce cycle est formé de deux grandes parties :

- la *Contention Free Period (CFP)* où le PC gère l'accès au médium sans contention (mode PCF)
- la *Contention Period (CF)* où le mode DCF est utilisé.

Au début du *Beacon Interval* et pour qu'il ait la priorité d'accès au médium, le PC écoute le canal pendant un temps *Priority InterFrame Space (PIFS, PIFS < DIFS)* et diffuse une trame balise (*Beacon frame*) à toutes les stations, cette trame contient le TBTT et la durée maximale du CFP nommée *CFP\_Max\_Duration*. Les stations qui reçoivent cette trame mettent à jour leur temporisateur appelé *Network Allocator Vector (NAV)*, l'affectant à *CFP\_Max\_Duration*. Ces stations décalent leurs souhaits d'accéder au médium avec CSMA/CA jusqu'à l'expiration de leur temporisateur.

Ensuite, le PC commence à interroger les stations associées en envoyant des trames CF-Poll afin de savoir si elles possèdent des données prioritaires à transmettre. La station destinataire du CF-Poll répond à son tour, après un temps SIFS, une trame d'acquiescement (CF-ACK) qui acquiesce la CF-Poll. Généralement, des données accompagnent les trames CF-ACK. Pour sa part, le PC acquiesce la trame qu'il reçoit de la station également après un intervalle SIFS. Cet acquiescement est généralement accompagné par une CF-Poll pour interroger une autre station. Si la station destinataire ne répond pas après un temps PIFS, le PC passe à l'interrogation de la station suivante. La Figure 1.6 illustre un exemple du *Beacon Interval* avec les différentes étapes qui se produisent durant cet intervalle.

Peu de vendeurs d'équipements Wifi implémentent le mode PCF dans leurs cartes réseau. Pour cette raison, ce mode est utilisé dans des applications privées. Par contre, toutes les cartes réseau sans fil communiquent à travers le mode DCF. Nous nous intéressons donc au mode DCF.

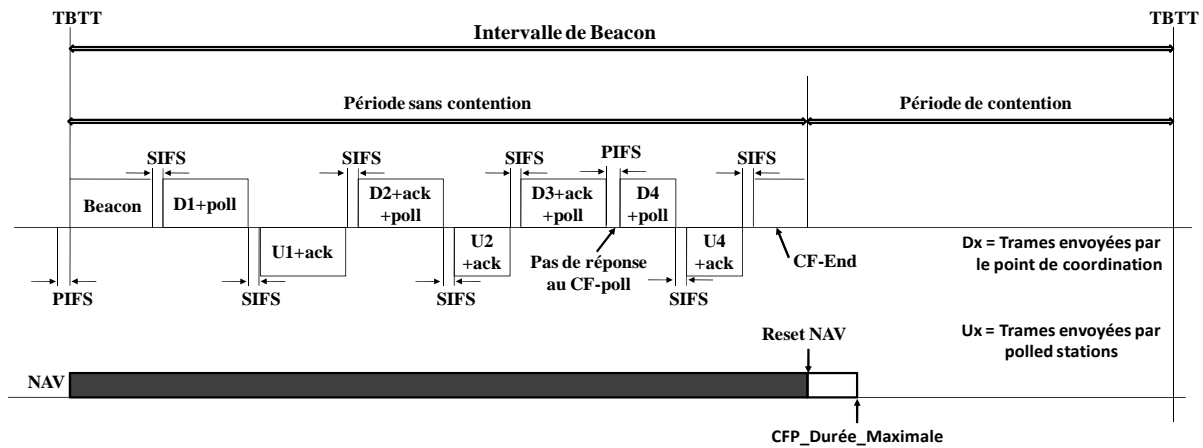


Figure 1.6 : Les règles d'accès dans PCF

### 1.5.1.3 Limitations des modes DCF et PCF

Durant la communication dans un réseau IEEE 802.11, la propagation des trames est confrontée à des attaques d'ondes perturbatrices, qui peuvent errer le contenu des trames et donc augmenter le taux de perte. Lors d'une transmission non réussie, l'émetteur est obligé de renvoyer la trame, ce qui augmente ainsi le délai de bout en bout ainsi que la gigue. Ces trois paramètres (délai, taux de perte et la gigue) sont des contraintes essentielles pour certaines applications, comme par exemple les applications temps réel soumises à de fortes contraintes de sûreté de fonctionnement, ou les applications de transmission vidéo (délai < 100ms). Malheureusement, le standard 802.11 ne répond pas à l'exigence de telles applications.

Dans le mode PCF, une station aura le droit de transmettre dès qu'elle reçoit une permission du PC. Mais le problème est qu'elle peut alors transmettre des trames infiniment, ce qui retarderait l'accès au médium pour les autres stations. La trame balise peut être retardée à cause de l'occupation du médium par des stations après un TBTT. Dans le pire cas, ce retard est équivalent à 4.9ms dans l'IEEE 802.11a et en moyenne de 250µs (Mangold, et al., 2002).

Dans le mode DCF, toutes les stations qui veulent envoyer des données, utilisent les mêmes paramètres de l'algorithme CSMA/CA. En d'autres termes, toutes les stations attendent un temps d'attente DIFS+Backoff Time, ce temps étant indépendant de l'urgence des données envoyées. Par conséquent, toutes les stations auront la même priorité pour accéder au canal. Plus le nombre de stations augmente, plus la probabilité de collision augmente, ce qui cause des retransmissions fréquentes. Tout cela conduit à des dégradations de la performance du système. Les applications ayant des contraintes de QoS sont les plus impactées par ces

dégradations. En conclusion pour le DCF, il n'y a aucune différenciation entre les flux et aucun privilège n'est attribué aux flux qui ont des contraintes de QoS.

Vu les limitations de ce standard 802.11, surtout pour les applications temps réel, des recherches ont été menées pour développer des solutions plus robustes afin de pouvoir gérer la QoS dans le standard. Ces solutions sont classifiées en trois catégories:

La première catégorie (Aad, et al., 2004), (Kwon, et al., 2003), (Pollin, et al., 2005), (Chatzimisios, et al., 2005) regroupe les solutions d'amélioration de performances par l'utilisation de techniques de changement des paramètres de backoff selon les conditions du réseau. Plus précisément, les changements peuvent se faire au niveau des valeurs de CW, de la loi de distribution choisie, etc...

La deuxième catégorie (Choi, et al., 2005), (Choi, et al., 2000) est consacrée aux solutions dans lesquelles de nouveaux mécanismes d'accès sont proposés en remplacement de ceux du standard 802.11 initial, afin d'adapter celui-ci à des applications temps réel.

La dernière catégorie (Aad, et al., 2001), (Zhao, et al., 2003), (Pattara-Atikom, et al., 2003) repose sur les solutions de différenciation de services dans le standard pour la gestion de différentes priorités.

Vu les limitations du 802.11, les solutions proposées par les auteurs dans les deux premières catégories présentent une complexité non négligeable et les méthodes proposées sont testées sur des petites échelles. Donc leur efficacité dans un réseau à grande échelle reste inconnue. En revanche, la dernière catégorie peut être une solution pour répondre aux besoins des applications multimédia et temps réel. L'IEEE 802.11e, extension du 802.11, se trouve dans cette catégorie.

### **1.5.2 IEEE 801.11e: Couche MAC**

L'IEEE a publié en 2005, le standard 802.11e (IEEE 802.11e, 2005). Ce standard introduit la notion de la QoS des trafics. Chaque trafic, selon ses contraintes de QoS, aura une priorité, qui favorise ou défavorise son accès au médium. Dans ce standard, les points d'accès (AP) et les stations sont appelés respectivement QAP (*QoS-enhanced Access Point*) et QSTA (*QoS-enabled Station*). Ce nouveau standard définit une nouvelle fonction de coordination appelée *Hybrid Coordiantion Function* (HCF). Cette dernière utilise deux mécanismes d'accès au médium, *l'Enhanced Distributed Channel Access* (EDCA) qui est un mode fondamental et *l'HCF-Controlled Channel Access* (HCCA) qui est optionnel. La Figure 1.7 présente la nouvelle couche MAC du standard 802.11e.

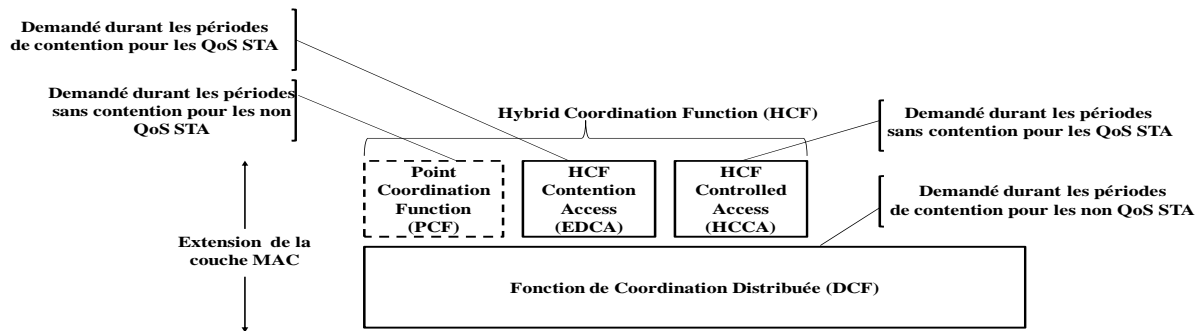


Figure 1.7 : Architecture de la couche MAC dans 802.11e

### 1.5.2.1 Enhanced Distributed Channel Access, EDCA

A l'identique du mode DCF de la norme 802.11, EDCA fournit une transmission asynchrone et fonctionne en mode ad-hoc ou en infrastructure. Dans ce mode, chaque flux aura une priorité. Chacune d'elle correspond à une probabilité d'accès au médium différente. EDCA utilise le même algorithme d'accès au médium que pour le DCF : le CSMA/CA. Mais le jeu d'intervalles de temps (*Backoff time*, DIFS) sur lequel cet algorithme repose a été personnalisé pour chaque priorité. Plus le flux est important (exemple: information critique de système de commande tel qu'une alarme) plus sa priorité est grande et plus le flux aura la possibilité d'accéder au canal grâce à un DIFS plus petit et un *Backoff time* tiré dans un intervalle plus petit.

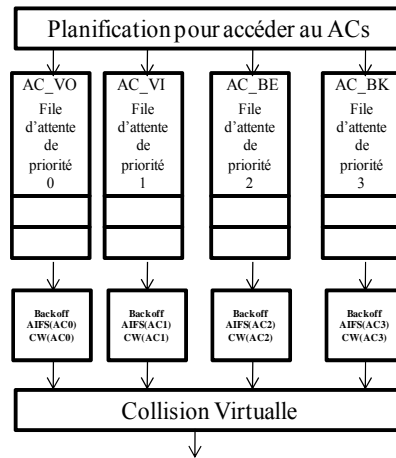
Les types de flux portent un identifiant nommé *Traffic Identifier* (TID). Selon leur besoin de QoS, c'est à dire selon leur priorité, TID varie de 0 à 15. Les flux de TID compris entre 0 et sept utilisent l'EDCA, tandis que les autres flux (TID=8,...,15) utilisent le mécanisme HCCA.

Dans l'EDCA, les huit niveaux sont appelés *User Priority* (UP). Chaque priorité est associée à une catégorie d'accès (*Access Category* ou AC). Le Tableau 2 indique les différentes priorités des flux ainsi que leur classification selon les ACs.

Priorité	User priority (UP)	802.1D Designation	Access Category (AC)	Designation
↓ Plus haut	1	BK	AC_BK	Background
	2	-		Background
	0	BE	AC_BE	Best Effort
	3	EE		Best Effort
	4	CL	AC_VI	Vidéo
	5	VI		Vidéo
	6	VO	AC_VO	Voix
	7	NC		Voix

Tableau 2: Relation entre les priorités et les catégories d'accès dans EDCA

En réalité, chaque AC utilise une file d'attente. Chacune possède ses propres paramètres et son propre comportement *EDCA Function* (EDCAF). La Figure 1.8 présente clairement les quatre AC qui se trouvent dans une carte réseau sans fil.



**Figure 1.8: Modèle d'implémentation de référence d'EDCA**

Rappelons que dans le mode DCF, les caractéristiques de temps d'attente sont les paramètres DIFS, CWmin et CWmax. Pour l'EDCA, DIFS est appelé *Arbitration IFS* (AIFS) et chaque AC(i), est caractérisé par un AIFS(i), CWmin(i), CWmax(i) où i représente les quatre AC.

Plus le flux présente des contraintes de QoS, plus son AC est grande, plus le temps d'attente pour accéder au médium doit être faible. Par conséquent, AIFS(i), CWmin(i) et CWmax(i) présenteront des petites valeurs. Un autre paramètre intégré dans EDCA est le *Transmit Opportunity*, TXOP(i). 802.11e définit cette variable comme étant un intervalle de temps durant lequel une station possède le droit de transmettre en continu.

Le déroulement des transmissions des données dans EDCA se fait comme suit:

Les trames qui arrivent vers la couche MAC sont divisées selon leurs priorités. Puis, elles entrent dans l'une des quatre files d'attente (AC). Dans l'EDCA, chaque AC se comporte à elle seule comme une station virtuelle. Plus précisément, chaque AC attend un temps AIFS(AC) et décrémente son propre temps Backoff si le médium est libre. De même, si le médium est occupé, tous les temps Backoff des AC se bloquent. En d'autres termes, chaque file exécute son propre algorithme CSMA/CA. Une collision peut se produire entre les différents ACs dans une même station physique. Dans ce cas, l'AC de haute priorité gagne l'accès au médium, tandis que l'AC de faible priorité prend un nouveau temps Backoff en doublant la fenêtre de contention, c'est comme si son paquet avait subi une collision avec une trame venant d'une autre station. Ce phénomène est appelé Collision Virtuelle.

Le comportement à l'identique du DCF se poursuit, après transmission des trames, chaque AC attend une trame d'acquiescement. Au cas où cette dernière n'arrive pas, l'AC double la fenêtre

de contention et exécute de nouveau la procédure de Backoff. La station recommence à transmettre la trame durant un nombre maximal de fois, équivalent à *Short/Long Retry Limit*. Après ces retransmissions, la trame est supposée perdue. En général, dans un SDCR sans fil, les trames sont de petite taille donc le *Short Retry Limit* est plutôt utilisé dans ces systèmes. Dans la suite de ce mémoire, le *Short Retry Limit* sera appelé *Nombre maximal de retransmission*.

Dans le but de respecter une QoS pour les trafics temps réels dans EDCA, le QAP peut demander à certains AC un contrôle d'admission. Cette demande sera effectuée via la trame de balise grâce à un champ appelé *Admission Control Mandatory (ACM)*. Si l'ACM d'un AC est égal à 1, cela aura comme signification que tous les trafics qui appartiennent à cet AC doivent faire une demande d'accès au QAP afin d'envoyer leurs trames. Au cas où ils n'ont pas eu l'autorisation de transmettre, ces trafics diminuent leurs priorités. Dans le cas contraire, si l'ACM d'un AC est égal à 0, cela signifie que ses trafics peuvent directement accéder au réseau.

En général, les quatre valeurs des paramètres de chaque AC, AIFS(i), CWmin(i), CWmax(i) et TXOP(i) sont définies dans le standard par défaut mais elles peuvent être changées dynamiquement par le QAP, qui envoie ces valeurs dans la trame de balise. Rappelons bien que ce cas n'existe pas dans le 802.11-DCF, les DIFS, CWmin et CWmax sont des valeurs fixes. Dans notre cas, nous utilisons les valeurs par défaut.

Le Tableau 3 montre les différentes valeurs des paramètres de chaque AC. Notons que aCWmin, aCWmax, aSlotTime ainsi que TXOP sont des attributs qui dépendent de la technologie choisie au niveau de la couche physique (802.11a, b ou g).

AC	CWmin	CWmax	AIFSN AIFS = SIFS + AIFSN × aSlotTime
AC_BK	aCWmin	aCWmax	7
AC_BE	aCWmin	aCWmax	3
AC_VI	$(aCWmin+1)/2 - 1$	aCWmin	2
AC_VO	$(aCWmin+1)/4 - 1$	$(aCWmin+1)/2 - 1$	2

**Tableau 3 : les paramètres de différenciation par défaut d'EDCA**

Dans EDCA, une option (NO ACK) est incluse : le récepteur peut ne jamais envoyer des trames d'acquiescement. L'émetteur suppose alors que la trame de données est arrivée correctement au destinataire. Une autre option qui est intégrée dans l'EDCA, c'est le BLOCK d'acquiescement où le récepteur envoie une seule trame d'acquiescement après avoir reçu plusieurs trames de données. Dans la trame d'acquiescement, il indiquera les trames de données qu'il a mal reçues. Comme dans le DCF, le mécanisme RTS/CTS peut être utilisé dans l'EDCA afin de réserver le médium pour échanger des informations entre deux stations.

Rappelons que ce mécanisme a pour but de diminuer les collisions entre différentes stations cachées.

### 1.5.2.2 HCF-Controlled Channel Access, HCCA

HCCA est utilisé seulement dans un réseau infrastructure et fournit une transmission synchrone. Ce mécanisme est similaire au mode PCF, avec cependant quelques modifications. Dans HCCA, PC est appelé HC (*Hybrid Coordinator*). Ce dernier sert à contrôler les trafics du réseau de telle sorte que chaque flux doit envoyer une demande au HC pour une obtenir une permission d'accès au médium. Sa demande décrit ses besoins de QoS. Le HC répond par une trame qui donne à la station un temps global maximal d'accès appelé *medium\_time*. Le HC autorise également à la station d'envoyer en continu durant un temps limité, HCCA-TXOP, pour qu'elle ne monopolise pas le médium. L'algorithme utilisé pour calculer le *medium\_time* est un algorithme ouvert, qui est un sujet de recherche.

Des études (Ansel, et al., 2004),(Choi, 2004) ont montré que l'algorithme standardisé pour calculer HCCA-TXOP est efficace pour les flux à débit constant mais l'est moins pour les flux à débit variable. Des travaux de recherche (Lee, et al., 2006), (Grilo, et al., 2003), (Passas, et al., 2006), (Inan, et al., 2006), (Yang, 2004) font des propositions afin de déterminer les valeurs de HCCA-TXOP pour obtenir une meilleure performance et assurer les besoins des applications.

Ce mode n'est pas implémenté dans les cartes réseau sans fil disponibles sur le marché, vu sa complexité de gestion des flux. Par conséquent, il est préférable de s'intéresser plutôt au mode EDCA. Ce dernier existe actuellement dans les nouvelles cartes sous le nom de WMM (*WiFi-Multimedia*). *Intel® Wireless WiFi Link 4965AGN* est un exemple d'une carte que l'on trouve actuellement sur le marché, qui peut gérer le mode EDCA.

## 1.5.3 Conclusion et hypothèses

Après cette présentation de tous ces travaux et standards, nous ne nous intéresserons pas au mode PCF et HCCA vu leur complexité et surtout leur non commercialisation, mais plutôt au mode DCF du 802.11 et à l'EDCA du 802.11e.

### 1.5.3.1 Hypothèses

Dans le but de prendre des précautions et de ne pas négliger des options intéressantes dans ces deux modes, des hypothèses sont posées :

- *Contrôle d'accès* : (Xiao, et al., 2004) comparent la différence lors de l'utilisation ou non des permissions d'accès du mode EDCA. Ils montrent qu'il est plus performant que les flux suivent la procédure de demande de permission plutôt que d'accéder directement au canal, mais cette procédure prend un temps non négligeable. Ce temps peut déstabiliser un système à temps réel comme un SDCR sans fil. **Pour cela,**

**l'hypothèse prise dans ce travail suppose que la permission d'accès est donnée à tous les flux.**

- *Transmission avec/ sans ACK* : (Wall, et al., 2003) étudient l'intérêt de l'option **NO ACK** dans le 802.11e sur différents types de trafics. Ils prouvent par ces simulations qu'il est plus intéressant d'avoir cette option pour les trafics à temps réel jusqu'au moment où le récepteur ne peut plus corriger les trames erronées en utilisant le *forward error correction* (FEC). Ils proposent même un plan dynamique avec des transmissions avec ou sans acquittement, suivant la situation du réseau. Dans nos contraintes, le délai de bout en bout est un paramètre important mais la fiabilité du réseau reste un paramètre à garder en considération. A titre d'exemple, nous pouvons citer la transmission d'un signal d'alarme d'une station (machine industrielle) vers un AP. La réception de ce signal doit être assurée et le plus vite possible. **Par conséquent, nous prenons l'hypothèse que toutes les transmissions sont avec acquittement.**
- *Utilisation du RTS/CTS* : selon le standard, ce mécanisme résout le problème des stations cachées et diminue les collisions entre ces stations. (Fitzpatrick, et al., 2006) vont plus loin, ils utilisent les paquets (RTS et CTS) pour donner la permission à un flux d'accéder au médium. Plus précisément, sa méthode consiste à ce que l'AP extraie la durée de transmission des trames de données à venir (NAV) qui se trouve dans l'entête de la trame RTS. Cette information servira à estimer le temps d'occupation du médium par les stations émettrices. Par conséquent, l'AP peut accepter ou rejeter une demande d'un flux selon l'influence de ce flux sur le réseau global. Par contre, ce mécanisme présente des inconvénients dans le cas où une des deux trames RTS ou CTS n'arrive pas à l'AP, à cause d'interférences par exemple. Une étude récente faite par (Shih, et al., 2009) montre aussi que ce mécanisme ne résout finalement pas le problème des stations cachées ! Ce travail présente des cas (ex : réseau de 4 stations) où les trames RTS et CTS peuvent entrer en collision. Cette situation amène à des collisions entre les paquets de données envoyés par des stations cachées. Notons que, cette étude présente un algorithme appelé *RTS collision avoidance* (RCA) qui résout ce problème.  
Dans notre cas, les stations appartiennent à une même zone de couverture, en d'autre terme, **nous considérons qu'il n'y a pas de stations cachées.**
- *TXOP* : rappelons que le TXOP est un intervalle de temps durant lequel une station particulière possède le droit de transmettre en continu. Ce paramètre n'a pas une grande influence dans notre type d'application car les données envoyées sont de petite taille. **Par conséquent, l'hypothèse consiste à ne pas considérer le TXOP.**

### 1.5.3.2 Attributs retenus pour l'étude

Dans ces deux modes, nous identifions plusieurs attributs. La manipulation de ces attributs est autorisée par le standard donc l'utilisateur peut les gérer:

- *Taille des paquets*: Cet attribut est géré au niveau de la couche Application. Rappelons que cette couche crée un paquet de données de taille (=Taille des paquets) spécifiée par l'utilisateur. Dans les systèmes contrôlés, les données échangées par les capteurs et le contrôleur concernent généralement de petites informations, donc les paquets envoyés sont de petite taille à l'échelle de quelques octets.
- *Période d'échantillonnage (Pcarte)* : Cet attribut spécifie la durée entre la prise de deux échantillons successifs. A chaque période, la couche application forme un paquet et l'envoie vers les files d'attente de la couche MAC.
- *Nombre maximal de retransmission* : Comme expliqué précédemment, dans le cas de la perte d'une trame envoyée, la station source continue à retransmettre cette trame *un nombre maximal* de fois. Après cela, le paquet sera considéré comme étant perdu.
- *Débit* : Cet attribut spécifie le débit maximal utilisé par la couche physique pour envoyer les trames. Ce débit dépend essentiellement de la technologie choisie dans cette couche.
- *Priorité* : La priorité entre les flux est un attribut gérable afin de donner une meilleure possibilité au flux pour l'accès au médium. Notons bien que cet attribut existe uniquement dans 802.11e.

Comme mentionné précédemment, ces attributs sont paramétrables. A titre exemple, nous pouvons citer la commande *iwconfig* sous UNIX, qui permet de régler certains de ces attributs:

**iwconfig interface [essid X] [nwid N] [mode M] [freq F] [channel C][sens S ][ap A ][nick NN ] [rate R] [rts RT] [frag FT] [txpower T] [enc E] [key K] [power P] [retry R] [commit]**

Avec **rate R** et **retry R** représentant respectivement le *Débit* et *nombre maximal de retransmission*.

Cisco présente sur ce lien (Cisco, 2010) une explication de 802.11e-EDCA et les options de ce mode implémentables sur ses cartes réseau ainsi que la manière de manipuler les attributs présentés précédemment

## 1.6 Influence des attributs réseau sur un système contrôlé en réseau

Rappelons que l'implémentation du réseau dans un SDCR sans fil introduit des délais, des pertes de paquet et de la gigue. Ces effets ont des influences négatives sur le comportement du système discret commandé surtout sur la performance temporelle et la sureté. Ces effets peuvent varier en fonction des attributs du réseau (*Pcarte, Débit,...*). Plusieurs travaux ont été faits pour étudier la gravité de l'influence de ces attributs sur la performance du système.

Un paramètre est défini dans (Jasperneite, et al., 2001) qui représente une image de la performance du SDCR. Ce paramètre est appelé temps de réponse ou *ResT*. Il est composé d'une part du temps mis par une trame depuis son émission du client vers le serveur et son

retour du serveur vers le client, appelé RTT en anglais (Round Trip Time) et d'autre part du temps nécessaire pour que le processeur traite les informations. Dans les SDCR sans fil, ce paramètre peut être défini comme le délai de bout en bout, entre l'occurrence de l'événement généré par un capteur (exemple : détection la présence ou l'absence d'un obstacle) et la réception par un actionneur du signal de commande émis en réaction de cet événement (exemple : un ordre d'arrêt de la voiture).

(Denis, et al., 2007) étudient *ResT* en fonction de la charge réseau. Un générateur envoie des requêtes de taille fixée afin de surcharger des équipements spécifiques. Ces derniers peuvent être la PC, le commutateur ou le lien Ethernet. Cette étude montre l'influence de ces équipements sur le système global quand ils sont en surcharge. Trois cas sont définis : 1) sans trafic générateur 2) des requêtes sont envoyées pour surcharger la PC and 3) des requêtes sont générées pour surcharger le lien Ethernet. Les résultats montrent que la valeur minimale de la distribution de *ResT* est plus faible dans le deuxième cas. Ces expériences montrent qu'avec un réseau possédant une grande bande passante, c'est surtout les capacités de traitement de la PC qui ont l'impact le plus fort sur *ResT*. Dans notre étude, nous n'auront malheureusement pas cette grande bande passante sur un réseau de type Wifi.

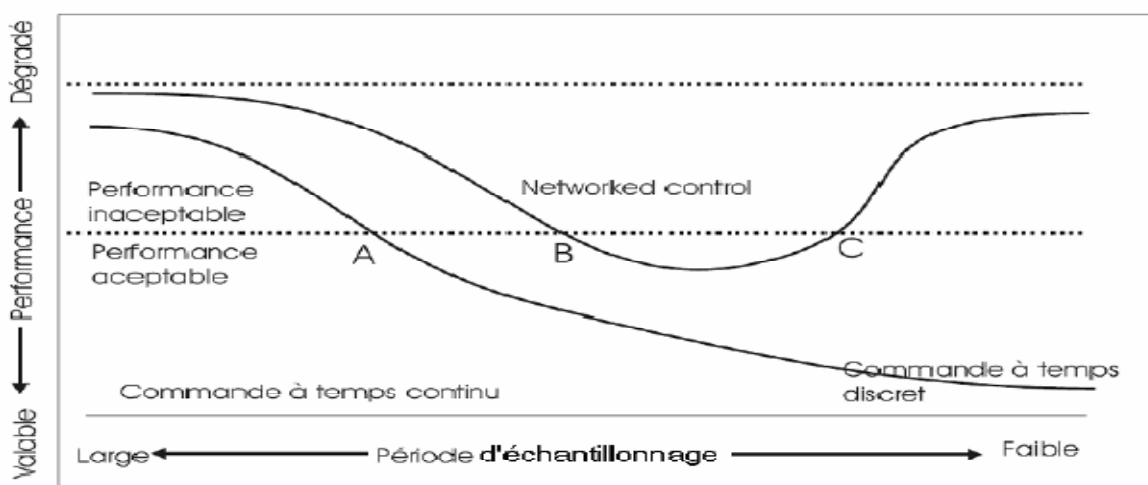
(Hasan, et al., 2007) comparent l'influence des différents protocoles de routage (DSR et AODV) sur le système selon différents débits. Ces résultats montrent que DSR est plus adaptatif à ces types de système. Dans notre cas, la structure du réseau est de type infrastructure donc les protocoles de routage ne sont pas pris en considération. (Colandairaj, et al., 2005) ont fait un travail intéressant basé sur l'étude des attributs du réseau (*nombre maximal de retransmission*, *Débit* disponible) sur la stabilité du système étudié. Ils font varier le *Débit* disponible et montre que plus cet attribut est grand plus le système est stable. Aussi, il regarde l'influence de l'attribut *Nombre maximal de retransmission*. Ils comparent deux valeurs pour cet attribut : 1 et 7. La comparaison montre que dans le premier cas, le système est plus stable que le deuxième. L'interprétation de ce résultat est que *Le rejet de certains anciens paquets peut diminuer le nombre de paquets envoyés sur le réseau donc le délai de bout en bout des paquets intéressants diminuera*. Le système reste stable tant que les informations arrivent. Ces conclusions sont aussi atteintes par (Habib, et al., 2009).

(Lian, et al., 2002) comparent l'influence de *Pcarte* sur la performance des différents types de système de contrôle. L'axe des ordonnées dans la Figure 1.9 désigne la performance du système. Cet axe varie entre valable et dégradé. Entre ces deux extrêmes, ils définissent deux plages dont l'une présente une zone de performance acceptable et l'autre présente une zone de performance non acceptable. Ce travail cite trois types de systèmes commande:

**Les commandes à temps continus** dont les signaux sont envoyées d'une manière analogique sans échantillonnage. Par conséquent, la performance est constante (ligne horizontale), indépendante de la période d'échantillonnage, et appartient à la zone de performance acceptable.

Le deuxième type rassemble **les commandes à temps discret**. Dans ce cas, les signaux sont échantillonnés. On remarque dans le cas où la période d'échantillonnage prend des valeurs grandes, que la performance du système est dégradée puisque les échantillons pris vont mal reconstruire le signal réel. Par contre, plus la période d'échantillonnage diminue, plus il y aura d'échantillons du signal, donc plus de précision. Ce qui aboutit à une convergence de la courbe de performance vers la zone de performance acceptable. Et plus il continue à décrémenter la période d'échantillonnage, plus la courbe tendra vers une meilleure performance.

Le dernier type de système est **le système contrôlé en réseau (NCS)**. Comme pour les commandes à temps discrets, dans le cas où  $P_{carte}$  présente des grandes valeurs, ces systèmes montrent une performance inacceptable. Puisque dans les deux types de systèmes, les signaux sont échantillonnés. La différence est que les systèmes contrôlés en réseau ont besoin de valeurs plus petites de  $P_{carte}$  ( $=P_B$ ) pour arriver à la zone de performance acceptable. Ceci est dû aux effets (délai, perte de paquets, gigue) du réseau sur le système. Plus  $P_{carte}$  diminue, plus le système aura des performances acceptables jusqu'à une certaine limite ( $=P_C$ ) où la courbe remonte de nouveau vers la zone de performance inacceptable. Et la performance du système commence à se dégrader. Ce comportement s'explique car le réseau est chargé par les nombreux paquets envoyés donc plus de délai et de perte de paquets, ce qui perturbe la performance du système.



**Figure 1.9 : Performance en fonction de la période d'échantillonnage**

En conclusion, ces travaux montrent qu'il est possible de tendre vers une performance acceptable en termes de temps de réponse en choisissant des plages appropriées pour la période d'échantillonnage des équipements du réseau.

D'autre part, les retards engendrés par le réseau peuvent avoir une influence sur le déterminisme de la commande, notamment dans le cas d'une gigue importante. En effet, considérons une même séquence logique d'entrées: a puis b. La commande présentée dans la

Figure 1.10 conduira à une situation différente selon la date d'occurrence de l'événement b. Si celui-ci arrive avant le début du troisième cycle de commande, les étapes X5 et X6 seront atteintes, sinon seule l'étape X5 sera activée. Pour régler ce problème, l'hypothèse classique consiste à considérer que le temps de cycle de la commande ( $P_{api}$ ) est sur cet exemple 3 fois plus rapide que le temps de réaction de la partie opérative (ce qui conduit à activer l'étape X5). Si le réseau introduit un retard fixe dans l'acheminement de l'information, il suffira de prendre ce retard pour évaluer le rapport entre temps de cycle de la commande et temps de réaction de la PO pour pouvoir garantir le déterminisme. En revanche, en présence d'une gigue importante sur le réseau, il sera très difficile de quantifier ce ratio et par voie de conséquence de garantir le déterminisme de la commande.

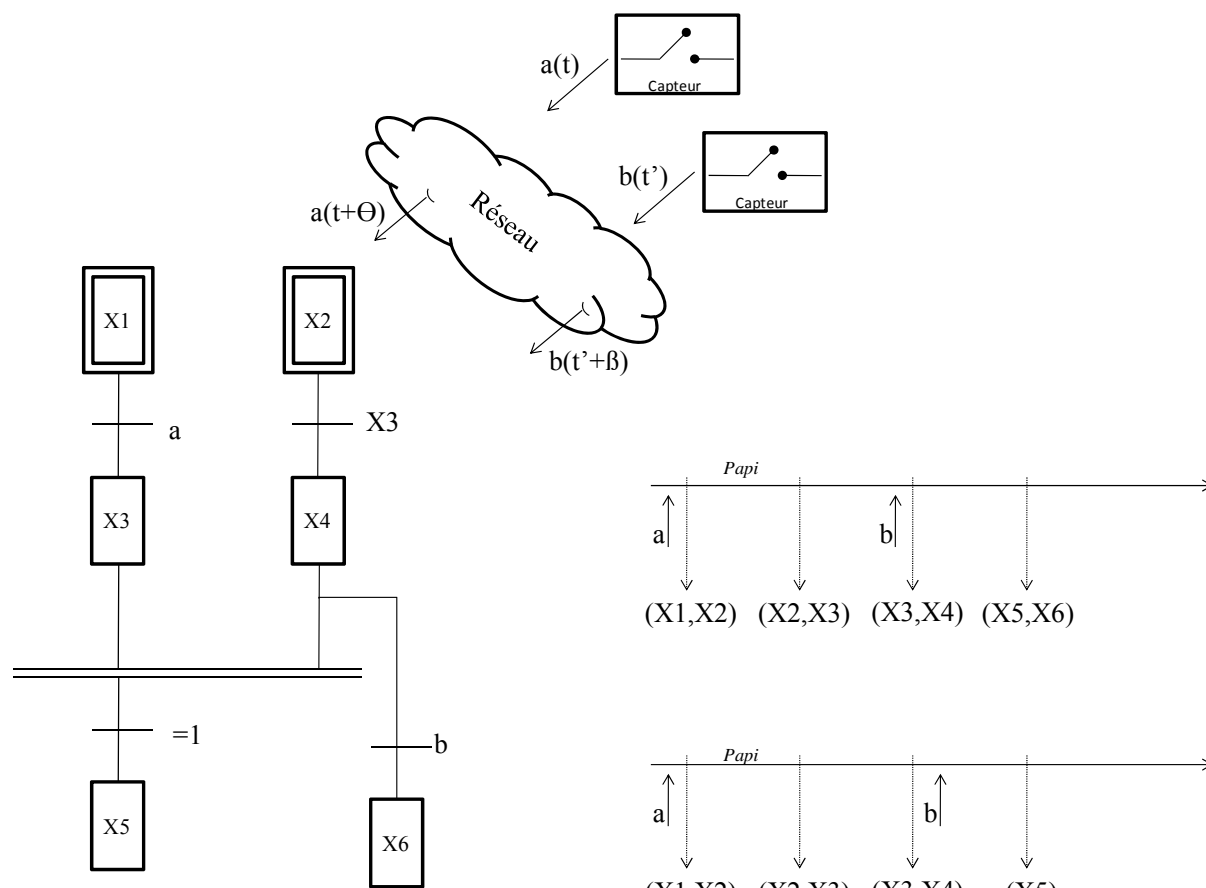


Figure 1.10 : Exemple d'un modèle de commande

## 1.7 Conclusion du chapitre

Dans cette étude, nous travaillons avec un système contrôlé nommé SDCR sans fil dont les informations échangées appartiennent à un espace d'état discret et évoluent dans des instants discrets de temps. De plus, le réseau choisi pour la communication entre les équipements est le réseau sans fil et spécialement le 802.11 et son extension le 802.11e. Ce réseau introduit des effets importants comme le délai et la perte de paquets, ce qui perturbe la performance du

système. D'où le but majeur de cette étude est d'étudier/améliorer la performance d'un SDCR sans fil en prenant en compte les effets du réseau.

Rappelons que chaque partie d'un SDCR sans fil présente plusieurs attributs gérables par l'utilisateur. Nous avons montré dans ce chapitre qu'il existe des plages de valeurs de ces attributs pour avoir une performance acceptable du système.

Dans le chapitre prochain, le problème posé est de trouver un moyen pour représenter ce type de système vu sa complexité et de surmonter les difficultés de modélisation. En plus, une étude des travaux précédents sera exploitée pour améliorer la performance des systèmes.

tel-00544932, version 1 - 9 Dec 2010

## **Chapitre 2**

# **Etat de l'art sur la modélisation, l'évaluation et l'amélioration d'un système contrôlé en réseau**

## 2.1 Introduction

Dans le chapitre précédent, nous avons positionné notre travail. Ce dernier consiste à étudier des systèmes discrets contrôlés à travers un réseau sans fil, spécifiquement 802.11 et 802.11e. La présence de ce type de réseau introduit des effets comme l'ajout d'un délai, des pertes de paquets et de la gigue, qui influent sur la qualité de service aussi bien que sur la qualité de commande. Ces effets négatifs peuvent être atténués en jouant sur les attributs du réseau (*nombre maximal de retransmission, débit,...*). De plus, ces effets sont non déterministes du fait de l'utilisation de l'algorithme CSMA/CA. Ce non déterminisme peut provoquer une instabilité dans ce type de système et peut changer sa performance. D'où l'intérêt de notre étude qui consiste à proposer une méthode d'amélioration de la performance du système global.

Afin de pouvoir proposer une méthode d'amélioration, nous devons passer par trois étapes :

1. Modéliser : trouver un moyen de représenter fidèlement un SDCR sans fil avec toutes ses parties et leurs attributs. (chapitre 2 et 3)
2. Etudier/Evaluer : ou analyser l'influence des attributs sur le système global en appliquant sur un cas d'étude. (chapitre 4)
3. Améliorer : proposer une méthode d'amélioration. (chapitre 4)

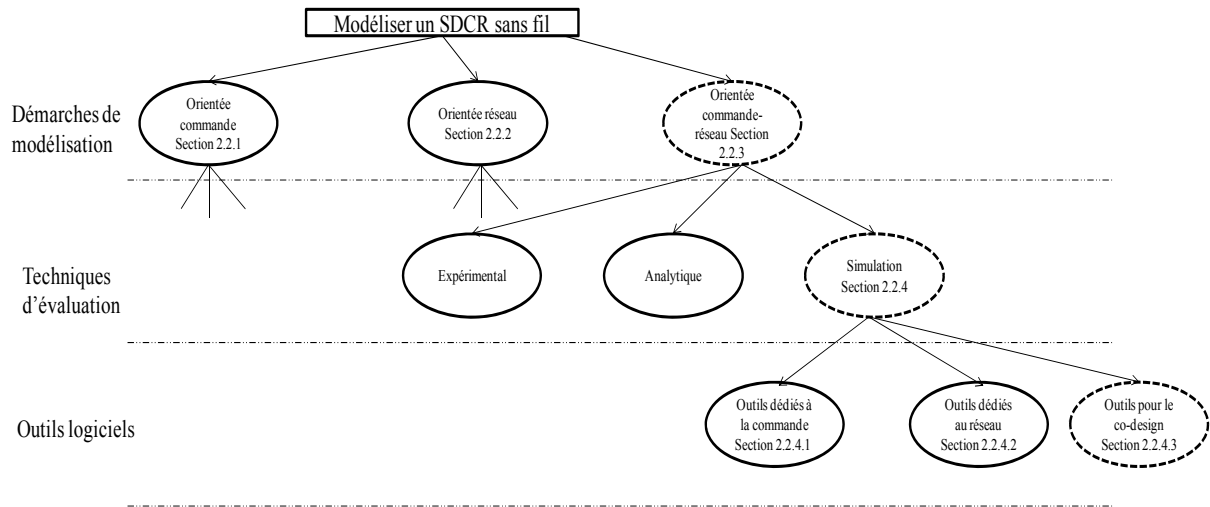
Dans ce chapitre, notre but est de dresser un état de l'art sur les moyens de modélisation d'un SDCR sans fil et les méthodes permettant d'augmenter leurs performances.

## 2.2 Modélisation

La Figure 2.1 détaille les différentes options recensées pour la modélisation des SDCR, les cercles pointillés sont les choix pris à chaque niveau.

Selon les objectifs de la modélisation (QdC, QdS), plusieurs approches sont envisageables (**Figure 2.1**, démarche de modélisation). Compte tenu de notre objectif qui consiste à prendre en compte le comportement de tous les équipements (réseau, PO et PC) d'un SDCR sans fil avec leurs attributs, nous avons retenu l'approche orientée commande-réseau. En ce qui concerne l'évaluation, plusieurs approches sont possibles, telles que l'expérimentation, l'évaluation analytique ou encore la simulation (Figure 2.1, Technique d'évaluation). L'approche retenue est l'approche par simulation au vu de son efficacité et de la complexité des modèles à évaluer. Enfin, il convient de sélectionner un outil logiciel capable de modéliser un SDCR en combinant une partie non déterministe (Réseau) et une partie déterministe (commande et partie opérative) (Figure 2.1, outils logiciels). Suite à l'étude des outils existants, deux outils ont retenu notre attention : l'un est dédié à la commande, MATLAB-Truetime ; et l'autre est dédié au réseau, OPNET. Ces deux outils étant dédiés à un domaine d'application spécifique, l'évolution vers un outil logiciel pour le co-design d'un

SDCR pourra se faire selon deux techniques : utilisation conjointe et intégrée de plusieurs outils ou bien intégration des modèles au sein d'un même outil.



**Figure 2.1 : Plan des choix pris dans ce chapitre**

Rappelons que les SDCR sans fil sont divisés en trois parties : la partie commande, la partie opérative et la partie réseau. Pour la modélisation, nous avons remarqué que les travaux existants sont classifiés selon trois catégories :

### **2.2.1 Première catégorie, orientée commande :**

Ces travaux se focalisent plutôt sur la qualité de contrôle (QdC) en étudiant les propriétés de la partie commande et de ses attributs (*Papi*). Ils se basent pour cela sur un modèle simplifié du comportement de la partie réseau. Après l'étape de modélisation, on passe à l'évaluation. Dans cette catégorie, nous pouvons citer trois approches : expérimentale, analytique et par simulation.

**Expérimentale :** (Addad, et al., 2008 a) construisent une plateforme pour analyser les variations de délai. Les résultats sont introduits dans un réseau de Pétri temporisé. Le réseau étudié est un réseau Ethernet où il n'y a pas de perte de paquets et le délai est dû juste au temps d'attente de paquets dans la file d'attente des commutateurs. Dans (Marsal, et al., 2006), les auteurs disent : «... le délai de filtrage et de traitement des trames dans les modules d'E/S, le délai de traversée d'un commutateur à vide, le délai de traitement des trames dans les coupleurs, ... seront déterminés à partir de documentations constructeur ou d'expérimentations ». Dans les réseaux sans fil, la perte de paquets est un facteur important et devra être prise en compte dans l'étude des SDCR.

**Analytique :** Dans (Addad, et al., 2008 b), les auteurs intègrent le comportement du réseau dans leur modèle de commande décrit en réseau de Pétri. Ils représentent le réseau par une simple place qui représente le délai. Ils supposent à l'avance que ce délai suit une loi connue.

Ils étudient le délai entre l'occurrence d'un événement et la réponse à celui-ci. Ces résultats sont comparés avec des simulations en utilisant les réseaux de Pétri colorés. Afin de réduire le délai maximal, ils montrent que la période de scrutation de la carte E/S *Pcarte* doit être multiple du cycle de l'API *Papi*.

Une autre méthode d'analyse déterministe consiste à utiliser le model checking. Rappelons que le model checking est un ensemble de techniques de vérification automatique de propriétés temporelles ou non, basées sur l'exploration de l'espace d'état. Plus précisément, un algorithme de model checking prend en entrée une abstraction du comportement du système réactif et une formule d'une certaine logique temporelle, et vérifie si l'abstraction satisfait ou non la formule. (Ruel, et al., 2008) se servent de ces techniques pour déterminer les délais maximaux de bout en bout. Un délai fixé arbitrairement est traduit sous la forme d'une propriété à vérifier. Il est ensuite augmenté ou diminué, de manière itérative, selon le succès ou l'échec de la preuve jusqu'à obtenir la borne. (Witsch, et al., 2006) traitent le problème de la même manière, il utilise le modèle checking afin de vérifier le comportement du système en temps réel. Le problème de cette étude est qu'ils ne prennent pas en compte les effets du réseau comme la perte des paquets.

D'autres approches prennent en compte les aspects stochastiques telles que celles proposées par (Greifeneder, et al., 2006) qui travaillent avec le model checking probabiliste (PMC) pour modéliser le réseau. Ces modèles sont implantés sur un model-checker appelé PRISM<sup>4</sup>. Ce travail fait l'hypothèse qu'il n'y a pas de perte de paquets durant la transmission des messages (dû aux collisions entre les paquets). Ils étudient l'influence du réseau sur une application académique. Dans ce travail, le réseau est considéré comme une boîte noire c'est-à-dire que la modélisation du réseau n'est pas détaillée.

Dans (Bordbar, et al., 2005), les comportements des différentes parties du système sont modélisés en utilisant des automates temporisés. Les auteurs présentent les délais introduits par le réseau comme des simples états. Ils considèrent que les valeurs du délai et de la gigue sont connues. L'outil UPPAAL est utilisé pour simuler des modèles écrits en XML.

### ***Simulation :***

Les travaux dans (Hu, et al., 2007) étudient la stabilité d'un système contrôlé en fonction des pertes de paquets induits par le réseau. Le problème se pose puisqu'ils considèrent que les pertes des paquets suivent une loi de distribution particulière. Dans ce cas, ils supposent que c'est la loi de Bernoulli.

---

<sup>4</sup> <http://www.prismmodelchecker.org/>

(Meunier, et al., 2007) modélisent des SDCR en utilisant l'outil CPNTool. De plus, ils évaluent la performance temporelle d'un SDCR. Par exemple la réactivité, en fonction de plusieurs attributs réseau et contrôleur.

*Dans cette catégorie, les travaux présentent le comportement de la partie commande d'une manière efficace avec les trois modèles (lecture, traitement et écriture). La période Papi est également prise en compte. Cependant, ces travaux ne représentent pas le comportement réel du réseau avec tous les attributs correspondants. Ils ne regardent pas le comportement aléatoire du réseau en termes de délai, perte des paquets et gigue. Ils essaient de simplifier le modèle du réseau en supposant que le réseau suivra un comportement bien défini*

### 2.2.2 Deuxième catégorie, orientée réseau

Ces travaux se focalisent plutôt sur la qualité de service (QoS) du réseau en considérant que la partie commande a peu d'influence sur le comportement global du système. De même que dans la catégorie commande, cette catégorie est évaluée par trois approches : expérimentale, analytique et par simulation:

**Expérimentale :** (Michaut, 2003), ainsi que (Jiang, et al., 2009) utilisent le réseau internet (avec le protocole UDP au niveau transport comme toujours pour les applications temporellement contraintes) entre un Maître et un esclave. L'esclave est un robot mobile qui envoie sa position et reçoit les ordres du Maître. Le Maître supporte le contrôleur : il mesure la QoS du réseau en utilisant des paquets estampillés (time-stamped packets). Ces mesures seront utilisées pour faire varier le gain du contrôleur.

**Analytique :**

**Déterministe :** La méthode déterministe permettant d'évaluer la QoS la plus répandue est le Network calculus (Cruz, et al., 2003), connue en français sous le nom de calcul réseau. Rappelons que le calcul réseau est basé sur l'algèbre (max, +) et permet de calculer les bornes maximales et minimales du délai (correspondant aux pires des cas) dans un réseau. Certains travaux utilisent cette approche (Georges, et al., 2005) pour estimer le délai maximal et minimal dans un réseau Ethernet. D'autres travaux tels que (Jasperneite, et al., 2002) travaillent sur la même approche, pour démontrer que l'utilisation du CSMA/CA peut être compatible avec les contraintes des systèmes temps réels.

**Probabiliste:** les travaux de (Miorandi, et al., 2004) utilisent la théorie des files d'attente pour modéliser un réseau sans fil. Pour réduire la complexité de ces équations, il néglige le temps d'attente aléatoire dû au *Backoff time*. Dans le but de valider cette méthode, les résultats analytiques sont comparés avec des simulations sur l'outil NS-2<sup>5</sup>(*Network Simulator*). Par

---

<sup>5</sup> [www.isi.edu/nsnam/ns/](http://www.isi.edu/nsnam/ns/)

contre, le délai introduit par la procédure du backoff a un effet considérable surtout dans les réseaux à charge importante. Aussi, le fait de négliger ce délai ne correspond pas à une modélisation réelle du réseau. Ce travail peut aussi être dans l'approche simulation puisque il y a une comparaison entre les résultats analytiques et simulations.

**Simulation :** Le travail dans (Karanam, et al., 2006) évalue la performance de l'IEEE 802.11e mode EDCA et HCCA dans un système contrôlé. Cette évaluation se fait sur le simulateur OPNET<sup>6</sup> (*Optimized Network Engeneering Tools*) en fonction du délai maximal induit par les deux modes. Dans une étude récente, (Boggia, et al., 2009) regardent l'impact des attributs MAC (*Nombre maximal de retransmission,...*) sur la performance d'un système contrôlé. Mais dans cette étude, la notion de contrôleur de type API n'existe pas. Ils utilisent l'outil Ns2 pour les simulations. (Xiangheng, et al., 2004) étudient l'influence des différents mécanisme d'accès au médium sur la stabilité du système. Il en déduit que plus le mécanisme d'accès est distribué et aléatoire, et plus la stabilité dans ces systèmes peut se dégrader. Plusieurs travaux (Juanole, 2002), (Kyung Chang, et al., 2002), (Neumann, 2007) traitent et évaluent la performance de ce système en utilisant le réseau Ethernet commuté (réseau Ethernet avec des commutateurs).

*Dans cette catégorie, les travaux analysent essentiellement la QoS de la partie réseau en particulier pour les réseaux dont le comportement est non déterministe. Cependant, ces études ne prennent pas en considération le comportement réel de la partie commande.*

### **2.2.3 Troisième catégorie, orientée commande-réseau,**

Dans cette catégorie, les travaux ont pour objectif de prendre en compte conjointement la QoS et la QdC des deux parties réseau et commande. (Branicky, et al., 2003) expliquent la complexité de cette étude qui prend en compte le comportement réel de chaque équipement dans un SDCR sans fil. Rappelons que ce comportement varie en fonction des attributs qui caractérisent ces équipements. Par la suite, ces attributs sont aussi considérés dans ces études.

De même que dans les catégories précédentes, cette catégorie est évaluée par trois approches : expérimentale, analytique et simulation:

A notre connaissance, il n'y a pas de travaux orientés commande-réseau en utilisant l'approche *analytique* au vu de la complexité des modèles impliqués. Concernant l'approche *expérimentale*, (Denis, et al., 2007) installent une plateforme pour présenter un système discret contrôlé en réseau utilisant Ethernet et mesurent le temps de réponse en fonction de la charge du réseau. Cette approche est couteuse surtout si le réseau étudié contient un grand nombre d'équipements.

---

<sup>6</sup> <http://www.opnet.com/>

**Simulation** : (Masri, et al., 2009) modélisent toutes les parties {PO, PC, Partie Réseau} en utilisant les réseaux de Petri à haut niveau. Les chercheurs (Anderson, et al., 2005), (Henriksson, et al., 2003) de l'université de LUND, Suède ont développé une librairie sur MATLAB appelé Truetime afin d'étudier l'influence des différentes entités du système contrôlé sur la performance globale.

*Cette catégorie est intéressante puisque elle vise toutes les parties du système en considération. Elle reste cependant peu abordée dans la mesure où les modèles nécessaires pour couvrir l'ensemble des parties réseau et commande et de leurs différents attributs sont complexes à obtenir et à analyser.*

Le Tableau 4 résume les travaux selon les trois catégories ainsi que leurs approches

Approches \ Catégories	orientée commande	orientée réseau	orientée commande-réseau
<b>Expérimentale</b>	(Addad, et al., 2008 a), (Marsal, et al., 2006)	(Jiang, et al., 2009)	(Denis, et al., 2007)
<b>Analytique</b>	(Addad, et al., 2008 b), (Ruel, et al., 2008), (Witsch, et al., 2006), (Greifeneder, et al., 2006), (Bordbar, et al., 2005)	(Georges, et al., 2005), (Jasperneite, et al., 2002), (Miorandi, et al., 2004)	
<b>Simulation</b>	(Hu, et al., 2007), (Meunier, et al., 2007)	(Karanam, et al., 2006), (Xiangheng, et al., 2004), (Boggia, et al., 2009)	(Branicky, et al., 2003), (Masri, et al., 2009), (Henriksson, et al., 2003), (Anderson, et al., 2005)

**Tableau 4 : Récapitulation de différentes approches et catégories**

Notre étude se situera dans la troisième catégorie (*orientée commande-réseau*).

Au niveau des trois approches d'évaluation présentées précédemment, nous remarquons que l'approche expérimentale est indispensable mais doit nécessairement être précédée d'une phase d'évaluation plus rapide et moins coûteuse (en particulier, pour les systèmes de grande dimension)

L'approche analytique est une approche efficace pour étudier le comportement du système dans des cas extrêmes (par exemple le calcul des pires cas). Cette approche peut nous amener à optimiser les attributs du système. Cependant, elle présente une complexité non négligeable lorsque l'on modélise l'ensemble des parties réseau et commande et de leurs différents attributs. Ce constat justifie l'utilisation de l'approche par simulation.

## 2.2.4 Approche par simulation

Le problème dans cette approche est de trouver l'outil logiciel qui permettra la modélisation et la simulation de l'intégralité d'un SDCR sans fil. (Colandairaj, et al., 2005) classent ces outils en deux grandes familles : les outils dédiés à la modélisation et l'évaluation de la commande et les outils dédiés à la modélisation et l'évaluation du réseau.

### 2.2.4.1 Outils logiciels dédiés à la modélisation et l'évaluation la commande

De nombreux outils sont dédiés à la modélisation et la simulation de la commande des SED comme par exemple MATLAB/Simulink<sup>7</sup>, CPNTools<sup>8</sup>, ControlBuild<sup>9</sup>, SCADE<sup>10</sup>, .... Ces outils présentent tous des similitudes quant à la modélisation utilisée, souvent basée sur un modèle à états (Réseau de Petri, Statecharts, Grafcet, ...) et sur les techniques de simulation. Dans la suite de cette section, nous avons choisi de nous focaliser sur l'outil logiciel MATLAB, en raison du nombre important de bibliothèques qu'il propose pour couvrir les différents aspects d'un SDCR sans fil, notamment les bibliothèques Truetime, Communications Toolbox, Real-Time Workshop, Simulink, ... MATLAB est un logiciel de calcul numérique produit par MathWorks. MATLAB a été conçu à la fin des années 1970. Il permet de réaliser des simulations numériques basées sur des algorithmes d'analyse numérique. Il peut donc être utilisé pour la résolution approchée d'équations différentielles, d'équations aux dérivées partielles ou de systèmes linéaires, etc.

D'après les créateurs de la boîte à outils Simulink, cette dernière est une plate-forme pour la simulation multi domaine et l'approche *Model-Based Design* des systèmes dynamiques. Il s'agit d'un environnement graphique interactif qui propose un ensemble personnalisable de blocs et peut être étendu pour des applications spécialisées. Pour modéliser la partie commande dans un système contrôlé en réseau, cette boîte Simulink fournit tous les outils nécessaires. Par contre, la modélisation de la partie réseau avec tous les attributs qui doivent entrer en considération n'est pas assurée.

### 2.2.4.2 Outils logiciels dédiés à la modélisation et l'évaluation du réseau

Dans le domaine de la recherche, plusieurs logiciels existent pour simuler un réseau. Les deux simulateurs les plus utilisés sont OPNET et NS-2.

---

<sup>7</sup> <http://www.mathworks.com/>

<sup>8</sup> <http://wiki.daimi.au.dk/cpntools/cpntools.wiki>

<sup>9</sup> <http://www.geensoft.com/en/article/controlbuild>

<sup>10</sup> <http://www.esterel-technologies.com/products/scade-suite/>

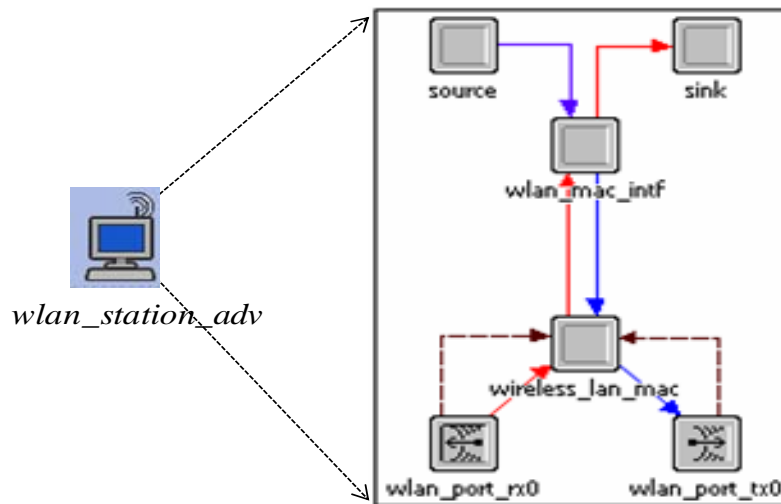
NS-2 est un « open source » outil accessible gratuitement. Il présente une difficulté due à l'absence d'interface graphique. De plus, sa librairie ne modélise pas tous les protocoles réseau.

OPNET est un environnement graphique utilisé dans la conception et l'étude des réseaux offrant une grande flexibilité. Il permet de travailler sur toutes les couches du modèle OSI en modélisant un grand nombre de protocoles et prenant en considération toutes les attributs qui spécifient chaque protocole. Sur cet outil, nous pouvons aussi modéliser des équipements mobiles en traçant leur chemin, en faisant varier leur vitesse de déplacement, ...

OPNET est commercialisé depuis 1986, il appartient à la famille de logiciels développés par la société MIL3. C'est un outil qui destine au début aux besoins militaires mais qui est actuellement devenu un produit commercial. Le laboratoire CRAN utilise une licence académique de ce logiciel. Notons aussi que plusieurs grands vendeurs de produits de communication comme Nokia, Ericsson, Cisco,... utilisent cet outil, et implémentent les modèles de leurs produits. Pour cela l'un des grands intérêts d'OPNET est de pouvoir simuler des produits réels qui existent dans un réseau réel. Les intérêts de ce logiciel ne s'arrêtent pas là. OPNET fournit aussi la possibilité d'implémenter de nouveaux algorithmes et de nouveaux modèles, notamment par l'utilisation du formalisme EFSM (Extended Finite State Machine), (Cheng, et al., 1993).

L'inconvénient d'OPNET réside dans le fait que c'est un simulateur purement réseau. En d'autres termes, il n'existe pas de bibliothèques représentant les modes de fonctionnement d'un API et dans lesquelles les modèles de commande seraient intégrés. Cependant pour tous les avantages précédemment énoncés, nous avons décidé de continuer à travailler avec cet outil.

Dans le but de modéliser les équipements qui forment un SDCR sans fil sur OPNET, nous devons choisir parmi la liste des stations et équipements fournis par la librairie OPNET. Chaque station dans cette librairie modélise différents protocoles et couches du modèle OSI et le choix sera fait selon l'intérêt de l'utilisateur. Dans notre cas, rappelons que nous nous intéressons juste aux trois couches (Physique, Mac et Application). Un type de station qui se trouve dans la librairie d'OPNET, qui correspond exactement à notre demande est la station « *wlan\_station\_adv* » (Figure 2.2).



**Figure 2.2 : L'architecture de *wlan\_station\_adv***

Cette station représente trois couches de modèle OSI : Physique, MAC et Application.

- La couche Physique est modélisée par deux processeurs : modèle de transmission (*wlan\_port\_tx0*) et modèle de réception (*wlan\_port\_rx0*). Ils permettent d'envoyer et de recevoir des signaux.
- La couche MAC est représentée par un seul processeur : *wireless\_lan\_mac*. Ce processeur modélise le comportement de l'accès au médium, en utilisant le standard IEEE 802.11 ou son extension IEEE 802.11e.
- La couche Application comporte deux processeurs, *source* et le *sink*. Le premier permet d'envoyer périodiquement des paquets à la couche MAC. Le deuxième reçoit immédiatement les paquets venant d'autres stations.

Notons aussi qu'il y a le processeur *wlan\_mac\_intf* qui permet de lier la couche Application à la couche MAC. Chaque processeur est modélisé en utilisant des EFSM, qui seront détaillés dans le chapitre suivant.

Plusieurs attributs peuvent être gérés dans cette station :

- *Caractéristiques de la couche physique* : 802.11a, 802.11b, ...
- *Adresse du destinataire* : l'adresse destinataire qui va recevoir les paquets. Notons bien que les paquets peuvent être envoyés en diffusion. Ce champ sera choisi « broadcast ». Nous avons aussi l'option que le destinataire soit choisi de manière aléatoire.
- *Taille des paquets* : la taille des paquets est définie au niveau de la couche Application
- *Période de livraison des paquets à la couche MAC* ou *Pcarte*
- *Traffic Type of Service* : indique la classe de service affectée aux paquets générés par une source. Dans le cas de 802.11e, ça sera les huit classes de priorité.
- *Data rate (Débit)*: dépend du type de la couche physique en Mbit/s
- *PCF-attributs* : Spécifie les attributs MAC 802.11 quand le mode PCF est utilisé

- *HCF-attributes* : Indique si les facilités introduites par le mode HCF (EDCA mode) sont supportées

### 2.2.4.3 Outils logiciels pour le co-design

Les outils présentés précédemment sont dédiés à un domaine spécifique (commande ou réseau) et basés sur des modélisations hétérogènes pour la représentation des aspects déterministes de la commande et stochastiques des réseaux.

*Partie réseau* : cette partie modélise tous les états d'un réseau. Durant une transmission, les paquets passent à travers les différentes couches OSI de la source vers le destinataire en traversant le médium. Tous les comportements des couches OSI et du médium doivent être modélisés afin de bien représenter le réseau. Rappelons que dans 802.11, les stations utilisent l'algorithme CSMA/CA pour accéder au médium. Cet algorithme prend un temps d'attente aléatoire *backoff time* dans le cas où le médium est occupé. En plus, les pertes de paquets proviennent des collisions et des perturbations inconnues de l'environnement du réseau. Par conséquence, le modèle de l'IEEE 802.11 est non déterministe.

*Partie commande/opérative* : ces parties sont déterministes. En d'autres termes, pour un état donné de la PC/PO, pour une même séquence des variables d'entrée, nous aurons toujours la même séquence des variables de sortie. Les modèles de cette partie évoluent avec certaines propriétés concernant la sécurité et le temps de réponse en utilisant des modèles à états et des transitions logiques.

***En résumé, un problème se pose au niveau cohabitation/coopération de ces parties (déterministe et non déterministe). L'intégration de ces modèles dans un ou plusieurs outils peut se faire de deux manières différentes : connexion entre deux outils ou intégration des modèles au sein d'un même outil.***

#### 2.2.4.3.1 Interconnexion entre outils

Cette méthode consiste à utiliser deux outils afin de représenter les systèmes contrôlés. Elle peut poser des problèmes au niveau synchronisation des traces d'exécutions sur les deux outils durant les simulations.

(Harding, et al., 2007) utilisent MATLAB et OPNET en parallèle pour modéliser un SCR sans fil. En d'autres termes, les modèles mathématiques (contrôleur et partie opérative) sont liés par des interfaces de type TCP/IP *sockets*. Les simulations sont faites sur des petits réseaux dont l'influence des paramètres du réseau (délai, perte de paquets) est négligeable. De même (Soglo, et al., 2006) simulent sous MATLAB la partie commande et sous NS-2 la partie réseau. Pour prouver l'efficacité de ces méthodes, elles doivent être testées sur des systèmes complexes.

(Shahidul Hasan, et al., 2005) utilisent deux outils séparément : NS-2 et MATLAB. Ils simulent le réseau indépendamment du système contrôlé. Les simulations fournissent des informations concernant les variations du délai et la perte de paquets. Par la suite, ces résultats sont implémentés dans un système contrôlé en réseau modélisé sur MATLAB. Il a prouvé que le système global reste stable même après un taux de perte égal à 75%. Cette méthode n'est pas pratique et spécialement si on veut étudier plusieurs attributs.

**Même si les techniques de synchronisation entre les outils proposées semblent efficaces, elles demeurent peu formalisées et n'offrent pas de garanties sur la maîtrise du temps et des traces d'exécutions de chaque outil.**

#### *2.2.4.3.2 Intégration des modèles au sein d'un même outil*

Pour éviter l'utilisation d'outils différents et interconnectés, une autre idée consiste à intégrer les modèles relatifs à un domaine d'application (commande ou réseau) dans un outil non prévu à cet effet à l'origine.

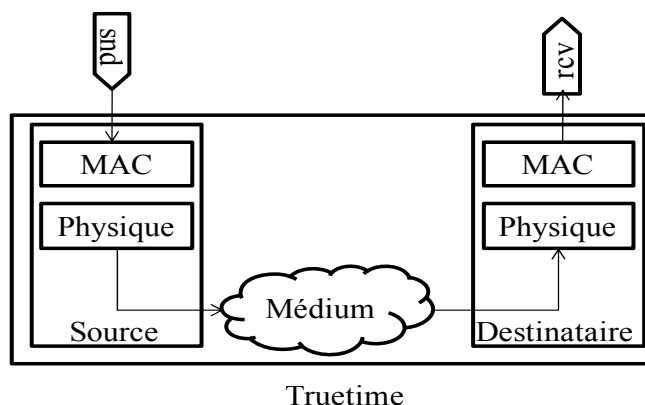
Une solution consiste à modéliser le comportement du réseau dans des outils dédiés à la commande. (Colandairaj, et al., 2005) intègrent un simulateur de réseau sur MATLAB en utilisant le langage C Mex. L'efficacité et la fiabilité de ce code sont inconnues. En plus, ce travail considère que les files d'attente dans les stations de transmission sont de taille infinie. Cette hypothèse peut être vraie dans un réseau de petite charge dont les paquets ne restent pas beaucoup dans les files d'attente. Cependant, ces paquets sont perdus à cause du débordement des files d'attente dans des réseaux chargés. (Masri, et al., 2009) essaient d'utiliser un outil à événement discret, Renew. Ils implémentent le modèle du réseau en utilisant les réseaux de Pétri de haut niveau. Son modèle de réseau est comparé avec des simulations sur Ns-2, puis ils implantent les PO, PC. Ils étudient la performance du système en fonction de différents types de réseau (Wifi, Ethernet,..).

Pour éviter le développement fastidieux d'une librairie réseau propre, il est peut être intéressant d'utiliser des outils existants, comme par exemple MATLAB, qui peut être complété par une librairie pour modéliser le réseau : Truetime<sup>11</sup>.

Cette librairie permet de représenter uniquement les couches MAC et physique des stations sources et destinataires. La Figure 2.3 présente l'architecture de cette librairie. Les paquets de la station source arrivant via le label « snd » sont traités par la couche MAC et physique afin d'être envoyés à la station destinataire. A son tour, elle traite ses paquets à travers les couches physiques et MAC. Les paquets sont alors visibles via « rev ». Par conséquent, l'utilisation de cette librairie nous oblige à modéliser aussi les couches réseaux absentes comme la couche Application.

---

<sup>11</sup> <http://www.control.lth.se/truetime>



**Figure 2.3 : Architecture de la librairie Truetime**

Une autre solution consiste à modéliser le comportement de la commande dans des outils dédiés au réseau tels qu'OPNET ou NS-2. L'idée est d'utiliser les formalismes EFSM ou C Mex habituellement exploités pour décrire de nouveaux protocoles, pour représenter au dessus de la couche application, la commande d'un SCR. En ce sens, (Kubler, et al., 2010) intègrent des modèles de commande de systèmes continus dans OPNET. Cette approche s'est révélée efficace et mériterait d'être évaluée pour la modélisation d'une commande de SED.

**Ces deux types d'outils montrent une efficacité pour modéliser une partie d'un SDCR sans fil. Ce constat justifie le choix de ce type d'approche pour notre modélisation d'un SDCR sans fil. Nous montrerons dans le chapitre 3 comment modéliser les parties non couvertes par ces outils (partie réseau pour les outils orientés commande et partie commande pour les outils orientés réseau).**

## 2.3 Amélioration d'un système contrôlé en réseau

De nombreux travaux ont été menés pour améliorer les performances des SCR. Nous les présentons en trois approches. Ils sont groupés selon la partie du système développée.

### 2.3.1 Première approche, orientée QdC

Elle consiste à adapter la commande en fonction de la QdS réseau.

(Nilsson, et al., 1998) développent un algorithme de contrôle en utilisant le contrôle LQG (Linear Quadratic Gaussian) afin de compenser la variation de délai du réseau. Dans ces simulations, ils supposent que la loi de distribution de délai est connue d'avance. (Chan, et al., 1995) proposent de créer un état de prédiction dans la partie commande. Cet état utilise la taille des files d'attente dans les équipements pour estimer le délai du au réseau. Par conséquent, il pourra donc avancer ou retarder la date de transmission des paquets.

Quelques travaux essaient d'estimer le délai et le nombre de paquets perdus afin d'adapter le contrôleur : (Zhang, et al., 2006) proposent notamment un estimateur de délai. Cet estimateur

fonctionne de manière dynamique en utilisant l'historique du délai et l'algorithme qu'il a défini. Par mesure de précaution, le délai estimé sera utilisé par le contrôleur en utilisant un contrôleur LQR (Linear Quadratic Regulator). (Mao, et al., 2007) utilisent une méthode pour tolérer les pertes dues au réseau dans des systèmes contrôlés. Cette méthode est basée sur le filtre de Kalman.

(Michaut, 2003) vise à doter les applications distribuées de moyens leur permettant de pallier l'insuffisance du système de communication en adaptant leur exécution. Il crée des blocs de traitement pour l'application. Chaque bloc est associé à un niveau donné de QoS. Selon les performances du réseau, l'exécution de l'application varie dynamiquement entre ces blocs. Par exemple : on associe à quatre blocs de QoS (définies par rapport au temps mesuré pour échanger des informations entre un émetteur et un récepteur) quatre gains d'un contrôleur proportionnel.

**Conclusion :** *Ces études sont particulièrement adaptées lorsque la QoS du réseau n'est pas gérable c'est à dire lorsque le réseau est non commandable.*

### 2.3.2 Deuxième approche, orientée QoS

Elle consiste à adapter le réseau pour améliorer sa propre QoS:

Cette approche gère et répartit les ressources du réseau dans le but de diminuer les effets du réseau en termes de délai, perte de paquets et de gigue. (Kim, et al., 2009) proposent une méthode pour calculer le délai maximum autorisé (MADB), qui garantit une stabilité du système. Les auteurs utilisent les valeurs de MADB pour varier la période d'échantillonnage afin de répartir la bande passante de manière équitable. Cette méthode peut gérer trois types de données: trafics temps réel, périodiques et apériodiques. Dans ce travail, il n'y a pas de perte de paquets. Pour échanger des données entre les équipements, le *factory instrumental protocol* (FIP) est utilisé.

Pour faire face aux défauts dus à la mobilité, aux interférences et à l'atténuation des signaux, (Yang, et al., 2009) proposent d'adapter la modulation qui est codée en fonction de la dégradation de la situation. Rappelons que l'adaptation de modulation est une technique permettant d'améliorer l'énergie émise, donc le débit transmis dans les canaux.

La technologie de communication utilisée dans (Boughanmi, et al., 2009) est ZigBee/IEEE 802.15.4. Plusieurs solutions sont mises en œuvre pour améliorer la stabilité du système. La première consiste à donner une priorité d'accès au médium pour certains trafics. Quant à la seconde, elle propose d'utiliser le mécanisme Blackburst qui consiste à réserver le médium pour la station envoyant le paquet Blackburst le plus long. La longueur du paquet Blackburst dépend de la priorité du trafic. La dernière possibilité consiste à utiliser le mécanisme GTS qui réserve un temps de transmission pour les stations qui veulent envoyer des données.

Les travaux de (Tipsuwan, et al., 2009) proposent une répartition équitable de la bande passante du réseau (faisant varier  $P_{carte}$ ) entre les équipements du système. Son algorithme vise une application continue. L'efficacité de sa méthode sur des applications discrètes n'est pas garantie, surtout que la variation de  $P_{carte}$  peut aboutir à des pertes d'événements furtifs dans les applications discrètes.

Plusieurs laboratoires ont travaillé sur cette problématique en particulier le laboratoire CRAN. Nous pouvons citer : les travaux de (Georges, et al., 2007) qui portent sur la construction d'un modèle fonctionnel et analytique du réseau Ethernet basé sur la théorie du calcul réseau. Ce modèle sera utilisé pour majorer les temps de traversée du réseau. Les travaux de (Rondeau, et al., 2001) et (Krommenaker, 2002) ont porté sur la reconfiguration de la topologie des réseaux Ethernet afin de respecter les contraintes temporelles prédéfinies.

Au niveau communication sans fil, nous pouvons encore citer dans le CRAN. Les travaux de (Salles, et al., 2007) essaient d'améliorer la performance des réseaux ZigBee. Cependant, nous focalisons notre travail sur le Wifi. L'amélioration du comportement de ce type de réseau se fait en utilisant l'extension de 802.11, IEEE 802.11e. Ce standard a été créé aussi pour limiter les inconvénients de 802.11. IEEE 802.11e est développée de telle sorte qu'elle supporte la QoS par la différenciation des classes de services au niveau couche MAC. Cette différenciation est effectuée de façon à ce que la couche MAC puisse acheminer les trafics temps réel et multimédia vers des files de haute priorité. Les trafics traditionnels sont alors dirigés vers des files de basse priorité. Théoriquement, le temps d'attente dans les files de haute priorité est inférieur à ceux dans les files de basse priorité.

Malheureusement, cette technique n'est pas toujours efficace. En effet, 802.11e a montré un meilleur comportement dans des cas très précis. Mais, par contre, dans des situations critiques comme par exemple une saturation, 802.11e montre des dégradations importantes de la QoS demandée par les applications multimédia. Le prochain paragraphe en témoigne.

### **2.3.2.1 Etudes du comportement d'EDCA**

(Lu-ming, et al., 2007) réalisent des simulations montrant l'influence de la charge réseau sur des trafics de priorités différentes. Deux scénarios sont simulés. Le premier simule un réseau avec des trafics de priorités 3 et 1. La conclusion est la suivante : l'augmentation de la charge augmente les débits des trafics de la priorité 3 mais diminue ceux de priorité 1. Ainsi, les délais des trafics de priorité 3 augmentent exponentiellement quand la charge du réseau augmente. Le deuxième scénario introduit un trafic d'urgence. Dans un réseau surchargé, les simulations montrent que les délais du trafic d'urgence sont identiques à ceux des trafics normaux.

De la même façon, (Grilo, et al., 2002) comparent la performance du réseau avec (mode EDCA) et sans priorité (mode DCF). En surveillant les flux montants et descendants, ils ont montré que dans le cas où la charge du réseau est importante, les flux montants avec priorité

présentent le même comportement que ceux sans priorité. Les mêmes observations sont faites avec les flux descendants. Les auteurs expliquent ce phénomène pour les flux descendants par le fait que ceux-ci doivent partager les ressources d'une manière équitable entre tous les équipements du réseau.

(Ni, 2005) a montré, par ses simulations, que plus le flux a une haute priorité plus son délai de bout en bout est petit. Cela s'explique par la domination du médium des flux à haute priorité. D'autre part, les flux à basse priorité souffriront d'un manque d'accès au canal. Ce manque peut dégrader la qualité des trafics. Le travail de (Cowling, et al., 2004) compare des flux avec des besoins en QoS de différentes caractéristiques, il montre que 802.11e est mieux adapté aux flux vidéo à un débit constant que les flux à débit variable.

La conclusion de ces travaux est que l'utilisation de priorités dans un réseau peut conduire à des effets négatifs pour les trafics de basse priorité, même arriver à une famine pour accéder au médium.

### **2.3.2.2 Proposer de nouveaux algorithmes pour le 802.11e pour améliorer sa performance :**

Le travail de (Varposhti, et al., 2009) propose une modification d'accès au médium. Sa méthode porte le nom *Collision Avoidance with Fading Detection* (CAFD). En cas d'échec de transmission, la station émettrice attend un certain temps au lieu de redoubler sa fenêtre de contention (CW). Ce temps d'attente est calculé en fonction de la probabilité de perte de paquets, de l'énergie de transmission des signaux, de l'énergie moyenne de réception,... (Carlson, et al., 2005) présentent son nouvel algorithme appelé *Distributed end-to-end Allocation of time slots for Real-time traffic* (DARE). Cet algorithme est basé sur la réservation du chemin (de l'émetteur vers le récepteur en passant par tous les intermédiaires). Il utilise des paquets de contrôle spéciaux pour maintenir et réparer ces chemins. Une comparaison de cet algorithme avec le 802.11e-mode EDCA montre son efficacité.

Dans le but de diminuer les collisions entre les flux, (Lin, et al., 2007) proposent de réinitialiser le *backoff time* à chaque fois que le canal est occupé. Son algorithme est nommé M-EDCF (Modified-EDCF). Les auteurs se placent dans un réseau à charge élevée et avec des trafics de différentes priorités. Les flux à petite priorité auront moins d'accès au médium donc l'écoulement de leur *backoff time* sera arrêté plusieurs fois. Par conséquent, il y aura plus de possibilités d'avoir les mêmes *backoff time* pour des trafics de priorité haute et basse. Ainsi, des collisions peuvent être induites. Pour éviter cette situation, les auteurs proposent que le *backoff time* soit décrémenté en continu ou qu'il soit réinitialisé à chaque fois qu'il est arrêté.

(Villalón, et al., 2008) expliquent que le délai majeur est dû au paramètre AIFS(AC) puisque la station doit attendre ce temps avant de recommencer de décrémenter son *backoff time*. Pour cela, les auteurs inventent un autre paramètre nommé BIFS. Ce paramètre porte des valeurs plus petites que celle d'AIFS(AC) normalisé. Des simulations montrent l'efficacité de cet

algorithmes. (Romdhani, et al., 2003) proposent une méthode d'ajustement lente de la fenêtre de contention après chaque transmission au lieu de la remettre à CWmin. Ce changement se fait en fonction des conditions du réseau et de la priorité du flux. Cette méthode a permis une meilleure différenciation de services.

L'idée de (Li, et al., 2006) est de faire varier la priorité de chaque flux selon le délai de bout en bout demandé, l'état de l'émetteur et le temps passé par la trame dans le réseau. (Zhu, et al., 2008) et (Pong, et al., 2003) proposent des algorithmes de permission des flux: (Zhu, et al., 2008) essaie d'améliorer la performance du réseau dans les deux modes (EDCA et HCCA). Pour cela, il propose une couche MAC intelligente qui favorise le passage dynamique d'un mode à autre en fonction de l'état du réseau.

(Xi, et al., 2008) proposent de changer l'architecture de la couche MAC. Comme nous l'avons déjà expliqué précédemment, la couche MAC est formée de quatre files d'attente. Chaque paquet venant de la couche supérieure est dirigé vers une file selon sa priorité. Rappelons aussi que chaque file fonctionne indépendamment de la file voisine. Cette architecture est appelée *Multiple state machine* (MSM), Figure 2.4 a). Les auteurs proposent de la changer en *Local Scheduler Multi Flow* (LSMF), Figure 2.4 b). La couche MAC sera alors formée d'une seule file d'attente, cette file changeant de priorité selon la priorité du paquet reçu. Un ordonnanceur des files d'attente est mis entre la couche application et la couche MAC. Cet ordonnanceur est formé de quatre files d'attente qui représentent les quatre priorités des flux et un contrôleur. Il reçoit les paquets de la couche application et les met dans les files d'attente correspondant à leurs priorités. Puis, le contrôleur décide quel paquet sera envoyé à la couche MAC. L'algorithme utilisé pour prendre la décision peut être Round-Robin, Weighted Fair Queuing (WFQ),....

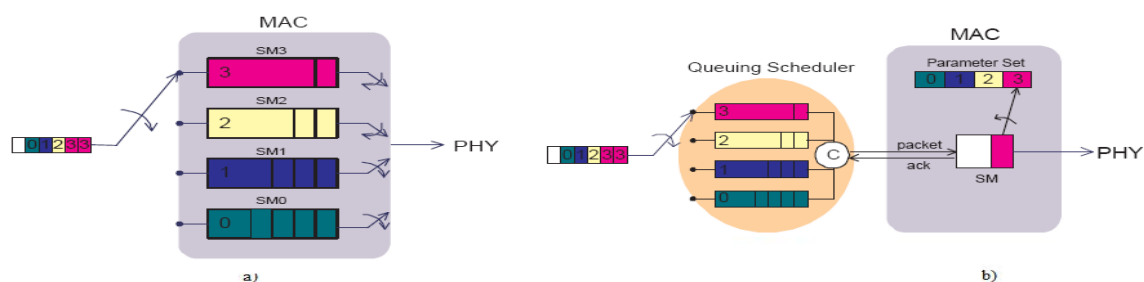


Figure 2.4 : a) Architecture de la couche MAC standardisée et b) proposé par (Xi, et al., 2008)

**Conclusion :** Ces études ne prennent pas en considération les besoins de la partie commande.

### 2.3.3 Troisième approche, orientée QoS- QdC,co-design:

Elle consiste à combiner les deux approches précédemment abordées.

(Liu, et al., 2004) proposent d'ajouter au modèle OSI classique une couche supplémentaire appelée *cross layer* qui prend en compte l'ensemble des paramètres commandes et réseau. La *cross layer* combine les différentes couches, afin d'optimiser leurs attributs, comme par exemple : *Pcarte*, le protocole MAC utilisé,... en considérant la performance du système.

(Branicky, et al., 2002) proposent de gérer les temps de transmissions de chaque équipement en leur affectant des priorités. Ces priorités dépendent essentiellement de la stabilité du système.

(Hongbo, et al., 2008) présentent une nouvelle structure pour la partie contrôle. Le contrôleur est alors équipé d'un régulateur. Ce dernier optimise la période d'échantillonnage *Pcarte* et les paramètres du contrôleur. Le régulateur utilise un algorithme à estimation de distribution. Cet algorithme est détaillé dans (Mühlenbein, et al., 1996). (Colandairaj, et al., 2006) font varier *Pcarte* en fonction de la Qualité de service du réseau et de la Qualité de performance de la commande recommandée. Ils proposent plusieurs algorithmes pour gérer l'adaptation de *Pcarte*.

(Brahimi, 2007) modélisent un SCR en utilisant des Réseaux de Petri de haut de niveau, en intégrant au modèle Ethernet Commuté, l'environnement applicatif : Contrôleur, Process... De plus, ils ont proposé des stratégies pour commander le réseau de façon à adapter sa QoS en regard de la QdC requise par l'application.

***Conclusion : L'inconvénient de tous ces travaux est qu'ils prennent des hypothèses réductrices afin de faciliter la modélisation du système. Par exemple : file d'attente de transmission avec une capacité d'un paquet, absence de perte de paquets. Ces hypothèses font que le comportement du modèle du système diffère de son comportement réel.***

L'approche co-design est intéressante mais il faut la conduire avec une représentation fine du système à commander et du système de communication.

Ces trois approches différentes permettent d'avoir une vision globale des moyens d'améliorer la performance. L'analyse de ces différentes études nous a permis de constater que :

1. Les algorithmes proposés présentent une complexité non négligeable, et demandent des temps d'exécution importants pour contrôler le système. Ces temps peuvent être des grands inconvénients dans des applications industrielles.
2. Ces algorithmes ne sont pas testés sur des applications à événements discrets. Par conséquent, leur efficacité n'est pas assurée.
3. L'efficacité de ces travaux n'est pas aussi assurée, puisque les auteurs prennent des hypothèses réductrices afin de réduire la difficulté de modélisation de ces systèmes. Ce qui fait que le comportement des modèles proposés diffère du comportement réel du système.

4. L'idée de varier la priorité des flux selon les besoins de l'application ((Li, et al., 2006)), peut être un champ intéressant surtout pour les applications à événements discrets.

Conclusion pour l'utilisation d'IEEE 802.11e EDCA :

5. Eviter les situations de famine de certains flux tout en bénéficiant des avantages du 802.11e mode EDCA
6. La majorité des algorithmes réseaux ne peut pas être implémenté dans les cartes sans fil qui se trouvent sur le marché actuel. Donc, elles restent des propositions théoriques. Pour cela, notre intérêt est de proposer une méthode acceptée par le standard et qui pourra être facilement intégrée dans les cartes sans fil.

Notre but est de pouvoir proposer une méthode de Co-conception (QdS-QdC), qui améliore la performance d'un SDCR sans fil. Cette méthode doit prendre en compte la QdS ainsi que QdC. Cette méthode sera détaillée dans les chapitres 3 et 4.

## **2.4 Conclusion du chapitre**

L'étude de l'impact du réseau d'un SDCR sans fil suggère une modélisation de différentes parties de ce système. Différentes approches sont exploitées dans ce chapitre pour répondre à cette demande. Au vu de la complexité des approches expérimentales et analytiques, l'approche par la simulation est choisie. Dans celle-ci, le problème repose sur le choix d'un simulateur d'autant plus que la nature des modèles du SDCR sans fil peut être différente (déterminisme/non déterminisme). Nous identifions deux outils : le premier, Matlab-Truetime, orienté « commande », présente une modélisation grossière du réseau. Le deuxième outil, OPNET orienté « réseau », présente des difficultés d'implémentation des modèles de commande.

En plus, une étude des travaux précédents est faite pour améliorer la performance des systèmes. Cette étude montre que les méthodes proposées ne sont pas toutes adaptables au SDCR sans fil.

Dans le chapitre suivant, nous allons détailler plus précisément chaque outil et suggérer des propositions afin de pouvoir modéliser un SDCR sans fil sur chacun de ces outils.



## Chapitre 3

# **Outils pour modéliser un SDCR sans fil**

### 3.1 Introduction :

Dans le chapitre précédent, deux outils sont identifiés pour modéliser un SDCR sans fil. Le premier est plutôt orienté « commande », MATLAB avec la boîte à outil Truetime; l'autre plutôt orienté « réseau », OPNET. Dans ce chapitre, nous allons exploiter ces deux outils afin de choisir celui qui modélise au mieux le comportement d'un SDCR sans fil.

Pour MATLAB-Truetime, des modifications seront apportées dans cette librairie pour qu'elle représente mieux la partie réseau. Dans la suite, pour différencier la bibliothèque modifiée, nous utiliserons la notation : *MATLAB-Truetime modifiée*.

Les modèles dans OPNET seront étudiés dans le but de voir la possibilité d'implémenter la partie Commande/PO dans cet outil. Enfin, une comparaison entre les deux outils sera faite.

Cette comparaison est divisée en deux parties (Figure 3.1):

- Comparaison au niveau réseau, sans implémentation d'un système contrôlé. Dans cette partie, seuls les émetteurs et les récepteurs de trafics sont présentés. La comparaison porte sur des paramètres réseau (délai minimal, maximal,...).
- Comparaison de ces outils sur un cas d'étude qui présente un SDCR sans fil. Dans cette partie, nous allons implémenter la partie commande/PO dans les deux outils. Cette comparaison porte sur le temps de réactivité de la PO en fonction des commandes envoyées par la PC.

Cette comparaison a trois buts principaux:

- Garantir la pertinence des modifications faites sur MATLAB-Truetime vis-à-vis d'OPNET considéré comme une référence par la communauté réseau.
- Epruver la faisabilité de l'implémentation de la partie Commande / PO sur OPNET.
- Montrer les limitations et également les avantages de chaque outil pour la modélisation conjointe de la commande et du réseau.

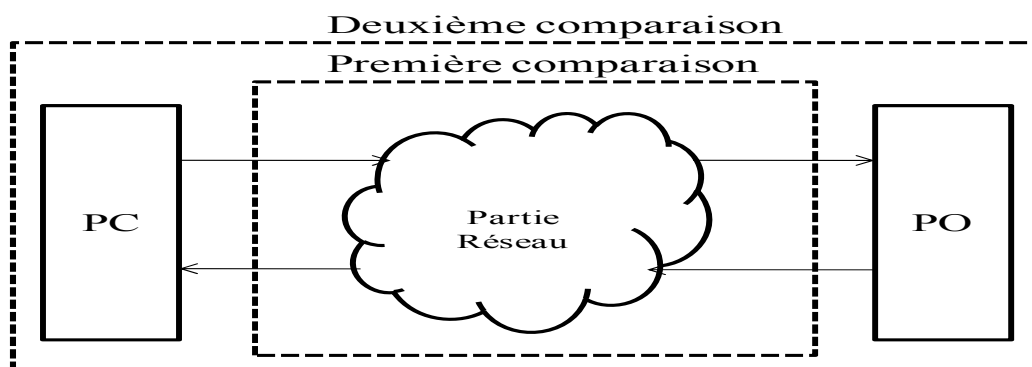


Figure 3.1 : les deux comparaisons faites

### 3.2 Outil orienté « commande » : Matlab-Truetime

Rappelons que MATLAB est orienté « commande ». Il est largement connu et utilisé dans la communauté « automatique » pour simuler et calculer des modèles mathématiques de systèmes dynamiques. Cet outil présente des lacunes pour simuler les réseaux. La modélisation sur MATLAB des modèles déterministes (Partie Commande/PO) se fait par des outils standardisés comme par exemple Simulink, Statechart (Harel, 1987). Par contre, la partie réseau non déterministe d'un SDCR sans fil est modélisée par la librairie Truetime.

Cette librairie est développée par l'université de LUND en utilisant le C++ Mex. Elle supporte deux types de réseaux sans fil: IEEE 802.11b/g (WLAN) et IEEE 802.15.4 (ZigBee). Elle est toujours en cours de développement, plusieurs versions sont proposées, la dernière date en juillet 2010, version 2.0 beta 6. Cette librairie peut être téléchargée gratuitement sur le site l'université de Lund.

La Figure 3.2 montre :

-3 entrées : « snd1 » est le label du flux entrant de la station source, x et y représentent les coordonnées des stations source et destinataire.

- 3 sorties : « rcv1 » est le label du flux sortant du réseau, la sortie « schedule » est faite pour voir la date d'entrée et de sortie de chaque paquet et finalement la sortie « P » concerne l'énergie consommée dans les nœuds (non utilisé dans notre cas).

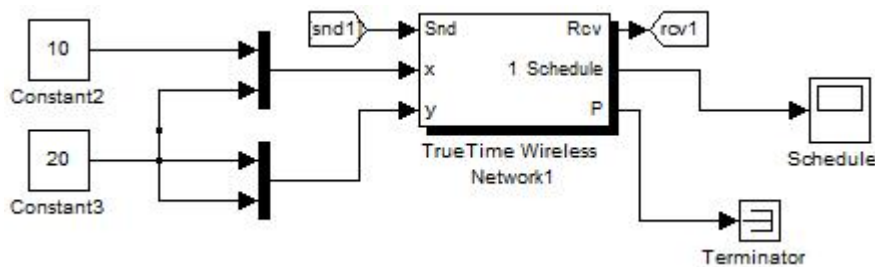
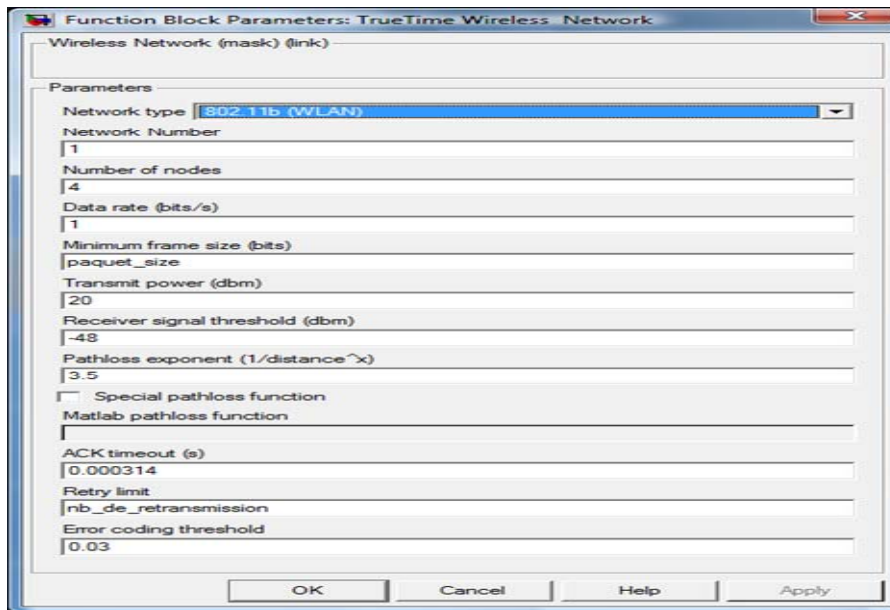


Figure 3.2 : Librairie Truetime

Cette librairie simule les réseaux sans fil en prenant en considération les différents attributs comme le *Nombre maximal de retransmission* des paquets, *Débit*, *la taille des paquets* et *Acktimeout* (le temps attendu par l'émetteur pour recevoir le paquet d'acquittement avant d'envoyer le paquet de nouveau) (voir Figure 3.3).



**Figure 3.3 : Les différents attributs manipulables sur Truetime**

Rapidement, nous remarquons que cette librairie offre une modélisation très grossière du réseau : le comportement de cette librairie est loin du comportement réel d'un réseau d'où son inconvénient majeur. De plus, dans cette librairie, le standard IEEE 802.11e n'est pas implémenté. Nous allons mettre en évidence ces écarts et faire des modifications afin de la rendre plus fiable. Pour cela, nous avons dû étudier la manière dont elle est codée et ajouter les modifications nécessaires afin de rendre son comportement le plus proche du standard. La librairie Truetime avec les modifications sera appelé *Truetime modifiée*.

Le tableau suivant présente les différents items qui ont été modifiés :

- Compléments pour représenter plus précisément le standard 802.11,
- Compléments pour prendre en compte la gestion des priorités introduite dans 802.11e.

La première colonne « **Modification** » présente les modifications qui sont faites. Par exemple, l'acquiescement, dans la librairie Truetime existante (non modifiée) n'existe pas. C'est pourquoi, nous intégrons le temps nécessaire pour envoyer une trame d'acquiescement ainsi que le temps SIFS entre la trame envoyée et sa trame d'acquiescement. Ce temps est ajouté lorsque la trame est bien reçue par le récepteur. La deuxième colonne présente le « **standard** » : ce dernier est notre référence. Cette colonne contient donc des parties directement issues des normes (IEEE 802.11, 1999) ou (IEEE 802.11e, 2005). La troisième colonne « *TrueTime modifiée* » détaille les modifications codées.

<b>Modification</b>	<b>Standard</b>	<b>Truetime</b>	<b>TrueTime modifiée</b>
<b>La variation de CW en fonction du nombre maximal de retransmission des paquets dans la fonction Backoff time</b>	<p>“If the backoff procedure is invoked because of a failure event [either reason c) or d) above], the value of CW[AC] shall be updated as follows before invoking the backoff procedure:</p> <p>b) Otherwise,</p> <p>1) If CW[AC] is less than CWmax[AC], CW[AC] shall be set to the value (CW[AC] + 1)*2 – 1.”</p>	<p>la variation de CW est définie par la variable: <i>collisionWindow</i> qui est égale <math>(1 \llcorner (\text{nbr} + \text{CWMIN}_{802\_11} - 1)) - 1</math>. dont nbr, <math>\text{CWMIN}_{802\_11}</math> représente le nombre maximal de retransmission et la valeur minimale de CW (CWmin) respectivement. Cette équation ne conserve pas l’ancienne valeur de <i>collisionWindow</i> afin de l’incrémenter en cas de retransmission</p>	<p>L’incrémentation de valeur CW pour chaque retransmission vérifie le standard en conservant l’ancienne valeur de <i>collisionWindow</i></p>
<b>Acquittement</b>	<p>“The reception of some frames, as described in 9.7, 9.2.8, and 9.3.3.4, requires the receiving STA to respond with an acknowledgment, generally an ACK frame, if the FCS of the received frame is correct. This technique is known as positive acknowledgment. Lack of reception of an expected ACK frame indicates to the source STA that an error has occurred”</p>	Pas de notion d’acquittement	<p>Insertion du temps (SIFS+Tack) nécessaire pour envoyer l’acquittement à l’émetteur en cas où le récepteur a bien reçu le paquet</p>
<b>Notion du bruit</b>	<p>It is a signal coming from a device outside the network. This signal may disturb the transmitted packet between the network devices.</p>	Pas de taux du bruit	<p>Le bruit intégré dans <i>TrueTime modifiée</i> est capable d’attaquer les paquets qui se propagent dans le medium et de les rendre incompréhensibles</p>
<b>Taille des paquets</b>	<p>“The transmitted value shall be determined from the LENGTH parameter in the TXVECTOR issued with the PHY-TXSTART.request primitive described in 12.3.5.4. The length field provided in the TXVECTOR is in bytes”</p>	La taille des paquets est en bit	La taille des paquets est en octets
<b>Unité du débit</b>	<p>This is an example from the standard document</p> <p>“This field indicates the data rate of the whitened PSDU from 1 Mbit/s to 4.5 Mbit/s in 0.5 Mbit/s increments.”</p>	L’unité du débit est bit/s	L’unité du débit est Mbit/s

<p><b>L'envoi des données se fait par paquet et non par bit</b></p>	<p><i>“medium access control (MAC) protocol data unit MPDU: The unit of data exchanged between two peer MAC entities using the services of the physical layer (PHY)</i></p> <p><i>This QoS (+)Null frame shall have a QoS Control field .... needed to send the MPDU that is ready for transmission.”</i></p>	<p>Possibilité d'envoyer les bits qui forment un paquet d'une manière discontinue</p>	<p>L'envoi des données se fait par des paquets encapsulés avec d'autres champs (entête Mac, entête physique,...)</p>
<p><b>Le temps de transmission des paquets doit prendre en considération les entêtes Mac,...</b></p>	<p><i>“the format for the PPDU including the DSSS PLCP Preamble, the DSSS PLCP Header, and the MPDU (...),The entire PLCP Preamble and Header(192bits) shall be transmitted using the 1 Mbit/s for more details, you can see the 15.2.2 PLCP frame format in standard document”</i></p>	<p>Temps de transmission=Taille du paquet/Débit</p>	<p>Temps de transmission= ((taille des paquets*8+Macheader)/debit))+Tp)*0.000001</p> <p>L'entête Mac=240bits</p> <p>Tp=Temps de transmission de l'entête physique=192µs</p>
<p><b>Priorité entre les flux</b></p>	<p><i>“Le standard 802.11 e ”</i></p>	<p>Cette librairie ne modélise que le 802.11</p>	<p>Les priorités entre les flux sont introduites. Pour chaque priorité, nous donnons un AIFS (c'est le DIFS dans le 802.11) différent selon la priorité du flux, ainsi CWmin et CWmax changent en fonction de cette priorité.</p>

Afin d'intégrer ces modifications, il fallait lire le code de cette librairie et essayer de comprendre le rôle de chaque fonction. Le code de cette librairie représente environ à 1300 lignes. Les modifications intégrées s'installent parfois à la place de fonctions déjà existantes. Les modifications implémentées sont équivalentes à peu près à 200 lignes. La Figure 3.4 montre quelques modifications faites dans la librairie Truetime. Rappelons que la librairie Truetime est codée en utilisant le C++.

```

//Modification de temps de transmission des paquets en considérant les différents entêtes//
static double remxtime(NWmsg *m) {
    //size en byte
    //datarate en Mbps

    //l'ancien version: TrueTime
    // return m->remaining / nwsys->datarate;

    //Nouvelle version :Truetime modifiée
    return (((m->remaining*8+240)/ nwsys->datarate)+192)*0.000001;
}

//Modification de la fonction BACKOFF//
static int backoff_802_11(int nbr){
    // ancien version: Truetime
    // For some reason I have decided to increase nbr before this function is
    // called. Therefore CWMIN-1
    //printf(" backoff_802_11 at time % f\n", nwsys->time);

    //Nouvelle version :Truetime modifiée
    int collisionWindow=CWMIN_802_11;
    int nb_de_retransmission=0;

    if (nbr==1)
    {
        collisionWindow=CWMIN_802_11;
    }
    else
    {
        for (nb_de_retransmission=0; nb_de_retransmission<(nbr-1);
        nb_de_retransmission++)
        {
            collisionWindow=collisionWindow*2+1;
        }
    }
}

```

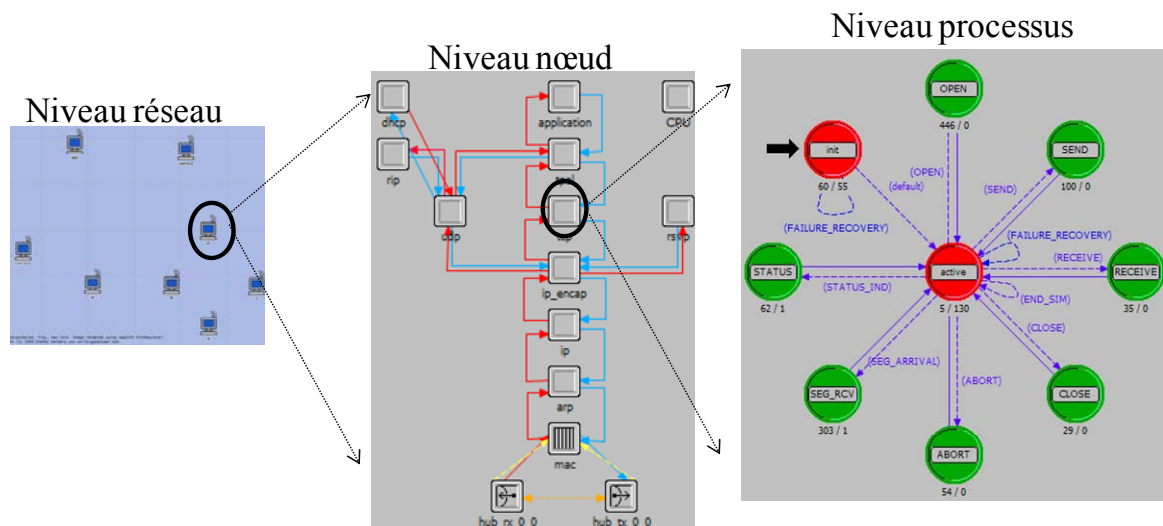
**Figure 3.4 : Quelques modifications faites sur le code de la librairie Truetime**

### 3.3 Outil orienté « réseau » : OPNET

Comme nous l'avons montré dans le chapitre 2, l'outil orienté «réseau», OPNET simule le comportement du réseau d'une manière efficace et satisfait parfaitement au standard. Par contre, il présente des difficultés d'implémentation des modèles de commande et PO. Par conséquent, nous allons détailler la manière dont cet outil fonctionne et essayer de faire un lien entre la modélisation d'un système sur OPNET et un outil standardisé qui peut représenter un système discret comme Statechart. Ce lien va nous permettre de voir la possibilité de modéliser un SDCR sans fil sur OPNET.

#### 3.3.1 Modélisation sur OPNET

OPNET utilise un modèle hiérarchique qui se base sur des frontières physiques et fonctionnelles décrivant d'une façon précise les topologies et les flux échangés dans un système de communication. Ce modèle hiérarchique présente trois niveaux de description (Figure 3.5).



Chaque processus est décrit par un ensemble d'états et transitions (EFSM) qui décrit son comportement

Figure 3.5 : Les modèles hiérarchiques sur OPNET

Pour chaque niveau, l'utilisateur peut utiliser un des modèles prédéfinis dans les bibliothèques d'OPNET ou proposer son propre modèle. Le niveau le plus haut de cette hiérarchie est le niveau réseau (*Network model*) qui représente la topologie physique d'un réseau de communication formé d'un ensemble de nœuds et de liens pour les interconnecter entre eux. À ce niveau, nous pouvons définir la position géographique et topologique ainsi que les caractéristiques des entités communicantes d'un réseau. Chacune des entités communicantes est décrite par un modèle de nœud (*Node model*). C'est le niveau intermédiaire, représentant

la structure fonctionnelle. Les nœuds sont composés d'un ensemble de blocs appelés modules. Les modules peuvent être soit prédéfinis par OPNET, tels que le module émetteur ou récepteur sans fil, et/ou par des modules dont on peut spécifier le comportement avec un modèle de processus (*Process model*). Le modèle processus est le niveau le plus bas dans la hiérarchie OPNET. Il permet de représenter le comportement d'un bloc à l'aide d'un diagramme d'états/transitions du type EFSM dont les actions sont écrites à l'aide de fonctions codées en C/C++. Ce code est appelé Proto-C. Chaque état contient deux blocs de code, à l'entrée de l'état (*Enter executive*) et à la sortie de l'état (*Exit executive*).

Nous pouvons identifier trois types d'états:

- état initial, représenté dans ce document par deux cercles concentriques (voir Figure 3.6, a)),
- état non forcé (*unforced state*, de couleur rouge), représenté dans ce document par un cercle continu (voir Figure 3.6, c)) : état pour lequel la désactivation nécessite l'occurrence d'un événement et la validation d'une condition logique,
- état forcé (*forced state*, de couleur verte), représenté dans ce document par un cercle pointillé (voir Figure 3.6, b)) : état pour lequel la désactivation ne nécessite que la validation d'une condition logique.

Pour les états non forcés, le code d'entrée (*Enter executive*) est exécuté lors de l'activation de l'état, et le code de sortie (*Exit executive*) est exécuté lors de l'occurrence d'un événement de sortie. Tant que la condition logique n'est pas validée, une boucle sur l'état (default) provoque de manière itérative la séquence suivante : nouvelle exécution du code d'entrée, attente d'une nouvelle occurrence de l'événement. Pour les états forcés, le code d'entrée est exécuté lors de l'activation de l'état, suivi du code de sortie sans nécessiter l'occurrence d'un événement. De même que pour les états non forcés, cette séquence est répétée jusqu'à validation de la condition de sortie.

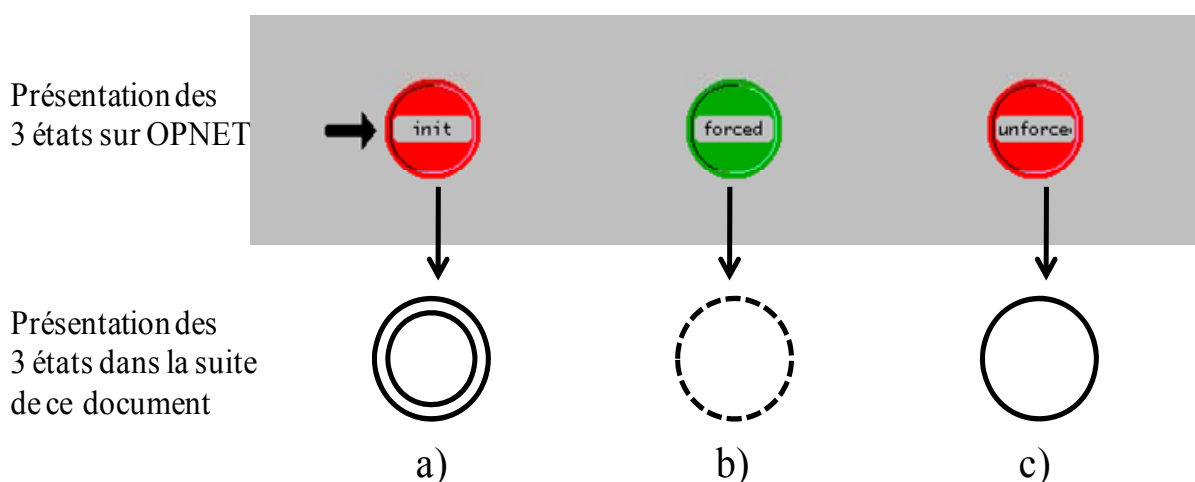


Figure 3.6 : Les trois types d'états

Afin de modéliser le comportement d'un nouvel équipement, nous utilisons le « *process model editor* » qui nous offre la possibilité de définir des états qui décrivent une situation spéciale du comportement de l'équipement. Ces états là sont reliés par des transitions. Le « *process model editor* » nous permet aussi de définir : des variables temporelles que nous pouvons utiliser juste durant l'exécution d'un processeur, des variables d'états qui peuvent être partagées au sein d'un même nœud et finalement les headers variables utilisées comme des variables globales pour tous les équipements au niveau réseau dans la simulation.

### 3.3.2 Simulation sur OPNET

OPNET est un simulateur à événement discret. A une date précise, un événement provoque une action dans le modèle dont le résultat est immédiatement disponible. En d'autre terme, le temps, durant les simulations, n'avance que lors de l'occurrence d'un événement (Figure 3.7). Cette manière de simuler présente plusieurs avantages par rapport à la simulation en temps réel dans lequel le temps s'écoule en continu ou échantillonné. En effet, cette dernière a comme principal inconvénient la précision des résultats qui dépend du taux d'échantillonnage pris. Plus la période d'échantillonnage est petite, plus la précision sur les résultats est grande mais plus l'espace d'état grandit. En plus, la simulation peut être inefficace si rien ne bouge (pas d'événement) pendant des longues périodes. Notons que la méthode en temps réel est utilisée par MATLAB.

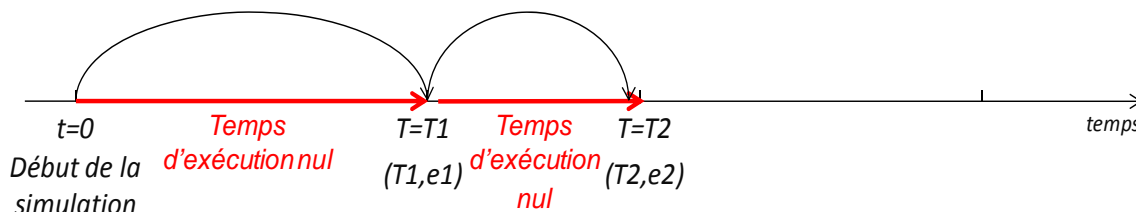


Figure 3.7 : Principe d'exécution d'OPNET

Afin de gérer tous les événements qui s'exécutent durant la simulation, OPNET possède une liste des événements. Cette liste contient trois colonnes : la date d'exécution, le type d'événement et finalement le module qui doit être exécuté. La tête de la liste correspond à l'événement qui doit être pris en compte en premier.

Finalement, OPNET offre des statistiques codées, il suffit de les cocher pour les surveiller. Mais l'outil permet aussi de coder ses propres statistiques afin de vérifier le comportement d'un équipement déjà modélisé.

Le « workflow » d'une simulation sous OPNET Modeler est illustré dans la Figure 3.8. Suite à la définition du modèle réseau à simuler, il faut choisir les bonnes statistiques à récupérer. Ensuite, l'exécution des simulations permet de récupérer les résultats et de pouvoir les analyser.

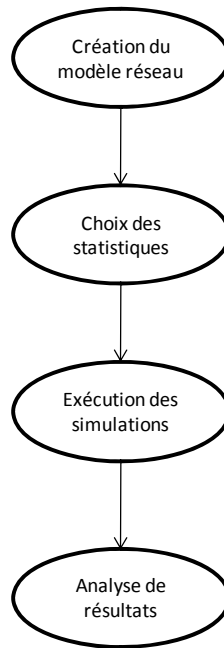


Figure 3.8 : Workflow sous OPNET

### 3.4 Comparaison OPNET et *TrueTime modifiée*

Afin de s'assurer que les modifications faites sur Truetime sont suffisantes et de l'exactitude des modèles discrets implémentés sur OPNET, une comparaison est faite entre ces outils. Cette comparaison est divisée en deux grandes parties :

*La première partie* consiste à valider la modélisation et la simulation du réseau en comparant : Truetime (la version que nous pouvons télécharger sur le site de l'université Lund<sup>12</sup>, sans modification), *TrueTime modifiée* (avec toutes les modifications déjà expliquées) et OPNET. Cette comparaison se fera au niveau réseau, en d'autres termes, cette comparaison sera faite sur un réseau formé de stations émettrices et réceptrices. Ces dernières envoient/reçoivent des paquets de taille fixe. Les résultats sur OPNET sont pris comme résultats de référence puisqu'il est un outil éprouvé par la communauté réseau. Donc la comparaison avec OPNET va nous assurer que la librairie *TrueTime modifiée* suit bien la procédure de backoff, la transmission des paquets et l'acquittement du récepteur. Par conséquent, la convergence des résultats entre OPNET et *TrueTime modifiée* validera les modifications faites sur la librairie Truetime.

*La deuxième partie* consiste à valider la modélisation et la simulation d'un SDCR sans fil en comparant : *TrueTime modifiée* avec OPNET sur un SDCR sans fil réel. La comparaison sera faite sur un cas d'étude particulier, nous regardons les résultats obtenus par ces deux outils en

<sup>12</sup> <http://www.control.lth.se/truetime/>

termes de sûreté et réactivité. Les résultats obtenus vont nous permettre de vérifier si la modélisation et le comportement d'un SDCR sans fil est identique dans les deux outils.

### 3.4.1 Première partie : Comparaison Truetime, *TrueTime modifiée*, OPNET

Pour comparer les trois simulateurs (Truetime, *TrueTime modifiée* et OPNET), nous réalisons différents scénarios. Rappelons l'hypothèse prise dans le chapitre 1 : pas de stations cachées donc les paquets RTS/CTS ne sont pas pris en compte.

Dans un premier temps et au niveau de la topologie du réseau, nous supposons que les stations envoient les données au même débit. Plus précisément, le *Débit* choisi est égal à 11Mbit/s et les stations émettrices envoient des paquets de 1000 octets avec une *Pcarte* de 1s. Le temps de simulation est 1h. Notons bien que chaque station émettrice envoie à un propre récepteur. On définit 5 scénarios, appelé *scenarioN*, dont N présente le nombre de stations émettrices des paquets. Exemple : *scenario2*, dans ce réseau, nous avons deux stations émettrices de paquets et deux réceptrices, chaque station émettrice envoie à sa propre station réceptrice.

Les résultats comparés sont le délai maximum, minimum et moyen de transmission des flux.

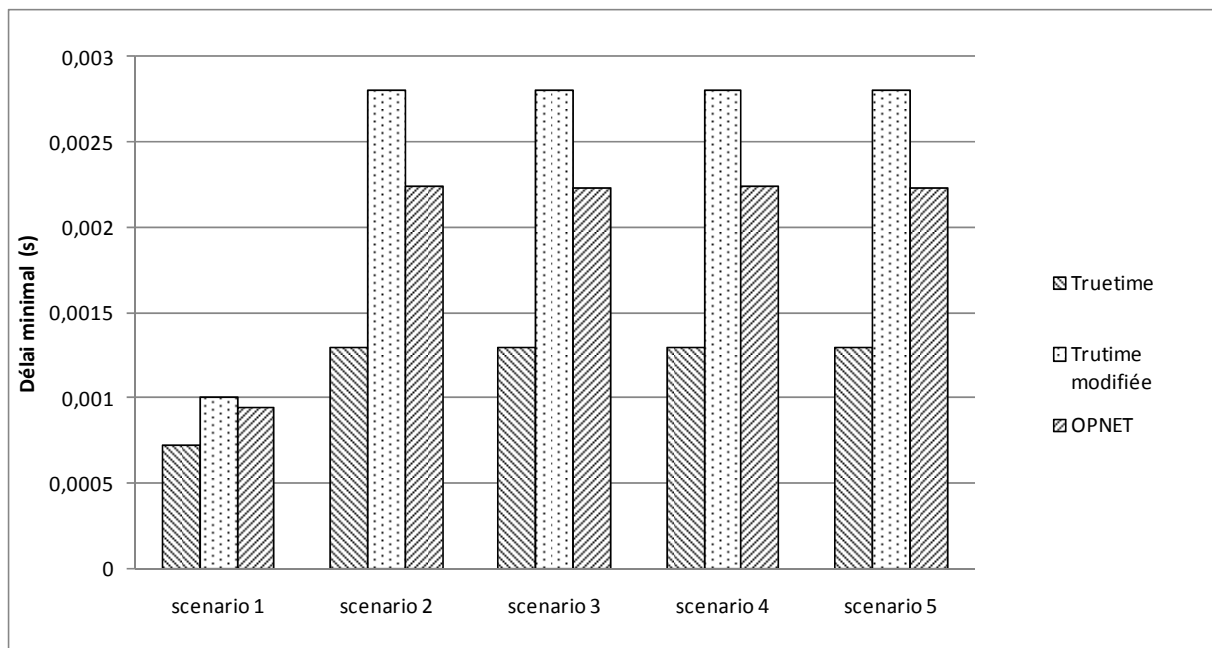
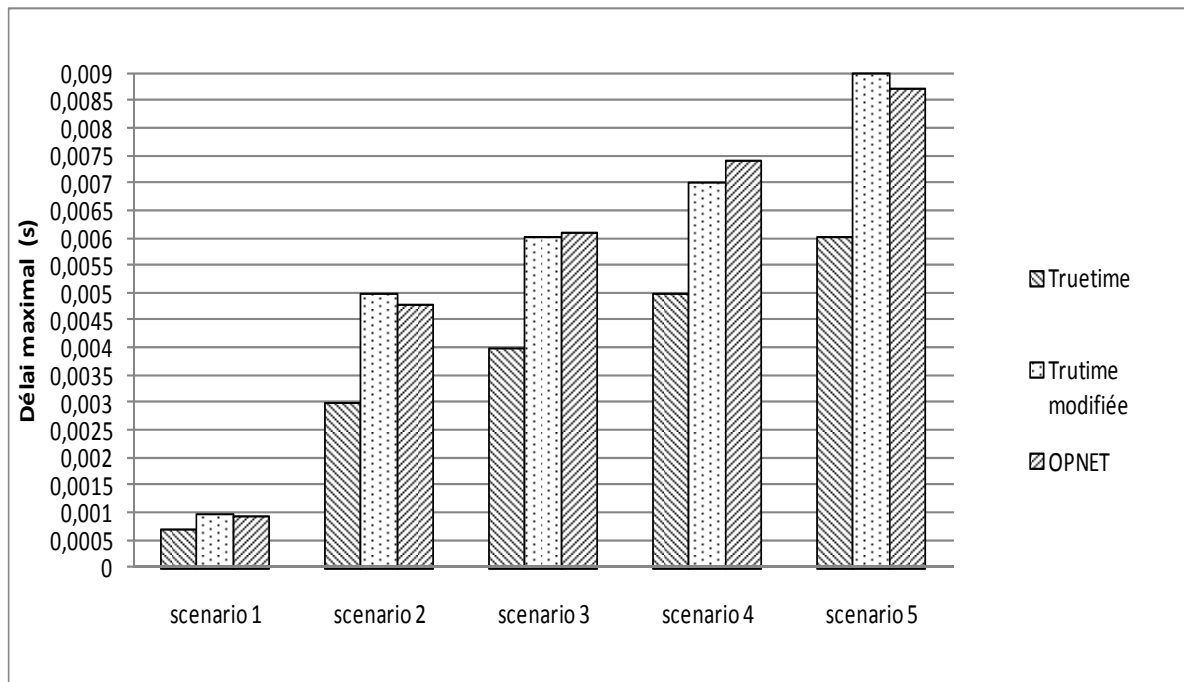
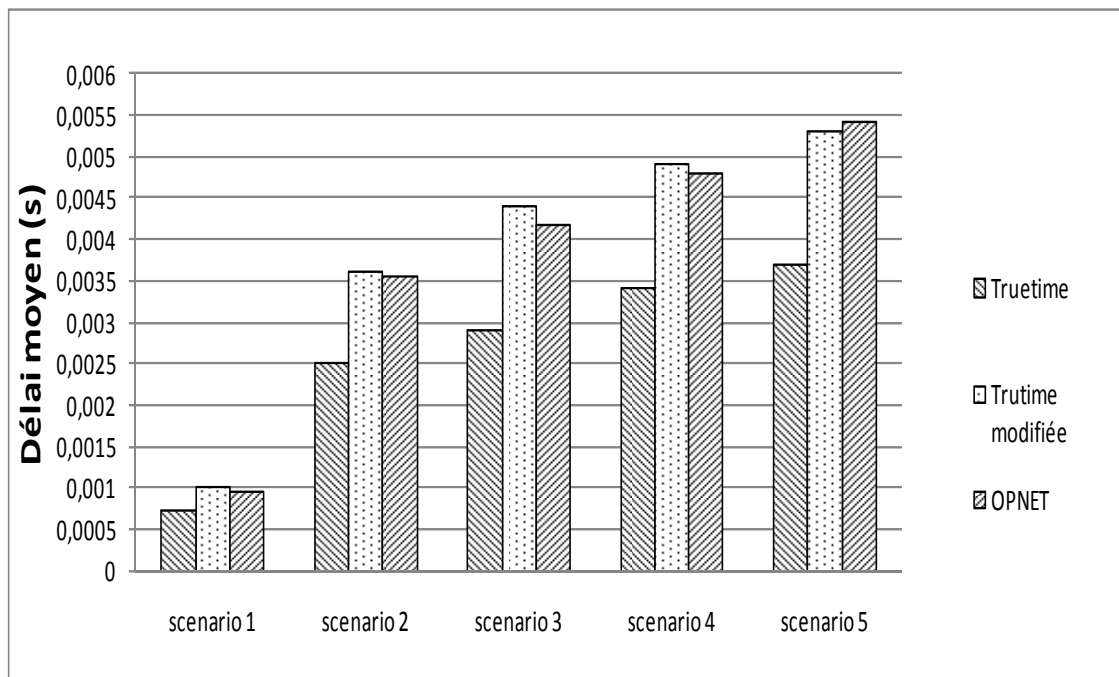


Figure 3.9 : Comparaison des trois outils par rapport au délai minimal



**Figure 3.10: Comparaison des trois outils par rapport au délai maximal**



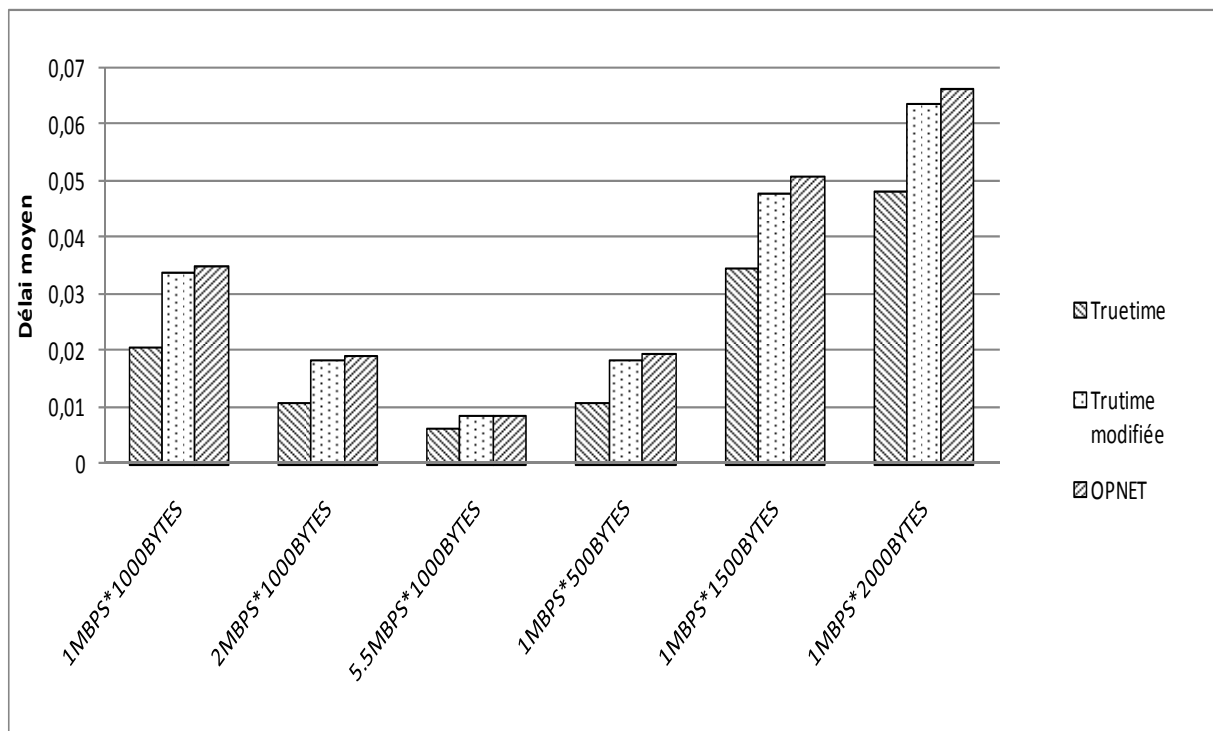
**Figure 3.11: Comparaison des trois outils par rapport au délai moyen**

Dans la Figure 3.9, nous remarquons bien que l'écart entre le délai minimal de Truetime et OPNET est plus grand que celui entre *TrueTime modifiée* et OPNET. Cette différence est due au fait que la librairie Truetime ne prend pas en compte l'entête physique des paquets envoyés. Notre correction prend en compte cet aspect, elle se rapproche d'OPNET. De même dans la

Figure 3.10 et Figure 3.11, les valeurs de délai maximal et moyen dans la librairie *TrueTime modifiée* est plus proche d'OPNET que celle de Truetime.

Cette différence des résultats entre Truetime et OPNET est due à la simplification de modélisation de la librairie Truetime. Par contre, l'écart des résultats entre *TrueTime modifiée* et OPNET arrive au maximum à 10%, ce qui est acceptable. Cette écart est du à la valeur du précision retenu. Par conséquent, *TrueTime modifiée* montre une convergence avec OPNET. Cette convergence valide la bonne utilisation des procédures d'accès au médium (ex : procédure de backoff,...) par les stations dans la librairie *TrueTime modifiée*. Par suite, elle prouve la nécessité des modifications faites et le bon comportement du réseau dans cette librairie.

Afin de généraliser cette conclusion (convergence des résultats entre *TrueTime modifiée* et OPNET), nous faisons maintenant varier le débit et la taille des paquets dans le scenario à 5 émetteurs et 5 récepteurs. Le choix des valeurs de débits suit les valeurs autorisées dans le standard IEEE 802.11b-couche Physique.



**Figure 3.12 : délai moyen dans différents scénarios**

La Figure 3.12 confirme la conclusion précédente, on voit toujours une convergence des résultats entre *TrueTime modifiée* et OPNET et une divergence par rapport à la librairie Truetime.

Après avoir validé le comportement du réseau (modélisation, simulation) sur les outils OPNET et *TrueTime modifiée*, nous allons procéder de même pour le comportement d'un SDCR sans fil.

Afin de simplifier les comparaisons, nous ignorons dans la deuxième partie la librairie Truetime (non modifiée) puisque les résultats précédents montrent un comportement loin du standard. Par conséquent, la deuxième partie sera juste une comparaison entre *TrueTime modifiée* et OPNET sur un SDCR sans fil.

### 3.4.2 Deuxième partie : Comparaison *TrueTime modifiée*, OPNET sur un SDCR sans fil

Dans cette partie, nous allons évaluer les résultats obtenus en utilisant *Truetime modifiée* avec ceux d'OPNET. Cette évaluation sera appliquée sur un cas d'étude. Pour atteindre ces résultats, nous devons passer par quatre étapes :

- Modélisation générique des parties d'un SDCR sans fil.
- Instanciation sur un cas d'étude
- Implémentation des modèles sur OPNET et *TrueTime modifiée*
- Comparaison des résultats obtenus dans les deux cas

#### 3.4.2.1 Modèles génériques d'un SDCR sans fil

Comme nous avons expliqué précédemment, un SDCR sans fil est formé d'une partie Réseau, de la partie Commande et de la partie Opérative. Rappelons que:

- Le *modèle de commande* représente les lois de commande, le fonctionnement cyclique de l'API et la gestion des paquets par le coupleur de communication,
- Le *modèle de partie opérative* représente l'état des actionneurs, des capteurs et la gestion des paquets par le coupleur de communication,
- Le *modèle du réseau* représente le comportement du réseau lors des échanges d'informations entre partie commande et PO.

La définition des modèles de commande et de la partie opérative s'inspire des travaux développés dans la thèse de Pascale Marangé (Marangé, 2008) que nous avons adapté pour tenir compte notamment de la désynchronisation entre les modèles de commande et de PO induit par le réseau.

#### Modèle de commande

Les capteurs envoient des variables appelées  $\{E_i, E_j, \dots\}$  à l'API. Cette dernière met à jour ses variables internes  $\{E_{iapi}, E_{japi}, \dots\}$ , traite les données et met à jour les variables de sorties appelées  $\{S_{iapi}, S_{japi}, \dots\}$ , voir Figure 3.13

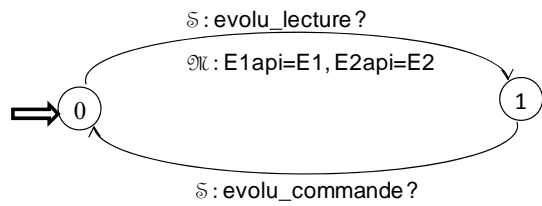
Le *modèle de fonctionnement de l'API* est représenté par trois modèles, ces modèles s'exécutent d'une manière périodique:

1. le *modèle de lecture* représente donc la mise à jour des variables de l'API. Quand un message arrive en entrée, l'API recopie son contenu dans ses variables internes. En d'autres termes, ce modèle fait simplement ces opérations " $E_{iapi} = E_i, E_{japi} = E_j \dots$ " (Figure 3.13, a))
2. le *modèle des lois de commande* représente l'évolution de la commande ; ce modèle passe d'un état à l'autre lorsque la transition est vraie et qu'il réceptionne le message commande. Ce modèle est décrit par un automate communicant défini par  $(\Sigma, X, x_0, \delta, F)$ , avec :
  - $\Sigma$  présente l'alphabet,  $\Sigma^*$  ensemble de tous les mots (suite finie d'éléments) que l'on peut construire sur  $\Sigma$
  - $X$  l'ensemble des états,
  - $x_0$  est l'état initial,
  - $\delta$  : est un ensemble de fonctions de transition  $(X \times \Sigma \rightarrow X)$  définies par une garde ( $\mathcal{G}$ ) qui permet d'exprimer une condition, par une synchronisation ( $\mathcal{S}$ ) qui permet de synchroniser les événements entre les automates par échange de messages. Les échanges sont notés de la manière suivante : pour l'émission *mess !* et pour la réception *mess ?*, et finalement la mise à jour ( $\mathcal{U}$ ) qui permet de mettre à jour les variables
  - $F$  est l'ensemble des états finaux
  - Le domaine de  $\delta$  est étendu de  $X \times \Sigma$  à  $X \times \Sigma^*$  c'est à dire  $\delta(x, se) = \delta(\delta(x,s), e)$  pour  $s \in \Sigma^*$  et  $e \in \Sigma$

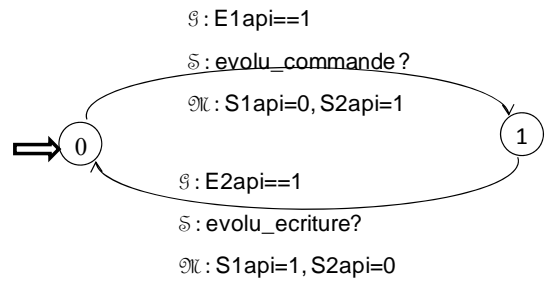
La Figure 3.13, b) présente un exemple trivial d'un modèle de commande pour manipuler un vérin, qui se déplace entre deux capteurs 1 et 2.  $S1_{api}$  et  $S2_{api}$  sont des variables internes de l'API.

3. le *modèle d'écriture* représente les mises à jour des sorties selon " $S_i = S_{iapi}, S_j = S_{japi} \dots$ " (Figure 3.13, c))

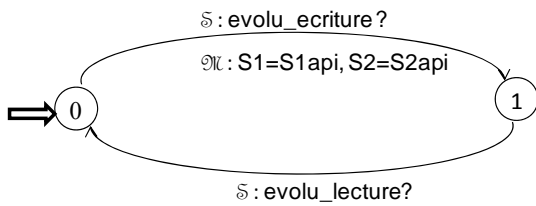
Pour synchroniser l'exécution des trois modèles du fonctionnement de l'API, un *modèle d'environnement* est intégré (Figure 3.13, d)). Ce modèle assure l'exécution séquentielle des modèles de *lecture*, *lois de commande* et *d'écriture*. Le cadencement est réalisé selon trois événements (*evolu\_lecture*, *evolu\_commande* et *evolu\_ecriture*) sans notion de temps.



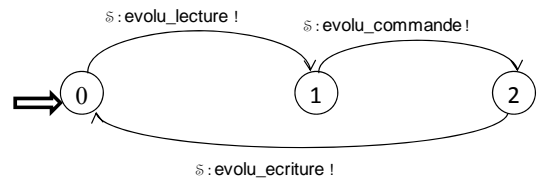
a)



b)



c)



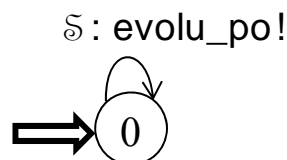
d)

**Figure 3.13: Modèle de fonctionnement de l'API: a) Lecture, b) loi de commande, c) écriture et d) environnement**

### Modèle de la partie opérative

La partie opérative est formée des capteurs et des actionneurs et ils sont aussi modélisés en utilisant les automates communicants. Rappelons que le comportement continu des actionneurs est modélisé par un ensemble d'états et de transitions. Ces derniers sont conditionnés par l'occurrence des événements venant des capteurs ou du contrôleur. Nous ne décrivons pas ici cette partie de façon générale mais nous le ferons par la suite sur un cas d'étude. Plusieurs attributs peuvent être considérés dans cette partie : exemple la *taille des capteurs*, *l'espacement entre ces capteurs*. Dans ce travail, nous considérons que ces attributs sont constants.

Le modèle de la partie opérative peut aussi avoir un modèle d'environnement (Figure 3.14), l'événement *evol\_po* provoque l'évolution du modèle de la PO.



**Figure 3.14 : Exemple d'un modèle d'environnement de la PO**

## Synchronisation entre les modèles d'environnement de la PC/PO

La synchronisation entre le modèle d'environnement de la PC et celui de la PO est nécessaire pour garantir l'observabilité de tous les événements du modèle de la PO par le modèle de la PC. Dans le cas contraire lorsqu'il n'y a pas de synchronisation entre ces modèles d'environnement, le modèle de PO a le droit d'évoluer  $N$  fois avant une évolution du modèle de la PC et vice-versa ( $N$  pouvant varier de 1 à l'infini). Afin d'éviter ce cas, une synchronisation « artificielle » est réalisée par l'introduction de deux jetons «  $t, t'$  » assurant la commutation d'un modèle à l'autre et l'évolution sur un cycle complet (*evolu\_lecture*, *evolu\_commande* et *evolu\_ecriture*) des modèles d'environnement (Figure 3.15). Cette commutation est représentée par deux variables  $x$  et  $y$ . La variable  $x$  désigne le nombre de cycle où la PC s'exécute par rapport aux  $y$  cycles de PO.

Les variables  $x$  et  $y$  sont modélisées par deux horloges, une horloge pour représenter le temps de cycle de l'automate *Papi* et l'autre pour représenter la dynamique de la PO. L'intérêt est de pouvoir représenter le rapport temps cycle commande / temps inertie PO.

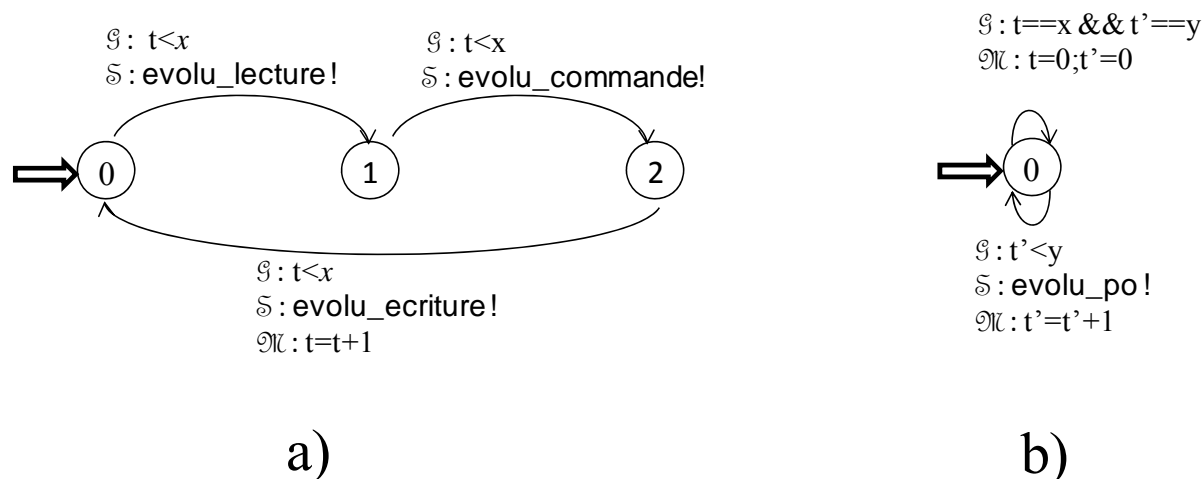
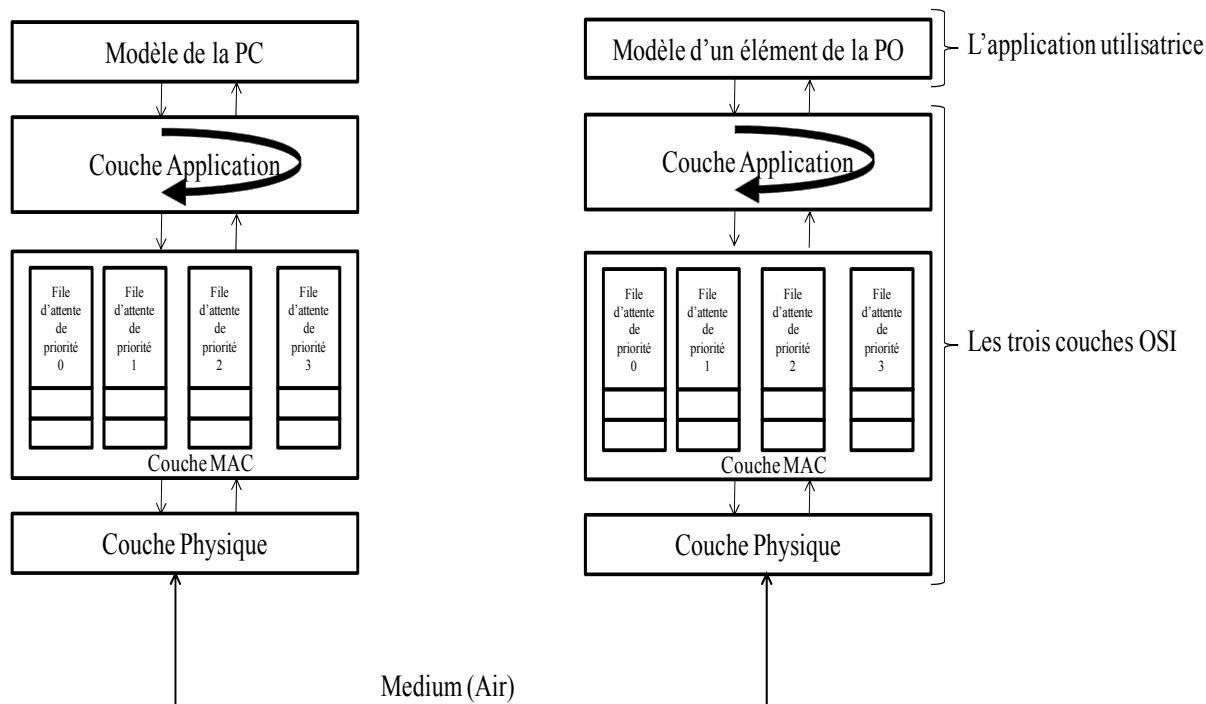


Figure 3.15 : Modèles d'environnement a) de la PC et b) de la PO

### Modèle du réseau

Le modèle du réseau (voir Figure 3.16) présente les interactions entre les entités du réseau. Rappelons que dans un SDCR sans fil, trois couches sont utilisées: Physique, Mac et Application. En plus, l'application utilisatrice est ajoutée, qui représente le modèle du comportement discret de la partie étudiée.



**Figure 3.16 : Modèle du réseau**

*Couche Physique:* Rappelons que cette couche est responsable à la transmission et la réception des signaux venant du médium. Plusieurs technologies peuvent être choisies dans un réseau sans fil, exemple : 802.11a, 802.11b, 802.11g. Chaque technologie envoie ces données sur une fréquence spécifique et avec un débit bien déterminé. Rappelons aussi que chaque paquet est encapsulé par un entête physique.

*Couche MAC:* Déjà expliquée en détails dans les chapitres précédents, implémente le standard IEEE 802.11 ou 802.11e

*Couche Application:* Détaillée dans le chapitre 1, cette couche transmet périodiquement (période= $P_{carte}$ ) et reçoit immédiatement des paquets à/de la couche MAC

*Application utilisatrice :* Cette application présente le comportement discret de l'entité représentée (partie opérative ou commande). Exemple : si cette entité est un capteur, cette application sera un modèle du comportement du capteur.

Plusieurs attributs peuvent changer le comportement du réseau comme déjà expliqué dans les chapitres précédents: *Taille des paquets*, *Période d'échantillonnage*( $P_{carte}$ ), *Priorité* : au niveau couche application. *Nombre maximal de retransmission* au niveau couche MAC. *Débit* au niveau couche Physique

### 3.4.2.2 Cas d'étude

Les modèles de simulation présentés dans les sections précédentes sont particularisés sur un cas d'étude (Figure 3.17) constitué d'un vérin pneumatique équipé d'un distributeur 5/3 et de dispositifs en réseau permettant sa commande, de trois capteurs de position  $S1$ ,  $S2$  et  $S3$ , d'un API où sont implantées les lois de commande du système (les signaux  $A$  et  $B$  déclenchent respectivement la rentrée et la sortie du vérin), et d'un système de communication Wifi (IEEE 802.11) permettant l'échange d'informations. Puisque nous travaillons avec un réseau en mode infrastructure, toutes les informations échangées doivent passer par un AP (point d'accès). Dans notre cas, l'AP est le coupleur de communication de l'API. Chaque capteur  $S_i$  délivre une variable logique  $s_i$  (qui représente l'exécution ou non du capteur). Les trois capteurs de position  $S1$ ,  $S2$  et  $S3$  sont reliés à une seule carte réseau. La commande ( $A$  et  $B$ ) et des observations ( $s_1$ ,  $s_2$  et  $s_3$ ) sont émises de manière périodique.

Nous avons considéré le scénario suivant: à partir de la position initiale  $S1$ , le vérin se déplace vers la position intermédiaire  $S2$ , ce dernier signale à l'API la présence du vérin ( $s_2=1$ ). Alors, l'API donne un ordre d'arrêt ( $A=B=0$ ) à ce vérin. Pour évaluer le comportement du vérin, un paramètre est appelé  $T_s$  qui représente le temps d'arrêt du vérin depuis  $S1$ . Ce paramètre varie en fonction du comportement du réseau : la charge du réseau, la *taille des paquets*, le *nombre maximal de retransmission*, la période de scrutation des coupleurs de communication ( $P_{carte}$ ) et la période de l'API ( $P_{api}$ ).

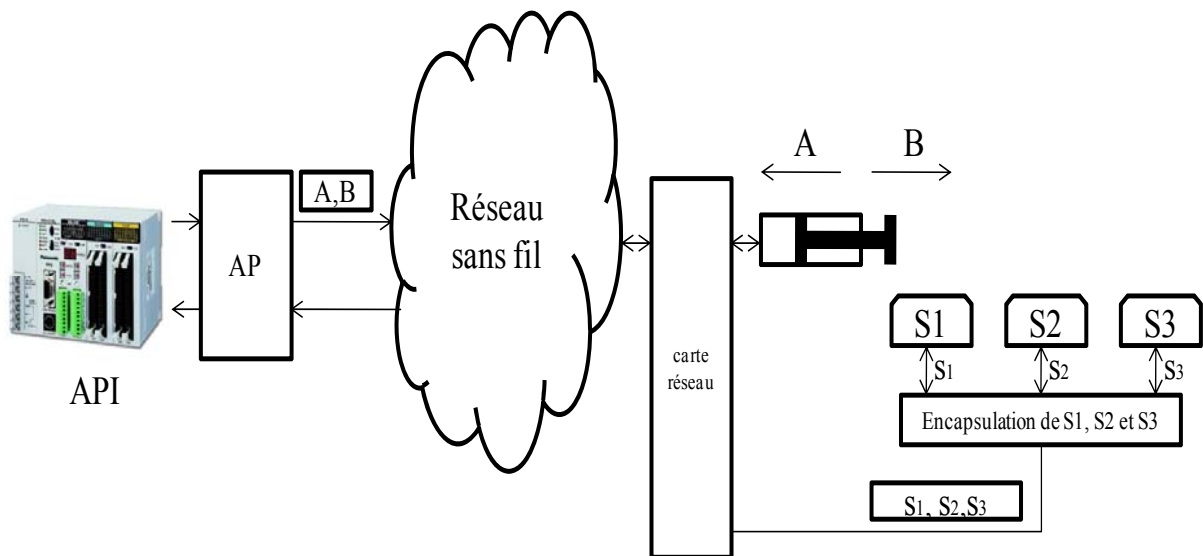


Figure 3.17 : Cas d'étude

### 3.4.2.3 Instanciation des modèles génériques sur un cas d'étude

Comme on l'a déjà expliqué, trois modèles se retrouvent pour présenter ce type de système : modèle de commande, partie opérative et le réseau. Un autre modèle est ajouté, est appelé modèle d'environnement, il sera expliqué par la suite.

## Modèle de la commande

Trois modèles décrivent le comportement de cette partie : modèle de lecture, loi de commande et écriture. Ces trois modèles s'exécutent d'une manière cyclique avec une période Papi. Les deux modèles de lecture et écriture servent à lire et mettre à jour les variables d'entrées et de sorties respectivement. Par contre, le modèle de loi de commande dans ce cas d'étude envoie toujours un ordre de sortie du vérin ( $B=1$ ). Mais lorsqu'il reçoit la présence du vérin sur le capteur intermédiaire S2 ( $s_2=1$ ), il donne l'ordre d'arrêt du vérin ( $A=B=0$ ). La Figure 3.18 montre les trois modèles (lecture, loi de commande et écriture) et le modèle d'environnement. S1api, S2api, Aapi et Bapi sont des variables internes de l'API.

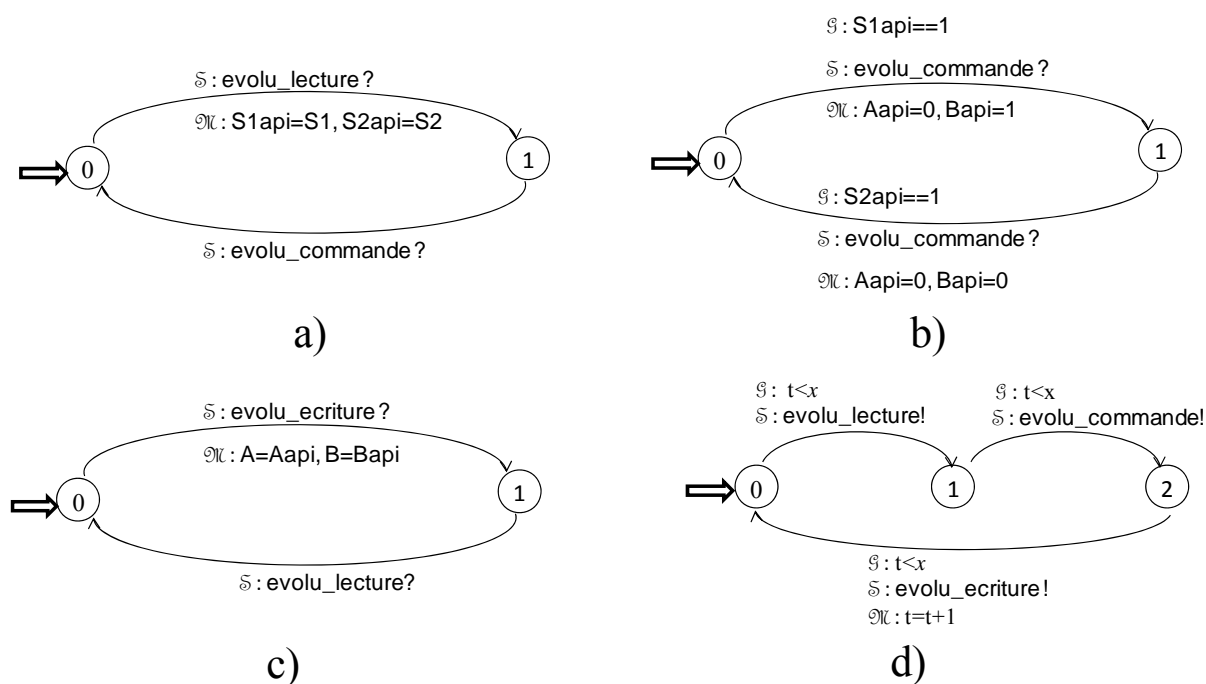


Figure 3.18 : Modèle de la partie commande

## Modèle de la partie opérative

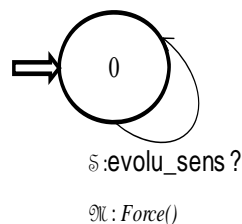
La partie opérative est formée des modèles des vérins et des capteurs. Selon (Marangé, 2008), un vérin est défini par deux modèles: modèle de sens et position.

Les ordres de l'API sont les entrées du modèle de sens, ce dernier (Figure 3.19) calcule la force appliquée au vérin. Pour faire ce calcul, ce modèle utilise la fonction  $Force()$ , pour plus de détails sur cette fonction (Marangé, 2008). La valeur de cette force peut provoquer trois types d'actions: force positive suffisante pour avancer le vérin, négative pour faire reculer le vérin ou finalement pas de force ce qui arrête le vérin. Ces trois types d'actions sont représentées par deux variables VersS1 et VersS2, qui seront les sorties du modèle de sens et envoyées au modèle de position.

Le modèle de position (Figure 3.20) définit la position du vérin sur sa ligne de mouvement entre les capteurs. Ces positions sont définies par rapport aux positions des capteurs. Quand un vérin est sous un capteur, il peut être dans l'une des trois actions : avancer, reculer ou arrêter. Pour cela, chaque capteur est représenté par trois états dans le modèle de position. Aussi, quand le vérin est entre deux capteurs, il peut être dans l'une des trois actions (avancer, reculer ou arrêter), donc l'espace entre deux capteurs est représenté aussi avec trois états. Quand ce modèle est exécuté, si ces entrées (VersS1 et VersS2) vérifient les conditions sur les transitions des états, le modèle de position passe d'un état à un autre. La sortie de ce modèle sera les variables  $s_i$  où  $i=1, \dots, N$ = nombre des capteurs sur la ligne de mouvement du vérin. Si cette variable est vraie ( $s_i = 1$ ), cela signifie que le vérin est au dessous du  $i$  ème capteur. Si  $s_i = 0$ , le capteur signale l'absence du vérin.

Et finalement, un modèle d'environnement (Figure 3.21) est ajouté afin d'assurer le cadencement des deux modèles sens et position et en plus l'alternance avec le modèle d'environnement de la PC. Dans notre cas d'étude, le modèle d'environnement de PO s'exécute deux fois plus vite que celui de la PC, d'où  $x=1$  et  $y=2$

Puisque dans ce cas d'étude les capteurs et le vérin sont reliés ensemble à une carte réseau donc nous avons plus besoin d'un modèle de capteur. Les variables booléennes  $s_i$  venant du modèle de la position seront directement envoyées à la PC.



**Figure 3.19 : Modèle de sens**

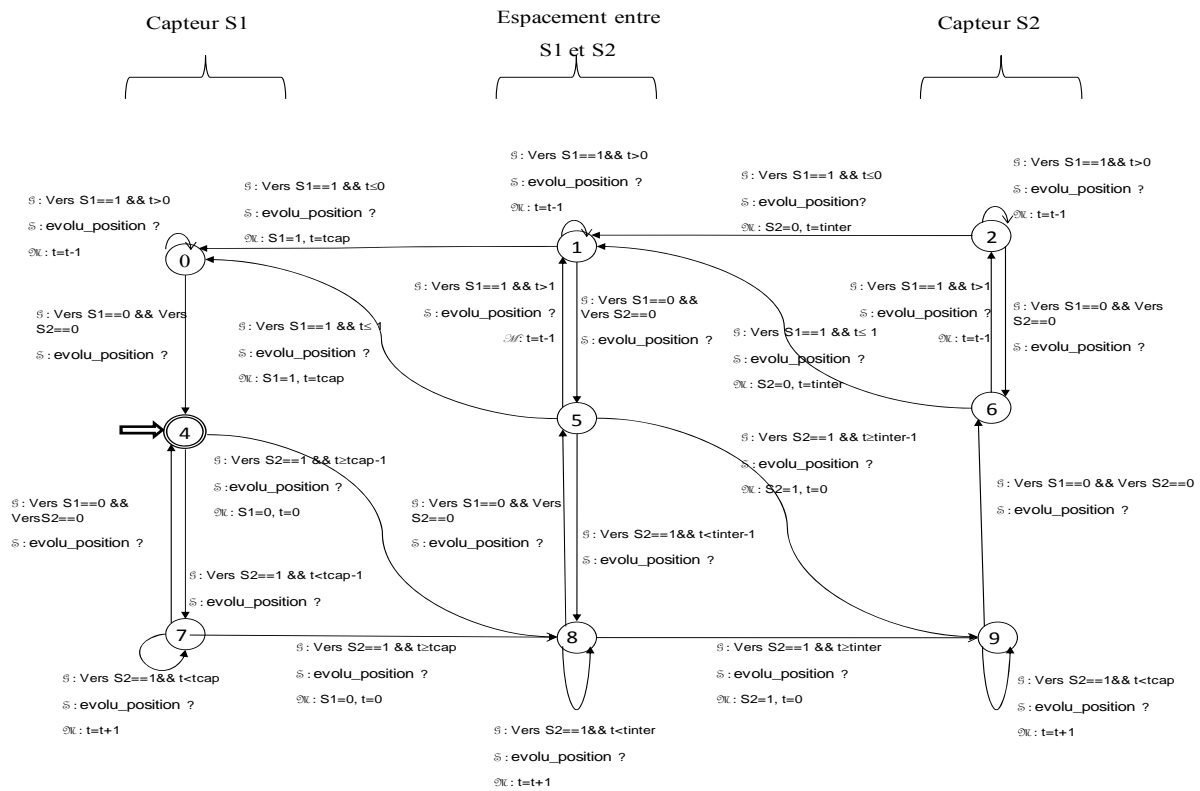


Figure 3.20 : Modèle de position

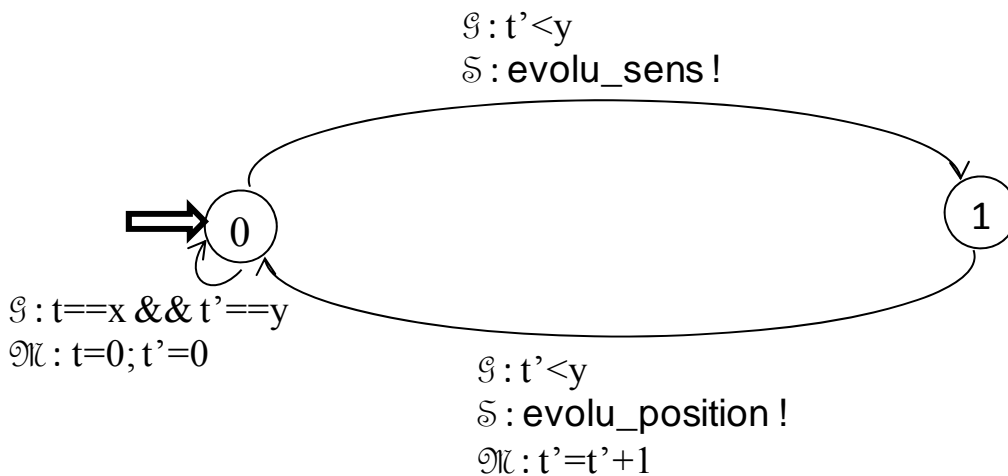


Figure 3.21 : Modèle d'environnement de la PO

Dans notre cas d'étude, deux attributs de cette partie sont considérés : *Taille des capteurs*, *l'espacement entre les capteurs*. La *Taille des capteurs* est traduite par un temps nécessaire pour qu'un vérin traverse un capteur, cette variable est représentée dans le modèle de position par la constante *tcap*. De même, *l'espacement entre les capteurs* représente le temps pour qu'un vérin passe entre deux capteurs, elle est représenté par la variable *tinter* dans le modèle de position.

## Modèle du réseau

Dans ce cas d'étude, nous identifions deux cartes réseau : une pour l'API et l'autre pour la partie opérative. Dans chaque carte, trois couches existent :

- Couche physique et MAC : leurs modèles se trouvent dans l'outil utilisé : MATLAB-*TrueTime modifiée* ou OPNET.
- Couche Application : Dans le cas d'OPNET, cette couche existe dans la station « *wlan\_station\_adv* ». Par contre, sur *Truetime modifiée*, cette couche doit être implémentée. Rappelons que les trois capteurs sont sur un même coupleur de communication. Donc, les valeurs  $s_1$ ,  $s_2$  et  $s_3$  sont encapsulées dans un même paquet.
- Et finalement, les applications utilisatrices sont implémentées dans OPNET et *Truetime modifiée*, qui décrivent les comportements des stations indiquées.

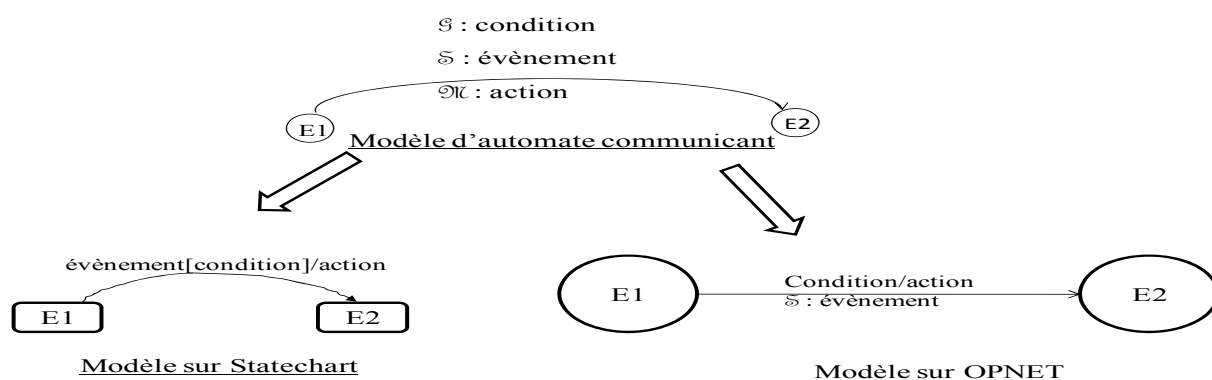
### 3.4.2.4 Implantation du cas d'étude sous OPNET et *TrueTime modifiée*

#### Concepts généraux

Rappelons que pour présenter un modèle sur MATLAB, nous utilisons les Statecharts et sur OPNET, nous utilisons le formalisme du type EFSM. Nous avons donc été amenés à définir deux transformations de modèles permettant le passage d'un automate communicant à un modèle Statecharts et un modèle OPNET (Figure 3.22). Les états des automates communicants correspondent à des états Statecharts et des états non forcés du modèle OPNET.

La garde ( $\mathcal{G}$ ), la synchronisation ( $\mathcal{S}$ ) et la mise à jour ( $\mathcal{U}$ ) sont traduit sous la forme d'une condition, événement et action respectivement portée par la traduction en Statechart et dans les modèles d'OPNET (EFSM).

Ces mécanismes de transformations nous permettent donc d'implémenter un modèle de type Automates Communicants sur les outils logiciels Matlab et OPNET.



**Figure 3.22 : Transformations d'un modèle d'automate communicant vers un modèle Statechart et vers un modèle OPNET**

Dans les modèles d'OPNET, rappelons qu'ils existent aussi des blocs de code d'entrée et de sortie, que nous allons utiliser pour définir et récupérer des statistiques de simulation.

### Principe général sur OPNET

Comme mentionné précédemment, chaque équipement qui veut échanger des données doit être équipé d'un coupleur de communication. Sur OPNET, chacun de ces équipements est représenté par la station "*wlan\_station\_adv*", qui intègre les trois couches OSI avec l'application utilisatrice (Figure 3.23). Rappelons que ces couches sont modélisées en utilisant les EFSM d'OPNET.

Chaque station gère ses propres attributs réseau, qui peuvent être différents avec d'autres stations. Dans ce chapitre, nous considérons que toutes les stations possèdent les mêmes attributs réseau. En d'autres termes, toutes les stations ont les mêmes couches physiques, MAC et Application.

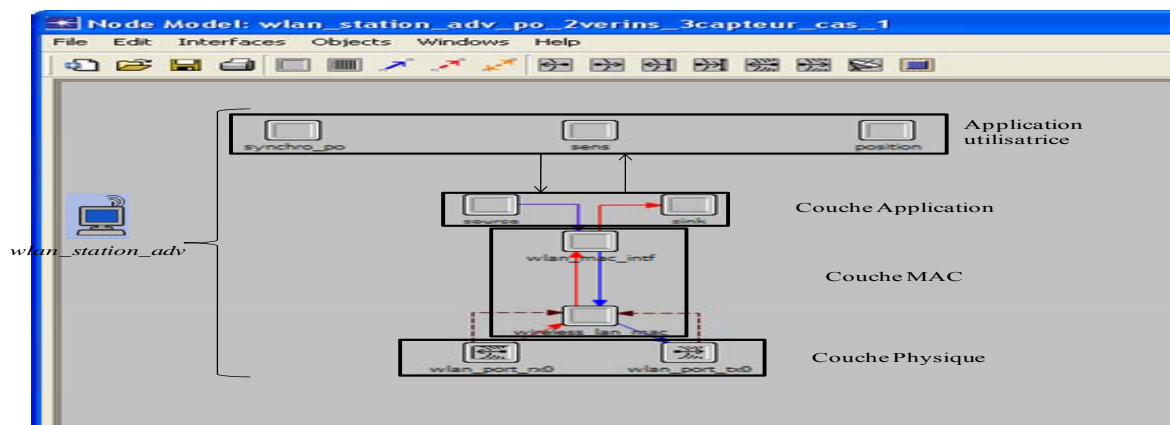


Figure 3.23 : Présentation un exemple de la modélisation d'un vérin en utilisant station *wlan\_station\_adv*

### Principe général sur MATLAB- *TrueTime* modifiée

Le comportement opérationnel des équipements (API et partie opérative) est représenté par des modèles développés en Statechart dans l'environnement MATLAB, et pour le réseau, nous utilisons la bibliothèque *TrueTime* modifiée. Comme nous l'avons déjà expliqué précédemment, cette bibliothèque simule seulement les couches MAC et Physique des flux envoyés et reçus. Ainsi donc, nous sommes obligés de modéliser les couches absentes par exemple la couche Application.

La Figure 3.24 représente l'architecture générale d'une modélisation sur MATLAB :

1. Les deux applications utilisatrices, modèles de la PC et de la PO, présentent le comportement des deux entités API et PO respectivement. Notons aussi que le modèle de la PC est équipé d'un *pulse generator* pour assurer l'exécution des trois modèles

(Papi) de l'API (modèle de lecture, modèle des lois de commande, modèle d'écriture). Ces trois modèles sont intégrés dans le bloc de l'API.

2. Puis, les couches applications sont présentées par deux blocs : le premier envoie périodiquement (Pcarte) des paquets vers le réseau, le deuxième reçoit les paquets et les transmet immédiatement vers l'application utilisatrice.
3. Finalement, la librairie *Truetime modifiée* sert à relier les équipements, c'est la partie réseau du SDCR sans fil.

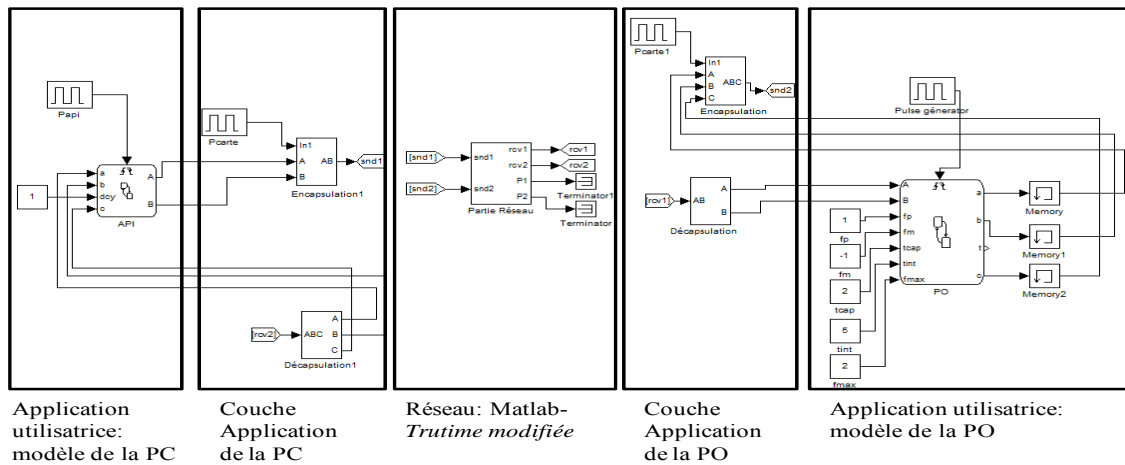


Figure 3.24 : Modélisation sur MATLAB

### Modélisation du cas d'étude sur OPNET et MATLAB-TrueTime modifiée

En utilisant les règles de transformations définies dans la Figure 3.22, on obtient les modèles PC, PO en EFSM et Statechart à partir des modèles du cas d'étude détaillés précédemment. La Figure 3.25 présente un exemple de passage de modèle d'écriture d'automate communicant en Statechart et OPNET.

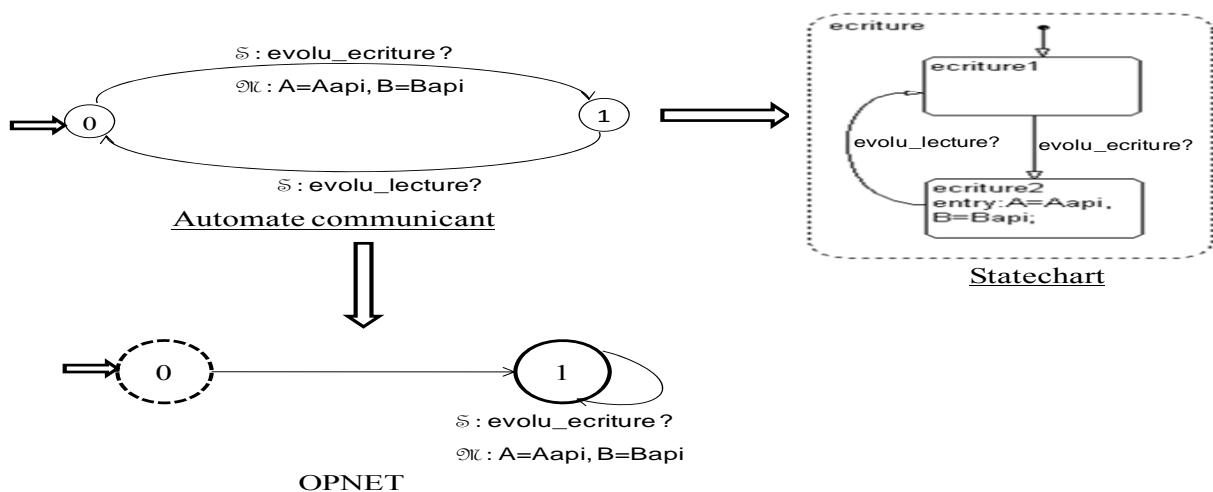


Figure 3.25 : Modèle de l'écriture en automate communicant, Statechart et OPNET

Pour implanter les modèles d'environnement de la PC et de la PO, la variable  $t$  utilisé pour synchroniser ces modèles et cadencer leur évolution (Figure 3.15) est remplacé par une horloge sur OPNET comme sur Truetime. Elle assure l'exécution cyclique de la PC (=Papi) ainsi que la dynamique de la PO.

Les Figure 3.26 et Figure 3.27 montrent la synthèse des différents modèles du cas d'étude (détaillés plus haut) réalisés sur *TrueTime modifiée* et OPNET respectivement,

- Figure 3.26 a) et Figure 3.27 a) : présentent le modèle de l'API, application utilisatrice, qui contient les trois modèles de fonctionnement de l'API (*modèle de lecture, loi de commande* et *écriture*). En plus, ces figures présentent la couche Application nécessaire pour envoyer périodiquement (*Pcarte*) les ordres et reçoit immédiatement l'information du réseau. Notons que les informations échangées sont encapsulées au niveau source et désencapsulées au niveau destinataire.
- Figure 3.26 b) et Figure 3.27 b) : La librairie *TrueTime modifiée* désigne les couches physiques et MAC. Notons qu'il y a deux entrées (snd1 et snd2) et deux sorties (rcv1 et rcv2), qui désignent les entrées, sorties de deux cartes réseau de l'API et la PO. Sur OPNET, nous utilisons deux stations de type « *wlan\_station\_adv* » qui présentent l'API et la PO.
- Figure 3.26 c) et Figure 3.27 c) : présentent le modèle de la PO, application utilisatrice, qui contient les deux modèles (*modèle de sens* et *position*) et la couche Application qui encapsule ( $s_1, s_2$  et  $s_3$ ) et les envoie périodiquement (*Pcarte*) vers l'API.
- Figure 3.26 d) et Figure 3.27 d), e) : les deux modèles de l'environnement assurent l'exécution du modèle de commande et du modèle de PO.

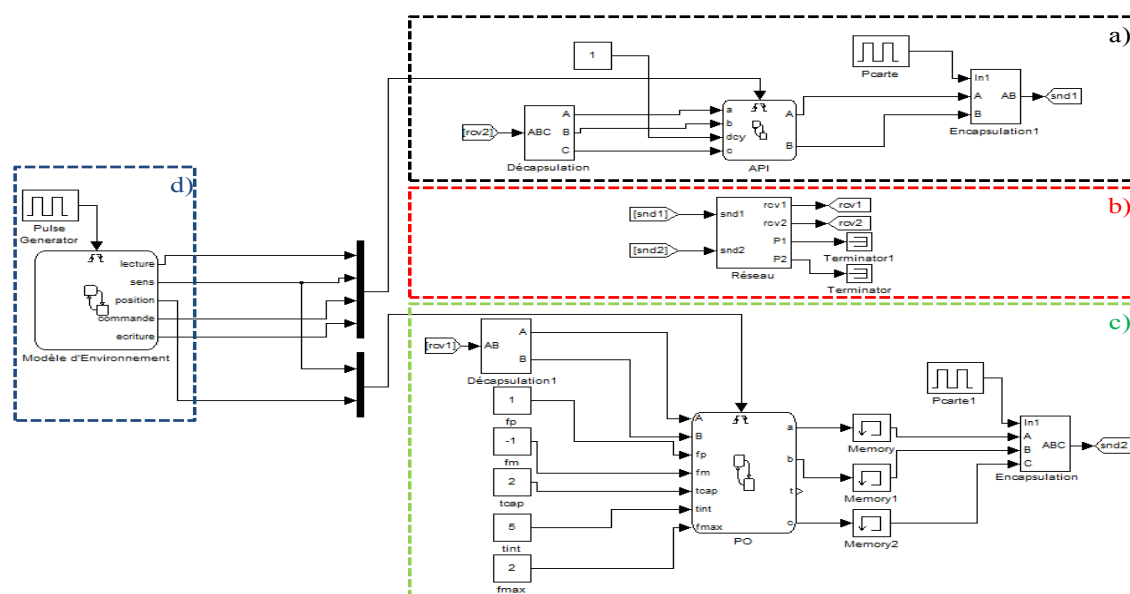


Figure 3.26 : Modélisation du cas d'étude sur *Truetime modifiée*

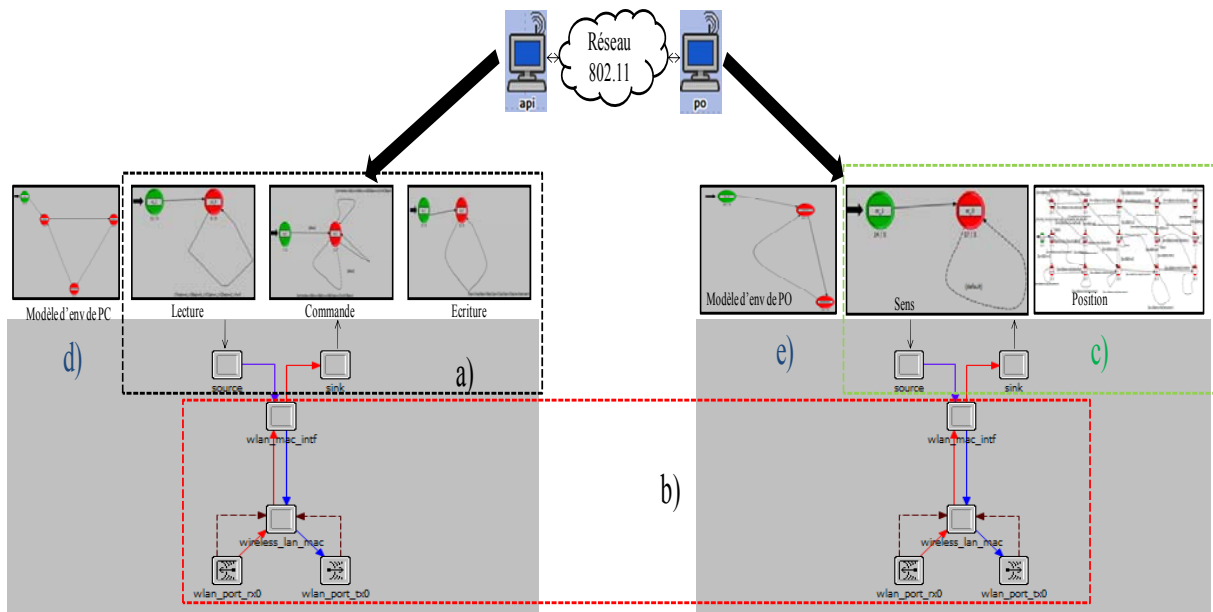


Figure 3.27 : Modélisation du cas d'étude sur OPNET

### 3.4.2.5 Comparaison des résultats sur OPNET et *TrueTime modifiée*

Après la modélisation sur les deux outils, nous allons simuler le cas d'étude en fonction des attributs du réseau et comparer les résultats. Nous définissons deux cas de simulation, cas 1 et 2, chacun est spécifié par des attributs du réseau différents. Pour chaque cas, le résultat observé est  $T_s$  (le temps d'arrêt du vérin dès qu'il reçoit l'ordre d'arrêt) en fonction de  $P_{carte}$ . Pour chaque  $P_{carte}$ , chaque simulation est répétée 50 ou 100 fois selon l'écart entre les valeurs obtenus de  $T_s$ . Dans les figures suivantes, la valeur de  $T_s$  représente la valeur moyenne des valeurs obtenues des simulations dans chaque cas.

**Cas 1 :** Débit=1Mb/s, Taille des paquets=100bytes,  $P_{api}$ = 0.02s, Taille des capteurs=0.004s, Espacement entre les capteurs =0.012s,  $P_{carte}$  varie entre 1ms et 18ms

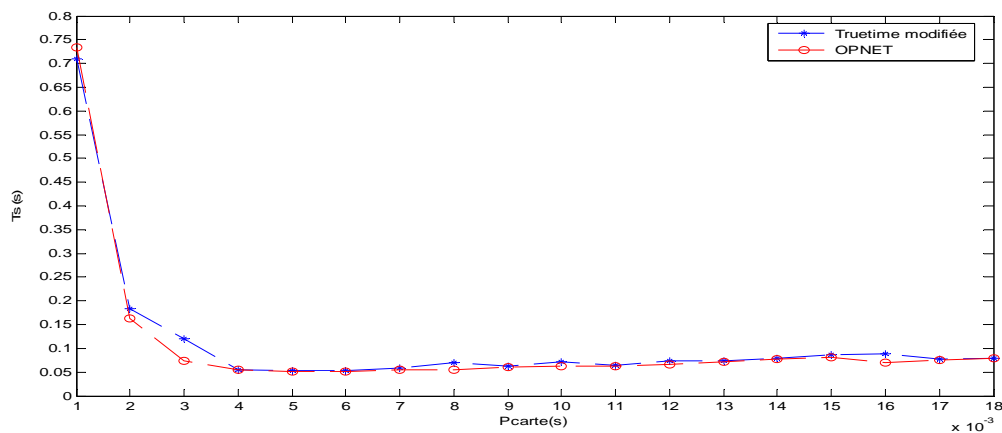
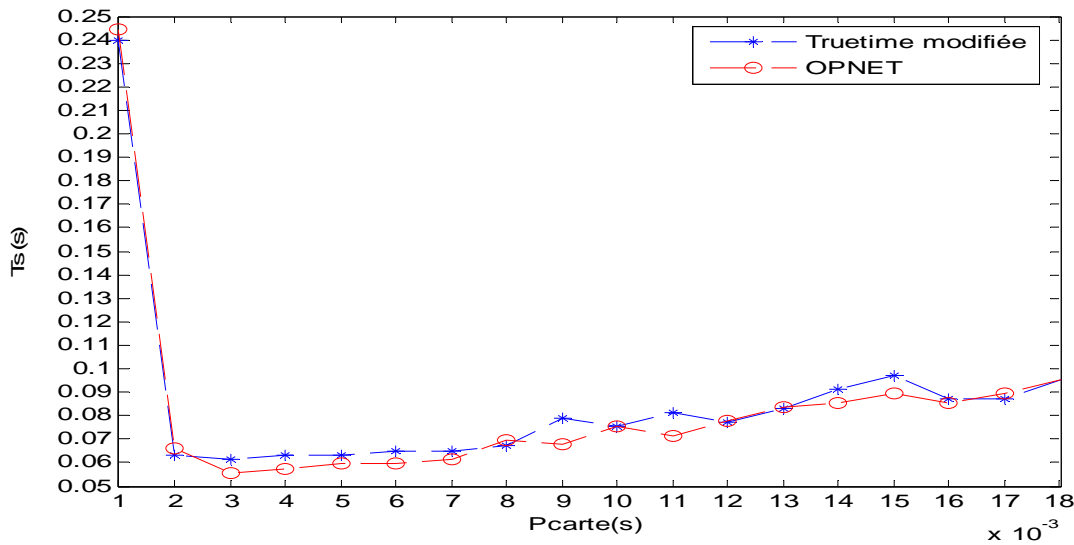


Figure 3.28 : Résultat obtenu dans le cas 1

**Cas 2 :** Débit=1Mb/s, Taille des paquets=10bytes,  $P_{api}$ = 0.02s, Taille des capteurs=0.014s, Espacement entre les capteurs =0.026s,  $P_{carte}$  varie entre 1ms et 18ms



**Figure 3.29 :** Résultat obtenu dans le cas 2

### Analyse et conclusion :

Pour les deux cas, nous remarquons la convergence entre les valeurs de  $T_s$  puisque l'écart moyen entre les valeurs dans les deux outils ne dépasse pas les 13%. Cette différence est due à la précision choisie (pas d'échantillonnage) sur MATLAB : plus j'augmente cette précision (diminuer pas d'échantillonnage) plus cet écart diminue mais l'espace d'état explose.

Cette convergence entre les résultats dans les deux outils confirme la bonne implémentation de la partie commande/PO sur OPNET ainsi que l'importance et la fiabilité des modifications faites sur la librairie Truetime.

L'interprétation de la variation de  $T_s$  en fonction de  $P_{carte}$  sera analysée dans le chapitre suivant.

Malgré ces modifications, la librairie *Truetime modifiée* présente quelques limites :

- Précision des résultats : Ce paramètre sur MATLAB est trop faible (0.1ms) par rapport à l'OPNET (1ps). Sur MATLAB, plus ce paramètre est petit, plus les résultats sont précis plus l'espace d'état croît.
- Dans *Truetime modifiée*, les caractéristiques (*taille des paquets, débit, ...*) des flux ne sont pas prises pour chaque flux, elles sont spécifiées pour le réseau en général. Cette limitation empêche les possibilités d'étudier un système dynamique dont les attributs des flux varient en fonction du temps.

### 3.5 Conclusion du chapitre

Dans le but de trouver un outil fiable pour une modélisation d'un SDCR sans fil et donc l'évaluation de ses performances, ce chapitre exploite les deux outils présentés dans le chapitre précédents. Des modifications sont ajoutées sur la librairie Truetime afin de la rendre plus proche du standard, et OPNET est étudié pour voir la possibilité d'implémenter des systèmes discrets comme la partie opérative. La convergence des résultats a confirmé la bonne modélisation dans ces deux outils. Par conséquent, ces deux outils (MATLAB-*Truetime modifiée* et OPNET) peuvent être utilisés pour présenter un SDCR sans fil. Notons que des limitations existent sur *Truetime modifiée*.

Le chapitre suivant sera consacré à étudier l'effet du réseau sur un SDCR sans fil en fonction des différents attributs existants. Un algorithme est également proposé dans le but d'améliorer la performance du système.

tel-00544932, version 1 - 9 Dec 2010

## Chapitre 4

# **Optimisation des attributs et proposition d'un algorithme afin d'améliorer la performance d'un SDCR sans fil**

## 4.1 Introduction :

Dans le chapitre précédent, nous avons déduit qu'OPNET et MATLAB-*Truetime modifiée* sont des outils efficaces pour modéliser un SDCR sans fil. Maintenant, nous pouvons étudier et améliorer la performance de ce type de systèmes. C'est l'objectif de ce chapitre. Il est divisé en deux grandes parties :

Première partie : Optimisation des valeurs des attributs d'un réseau de communication sans fil dans un SDCR sans fil

Tout d'abord, nous étudierons l'impact des attributs du réseau sur le comportement du réseau puis sur la performance du système en termes de sûreté et sécurité. Ensuite, nous nous intéressons à optimiser le choix des attributs du réseau. Afin d'atteindre ce but, nous procédons à des simulations dans un cas réel (taux de charge, perte de paquets,...) en particulier pour mettre en évidence l'impact de l'attribut *Nombre maximal de retransmission*, nous complétons ces simulations par une étude analytique dans les pires des cas en particulier pour analyser et optimiser les attributs *Pcarte*, taille des paquets. Les calculs analytiques seront établis dans cette partie pour optimiser le choix des attributs du réseau. Ces calculs sont appliqués sur le cas d'étude décrit dans le chapitre 3. Des simulations sur MATLAB-*Truetime modifiée* sont réalisées pour valider l'exactitude des calculs analytiques. Et finalement, des plages des attributs du réseau sont déduites.

Deuxième partie : Proposition d'un algorithme afin d'améliorer la performance d'un SDCR sans fil.

Rappelons que dans le chapitre 2, nous avons détaillé les trois approches existantes pour améliorer la performance du système. La première approche consiste à développer la partie commande en négligeant le comportement réel du réseau (*orientée QdC*). Les travaux menés dans la seconde approche portent sur la partie réseau. Les ressources réseaux doivent être gérées afin de diminuer les effets du réseau sur le système en termes de délais et pertes de paquets. Les besoins de la partie commande ne sont pas pris en considération dans cette approche (*orientée QdS*). La dernière approche, appelée commande-réseau, prend en compte les deux parties (réseau et commande). Dans cette approche, notre travail consiste à prendre en considération tous les attributs des différents équipements dans le SDCR sans fil (*orientée QdC-QdS*).

Dans cette partie, nous proposons un algorithme pour améliorer la performance du système dans le cas d'un réseau chargé. Cet algorithme prend en charge les besoins de la PC et le comportement du réseau. Il fonctionne d'une manière dynamique. En d'autres termes, d'après le modèle des lois de commande et la QdS courante du réseau, il affecte des priorités aux différentes stations émettrices. Pour cela, nous allons utiliser dans cette partie l'extension d'IEEE 802.11, l'IEEE 802.11e déjà détaillée dans le chapitre 1.

Dans le but de prouver l'efficacité de cet algorithme, une application académique est mise en place. Cette application présente plus de complexité que celle du chapitre 3. L'évaluation de la performance du système sera étudiée en termes de sécurité et sûreté. Vu les limitations de l'outil MATLAB-*Truetime modifiée*, la modélisation de l'application étudiée est faite sur OPNET. Les simulations montrent une amélioration de la performance du système surtout dans des cas où le réseau est surchargé.

## 4.2 Optimisation des valeurs des attributs d'un réseau de communication sans fil dans un SDCR sans fil

Dans cette section et en premier temps, nous allons étudier l'effet des attributs (*nombre maximal de retransmission, charge du réseau,...*) sur les performances du réseau global (perte de paquets,...). Dans un deuxième temps, nous allons optimiser ces attributs en regardant leurs influences sur un SDCR sans fil.

### 4.2.1 Etudier/optimiser l'influence des attributs réseau sur le comportement du réseau sans fil, (Habib, et al., 2009)

La combinaison des valeurs des attributs du réseau nous amène à des comportements différents du réseau en termes de délai et de taux de perte. Le but de cette section est de constater l'influence de ces attributs sur le réseau global. Cette section est nécessaire afin d'optimiser le choix des valeurs de ces attributs pour assurer un minimum d'influence du réseau sur le système.

Nous définissons deux nouveaux attributs : *charge du réseau* et *pourcentage de bruit*.

**Charge du réseau (%)** : c'est un attribut fusionnant *Pcarte* et *taille des paquets*. Il représente le pourcentage des débits utiles des trafics envoyés par rapport au débit autorisé. Cet attribut est formé de la manière suivante:

$$\text{Charge du réseau} = 100 * (\sum_{i=1}^N (\text{taille des paquets} / P_{\text{carte}}) \text{ de station}_i) / \text{Débit}$$

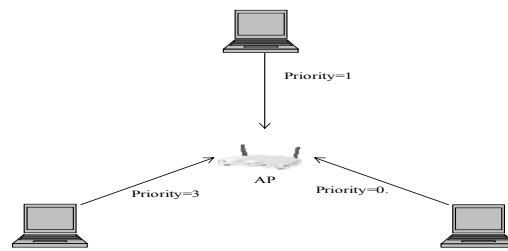
où les stations émettrices sont numérotées de 1 à N,

Station<sub>i</sub> représente la i<sup>ème</sup> station

**Taux de bruit (%)** : Dans un milieu industriel, le réseau est perturbé par des bruits externes. Ce bruit est quantifié en pourcentage. Le *taux de bruit* représente le pourcentage des paquets corrompus (reçus erronés) par rapport à l'ensemble des paquets qui transitent sur le médium. Ce taux suit une loi uniforme.

Le réseau expérimental étudié est un réseau en mode infrastructure, donc tous les paquets sont envoyés vers l'AP. Ce réseau présente la même architecture que dans un SDCR sans fil mais

sans les applications temps réels (capteur, contrôleur) de chaque station. Ce réseau est formé par trois stations qui envoient des données vers un AP. Les flux dans ces stations ont des priorités 0, 1 et 3 respectivement. Voir Figure 4.1



**Figure 4.1 : Cas d'étude**

Les valeurs des attributs choisis pour faire les simulations :

- On suppose que le *débit* est fixe, égal à 6Mb/s.
- Le *pourcentage de bruit* varie entre 0, 30, 50 et 70%.
- Le *nombre maximal de retransmission* varie entre 2, 3, 4, 5 et 6.
- La *charge du réseau* varie entre 15, 30, 60, 120%

La durée de chaque simulation est de 1 heure sur OPNET (OPNET v14.5). On suppose que cette durée est suffisante pour que nous ayons une idée du comportement du réseau. La Figure 4.2 montre les résultats après variation des différents attributs énoncés précédemment. L'information récupérée en résultat est le taux de perte. Ce dernier est égal à :

$$100 - (100 * (\text{paquets reçu par l'AP} / \sum_{i=1}^N (\text{paquets envoyés par la station}_i)))$$

### Analyse des résultats :

Dans Figure 4.2 a), nous nous situons dans un réseau à charge réduite (*charge du réseau*= 15%), donc il n'y a pas un grand nombre de paquets qui sont envoyés au canal et par conséquent, il n'y a pas un effet important sur le réseau en terme de perte de paquets. Dans ce réseau, l'augmentation du *Nombre maximal de retransmission* diminue le taux de perte, puisque la retransmission offre plus de possibilité pour que les paquets envoyés arrivent au destinataire (AP). Nous remarquons cette diminution même avec un *taux de bruit* égal à 70%.

Dans le cas où la charge est de 30% (Figure 4.2 b)), cette diminution du taux de perte continue jusqu'à ce que le *Nombre maximal de retransmission* soit égal à 4 et 5 dans le cas où le *taux du bruit* est de 50 et 70%, en raison du même phénomène qu'expliqué précédemment. Ensuite, ce taux de perte commence à augmenter avec le *Nombre maximal de retransmission*. Cette évolution est due à l'augmentation des paquets envoyés sur le médium. En effet, plus on aura un important *taux du bruit*, plus on aura de paquets perdus ou corrompus, ce qui oblige la station émettrice à retransmettre ce paquet. Cette retransmission va entraîner une charge

supplémentaire du canal. Par la suite, le réseau devient surchargé et il y aura d'avantage de possibilité de collisions entre les paquets. Par conséquent, cet effet augmente le taux de perte.

Dans le cas où la charge du réseau est de 60 et 120%, un grand nombre de paquets sont envoyés de la couche Application vers la couche MAC dans chaque station. Un autre phénomène peut alors causer des pertes : le débordement des files d'attente dans les couches MAC des stations émettrices. En plus, la retransmission de paquet amène à des collisions. Par conséquent dans la Figure 4.2 c), nous remarquons clairement cette augmentation qui suit le *Nombre maximal de retransmission*.

Résumé de ces simulations:

L'attribut Nombre maximal de retransmission peut jouer un rôle négatif dans le comportement du réseau. Ce rôle commence à être observé dans des cas où la charge du réseau est de 60 et 120%. Dans ces situations, cet attribut doit prendre une valeur minimale. Pour une charge de 30%, la valeur optimale de cet attribut dépend essentiellement du taux de bruit. Par contre, dans le cas où la charge est de 15%, la valeur de cet attribut doit être maximale afin de diminuer le taux de perte.

En général, dans un système contrôlé en réseau, la **charge du réseau** est importante surtout si le système étudié est formé par un grand nombre de stations émettrices ou par des applications qui requièrent l'utilisation de période de scrutation  $P_{carte}$  petites dans le but d'avoir plus de précision sur l'état de l'application. Par conséquent, La valeur de l'attribut *Nombre maximal de retransmission* doit prendre une valeur minimale.

Notons enfin que même s'il doit tout mettre en œuvre pour s'en prémunir, la réduction préalable du bruit doit faire partie des priorités d'un concepteur d'applications.

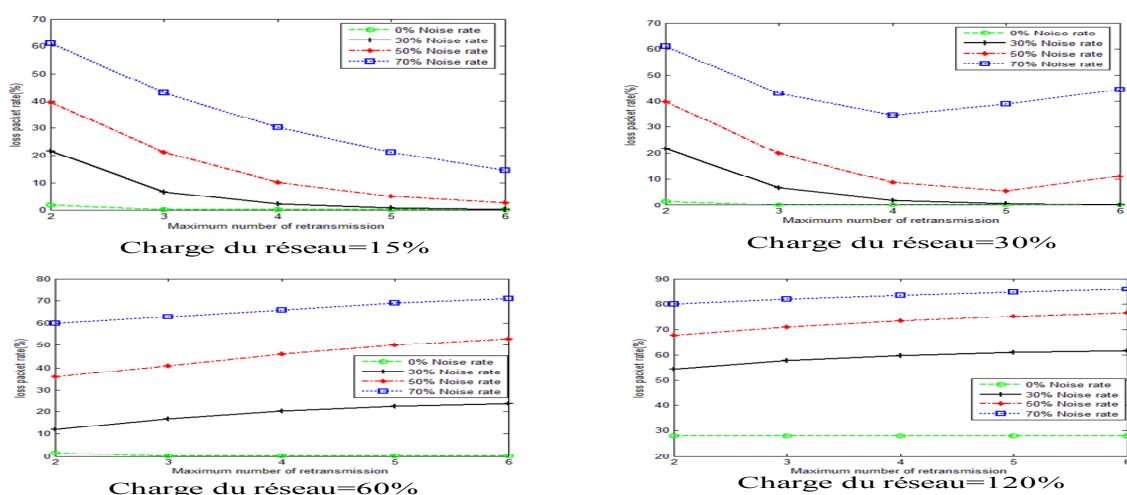


Figure 4.2 : taux de perte en fonction du *Nombre maximal de retransmission* pour différentes charge du réseau

Après optimisation du *Nombre maximal de retransmission*, nous allons ensuite optimiser le choix des autres attributs. Rappelons la conclusion obtenue dans le chapitre 1 : nous avons montré qu'il existe des plages d'attributs qu'il faut choisir ainsi que des plages qu'il faut éviter afin d'avoir une performance acceptable.

Le choix de ces attributs doit être fait dans un cas où le comportement des parties du système est le plus mauvais possible (pire des cas), plus précisément, la partie réseau induit dans ce cas un délai et taux de perte maximaux ainsi que la partie commande prend un temps maximal pour traiter les données. Par conséquent, ce choix favorisera une stabilité au système dans les cas les moins « normaux ».

L'approche par simulation nous donne une idée générale du comportement du système mais elle peut ne pas simuler toutes les possibilités du comportement, exemple : le cas où le système se trouve dans ces pires des cas. Par contre, l'approche analytique peut nous amener à des pires des cas du comportement du système. Ces cas seront évalués selon la performance acceptable du système. Par conséquent, nous choisirons les bonnes valeurs des attributs.

Pour atteindre ces cas, nous allons analyser analytiquement tous les facteurs qui forment un *ResT* (image de la performance) puis essayer de maximiser ces facteurs pour arriver à une valeur maximale de *ResT* (pire cas)

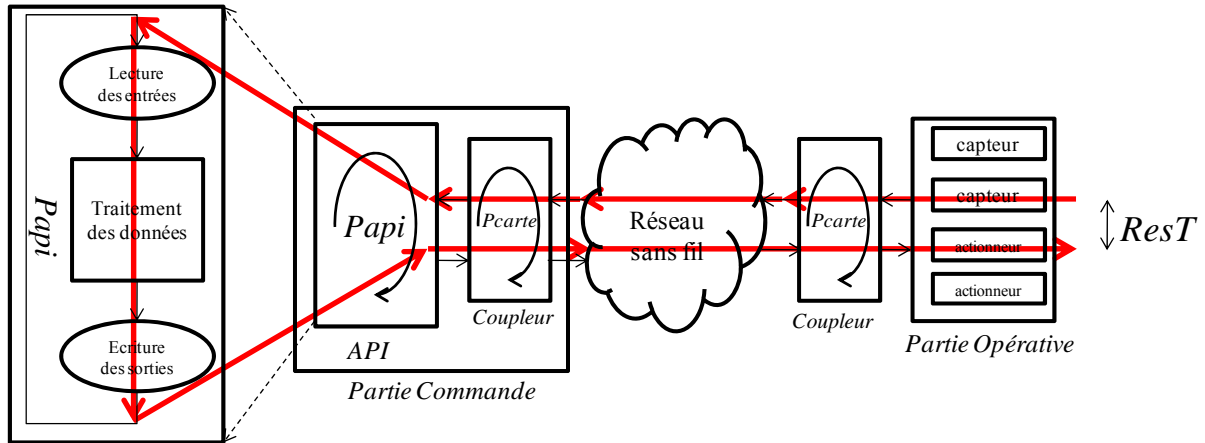
Donc la suite sera organisée comme ci :

- quantifier, de manière analytique, les temps de réponse *ResT* dans les pires cas (minimum et maximum) sur le cycle complet d'un système réactif (réception d'un stimuli en provenance des capteurs et/ou des interfaces de commande, traitement, et émission d'un ou plusieurs signaux à destination des actionneurs) en intégrant les caractéristiques d'un réseau sans fil IEEE 802.11 ou IEEE 802.11e. Notons bien que le calcul de *ResT* minimal est fait pour valider la méthode suivie pour le calcul analytique.
- valider les résultats obtenus de manière analytique par un ensemble de simulations réalisées sous l'environnement MATLAB-*TrueTime modifiée*
- évaluer l'influence des paramètres du réseau sur les temps de réponse d'un cas d'étude et optimiser le choix des attributs.

#### **4.2.2 Estimation analytique des délais de bout en bout dans le pire cas**

Rappelons le fonctionnement d'un SDCR sans fil. L'API lit les variables d'entrées, les traite et met à jour les variables de sorties selon un cycle caractérisé par une période *Papi*. Tous les échanges d'information entre les équipements du système se font à travers le réseau. Ces équipements sont munis d'un coupleur de communication afin d'accéder au médium. Ce coupleur est caractérisé par un cycle de période nommé *Pcarte*. Les flux sont majoritairement périodiques avec quelques échanges asynchrones tels que des signaux d'alarmes. Un échange

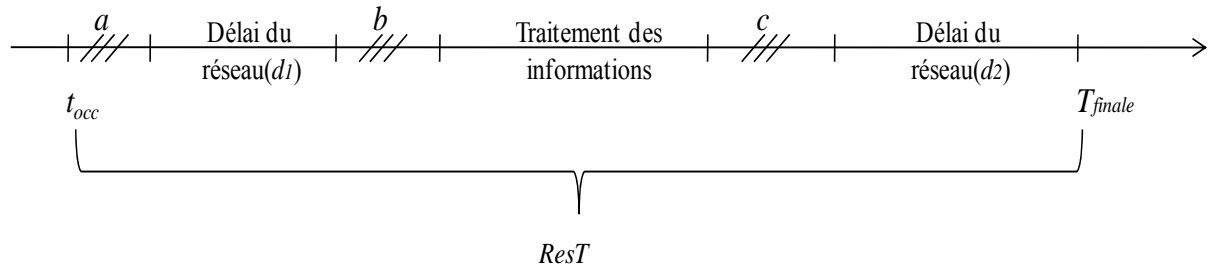
ordinaire se déroule de la manière suivante (Figure 4.3): (i) Le capteur transmet périodiquement des données vers l'API via le réseau, (ii) l'API les traite et (iii) envoie via le réseau des signaux de commande aux actionneurs en fonction des lois de commande implantées.



**Figure 4.3 : échange ordinaire d'information dans un SDCR**

Rappelons que le délai de bout en bout est le  $ResT$  qui représente le temps entre l'occurrence de l'événement généré par un capteur et la réception par un actionneur du signal de commande émis en réaction, est calculée en additionnant tous les retards survenus au cours de la transmission des données depuis la source (capteur) vers le destinataire (actionneur) à travers les différents équipements intermédiaires. Nous faisons l'hypothèse que  $n$  retransmissions d'un message,  $n$  étant le paramètre fixant le *nombre maximal de retransmissions* autorisés en cas de pertes, pourront être réalisés avant la période d'émission des messages par la PO, c'est-à-dire la période de scrutation  $P_{carte}$  associée. La Figure 4.4 décrit les étapes pour le calcul de ce délai ( $ResT$ ), (Habib, et al., 2009):

- l'occurrence d'un événement sur la PO a lieu à la date  $t_{occ}$  ;
- l'émission du message associé est retardé d'un temps  $a$  représentant le temps pour que le coupleur de communication prenne en compte cet événement ;
- l'information est transmise au destinataire (API) avec un retard  $d1$  ;
- le message est lu par l'API avec un autre retard  $b$  dû à la période du cycle ( $P_{api}$ ) pour que l'API prend en compte ce message ;
- le message est ensuite traité par l'API, l'exécution des lois de commande pouvant nécessiter plusieurs cycles ( $k$ ) avant l'obtention des signaux de commande,
- le message de commande subit ensuite un retard  $c$  dépendant de la période de scrutation du coupleur de communication ;
- enfin, le message arrivera à destination avec un retard  $d2$  à la date  $T_{finale}$ .



**Figure 4.4 : Décomposition de  $ResT$  estimé**

Pour pouvoir évaluer ce délai, il convient:

- D'évaluer le temps de communication dans un réseau dans le pire des cas (section 4.2.2.1)
- D'évaluer les temps  $a$ ,  $b$  et  $c$  et le temps de traitement des informations (section 4.2.2.2)

Notre objectif est de calculer  $ResT$  minimal et maximal.

#### 4.2.2.1 Estimation analytique des délais maximum et minimum de bout en bout dans un réseau sans fil, IEEE 802.11e mode EDCA

**Remarque importante :** Dans ce type de réseau non déterministe, il n'y a évidemment pas de borne supérieure des délais de bout en bout. Nous appelons ici délai maximum, la borne supérieure du temps que met un paquet qui réussit à parvenir au destinataire.

L'étude de ces cas doit être déterministe et non pas probabiliste, même si la possibilité d'atteindre ces cas reste rare. Par conséquent, nous nous focalisons sur le calcul analytique des délais de bout en bout minimal et maximal dans un réseau. Ce calcul (Habib, et al., 2009) doit prendre en compte le temps nécessaire pour qu'une trame accède au médium (AIFS, *Backoff Time*), le temps de transmission d'une trame et de son acquittement, l'attribut *Nombre maximal de retransmission*, le temps d'attente d'une trame dans les files d'attente au niveau de la couche MAC.

#### Durée de transmission d'un paquet au niveau couche physique

Afin de pouvoir calculer les délais minimal et maximal, l'étude de la durée de transmission d'une trame au niveau de la couche Physique est nécessaire. Elle est aussi bien dépendante des différentes technologies utilisées (802.11a, b et g) que de la taille de la trame confiée à cette couche. Cette taille dépend des entêtes ajoutés lors du passage d'un paquet dans une couche. Au niveau de la couche application, un paquet est créé avec une taille  $x$  octets. Arrivant à la couche MAC, il est encapsulé par un entête appelé entête MAC. Le même principe se produit dans la couche Physique, le paquet reçu est encapsulé par un entête

physique. Le protocole et la technologie utilisés font varier la taille des entêtes, ce qui modifiera la durée de transmission nécessaire à l'envoi d'un paquet.

**Au niveau couche MAC**, d'après le standard 802.11e, cet entête a une taille de 288 bits mais en pratique il existe des champs inutilisés, ce qui peut réduire la taille de cet entête. En général, cet entête est d'une longueur de 240 bits.

**Au niveau couche Physique**, rappelons qu'il existe trois technologies couramment utilisées: 802.11 b, a et g.

Dans la technologie 802.11b, chaque paquet reçu est encapsulé par un préambule de 144 bits plus un *PLCP header* qui est d'une taille de 48 bits. Ces bits servent entre autre à la synchronisation entre l'émetteur et le récepteur. Toutes les stations qui se trouvent dans la même zone de communication doivent comprendre ces bits, même s'ils ont un mauvais rapport signal sur bruit et ne peuvent pas recevoir à haut débit. Ils sont donc envoyés en utilisant le plus faible débit possible de 1Mbit/s. Les bits suivants sont envoyés à 2, 5.5 ou 11Mbit/s (vitesses retenues dans le standard). Donc, pour un paquet d'une taille de  $x$  octets transmis à un débit égal à  $Débit$ , nous aurons besoin d'un temps ( $\mu s$ ) de :

$$Durée\ de\ transmission(x, Débit) = \frac{(x * 8 + 240)}{Débit} + \frac{(144 + 48)}{1}$$

En ce qui concerne les paquets d'acquiescement (ACK), ils doivent être envoyés avec le débit minimal autorisé équivalent à 1Mbit/s. La taille du paquet d'ACK est de 112 bits (au niveau couche MAC), donc :

$$Durée\ de\ transmission(ACK) = \frac{112}{1} + \frac{(144 + 48)}{1}$$

Comme, l'unité de temps (slot time) dans le 802.11b est égale à  $20\mu s$ ,  $SIFS=10\mu s$ ,  $DIFS=SIFS+2*slot\ time=50\mu s$

Dans la technologie 802.11a ou g, le temps de transmission est identique puisqu'il utilise la même technologie de transmission, à savoir l'OFDM (Orthogonal Frequency Division Multiplexing). Figure 4.5 montre les encapsulations au niveau de la couche physique.

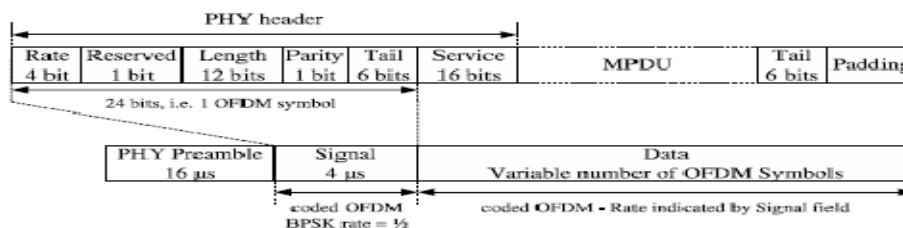


Figure 4.5: Structure de la trame en utilisant 802.11a ou g

(Krommenacker, et al., 2005) calcule la durée de transmission nécessaire, donc :

$$\text{Durée de transmission}(\mathbf{x}, \text{Débit}) = T_s * \left( 5 + \text{ceil} \left( \frac{\text{SRV} + \mathbf{x} + T_{\text{ail}}}{\text{Débit} * T_s} \right) \right)$$

Avec  $T_s = \text{OFDM symbol time slot} = 4 \mu\text{s}$ ,  $\text{SRV} = \text{Service field length} = 16 \text{ bits}$ ,  $T_{\text{ail}} = \text{Tail field length} = 6 \text{ bits}$ ,  $\text{ceil}()$  = fonction mathématique qui arrondit à l'entier supérieur le plus proche.

Pour envoyer les trames ACK, le débit choisi est le plus grand parmi {6, 12, 24 Mbit/s, dans le cas de 802.11a} et qui est inférieur ou égal au débit utilisé pour la transmission des données. Par exemple, si les données sont envoyées à 54Mbit/s, le débit utilisé pour envoyer les acquittements est 24Mbit/s. Donc en utilisant la formule précédente, la durée de transmission d'une trame ACK est de 28  $\mu\text{s}$  (avec, pour 802.11a, le slot time égal à 9 $\mu\text{s}$ , et SIFS= 16 $\mu\text{s}$ ).

Nous allons maintenant détailler la méthode suivie pour calculer le délai minimal et maximal suivant le standard. Ces résultats sont ensuite comparés à des simulations afin de les vérifier.

### Calcul du délai minimal

Pour calculer le délai minimal, nous devons nous situer dans la meilleure situation de transmission des trames. Cette situation est décrite par : *Si le canal est libre durant cette période, et si la file d'attente dans la couche liaison de l'émetteur est vide, la trame est envoyée sans attente.* Nous supposons également que le récepteur reçoit la trame dès la première tentative. Cette situation peut se produire dans n'importe quel réseau. Par conséquent, le délai minimal estimé n'est autre que le *temps de transmission* calculé précédemment puisque le paquet n'attend ni durant un *Backoff Time*, ni durant l'attente de la disponibilité du médium. D'où :

$$\text{délai minimal} = \text{Durée de transmission}(\mathbf{x}, \text{Débit})$$

### Calcul du délai maximal

Pour se placer dans ce pire des cas, il faut comprendre les différentes étapes qui forment ce délai. Rappelons le fonctionnement du CSMA/CA: *la station prend un temps Backoff Time (BT) durant lequel elle attend. Elle reste bloquée tant que le canal est occupé. Dès lors que le médium est libre pendant une durée égale à AIFS, la station recommence à décrémenter son ancien BT. Une fois que le BT atteint la valeur zéro, la station transmet la trame sans attente supplémentaire.*

$$\text{BT} = \text{rand}[0, \text{CW}] * \text{slot time}$$

avec  $\text{rand}[\ ]$  est une fonction  
qui suit une loi de distribution uniforme

D'après le standard, nous déduisons que le délai est dû à la valeur du BT choisi, au nombre de fois où la station s'arrête de décrémenter son BT si le médium est occupé et à l'AIFS. Afin d'estimer le délai maximal, il suffit de maximiser ces valeurs. Nous proposons donc un algorithme pour le calcul du délai maximal:

La station émettrice étudiée est notée par *station étudiée*. Nous supposons qu'il existe N stations émettrices différentes de la *station étudiée* notées  $station_i$  avec  $i=1, \dots, N$

- la *station étudiée* prend une valeur maximale de BT, égale à CW,
- La *station étudiée* envoie sa trame un *Nombre maximal de retransmission* de fois,
- les autres stations qui veulent envoyer des données ( $station_i$ ), prennent des  $BT_i$  de manière aléatoire pourvue que  $BT_i$  respecte les conditions suivantes:

$$(AIFS(AC_i) + BT_i) station_i < (AIFS(AC_{station\ étudiée}) + BT) station\ étudiée$$

$$(AIFS(AC_i) + BT_i) station_i \neq (AIFS(AC_j) + BT_j) station_j$$

Les valeurs de  $BT_i$  sont prises de telles façon afin d'arrêter un nombre maximal de fois la décrémentement du BT de la *station étudiée*.

Le code est écrit sous MATLAB, il prend en compte la priorité, la taille des paquets, le *Nombre maximal de retransmission*, le *Débit* et *Pcarte*. Cet algorithme tient compte également du temps d'attente par les trames dans les files d'attente de la couche MAC.

### Application du calcul analytique sur un cas d'étude

L'estimation du délai est faite sur le réseau étudié dans la section 4.2.1 dans le cas de 15% de charge. La Figure 4.6 présente les délais maximal et minimal en fonction de *Nombre maximal de retransmission* pour la station à priorité 1.

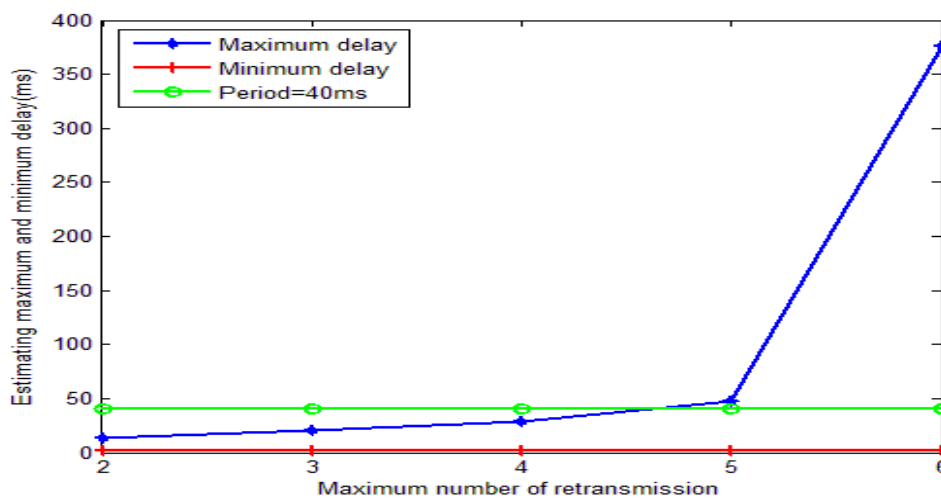


Figure 4.6 : Délai maximal estimé en fonction du *Nombre maximal de retransmission* attribut

La ligne rouge (+) représente le délai minimal, puisque nous considérons que le paquet arrive dès la première tentative, donc ce délai est indépendant du *Nombre maximal de retransmission*. De plus, chaque trame est envoyée directement sans attendre un *backoff time*. Pour ces raisons, le délai minimal est une ligne horizontale.

Par rapport au délai maximal (ligne bleue, ♦), nous remarquons que ce délai augmente d'une manière linéaire jusqu'à ce que le *Nombre maximal de retransmission* soit égal à 4, puis il augmente exponentiellement pour les valeurs 5 et 6. Cette augmentation est due au délai qui dépasse *Pcarte* (ligne verte, ○). Plus simplement, quand une trame reste dans la file d'attente de la couche MAC un temps plus grand que *Pcarte*, une autre trame arrive de la couche supérieure. Cette trame doit attendre la transmission de l'ancienne trame avant de commencer sa procédure de backoff, d'où l'ajout d'un délai supplémentaire. Par la suite, plus le *Nombre maximal de retransmission* est grand, plus les trames qui arrivent dans la file d'attente souffrent de ce délai d'attente supplémentaire. Par conséquent, leur délai maximal augmente exponentiellement. Plusieurs travaux (Nilsson, 1998) estiment que le délai maximal ne doit pas dépasser *Pcarte* afin que ce délai soit toujours acceptable.

Pour valider notre estimation, une comparaison est faite avec les valeurs de délais maximales obtenus sur OPNET (Figure 4.7) pour le même réseau étudié précédemment avec une charge de 15%. Afin d'avoir des valeurs importantes du délai maximal, nous introduisons des taux de bruit de valeurs moyennes différentes. Nous remarquons dans tous les cas que les délais maximaux simulés ne dépassent pas les délais maximaux calculés (ligne bleue, ♦) pour les trois flux, ce qui nous permet de valider notre travail.

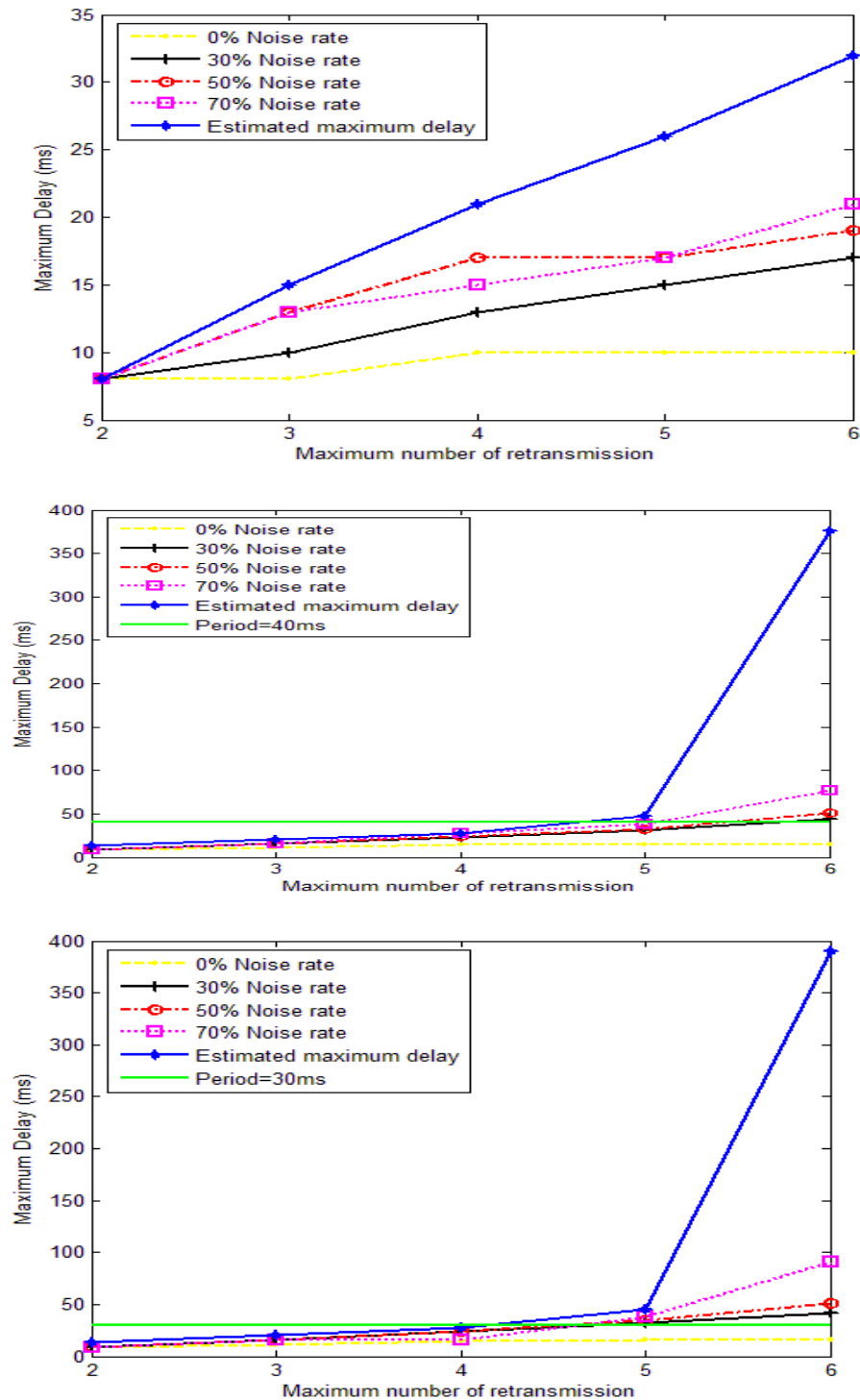
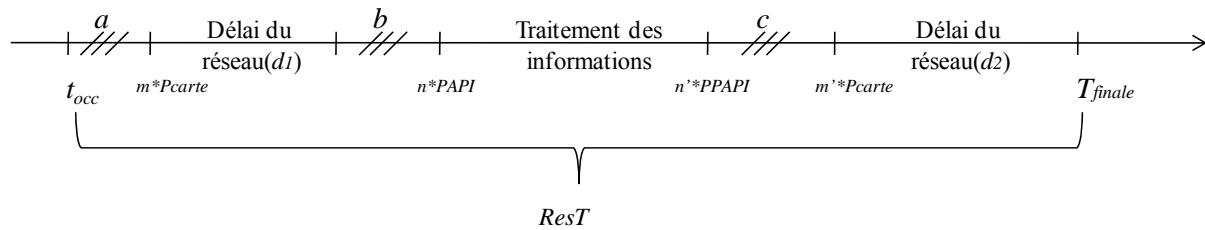


Figure 4.7 : délai maximal simulé sur OPNET pour les 3 flux

En conclusion, nous avons pu calculer par la méthode proposée le délai maximal et minimal, Ceci va nous servir au calcul de  $ResT$  minimal et maximal

#### 4.2.2.2 Calcul de $ResT$ maximal et minimal

La Figure 4.8 rappelle les différentes étapes pour estimer la valeur de  $ResT$



a représente le temps pour que le coupleur de communication prenne en compte cet événement  
 b est un retard dû à la période d'échantillonnage ( $P_{carte}$ ) du coupleur de communication de l'API  
 c dépend de la période de scrutation du coupleur de communication

**Figure 4.8 :  $ResT$  simulé**

Le temps de réponse de bout en bout depuis l'occurrence d'un événement capteur jusqu'à la réception du message associé de commande par un actionneur sera donné par  $ResT = T_{finale} - t$  selon :

$$ResT = a + d_1 + b + k * P_{api} + c + d_2 \quad [4.1]$$

$$\begin{cases} a = m_1 * P_{carte} \\ b = m_2 * P_{api} \\ c = m_3 * P_{carte} \end{cases} \quad \text{avec } 0 \leq m_i \leq 1, i=1,2,3$$

où  $d_1$  et  $d_2$  représentent le retard induit par le réseau.

Le coupleur de communication peut prendre en compte immédiatement l'événement déclenché par le capteur. Par conséquent, la valeur de « a » sera égale à zéro. A l'inverse, cet événement peut aussi attendre un temps maximal de  $P_{carte}$  pour qu'il soit pris en compte, donc « a » varie entre 0 et  $P_{carte}$ . D'où l'équation  $a = m_1 * P_{carte}$ . Le même principe est utilisé pour calculer « b » et « c ».

Le temps de traitement est égal à  $k * P_{api}$  avec  $0 < k < k_1$ , où  $k_1$  représente le nombre maximum de cycle pour le traitement de l'information.

D'après la section précédente qui estime le délai de bout en bout entre émetteur et récepteur pour un réseau sans fil IEEE 802.11e en mode EDCA, ce délai est toujours borné entre  $D_{min}$  et  $D_{max}$

Ces estimations conduisent à encadrer les valeurs  $d_1$  et  $d_2$  de l'équation [4.1] :

$$D_{1min} \leq d_1 \leq D_{1max}$$

$$D_{2min} \leq d_2 \leq D_{2max}$$

$ResT$  est minimum lorsque l'occurrence de l'événement émis par le capteur a lieu de manière synchrone à la période du coupleur de communication à une date multiple de  $P_{carte}$ . Dans ce cas, la valeur de  $a$  sera donc nulle. Cette information traverse le réseau en un temps minimum  $D_{1min}$ . De la même manière, le temps  $ResT$  sera minimum si l'arrivée du message au niveau du coupleur de l'automate programmable est synchrone avec la période de ce dernier

avec  $b = 0$ , si l'API traite cette information en un seul cycle, et enfin si le message est instantanément émis ( $c = 0$ ). Le destinataire (PO) reçoit la réponse après un délai minimum  $D_{2min}$ . La valeur de  $ResT$  minimale est donc donnée par l'équation [4.2] suivante

$$ResT \geq D_{1min} + Papi + D_{2min} \quad [4.2]$$

A l'inverse,  $ResT$  est maximum lorsque les coupleurs lisent où émettent les informations avec un temps maximal égal à  $Pcarte$ . De manière similaire, les temps de transmissions (aller et retour) seront choisis respectivement égaux  $D_{1max}$  et  $D_{2max}$ . Enfin, le traitement des informations par l'API nécessitera un nombre maximum de cycle égal à  $k_1 + 1$  (1 cycle pour la prise en compte des entrées et  $k_1$  cycles pour leur traitement). Le  $ResT$  maximal sera donc donné par l'équation [4.3] suivante :

$$ResT \leq Pcarte + D_{1max} + (k_1 + 1) * Papi + Pcarte + D_{2max} \quad [4.3]$$

Notons que cet intervalle dépend de  $Papi$ ,  $Pcarte$ , et indirectement au travers des valeurs établies  $D_{1min}$ ,  $D_{2min}$ ,  $D_{1max}$  and  $D_{2max}$ . qui dépendent de la charge du réseau, du *nombre maximum de retransmission* des paquets et de la *taille des paquets*.

Ces calculs doivent être vérifiés par des simulations sur un cas d'étude. Nous choisissons le même cas d'étude étudié dans le chapitre 3, vérin horizontal avec trois capteurs et une PC. L'outil utilisé est MATLAB-*Truetime modifiée*

#### 4.2.2.3 Application des résultats analytiques sur un cas d'étude

Au temps  $t_{occ} = t_{s2}$ , le vérin arrive au capteur  $S_2$  (occurrence de l'événement  $s_2$ ), un message est envoyé à l'API qui répond par un message de commande ( $B=0$ ) avec un délai  $ResT$  égal à  $T_{finale} - t_{s2}$  selon les calculs analytiques présentés en section 4.2.2, ce qui, au final, provoque l'arrêt du vérin après un temps d'inertie  $T_{inertie}$ . Rappelons que  $T_s$  est la date d'arrêt réel du vérin.

$$T_s = t_{s2} + ResT + T_{inertie} \quad [4.4]$$

Le temps  $T_{smax}$  correspond aux valeurs maximales des temps  $t_{s2}$ ,  $ResT$  et  $T_{inertie}$ .

$t_{s2}$  représente le temps requis par le vérin pour atteindre le capteur  $S_2$ . Comme son départ de  $S_1$  dépend de  $ResT$ ,  $t_{s2}$  dépend aussi de  $Pcarte$  (l'information « vérin sur  $S_1$  » transmise à l'origine des temps ne déclenche le départ physique du vérin qu'après un  $ResT$  et une inertie mécanique).

La valeur maximale de  $T_{inertie}$  ( $T_{maxinertie}$ ) correspond à deux cycles de l'API compte tenu des caractéristiques technologiques du capteur (plage de détection) et du vérin (vitesse de course). Pour plus de détails, se référer à (Marangé, 2008)

$$T_{maxinertie} = 2 * Papi \quad [4.5]$$

La valeur maximale de  $ResT$  a été calculée par l'équation [4.3]. Dans notre cas, les paramètres du réseau, les priorités et les périodes de scrutation  $P_{carte}$  des coupleurs de communication sont identiques pour les messages émis par la PO à destination de l'API ou inversement d'où  $D_{1max} = D_{2max}$ . Enfin, dans la commande proposée, l'API génère la commande appropriée en un seul cycle de calcul. La valeur maximale de  $ResT$  peut donc être donnée par :

$$ResT_{max} = 2 * P_{carte} + 2 * D_{1max} + 2 * P_{api} \quad [4.6]$$

La valeur  $T_{smax}$  peut donc être établie comme suit :

$$T_{smax} = t_{S2max} + 2 * P_{carte} + 2 * D_{1max} + 2 * P_{api} + T_{inertie} \quad [4.7]$$

De la même manière, la valeur de  $T_s$  sera minimale ( $T_{smin}$ ) pour la valeur minimale de  $ResT$  donnée par l'équation [4.2] avec  $D_{1min}$  égal à  $D_{2min}$  et pour un temps d'inertie supposé nul. Il en découle que:

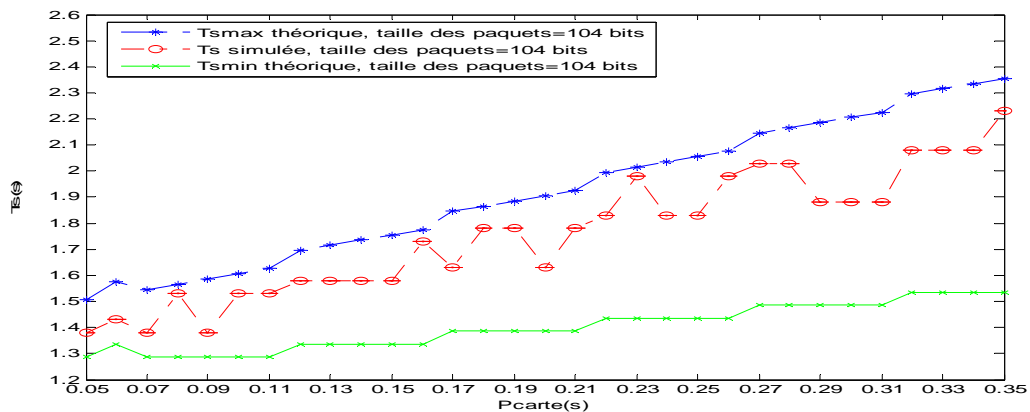
$$T_{smin} = t_{S2min} + 2 * D_{1min} + P_{api} \quad [4.8]$$

#### 4.2.2.4 Validation des résultats par des simulations

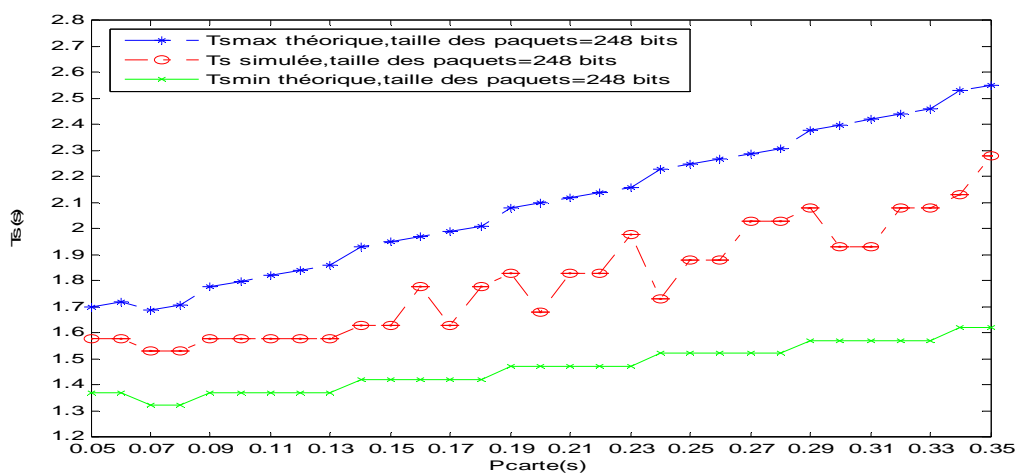
- Les pertes de paquets causées par les interférences, les obstacles et les radiations ne sont pas considérées.
- Le *Débit* est fixé à 11Mbits/s, c'est le débit maximal autorisé pour l'IEEE 802.11b-couche physique.
- le *nombre maximum de retransmission* est fixé à 1 car nous nous intéressons plutôt aux réseaux à charge importante et d'après l'étude faite dans la section 4.2.1, cet attribut doit prendre une valeur minimale (égale à 1).
- $P_{api}$  est fixée à 0.02s.
- L'approche de Monte Carlo est utilisée, chaque simulation est répétée 50 ou 100 fois selon l'écart entre les résultats obtenus.

Notons bien que pour chaque valeur de  $P_{carte}$ , nous obtenons dans les simulations trois valeurs de  $T_s$  simulé (exemple dans la Figure 4.9, c)), elles représentent la valeur minimale, moyenne et maximale simulé de  $T_s$ .

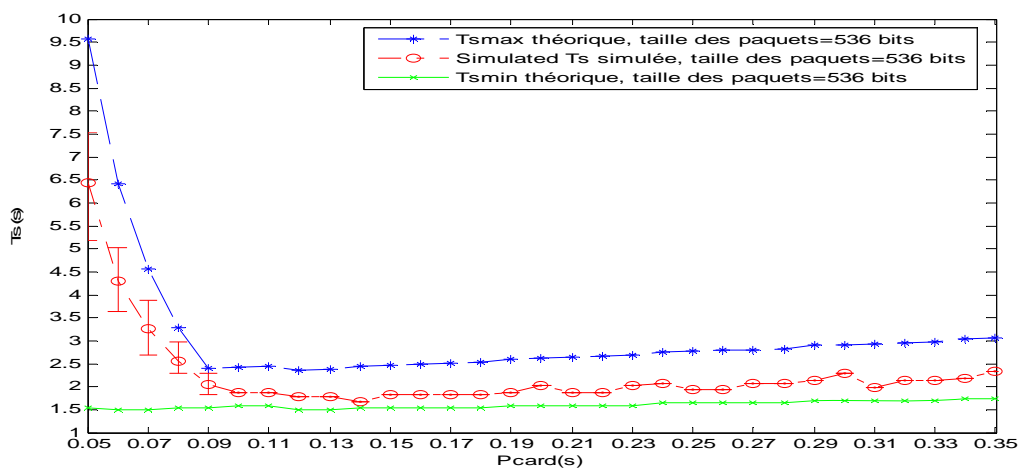
L'influence de  $P_{carte}$  sur  $T_s$  pour trois *tailles de paquets* égale à 104, 248 et 536 bits est étudiée. Les résultats présentés par la Figure 4.9 permettent de confirmer que la valeur  $T_s$  reste bien comprise entre les valeurs  $T_{smin}$  et  $T_{smax}$  calculées de manière analytique. Cette confirmation nous valide les calculs analytiques proposés.



a) taille des paquets = 104 bits



b) taille des paquets = 248 bits



c) taille des paquets = 536 bits

Figure 4.9 : Valeur de  $T_s$ ,  $T_{smin}$  et  $T_{smax}$  en fonction de  $P_{carte}$

### 4.2.3 Interpretation des résultats et optimisation des attributs

Nous allons interpréter le cas où la taille des paquets est égale à 536 bits (Figure 4.9, c)), pour chaque valeur de  $P_{carte}$ , nous avons trois valeurs de  $T_s$ , qui correspondent aux valeurs minimale, maximale et moyenne obtenues dans les simulations. Trois zones peuvent être identifiées :

- $0.05 < P_{carte} \leq 0.10s$  : dans cet intervalle, l'augmentation de la période du coupleur fait décroître fortement  $T_s$  ; ceci peut s'expliquer par le fait que la charge générée sur le réseau est réduite, ce qui entraîne une diminution des délais de transmission.
- $0.10 < P_{carte} \leq 0.35s$  : dans cet intervalle la charge du réseau est faible. C'est pour cela qu'on voit que les valeurs simulées minimale, moyenne et maximale de  $T_s$  sont identiques. Le temps de réponse reste relativement constant.
- $P_{carte} > 0.35s$  : le vérin peut ne jamais s'arrêter ( $T_s = \infty$ ) parce que le capteur S2 n'envoie pas l'information indiquant sa présence ( $s_2 = 1$ ) à la partie commande. En effet, ceci correspond au cas où le vérin est présent sur la zone de détection du capteur entre 2 scrutations.

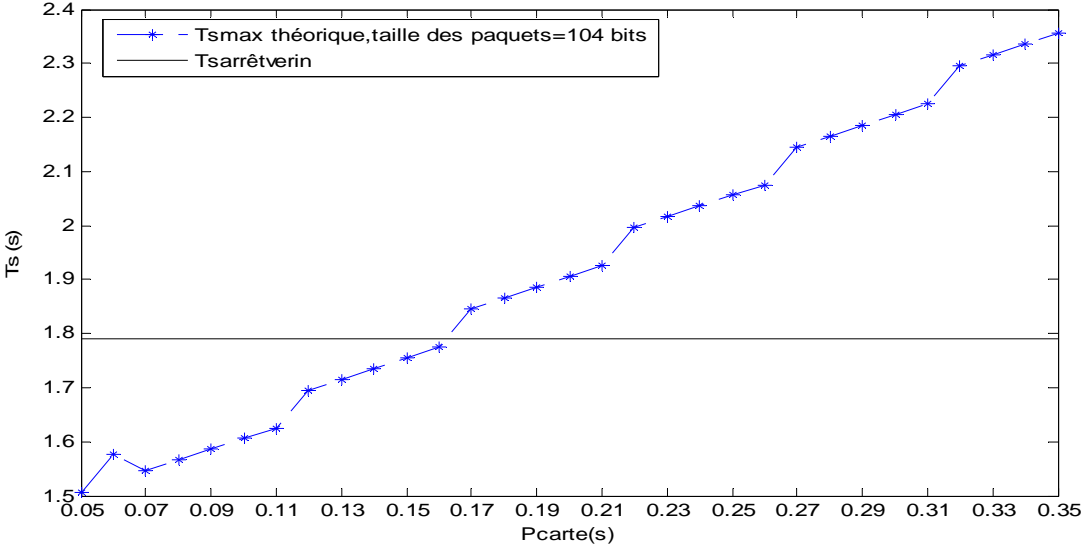
Les profils des courbes des Figure 4.9 a) et b) sont similaires à ceux de la Figure 4.9 c) dans le cas où  $0.10 < P_{carte} \leq 0.35s$ . Dans ces cas, la charge du réseau est faible (pour cela, on voit que les valeurs simulées minimale, moyenne et maximale de  $T_s$  sont identiques) donc son influence est négligeable. Plus  $P_{carte}$  est grande, plus le temps d'attente de l'information dans la file d'attente dans le coupleur est important. Ceci explique l'augmentation de  $T_s$  en fonction de  $P_{carte}$ .

Rappelons que le but de cette étude (première partie de ce chapitre) est d'optimiser le choix des attributs du réseau.

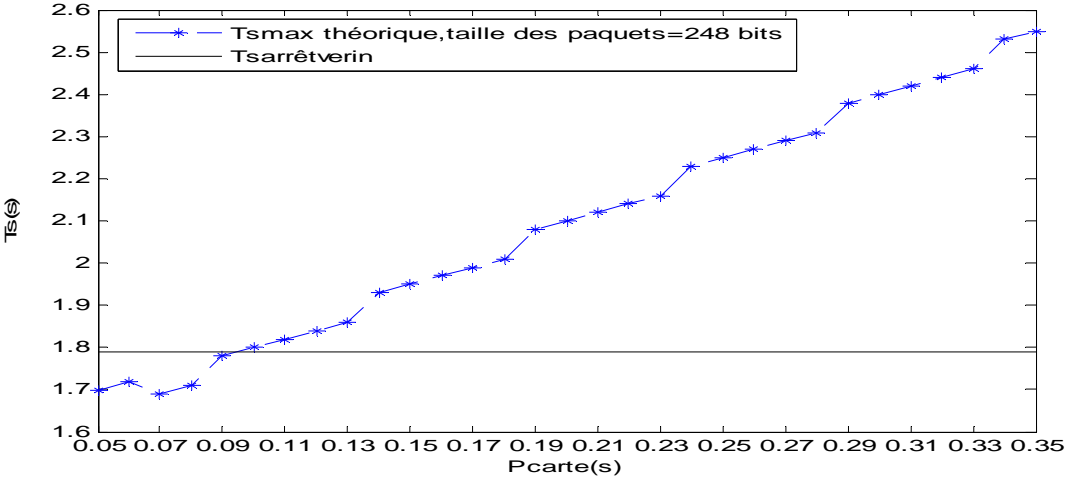
Nous nous intéressons donc aux propriétés comportementales du SDCR sans fil. Pour atteindre les objectifs de la commande (performance acceptable), c'est-à-dire que le vérin s'arrête effectivement sur le capteur S2, et compte tenu des caractéristiques technologiques de la PO (vitesse du vérin, inertie, plage du capteur...), il a été mesuré que le  $T_s$  doit être inférieur à 1,79s ( $T_{arrêtcylinder}$ ) (= temps nécessaire par le vérin traverse S1, l'espacement entre S1&S2 et le capteur S2). Pour cela, nous allons regarder les résultats des pires des cas, plus précisément, le  $T_{smax}$  théorique. Les attributs du système seront choisis dans ce cas. Ainsi, nous serons sûrs que le comportement du système appartiendra à une zone de comportement acceptable, Figure 4.10.

Dans notre cas,  $T_{smax}$  théorique est comparé avec  $T_{arrêtcylinder}$ . Si  $T_{smax}$  théorique est plus petit que  $T_{arrêtcylinder}$  donc dans le pire des cas le vérin s'arrêtera au capteur intermédiaire. Nous pouvons donc en conclure qu'une taille de paquet trop importante (en l'occurrence dans notre cas 536 bits) peut conduire à un comportement non admissible du système commandé. En revanche, les valeurs de  $T_{smax}$  théorique correspondant aux tailles de paquets égales à 104 et 236 bits proposent, quant à elles, quelques plages des attributs admissibles,  $P_{carte}$  entre 0.05 et 0.16 et  $P_{carte}$  entre 0.05 et 0.09 respectivement. En effet, pour ces valeurs de  $P_{carte}$ ,

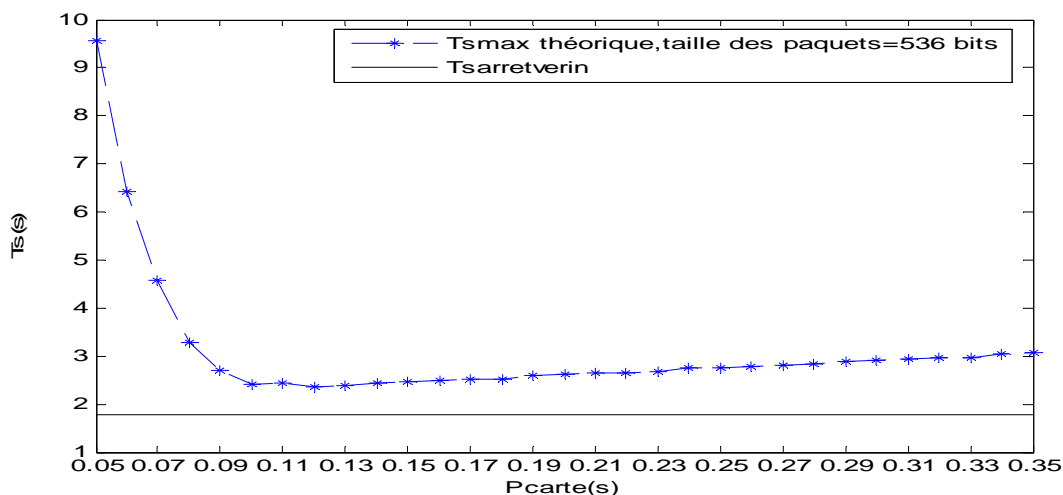
la valeur de  $T_{smax}$  théorique est inférieure au temps maximum admissible de  $T_{arrêtcylinder}$  donc vérifie l'objectif initial de la commande. C'est donc ces valeurs qu'il conviendra de choisir lors de la définition des attributs du SDCR sans fil.



a) taille des paquets = 104 bits



b) taille des paquets = 248 bits



c) taille des paquets = 536 bits

Figure 4.10 : Variation de  $T_s$  en fonction de  $P_{carte}$

#### 4.2.4 Conclusion de la première partie:

Dans cette partie, nous avons évalué le comportement temporel d'un SDCR sans fil. Afin d'optimiser le choix des attributs de ce système, nous avons calculé le  $ResT$  dans les pires des cas. Ces calculs sont comparés avec des simulations sur MATLAB-*Truetime modifiée*. Des plages de valeurs des attributs sont définies où la performance du système reste acceptable dans les pires des cas.

Dans notre étude, nous nous intéressons aussi à améliorer la performance du système surtout dans les réseaux à grande charge (le cas d'étude précédent où la taille des paquets est égal à 536bits). La suite du chapitre sera dirigée dans ce sens :

Nous allons proposer un algorithme de gestion des attributs réseau en fonction des besoins de la partie commande. Cette gestion utilise les priorités des flux. Donc dans la suite, nous travaillons avec l'IEEE 802.11e

### 4.3 Proposition d'un algorithme afin d'améliorer la performance d'un SDCR sans fil

Comme nous l'avons vu précédemment dans l'étude de  $T_s$  en fonction de  $P_{carte}$ , dans le cas où la taille des paquets est égale à 248 et 104bits (donc le réseau est moins chargé que si les paquets font 536bits), le choix des attributs du réseau est important puisqu'il réduit l'influence de la QoS sur la stabilité du système. Et nous avons déduit finalement des plages de  $P_{carte}$  qui garantissent une performance acceptable du système. Dans le cas où la taille des paquets est égale à 536bits (réseau chargé), le vélin ne suit pas la commande donnée au temps (arrêt sur le capteur intermédiaire), donc le choix des attributs du réseau n'a pas d'effet dans ce cas et il ne sera pas une solution pour avoir une performance acceptable du système.

D'où l'idée de proposer un algorithme d'amélioration de la performance utile surtout dans les réseaux à grande charge. Rappelons que cet algorithme est basé sur l'utilisation du standard IEEE 802.11e, qui favorise une QoS entre les flux. Dans la suite, nous allons détailler cet algorithme (Habib, et al., 2010).

### 4.3.1 Algorithme proposé

Comme nous l'avons vu dans le chapitre 1, les priorités 802.11e peuvent être un avantage pour améliorer la performance du système. Elles donnent plus de facilités aux trafics ayant des priorités élevées pour accéder au médium. En conséquence, on peut observer une diminution du délai de bout en bout depuis la source vers le destinataire pour les trafics. Cependant, dans ce cas, les débits reçus par les destinataires augmentent. Cet avantage peut aussi être un inconvénient pour les autres trafics qui ont des basses priorités dans le sens où ils accèdent moins au médium. C'est pourquoi l'allocation des priorités entre les flux doit être optimisée. La variation dynamique des priorités entre les flux est une méthode optimale pour utiliser cette caractéristique. En d'autres termes, le système l'utilise juste quand il estime que l'information attendue n'arrivera pas à temps. Afin d'éviter une trop grande influence sur les autres trafics, le système arrêtera de faire varier la priorité quand il recevra l'information attendue dans des délais acceptables. Pour améliorer les performances du réseau, le système peut ordonner à certains équipements d'arrêter de transmettre si leur information n'est plus utile à la PC. Par conséquent, moins d'équipements tentent d'accéder au médium.

Rappelons qu'il existe quatre priorités (0, 1, 2, et 3) dans l'IEEE 802.11e. La priorité 0 est la plus petite et la priorité 3 la plus grande. Une étude importante faite par (Villalón, et al., 2008) montrent que le délai de bout en bout provient essentiellement du temps AIFS. Dans 802.11e, pour les priorités 0, 1, 2 et 3, AIFS est égal respectivement à 7, 3, 2 et 2. Par conséquent, nous pouvons clairement remarquer que l'AIFS de priorité 0 est le plus grand et présente un grand écart avec celle de priorités 1, 2 et 3. Pour éviter des délais trop importants, le système fera donc varier les priorités des flux entre 1, 2 et 3.

Rappelons que le SDCR sans fil étudié est en mode infrastructure, donc que tous les flux passent par le point d'accès. Ce point d'accès est supporté par l'API. L'algorithme est implémenté dans l'API. Il propose que la PC gère les variations des priorités des flux issus des capteurs. Cette variation sera fonction de l'intérêt courant des informations qu'ils transmettent, de l'état du réseau et de l'état courant du modèle de commande.

Au début, nous considérons que tous les trafics ont la même priorité appelée *priorité par défaut*. L'algorithme proposé définit dynamiquement trois ensembles d'équipements basés sur la criticité des informations qu'ils envoient et élaborés selon la séquence suivante:

- La PC choisit les équipements qui envoient des *informations critiques*. Ces dernières sont essentielles pour elle pour prendre une décision immédiate. Pour cela, ces informations doivent arriver à temps.

- La PC choisit ensuite les équipements qui n'envoient pas d'informations importantes pour sa décision immédiate mais peuvent avoir une influence sur les prochaines décisions. On les appelle *informations normales*.
- Et finalement, la PC choisit les équipements, dont les informations n'influent pas sur la décision à court ou moyen terme, On les appelle *informations non importantes*.

Formellement, rappelons que le *modèle des lois de commande* représente l'évolution de la commande. Il utilise les automates communicants. Ces derniers sont définis par  $(\Sigma, X, x_0, \delta, F)$ . Nous définissons  $\Sigma_{PO}$ , un sous-ensemble de  $\Sigma$  défini comme l'ensemble des événements venant des équipements du PO. Dans notre cas, ces équipements sont les capteurs qui envoient des informations à la PC. Notons que:

$$\Sigma_{PO} = \{s_{1fm}, s_{1fd}, s_{2fm}, s_{2fd}, \dots, s_{ifm}, s_{ifd}\} \subset \Sigma$$

Chaque capteur<sub>i</sub> est capable de générer deux événements,  $s_{ifm}$  qui désigne le front montant du signal émis par le capteur (présence du vérin) au dessous de ce capteur, et  $s_{ifd}$  qui désigne le front descendant de ce signal (absence du vérin). Périodiquement (*Papi*), l'algorithme choisit trois ensembles d'événements  $C$ ,  $N$  et  $U$  qui sont basées sur l'état courant du modèle de PC appelé state  $x_{cur} \in X$ . Chaque ensemble sera utilisé pour gérer d'une manière spécifique les priorités des flux issus des capteurs.

**Définition 1.a:**  $C$  est un ensemble d'événements liés à des *informations critiques*, i.e.:

$$C = \{s \in \Sigma_{PO} : \exists w \in (\Sigma \setminus \Sigma_{PO})^*, \delta(x_{cur}, ws) \text{ est défini}\}$$

où  $(\Sigma \setminus \Sigma_{PO})^*$  représente tous les mots composés d'événements appartenant à  $\Sigma$  et n'appartenant pas à  $\Sigma_{PO}$ .

**Définition 1.b:**  $N$  est un ensemble d'événements liés à des *informations normales*, i.e.:

$$N = \left\{ s' \in \Sigma_{PO} : s' \notin C, \exists w, w' \in (\Sigma \setminus \Sigma_{PO})^*, s \in \Sigma_{PO}, \right. \\ \left. \delta(x_{cur}, wsw's') \text{ est défini} \right\}$$

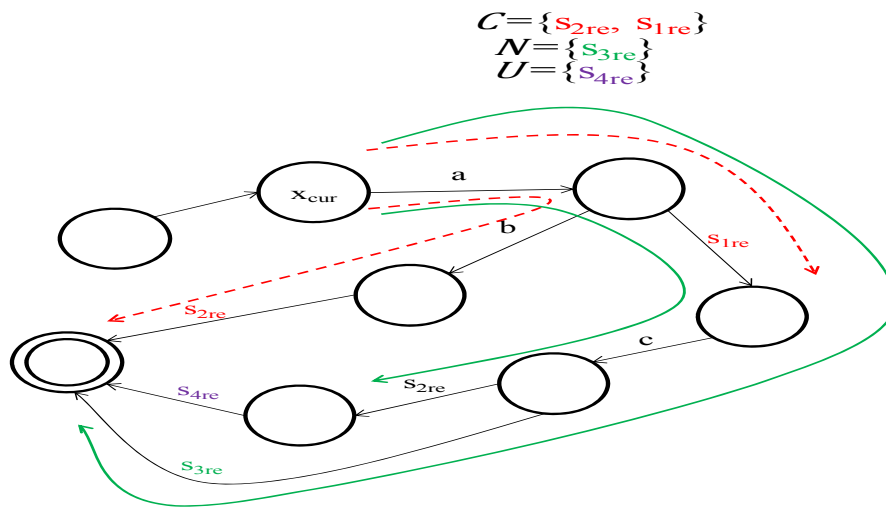
**Définition 1.c:**  $U$  est un ensemble d'événements liés à des *informations non importantes*, i.e.:

$$U = \{s'', s'' \in \Sigma_{PO} \setminus (C \cup N)\}$$

Exemple de détermination de ces trois ensembles d'événements :

La Figure 4.11 est un exemple de modèle des lois de commande. Nous supposons que ce modèle est dans l'état  $x_{cur}$ . Afin de déterminer les éléments du premier ensemble  $C$ , l'algorithme cherche tous les mots dont les éléments, sauf le dernier, n'appartiennent pas à  $\Sigma_{PO}$ . Dans cet exemple les flèches pointillées montrent 2 mots :  $a-b-s_{2re}$  et  $a-s_{1re}$ . Donc  $s_{2re}$  et  $s_{1re}$  appartiennent à  $C$ . Pour former l'ensemble  $N$ , le même principe est suivi à partir des états

atteints par les mots précédents, à condition de ne pas déjà appartenir à l'ensemble  $C$ . Par exemple, à partir de  $a-s_{1re}$ , le mot  $c-s_{3re}$  positionne  $s_{3re}$  dans  $N$ . Finalement, tous les éléments restants dans  $\Sigma_{PO}$  qui n'appartiennent ni à  $C$  ni à  $N$ , seront dans l'ensemble  $U$ .



**Figure 4.11: Détermination des trois ensembles d'événements  $C$ ,  $N$  et  $U$  sur un exemple d'un modèle des lois de commande**

Après cette étape, l'algorithme extrait des ensembles ( $C$ ,  $N$  et  $U$ ) trois ensembles de capteurs ( $C_{capteur}$ ,  $N_{capteur}$  et  $U_{capteur}$ ) qui envoient des *informations critiques*, *normales* et *non importantes*. Si un capteur est associé à deux événements appartenant à deux ensembles d'événements différents, il devra appartenir à l'ensemble de capteurs associé à l'événement le plus critique.

Ces trois ensembles de capteurs seront spécifiés d'une manière dynamique à chaque période de cycle de commande *Papi*. Et pour chacun, on générera différemment les priorités des flux associés :

- Dans le cas où la PC estime que le réseau est dans un état chargé, il sait que les *informations critiques* peuvent ne pas arriver à temps, ce qui risque de conduire à ce que le système ait des comportements anormaux. Pour éviter cette situation, la PC augmente la priorité du flux des capteurs qui envoient ces informations. Cette augmentation de priorité donnera au flux traité plus de possibilité d'accès au médium donc il transmettra plus vite ces *informations critiques*. L'estimation de l'état courant du réseau est faite par la surveillance du temps de réponse  $ResT_i$  de chaque capteur  $i$ .  $ResT_i$  est ensuite comparé à un seuil. Ce seuil est défini au préalable : sa valeur doit assurer un comportement normal du système. En d'autres termes, si le  $ResT_i$  dépasse ce seuil, la priorité du capteur  $i$  sera incrémentée. Si non, la priorité reste à la valeur courante. Plusieurs méthodes existent pour mesurer  $ResT_i$ , nous détaillerons la méthode utilisée dans le cas d'étude au paragraphe 4.3.2.5.

- Les capteurs qui envoient des *informations normales* ont une priorité maintenue à *priorité par défaut* (la valeur de cette priorité est égale à 1).
- Finalement, les capteurs qui envoient des *informations non importantes* surchargent le médium (c'est du moins ce que nous considérons) par l'envoi de leur différentes trames sur le médium. Ces trames contiennent des informations qui seraient ignorées par la PC. Aussi, la PC ordonne à ces capteurs d'être en veille ou en autres termes d'arrêter d'envoyer des trames. Cet ordre est transmis par la PC à ces capteurs sous la forme d'une priorité *Nul*.

En résumé, cet algorithme est implémenté dans le modèle de commande. Il est exécuté périodiquement suivant la période de l'API (*Papi*). Cette exécution génère les trois ensembles d'événements  $C$ ,  $N$  et  $U$  puis les trois ensembles de capteurs  $C_{\text{capteur}}$ ,  $N_{\text{capteur}}$ ,  $U_{\text{capteur}}$ . Il estime ensuite  $ResT_i$  pour chaque équipement appartenant à  $C_{\text{capteur}}$  et finalement calcule les priorités des flux de tous les capteurs.

---

#### Algorithme – Priorities allocation

---

*/\* initialization \*/*

$P(S) \leftarrow \text{priorité par défaut}$  */\* Tableau, de dimension k, des priorités des k*

*/\*capteurs S, initialisés à la priorité par défaut\*/*

Seuil  $\leftarrow$  default\_TH

$\text{Capteurs} = \{\text{capteur}_1, \text{capteur}_2, \dots, \text{capteur}_k\}$

*/\*l'algorithme est implémenté dans le modèle de la loi de commande\*/*

*/\*initialisation des trois ensembles d'événements\*/*

Construire  $C$  à partir de l'état courant  $x_{\text{cur}}$  du modèle de commande ;

Construire ensuite  $N$  à partir de l'état courant  $x_{\text{cur}}$  du modèle de commande ;

Construire ensuite  $U$  à partir de l'état courant  $x_{\text{cur}}$  du modèle de commande

*/\*initialisation des trois ensembles de capteurs\*/*

$C_{\text{capteur}} = \{S_i \in \text{Capteurs} : s_{\text{ifm}} \in C \text{ ou } s_{\text{ifd}} \in C\}$ ,

$N_{\text{capteur}} = \{S_i \in \text{Capteurs} \setminus C_{\text{capteur}} : s_{\text{ifm}} \in N \text{ ou } s_{\text{ifd}} \in N\}$ ,

$U_{\text{capteur}} = \{S_i \in \text{Capteurs} \setminus (C_{\text{capteur}} \cup N_{\text{capteur}})\}$

**For each**  $S_i \in C_{\text{capteur}}$  **do**

    Evaluer  $ResT(S_i)$  */\*voir § 4.3.2.5\*/*

**If**  $ResT(S_i) > \text{Seuil}$  **and**  $P(S_i) < 3$

**Then**  $P(S_i) \leftarrow P(S_i) + 1$

**Else**  $P(S_i) \leftarrow P(S_i)$

**End\_if**

**End\_for**

**For each**  $S_i \in N_{\text{capteur}}$  **do**

$P(S_i) \leftarrow \text{priorité par défaut}$

**End\_for**

```

For each  $S_i \in U_{\text{capteur}}$  do
     $P(S_i) \leftarrow \text{Nul}$  /* arrêt de transmission */
End_for
Return ( $P(S)$ )

```

---

Rappelons que la définition de  $C$ ,  $N$  et  $U$  est basée sur l'état courant du modèle de commande  $\{A\}$ . Ce dernier peut être décrit de manière modulaire par la composition synchrone de plusieurs modèles  $\{A_1, A_2 \dots A_n\}$ . Les lemmes suivants permettent d'adapter l'algorithme dans ce cas:

**Lemme 1 :** Si  $\Sigma_{A_i} \cap \Sigma_{A_j} = \emptyset$ , i.e.  $A_i$  et  $A_j$  sont des alphabets qui n'ont pas d'éléments communs, alors  $C(A_i \| A_j) = C(A_i) \cup C(A_j)$ , (resp.  $N$  et  $U$ ). Autrement dit, si les alphabets ( $A_i$  et  $A_j$ ) de deux modèles ont une intersection vide, l'énumération de  $C$  (resp.  $N$  et  $U$ ) du modèle dont l'alphabet  $A_i \| A_j$  est équivalent à l'énumération de  $C$  de chaque modèle d'alphabet  $A_i$  et  $A_j$  (resp.  $N$  et  $U$ ).

**Lemme 2:** Si  $\Sigma_{A_i} \cap \Sigma_{PO} = \emptyset$ , i.e. l'alphabet  $A_i$  ne contient pas d'événements venant des capteurs alors,  $C(A_i) = N(A_i) = U(A_i) = \emptyset$ .

### 4.3.2 Cas d'étude

Afin d'étudier l'impact des attributs du réseau/contrôle dans un SDCR sans fil, une application particulière est réalisée (Figure 4.12). Une partie de cette application ressemble à celle étudiée avant. Cette nouvelle application présente des contraintes plus intéressantes puisqu'elle contient plus de stations émettrices donc le réseau sera plus chargé et qu'elle introduit une contrainte applicative supplémentaire (éviter la collision entre les deux vérins). Cette application est formée d'une PC (API) qui envoie des ordres et reçoit des informations de la partie opérative. Cette dernière est composée de deux vérins : le vérin1 se déplace horizontalement alors que le vérin2 se déplace verticalement. Dans le trajet du vérin1, trois capteurs ( $S1$ ,  $S2$  et  $S3$ ) sont mis en place afin de détecter sa présence. De la même façon, deux capteurs sont installés ( $S4$  et  $S5$ ) sur le trajet du vérin2. Les deux vérins partagent une zone commune. Tous les capteurs (capteur <sub>$i$</sub> ) possèdent des cartes E/S réseau sans fil qui envoient périodiquement ( $=P_{\text{carte}}$ ) des données booléennes appelées  $s_i$  (présence du vérin, événement  $s_{\text{ifm}}$ , ( $s_i=1$ ) ou absence du vérin, événement  $s_{\text{ifd}}$ , ( $s_i=0$ )) à la PC.

Cette configuration est peu réaliste industriellement et technologiquement pour un vérin mais peut être représentative des nouvelles architectures de commande basées sur des actionneurs et capteurs sans fil.

L'utilisation d'une carte E/S réseau par capteur (au lieu de lier tous les capteurs sur une carte) a pour objectif d'augmenter le nombre de stations émettrices, et par conséquent, de charger le réseau plus rapidement. Ceci a pour effet d'avoir plus de possibilités de collisions entre les paquets envoyés et un délai induit plus important. Cette situation est intéressante pour mesurer l'efficacité de l'algorithme proposé.

La PC calcule tous les *Papi* les ordres à transmettre aux deux vérins (*avancer, reculer, arrêter*), qui correspondent aux quatre variables booléennes (A, B, E, D). Elle crée des paquets avec différents champs. Chaque champ correspond à un vérin spécifique. Les ordres sont mis dans ces champs et les paquets sont diffusés à tous les équipements du réseau d'une manière périodique (=P*carte*). La méthode de diffusion minimise le nombre de paquets envoyés par la PC. Après la réception du paquet diffusé, chaque vérin lit son propre champ. Le vérin1 lit le champ 1 qui correspond aux variables Booléennes A et B. Ces variables représentent l'ordre de déplacement (*avancer, reculer, arrêter*). De la même façon, le vérin2 lit le champ 2 qui contient les deux variables booléennes E et D (Figure 4.13).

Un nouveau paramètre est défini appelé *tour (run)*. Il représente l'aller/retour du vérin (déplacement de sa position initiale à sa position finale et retour de sa position finale à sa position initiale). Pendant la simulation, les deux vérins sont en mouvement : ils font des *tours*. Durant ces déplacements, les vérins ne doivent pas être en même temps dans la zone commune. Sinon, il y a collision.

L'objectif est d'effectuer le nombre maximal de *tours* des deux vérins sans collisions.

Pour éviter ces collisions, lorsque le vérin2 arrive dans la zone commune, la PC ordonne au vérin1 de s'arrêter dès qu'il arrive sur le capteur S2. Quand le vérin2 a fini son tour, le vérin1 aura la priorité de se déplacer et donc de continuer son tour. Dès que le vérin1 quitte la zone commune, le vérin2 aura le droit de commencer un nouveau tour.

A cause de l'influence du réseau, les ordres de la PC ou les informations venant des capteurs peuvent être en retard. De plus, les deux vérins peuvent être présents en même temps dans la zone commune. Donc deux paramètres principaux sont étudiés pour évaluer:

- La sécurité : nombre de collisions qui n'ont pas pu être évitées en raison de l'influence du réseau.
- La performance : nombre de *tours (run)* que les deux vérins ont pu faire durant un temps précis puisque les paramètres réseaux peuvent influencer sur la vitesse des vérins.

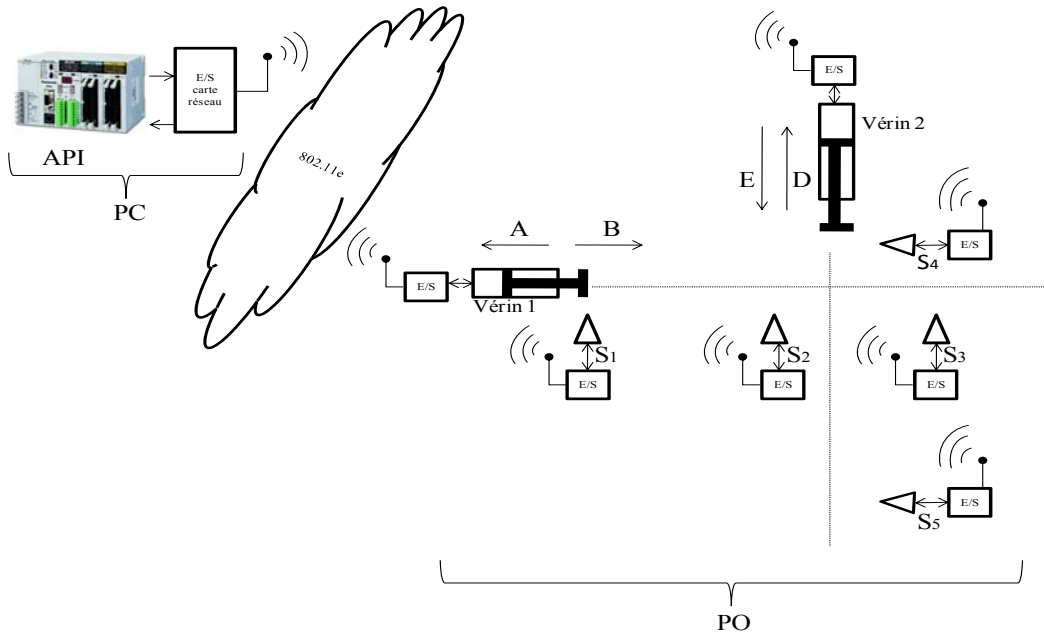


Figure 4.12: Architecture du SDCR sans fil du cas d'étude traité

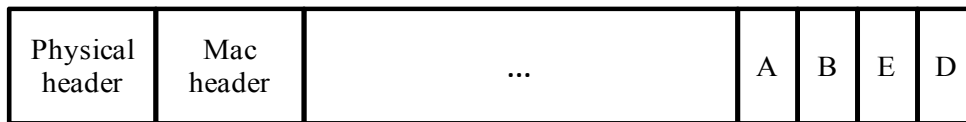


Figure 4.13: Le format du paquet de diffusion sans utiliser l'algorithme proposé

#### 4.3.2.1 Modélisation du système étudié sur OPNET

Rappelons que les deux outils (MATLAB-*Truetime modifiée* et OPNET) peuvent être utilisés pour représenter un SDCR sans fil. Cependant à cause des limitations de l'outil MATLAB-*Truetime modifiée* énoncées dans le chapitre 3, l'implémentation de l'algorithme proposé se fera sur OPNET.

Dans cette partie, la modélisation du cas d'étude expliqué précédemment est détaillée. Comme il a été vu dans les chapitres précédents, la modélisation du SDCR sans fil est partagée en trois parties : la partie opérative formée des vérins et des capteurs, partie réseau qui modélise tout le comportement des différents couche OSI de chaque station dans le réseau avec les effets du médium. Finalement, la partie commande pilote la partie opérative. Nous décrivons également l'implémentation de l'algorithme proposé sur OPNET.

#### 4.3.2.2 Modélisation de la Partie Opérative

Dans cette étude, la partie opérative est formée des deux vérins : 3 capteurs sont associés au vérin1 et 2 capteurs pour le vérin2. La modélisation du comportement des vérins est déjà expliquée dans les chapitres précédents. Compte tenu du fait que les capteurs et les actionneurs possèdent tous leurs propres coupleurs de communication, ils utiliseront des

stations différentes pour émettre et recevoir des paquets. Se pose alors un problème : les fronts montants et descendants des capteurs correspondent au franchissement d'une transition dans le modèle d'actionneur associé à une station « actionneur ». Pour que ces événements soient vus par les stations « capteur », nous utilisons une variable logique partagée entre une station « actionneur » et une station « capteur ». Cette variable sera mise à 1 lors du front montant d'un capteur et mise à 0 lors de son front descendant. Compte tenu de l'évolution de la position de l'actionneur, la station « capteur » pourra ainsi immédiatement émettre cette variable à destination de la commande.

#### 4.3.2.3 Modélisation du réseau

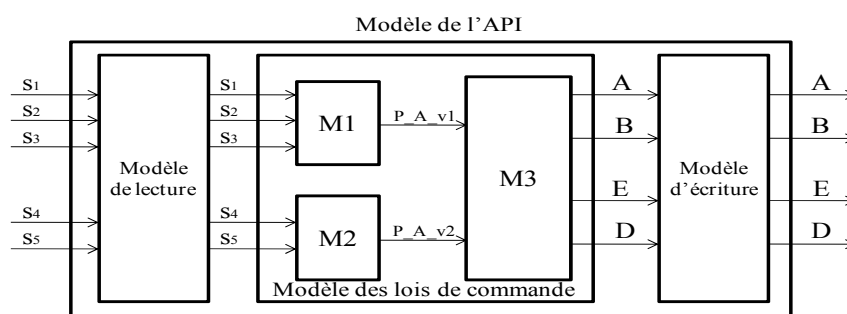
Ce modèle présente le comportement du réseau. Rappelons que chaque station est formée de trois couches du modèle OSI à laquelle s'ajoute l'application utilisatrice.

- Couche Physique: Cette couche reçoit et transmet les signaux. Dans ce cas d'étude, nous utilisons le 802.11b.
- Couche MAC: nous choisissons le 802.11e afin d'appliquer l'algorithme proposé
- Couche Application: Elle joue l'intermédiaire entre les couches basses et l'application utilisatrice.
- Application utilisatrice: dépend de type de la station étudiée : capteur, vérin,...

Les attributs classiques suivants sont paramétrables : *Taille des paquets*, *Période d'échantillonnage*( $P_{carte}$ ), *Nombre maximal de retransmission*, *Débit*

#### 4.3.2.4 Modélisation du commande

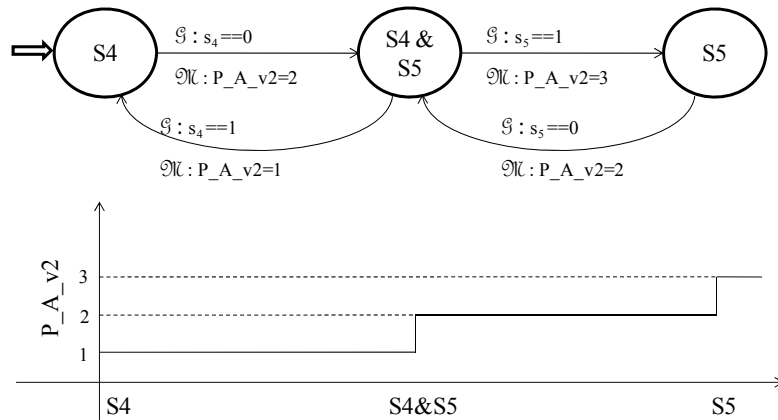
Afin d'éviter l'explosion de l'espace d'état dans le *modèle des lois de commande*, on a divisé ce modèle en trois parties : M1, M2 et M3. Voir Figure 4.14



**Figure 4.14 : Modèle de l'API**

M1 et M2 sont des ensembles d'états qui décrivent la position des deux vérins respectivement 1 et 2. Chaque état correspond à une zone de présence du vérin. La Figure 4.15 présente le modèle M2, avec la variation de  $P\_A\_v2$  en fonction de l'état courant de M2. La variable  $P\_A\_v2$  est la sortie du modèle M2. Elle varie entre 1, 2 et 3. Les entrées de ce modèle sont

les valeurs venant des capteurs S4 et S5. La sortie est la valeur P\_A\_v2. Au début le vérin2 est à proximité du capteur S4 (P\_A\_v2=1). Lorsque ce modèle reçoit un front descendant de S4 ( $s_{4fd}, s_4=0$ ), il interprète que le vérin2 a quitté le capteur S4 et qu'il est entre S4 et S5. Il franchit la transition entre l'état S4 et l'état S4&S5 et met ainsi à jour la valeur de P\_A\_v2. Dans ce cas, P\_A\_v2 sera égale à 2. De la même façon, il y a franchissement vers l'état S5 dès qu'il reçoit une présence du vérin2 sur le capteur S5 ( $s_{5fm}, s_5=1$ ) et P\_A\_v2 prend alors la valeur 3. Le modèle M1 a été construit suivant le même raisonnement. Notons bien que le modèle M1 est formé de 5 états et que P\_A\_v1 varie de 1 à 5.



**Figure 4.15 : Architecture du modèle M2**

M3 est un ensemble d'états qui appartient à toutes les combinaisons possibles des ordres envoyés. M3 prend P\_A\_v1 et P\_A\_v2 comme variables et envoie des ordres vers la partie opérative (Figure 4.16). Chaque état porte les mises à jour des ordres (B, A, E et D) destinés aux vérins (nous avons procédé ainsi pour minimiser la charge des transitions). Par exemple, l'état 1010 signifie que : B=1, A=0, E=1 et D=0. Nous trouvons 9 combinaisons possibles pour ces 4 variables. C'est pourquoi nous remarquons 9 états dans la Figure 4.16 en plus d'un état initial. Des variables internes et externes sont définies pour les transitions entre ces états.

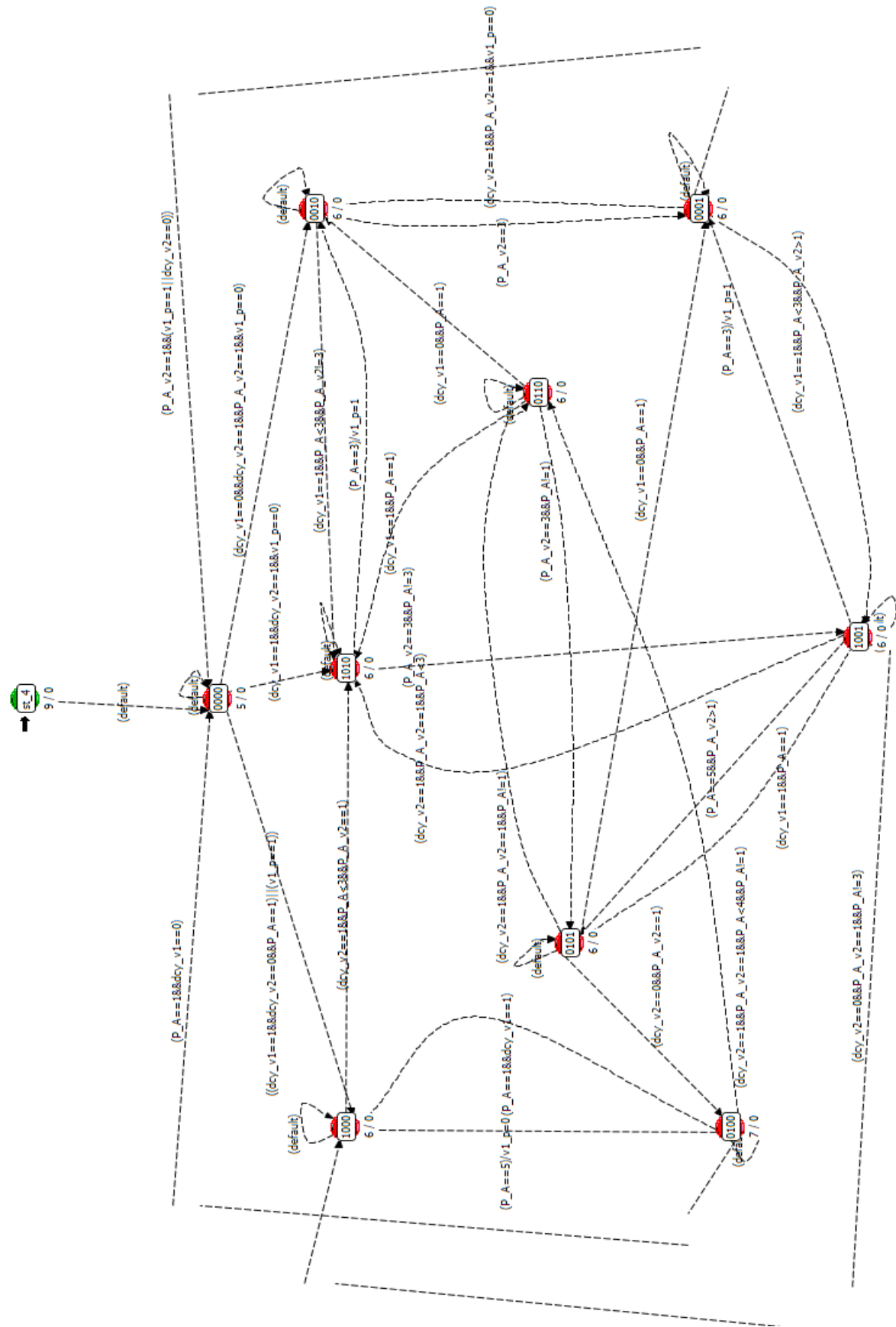


Figure 4.16 : M3 modèle description

Les variables externes sont :

- P\_A\_v1, P\_A\_v2, représentent les positions des vérins estimés par M1 et M2
- dcy\_v1, dcy\_v2 : Les deux vérins sont mis en marche par l'action des boutons dcy\_v1 et dcy\_v2. Afin d'être sûrs de la modélisation, nous avons simulé le système sans le réseau et fait varier dcy\_v1 et dcy\_v2 afin de couvrir tous les cas possibles et corriger tous les cas imprévus. Après cette étape, nous avons fixé ces boutons pour qu'ils soient toujours appuyés donc dcy\_v1= dcy\_v2=1

Les variables internes sont :

1. v1\_p, qui représente la priorité du vérin 1 par rapport au vérin 2. Par exemple, si v1\_p=1, alors le vérin1 est plus prioritaire que vérin2 pour réaliser/continuer son tour.

#### 4.3.2.5 Implémentation de l'algorithme proposé

Pour modéliser l'algorithme proposé, l'API encapsule dans son paquet de diffusion cinq champs supplémentaires (du champ 3 au champ 7). Ces champs représentent les priorités calculées pour chaque capteur (de S1 à S5 - Figure 4.17). Chaque capteur lit son propre champ et envoie son flux avec une priorité égale à la valeur de son propre champ. La valeur de ce champ peut être égale à zéro, ce qui signifie que l'API ordonne à ce capteur d'arrêter de transmettre. Ce champ peut également prendre une valeur entre 1 et 3. Notons bien que tous les trafics ont une priorité par défaut égale à 1.

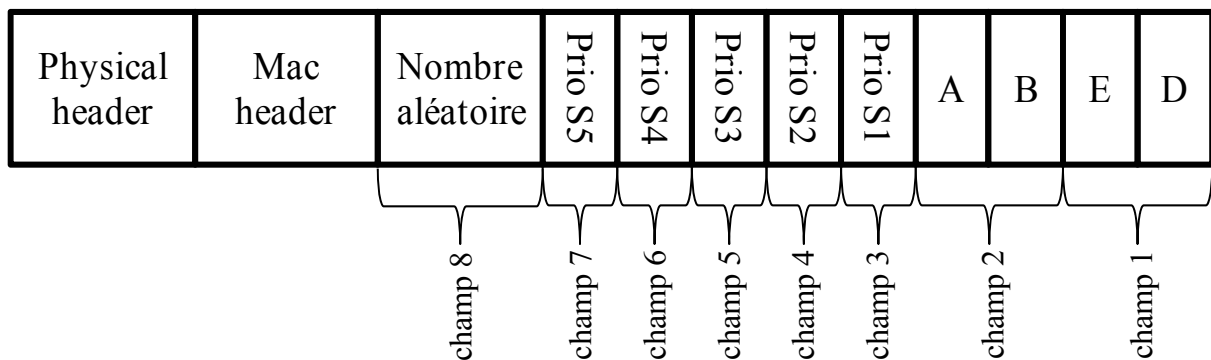


Figure 4.17 : Le format du paquet de diffusion en utilisant l'algorithme proposé

Comme nous l'avons déjà vu précédemment, cet algorithme est ajouté dans la station contenant le modèle de lois de commande. Il consiste à sélectionner trois ensembles :  $C$ ,  $N$  et  $U$  qui représentent des informations respectivement *critiques*, *normales* et *non importantes*. Les équipements envoyant ces informations sont des capteurs qui appartiennent à  $\Sigma_{PO}$ . Dans notre cas,

$$\Sigma_{PO} = \{s_{1fm}, s_{1fd}, s_{2fm}, s_{2fd}, s_{3fm}, s_{3fd}, s_{4fm}, s_{4fd}, s_{5fm}, s_{5fd}\}$$

Le passage d'un état à un autre dans M3 dépend seulement des variables  $P\_A\_v1$  et  $P\_A\_v2$ . Ces deux variables n'appartiennent pas au  $\Sigma_{PO}$ . M1 et M2 sont deux modèles qui travaillent indépendamment. En d'autres termes, il n'y a pas d'entrées, sorties ou variables internes communes. Par conséquent, et d'après les lemmes 1 et 2 section 4.3.1, l'algorithme proposé est appliqué pour chaque modèle M1 et M2. Plus précisément, on aura :

$$\begin{aligned} C &= C(M1) \cup C(M2) \\ N &= N(M1) \cup N(M2) \\ U &= U(M1) \cup U(M2) \end{aligned}$$

$\Sigma_{PO} = \Sigma_{PO-M1} \cup \Sigma_{PO-M2}$  , avec  $\Sigma_{PO-M1} = \{s_{1fm}, s_{1fd}, s_{2fm}, s_{2fd}, s_{3fm}, s_{3fd}\}$  et  $\Sigma_{PO-M2} = \{s_{4fm}, s_{4fd}, s_{5fm}, s_{5fd}\}$

Pour réaliser les simulations, nous avons choisi les valeurs des paramètres suivants :

- $ResT_i$  est comparé à un seuil, la valeur de ce seuil est supposé égale à la taille des capteurs.
- La *priorité par défaut* est la priorité 1 dans IEEE 802.11e
- Rappelons que le paramètre  $ResT_i$  n'est autre que le *round-trip time* (RTT) pour chacun des capteurs dont les événements appartiennent aux ensembles  $C(M1)$  et  $C(M2)$  auquel on ajoute le temps de traitement demandé par la PC (*Papi*). RTT est le temps mis entre l'échange d'information entre la PC et le capteur. Pour mesurer ce RTT, l'API encapsule un nombre aléatoire (champ 8) dans le paquet de diffusion (Figure 4.17). Après réception de ce paquet, le capteur lit ce nombre et l'encapsule dans le prochain paquet à envoyer. Ainsi, quand la station API reçoit ce paquet, elle calcule aisément le temps de réponse  $ResT_i$ .

Notons aussi que la station API incrémente également sa propre priorité du trafic afin d'avoir plus de possibilités d'accès au médium.

La Figure 4.18 montre la manière dont le trafic de S2 change de priorité quand le vérin1 arrive au niveau de ce capteur. L'axe des X présente le temps (en secondes). L'axe des Y présente les données envoyées (en paquets) par la couche application vers la couche MAC pour chaque priorité. L'axe des Y dans les Figure 4.18 a) et b) présente le trafic reçu par la couche MAC pour les priorités 1 et 2.

Quand le vérin est entre S1 et S2, l'algorithme met S1 dans l'ensemble  $U_{capteur}$  puisque ces informations ne sont plus importantes pour les décisions de l'API. Le capteur S3 sera quant à lui dans l'ensemble  $N_{capteur}$  puisque ces informations sont importantes pour une future décision. Par contre, l'algorithme confirme que S2 envoie des *informations critiques* (il appartient à  $C_{capteur}$ ). Par conséquent, la station API regarde le temps de réponse ( $ResT$ ) du capteur S2. Au début, la priorité des données envoyées est égale à 1 (zone 1) (Figure 4.20 a)). Après un certain temps,  $ResT$  du capteur S2 sera plus grand que le seuil défini. Par

conséquence, la PC décide d'augmenter la priorité de S2. Donc, il augmente sa priorité à 2 (zone 2), (voir Figure 4.18 a, b)). Dans ce cas, la couche application du capteur S2 envoie ces paquets dans la file d'attente de priorité 2 au lieu de 1 (Figure 4.20 b)). Après un certain temps, le vérin1 dépasse le capteur S2. L'algorithme considère maintenant que S2 envoie des *informations non importantes*. La PC ordonne à S2 d'arrêter de transmettre. Dans la zone 3, on remarque que quand le vérin1 dépasse S2, la couche application de S2 arrête d'envoyer des paquets vers la couche MAC.

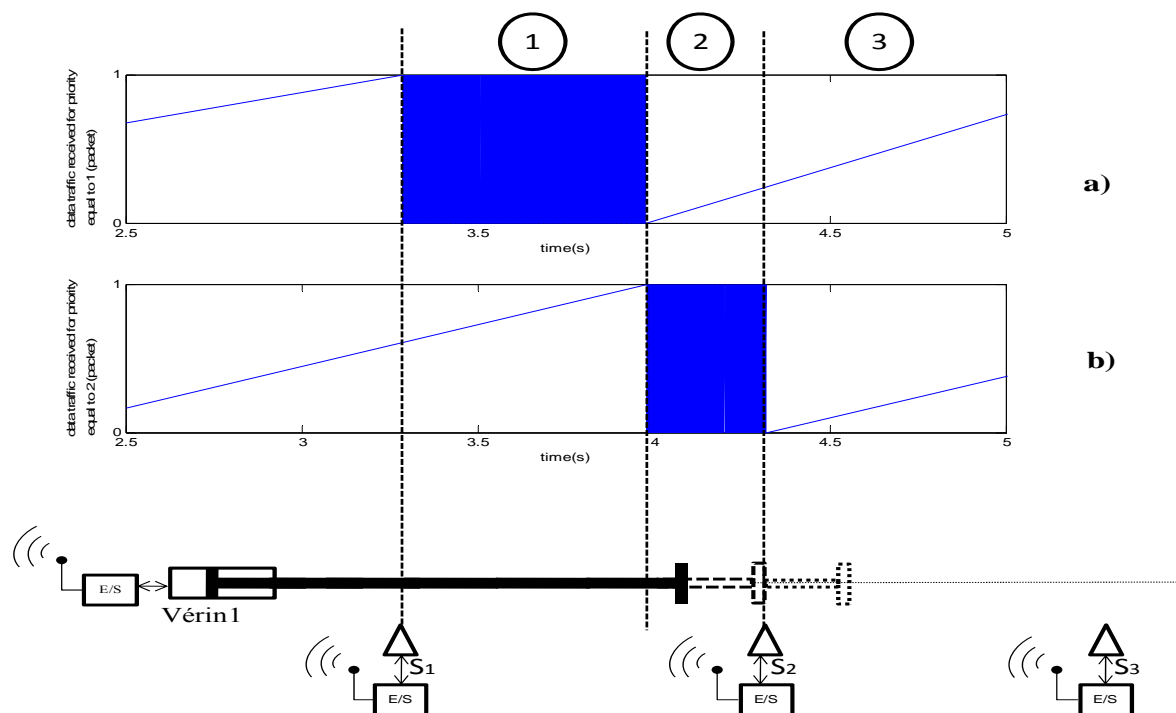


Figure 4.18: Variation du flux reçu par la couche MAC dans le capteur S2

#### 4.3.2.6 Précautions

Deux problèmes majeurs sont identifiés. Le premier se pose quand la station API envoie un ordre d'arrêt de transmission à un capteur. Ce dernier arrête d'envoyer des paquets de la couche application vers la couche MAC. Mais la couche MAC contient toujours d'anciens paquets qui tentent toujours d'accéder au médium. Si ces paquets arrivent à la station API, ils peuvent perturber l'estimation de la position du vérin. Pour éviter cette situation, la station API rejette tous les paquets qui viennent des capteurs qui ont des ordres d'arrêt.

Le deuxième problème se trouve dans le changement de priorité pour une station. Le changement de priorité consiste à basculer d'une file d'attente à une autre, c'est-à-dire que la couche application va mettre ces paquets dans une nouvelle file d'attente. Cependant, les paquets restants dans l'ancienne file d'attente tentent également d'accéder au médium. En effet, chaque file fonctionne comme si elle était seule dans la station. Par conséquent, le récepteur peut recevoir des anciens paquets et perturber l'estimation de position du vérin. Pour éviter cette situation, chaque station réceptrice sauvegarde la date de création des

paquets reçus d'une station source. La station réceptrice élimine chaque paquet qui possède une date de création plus petite que celle sauvegardée dans sa mémoire.

#### 4.3.2.7 Simulation et analyse des résultats

Après la modélisation du système sous OPNET (Figure 4.19), nous étudions ici l'influence du réseau sur le système global selon deux cas de figure :

- Premier cas : sans algorithme, donc tous les flux ont la même priorité égale à 1
- Deuxième cas : avec algorithme, donc les priorités varient en fonction de l'état de la PC et la QoS du réseau.

Voici les attributs de simulation :

Attributs	Valeurs
<i>Papi</i>	3ms
Taille des paquets envoyés par l'API	7octets
Taille des paquets envoyés par les capteurs	5octets
Nombre maximal de retransmission	1
Taille d'un capteur	30ms
Espacement entre les capteurs	1s
Débit	11Mb/s
Pcarte	de 1 à 25ms

L'approche de Monte Carlo est utilisée, chaque simulation est répétée 50 ou 100 fois selon l'écart entre les résultats obtenus. Le temps de simulation est de 200s.

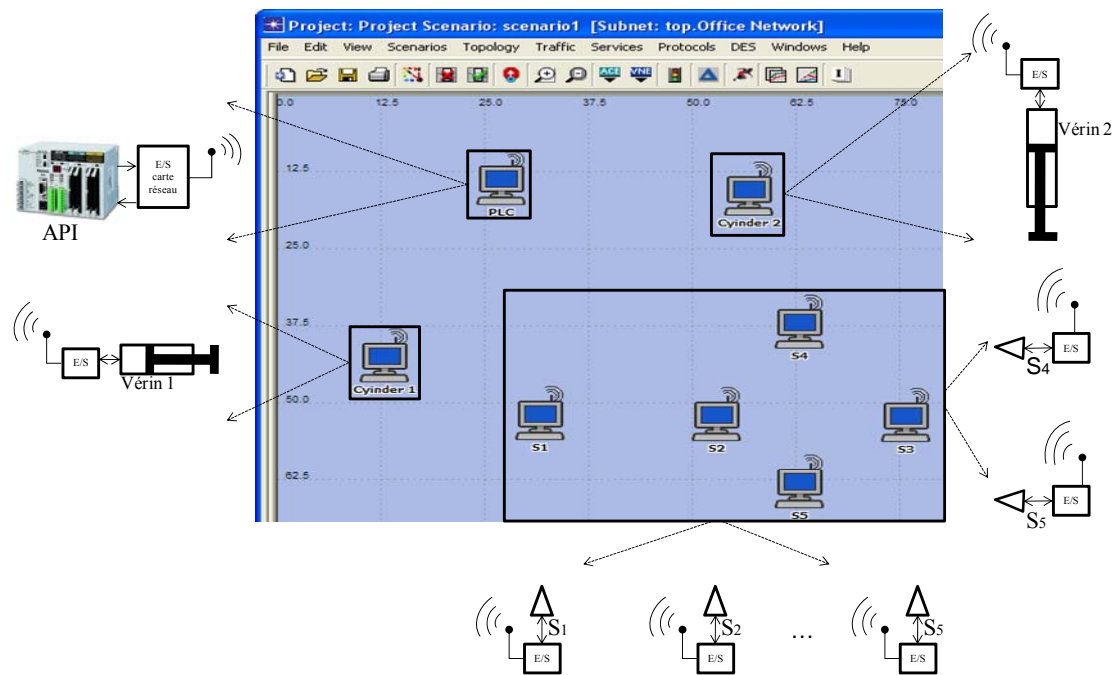
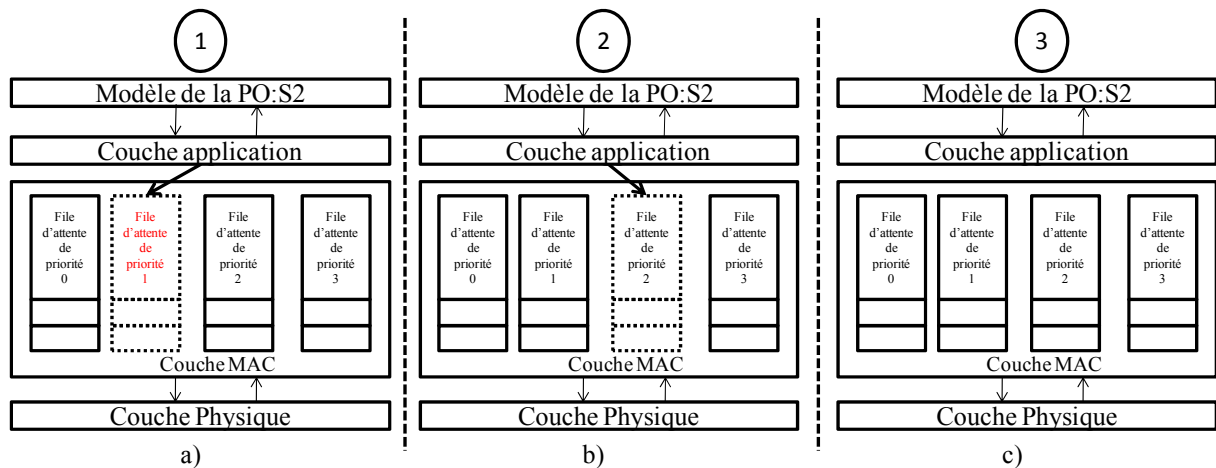


Figure 4.19: Modélisation du cas d'étude sous OPNET



**Figure 4.20 : Variation du choix de la file d'attente dans la station S2**

La Figure 4.21 (croix) représente le nombre de collisions en fonction de  $P_{carte}$  pendant une simulation. La Figure 4.22 (croix) représente le nombre de *tours (run)* réalisés par les deux vérins. Ces simulations sont faites sans l'algorithme proposé. Afin d'analyser ces courbes, nous les découpons en trois parties :

1.  $1 \leq P_{carte} \leq 4ms$ , dans les deux premiers scenarios avec  $P_{carte}$  égale à respectivement 1 et 2ms, un grand nombre de paquets sont envoyés au médium, ce qui surcharge le réseau. Il y aura des pertes de paquets et une augmentation du délai. Par conséquent, M1 et M2 vont mal estimer les positions des vérins et provoquer des ordres erronés émis par la station API. Ces ordres vont empêcher les vérins de continuer leurs tours. Nous remarquons ces résultats dans la Figure 4.22 (croix bleues), les vérins font seulement respectivement un et deux tours sur l'ensemble de la simulation. Pour 3ms et 4 ms, le délai et la perte de paquets commencent à diminuer, les vérins peuvent donc faire plus de tours. Mais le délai est toujours important donc les vérins peuvent entrer en collision, ce que nous remarquons dans les deux scenarios. Ces collisions décrémentent dans les deux cas.
2.  $5 \leq P_{carte} \leq 10ms$ , dans cet intervalle, le réseau induit un délai petit et pas de perte de paquets considérables. Donc l'influence du réseau dans ce cas est presque nulle. Ainsi, il n'y a clairement pas de collisions lors du déplacement des vérins. C'est pourquoi, nous voyons qu'ils font un nombre maximal de tours pendant les simulations. Pour optimiser les attributs du réseau,  $P_{carte}$  doit donc être choisie dans cette marge.
3.  $11 \leq P_{carte} \leq 25ms$ , dans ce cas,  $P_{carte}$  est plus grand que dans les cas précédents. L'influence du réseau est de plus en plus négligeable. Cependant, les simulations montrent des collisions importantes et une chute du nombre de tours réalisés par les deux vérins. Ces résultats sont dus au fait que les capteurs n'envoient pas l'information de « présence du vérin » ( $s_i=1$ ) à la PC. En d'autres termes, quand le vérin arrive à proximité d'un capteur, ce dernier doit indiquer sa présence. Pour envoyer cette information de la couche Application vers la couche MAC, le capteur doit attendre le front montant de la carte E/S réseau cycle ( $P_{carte}$ ). Or, le temps

d'attente est très grand puisque la période d'E/S carte ( $P_{carte}$ ) est grande. Le vérin dépassera donc le capteur. Ainsi, l'information de l'absence du vérin sera présente avant même que l'information précédente (présence du vérin) ne soit envoyée. La PC ne va donc pas recevoir de nouvelles informations de ce capteur, et n'ordonnera pas l'arrêt du vérin. La collision sera inévitable. En conclusion, cet intervalle ne sera pas choisi

C'est la première partie,  $1 \leq P_{carte} \leq 4ms$ , qui est la plus intéressante car la charge du réseau est importante. Cette situation est fréquemment rencontrée dans les applications temps-réel qui demandent une  $P_{carte}$  assez petite pour avoir plus de précision sur les valeurs mesurées. C'est dans ces cas que l'algorithme que nous proposons doit prouver son efficacité.

Les mêmes simulations sont donc réalisées en utilisant l'algorithme proposé. Les Figure 4.21 et Figure 4.22 (cercle) montrent les résultats obtenus. Nous remarquons que :

Les mêmes résultats avec et sans algorithme sont observés avec  $P_{carte}$  entre 5 et 10ms parce que l'influence du réseau n'est pas très grande. Dans le cas où  $P_{carte}$  est élevée (3eme partie), nous ne remarquons pas de changement important par rapport aux résultats sans algorithme. Les collisions sont toujours présentes et cela pour les mêmes raisons que sans l'utilisation de l'algorithme. Notons aussi que l'algorithme proposé réagit juste au niveau couche MAC, le changement de priorités et l'arrêt des capteurs minimisent l'influence du réseau sur le système global. Par contre, il n'agit pas sur les attributs de  $P_{carte}$ . C'est pourquoi, nous observons que cet algorithme ne présente pas un grand changement dans ces deux cas.

Par contre dans le premier cas ( $P_{carte}$  entre 1 et 4 ms), où la charge est importante, l'apport de l'algorithme est significatif. En effet, on n'observe plus de collisions entre les vérins et un grand nombre de tours a été réalisé. L'efficacité de l'algorithme proposé est vérifiée dans ce cas. Cependant, il est important de noter que dans le cas où  $P_{carte}$  est égale à 1ms, des collisions sont toujours présentes. Dans ce cas, la couche application envoie périodiquement un paquet toutes les millisecondes à la couche MAC. Cette dernière tente d'envoyer rapidement les paquets qui se trouvent dans les files d'attente. Mais le temps d'envoi d'un paquet avec la réception de son acquittement est d'environ  $0,5ms + \text{temps d'attente (Backoff time+AIFS)}$  (dans le cas de 802.11b-couche physique). Donc pendant chaque période (1ms), il y a maximum 2 stations qui peuvent envoyer des informations (si nous supposons que le temps d'attente est nul) et 4 stations qui n'ont pas pu envoyer. Par conséquent, les files d'attente dans les couches MAC dans les stations seront pleine rapidement et ignoreront les paquets en provenance de la couche supérieure. Or, ces paquets ignorés peuvent contenir des informations importantes. Il est alors possible de conclure que la variation dynamique des priorités des trafics et l'arrêt de certains trafics n'est pas suffisant pour obtenir une meilleure performance dans le cas où  $P_{carte}$  est égale à 1ms.

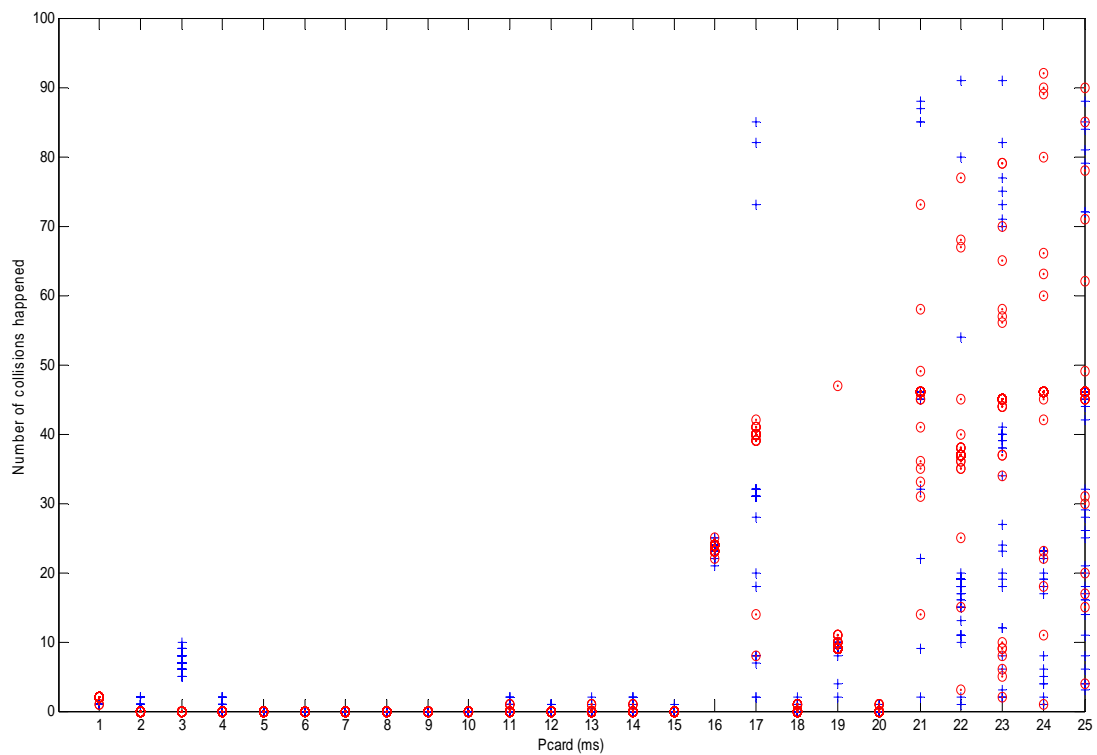


Figure 4.21 : Nombre de collision fait sans et avec l'utilisation de l'algorithme proposé

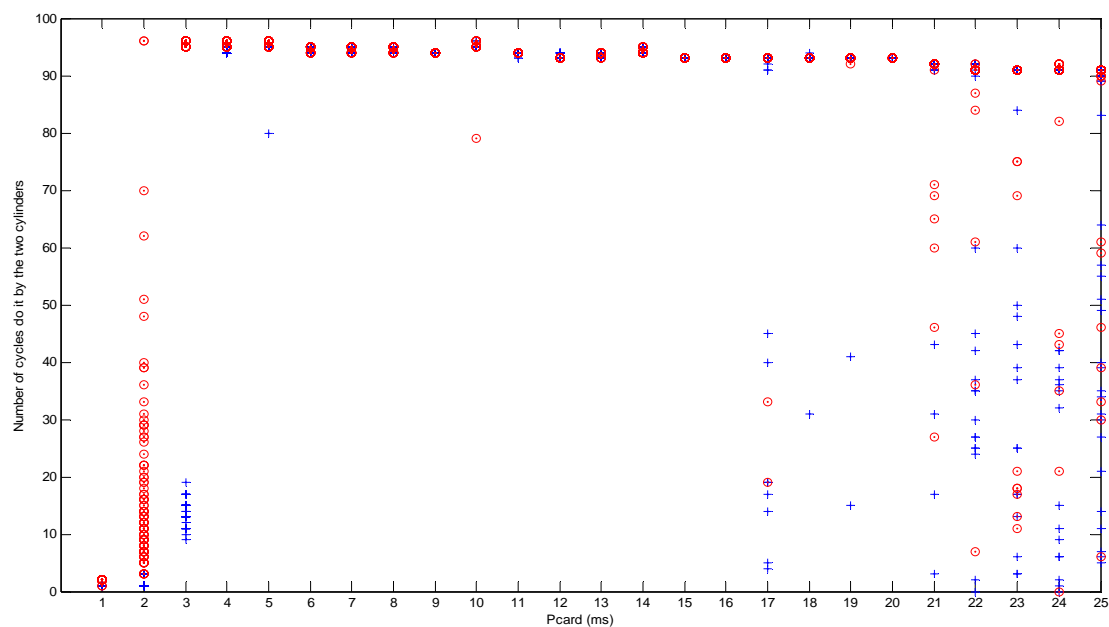


Figure 4.22 : Nombre de tours faits sans et avec l'utilisation de l'algorithme proposé

## 4.4 Conclusion du chapitre

La QoS du réseau peut dégrader la performance du SDCR sans fil. Cette dégradation est constatée surtout dans les réseaux chargés. Pour éviter ces dégradations, un algorithme de manipulation des priorités des flux est proposé. Selon les besoins de la commande et l'état du réseau, les priorités des flux sont dynamiquement modifiées en fonction de la pertinence des informations envoyées dans ces flux. Cet algorithme testé sur un cas d'étude en utilisant l'outil OPNET a montré son efficacité dans le cas d'un réseau chargé (dans notre cas par des périodes  $P_{carte}$  petites).

# Conclusion et perspectives

## I. Conclusion

Le travail dans ce rapport de thèse se focalise sur l'étude des systèmes discrets contrôlé en réseau sans fil. Ce type de système appartient à la famille des systèmes contrôlé en réseau avec une différence que l'espace d'état est un ensemble discret et l'état change seulement à certains instants du temps, d'une manière instantanée. De plus, le réseau intégré dans notre étude est un réseau sans fil (IEEE 802.11) au vu des avantages qu'il fournit comme le coût de câblage ... Par contre, ce type de réseau est plus sensible aux bruits d'où son comportement plus fragile par rapport à d'autres types de réseaux.

Un SDCR sans fil est formé de trois parties : la partie commande, formé des automates programmables (API); la partie réseau sans fil, 802.11 et la partie opérative formée des capteurs et des actionneurs dont le comportement continu des actionneurs est résumé par des états. Chacun de ces états décrit un comportement spécial de ces actionneurs. Notons aussi que chaque partie est caractérisée par des attributs gérables par l'utilisateur.

La performance d'un SDCR sans fil dépend des valeurs des attributs des différentes parties de ce système. Elle est évaluée en termes de temps de réponse, déterminisme et sûreté.

Les travaux existant qui essaient d'améliorer les performances d'un SCR, sont groupés en trois grandes approches : l'approche *orientée QdC*, qui consiste à adapter la commande relativement aux performances de communication, et l'approche *orientée QdS* dont le but est d'optimiser les performances du réseau afin de minimiser les effets du réseau (délai, taux de pertes,...) sur la commande. La dernière approche choisie dans cette étude, *orientée QdC-QdS*, consiste à analyser le couple Qualité de Service offert par le réseau et Qualité de Contrôle exigée par l'application. Cette approche nécessite un outil d'analyse permettant de modéliser globalement un SDCR sans fil. La complexité est que cet outil doit modéliser des parties d'un SDCR sans fil de natures différentes (déterministe/ non déterministe). Le choix s'oriente sur deux outils : MATLAB-Truetime, plutôt utilisé par la communauté automatique et OPNET, utilisé par la communauté réseau.

Nous avons apporté des modifications à la librairie Truetime pour qu'elle représente mieux la partie réseau. De même, les modèles dans OPNET sont étudiés dans le but de voir la possibilité d'implémenter la partie Commande/opérative dans cet outil. Des comparaisons entre ces deux outils sont faites au niveau réseau et sur un cas d'étude qui présente un SDCR sans fil. La convergence des résultats nous a assuré du bon comportement de ces outils, puis de l'exactitude des modifications faites sur MATLAB-Truetime, ainsi que de l'implémentation des différentes parties d'un SDCR sans fil sur OPNET.

Dans le but d'optimiser le choix des valeurs des attributs d'un SDCR sans fil, une étude analytique est faite pour calculer les temps de réponse dans les pires des cas. Ces calculs analytiques sont réalisés sur un cas d'étude, puis comparés à des résultats de simulations pour vérifier leur exactitude.

Dans les SDCR sans fil dont le réseau est chargé, l'optimisation de valeurs des attributs ne sera pas une solution suffisante pour avoir une performance acceptable du système. D'où notre proposition d'un algorithme qui alloue dynamiquement les priorités définies par le standard IEEE 802.11e sur les trafics de communication en fonction des besoins de la commande et du comportement du réseau. Cet algorithme est évalué sur un cas d'étude, pour lequel la communication sans fil induit des risques de collision entre des équipements d'une installation industrielle, en utilisant l'environnement de simulation précédemment défini. Les simulations montrent une amélioration de la performance du système, en particulier lorsque le système est chargé.

## II. Perspectives :

Les travaux réalisés durant cette thèse ouvrent la porte à un ensemble de perspectives. Nous envisageons deux plans de perspectives, à court et long terme:

A court terme :

Afin de confirmer l'efficacité de l'algorithme proposé, il est nécessaire de procéder à une expérimentation sur une plateforme réelle. Dans le laboratoire CRAN, nous disposons de plusieurs plateformes qui, constituées d'automates, PDA, capteurs sans fil, ... sont idéales pour tester notre algorithme. Cette validation expérimentale sera réalisée en complément d'une phase préalable d'émulation du réseau sous OPNET couplée avec la plateforme physique réelle. Plusieurs difficultés techniques telles que l'accès aux pilotes des cartes réseaux sans fil et points d'accès sont à surmonter. De plus, l'offre commerciale de produits IEEE 802.11e devra être soigneusement étudiée afin de retenir les matériels les plus ouverts permettant de mettre en œuvre nos propositions. Le temps estimé pour cette implémentation est d'environ quatre mois.

De plus, si nous avons enrichi *Truetime* en implémentant les mécanismes de priorités entre les flux conformément au standard IEEE 802.11e, nos modifications relatives à cette gestion des priorités n'ont pas été réellement testées dans la mesure où la simulation de l'algorithme a été réalisée avec OPNET. Une campagne de simulations doit donc être lancée dans ce but. Les résultats obtenus seront comparés à ceux produits sur les mêmes scénarios par OPNET, considéré comme référence.

Enfin, dans le but de diffuser rapidement la librairie *Truetime modifiée* à la communauté scientifique, des contacts ont été établis avec les auteurs de cette librairie. L'objectif est d'inclure ces modifications sur le site officiel de l'université de Lund pour téléchargement,

sachant que cette procédure requiert une validation préalable par l'université de Lund, des modifications que nous avons apportées.

A long terme :

Des études doivent être faites pour voir l'efficacité de l'algorithme dans le cas où le réseau présente des stations cachées. L'utilisation nécessaire dans ce cas du mécanisme RTS/CTS modifie considérablement les performances réseau, car elle introduit des délais supplémentaires, ce qui aura un impact évident sur la performance de ce système.

Dans l'algorithme proposé, la quantité d'information produite par chaque équipement est fixe et connue (par la valeur de *Pcarte* et la taille des données). Une autre perspective intéressante est d'intégrer à cet algorithme une possibilité de variation dynamique des émissions (variation de *Pcarte*) de chaque capteur. Les capteurs dits "critiques" pourraient aussi transmettre plus fréquemment pour augmenter la réactivité du système.

Dans cette thèse, nous avons essentiellement agi sur les attributs du réseau (priorités, *Pcarte*, ...) pour garantir la qualité de contrôle de la commande en fonction de son état courant ainsi que de la qualité de service du réseau. Il serait intéressant d'essayer d'agir également sur la partie commande, au delà de l'adaptation de *Pcarte* (gestion de modes, commutation de lois de commande, reconfiguration,...), afin d'améliorer encore les performances du système.



# Bibliographie

**Aad, I. et Castelluccia, C. 2001.** *Differentiation mechanisms for IEEE 802.11.* Anchorage (Alaska) : Proceedings. IEEE Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies., 2001.

**Aad, I., et al. 2004.** *Enhancing IEEE 802.11 MAC in congested environments.* Boston, Massachusetts, USA : IEEE ASWN, 2004. pp. 82-91.

**Addad, B. et Amari, S. 2008 a.** *Delay Evaluation and Compensation in Ethernet-Networked Control Systems.* s.l. : 16th International Conference on Real-Time and Network Systems, 2008 a.

—. **2008 b.** *Modeling and Response Time Evaluation of Ethernet-based control Architectures using Timed Event Graphs and Max-Plus Algebra.* USA : 4th annual IEEE Conference on Automation Science and Engineering, 2008 b.

**Alves, M., et al. 2002.** *Real-Time Communications over Hybrid Wired/Wireless PROFIBUS-based Networks.* s.l. : In Proc. 14th Euromicro Conference on Real-Time Systems, 2002. pp. 142 – 51.

**Anderson, M., et al. 2005.** *Simulation of Wireless networked control systems.* Spain : 44th IEEE conference on decision and control, and the european control conference, 2005.

**Ansel, P., Ni, Q. et Turletti, T. 2004.** *An Efficient Scheduling Scheme for IEEE 802.11e.* University of Cambridge, UK : Proc. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks Cambridge, 2004.

**Boggia, G., et al. 2009.** *A Simulation-based Performance Evaluation of Wireless Networked Control systems.* Mallorca : IEEE Emerging Technologies & Factory Automation, ETFA'09. , 2009.

**Bordbar, B. et Anane, R. 2005.** *An Architecture for Automated QoS Resolution in Wireless Systems.* Taiwan : 19th International Conference on Advanced Information Networking and Applications (AINA'05), 2005.

**Boughanmi, N., Y.-Q., Song et E., Rondeau. 2009.** *Wireless networked control system using ZigBee/IEEE 802.15.4.* Moscou : 13th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'2009, 2009.

**Brahimi, B. 2007.** *Proposition d'une approche intégrée basée sur les réseaux de Petri de Haut Niveau pour simuler et évaluer les systèmes contrôlés en réseau.* s.l. : CRAN, 2007.

**Branicky, M. S., Phillips, S. M. et Zhang, W. 2002.** *Scheduling and Feedback Co-Design for Networked Control Systems.* LAS VEGAS : IEEE CONF. ON DECISION AND CONTROL, 2002.

**Branicky, M., Liberatore, V. et Phillips, S. 2003.** *Networked control system co-simulation for co-design*. Denver, USA : in Proc. American Control Conference, 2003.

**Carlson, E., et al. 2005.** *A Performance Comparison of QoS Approaches for Ad Hoc Networks:802.11e versus Distributed Resource Allocation*. Nicosia, Cyprus : European Wireless, 2005.

**Caspi, P. 2003.** *Automatique continue, automatique discrète, informatique industrielle :le triangle des Bermudes ?* Metz : Colloque sur la modélisation des systèmes réactifs, MSR'03, 2003.

**Chan, H. et Özgüner, Ü. 1995.** *Closed-loop control of systems over a communication network with queues*. s.l. : International Journal of Control, 1995. pp. 493-510. Vol. 62. 0020-7179.

**Chatzimisios, P., et al. 2005.** *A simple and effective backoff scheme for the IEEE 802.11 MAC protocol*. Orlando, Florida : CITSA, 2005.

**Cheng, K-T et Krishnakumar, A.S. 1993.** *Automatic Functional Test Generation Using The Extended Finite State Machine Model*. s.l. : International Design Automation Conference (DAC), 1993. pp. 86–91.

**Choi, J., et al. 2005.** *EBA: an enhancement of the IEEE 802.11 DCF via distributed reservation*. s.l. : IEEE Transactions on Mobile Computing, 2005. pp. 378-390. Vol. 4. 1536-1233.

**Choi, S. et Shin, K. G. 2000.** *A unified wireless LAN architecture for real-time and non-real-time communication services*. s.l. : IEEE/ACM Transactions on Networking, 2000. pp. 44-50. Vol. 8.

**Choi, W.-Y. 2004.** *A Centralized MAC-Level Admission Control Algorithm for Traffic Stream Services in IEEE 802.11e Wireless LANs*. s.l. : AEÜ – International Journal of Electronics and Communications, 2004. pp. 305–309. Vol. 58.

**Chow, M. et Tipsuwan, Y. 2001.** *Network-based control systems: a tutorial*. Denver, CO, USA : The 27th Annual Conference of the IEEE Industrial Electronics Society, 2001. pp. 1593 - 1602. Vol. 3.

**Cisco. 2010.** *Voice over Wireless LAN 4.1 Design Guide. WLAN Quality of Service*. [En ligne] 2010.  
[http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/vowlan/41dg/vowlan\\_ch2.html](http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/vowlan/41dg/vowlan_ch2.html).

**Colandairaj, J., Irwin, G W et Scanlon, W G. 2006.** *An integrated approach to wireless feedback control*. Glasgow : CD Proc. UKACC International Control Conference, 2006. 0947649549.

**Colandairaj, J., Irwin, G. W. et Scanlon, W. G. 2005.** *Analysis of an ieee 802.11b wireless networked control system.* Prague, Czech Republic : 16th IFAC World Congress, 2005.

**Colandairaj, J., Scanlon, W. et Irwin, G. 2005.** *Under wireless networked control systems through simulation.* s.l. : IEEE computing&control engennering, 2005. Vol. 16.

**Cowling, J. et Selvakennedy, S. 2004.** *A Detailed Investigation of the IEEE 802.11e HCF Reference Scheduler for VBR Traffic.* Poland : International Conference on Computational Science, ICCS, 2004.

**Cruz, R. L. et Blanc, A. P. 2003.** *A Service Abstraction with Applications to Network calculus.* s.l. : in Proc. 41th Annual Allerton Conference on Communication, Control, and Computing, 2003. pp. 1154-1163.

**Denis, B., et al. 2007.** *Measuring the impact of vertical integration on response times in Ethernet fieldbuses.* Greece : 12th IEEE Conference on Emerging Technologies and Factory Automation, 2007.

**Ferreira, L., Alves, M. et Tovar, E. 2002.** *Hybrid Wired/Wireless PROFIBUS Networks Supported by Bridges/Routers.* Sweden : in Proc. 2002 IEEE Workshop on Factory Communication Systems, WFCS'2002, 2002.

**Fitzpatrick, J., Murphy, S. et Murphy, J. 2006.** *RTS/CTS Based Endpoint Admission Control for VoIP Over 802.11e.* Dublin, Ireland : 9th IFIP/IEEE International conference on management of multimedia and mobile networks and services, MMNS 2006, 2006.

**Georges, J.P., Divoux, T. et Rondeau, E. 2005.** *Confronting the performances of a switched Ethernet network with industrial constraints by using the network calculus.* s.l. : International Journal of Communication Systems, 2005. pp. 877 - 903. Vol. 18.

**Georges, J-P., Divoux, T. et Rondeau, E. 2007.** *Evaluation d majorants des délais de transmission pour les systèmes contrôlés en réseau.* s.l. : Traité IC2 Information-Commande-Communication, Hermès Science, Lavoisier, 2007. 978-2-7462-1513-9.

**Greifeneder, J. et Frey, G. 2006.** *Optimizing Quality of Control in Networked Automation Systems using Probabilistic Models.* Prague, Czech Republic : Emerging Technologies and Factory Automation, 2006.

**Grilo, A. et Nunes, M. 2002.** *PERFORMANCE EVALUATION OF IEEE 802.11E.* Portugal : Proceedings of the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2002. I.

**Grilo, A., MACEDO, M. et Nunes, M. 2003.** *A Scheduling Algorithm for QoS support in IEEE 802.11e Networks.* 3. s.l. : IEEE Wireless Communications, 2003. pp. 36- 43. Vol. 10.

**Habib, G., Divoux, T. et Pétin, J.F. 2009.** *Estimating Maximum and Minimum Delays for Wireless Discrete Networked Control Systems*. Prague : Wireless Telecommunication Symposium (WTS'09), 2009.

**Habib, G., et al. 2009.** *Evaluation de l'influence d'un réseau de communication sans fil sur la commande d'un SED*. Nantes : MSR, 2009.

**Habib, G., Pétin, J.F. et Divoux, T. 2010.** *Control-based algorithm for the management of IEEE 802.11e priorities within a Wireless Networked Discrete Control System*. Nice : s.n., 2010.

**Harding, C., Griffiths, A. et Yu., H. 2007.** *An Interface between MATLAB and OPNET to Allow Simulation of WNCs with MANETs*. London, UK : Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control, 2007.

**Harel, D. 1987.** *Statecharts: a visual formalism for complex systems*. s.l. : Science of computer programming, 1987. 8.

**Hasan, M S, et al. 2007.** *Simulation of Distributed Wireless Networked Control Systems over MANET using OPNET*. UK : Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control, 2007.

**Henriksson, D., Cervin, A. et Årzén, K.-E. 2003.** *TRUETIME: Real-time Control System Simulation with MATLAB/Simulink*. Denmark : In Proceedings of the Nordic MATLAB Conference, 2003.

**Hongbo, L., et al. 2008.** *Intelligent Scheduling Controller Design for Networked Control Systems Based on Estimation of Distribution Algorithm*. s.l. : Tsinghua Science & Technology, 2008. pp. 71-77 . Vol. 13.

**Hu, S. et Yan, W.-Y. 2007.** *Stability robustness of networked control systems with respect to packet loss*. s.l. : Automatica, 2007. pp. 1243-1248. Vol. 43.

**IEEE 802.11. 1999.** *IEEE Standards for Information Technology Specific Requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Edition (ISO/IEC 8802-11: 1999)*. 1999.

**IEEE 802.11a. 1999.** *IEEE Standard for Information technology—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications— Amendment 1: High-speed Physical Layer in the 5 GHz band*. 1999.

**IEEE 802.11b. 1999.** *Wireless LAN MAC and PHY specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band*. 1999.

**IEEE 802.11e. 2005.** *Draft Supplement to STANDARD FOR Telecommunications and Information Exchange Between Systems-LAN/MAN Specific Requirements - Part 11: Wireless MAC and Physical Layer specifications: Medium Access Control Enhancements for QoS, IEEE 802.11e/Draft 4.2*. 2005.

**IEEE 802.11g. 2003.** *IEEE Standard for Information technology—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications—Amendment 4: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band.* 2003.

**Inan, I., Keceli, F. et Ayanoglu, E. 2006.** *An adaptive multimedia QoS Scheduler for 802.11e wireless LANs.* Istanbul, Turkey : In Proceedings of the 2006 IEEE International Conference on Communications, 2006. pp. 5263-5270.

**Jasperneite, E. et Neumann, P. 2001.** *Switched Ethernet for factory communication.* France : In Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation, 2001.

**Jasperneite, J., et al. 2002.** *Deterministic real-time communication with switched Ethernet.* 4th IEEE International Workshop on Factory Communication Systems(WFCS'02). Sweden : In Proceedings of 4 th IEEE International Workshop on Factory Communication Systems, 2002.

**Jiang, W., et al. 2009.** *A remote observer and controller with adaptation to the network Quality of Service.* Budapest, Hungary : 10th European Control Conf., EUCA-IFAC-IEEE, 2009.

**Juanole, G. 2002.** *Quality of service of communication networks and distributed automation: models and performances.* Barcelona, Spain : 15th Triennial World Congress, 2002.

**Karanam, S.P., Trsek, H. et Jasperneite, J. 2006.** *Potential of the HCCA scheme defined in IEEE802.11e for QoS enabled Industrial Wireless Networks.* s.l. : IEEE International Workshop on Factory Communication Systems, 2006.

**Kim, D.-S., Choi, D.-H. et Mohapatra, P. 2009.** *Real-time scheduling method for networked discrete control systems.* s.l. : Control Engineering Practice, 2009. pp. 564-570. Vol. 17.

**Krommenacker, N. et Lecuire, V. 2005.** *Building industrial communication systems based on iee 802.11g wireless technology.* Catania, Italy : ETFA'05: 10th IEEE Conference on Emerging Technologies and Factory Automation, 2005.

**Krommenaker, N. 2002.** *Heuristiques de conception de topologies réseaux application aux réseaux locaux industriels.* s.l. : PhD thesis. Centre de Recherche en Automatique de Nancy (CRAN), Université Henri Poincaré, 2002.

**Kubler, S., Rondeau, E. et Georges, J.-P. 2010.** *Continuité de service sur Ethernet Industriel.* Nancy : Sixième Conférence Internationale Francophone d'Automatique, 2010.

**Kwon, Y., Fang, Y. et Latchman, H. 2003.** *A novel mac protocol with fast collision resolution for wireless lans.* San Francisco, USA : Infocom, 2003.

**Kyung Chang, L. et Suk, L. 2002.** *Performance evaluation of switched ethernet for real-time industrial communications*. s.l. : Computer standards & interfaces, 2002. Vol. 24.

**Lee, H.-J. et Kim, J.-H. 2006.** *A optimal CF-poll piggyback scheme in IEEE 802.11e HCCA*. Suwon : The 8th International Conference Advanced Communication Technology, ICACT, 2006.

**Lessard, A. et Gerla, M. 1988.** *Wireless communication in the automated factory environment*. s.l. : IEEE Network Magazine, 1988. pp. 64-69. Vol. 2.

**Li, J., Li, Z. et Mohapatra, P. 2006.** *APHD: End-to-End Delay Assurance in 802.11e Based MANETs*. San Jose, California, Etats Units : 3rd Annual International Conference on Mobile and Ubiquitous Systems, 2006.

**Lian, F.-L., Moyne, J. et Tilbury, D. 2002.** *Network Design Consideration for Distributed Control Systems*. s.l. : IEEE Transactions on Control Systems Technology, 2002. pp. 297 - 307. Vol. 10.

**Lin, W.-Y. et Wu, J.-S. 2007.** *Modified EDCF to improve the performance of IEEE 802.11e WLAN*. s.l. : Computer Communications, 2007. pp. 841-848. Vol. 30.

**Liu, X. et Goldsmith, A. 2004.** *Wireless Network Design for Distributed Control*. BAHAMAS : 43rd IEEE Conference on Decision and Control, 2004. 0-7803-8682-5.

**Lu-ming, C., et al. 2007.** *Performance evaluation on IEEE 802.11e considering emergency calls in congested situation*. s.l. : THE JOURNAL OF CHINA UNIVERSITIES OF POSTS AND TELECOMMUNICATIONS, 2007. Vol. 14.

**Mangold, Stefan, et al. 2002.** *IEEE 802.11e Wireless LAN for Quality of Service*. Florence,italy : European Wireless, 2002.

**Mao, Z.-H. et Jiang, B. 2007.** *Fault Estimation and Accommodation for Networked Control Systems with Transfer Delay*. s.l. : ACTA AUTOMATICA SINICA, 2007. pp. 738-743. Vol. 33.

**Marangé, P. 2008.** *Synthèse et filtrage robuste de la commande pour des systèmes manufacturiers sûrs de fonctionnement*. PhD thesis, Université de Reims Champagne-Ardenne. 2008.

**Marsal, G., Denis, B. et Faure, J.F. 2006.** *Évaluation des délais de réactivité des architectures de commande distribuées sur réseau Ethernet*. Bordeaux, France : Conférence Internationale Francophone d'Automatique, CIFA '06, 2006.

**Masri, A., Bourdeaud'hui, T. et Toguyeni, A. 2009.** *A Component-Based Approach Based on High-Level Petri Nets for Modeling Distributed Control Systems*. s.l. : International Journal on Advances in Intelligent Systems, 2009. pp. 335-353. Vol. 2.

**Meunier, P., Denis, B. et Lesage, J.-J. 2007.** *Temporal performance evaluation of control architecture in automation systems*. slovenia : Eurosim, 2007.

**Michaut, F. 2003.** *Adaptation des applications distribuées à la Qualité de Service fournie par le réseau de communication.* Nancy : CRAN, 2003.

**Miorandi, D. et Vitturi, S. 2004.** *Performance analysis of Producer/Consumer protocols over IEEE802.11 wireless protocols.* s.l. : WFCS, 2004.

**Mühlenbein, H. et Paaß, G. 1996.** *From Recombination of Genes to the Estimation of Distributions I. Binary Parameters.* s.l. : Lecture Notes In Computer Science, 1996. pp. 178-187. Vol. 1141. 978-3-540-61723-5.

**Neumann, P. 2007.** *Communication in industrial automation—What is going on?* s.l. : Control engineering practice, 2007. pp. 1332–1347.

**Ni, Q. 2005.** *Performance Analysis and Enhancements for IEEE 802.11e Wireless Networks.* ETATS-UNIS : IEEE Network, 2005. pp. 21-27. Vol. 19.

**Nilsson, J. 1998.** *Real-time control systems with delays.* s.l. : Ph.D. dissertation, Dept. automatic control, Lund Institute of Technology, Lund, Sweden, 1998.

**Nilsson, J., Bernhardsson, B. et Wittenmark, B. 1998.** *Stochastic analysis and control of real-time systems with random time delays.* s.l. : Automatica, 1998. Vol. 34.

**OPNET v14.5.** OPNET. [En ligne] <http://www.opnet.com/>.

**Passas, N., Skyrianoglou, D. et Mouziouras, P. 2006.** *Prioritized support of different traffic classes in IEEE 802.11e wireless LANs.* s.l. : Computer communications, 2006. pp. 2867–2880. Vol. 29.

**Pattara-Atikom, W., Krishnamurthy, P. et Banerjee, S. 2003.** *Comparison of distributed fair QoS mechanisms in wireless LANs.* San Francisco, USA : Globecom, 2003. pp. 553-557.

**Pollin, S. Motamedi, et al. 2005.** *Delay improvement of IEEE 802.11 distributed coordination function using size-based scheduling.* Seoul, Korea : ICC, 2005.

**Pong, D. et Moors, T. 2003.** *Call Admission Control for IEEE 802.11 Contention Access Mechanism.* San francisco, USA : GLOBECOM, 2003.

**Romdhani, L., Ni, Q. et Turletti, T. 2003.** *Adaptive EDCAF: Enhanced Service Differentiation for IEEE 802.11 Wireless Ad-Hoc Networks.* New Orleans, LA, Etats Units : WCNC, 2003.

**Rondeau, E., Divoux, T. et Adoud, H. 2001.** *Study and method of Ethernet architecture networks with different topologies used in automation systems.* Nancy, France : 3rd IFAC International Conference on Fieldbus Systems and their Applications (FET'2001), 2001.

**Ruel, S., De Smet, O. et Faure, J.-M. 2008.** *Building effective formal models to prove time properties of networked automation systems.* Suède : 9th International Workshop On Discrete Event Systems, WODES'08, 2008.

**Salles, N. et Krommenacker, N. 2007.** *Analyse de performances de la période sans contention de IEEE 802.15.4 pour des applications industrielles temps réel.* Nantes, France : 5ème École d'été Temps Réel, 2007.

**Shahidul Hasan, M., et al. 2005.** *Modeling Delay and Packet Drop in Networked Control Systems Using Network Simulator NS2.* s.l. : international journal of automation and computing, 2005. pp. 187-194. Vol. 2.

**Shih, K., et al. 2009.** *On avoiding RTS collisions for IEEE 802.11-based wireless ad hoc networks.* s.l. : Computer Communications, 2009. pp. 69-77. Vol. 32. 0140-3664.

**Soglo, A.B. et YANG, X. 2006.** *Networked control system Simulation Design and its application.* s.l. : Tsinghua science and technology, 2006. pp. 287-294 . Vol. 11.

**Tipsuwan, Y., Kamonsantiroj, S. et Chongstitvattan, P. 2009.** *An auction-based dynamic bandwidth allocation with sensitivity in a wireless networked control system.* s.l. : Computers & Industrial Engineering 57, 2009. pp. 114-124. Vol. 57. 0360-8352.

**Varposhti, M. et Movahhedinia, N. 2009.** *Supporting QoS in IEEE 802.11e wireless LANs over fading channel.* s.l. : Computer Communications, 2009. pp. 985-991. Vol. 32.

**Villalón, J., et al. 2008.** *B-EDCA: A QoS mechanism for multimedia communications over heterogeneous 802.11/802.11e WLANs.* s.l. : Computer Communications, 2008. pp. 3905-3921. Vol. 31.

**Wall, Joshua et Khan, Jamil Y. 2003.** *An Advanced ARQ Mechanism for the 802.11 MAC Protocol.* Melbourne : in Proceedings of Australian Telecommunications, Networks and Applications Conference(ATNAC), 2003.

**Witsch, D., et al. 2006.** *Performance analysis of industrial Ethernet networks by means of timed model-checking.* St-etienne : 12th IFAC Symposium on Information Control Problems in Manufacturing, INCOM 2006, 2006.

**Xi, Weihua Helen, Munro, Alistair et Barton, Michael. 2008.** *Architecture of achieving QoS for multiple flows per node in Wlans.* Enschede : IWQoS, 2008.

**Xiangheng, L. et Goldsmith, A. 2004.** *Wireless medium access control in networked control systems.* Boston MA , Etats-unis : Proceedings of the 2004 American Control Conference ACC, 2004.

**Xiao, Y. et LI, H. 2004.** *Evaluation of Distributed Admission Control for the IEEE 802.11e EDCA.* s.l. : IEEE Commun. Magazin, 2004. pp. S20–S24. Vol. 42.

**Yang, L. 2004.** *P-HCCA New Scheme for Real-time Traffic with QoS in IEEE 802.11e Based Networks.* s.l. : APAN Network Research Workshop, 2004.

**Yang, L., et al. 2009.** *Analysis and Design of Wireless Networked Control System Utilizing Adaptive Coded Modulation.* s.l. : Acta Automatica Sinica, 2009. pp. 911-918. Vol. 35.

**Zhang, L. et Hua-Jing, F. 2006.** *A novel controller design and evaluation for networked control systems with time-variant delays.* s.l. : journal of the Frankil institute, 2006. Vol. 343.

**Zhao, J., et al. 2003.** *Distributed MAC adaptation for WLAN QoS differentiation.* San Francisco, USA : GLOBECOM '03. IEEE Global Telecommunications Conference, 2003., 2003. pp. 3442 - 3446. Vol. 6.

**Zhu, R., Wang, J. et Ma, M. 2008.** *Intelligent MAC model for traffic scheduling in IEEE 802.11e wireless LANs.* s.l. : Applied Mathematics and Computation, 2008. pp. 109-122. Vol. 205.