Influence des fautes transitoires sur la fiabilité d'un système commandé en réseau

Rony Ghostine

Institut National
Polytechnique de Lorraine

Ecole doctorale IAEM Lorraine
Département de Formation Doctorale en Automatique

Influence des fautes transitoires sur la fiabilité d'un système commandé en réseau

THÈSE

Présentée et soutenue publiquement le 12 Juin 2008

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

Spécialité Automatique, Traitement du signal et Génie Informatique

Par

RONY GHOSTINE

Composition du jury

Président :		
Rapporteurs :	J-M Faure	Professeur à l'Institut Supérieur de Mécanique de Paris
	L. Cauffriez	Maître de Conférence HDR Université de Valencienne et du Hainaut-Cambresis
Examinateurs :	J-F. Aubry	Professeur à l'Institut National Polytechnique de Lorraine
	J-M. Thiriet	Professeur à l'Université Joseph Fourier de Grenoble
	E. Rondeau	Professeur à l'Université Henri Poincaré Nancy 1
	P. Barger	Maître de Conférence à l'Université de Technologie de Compiègne

Centre de Recherche en Automatique de Nancy
UMR 7039 – Nancy Université – CNRS
2, Avenue de la Forêt de Haye
54516 Vandoeuvre-Lès-Nancy
Tél. +33 (0) 83 59 59 59 Fax +33 (0) 3 83 59 56 44

A mes parents vous m'avez
mis sur le bon chemin.
A mes deux frères Rabih et Joe
Je vous aime à vous tous !

Remerciements

Les travaux présentés ont été réalisés au sein de l'équipe SACSS (Systèmes Automatisés Contraints par la sûreté de fonctionnement et la Sécurité) du groupe thématique SURFDIAG (Sûreté de fonctionnement et diagnostic des systèmes) du Centre de Recherche en Automatique de Nancy (CRAN).

Les travaux ont été dirigés par messieurs Jean-Marc Thiriet et Jean-François Aubry, à qui je tiens à exprimer ma plus grande et profonde gratitude pour leur soutien, leurs conseils et pour leur disponibilité. Merci encore pour tout l'intérêt que vous avez porté à mes travaux.

Je remercie Monsieur Jean-Marc Faure et Monsieur Laurent Cauffriez de m'avoir fait l'honneur d'être rapporteurs de ces travaux de thèse. Je remercie aussi Messieurs Eric Rondeau et Pavol Barger d'avoir accepté d'être examinateurs de ces travaux.

Je remercie également tous les membres de l'équipe SACSS, je suis reconnaissant pour l'intérêt que vous avez porté à mes travaux, pour toutes les réunions et les discussions enrichissantes que nous avons eues.

J'adresse également tous mes remerciements à l'ensemble des personnels du CRAN et de l'ESSTIN (Ecole Supérieure des Sciences et Technologies de l'Ingénieur de Nancy), personnes et amis que j'ai pu côtoyer durant ces années de thèse pour leur soutien et leur bonne humeur. Je pense notamment à Ramy, Belynda, Alex, William, Dominique, Maxim, Abdel, Philippe, Laurentiu, Jan, Abdelhak, Sallak, Gabriel, Sylvie, Florianne…

Mes remerciement vont aussi à Aurore 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 ne saurais terminer cette page, sans adresser mes remerciements à tous les amis qui m'ont encouragé et assisté de leur mieux : ce sont Sami Sassine, Elie Chahine, Karim Yazback, Alex Muler, Hadi Sader, Nicolas Haddad, Jean Atmé, Rabih Sader, Jad Sader, Mark Barakat, Richard Barakat, Fadi Eldeek, Ziad Eldeek, Adib Haddad, Fadi Afif, Dany Wazen, Jan Galdun, Said Eid, Rana Eid, hala Hannoun, Hiba Hannoun, Jessica Eid.

Je tiens à remercier tous ceux qui, de près ou de loin, ont participé à la réalisation finale de ce manuscrit.

Ce doctorat doit, aussi, beaucoup 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, Rabih, Joe, et Ramy qui m'ont toujours soutenu et encouragé dans les moments les plus difficiles.

« Nul ne peut atteindre l'aube sans
passer par le chemin de la nuit »
Jubran Khalil Jubran

Sommaire

Chapitre 1 Systèmes commandés en réseau 15
1.1 Introduction 15
1.2 Systèmes commandés en réseau (SCR) 15
1.2.1 Structure Directe 16
1.2.2 Structure hiérarchique 17
1.3 Problématique des SCR 18
1.3.1 Commande du réseau (*control of network*) 19
1.3.2 Commande via le réseau (*control over network*) 21
1.3.2.1 Modélisation du retard pour un système à temps continu 21
1.3.2.2 Modélisation du retard pour un système à temps discret 21
1.3.2.3 Méthodes pour accommoder les retards induits par le réseau 22
1.3.3 Approche intégrée co-design 23
1.4 Protocoles pour les systèmes distribués 23
1.4.1 Accès au medium 24
1.4.1.1 Accès unique 24
1.4.1.2 Accès multiple 24
1.4.2 Contrôle de la communication sur le bus 26
1.4.2.1 Comparaison 26
1.5 Qualité de service 27
1.5.1 Paramètres de la qualité de service 27
1.5.1.1 Délai 27
1.5.1.2 Gigue 28
1.5.1.3 Taux de pertes 28
1.5.2 Qualités de performances 29
1.5.2.1 Temps moyen de retard (*Total Average Delay*) 29
1.5.2.2 Efficacité du réseau (*Efficiency of Networks*) 29
1.5.2.3 Utilisation du réseau 30
1.5.2.4 Stabilité du réseau 30
1.6 Influence du réseau sur les performances d'un SCR 30
1.6.1 Période d'échantillonnage 31
1.6.2 Influence du retard sur un SCR 32
1.6.2.1 Influence d'un retard constant 32
1.6.2.2 Influence d'un retard variable 33
1.6.3 L'influence des pertes sur les performances d'un SCR 33
1.7 Conclusion 33
Chapitre 2 Sûreté de fonctionnement des systèmes commandés 35
2.1 Introduction 35
2.2 Méthodes pour la sûreté de fonctionnement 36
2.2.1 Méthodes pour les systèmes statiques 36
2.2.1.1 Diagramme de succès 36
2.2.1.2 Arbre de défaillance 37
2.2.1.3 Analyse des modes de défaillance, de leurs effets et de leurs criticités (AMDEC) 38
2.2.1.4 Arbre de conséquence 38
2.2.2 Méthodes pour les systèmes réparables 39
2.2.2.1 Processus Markoviens 39
2.2.3 Méthodes de sûreté de fonctionnement dynamiques 40
2.2.3.1 Réseau de Petri 40

2.2.3.2 Automate à états 41
2.2.3.3 Arbre de défaillance dynamique 42
2.2.3.4 Réseaux bayésiens RB 42
2.2.3.5 Evaluation des paramètres de la SdF par simulation 43
2.3 Sûreté de fonctionnement des systèmes automatiques 45
2.3.1 Types de défaillances 45
2.3.1.1 Défaillance au niveau capteur 45
2.3.1.2 Défaillance au niveau actionneur 46
2.3.1.3 Défaillance au niveau régulateur 47
2.3.2 Aspect dynamique 47
2.3.3 Travaux concernant la fiabilité des systèmes commandés 48
2.4 Intégration du réseau dans l'étude de la sûreté de fonctionnement 52
2.4.1 Défaillance de la fonction communication 52
2.4.1.1 Types de fautes 53
2.4.2 Influence des défaillances réseau sur la sûreté de fonctionnement d'un système 58
2.5 Choix de l'outil de modélisation 60
2.6 Introduction à la modélisation Möbius/tool 63
2.6.1 Modèle atomique 63
2.6.1.1 Stochastic Activity Network 65
2.6.2 Modèle composé 69
2.6.3 Fonction de récompense 70
2.6.4 Study 70
2.6.5 Solveur 71
2.7 Conclusion 71
Chapitre 3 Modélisation et simulation des SCR en vue de l'évaluation de la SdF 73
3.1 Introduction 73
3.2 Modélisation du fonctionnement et du dysfonctionnement du réseau CAN 74
3.2.1 Modélisation fonctionnelle du Protocole CAN 75
3.2.1.1 Validation croisée du fonctionnement du réseau CAN 77
3.2.2 Modélisation dysfonctionnelle 78
3.3 Modélisation des composants classiques d'un système bouclé 82
3.3.1 Capteur 83
3.3.2 Régulateur 85
3.3.3 Actionneur 87
3.3.4 Processus 88
3.4 Modèle composé 92
3.5 Exemple de simulation 93
Influence du retard fixe : 93
3.6 Conclusion 94
Chapitre 4 Influence des fautes transitoires sur la fiabilité des SCR 97
4.1 Introduction 97
4.2 Influences des fautes 97
4.2.1 Critères de performances 98
4.2.2 Influence du retard variable 99
4.2.3 Influence des pertes des messages 102
4.2.4 Superposition de retards variables et pertes des messages 104
4.3 Vers une étude de la fiabilité 105
4.3.1 Critères d'évaluation 106
4.3.2 Défaillance par dépassement 106

4.3.2.1 Influence du retard variable sur la défaillance par dépassement 107
4.3.2.2 Influence des pertes sur la défaillance par dépassement 108
4.3.2.3 Influence de la superposition de retards variables et de la perte de messages sur la défaillance par dépassement 109
4.3.3 Défaillance par temps de réponse 110
4.3.3.1 Influence du retard variable sur la défaillance par temps de réponse 110
4.3.3.2 Influence des pertes sur la défaillance par temps de réponse 111
4.3.3.3 Influence de la superposition du retard variable et des pertes des messages sur la défaillance par temps de réponse 112
4.3.4 Défaillance par stabilité 113
4.3.5 Evaluation de la fiabilité 114
4.4 Intégration effective du réseau 116
4.4.1 Influence du trafic externe 119
4.4.1.1 Trafic périodique 119
4.4.1.2 Trois boucles partageant le même medium 122
4.5 Vers un logiciel pour l'étude de la sûreté de fonctionnement des systèmes commandés en réseau 124
4.6 Conclusion 126
Conclusion générale 127
ANNEXE A : Möbius tool et CPN tools 143
ANNEXE B : Trois boucles partageant le même medium de communication 151

Introduction

Ce travail s'inscrit dans le cadre de l'évaluation de la sûreté de fonctionnement (Sdf) des systèmes commandés en réseau (*Networked Control System*). La problématique se trouve à l'intersection de plusieurs axes de recherche : les problèmes de modélisation et d'évaluation, notamment l'évaluation de la sûreté de fonctionnement, et le domaine des réseaux de communication.

L'étude de la Sdf des systèmes commandés en réseau n'a longtemps été que partielle. On a pu distinguer deux types d'approches : les approches orientées "automatique" où le réseau de communication est totalement ignoré [Moncelet et al, 1998] et les approches centrées "réseau" où la performance du réseau est utilisée sans prise en compte de l'application [Portugal et Carvalho, 2004] [Navet et al, 2000]. Dans les approches orientées automatique, les fautes sont de nature permanente, et la défaillance du système est définie par rapport à la défaillance des composants et de leurs logiques de fonctionnement. Dans les approches réseau, la défaillance est définie comme le non respect du délai de délivrance d'un message.

L'utilisation de réseaux et de composants électroniques numériques à la place de certaines parties mécaniques (ex : *X-by-Wire* dans l'automobile) ou électroniques analogiques rend le système plus sensible aux perturbations et exige des protections particulières. En effet, on imagine mal une colonne de direction déformée au passage de la proximité d'un radar puissant. Cette réalité entraîne l'importance des études de sûreté de fonctionnement de ces systèmes. Dans ce contexte, nous avons fait le choix de concentrer notre analyse sur la quantification de l'influence de fautes transitoires sur la fiabilité du système.

Le réseau est par nature très étendu dans l'espace, ce qui fait de lui un élément particulièrement sensible aux perturbations responsables des fautes transitoires. Nous proposons donc une approche pour l'évaluation de la sûreté de fonctionnement (SdF) d'un système commandé en réseau, vis-à-vis de ces fautes transitoires. Le but est plus précisément de trouver une relation qui lie la fiabilité du système entier aux probabilités des retards ou des pertes des messages dans le réseau.

Une difficulté de ce type d'étude est que les fautes affectant le réseau n'ont pas toujours le même effet ; par exemple dans une boucle fermée distribuée autour d'un réseau la perte des messages dans la phase transitoire n'a pas le même effet qu'en régime permanent. Nous nous trouvons dans une situation de « fiabilité dynamique » puisque la relation entre la défaillance du système et la défaillance des composants change au cours du temps.

Dans les études orientées automatique, on trouve des modélisations du comportement du système considérant la présence d'un retard constant dans la boucle (étude de robustesse, limite de stabilité…). Dans la réalité industrielle, les perturbations du réseau sont aléatoires, les éléments connectés au réseau fonctionnent de manière asynchrone et plusieurs échanges de messages sont nécessaires dans une même période d'échantillonnage. Il s'ensuit que dans le cas général une étude analytique est difficilement réalisable.

La capacité des systèmes de commande à compenser les effets de certaines défaillances de composants (comme le réseau) amène à redéfinir le concept de défaillances du système. La conséquence est que l'évaluation de la fiabilité prévisionnelle du système est dépendante de l'évaluation fonctionnelle et devient très difficile voire impossible avec les méthodes traditionnelles de la sûreté de fonctionnement. Pour surmonter ces difficultés, une approche basée sur la modélisation en vue de la simulation est proposée.

La modélisation est basée sur les SAN (*Stochastic Activity Network*) qui sont une variante des réseaux de Petri. Les SAN conservent toutes les capacités de modélisation des réseaux de Petri stochastiques par le biais des activités stochastiques et les avantages des réseaux de Petri colorés par les structures associées à chaque place. De plus les entrées et les sorties dans SAN permettent une facilité de modélisation et un accès aux marquages de toutes les places à chaque instant. Tous ces éléments font des SAN un outil efficace et adéquat pour notre besoin de modélisation dans le but d'avoir un support de simulation. Par contre si le but de la modélisation est la vérification des propriétés algébriques, l'utilisation des SAN exige quelques restrictions.

L'approche de Monte-Carlo est utilisée pour l'évaluation statistique des paramètres de sûreté de fonctionnement grâce à de nombreuses simulations (ou histoires).

Le **premier chapitre** situe le contexte de la thèse et introduit les systèmes commandés en réseau: leurs spécificités, leurs caractéristiques, l'avantage de l'utilisation du réseau et les différents types d'accès au medium. Le chapitre 1 se termine par une discussion sur l'influence du retard et plus généralement de la qualité de service du réseau sur les performances du système.

Le **deuxième chapitre** est consacré à l'étude de la sûreté de fonctionnement, notamment la fiabilité des systèmes commandés. La problématique est présentée en regard des différents travaux menés auparavant. Les difficultés inhérentes à l'utilisation de systèmes commandés en réseau vis-à-vis de l'évaluation de la sûreté de fonctionnement sont mises en évidence et le besoin d'une approche de simulation est justifié. La dernière partie de ce chapitre présente les SAN qui sont le formalisme choisi pour la modélisation. Les SAN ont été choisis pour leur capacité à modéliser un système complexe dans lequel cohabitent des variables discrètes et des événements, pouvant être aléatoires ou déterministes.

Le **troisième chapitre** propose un environnement pour la modélisation et la simulation des systèmes commandés en réseau (SCR). La modélisation de tous les composants y compris le réseau est faite de manière hiérarchique, en partant des modèles de base qui représentent chacun un composant du système. La jonction des modèles de base forme le modèle global représentant le système. Nos modèles sont paramétrés de façon à faciliter le changement des paramètres. Cela permet de proposer des structures génériques de modèles [Ghostine et al, 2007a]. Le réseau utilisé comme support pour notre étude est CAN (*Controller Area network*). Le but de ces modèles est de former un support de simulation pour l'évaluation de la sûreté de fonctionnement en présence des fautes transitoires.

Le **quatrième chapitre** est dédié à l'étude de l'influence des fautes transitoires du réseau sur la fiabilité des systèmes commandés en réseau. Dans un premier temps, nous cherchons à identifier l'incidence de deux types de défaillances fugitives : la perte d'un échantillon d'une part et le retard d'un échantillon dans la boucle de régulation (sans considérer la présence d'un réseau) d'autre part. L'influence de la superposition des deux types de perturbations est ensuite étudiée, mettant en évidence les effets cumulatifs. Cela peut être rapproché du concept de coupe. En effet les défaillances injectées indépendamment peuvent avoir un effet acceptable par contre simultanément il est possible qu'elles produisent un effet inacceptable. En pratique l'origine des pertes ou retards est due à la présence du réseau, il faut donc l'introduire dans le modèle. Le caractère aléatoire n'est plus associé directement à la défaillance mais plutôt à la cause qui est susceptible de la produire. Un réseau est normalement partagé par plusieurs applications (plusieurs boucles dans les SCR), la dernière partie du chapitre met en évidence l'influence du partage du réseau sur la fiabilité.

Ces travaux s'inscrivent dans la démarche de construction d'un outil d'aide à la conception des systèmes sûrs de fonctionnement. Cette approche peut servir pour valider les systèmes où le réseau est un élément central.

Chapitre 1 Systèmes commandés en réseau

1.1 Introduction

Une tendance importante dans les systèmes industriels et commerciaux modernes est d'intégrer les fonctions de communication, les fonctions de calcul et celles de contrôle aux différents niveaux du système. La connexion traditionnelle point à point, qui lie le contrôleur central aux différents composants a été mise en application avec succès dans l'industrie pendant des décennies. Cependant, les installations physiques et l'extension des fonctions ont poussé les limites de l'architecture point à point. Par conséquent, une relation point à point traditionnelle n'est plus appropriée pour répondre à de nouvelles exigences, telles que la modularité, la décentralisation de la commande et l'intégration du diagnostic. Les bus de terrain sont utilisés pour répondre à ces nouvelles exigences [Thomesse, 1998].

1.2 Systèmes commandés en réseau (SCR)

Les systèmes commandés en réseau (*Networked Control System*) [Walsh et al, 1999] sont des systèmes automatiques traditionnels où les capteurs, contrôleurs, actionneurs et autres composants sont distribués autour d'un medium de communication (Figure 1.1). Les capteurs, contrôleurs et actionneurs constituent avec les systèmes de communication les éléments d'un SCR.
Le medium de communication sert à assurer les fonctions suivantes :

- établir un lien de communication pour envoyer l'information entre chaque producteur et les consommateurs concernés,
- respecter les contraintes de temps réel imposées par l'application,
- faciliter la maintenance (détection des fautes, simplification du câblage),
- participer à la reconfiguration (réorganisation des échanges suite à la défaillance d'un composant ou suite à l'adjonction d'un nouveau composant).

Le réseau de terrain répond à ces exigences, de plus son utilisation offre la possibilité d'implémenter des lois de commande sophistiquées [Lee, 2001]. Les avantages des SCR sont la réduction des coûts de câblage [Gallara, 1984] l'aide au diagnostic et à la maintenance [Iung, 2002] et l'amélioration de la flexibilité dans la conception des systèmes.

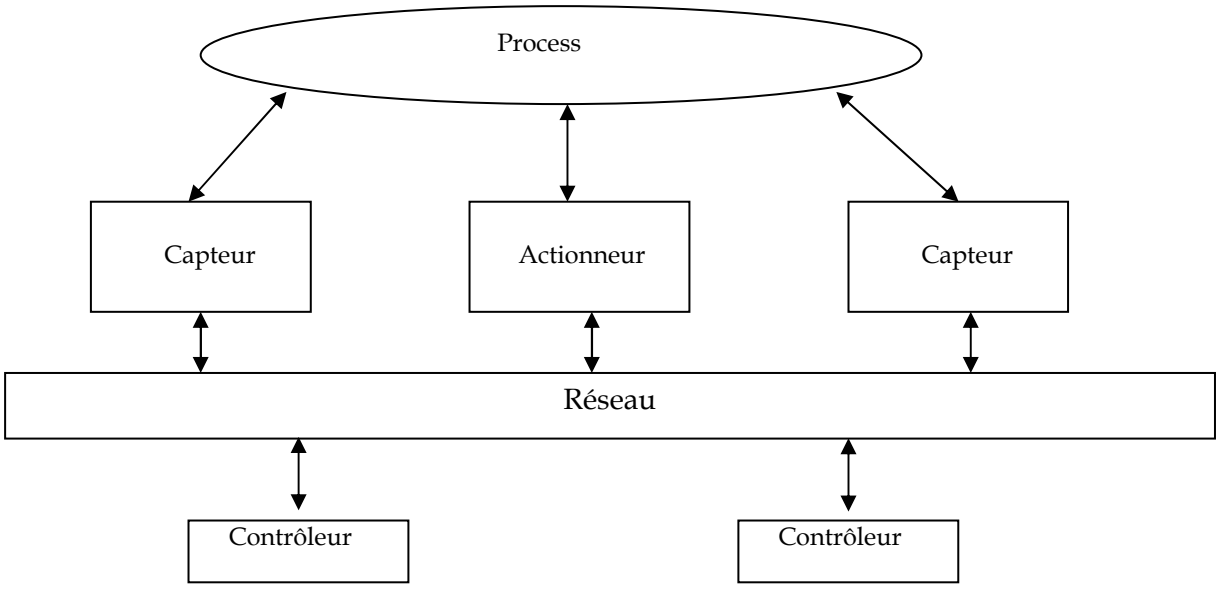


Figure 1.1 : Installation typique d'un SCR

Deux configurations générales typiques de SCR sont présentées dans les parties suivantes.

1.2.1 Structure Directe

Structure directe : Le SCR dans la structure directe se compose d'un contrôleur et d'un site à distance contenant un système physique, des capteurs, des actionneurs et d'autres composants. Le contrôleur et le système distant sont physiquement situés à différents endroits et sont directement liés par un réseau de communication afin d'effectuer la commande en boucle fermée à distance comme cela est illustré dans la figure 1.2. Le signal de commande est encapsulé dans un message et envoyé au système distant par l'intermédiaire du réseau.

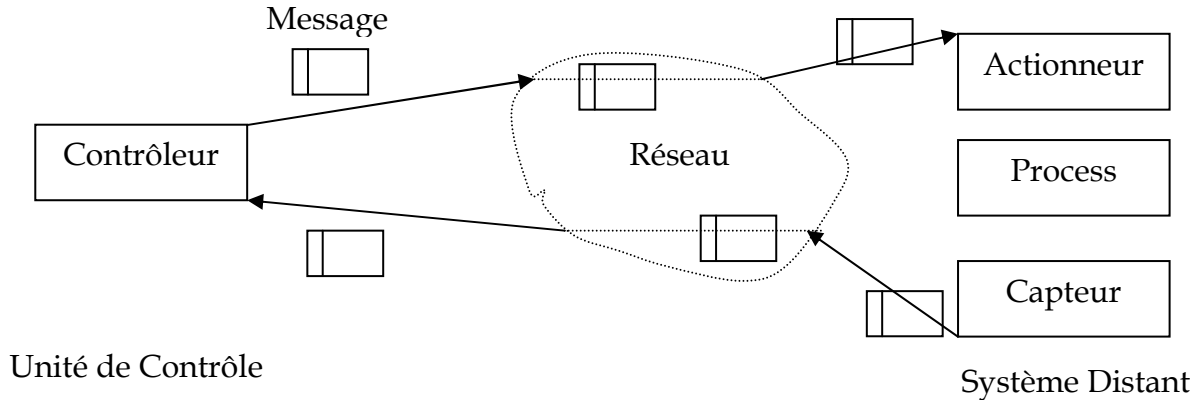


Figure 1.2 : Structure Directe

Le système distant renvoie alors la sortie du système au contrôleur mettant la mesure du capteur dans un message destiné au contrôleur. Plusieurs contrôleurs peuvent

être implémentés dans une seule unité de commande pour commander plusieurs boucles fermées dans une structure directe. Des exemples de SCR utilisant la structure directe peuvent être trouvés dans [Overstreet, 1999], [Tipsuwan, 2001].

1.2.2 Structure hiérarchique

Structure hiérarchique : la structure hiérarchique de base se compose d'un contrôleur principal et d'un système à distance contenant un contrôleur (contrôleur distant), un capteur et un système physique à réguler (figure 1.3). Périodiquement, le contrôleur principal calcule et envoie la consigne dans une trame par l'intermédiaire d'un réseau.

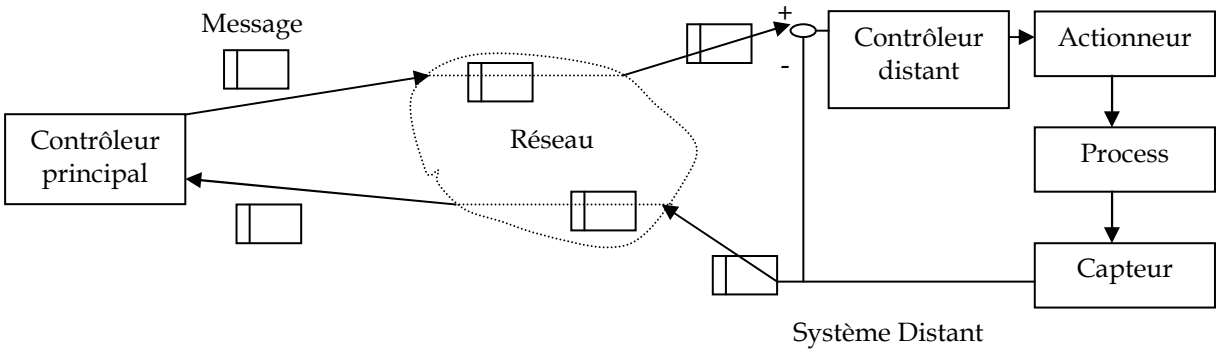


Figure 1.3 : Structure Hiérarchique

Le système à distance traite la consigne pour effectuer localement la commande en boucle fermée et renvoie la mesure du capteur au contrôleur principal. Le contrôleur distant doit satisfaire les performances de la boucle fermée avant de recevoir le message du contrôleur principal. Cette structure est employée dans plusieurs applications comme celles des robots mobiles [Tipsuwan, 2002] et de télé-opération [Tarn, 1998].

L'utilisation de la structure directe ou hiérarchique dépend des conditions d'application et des préférences du concepteur. Par exemple, un robot manipulateur exige habituellement plusieurs moteurs qui tournent simultanément ensemble. Il peut être plus commode et plus robuste d'employer un contrôleur existant et de formuler le problème dans une structure hiérarchique. D'autre part, 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.

.

1.3 Problématique des SCR

Un point important à considérer en étudiant les SCR est que le comportement et la performance du système dépendent largement des caractéristiques et de la qualité de service du réseau, comme le mode d'accès au medium, la largeur de la bande passante, le trafic sur le réseau [Galdun et al, 2007]. Pour cette raison, l'étude d'un SCR doit toujours être une étude globale ou une approche de co-design [Berbra et al, 2007]. Dans les applications de commande, les réseaux candidats doivent chaque fois que c'est possible vérifier deux contraintes : des délais bornés et une transmission garantie, c'est-à-dire qu'un message doit être envoyé et consommé avant des dates limites fixées par avance. Ces contraintes motivent l'utilisation des réseaux de communication avec des modes d'accès au medium capables d'ordonnancer les messages d'une façon à satisfaire les exigences de temps réel. Plusieurs protocoles ont étés proposés pour répondre aux besoins des systèmes automatiques commandés en réseau. Typiquement, une transmission bornée peut être réalisée par des protocoles d'accès non destructifs comme le protocole *Controller Area Network* (CAN) [CAN, 1990], ou par des protocoles à gestion d'accès 'déterministe', comme *Profibus* [Profibus, 1996].

L'analyse et la conception des systèmes en boucle fermée font référence à la théorie de la commande. Généralement les processus à contrôler sont de nature continue, l'utilisation des ordinateurs et des réseaux dans la boucle de commande demande la discrétisation du temps, ce qui implique que les variables du système ne sont connues qu'à des instants donnés. Dans les boucles fermées distribuées autour d'un réseau, les informations sont envoyées sur le réseau ; les activités suivantes sont exécutées à chaque période d'échantillonnage (figure 1.4) :

- Le capteur prélève des mesures sur le système commandé et prépare un message pour être envoyé via le réseau à destination du contrôleur. Cette tâche inclut la conversion analogique numérique et la préparation du message (encapsulation des données dans une trame). Le temps de cette tâche entraîne un retard *ts*. Ce retard peut être considéré comme constant et dépend des caractéristiques physiques et de l'architecture numérique du capteur.

- Le message du capteur est envoyé à travers le réseau au contrôleur. Cette tâche induit un délai de transmission, *tsc*. Ce délai dépend de la taille du message, du trafic circulant sur le réseau, du protocole et du medium de communication.

- A la réception d'un message de la part du capteur, le contrôleur, suivant son algorithme de commande, élabore une nouvelle commande pour l'envoyer à l'actionneur : le temps de calcul et le temps de codage induisent un nouveau retard *tc.*

- De même le message du contrôleur subit un délai avant de gagner l'accès au medium suivi du temps de transmission : le temps total est désigné par *tca*. Ce temps est en fait constitué des trois délais suivants : $T_{attente}$, $T_{bloqué}$ et T_{prop} (voir § 1.5.1.1 du chapitre 1).

- L'actionneur reçoit le message du contrôleur, décode le message et, après la conversion numérique analogique, applique la commande sur le processus. Ce nouveau délai au niveau actionneur est noté *ta.*

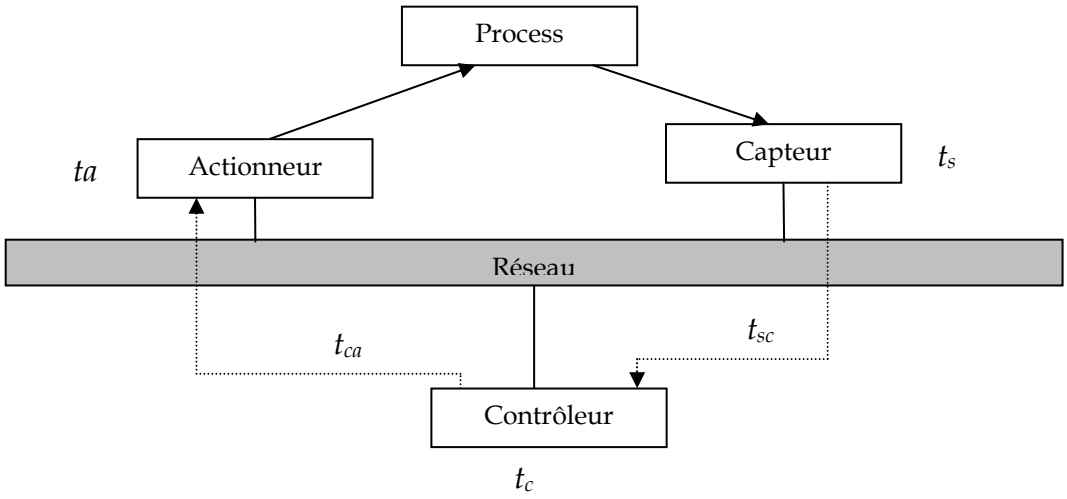


Figure 1.4 : Délais dans une boucle de régulation [Yépez et al, 2002]

Les délais de transmission sur le medium de communication sont rarement constants, ils sont variables selon le protocole utilisé et le trafic circulant sur le medium ; ces messages peuvent subir des délais aléatoires voire des pertes. L'environnement extérieur peut également avoir des conséquences sur ces délais, comme les perturbations électromagnétiques et d'autres perturbations qui peuvent affecter des messages en transmission, ce qui nécessite des retransmissions qui entraînent des délais supplémentaires.
Plusieurs études ont traité des problèmes liés aux systèmes commandés en réseau. Dans [Zhang et al, 2001], les auteurs notent deux approches dans l'accommodation des SCR aux problèmes posés par le réseau. Une première solution consiste à concevoir un système de commande indépendant des délais et des pertes mais de concevoir un protocole de communication qui en minimise l'occurrence (*control of network*). La seconde approche revient à traiter le protocole réseau et le trafic comme des conditions données et de concevoir des stratégies adéquates avec ces informations (*control over network*).

1.3.1 Commande du réseau (*control of network*)

Ces méthodes consistent à concevoir un réseau qui répond au mieux à la qualité de service exigée par l'application, il s'agit de gérer au mieux les ressources de

communication afin de répondre aux besoins de l'application de commande en minimisant les retards et les probabilités des pertes des messages. Ces approches peuvent être classées en deux catégories :

- méthodes basées sur des algorithmes d'accès au medium,
- méthodes basées sur des algorithmes d'ordonnancement au niveau des files d'attentes.

Les méthodes basées sur des algorithmes d'accès au medium suggèrent de proposer des nouveaux protocoles d'accès ou bien d'améliorer des protocoles existants en ajoutant des sous-couches, le but étant de compenser les retards et dimunier les taux de pertes de messages. On peut citer les travaux de [Walsh et al, 1999] qui ont proposé un protocole d'accès au medium appelé *Try-once-Discard* (TOD), protocole basé sur les caractéristiques du NCS étudié. La priorité est donnée au nœud qui émet le signal qui a le plus varié par rapport à l'émission précédente (une pondération est en outre attribué à chaque nœud). D'autres méthodes s'appuient sur des protocoles existants dans le but de les améliorer afin de répondre à des contraintes de temps réels. La plupart de ces travaux ont concerné le réseau Ethernet : en effet, sa méthode de contrôle d'accès au medium peut générer des collisions durant la transmission : de plus l'algorithme pour résoudre les collisions est de nature stochastique, c'est pourquoi l'Ethernet classique a été utilisé uniquement dans des applications non critiques. Plusieurs travaux ont été menés dans ce domaine, des travaux visant à améliorer les performances d'Ethernet, voire rendre déterministe les réseaux basés sur Ethernet. L'idée est de rendre Ethernet capable de distinguer entre deux trafics, le trafic temps-réel et le trafic non temps-réel. Dans [Yavatkar, 1992] les auteurs proposent un algorithme appelé *Predictable Carrier Sense Multiple Access* (PCSMA). Les trames sont classées en deux types, les messages périodiques qui correspondent aux messages temps-réel et des messages non périodiques pour les messages non temps-réel, l'accès est modifié de telle sorte que si une collision se produit entre une trame périodique et une trame non périodique, cette dernière arrête la transmission. Pour les collisions d'autres types, le protocole CSMA/CD est suivi. D'autres solutions comme *Realtime Ethernet* [Chiueh, 1994] proposent l'utilisation de la technique de passage du jeton. Tant qu'il n'y a pas besoin de trafic temps-réel les messages sont gérés selon le protocole CSMA/CD. Si une station veut envoyer du trafic temps-réel, elle demande le jeton pour passer en mode temps-réel et attend l'acquittement de tous les autres nœuds sur le réseau, l'accès au mode temps-réel est parfois géré par une station moniteur [Pritty, 1995], ce qui nécessite un matériel dédié et du logiciel non compatible avec le standard Ethernet. L'avènement des commutateurs permet de développer des architectures sans collisions Ethernet commuté [Lee, 2002] mais introduit également des systèmes beaucoup plus complexes ainsi que des temps de latence à chaque traversée d'un commutateur.

Les méthodes basées sur les algorithmes d'ordonnancement au niveau des files d'attentes sont nombreuses, elles sont pour la plupart algorithmiques, on peut citer les FIFO (*First In First Out*), LIFO (*Last In First Out*), PRIOR (servis selon la priorité des messages), WFQ (*weighted round robin*), EDF (*Earliest deadline First*) [Lui et al, 2003] [Georges et al, 2005]. D'autre algorithmes on été proposés suivant les besoins

comme l'algorithme appelé (m,k)-WFQ proposés par Koubaa [Koubâa, 2004] qui intègrent les contraintes (m,k)-firm [Wang et al, 2000].

1.3.2 Commande via le réseau (*control over network*)

La problématique des SCR s'inscrit aussi dans le cadre général des systèmes à retard. Les SCR sont des systèmes à retards où le retard est dû à l'utilisation du réseau. Les points clés d'un système à retard sont la modélisation du retard, la commande optimale et la robustesse [Richard, 2003]. Ces travaux supposent que les informations concernant le retard sont connues, ou calculables.

L'étude des performances d'un système continu commandé fait appel à la représentation d'état. La transformée de Laplace pour les systèmes à temps continu et la transformée en *z* pour les systèmes à temps discret permettent de modéliser les retards. Ces transformées ne permettent toutefois pas de prendre facilement en compte toutes les formes de retard temporel.

1.3.2.1 Modélisation du retard pour un système à temps continu

La représentation de Laplace permet de modéliser les systèmes régulés à temps continu. Dans le cas d'un retard constant, la transformée du retard est :

$$e^{-ts}$$

La représentation en p concerne naturellement les systèmes continus ; dans le cas des systèmes échantillonnés, il faut faire apparaître le phénomène de quantification à l'aide d'un bloqueur.
Les retards non constants ne sont pas directement pris en compte sous forme de transformée en p car le calcul direct (au niveau du produit de convolution servant à la définition de la transformée) n'est plus possible.
Les SCR étant par nature discrets, il n'est pas possible de les modéliser en p. Dans le cas où l'algorithme de commande discret a été transposé d'un contrôleur continu, il est cependant envisageable de conserver le modèle du contrôleur continu comme une approximation du contrôleur discret. Cette approche a le mérite de permettre d'obtenir facilement des indications sur l'influence possible de l'introduction des retards. Une telle approche a été utilisée dans de nombreuses études qui cherchaient avant tout à montrer que le retard peut être dangereux [Kocick et Sorel, 2000] [Blum et Juanole,1999].

1.3.2.2 Modélisation du retard pour un système à temps discret

La transformée en *z* est la représentation naturelle des systèmes échantillonnés, elle permet de modéliser à la fois la loi de commande et le système physique sous l'hypothèse que les instants d'échantillonnage et de commande sont synchronisés et

périodiques. La représentation du système physique est discrétisée, ce qui simplifie la modélisation en ne nécessitant plus l'utilisation d'un bloqueur (inclus implicitement dans la transformée en *z*). La notation en *z* permet de lier des échantillons. Les échantillons sont obtenus par une acquisition périodique, l'opérateur z^{-1} peut être vu comme un retard d'une période d'échantillonnage (appelée T dans la suite).
La représentation d'un retard inférieur à une période d'échantillonnage n'est pas possible, car par définition de la transformée en *z*, le temps n'a pas de sens entre deux échantillons. On peut par contre modéliser des retards supérieurs à une période d'échantillonnage, a condition que ce retard soit multiple de la période. Ainsi un retard de k périodes se modélise par :

$$z^{-k}$$

Pour représenter des retards inférieurs à la période, la technique la plus utilisée consiste à considérer une modélisation en z avec une période T' telle que T la vraie période d'échantillonnage soit un multiple de T'. On peut alors formaliser des retards qui sont des multiples de T' et pas seulement de T. La thèse de Nilsson présente ces différentes techniques [Nilsson, 1998].
Il est possible de considérer des versions adaptées des transformées en *z*, mais leur utilisation n'est pas toujours simple à mettre en œuvre. On peut citer la transformée en *z* modifiée [Aström et Wittenmark, 1990]. Cependant, la prise en compte des retards variables reste délicate et, même si différentes propositions on été faites, elles restent difficilement exploitables de manière analytique.
En conclusion, la modélisation sous forme de transformée en *z* du système régulé est intéressante mais n'autorise pas de considérer une grande variété de modèle d'implémentation. L'intérêt principal de la méthode est qu'elle permet de conclure rapidement dans le cas d'un retard constant équivalent à une période. Dans le cas général, si pour un retard constant équivalent à une période il n'y a pas de dégradation importante de la qualité, on considère que c'est aussi le cas pour tout retard de durée inférieure. Il est cependant possible de trouver des cas où ce n'est pas vérifié [Ogata, 1987].

1.3.2.3 Méthodes pour accommoder les retards induits par le réseau

L'objectif de ces approches est de prendre en compte l'influence du réseau dans les diverses expressions de la commande. Halevi et Ray [Halevi et Ray, 1988] ont proposé une méthode basée sur un modèle déterministe étendu pour commander un système linéaire avec des délais périodiques. Nilson [Nilson, 1998] introduit différents modèles de retards dans les SCR, le retard pouvant être considéré comme fixe ou stochastique selon les cas, Nilsson part de l'hypothèse que le retard $t < T$ où T est la période d'échantillonnage du système. L'un des gros problèmes rencontrés lors de la conception des SCR sont les retards variables dus à l'utilisation des réseaux dans la transmission des données. Des méthodes basées sur les files d'attentes [Lelevé, 2000] [Kosgue, 1996] ont été utilisées pour rendre « déterministes » les délais aléatoires induits pas le réseau. La commande et les mesures sont stockées dans une file d'attente (FIFO en général) et sont ensuite émises à période constante. Bien que

ces méthodes rendent le retard constant, elles introduisent des retards additionnels qui peuvent dégrader la performance du système.

Différents travaux on été menés au CRAN autour de cette problématique, on peut citer les travaux de [Michaut, 2003] concernant l'adaptation de l'application à la qualité de service et du réseau. D'autres travaux [Rondeau, 2001] se sont portés sur les aspects reconfiguration et plus particulièrement sur l'optimisation de la topologie réseau Ethernet. Les travaux de [Georges, 2005] s'intègrent aussi dans la problématique des SCR et proposent une méthode de calcul de majorant des délais de traversée des réseaux Ethernet commuté, en faisant appel à la théorie nommée calcul réseau [Cruz, 1991].

1.3.3 Approche intégrée co-design

Les approches de commande via le réseau intègrent la qualité de service du réseau comme une donnée d'entrée pour la conception des systèmes automatiques. A l'inverse, les approches de commande du réseau prennent comme entrées les contraintes du système commandé, par la suite le réseau est conçu d'une manière a satisfaire au mieux ces contraintes. Certains travaux visent à considérer simultanément les informations concernant le réseau (retard, taux de pertes, gigue,…) et celles du système commandé (stabilité, CSD,…), ces travaux sont relativement récents [Brahimi, 2007]. Dans [Juanole et Mouney, 2005], les auteurs signalent divers points importants à prendre en compte dans une approche de co-design. Le point essentiel est que l'ordonnancement des tâches (dans un contexte de ressources partagées) et la priorité des messages doivent être changés en ligne en fonction des performances des systèmes commandés.

1.4 Protocoles pour les systèmes distribués

Un système distribué se compose d'un ensemble de nœuds qui peuvent être des contrôleurs, des capteurs, des actionneurs et d'autres composants, reliés à un medium de communication. La topologie peut être une topologie en étoile ou en arbre. La topologie la plus répandue est la topologie en bus, cette topologie permet à tous les nœuds reliés un accès direct aux messages transmis sur le bus, tous les nœuds reçoivent le même signal en même temps. Le protocole de transmission est responsable de différentes tâches telles que l'accès au bus et l'ordonnancement des messages. Cette section est consacrée à la comparaison des différents types de protocoles. Chaque protocole se caractérise par sa stratégie d'accès au medium ainsi que par la gestion de la responsabilité de la communication (maître-esclave, jetons…).

1.4.1 Accès au medium

L'accès au medium se fait généralement de deux manières, accès multiple et accès unique.

1.4.1.1 Accès unique

L'accès unique ou accès contrôlé ne permet à une station d'envoyer des messages que dans des tranches de temps déterminées à l'avance. On distingue deux types d'accès contrôlé : centralisé et décentralisé. Lorsque la gestion des accès est centralisée dans une station maître, la méthode d'allocation des droits d'accès se fait par interrogation d'autres stations dites esclaves.
La gestion décentralisée est basée sur la circulation entre les stations d'un droit d'accès, appelé jeton. Une station qui reçoit le jeton émet une ou des trames si elle a des informations à transmettre, pendant une durée limitée, et émet le jeton vers la station suivante. Si elle n'a rien à envoyer, elle émet directement le jeton.
La durée individuelle d'émission étant limitée, il est donc facile de déterminer le temps maximal de rotation de jeton, pour cette raison ces méthodes sont qualifiées de déterministes. Deux méthodes d'accès par jeton sont particulièrement connues : l'une sur une topologie en bus (*Token Bus*), et l'autre sur une topologie en boucle (*Token Ring*).

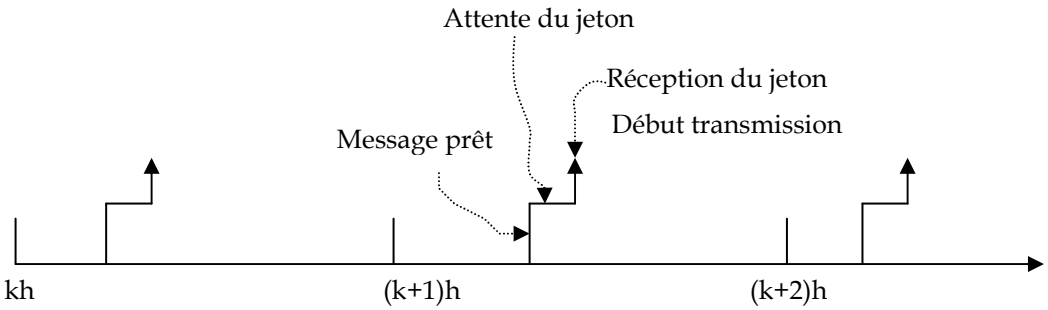


Figure 1.5 : Accès unique (token ring)

Un diagramme de synchronisation pour ce type de réseau est montré sur la figure 1.5.

1.4.1.2 Accès multiple

L'accès multiple permet à chaque station d'émettre à n'importe quel instant à condition que le medium soit libre. CAN (*Controller Area Network*) et Ethernet sont deux réseaux employant un accès multiple au medium (*Carrier Sens Multiple Access*).

La figure 1.6 montre deux nœuds transmettant des messages via un medium partagé. Avant d'envoyer des messages un nœud surveille le medium, si le medium est libre (pas de messages circulant sur le medium), il commence immédiatement la transmission, sinon, si le medium n'est pas libre, il attend jusqu'à ce que le réseau soit libéré. Quand deux nœuds ou plus essaient de transmettre simultanément, une collision se produit.
La manière de résoudre la collision dépend de chaque protocole. Dans DeviceNet dont les couches physique et liaison suivent le protocole CAN, chaque message est attribué d'une priorité ; lors d'une collision le message le plus prioritaire continue normalement la transmission, les autres stations arrêtent la transmission et retentent l'envoi une fois que le medium redevient disponible : CSMA/*deterministic collision resolution* (figure 1.6 : cas 2).

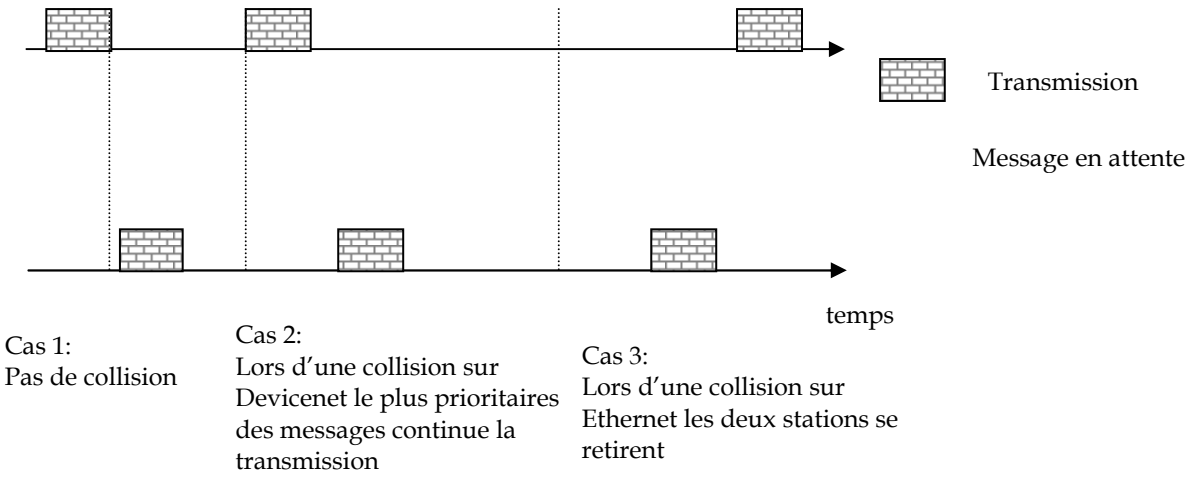


Figure 1.6 : Détections des collisions

Ethernet utilise une technique de détection de collision (*CSMA/ Collision Detection*). Quand il y a une collision, tous les nœuds affectés arrêtent la transmission. Chaque station attend un temps aléatoire décidé par l'algorithme d'attente (*backoff*), et retransmet à nouveau le message (figure 1.6 : cas 3). Le temps d'attente est calculé par tirage aléatoire dans un domaine qui croît exponentiellement avec le nombre de collisions subies par la trame tant que celui-ci est inférieur à 16. Cette méthode permet de freiner l'arrivée des messages lorsque le medium est fortement chargé. Si le nombre de collisions enregistrées par la trame dépasse une limite (16 en principe), la tentative d'émission est abandonnée.

1.4.2 Contrôle de la communication sur le bus

La décision de quand un message doit et peut être envoyé pourrait être prise par plusieurs entités du système distribué. Le contrôle de communication peut être assuré par :

- un seul nœud de contrôle du bus,
- plusieurs nœuds (Multi maître),
- un jeton : le nœud qui reçoit le jeton a la pleine autorité sur le bus.

Un protocole avec un seul maître se fonde sur un nœud principal pour manipuler la communication. Ce nœud principal résout les problèmes de conflits possibles. Il est à noter que sans redondance du nœud maître, une défaillance sur ce nœud provoquera la défaillance totale du système. Puisque le nœud principal prend soin de l'administration de messages, il est facile d'analyser et de surveiller la communication. Le temps de calcul pour ce concept sera ainsi élevé à cause des messages de commande des nœuds participants. Ce temps de calcul doit compter les messages de contrôle envoyés par la station maître et les messages d'accord envoyés par les autres stations avant chaque tentative de transmission.

Un système qui souhaite obtenir une robustesse plus élevée peut choisir une approche qui se compose de plusieurs nœuds qui peuvent contrôler la transmission. Tous les nœuds peuvent contrôler la communication dans un système multi maître, et donc il est plus facile de concevoir un système robuste. Dans un système tolérant aux fautes aucun nœud défaillant ne devrait dégrader la communication et des dates limites devraient être tenues.

1.4.2.1 Comparaison

Dans la littérature on trouve plusieurs rapports comparatifs selon différents critères. Dans son rapport, [Rushby, 2003] décrit et compare quatre protocoles de communication destinés à être implantés au cœur des systèmes critiques : deux protocoles destinés à l'avionique (SPIDER, développé par la NASA, et SAfeBUS, développé par Honeywell), et deux protocoles destiné à l'automobile (TTP/C et FlexRay). Une comparaison entre TTP/C et CAN [CAN, 1990] peut se trouver dans les rapports de H. Kopetz [Kopetz, 1998], Kopetz part du principe que le déterminisme et la composabilité[1] servent la sûreté de fonctionnement d'un système. Le livre du groupe CIAME [CIAME, 1999] semble être le rapport le plus complet pour la comparaison des réseaux de terrain dédié à l'utilisation dans le domaine des systèmes commandés en réseau, la comparaison est faite sur plusieurs critères, des critères de coûts, de sécurité ainsi bien que des critères liés à la sûreté de fonctionnement. [Lian et al, 2001] donne une analyse et une comparaison de trois types de réseaux particuliers : Ethernet, DeviceNet et ControlNet. Les

[1] Composabilité : une conception composable est une conception au sein de laquelle les applications individuelles ne sont pas affectées par les choix des autres applications avec lesquelles elles sont intégrées.

caractéristiques prises en compte sont la taille des données et la périodicité des messages. Les besoins comprennent la garantie de transmission et des délais bornés. Les conclusions de ces travaux sont les suivantes : malgré les haut débits du réseau Ethernet, ce réseau est pénalisé par le non déterminisme de sa méthode d'accès au medium et n'est pas recommandé sauf pour des applications non sensible au temps. Pour des communications caractérisées par des contraintes déterministes, Lian ne retient que des réseaux de terrain classiques comme DeviceNet ou ControlNet. Les résultats montrent aussi que DeviceNet est plus performant pour des messages courts, et pour un trafic élevé.

1.5 Qualité de service

Le réseau de communication sert à transmettre les messages entre les différents composants d'une application. Selon l'application qu'il soutient le réseau est amené à répondre à des qualités de service exigées par l'application. On distingue deux types d'applications :

- Application temps-réel : les applications dites temps-réel ont un besoin ferme en terme de délai et de variation de délai (gigue). Pourtant elles peuvent tolérer quelques pertes de paquets. Les systèmes commandés en réseau sont bien des applications temps-réel, notamment celles avec des périodes d'échantillonnages courtes de l'ordre de quelques ms.

- Application non temps-réel : les applications dites non temps-réel sont plus sensibles aux pertes de paquets et redoutent moins des délais élevés. L'intégrité des données est souvent l'aspect le plus critique pour le déroulement correct de l'application. Souvent dans ce cas on utilise des mécanismes d'acquittement sans aucune garantie de bornes sur les délais et la gigue (*Transport Control Protocol*).

1.5.1 Paramètres de la qualité de service

1.5.1.1 Délai

Lors d'une transmission via un réseau on peut distinguer plusieurs délais :
- les délais de la station émettrice,
- les délais de transmission sur le réseau,
- les délais à la station destinataire.

Avant d'être reçu, chaque message a subi un certain nombre de retards :
- T_{pre} temps nécessaire au codage dans la station source,
- T_{tampon} temps d'attente dans la station source avant émission nécessaire au vidage de la fille d'attente (tampon),

- $T_{bloqué}$ temps d'attente d'accessibilité au médium (disponibilité et droit d'accès),
- T_{prop} durée de la transmission sur le médium.

Une fois que le message est reçu par la station destinataire, le décodage du message nécessite un temps T_{post} avant d'être prêt à la consommation (figure 1.7).

Le délai est le paramètre le plus important pour les applications temps réel, il est défini comme étant l'intervalle de temps entre le moment de production d'une information jusqu'à l'arrivée de cette information chez son destinataire :

$$T_{delai} = T_{pre} + T_{tampon} + T_{bloqué} + T_{prop} + T_{post} \qquad (1.1)$$

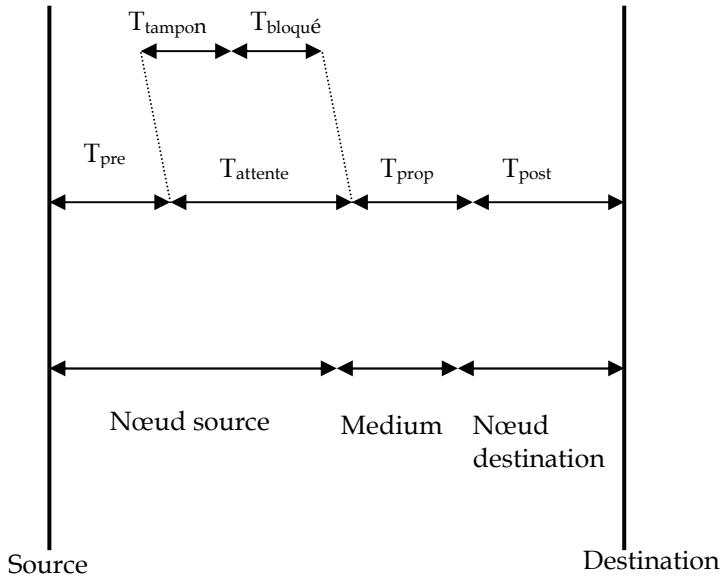


Figure 1.7 : Délais de bout en bout

1.5.1.2 Gigue

La gigue est aussi une métrique importante pour les applications temps-réel. Elle se réfère généralement à la variation du délai, entre le délai maximum et le délai minimum.

1.5.1.3 Taux de pertes

Le taux de pertes représente le pourcentage des messages qui ne peuvent pas atteindre leur destination dans un intervalle de temps spécifique. Cette perte peut être le résultat d'un rejet de paquets lorsque les ressources sont saturées ou indisponibles, ou d'un dépassement d'échéance. Pour une application temps-réel un message arrivant au delà de son échéance risque de n'être plus utile pour l'application, c'est-à-dire perdu pour l'application.

1.5.2 Qualités de performances

Dans cette section sont présentés des paramètres de performances dans le réseau [Lian et al, 2001], ces paramètres sont liés à ceux de la qualité de service de l'application et servent à donner une image de l'état du réseau.

1.5.2.1 Temps moyen de retard (*Total Average Delay*)

Pour une durée déterminée, on peut calculer la somme T_{delai}^{somme} et la moyenne T_{delai}^{moyen} des délais que subit chaque nœud sur le réseau :

$$T_{delai}^{somme} = \sum_{i \in N_{node}} \sum_{j=1}^{M(i)} T_{delai}^{(i,j)} \qquad (1.2)$$

$$T_{delai}^{moyen} = \frac{1}{N} \sum_{i \in N_{node}} \left[\frac{\sum_{j=1}^{M(i)} T_{delai}^{(i,j)}}{M(i)} \right] \qquad (1.3)$$

où N est le nombre total de nœuds sur le réseau, N_{node} est l'ensemble des nœuds du réseau, et $M(i)$ est le nombre de messages envoyés par le i[ème] nœud. $T_{delai}^{(i,j)}$ est le délai du j[ème] message envoyé par le i[ème] nœud. Si un nœud i envoie un message périodiquement, le nombre de messages envoyés par celui-ci est donné par la formule :

$$M(i) = \left\lfloor T_{total} \Big/ T_{per}^{(i)} \right\rfloor^{2} \qquad (1.4)$$

où $T_{per}^{(i)}$ représente la période d'envoi du nœud i et T_{total} est le temps total de la mission.

1.5.2.2 Efficacité du réseau (*Efficiency of Networks*)

L'efficacité du réseau P_{eff} est définie comme étant le rapport entre le temps de propagation des messages (somme des temps de propagation de l'ensemble de messages pour l'ensemble des nœuds, avec $T_{prop}^{(i,j)}$ représentant le temps de propagation du j[ème] message du i[ème] nœud) et le temps total T_{delai}^{somme} pris pour transmettre les messages y compris les temps d'attente.

[2] $\lfloor x \rfloor$ Représente le plus grand entier inférieur à x.

$$P_{eff} = \frac{\sum_{i \in N_{node}} \sum_{j=1}^{M(i)} T_{prop}^{(i,j)}}{T_{delai}^{somme}} \qquad (1.5)$$

Ainsi, si $P_{eff} \to 1$, la plus grande partie du délai vient des délais de propagation et la performance du réseau est acceptable. Dans le cas contraire $P_{eff} \to 0$, la performance du réseau est mauvaise puisque les délais d'attente dus à des collisions prennent la majorité sur le temps de propagation.

1.5.2.3 Utilisation du réseau

L'utilisation du réseau P_{util} est définie par le rapport entre le temps total de propagation et le temps total de la mission T_{total} en prenant en compte la retransmission éventuelle de messages.

$$P_{util} = \frac{\sum_{i \in N_{node}} \sum_{j=1}^{M(i)} (T_{prop}^{(i,j)} + T_{reprop}^{(i,j)})}{T_{total}} \qquad (1.6)$$

Où $T_{reprop}^{(i,j)}$ est le temps pris pour retransmettre le j^{ième} message du i^{ème} nœud. Si $P_{util} \to 0$, il y a suffisamment de bande passante pour d'autres applications et de nouveaux nœuds peuvent partager le réseau. $P_{util} \to 1$ veut dire que le réseau est presque saturé.

1.5.2.4 Stabilité du réseau

La stabilité du réseau est définie par le nombre de messages dans les tampons de chaque nœud, si ce nombre atteint un nombre donné, le réseau est instable. La stabilité du réseau est directement liée à la perte des messages. En général, les tampons sont limités et une fois que les tampons sont saturés, des messages vont être supprimés pour les libérer.

1.6 Influence du réseau sur les performances d'un SCR

Après avoir présenté les SCR en général, les différents protocoles de communication, leurs méthodes d'accès au bus, les notions de qualité de service et de performance dans les réseaux, cette partie est consacrée à l'étude de l'influence de l'utilisation des réseaux sur les paramètres de performances d'un SCR.

1.6.1 Période d'échantillonnage

Théoriquement la période d'échantillonnage doit être assez rapide pour qu'on puisse reconstruire correctement un signal. D'après le théorème de Shanon [Goodwin, 2001] la fréquence d'échantillonnage doit être au moins deux fois supérieure à la fréquence significative *fp* :

$$f > 2 \times f_p \qquad (1.7)$$

Souvent les fréquences utiles du système sont difficiles à connaître, pour cela on utilise une méthode plus simple qui repose sur la réponse du système en boucle ouverte, cette méthode définie par [Ogata, 1987] consiste à choisir une :

Fréquence d'échantillonnage qui est quatre à cinq fois inférieure au temps de montée du système. (1.8)

La figure 1.8 montre bien que le choix de la période d'échantillonnage est décisif et influe directement sur les performances du système. Dans ses travaux, [Lian, 2001] présente une méthode pour calculer les bornes B et C (figure 1.8) pour lesquelles le SCR satisfait des niveaux de performance.

Les théories traditionnelles des systèmes commandés en réseau supposent que la période d'échantillonnage est fixe, ce qui signifie que les informations concernant le processus sont envoyées au contrôleur aux instants, *hk, k=1…n*, où *h* est la période d'échantillonnage constante. Cette hypothèse simplifie considérablement l'analyse de stabilité et de performance du système.

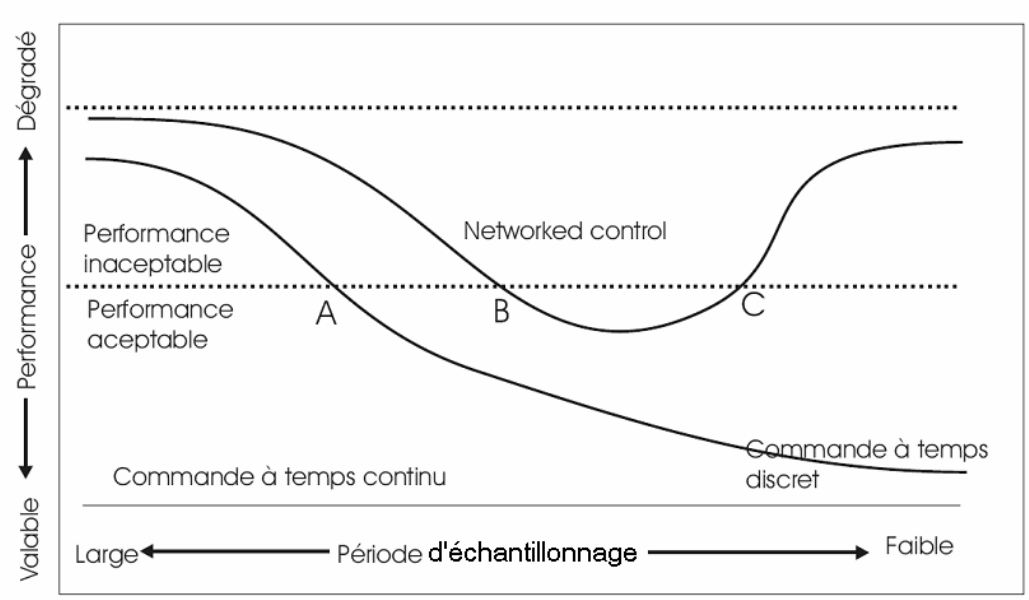


Figure 1.8 : Performance et période d'échantillonnage [Lian, 2001]

Cependant, l'hypothèse d'une période fixe ne doit pas être imposée à l'analyse de SCR. Dans la pratique la plupart des réseaux ne peuvent pas garantir une période constante de transmission, ils peuvent normalement garantir de transmettre un message avant sa date d'échéance.

1.6.2 Influence du retard sur un SCR

Les retards sur les messages se produisent quand les capteurs, les actionneurs, les contrôleurs et autres composants échangent des données à travers le réseau. Ce retard peut réduire la performance et même déstabiliser le système. Dans les réseaux *CSMA* les délais sont aléatoires et le pire temps de transmission n'est pas borné. Dans les réseaux à accès unique les délais se produisent en attendant l'arrivée de jeton si on ne considère pas les perturbations extérieures, dans ce cas ces délais peuvent être périodiques.
Andref [Andref, 1994] dans sa thèse a prouvé à l'aide d'un exemple simple que les délais sur les messages peuvent déstabiliser le système.
Juanole et Blum dans leurs études [Juanole et Blum, 1999] ont présenté une méthode basée sur les réseaux de Petri stochastiques permettant de quantifier l'influence des performances temporelles d'une application (temps de réponse, perte des messages) sur la marge de phase d'un système automatique distribué.
Dans de nombreuses études sur la commande des systèmes dits 'à retard' [Richard, 2003], [Ray, 1994], [Marti, 2001], [Lee, 1994], l'objectif des travaux est généralement de proposer de nouvelles lois de commande adaptées à la présence de retards. On distingue deux types de retards : les retards constants et les retards variables.

Les retards constants sont simples à modéliser et correspondent à un phénomène de temps mort. Au contraire, la présence d'un retard variable est plus complexe à modéliser et son influence se révèle délicate à étudier.

1.6.2.1 Influence d'un retard constant

Il est possible d'estimer le plus grand retard possible vis-à-vis de la propriété de stabilité. Le critère le plus adapté est la marge de phase (liée à la représentation fréquentielle des systèmes). La marge de phase doit être supérieure à une certaine valeur $\phi_{\min}$ pour garantir la stabilité. La présence d'un retard constant agit directement sur la marge de phase. On a une relation directe entre la stabilité et le retard constant. Si on considère un critère comme le temps de réponse dans le cas du changement de consigne, ou un critère intégral de qualité sur le fonctionnement moyen du système (l'intégrale de l'erreur absolue par exemple), il est très difficile de trouver analytiquement une relation entre la qualité obtenue et la présence de ce retard.
Dans le cas général les retards ne sont pas constants, comme cela a déjà été vu, à cause des délais *tsc* et *tca* (délai d'attente et de transmission sur le medium).

1.6.2.2 Influence d'un retard variable

L'étude de l'influence d'un retard variable est beaucoup plus compliquée à mettre en œuvre. Si les retards sont variables : ils correspondent pour le système régulé à la présence d'une suite de retards $\delta\{t_k\}$ appliqués aux instants t_k. On cherche donc à caractériser la qualité du système vis-à-vis de cette suite. Dans le cas où cette suite est parfaitement connue (connaissance de chaque valeur de la suite), il est possible de trouver la réponse du système et les valeurs des différentes métriques de qualité peuvent être calculées. Souvent ces variables ne sont pas connues par avance. Si cette suite ne peut pas être connue à l'avance, certaines propriétés (les bornes min et max, moyenne…) de cette suite peuvent être utilisées pour estimer les évolutions possibles du système. Pour plus d'informations voir la thèse de M.Torngren [Martin, 1995].

D'autres travaux permettent de conclure sur la stabilité du système connaissant les bornes de la suite, il s'agit de conditions suffisantes mais non nécessaires. En d'autres termes on peut prouver que le système régulé est stable pour $t_k \in$ [tmin, tmax], par contre il est plus difficile de prouver que ces limites sont les meilleures possibles.

Dans le cas particulier où la suite possède une sous-suite qui se répète d'une manière périodique, on dispose de résultats spécifiques, dans ce cas on peut donner des critères sur la stabilité du système [Moon, 1996].

1.6.3 L'influence des pertes sur les performances d'un SCR

Les premiers travaux qui ont pris en compte l'influence de l'indisponibilité de la commande sur les performances d'un système automatique datent de 1988 [Kim, 1988].

Kim et al ont défini la notion du CSD (*control system deadline*), qui est la durée maximum pendant laquelle le contrôleur peut ne pas délivrer une commande sans que le système soit instable ou sans que le système passe dans un état dangereux. Pour la stabilité des systèmes distribués en présence de pertes, on peut se référer aux travaux de [Babak, 2003], [Zhang et al, 2001], ces travaux visent à estimer l'influence du taux de pertes sur la stabilité. Même si certains résultats analytiques sont disponibles, ils permettent donc essentiellement d'obtenir des critères de stabilité. Ces critères correspondent à des conditions suffisantes de stabilité mais ne permettent pas de conclure au mauvais fonctionnement du système en cas de non respect de ces critères.

1.7 Conclusion

L'utilisation des réseaux dans les systèmes automatiques présente plusieurs avantages en terme de réduction de câblage et de facilité de maintenance, mais en même temps rend l'étude des SCR plus complexe. A cause de l'utilisation du réseau, les retards ne sont plus négligeables et doivent être pris en compte. Dans le cas où les

retards sont constants ou bornés, le système peut être modélisé par les approches traditionnelles (Laplace, z). Cette hypothèse n'est pas toujours vérifiée, surtout lorsque la qualité de service du réseau change au cours du temps : ce qui se traduit par des instants d'activations non périodiques.

Il est possible dans certains cas d'avoir une expression analytique donnant une caractéristique de qualité d'un système en fonction des caractéristiques temporelles du réseau. Dans le cas général (retards variables), l'étude analytique est difficilement réalisable.

La plupart des études se focalisent sur la stabilité du système [Zhang et al, 2001]. Cependant même dans le cas où on sait garantir mathématiquement la stabilité, on peut être en présence d'une dégradation importante de la qualité de contrôle. Cette dégradation ne peut pas toujours être évaluée de manière analytique. Pour une étude de la sûreté de fonctionnement, il ne suffit pas d'étudier les limites de l'instabilité du système car la défaillance peut arriver bien avant. Le deuxième chapitre présente un état de l'art de la sûreté de fonctionnement des systèmes commandés.

Chapitre 2 Sûreté de fonctionnement des systèmes commandés

2.1 Introduction

On appelle sûreté de fonctionnement prévisionnelle d'un système l'évaluation des paramètres de fiabilité, maintenabilité, disponibilité et sécurité à partir de la connaissance de ces mêmes paramètres pour les composants constituant le système. L'idéal est d'établir une fonction décrivant formellement cette dépendance. Dans le cas de la fiabilité, cette fonction s'appelle structure fiabiliste ou encore fonction de fiabilité. Elle peut prendre de nombreuses formes selon la méthode d'analyse utilisée pour l'établir, par exemple une expression analytique (fonction de structure selon A. Kaufmann) [Kaufmann et al, 1975], un graphe (diagrammes de succès, réseaux de fiabilité, graphe de Markov, RdP), un arbre (arbre de défaillances).

Lorsque cette fonction de structure est invariante dans le temps, on parle de fiabilité statique, dans le cas contraire on parle de fiabilité dynamique. C'est par exemple le cas des systèmes qui se reconfigurent suite à l'occurrence de certains événements. A chaque configuration correspond une forme de la structure fiabiliste. Certaines des méthodes précédemment citées ont été adaptées de manière à prendre en compte certains aspects spécifiques de la fiabilité dynamique (arbres de défaillance dynamiques…).

Lorsque cette fonction n'est pas accessible, on cherche alors à la remplacer par un modèle de simulation dysfonctionnel du système qui permettra d'évaluer les paramètres par une statistique sur un grand nombre d'histoires. C'est la méthode dite de Monte-Carlo. Ces modèles sont généralement de type état – transition (automates à états finis ou RdP stochastiques).

Lorsque les changements d'état du système ne dépendent plus seulement des défaillances ou des réparations des composants du système mais aussi des variables continues de ce système (par exemple franchissement d'un seuil) on a affaire à un système dynamique hybride ; pour un tel système également, la fonction de structure est dynamique.

Aujourd'hui, le concept de fiabilité dynamique recouvre aussi les autres aspects de la sûreté de fonctionnement : maintenabilité, disponibilité et sécurité.
Les systèmes commandés en réseau sont des systèmes automatiques classiques où les différents composants sont distribués autour d'un medium de communication. Pour

la sûreté de fonctionnement, la problématique de l'évaluation hérite à la fois des contraintes et caractéristiques des systèmes automatiques classiques connus comme étant des systèmes dynamiques, ainsi que de la problématique des systèmes distribués autour d'un medium de communication. Ce chapitre introduit ces différents aspects. Au préalable, nous allons brièvement rappeler les différentes méthodes utilisées dans l'étude de la sûreté de fonctionnement.

2.2 Méthodes pour la sûreté de fonctionnement

Les méthodes d'évaluations quantitatives de prévision des fautes peuvent être divisées en 2 catégories [Zwingelstein, 1999] :
– les méthodes dites *statiques* : la fonction de structure ne change pas au cours du temps,
– les méthodes dites *dynamiques* : la fonction de structure change au cours du temps.

2.2.1 Méthodes pour les systèmes statiques

Cette partie présente les méthodes statiques les plus utilisées dans l'étude de la sûreté de fonctionnement. L'intérêt est ici d'avoir une vision globale de ce que ces méthodes peuvent apporter, les méthodes dynamiques étant pour une part constituées d'extensions de ces méthodes statiques.

2.2.1.1 Diagramme de succès

Le diagramme de succès (figure 2.1) est un modèle permettant de représenter le comportement du système. Ce modèle est caractérisé par un diagramme avec une entrée E et une sortie S et des blocs représentant les éléments du système. Les arcs reliant les blocs traduisent les relations entre les différents éléments du système. La défaillance d'un composant se traduit par la disparition sur le diagramme du bloc correspondant. S'il n'y a plus alors de chemin pour aller de l'entrée à la sortie, le système est défaillant. Par conséquent la structure du diagramme naturellement proche de la structure du système contient un certain nombre de chemins de succès. Plus il y en a, plus le système est fiable.

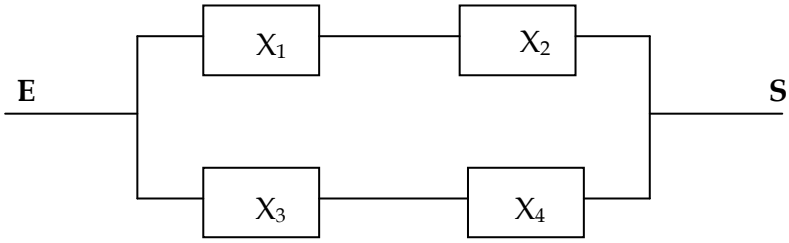


Diagramme de succès: {X1- X2 ; X3- X4}

Figure 2.1 Diagramme de succès

- Les blocs dont la défaillance entraîne la défaillance du système sont placés en série. Par exemple, pour un système de n éléments en série, l'unique chemin de succès est constitué de l'ensemble des éléments du système. Si l'on considère l'événement E_i : l'élément C_i fonctionne à l'instant t, alors la probabilité que le système fonctionne à l'instant t est définie par la relation :

$$P_{\text{système fonctionne}} = \prod_{i=1}^{n} P(E_i)$$

- Les blocs dont la défaillance ne provoque la défaillance du système qu'en combinaison avec d'autres sont disposés en parallèle. Pour un système constitué de n éléments en parallèle, il existe n chemins de succès, chacun d'eux étant composé d'un élément. Si l'on considère l'événement $\overline{E_i}$: l'élément C_i est défaillant à l'instant t alors la probabilité que le système soit dans un état de fonctionnement à l'instant t est définie par la relation :

$$P_{\text{système fonctionne}} = 1 - \prod_{i=1}^{n} P(\overline{E_i})$$

L'exploitation des diagrammes de succès permet de quantifier la fiabilité de systèmes représentables par des associations parallèle série ou série parallèle non réparables. La présence d'un composant associé ni en parallèle ni en série peut être prise en compte grâce au théorème des probabilités totales. Cependant, si les éléments du système peuvent revêtir plus de deux états (état défaillant et état de bon fonctionnement), alors cette méthode n'est plus applicable. Elle ne prend pas en compte le cycle de défaillance/réparation des composants.

2.2.1.2 Arbre de défaillance

L'arbre de défaillance est une méthode classique dans le domaine de la sûreté de fonctionnement. Appelée aussi Arbre des Causes, Arbre des Fautes ou Arbre des Défauts, elle est née en 1962 dans les bureaux de la société BELL Téléphone. Il s'agit là d'une technique déductive (on ne cherche pas les effets d'une défaillance mais ses causes) basée sur la recherche des combinaisons d'événements pouvant conduire à une défaillance.
On part d'un événement indésirable unique et bien défini. Dans le cas de l'étude de la fiabilité ou de la disponibilité, cet événement indésirable est le non-fonctionnement du système. L'arbre de défaillance représentera graphiquement les combinaisons des événements qui conduisent à la réalisation de cet événement indésirable.
L'arbre est constitué de niveaux successifs d'événements reliés entre eux par des opérateurs logiques. Chaque événement issu d'un opérateur est obtenu par la combinaison des événements d'entrée de cet opérateur. Il est important de noter que contrairement à l'AMDEC, dont l'objectif est avant tout qualitatif, la méthode des arbres de défaillance est fréquemment utilisée pour des évaluations quantitatives. En effet, des probabilités d'apparition (pour une durée donnée) peuvent être associées à des occurrences d'événements, et des opérations sur ces probabilités peuvent être réalisées à partir des opérateurs qui relient les événements entre eux. L'exploitation d'un arbre de défaillance repose essentiellement sur l'observation des coupes minimales. Une coupe est un ensemble d'événements susceptibles d'entraîner

l'événement redouté situé à la racine de l'arbre. Une coupe est dite minimale lorsqu'elle n'en contient aucune autre. Cette approche permet de mettre en lumière les événements critiques qui peuvent causer l'événement redouté et d'en évaluer la probabilité d'apparition. La probabilité de l'événement sommet se calcule par la probabilité du polynôme réunion des coupes minimales par le théorème de Poincaré ou par transformation de ce polynôme par le théorème de Shannon (transformation de l'arbre en arbre de décision binaire).

2.2.1.3 Analyse des modes de défaillance, de leurs effets et de leurs criticités (AMDEC)

Cette méthode, qui fut employée pour la première fois dans les années 1950 dans le domaine militaire, fait aujourd'hui l'unanimité dans toutes les industries à risque (nucléaire, militaire, énergie,...). Il s'agit d'une méthode inductive dont l'objectif est de qualifier les causes et les effets des défaillances qui peuvent affecter les composants (ou sous-systèmes) d'un système. La première étape de la méthode est l'identification de tous les modes de défaillance -et de leurs causes- de chaque composant. La seconde étape est basée sur l'étude des effets de ces modes de défaillances. On juge alors la gravité g de chaque événement que l'on peut quantifier à partir des niveaux de gravité. Certaines caractéristiques comme la fréquence d'apparition f de la défaillance et sa détectabilité d peuvent aussi apparaître. Il est alors possible de calculer la criticité C de la défaillance. Cette méthode est généralement déployée pour identifier des composants critiques (coupes d'ordre 1) et proposer la mise en place de barrières visant à les éliminer (remplacement par des coupes d'ordre 2 au moins). Elle est toujours préalable à une étude par arbres de défaillances.

2.2.1.4 Arbre de conséquence

La méthode de l'arbre de conséquence (ou arbre d'événements) consiste à établir les conséquences faisant suite à un événement initiateur et les enchaînements possibles des fonctionnements ou dysfonctionnements des différentes parties du système (phénomènes de propagation). Un exemple est donné figure 2.2.

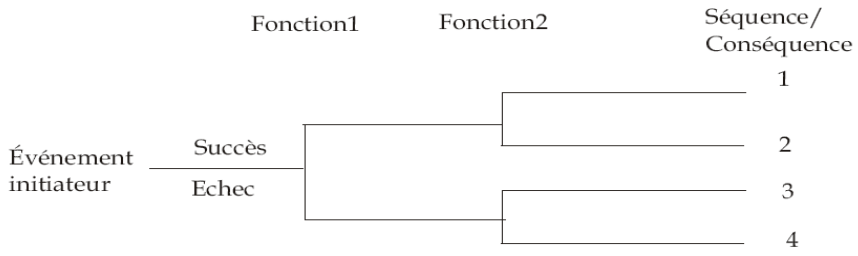


Figure 2.2 Arbre de conséquence

Cette méthode permet par propagation d'évaluer les probabilités d'occurrence d'un événement associé à une branche quelconque de l'arbre en tenant compte des

dépendances entre événements amonts (notamment les dépendances de type cause commune et de type séquentiel).

L'intérêt de cette méthode réside dans l'exploration systématique des séquences suite à une défaillance et dans le fait que la quantification soit possible. Bien entendu, cette exploration est possible à condition d'avoir au préalable réalisé une analyse fonctionnelle du système. Cette remarque s'applique néanmoins à toutes les méthodes d'analyse de la sûreté de fonctionnement.

2.2.2 Méthodes pour les systèmes réparables

Les méthodes présentées jusqu'ici n'ont pas été conçues pour l'étude des systèmes réparables. La modélisation Markovienne est largement utilisée pour cela, lorsque les défaillances et réparations des composants ont des distributions exponentielles. Les réseaux de Petri stochastiques sont d'un usage plus récent en sûreté de fonctionnement.

2.2.2.1 Processus Markoviens

Un processus markovien est un processus stochastique sans mémoire, c'est-à-dire que l'état futur ne dépend que de l'état actuel. On le représente généralement par un graphe. La construction d'un graphe de Markov pour l'analyse de la sûreté de fonctionnement consiste à identifier les différents états de marche et de défaillance du système comme des combinaisons des états des composants (défaillants ou non défaillants) et à associer le passage d'un état à un autre à un dysfonctionnement ou à une réparation de composant. A chaque transition, de l'état E_i vers l'état E_j, est associé un taux de transition λ_{ij} (taux de défaillance ou de réparation d'un composant) défini de telle sorte que $\lambda_{ij} \times dt$ est égal à la probabilité de passer de E_i vers E_j entre deux instants très proches t et $t+dt$ sachant que l'on est dans l'état E_i à l'instant t. On peut alors écrire un ensemble d'équations différentielles dont les variables sont les probabilités de séjour dans chacun de ces états. Pour assurer la propriété de Markov, et par la suite résoudre simplement ces équations il faut que les taux λ_{ij} qui apparaissent comme des coefficients de ces équations soient indépendants du temps. Cela signifie que les instants de défaillance ou de réparation de chaque composant doivent être distribués exponentiellement. Malheureusement, l'utilisation des distributions exponentielles n'est pas toujours adéquate pour modéliser les comportements des composants. Pour prendre en compte d'autres types de distribution ou pour prendre en compte des politiques de réparation tenant compte du passé, des extensions ont été proposées (processus semi-Markoviens, méthodes des états fictifs…).
En dépit de leur simplicité conceptuelle et de leur aptitude à pallier certains handicaps des méthodes classiques, les graphes de Markov souffrent de l'explosion du nombre des états car le processus de modélisation implique l'énumération de tous

les états physiquement possibles et de toutes les transitions physiquement possibles entre ces états.

2.2.3 Méthodes de sûreté de fonctionnement dynamiques

Les méthodes statiques présentées précédemment ne sont pas appropriées pour évaluer la fiabilité des systèmes complexes ayant des capacités de reconfiguration ou d'autres types de dépendances temporelles. L'étude de la sûreté de fonctionnement de ces types de systèmes requiert l'utilisation de modèles comportementaux, dans lesquels on modélise explicitement les processus aléatoires qui font évoluer le système d'état en état, jusqu'à ce qu'il atteigne une classe d'états indésirables (défaillance, danger, performance dégradée…).

2.2.3.1 Réseau de Petri

La dynamique d'un système fait que le comportement futur dépend du comportement passé. L'approche Markovienne ne permet pas de prendre en compte cet aspect à cause de l'absence de mémoire. Certaines extensions comme la méthode des états fictifs permet de pallier ces difficultés mais au prix de multiplication du nombre des états qui peut devenir rapidement très important. Donc le problème majeur de la construction d'un graphe de Markov est l'explosion du nombre d'états et le manque de représentativité de certains phénomènes ou mécanismes physiques. Une des réponses à cette problématique dans le cadre de l'évaluation de la sûreté de fonctionnement d'un système complexe est le recours aux réseaux de Petri qui permettent par exemple de prendre en compte la notion de mémoire grâce au concept de marquage.

2.2.3.1.1 Réseau de Petri stochastique

Les réseaux de Petri stochastiques RPS ont été définis par Florin [Florin et al, 1979] pour répondre à des problèmes informatiques liés à la sûreté de fonctionnement. Une transition n'est ainsi plus franchie systématiquement dès sa validation comme dans le réseau de Petri autonome mais on lui associe un taux de franchissement : la probabilité de passer d'un marquage M_i à un autre marquage M_j par franchissement de cette transition entre les instants t et t+dt, sachant qu'à l'instant t, on se trouve dans le marquage M_i. Dans le cas où ces taux sont constants, le graphe des marquages du RdP est homogène à un processus de Markov.
En plus de permettre l'évaluation de paramètres de sûreté de fonctionnement, les réseaux de Petri stochastiques permettent aussi d'évaluer des processus caractéristiques du fonctionnement. Par exemple, on pourra calculer le temps de bon fonctionnement entre deux défaillances, le temps de réparation ou dans certain cas la durée opérationnelle d'une machine, les taux de production, l'évolution des stocks, etc. La modélisation des systèmes nécessite parfois de prendre en compte des transitions non stochastiques (immédiates). Dans ce but une extension des réseaux de

Petri stochastiques a été proposée : les réseaux de Petri stochastiques généralisés RPSG [Ajmone et al, 1984].

2.2.3.1.2 Réseau de Petri coloré

Le réseau de Petri coloré a été défini pour enrichir l'information portée par les jetons [Jensen, 1997]. Dans un réseau de Petri non coloré, la distinction entre les jetons d'une même place est impossible. Dans un réseau de Petri coloré, à chaque jeton est associée une couleur, la couleur peut être un n-uplet et peut donc porter une information complexe telle que par exemple la nature et la position d'un objet dans un stock, ou bien une trame de donnée avec une valeur et une adresse de destination. Chaque transition peut être franchie de différentes manières représentées par les différentes couleurs de franchissement que l'on associe à la transition. Les réseaux de Petri colorés sont largement utilisés dans la modélisation des systèmes complexes : les systèmes mécatroniques [Moncelet, 1998], les systèmes automatiques distribués [Barger, 2003], les protocoles de communication [Poulard et al, 2004]. De nombreux outils mêlent coloration des jetons et transitions stochastiques.

2.2.3.2 Automate à états

Les automates à états font partie des formalismes très utilisés dans la description des systèmes à événements discrets. A tout changement d'état est associé un événement de l'ensemble des événements possibles. Ce formalisme a été étendu sous la forme des automates hybrides pour modéliser correctement les systèmes dynamiques hybrides. Les automates hybrides sont une extension des automates temporisés [Alur et al, 1995]. Informellement, un automate hybride est l'association d'un automate à états finis et d'un ensemble d'équations dynamiques continues pilotées par ce dernier.
Les automates stochastiques associent aux événements des taux de probabilité d'occurrence. Le point faible de ce formalisme est, comme les graphes de Markov (qui sont des automates à états stochastiques), l'explosion combinatoire du nombre d'états du graphe. Pour éviter ce problème, dans le cas de la modélisation des systèmes complexes pouvant être découpés en sous-systèmes, il est possible de construire un modèle d'automate pour chacun d'eux et de les composer ensuite pour élaborer l'automate correspondant au système global. La composition se fait soit par synchronisation entre les automates des différents sous-systèmes, soit par messages, soit par variables partagées. En dehors de l'hypothèse markovienne, il est difficile d'utiliser ce formalisme pour une évaluation analytique des performances. Ils peuvent être utilisés comme support de simulation ou comme moyen systématique de recherche de l'ensemble des séquences d'événements menant le système dans des états particuliers, comme par exemple des états dangereux. On recherche alors le langage reconnu par l'automate vis-à-vis de ces états considérés comme terminaux [Hamidi, 2005].

2.2.3.3 Arbre de défaillance dynamique

Les modèles comportementaux sont bien plus difficiles à construire et à exploiter que les modèles structurels (ou statiques) comme les arbres de défaillances. Pour cela certains travaux ont visé à augmenter le pouvoir de modélisation des arbres de défaillances classiques en ajoutant de nouvelles portes pour prendre en compte des aspects tel que les dépendances entre les composants, les fautes de cause commune, l'ordre d'arrivée des événements [Dugan et al, 1992]. Ce nouveau formalisme appelé arbre de défaillance dynamique permet aux spécialistes des arbres de défaillances de construire des modèles simples susceptibles de donner des résultats quantitatifs et qualitatifs très utiles.
Dans le même but d'extension de l'arbre de défaillance pour supporter la modélisation des systèmes complexes, Marc Bouissou et al ont défini les BDMP (*Boolean logic Driven Markov Process*) [Bouissou et Bon, 2003].

Un BDMP permet de définir un graphe markovien "localement", par la possibilité de calculer tous les états successeurs d'un état donné, avec les taux de transition qui y mènent. Une telle définition se prête à des méthodes de calcul originales, fondées sur l'exploration de séquences dans le graphe des états.
Partant de l'arbre de défaillances, on peut donner le principe simplifié du formalisme des BDMP en disant qu'il remplace :

- les modèles simples de feuilles d'un arbre de défaillances par des Processus de Markov quelconques. Les états de ces processus sont classés en deux catégories. Suivant la catégorie à laquelle appartient l'état d'une feuille à un instant donné, "l'événement" correspondant à cette feuille est considéré comme Vrai ou Faux.
- l'indépendance totale des feuilles d'un arbre de défaillances par des dépendances simples. Chaque feuille a deux modes "sollicité" et "non sollicité", correspondant à deux processus de Markov différents. Le choix du mode dans lequel une feuille se trouve à un instant donné est déterminé par la valeur (Vrai ou Faux) d'un ensemble de feuilles. Les transitions entre ces deux modes définissent éventuellement des états instantanés dans lesquels on peut déclencher des transitions instantanées probabilisées (pour modéliser par exemple des refus de démarrage).

Dans [Bouissou et Dutuit, 2004], les auteurs ont comparé le formalisme de BDMP et les réseaux de Petri pour le calcul de la fiabilité d'un système électrique avec deux modes de fonctionnement. Les résultats sont relativement proches. L'étude a montré que, dans un grand nombre de situations, les BDPM donnent des résultats satisfaisants pour un effort de modélisation très faible, mais pour modéliser des flux de matière ou d'objets, les réseaux de Petri restent le modèle le mieux adapté.

2.2.3.4 Réseaux bayésiens RB

Les réseaux bayésiens sont un formalisme de modélisation probabiliste, ils sont formés d'un graphe orienté sans circuit dans lequel les nœuds représentent les variables d'un système et les arcs représentent les dépendances et les liens entre les

variables. Initialement apparu dans le domaine de l'intelligence artificielle, les réseaux bayésiens sont devenus une méthode de plus en plus utilisée pour la modélisation probabiliste dans différents domaines, notamment dans le domaine de l'analyse de risques, la sûreté de fonctionnement et la maintenance des systèmes [Weber et Jouffre, 2003]. D'après Castillo et al [Castillo et al, 1997], les réseaux bayésiens sont une généralisation des arbres de défaillance, il existe toujours une possibilité de transformer un arbre de défaillance en un réseau bayésien, cependant la réciproque n'est pas vraie. Certains travaux font le lien entre les arbres de défaillances dynamiques et les réseaux bayésiens [Boudali et Dugan, 2005]. Toutefois, il existe très peu de références bibliographiques dans lesquelles nous pouvons trouver une véritable comparaison. Dans [Bobbio et al, 2003], les auteurs présentent les trois méthodes RB, arbre de défaillance et RdPS avec leur application à la sûreté de fonctionnement d'un système de turbine à gaz. Cependant, l'article ne propose pas de comparaison réelle des méthodes. Dans certains cas, il est en effet difficile de modéliser sous la forme d'un RB ce qui est modélisé sous la forme d'un RdPS : par exemple le partage de ressources est un problème difficilement modélisable par un réseau bayésien. L'efficacité de l'une où l'autre des méthodes repose sur le cadre de l'application étudiée.

2.2.3.5 Evaluation des paramètres de la SdF par simulation

Certaines formes de la fonction de structure permettent de trouver une solution analytique pour le calcul des paramètres de la sûreté de fonctionnement notamment lorsque tous les taux de transition sont constants. C'est encore possible dans quelques cas particuliers en dehors de cette hypothèse au prix de modèles mathématiques très complexes [Cocozza et al, 2004].

Dans les cas où des solutions analytiques ne sont pas réalisables, réseaux de Petri et automates d'états sont alors des modèles utilisables pour trouver des solutions à l'évaluation quantitative dans le cadre de la simulation selon la méthode de Monte-Carlo.
D'une manière générale, une simulation de Monte-Carlo consiste à construire un modèle représentatif du comportement du système, dans un environnement donné, et à analyser l'évolution de ce dernier, à partir d'un grand nombre d'expériences ou 'histoires'. A chaque expérience, le comportement du modèle simulé est différent ; ces différences sont explicitement liées à la nature stochastique des processus relatifs à chaque entité du système (durée de réparation, occurrences de défaillances, temps écoulé entre deux défaillances, etc…). Dans le cadre de la sûreté de fonctionnement, l'algorithme de simulation consiste à obtenir dans un premier temps pour chaque expérience réalisée un ensemble de valeurs caractérisant les paramètres de fiabilité, maintenabilité, disponibilité. Dans un second temps, la valeur moyenne de chacun de ces paramètres pour l'ensemble des histoires est estimée. Le nombre d'expériences doit être suffisamment grand pour obtenir des résultats d'un niveau de confiance acceptable.

Lors d'une estimation de la fiabilité, chaque histoire générée par la simulation est arrêtée dès que le système entre dans un état défaillant (état absorbant), un compteur relatif $n_{déf}(t)$ lié à la défiabilité évalue le nombre d'histoires où le système s'est retrouvé dans un état défaillant. La fiabilité est définie par la relation suivante :

$$R(t) = 1 - \frac{n_{déf}(t)}{N} \qquad (2.1)$$

Où $n_{déf}(t)$ et N représentent respectivement le nombre d'histoires ayant évolué vers un état de défaillance avant l'instant t et le nombre total d'histoires simulé.

On peut également estimer le MTTF[3] du système en calculant le temps moyen d'accès aux états de défaillance du système considérés comme absorbants.

La démarche pour évaluer la disponibilité est sensiblement identique mais dans ce cas la simulation de chaque histoire n'est pas arrêtée après une défaillance. Le système peut être réparé, puis évoluer de nouveau vers un état de défaillances. A chaque instant t de la simulation, le compteur $n_f(t)$ estime le nombre de défaillances à l'instant t pour l'ensemble des histoires. La disponibilité est définie par la relation suivante :

$$A(t) = 1 - \frac{n_f(t)}{N} \qquad (2.2)$$

Où $n_f(t)$ et N représentent respectivement le nombre de défaillances à l'instant t sur l'ensemble des histoires et le nombre total d'histoires.
Le nombre N d'histoires simulées détermine bien entendu la précision du résultat. On peut donc être amené à déterminer le nombre N nécessaire pour obtenir une précision donnée. En fait, on ne peut garantir que la probabilité pour que N assure un résultat avec une précision donnée.

L'avantage de la simulation de Monte-Carlo est qu'elle peut facilement générer des solutions à des problèmes industriels complexes, notamment les systèmes non markoviens et évaluer des distributions de probabilité qui ne peuvent pas l'être dans des études analytiques. Par contre, il est évident que, selon le mode de programmation ou les performances à évaluer, la simulation peut être gourmande en ressources informatiques et les temps de calculs peuvent être très longs. Différentes méthodes ont été développées, comme les méthodes de transition forcées depuis le milieu des années 80 qui permettent de réduire le nombre d'histoires à simuler. D'autres techniques se sont basées sur la réduction de temps d'une histoire. Des méthodes pour accélérer la simulation ont été développées dans [Champagnat, 1998].

[3] MTTF : Mean Time To Failure (temps moyen jusqu'à occurrence de la première défaillance)

2.3 Sûreté de fonctionnement des systèmes automatiques

Un système automatique est un ensemble de capteurs, d'actionneurs, de régulateurs et d'autres composants organisés pour accomplir une mission en minimisant les interventions humaines. Dans le mémoire nous considérons le système global appelé également système automatique ou système commandé, comme étant composé d'un système de commande (actionneur, contrôleur, capteur…) et d'un système physique ou processus.

Les principales missions de ces systèmes sont celles des systèmes de commande [Aubry et Zanne, 1991] :

- maintenir des variables (discrètes ou continues) entre des limites proches d'une consigne donnée par l'utilisateur,
- assurer l'aide au diagnostic des défaillances du système,
- assurer les stratégies de tolérance aux fautes implémentées dans l'algorithme de commande.

Les systèmes automatiques peuvent avoir des caractéristiques qui rendent l'étude plus difficile [Moitessier et al, 1991] :

- la cohabitation de phénomènes continus et d'événements discrets (systèmes hybrides),
- l'existence de plusieurs modes de fonctionnement,
- des fréquences d'échantillonnages élevées,
- le besoin d'un retard faible entre la détection d'un événement et l'action associée.

2.3.1 Types de défaillances

Les défaillances peuvent arriver sur les différents composants du système.
Dans ce qui suit sont exposées les défaillances relatives à chaque composant :

2.3.1.1 Défaillance au niveau capteur

- Les défaillances en valeur : les capteurs envoient des données erronées. Par exemple, le capteur de température peut, suite à un défaut de fabrication, présenter un biais constant par rapport à la valeur réelle. Il est également imaginable que la mesure de la température soit momentanément modifiée par une perturbation électromagnétique, dans le cas d'un capteur numérique.

- Les défaillances temporelles : les capteurs envoient les données trop tardivement. Par exemple, le capteur de la sonde lambda ne répond pas

à la sollicitation du système de contrôle-moteur dans le délai de réponse spécifié. Cette défaillance peut se produire à chaque sollicitation, mais souvent elle peut être considérée comme une perturbation transitoire. Ce problème de défaillance temporelle des capteurs est par exemple traité dans [Poledna, 1995a].

- Les défaillances par arrêt : les capteurs n'envoient plus de données (silence) ou une valeur constante (figement). Par exemple, le mauvais branchement du capteur peut entraîner de telles conséquences.

- Les défaillances incohérentes ou défaillances byzantines : dans le cas où le capteur est lié à deux régulateurs, les défaillances de capteur peuvent être perçues différemment par les régulateurs. Il est possible par exemple que le capteur envoie deux valeurs différentes aux deux régulateurs. Les problèmes des défaillances byzantines des capteurs sont exposés dans [Poledna, 1995b].

Il est possible aussi que des combinaisons entre les différents modes de défaillance se présentent. Par exemple, les valeurs d'un capteur peuvent être à la fois erronées en valeur et décalées dans le temps. De nombreuses techniques de tolérances aux fautes sont employées afin d'assurer le fonctionnement correct du système global même en présence de défaillances des capteurs, comme la réplication du capteur ou la relecture du même capteur qui permet surtout de tolérer des fautes transitoires.

2.3.1.2 Défaillance au niveau actionneur

Les actionneurs constituent une autre source de défaillances :

- Les défaillances en valeur : l'actionneur agit de façon trop forte ou trop faible. Par exemple un injecteur débite trop d'essence dans le cylindre.

- Les défaillances temporelles : l'actionneur agit avec un décalage temporel.

- Les défaillances par arrêt : l'actionneur ne fait plus d'action ou une action constante. Par exemple une vanne reste bloquée dans une position ouverte et ne réagit plus à la commande du régulateur.

De façon analogue aux défaillances des capteurs, il est possible que des combinaisons entre les différents modes de défaillances se présentent. Au niveau système global, de nombreuses techniques de tolérances aux fautes sont employées afin d'assurer le fonctionnement correct du système. La technique la plus courante est la réplication de l'actionneur permettant surtout de tolérer des fautes permanentes qui sont dues à des défaillances en valeur ou à des défaillances par arrêt. La réplication de l'actionneur peut être série ('et logique') ou parallèle ('ou logique') ou par vote, dans

ce cas l'action effectuée correspond à la majorité du vote ce qui nécessite au minimum 3 actionneurs.

2.3.1.3 Défaillance au niveau régulateur

Les régulateurs sont généralement des microcontrôleurs qui sont essentiellement composés d'un processeur, d'une mémoire et d'interfaces. De tels composants peuvent présenter une multitude de modes de défaillances qui résultent souvent d'un enchaînement faute → erreur → défaillance relativement complexe qui est difficile à reproduire ou à analyser. Afin de pouvoir maîtriser les défaillances des régulateurs, de nombreuses techniques de sûreté de fonctionnement sont appliquées, [Laprie et al, 1995].

2.3.2 Aspect dynamique

Bouissou et Dutuit [Bouissou et Dutuit, 1996] montrent sur un simple système automatique de maintien de la température les difficultés rencontrées lors d'une évaluation de la sûreté de fonctionnement d'un système automatique:

- ce sont des systèmes séquentiels : c'est-à-dire que l'occurrence de la défaillance (événement redouté) ne dépend pas seulement de l'occurrence des événements mais aussi de leur ordre d'arrivée,
- ce sont des systèmes non cohérents : une faute peut masquer une autre faute et retarder l'apparition de la défaillance globale,
- chaque composant possède plusieurs modes de défaillances donc les défaillances sont non booléennes.

Un des problèmes dans les études de la fiabilité des systèmes industriels complexes est de prendre en compte, d'une façon réaliste, les interactions dynamiques existant entre les paramètres physiques (comme la pression, la température, le débit d'un liquide, etc…) et le comportement nominal ou dysfonctionnel des composants [Dutuit et al, 1997]. C'est bien le cas dans les systèmes automatiques classiques où la défaillance du système dépend des grandeurs physiques.

Par définition, un système hybride est un système qui nécessite dans sa description la prise en compte de sa dynamique continue et de sa dynamique discrète. La dynamique continue est représentée par des variables continues, la dynamique discrète est représentée par les changements d'états dus à l'occurrence d'événements. Ces deux aspects rendent la modélisation hybride indispensable [Siu,1994].

L'étude des systèmes hybrides se fait selon deux approches :

- approche intégrée prenant en compte au sein d'un même formalisme, les aspects continus et discrets ;

- approche séparée qui fait coopérer deux modèles différents : un pour les aspects continus et un pour les aspects discrets.

L'approche intégrée englobe des modèles issus de l'extension de modèles continus comme les Bond Graph à commutation [Buisson, 1993] et ceux issus de l'extension de modèles à événements discrets comme les réseaux de Petri hybrides [David et Alla, 1989]. L'approche séparée regroupe les modèles à base d'Automates hybrides, de Statecharts hybrides, de réseaux de Petri mixtes ou de réseaux de Petri Prédicats-Transitions-Différentiels (RdP PTD) [David et Alla, 1997] [Medjoudj, 2006].

Pour les automaticiens un système dynamique hybride est vu comme un système décrit par deux modèles interagissant : un ensemble d'équations d'états sur des variables continues et un automate à états finis sur un ensemble d'événements discrets.

Pour les fiabilistes, la fiabilité dynamique des systèmes (*dynamic reliability*) ou encore *Probabilistic Dynamics* [Devooght et Smidts, 1992], est une discipline récente dans la sûreté de fonctionnement définie comme [Kermisch et Labeau 2002] :

> *Une partie de sûreté de fonctionnement qui étudie de manière intégrée le comportement des systèmes industriels complexes affectés par une évolution dynamique continue sous-jacente.*

Les méthodes de fiabilité dynamique des systèmes doivent prendre en compte les interactions du système, notamment entre la partie continue et la partie dysfonctionnelle (discrète). Différentes approches traitant le problème de l'évaluation de la fiabilité dynamique sont proposées [Labeau, 1998] [Chabot et al, 1998] [Labeau, 2003] [Pérez et al, 2007] [Cabarbaye et Laulheret, 2005].

On peut remarquer que l'étude de la sûreté de fonctionnement d'un système automatique peut être vu d'une part comme étant un problème dynamique puisque la défaillance d'un composant peut ne pas avoir au cours du temps la même conséquence, et d'autre part comme problème hybride puisque les défaillances du système dépendront de son état continu. Toute approche pour l'étude de la sûreté de fonctionnement de tels systèmes doit donc tenir compte de ces deux aspects.

2.3.3 Travaux concernant la fiabilité des systèmes commandés

Comme cela a été dit précédemment, la fiabilité est définie comme étant l'aptitude d'une entité à exécuter une fonction requise pendant une durée donnée, la cessation de cette aptitude est la défaillance de l'entité. La fiabilité se mesure par la probabilité de non défaillance sur une durée donnée.

La fonction requise est souvent définie de manière binaire : on associe a l'entité une variable binaire état de fonctionnement dont le changement représente la défaillance.

Un système est un ensemble d'entités appelées composants, sa fiabilité est définie par une fonction de structure dépendant des états des différents composants. L'information sur les défaillances des composants est suffisante pour définir la défaillance du système.

Les systèmes commandés (par des boucles de régulation) sont caractérisés par une certaine robustesse aux perturbations. Dans certains cas, cette robustesse peut leur donner un caractère de tolérance aux défaillances des composants. Leur rôle est de ramener une sortie à être proche d'une consigne aussi bien que possible. De ce fait la frontière entre qualité de performance et fiabilité doit être clarifiée.

Dans [Hongbin Li et Qing Zhao, 2005], les auteurs proposent une approche basée sur les coupes minimales et la simulation sous Matlab pour calculer la fiabilité d'un système automatique continu. L'originalité de ce travail est de faire le lien entre les performances de la boucle d'automatique et la définition de la défaillance du système. La fonction requise est définie par un ensemble de performances O à vérifier et non plus par un simple booléen (marche ou panne):

> O= {O_1=stabilité, O_2=erreur statique, O_3=temps de montée, O_4= temps de réponse, O_5=dépassement}

Les défaillances considérées sur les capteurs et les actionneurs sont des défaillances permanentes:

- La sortie est figée sur une valeur constante
- La sortie est déviée par rapport à la valeur initiale

La modélisation dysfonctionnelle se fait en ajoutant des matrices et des gains sur le modèle fonctionnel initial (Matlab).
Par exemple pour prendre en compte les défaillances en valeur du capteur, il suffit d'ajouter à sa sortie un gain pour obtenir l'écart entre la valeur exacte et la valeur déviée.
L'étude peut se résumer par les étapes suivantes :

- prendre toutes les combinaisons de fautes possibles, et associer à chaque combinaison le modèle dysfonctionnel équivalent,
- pour chaque modèle dysfonctionnel tester si l'ensemble des critères est vérifié,
- déterminer toutes les coupes minimales (combinaison entraînant la non satisfaction du critère),
- calcul de la fiabilité à partir de la probabilité des coupes.

L'étude a montré que la fiabilité d'une boucle fermée ne dépend pas seulement des redondances des capteurs et des actionneurs mais également des caractéristiques du régulateur. L'un des points faibles de cette approche est la considération de toutes les combinaisons, c'est une étape qui peut être assez longue et qui consomme beaucoup de temps. Elle peut encore être beaucoup plus lente si on considère que les composants de la boucle sont distribués autour d'un réseau. En effet, le réseau génère

des retards variables, dans ce cas il faut considérer les combinaisons possibles en ajoutant les retards variables. Il est important de noter que toutes les défaillances sur les composants sont permanentes et sont injectées au début de la simulation. Cette hypothèse était prise pour simplifier la recherche des coupes minimale. L'influence d'une défaillance au niveau composant dépend en effet de l'instant de son occurrence. Un autre point négatif est la non prise en compte des fautes transitoires.

Des études ont abordé le cadre des systèmes échantillonnés avec une période d'échantillonnage fixe (l'actionneur reçoit une commande périodiquement tous les T unités de temps). L'étude de Jumel [Jumel et al, 2005] montre sur un exemple simple (régulation d'un liquide dans une cuve) comment lier la probabilité d'apparition d'une erreur de contrôle avec la probabilité de défaillance du système en connaissant la dynamique du système régulé et les algorithmes de contrôle. Le but est de comparer différentes architectures opérationnelles en concurrence lorsque les contraintes de sûreté de fonctionnement sont importantes. Les auteurs proposent de calculer le temps moyen entre deux défaillances du système et la probabilité de défaillance en une heure de fonctionnement. Les fautes considérées sont des fautes qui entraînent des erreurs sur l'information. Le terme erreur sur une information exprime le fait que la valeur de l'information consommée à un instant par une activité n'est pas cohérente avec l'état instantané du système. Par exemple, dans le cas d'un système échantillonné et pour un échantillon donné, une telle défaillance peut provenir d'une défaillance en valeur d'un capteur, ou d'une faute de transmission de l'information (trame altérée) dans le cas des systèmes commandés en réseau. La défaillance de la commande aura lieu si la consigne appliquée par l'actionneur est incohérente avec l'état courant du système. La défaillance de l'information de commande peut alors éloigner le système de son état normal désiré et conduire à la défaillance du système global. Le calcul des métriques de la fiabilité se fait après la modélisation du problème avec une chaîne de Markov. En effet, l'évolution du niveau de la cuve peut être modélisé par une chaîne de Markov, son espace d'état étant discret et toute l'information importante pour l'évolution future de la cuve étant contenue dans la valeur de l'état courant. La règle générale de transition entre états est qu'en cas d'erreurs, le niveau de la cuve est modifié de trois unités (de niveau) alors qu'une consigne correcte le modifie d'une seule unité. La seule défaillance considérée est le débordement de la cuve. Ce système de cuve est relativement simple (intégrateur au sens de l'automatique). Dans [Jumel, 2005], Jumel discute la possibilité de généraliser cette méthode à des systèmes d'ordre supérieur au sens de l'automatique.

Les travaux de Moncelet [Moncelet, 1998] s'inscrivent bien dans la problématique de l'évaluation d'un système de commande. Ces travaux traitent d'une manière qualitative et quantitative de la sûreté de fonctionnement des systèmes mécatroniques [Khalfaoui, 2003]. Ces travaux portent sur la détermination des séquences d'événements redoutés et l'estimation de leurs probabilités d'occurrence par la simulation de Monte-Carlo. La démarche consiste à modéliser qualitativement le système à étudier avec le formalisme des réseaux de Petri colorés [Jensen, 1997] et à générer le graphe des marquages accessibles. Tous les chemins menant vers un état

redouté sont ensuite identifiés et caractérisés en terme d'enchaînements d'actions, une action peut être une défaillance ou bien un changement de mode de fonctionnement d'un composant du système. L'estimation de la probabilité d'occurrence de ces scénarios redoutés se fait par une simulation de Monte-Carlo du modèle quantitatif du système. Le principal obstacle de l'analyse qualitative est l'explosion combinatoire du nombre d'états et l'analyse quantitative souffre des temps de simulation.

Les travaux de thèse de Schoenig [Schoenig, 2004] s'inscrivent dans la lignée de travaux en sûreté de fonctionnement des systèmes mécatroniques. Ces travaux sont orientés vers une méthode graphique et analytique : les graphes de Markov. De plus, la construction d'un tel modèle est facilitée par son analogie avec les formalismes à états transitions (Statecharts, réseaux de Petri). La thèse de Schoenig donne l'exemple d'un système embarqué pour lequel le graphe de Markov illustre les différents modes de fonctionnement existants (modes nominaux, dégradés, états redoutés) et la façon d'y accéder. La complexité d'évaluation des attributs de sûreté de fonctionnement est liée à l'interaction permanente entre le comportement du système et le processus de défaillance. Le principe général de l'approche consiste à coupler les deux méthodes d'analyse classiques : la simulation et les graphes de Markov, afin de contourner les limites intrinsèques à ces méthodes (temps de simulation, explosion combinatoire). L'approche proposée consiste à réaliser une agrégation des états élémentaires d'un graphe de Markov. L'influence du système sur le processus de défaillance est intégrée dans le modèle par le biais de termes de pondération des taux de défaillance. Ceux-ci permettent de prendre explicitement en compte la dynamique interne du système. Ces coefficients sont évalués par une simple simulation permettant ainsi de traiter des systèmes complexes. Ce type d'approche, qualifié de dynamique, permet de traiter des systèmes fortement dépendants du temps (par exemple, en cas d'existence de reconfigurations matérielles ou logicielles).

Le temps maximum pendant lequel le régulateur peut être indisponible avant la défaillance noté encore CSD (voir chapitre 1 §1.6.3) est intégrée dans [Shin et Guo, 1995] pour l'évaluation de la fiabilité dans les systèmes temps-réel (figure 2.2).

Supposons un système composé de n processeurs et que le système tombe en panne si 50% des processeurs sont dans un état de panne, c'est-à-dire si $n/2$ processeurs sont défaillants. Chaque état F_i représente le cas où i processeurs sont en panne. Normalement, si on ne prend pas en compte le CSD du système, le système tombe directement en panne après la défaillance de $n/2$ processeurs. L'état V est ajouté pour prendre en compte le CSD du système, maintenant le système ne tombe pas directement en panne mais il attend un temps D=CSD avant d'entrer dans un mode défaillant. Imaginons qu'il y ait des réparations sur les processeurs, dans ce cas le système peut passer de l'état V vers l'un des état à droite, ce qui n'est pas possible si l'on n'a pas ajouté l'état V.

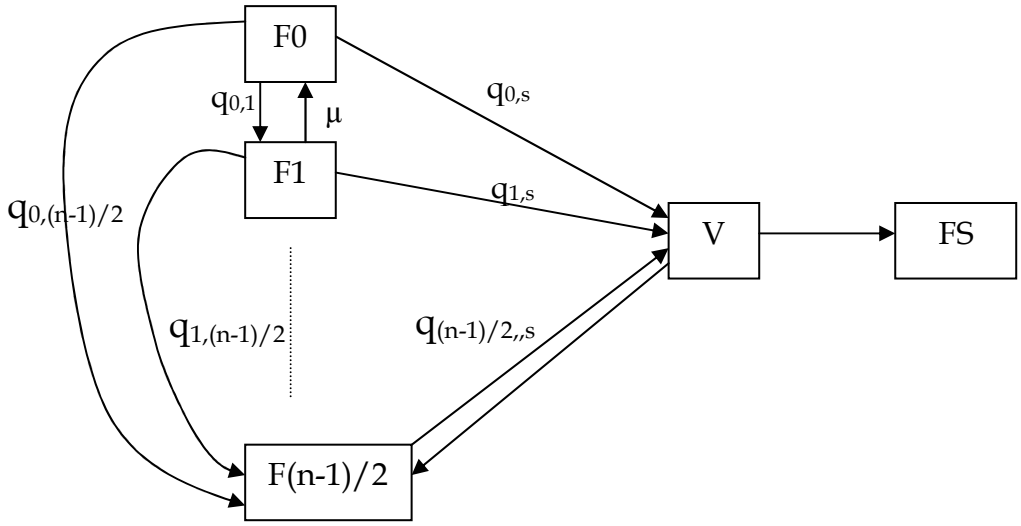


Figure 2.2 : Intégration du CSD dans l'étude de la disponibilité

2.4 Intégration du réseau dans l'étude de la sûreté de fonctionnement

Un réseau est conçu pour respecter une certaine qualité de service dans des conditions données (taille des messages, pourcentage d'utilisation, nombre de stations…). La défaillance du réseau peut être définie par la cessation de l'aptitude du réseau à respecter la qualité de service requise. Dans le cas où plusieurs applications partagent le même réseau, la mission du réseau sera de satisfaire au mieux les contraintes de toutes les applications. Il se peut que le réseau satisfasse un nombre déterminé de contraintes mais pas toutes les contraintes de toutes les applications, dans ce cas le réseau peut être source de fautes pour certaines de ces applications.

2.4.1 Défaillance de la fonction communication

Un réseau est composé de deux types d'éléments, nœud et lien de communication. Chaque nœud est constitué d'un CPU, une mémoire, une interface... les liens de communications interconnectent les différents nœuds. Le réseau a pour rôle de transmettre et surveiller les messages envoyés par les différents nœuds. Les défaillances du réseau dépendent du mode de fonctionnement du réseau. Souvent, dans les applications industrielles, deux modes sont utilisés [Portugal et Carvalho, 2001] :

- Mode redondant (*redundant mode*) : ce modèle est proposé pour prendre en compte la redondance des nœuds. Le réseau possède (*m*) nœuds

redondés et on suppose que la fonction communication se déroule correctement si au moins (*n-m*) nœuds peuvent communiquer.

- Mode dégradé (*degraded mode*) : (*n-m*) nœuds spécifiques sont nécessaires pour le bon fonctionnement, les (*m*) nœuds restants ne sont pas obligatoires pour le fonctionnement du réseau mais leurs défaillances amènent le réseau dans des modes dégradés.

Quand le réseau n'arrive pas à délivrer toutes les demandes de messages à envoyer, le nombre des messages dans les tampons augmente considérablement. De par leur nature les tampons sont limités c'est-à-dire qu'ils ne peuvent contenir qu'un nombre limité de messages, dans ce cas pour libérer les tampons des mécanismes servent à écraser les anciens messages, ou bien des messages de faible priorité, ce qui revient à considérer que ces messages peuvent être perdus.

D'autre part les perturbations sur le réseau peuvent être à l'origine des pertes de messages. Les perturbations comme les perturbations électromagnétiques sont capables d'affecter des messages en transmission. Si le protocole ne prend pas en compte la retransmission des messages affectés, les messages sont perdus. Dans le cas où le protocole permet la retransmission, la perte aura lieu si les perturbations affectent de nouveau les messages retransmis.
La perte de l'information peut être source d'instabilité pour un système. La plupart des études sur la perte de l'information au sein d'un système sont orientées vers l'influence du medium de communication, puisque c'est la zone la plus sensible aux perturbations.

2.4.1.1 Types de fautes

Pour mieux comprendre les différents dysfonctionnements du réseau amenant à des pertes de messages, ou bien à des retards non tolérables, il faut d'abord connaître les fautes qui apparaissent sur le réseau ainsi que leurs sources.
Les modes de défaillances peuvent avoir lieu sur n'importe quelle couche du réseau. Une analyse des modes de défaillance, de leurs effets et de leur criticité (AMDE) sur les différentes couches du réseau est présentée dans [Cauffriez, 2004]. Les effets sur le système sont classés en quatre catégories : information perdue, information erronée (*correctness*), information retardée (*temporally invalid information*) et surcharge du trafic.

Les fautes et leurs sources sont très diverses ; suivant le besoin d'une intervention de réparation ou non, on peut distinguer deux types de fautes :

- Fautes permanentes : ce sont des fautes dues à des défaillances physiques comme la défaillance physique du medium. Une intervention de réparation est nécessaire pour réparer le composant défaillant.

- Fautes transitoires (*transient faults*) : Le réseau est momentanément indisponible, aucune intervention n'est nécessaire. Cela est généralement dû à des perturbations extérieures comme les interférences électromagnétiques. La mobilité dans le cas des réseaux sans fil peut aussi être une source de fautes transitoires : par exemple un engin de type drone commandé via un réseau sans fil qui traverse des zones perturbées (slaloms entre des arbres…).

2.4.1.1.1 Fautes permanentes

L'étude de comportement des réseaux quand certains de leurs composants sont sujets à des défaillances permanentes est traitée par la théorie de la fiabilité des réseaux [Khadiri, 1992]. Dans cette théorie, l'analyste construit un modèle autour du concept de graphe et inclut les informations disponibles sur des propriétés dépendant à la fois des composants (les nœuds et les canaux) et du service que doit fournir le réseau (par exemple, la charge du réseau). L'ensemble de ces données et la structure du réseau conduisent à la définition de mesures de disponibilité et au développement d'algorithmes pour les évaluer. La mesure de disponibilité la plus largement étudiée dans ce contexte est la probabilité, à un instant fixé, que deux sites puissent communiquer entre eux dans un réseau où les centres sont supposés parfaits, et où seuls les canaux sont sujets à des défaillances aléatoires et indépendantes. La probabilité qu'un canal fonctionne est une donnée du problème, appelée disponibilité élémentaire du canal. La probabilité que le réseau fonctionne est la disponibilité source-terminal. Une autre mesure souvent étudiée est la disponibilité tous-terminaux, c'est-à-dire la probabilité que tous les sommets du réseau soient reliés entre eux.

Le problème du calcul de ce type de mesure est appelé la connexité stochastique. Un grand nombre de travaux lui ont été consacrés ; des synthèses peuvent être consultées dans [Locks et Satayanarayana, 1986]. Du point de vue algorithmique, le problème de la fiabilité d'un réseau est un problème NP-difficle [Doulliez et Jarnoulle, 1972]. De ce fait, les méthodes exactes deviennent impraticables pour des réseaux de grande taille.

Cependant, ce type de méthode peut parfois être insuffisante. En effet, elle ignore totalement le travail demandé au réseau. Ainsi, pour un réseau de distribution, on pourra souhaiter prendre en compte la capacité des lignes ainsi que le flux circulant dans le réseau. De même on peut inclure le délai de transmission d'une ligne et s'intéresser au temps de transfert des données. Ce dernier paramètre est important surtout dans des applications temps-réel.

2.4.1.1.2 Fautes transitoires

L'analyse pour ces types de fautes est différente de celle utilisée dans le cas des fautes permanentes. L'étude se fait sur l'influence des fautes fugitives ou bien transitoires sur les messages circulant sur le réseau. Les fautes transitoires sont capables d'altérer des messages en cours de transmission et de changer leur contenu (*Bit Flip*). Si les mécanismes de détection de fautes du réseau ne détectent pas ce changement, le message est reçu erronné par la station réceptrice. Si les mécanismes de détection détectent le changement du contenu du message, le message en cours est ignoré et une nouvelle demande est lancée pour le transmettre à nouveau. Ce temps ajoute un retard supplémentaire sur les transmissions et les messages risquent de manquer leur délai. L'occurrence des fautes dépend du milieu extérieur et est plus difficile à estimer que dans le cas des fautes permanentes où les données sont des probabilités sur les défaillances de canaux. Des travaux tentant de modéliser l'occurrence de ces fautes sont présentés dans le paragraphe suivant.

[Hansson, 2002] présente un modèle simplifié pour modéliser les fautes fugitives en considérant deux sources de fautes :

1. source intermittente : cette source est présente durant toute la durée de la mission T et elle est caractérisée par une période T_i et une durée de perturbation L_i $(L_i<T_i)$. Durant une perturbation tous les messages envoyés sur le medium seront altérés (figure 2.3).

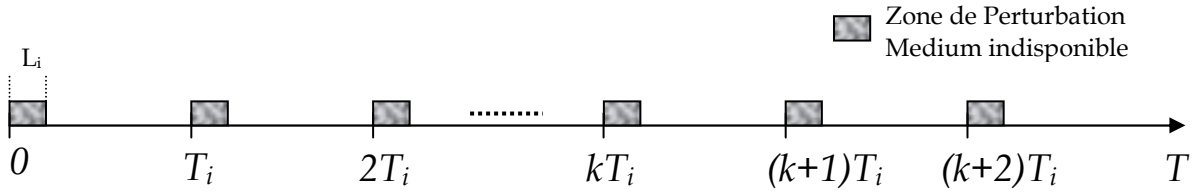


Figure 2.3 : Sources intermittentes

2. Source transitoire : source de durée limitée qui se produit une seule fois durant la mission T du système, elle est caractérisée par une période T_j, une durée de perturbation L_j et un nombre de perturbation n_j, avec $(L_j<T_j$ *et* $n*T_i<T)$ (figure 2.4).

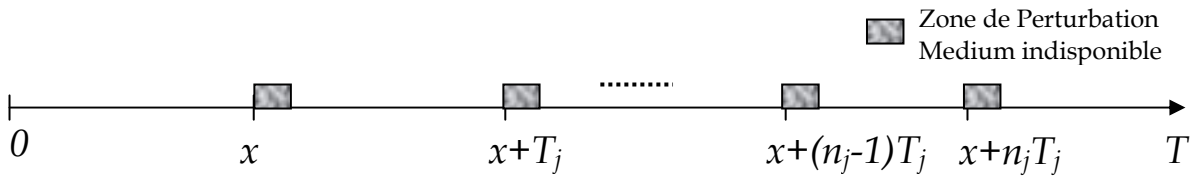


Figure 2.4 : Sources transitoires

Le modèle précédent est un peu simplifié. [Navet et al, 2000] et [Portugal et Carvalho, 2004] ont proposé des modèles probabilistes de fautes dans le but

d'obtenir la probabilité qu'un message ne respecte pas son échéance. Ces modèles permettent de prendre en compte la fréquence d'arrivée des erreurs et leurs durées. La fréquence d'apparition est telle que le nombre d'occurrence suit une loi de Poisson. D'autres auteurs [Buckingham, 1983] [Broster et al, 2004] ont également utilisé la loi de Poisson pour estimer l'influence des fautes transitoires dans les systèmes électroniques.
Dans le modèle de *Portugal*, les fautes suivent une loi de Poisson de taux λ, le but étant toujours de calculer la probabilité pour qu'un paquet soit correctement envoyé i.e non affecté par les fautes extérieures.

La probabilité pour que n fautes arrivent pendant l'intervalle de temps t est :

$$P(t,n) = \frac{(\lambda t)^n}{n!} e^{-\lambda t} \qquad (2.3)$$

La probabilité $P_{TP}(n)$ que n fautes apparaissent durant la transmission du paquet est :

$$P_{TP}(n) = \int_0^{+\infty} P(t,n).dF_{LP}(t) \qquad (2.4)$$

$dF_{LP}(t)$ est la fonction de la répartition de la longueur des paquets.

Un paquet est correctement transmis si :

- aucune faute ne se produit durant la transmission du paquet,
- aucune faute qui précède la transmission n'affecte le paquet (Figure 2.5).

Soit P_{AF_p} la probabilité qu'aucune faute ne se produit **pendant** la transmission d'un paquet :

$$P_{AF_p} = P_{TP}(0) = \int_0^{+\infty} e^{-\lambda t}.dF_{LP}(t) \qquad (2.5)$$

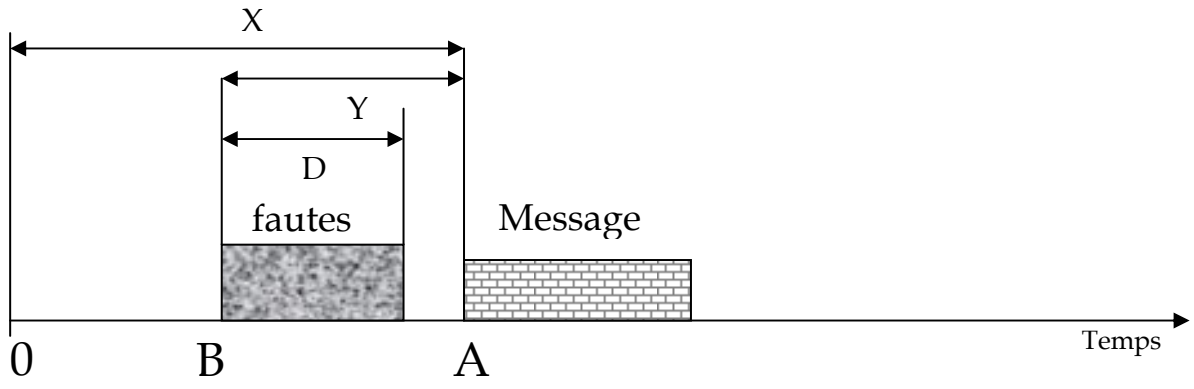


Figure 2.5 : Transmission et occurrence des fautes

Un message transmis à l'instant A est affecté par une faute arrivant à l'instant *B* de durée *D*, si *B<A* et *D>(A-B)*. On suppose que la durée d'une faute est une variable aléatoire de fonction de répartition $F_{DF}(t)$.

La probabilité pour qu'une faute arrivant entre 0 et *X* ait une durée *D* plus grande que *Y* :

$$P_Y = \frac{1}{X}\int_0^Y [1 - F_{DF}(t)].dt \qquad (2.6)$$

La probabilité pour que le paquet ne soit pas affecté par des fautes **précédentes** :

$$P_{NA_{prec}} = \sum_{n=0}^{+\infty} P(X,n)(1-P_Y)^n \qquad (2.7)$$

Finalement la probabilité pour qu'un paquet soit correctement envoyé est égale au produit de deux termes :

$$P_{PS} = P_{AF_P} \times P_{NA_{prec}} \qquad (2.8)$$

Dans le cadre d'une étude de l'influence des fautes transitoires sur la qualité de performance d'un système *Steer-by-Wire*, Wilwert [Wilwert, 2005] propose un modèle d'erreur créé à partir de données réelles. Il considère que pour une perturbation au dessus d'un champ de *100V/m*, une dégradation d'un système *Steer-by-Wire* est tolérée, or sur un système de type *Steer-by-Wire* une simple dégradation de service peut rapidement se transformer en événement catastrophique. La criticité du système amène à adapter l'approche pire cas suivante ; n'ayant aucune preuve du contraire, l'auteur fait l'hypothèse qu'un niveau de champ supérieure à *100V/m* est susceptible de perturber un système *Steer-by-Wire*. En partant des hypothèses prises, la figure 2.6 illustre l'extraction du modèle de faute à partir de mesures prises sur un trajet donné.

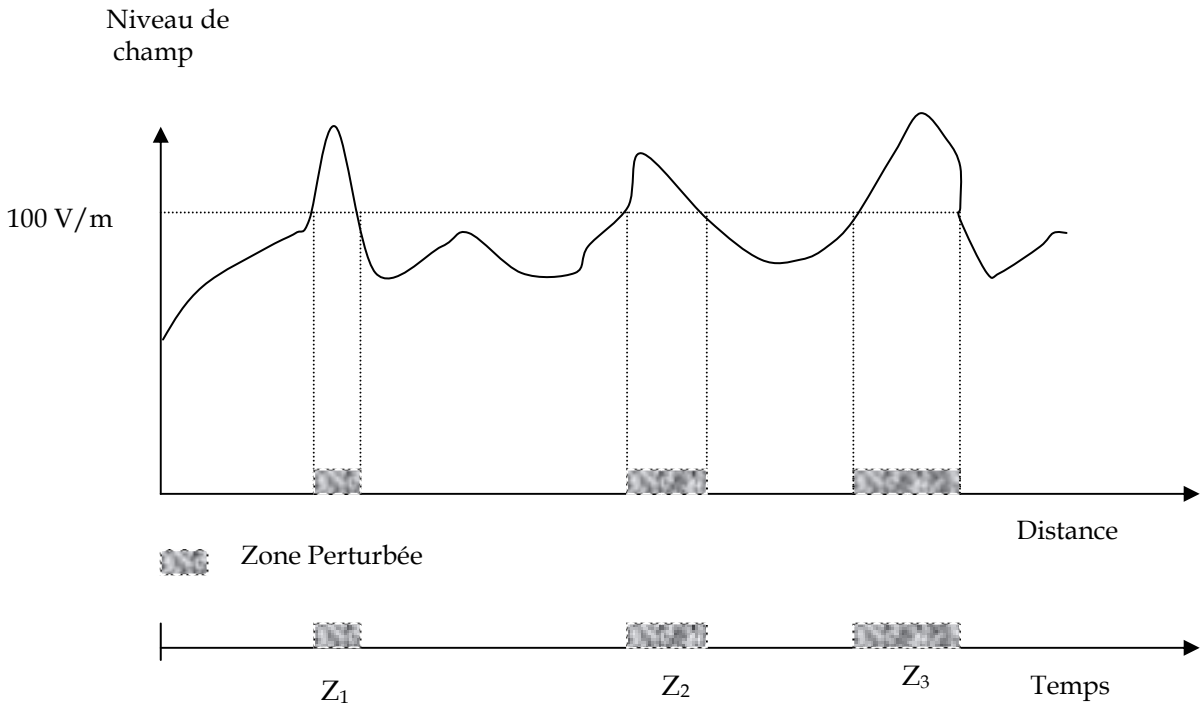


Figure 2.6 : Extraction des zones dont le niveau de champ est supérieure à celui en phase de test et création du modèle de faute

L'intérêt du modèle de Wilwert est qu'il est créé à partir de données réelles.
Dans le cas général, l'estimation de l'arrivée des fautes est une tâche difficilement réalisable [Kim et Shin, 1994]. Comme cela a été précisé précédemment, nous utilisons la loi de Poisson pour la modélisation de l'arrivée des fautes.

2.4.2 Influence des défaillances réseau sur la sûreté de fonctionnement d'un système

Les études précédentes se focalisent sur le réseau lui-même, elles ne considèrent pas l'application ou le système utilisant le réseau.
Lorsqu'on étudie un réseau, il est en effet possible de définir divers modes de fonctionnement :

- réseau perturbé électro-magnétiquement,
- réseau très chargé,
- réseau disponible avec une bonne qualité de service.

Seul le premier de ces modes pourrait être considéré comme défaillant.
Lorsqu'on analyse un système commandé en réseau, le réseau est un élément constitutif du système et il se peut que certains de ces modes de fonctionnement, considérés comme non défaillants (réseau surchargé), entraînent une défaillance au niveau du SCR. Ce paragraphe présente les travaux qui prennent en compte l'influence du réseau sur la sûreté de fonctionnement des systèmes commandés en réseau.

[Conrard, 1999] traite la question de l'influence des composants dits intelligents et des réseaux de terrain sur les critères de la sûreté de fonctionnement. Il signale l'importance d'intégrer ces critères dès la phase de la conception du système. Une relation entre la sûreté de fonctionnement et la rentabilité est développée. La méthode proposée consiste à :

- Modéliser l'architecture fonctionnelle du système,
- L'enrichir par des informations relatives à la sûreté de fonctionnement,
- Etablir une architecture matérielle préliminaire,
- Optimiser l'architecture matérielle.

Cette approche permet de prendre en compte dès la phase de conception les différentes caractéristiques des composants dits intelligents ainsi que les réseaux de terrain, elle permet au concepteur de faire le choix, en proposant des architectures qui minimisent le coût et maximisent les critères relatifs à la sûreté de fonctionnement et ceux de la rentabilité. Le point fort de cette approche est la prise en compte de tous les composants incluant le réseau dès la phase de conception. Le point faible de cette approche est sa limitation aux aspects statiques. Des propriétés comme le vieillissement, les fautes transitoires et les configurations ne peuvent pas être considérées dans l'étude.

Dans la suite de ces travaux, Barger [Barger, 2003] propose une méthode basée sur les réseaux de Petri colorés et la simulation de Monte-Carlo pour répondre à des questions liées à l'aspect dynamique, comme les évolutions du systèmes au cours du temps, son histoire, son vieillissement, mais aussi les divers modes de fonctionnement du système. Dans ces travaux, seules les fautes permanentes sont considérées. Et les fautes sur le réseau sont réduites aux pertes des messages.

Cedric Wilwert [Wilwert, 2005] dans ses travaux de thèse déjà cités auparavant propose une méthode pour évaluer quantitativement l'influence des fautes transitoires dues à l'environnement et des performances temps réel sur la probabilité de défaillance catastrophique d'un système *Steer-by-Wire*. Les fautes considérées sont les fautes au niveau du système de communication. Chaque message est dupliqué n fois et envoyé sur le medium. D'après Wilwert, une faute au niveau système de communication est un événement extérieur au système *Steer-by-Wire* (par exemple une perturbation électro-magnétique) qui peut affecter un message en cours de transmission. L'occurrence de défaillances au niveau communication se produira si un message est affecté par les fautes extérieures. Les défaillances du système de communication peuvent être considérées comme des fautes au niveau du système *Steer-by-Wire*. L'occurrence d'une défaillance *Steer-by-Wire* devient une faute au niveau véhicule. Cette faute peut ne pas conduire à une situation telle qu'elle affecte la sécurité des occupants du véhicule et de son environnement. Intuitivement, il faudra certainement plusieurs fautes consécutives pour que le véhicule ne soit plus sûr. Le nombre maximum de fautes consécutives tel que le système soit toujours sûr est appelé pire retard tolérable par le véhicule. Wilwert propose de l'estimer par simulation à partir d'un modèle Matlab/simulink. Le principal apport de la thèse est

l'utilisation du pire retard tolérable pour l'aide à la conception et à la reconfiguration d'un système soumis à une contrainte de sûreté. Notons qu'il y a des situations non prises en compte par Wilwert, comme par exemple le cas où un petit retard inférieur au pire retard tolérable survient dans un mode de fonctionnement dégradé (secours) ou le cas de la répétition fréquente de retards inférieurs au pire retard tolérable.

La difficulté de modéliser les fautes transitoires et de faire le lien entre les fautes sur le réseau et leur influence sur le système s'ajoute aux difficultés rencontrées avec les systèmes automatiques classiques. Un autre paramètre à prendre en compte quand on utilise le réseau est le retard, qui peut être variable. Bien que le retard soit considéré comme un fonctionnement normal du point de vue du réseau, ce paramètre peut être la cause de la défaillance, vu du système. Donc même si le réseau n'est pas défaillant du point de vue communication, sa qualité de service peut être insuffisante pour l'application à un moment donné à cause d'une perturbation accidentelle ou d'une surcharge momentanée. Pour ces raisons, l'utilisation des méthodes traditionnelles de la sûreté de fonctionnement est très difficile voire impossible. Pour surmonter ces difficultés, on propose une approche basée sur la modélisation et simulation couplée avec l'approche de Monte-Carlo. La modélisation est la première étape de toute étude de sûreté de fonctionnement. Comme cela a été vu, les réseaux de Petri sont un formalisme puissant du point de vue du pouvoir de modélisation. Les réseaux de Petri stochastiques permettent de modéliser les phénomènes aléatoires tel que l'occurrence des fautes transitoires ; les réseaux de Petri colorés ont un pouvoir de modélisation nécessaire surtout pour la représentation des trames circulant sur le medium et la modélisation du système à réguler.

2.5 Choix de l'outil de modélisation

Une fois prise la décision de travailler avec les réseaux de Petri colorés, il reste à choisir l'outil de modélisation et d'analyse avec le potentiel de satisfaire la plupart des besoins spécifiés. Ce choix sera comme souvent un compromis, car l'outil idéal n'existe malheureusement pas.

Plusieurs logiciels permettent l'utilisation des RdP. Plus de quarante logiciels existent actuellement. La plupart des logiciels contiennent des interfaces graphiques, des solveurs markoviens et des modules de simulations.

Après avoir éliminé d'emblée les logiciels peu cités dans la littérature, le choix se fait sur deux ensembles de critères. Le premier ensemble contient des critères imposés. Si le logiciel ne répond pas à un, il est automatiquement éliminé. Le deuxième ensemble est représenté par des critères qui peuvent rendre le travail plus efficace.

Parmi les critères imposés :

- l'existence de jetons colorés,
- la présence d'expressions stochastiques,
- la présence d'un simulateur pour les réseaux de Petri colorés,

- l'existence d'arcs inhibiteurs. Bien que leur utilisation empêche la vérification des propriétés algébriques des réseaux (ce qui n'est pas un besoin dans notre approche), ce besoin est justifié par la simplification possible du modèle : par exemple dans le cas d'une transmission qui aura lieu si et seulement s'il n'y pas de perturbation sur le medium, donc la transition responsable de la transmission sera franchie s'il n'y a pas de perturbations extérieure (jeton dans la place qui représente les perturbations).

Parmi les critères secondaires :

- disponibilité du logiciel,
- coût de son acquisition,
- conception modulaire,
- langage de programmation (on donne une priorité pour les logiciels dont la programmation se fait par des langages de programmation bien connus comme C++, plutôt que les langages propriétaires),
- la documentation, l'évolution et le suivi du logiciel sont encore pris en compte par notre choix.

Finalement, 4 logiciels ont été retenus :

- Möbius tool,
- CPNtools,
- Miss-Rdp,
- GreatSPN.

Le tableau suivant compare les quatre logiciels.

Tableau 2.1 : Comparaison des logiciels

	GreatSPN	Miss-Rdp	CPNtools	Möbius tool
Aspects coloré	intégré	intégré	Spécialement conçu pour l'utilisation des réseaux de Petri colorés (+)	Possibilité de prendre l'aspect coloré en définissant des structures
Aspect stochastique	Intégré	Intégré	Non conçu initialement pour l'utilisation des réseaux de Petri stochastiques, possibilité de prendre en compte l'aspect stochastique en ajoutant du	Intégré

			code ML (-)	
Arc inhibiteur	intégré	intégré	intégré	intégré
Modélisation hiérarchique	possible	En voie d'implantation	possible	Utilisation très facile, par le biais des options *Replicate* et *Join* (+)
Langage de programmation	C	C++	ML (-)	C++
Solveur markovien	Intégré avec possibilité de vérifier des propriétés algébriques (+)	intégré	intégré	intégré
Interface de simulation	Simulation des réseaux de Petri colorés en cours de développement	Simulation des réseaux de Petri colorés en cours de développement	Simulateur puissant intégrant les réseaux de Petri colorés	Simulateur puissant intégrant les réseaux de Petri colorés ainsi que l'utilisation directe de l'approche de Monte-Carlo (+)

Bien que GreatSPN dispose d'un solveur Markovien efficace, son module de simulation pour les réseaux de Petri colorés est en cours de finalisation.

MissRdp est très intéressant, la conception hiérarchique dans ce logiciel est actuellement en voie d'implantation. Les références des travaux utilisant ce logiciel pour des réseaux colorés sont très rares.

Le logiciel Möbius tool est conçu pour l'utilisation d'une famille des réseaux de Petri. L'introduction des structures permet la prise en compte des jetons colorés. Il intègre des modules comme l'interface graphique, un solveur markovien et des modules de simulations. Il est bien structuré et permet la modélisation hiérarchique et l'utilisation des entiers et des réels dans la composition des jetons. La programmation se fait en C++.

CPNtools, n'est pas conçu à l'origine pour modéliser les systèmes stochastiques. Mais avec son grand pouvoir d'expression, il permet d'exprimer plus ou moins des modèles équivalents aux RdP stochastiques.

Les deux derniers logiciels, Möbius tool et CPNtools, répondent bien aux critères imposés et chacun d'entre eux pourrait être utilisé dans nos modèles. Une préférence est donnée pour l'utilisation de *Möbius*, pour les raisons suivantes :

- le temps de calcul,
- la facilité de programmer avec C++ en comparant avec la modélisation en ML dans CPN tools,
- les différentes fonctions qui facilitent la modélisation avec *Möbius* comme les fonctions de *replicate* et *join* qui permettent de combiner des modèles simples en un seul modèle complexe,
- l'organisation du logiciel qui sépare les parties : modélisation, déclaration des variables globales, et déclaration des paramètres à évaluer,
- l'intégration d'une interface permettant l'utilisation de la méthode de Monte-Carlo.

On trouvera en annexe A une étude comparative de Möbius et CPNtools sur un exemple de système commandé en réseau.

2.6 Introduction à la modélisation Möbius/tool

Le but de cette partie est de présenter le logiciel *Möbius tool* et les mécanismes de base qui par la suite permettent de développer des modèles plus complexes et accéder à leur étude. *Möbius tool* est un logiciel dédié à l'étude de la sûreté de fonctionnement et à l'évaluation des performances des systèmes complexes. Il permet l'utilisation des arbres de défaillances, et des réseaux de Petri.

2.6.1 Modèle atomique

Toute étude dans *Möbius* commence par des modèles de base appelés modèles atomiques. Par la suite ces modèles sont regroupés pour former des modèles plus complexes.

Möbius permet la modélisation avec les *Stochastic activity network* SAN [Sanders et Meyer, 2002] qui sont une variante des réseaux de Petri. Möbius représente de manière un peu différente les conditions de franchissement des transitions et le changement de marquage. Ceci se fait par l'intermédiaire de deux nouveaux éléments associés à chaque transition : la porte d'entrée et la porte de sortie.

Chaque modèle atomique dans *Möbius* est formé de places simples, de places étendues (l'équivalent des places colorées), d'activités instantanées, d'activités temporelles, de portes d'entrée et de portes de sorties (*input gate* et *output gate*). Tous ces composants créent la partie statique d'un réseau SAN, les jetons formant la partie dynamique. Chaque composant possède différents attributs.

Places : les places sont de deux types : simples et étendues. Dans *Möbius,* la notion de place étendue est utilisée pour représenter les places colorées. Chaque place simple possède un nom et un marquage initial, elle est graphiquement

représentée comme un cercle bleu. Une place étendue possède un nom et une structure : la structure sera l'équivalent de la couleur. La notion de structure a été ajoutée pour supporter la manipulation des données. Une structure peut être formée par le produit cartésien de différents champs :

$$\text{Structure} = \{\text{int} \times \text{short} \times \text{double} \times \text{char} \times \text{float}\}$$

Exemple d'une structure trame={ID (int), valeur (float), taille (int)}, la signification de cette trame est que la source dont l'adresse *ID* (entier) envoie un message sur le réseau, l'information envoyée a pour valeur *valeur* (float), *taille* (int) est le nombre en bits du message.

Activités : une activité peut être de deux types, instantanée et temporelle. Une activité instantanée possède un nom et un nombre de cas. Si le nombre de cas est plus grand que un, une probabilité est associée à chaque cas. La somme des probabilités de tous les cas doit être égale à un. Une activité temporelle possède tous les attributs d'une activité instantanée plus un champ dédié au temps d'activation de l'activité. Le temps est stochastique et peut être défini suivant plusieurs lois de probabilité (exponentielle, normale, binomiale…). Le temps est associé aux activités, ceci correspond exactement aux RdP T-temporisés dans [David et Alla, 1997].

Portes d'entrée : les portes d'entrée sont représentées graphiquement par un triangle rouge dont le sommet est du côté des places d'entrée et la base est du côté de l'activité. Une porte d'entrée relie une ou plusieurs places à une seule activité. Une porte d'entrée possède un nom et deux fonctions : la fonction de validation qui définit les conditions d'activation d'une activité et une fonction de porte qui permet de modifier le marquage des places en entrée une fois l'activité franchie.

Portes de sortie : les portes de sortie sont représentées par un triangle noir dont la base est du côté de l'activité et le sommet est du côté des places en sortie. Une porte de sortie possède un nom et une fonction de porte qui permet de modifier le marquage des places après le franchissement de l'activité.

Les arcs : les arcs relient les places au portes d'entrée, les portes d'entrée aux activités, les activités aux portes de sorties et les portes de sorties aux places de sortie. On a toujours cet ordre, Place d'entrée → arc → porte d'entrée → arc → activité → arc → porte de sortie → arc → place de sortie.

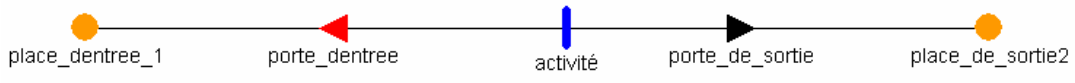


Figure 2.7 : Modèle atomique sous Möbius

2.6.1.1 Stochastic Activity Network

Plusieurs extensions des réseaux de Petri ont été proposées pour augmenter le pouvoir de modélisation et pour répondre à des problèmes spécifiques :
- l'arc inhibiteur, placé entre une place P_i et une transition T_i, ne permet la validation de la transition concernée que si la place P_i ne contient aucune marque,
- les réseaux de Petri temporisés prennent en compte une durée constante entre le début et la fin d'une opération.
Cependant, il existe des phénomènes qui ne peuvent pas être modélisés avec des durées constantes, c'est le cas par exemple du temps de bon fonctionnement entre deux pannes d'une machine. Dans ce cas, on peut utiliser les réseaux Petri stochastiques, qui ont été introduits par Florin et Natkin [Florin et Natkin, 1985]. L'hypothèse la plus couramment utilisée est que les temporisations sont distribuées selon une loi exponentielle.
Quand un réseau stochastique contient des transitions immédiates en plus des temporisations exponentielles, le réseau est un réseau de Petri stochastique généralisé (RPSG) [Ajmone et al, 1984]. Ciardo et al [Ciardo et al, 1991] introduisent quelques extensions sur les RPSG, comme les arcs multiples, les fonctions de validation et les priorités sur les transitions. Même avec les extensions de Ciardo, le réseau peut toujours être converti en une chaîne de Markov. Une extension particulière des RdPSG, les *Stochastic Reward Nets* (SRNs) [Ciardo et al, 1993] a été introduite également par Ciardo et al en exprimant différents aspects du système qui seraient explicitement représentés par des places et des transitions dans le RdPSG, par le biais d'expressions arithmétiques ou logiques du taux de récompenses dans le SRN. Enfin, une autre extension des RdP sont les *Stochastic Activity Networks* (SANs) qui ont été introduits par Meyer et al [Mogavar et Meyer, 1985]. Les SAN sont des SRN possédant une extension pour définir les fonctions de validation des transitions, ainsi que le changement de marquage après le franchissement des marquages. Ceci se fait à travers des portes d'entrées et des portes de sorties. Comme les réseaux de Petri stochastique RdPS étaient définis à partir des réseaux de Petri, les réseaux d'activité stochastiques *SAN* ont été définis à partir des réseaux d'activité *RA*. Dans la suite on donne la définition formelle des réseaux d'activité ainsi que les règles de franchissement des activités.

2.6.1.1.1 Réseau d'activité

Un réseau d'activité est défini par : $RA = (P, A, I, O, \gamma, \tau, \iota, o)$ [Sanders et Meyer, 2002] :

- P est un ensemble fini de places,
- A est un ensemble fini d'activités,
- I est un ensemble fini de portes d'entrée,
- O est un ensemble fini de portes de sortie,

- $\gamma : A \to IN^+$ détermine le nombre de cas pour chaque activité,
- $\tau : A \to$: {*temporelle, instantanée*} spécifie le type de chaque activité,
- $\iota : I \to A$ associe les portes d'entrée aux activités,
- o est une application qui associe chaque porte de sortie à un cas c d'une activité a, $o : O \to \{(a,c) \mid a \in A, c \in \{1,2,...,\gamma(a)\}\}$.

On peut remarquer à partir de la définition que chaque porte d'entrée ou de sortie est associée à une seule activité.

Nous pouvons maintenant préciser les notions de marquage, portes d'entrée et portes de sortie, avant de donner la définition d'un SAN. Soit P l'ensemble de toutes les places du réseau. Si *S* est un ensemble de places tel que $(S \subseteq P)$, un marquage de *S* est une fonction $\mu : S \to IN$. De la même manière, l'ensemble des marquages possibles de *S* est un ensemble de fonctions $M_S = \{\mu | \mu : S \to IN\}$. Une porte d'entrée (*input gate*) peut maintenant être définie comme un triplet *(G, e, f),* où $(G \subseteq P)$ est l'ensemble des places d'entrées de la porte, $e : M_G \to \{0,1\}$définit la condition d'activation de la porte et $f : M_G \to M_G$ est la fonction d'entrée de la porte. De la même façon, une porte de sortie (*output gate*) est définie comme le couple *(G, f),* où $(G \subseteq P)$ est l'ensemble des places de sorties associées à la porte et $f : M_G \to M_G$ constitue la fonction de sortie de la porte. Étant donné un *activity network* et les notations introduites ci-dessus, un *stochastic activity network* est formé en ajoutant les fonctions *C*, *F* et *G* où *C* spécifie la distribution de probabilité de sélection des cas, F représente les fonctions de distribution de probabilité des délais associés aux activités et G décrit les ensembles de "marquages de réactivation" pour chaque marquage possible.

2.6.1.1.2 Réseau d'activité stochastique

Un réseau d'activité stochastique *RAS* est un quintuplet:

$$RAS = (RA, \mu_0, C, F, G)$$

Avec :

1) *RA* est un réseau d'activité $RA = (P, A, I, O, \gamma, \tau, \iota, o)$.

2) $\mu_0 \in M_P$ est le marquage initial et μ_0 est un marquage stable dans lequel *RA* est stable.

3) Pour chaque activité *a* et pour chaque marquage μ dans lequel *a* est validée C_a est une application définie par : $\mu \times \{1,...,\gamma(a)\} \to [0,1]$. Cette fonction sert à décider quel cas choisir parmi tous les cas associés à l'activité.

4) F est une fonction qui affecte à chaque activité une durée de temps appelée durée d'activation : $F : M_P \times IR \rightarrow [0,1]$. F est définie pour chaque activité. Supposons qu'une activité a est valide dans un marquage donné μ, $F_a(\mu,\tau)$ est une variable aléatoire avec une distribution de probabilité qui peut suivre une loi exponentielle ou n'importe quelle autre loi de probabilité. De plus, $F_a(\mu,\tau) = 0$ pour tout $\tau \leq 0$.

5) Etant donné un marquage stable μ et une activité a valide dans μ, la fonction de réactivation est définie de la manière suivante :

$$G_a : \mu \rightarrow \varphi(\mu)$$

$G_a(\mu)$ est un ensemble de marquages appelé marquages de réactivation.

Une activité temporelle représente une opération dans un système modélisé. Le début et la fin de l'opération doivent être bien définis. Dans ce contexte le début d'une opération est signalé par l'activation d'une activité. L'activation d'une activité arrive si :

1) L'activité devient valide figure 2.8.a,
2) L'activité se termine et reste valide figure 2.8.b.

Après la validation d'une activité deux cas se présentent, ou bien l'activité est tirée (activité complétée) comme dans les cas a et b de la figure 2.8, ou bien l'activité est abandonnée comme dans la figure 2.8.c.

Notons qu'à chaque activation d'une transition, la fonction F lui associe une durée de temps appelée temps d'activation. Si l'activité reste valide pendant la durée d'activation, l'activité est tirée, dans le cas contraire l'activité est abandonnée.

Deux autres fonctions sont associées à chaque activité. La fonction C qui détermine le choix d'un cas parmi d'autres une fois l'activité complétée. Cette fonction retourne une probabilité sur les cas. La somme de la probabilité de tous les cas est égale à 1, ces probabilités dépendent des marques des places attribuées à la porte d'entrée de chaque activité.

Une fonction attribuée aux activités est la fonction de réactivation : cette fonction retourne un ensemble de marquages appelé marquages de réactivation. Dans ce cas l'activité est abandonnée et réactivée à nouveau ; ce cas peut se présenter si une autre activité se termine avant le franchissement de la première activité. Cet événement change le marquage du réseau : si le nouveau marquage appartient à l'ensemble des marquages de réactivation, alors l'activité en question est réactivée et une nouvelle durée d'activation est donnée par la fonction F figure 2.8.d

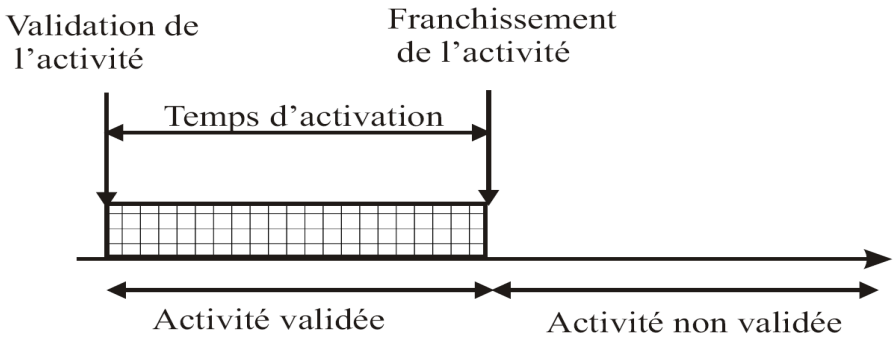


Figure 2.8.a

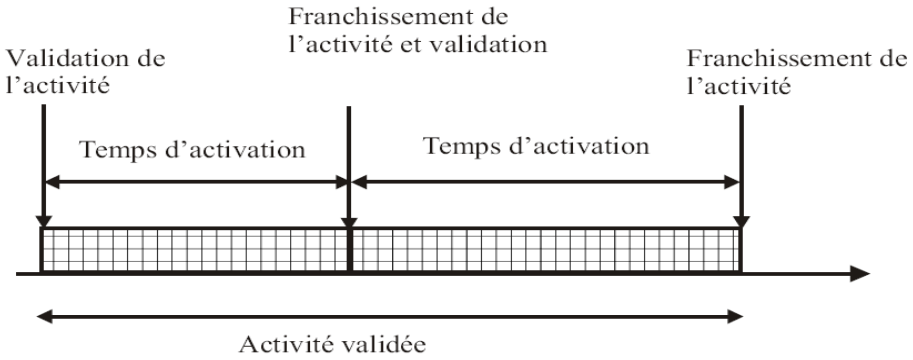


Figure 2.8.b

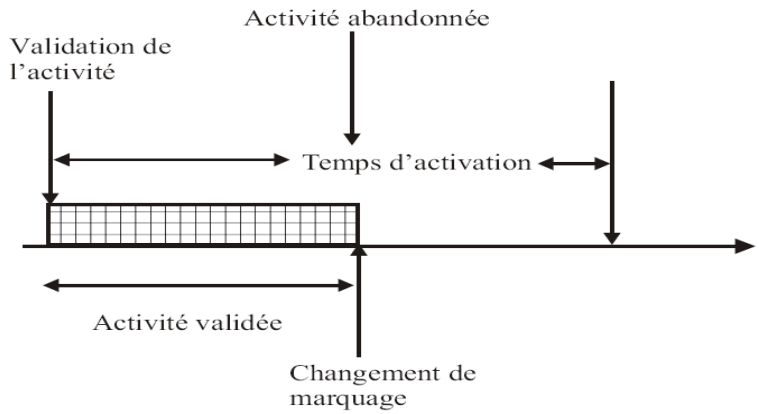


Figure 2.8.c

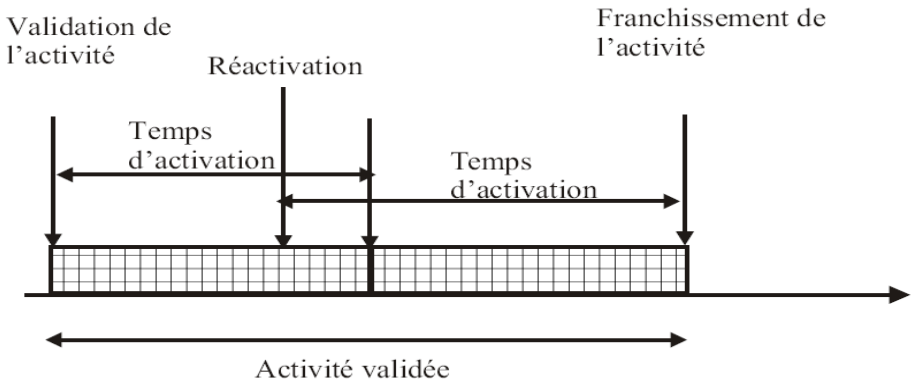


Figure 2.8 : Activation, réactivation, et franchissement des activités temporelles

2.6.2 Modèle composé

Le principe de jonction de plusieurs modèles pour former un modèle plus complexe a été défini pour les *RAS* dans [Sanders et Meyer, 1991]. *Möbius* permet la jonction des modèles atomiques grâce aux fonctions *join* et *replicate*.

Replicate : prenons l'exemple d'un système qui contient n composants identiques, la prise en compte de ces n composants dans la modélisation peut se faire en faisant un modèle représentant le composant et n-1 copies du même composant. Cela est possible en utilisant la fonction *replicate*. La fonction *replicate* possède un seul attribut qui définit le nombre de copies.

Join : la fonction *join* permet la jonction de plusieurs modèles. La seule condition pour joindre deux ou plusieurs modèles est que ces modèles partagent une ou plusieurs places appelées places communes ou places partagées. Les places communes doivent être de même nature i.e contenant la même structure et le même marquage initial. C'est à l'intérieur de la fonction *join* qu'on définit les places partagées.

L'exemple présent illustre un système formé de 5 capteurs, 2 régulateurs, 5 actionneurs (2.9).

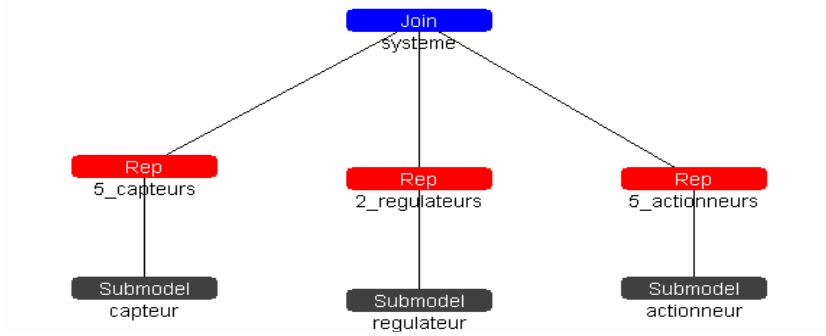


Figure 2.9 : Exemple d'utilisation des fonctions replicate et join

Les modèles formés par l'option *replicate* possèdent tous la même structure, les mêmes jetons et les mêmes variables. Si durant la simulation un jeton est ajouté à une place, un jeton sera ajouté de même sur les autre copies, ceci peut être intéressant si on considère une modélisation dysfonctionnelle et que l'on cherche à évaluer la probabilité de défaillance du système. Dans le cas d'une modélisation fonctionnelle, l'option *replicate* ne peut pas être utilisée.

2.6.3 Fonction de récompense

Cette fonction permet d'évaluer des paramètres définis sur le modèle, comme par exemple le temps de séjour dans une place ou la probabilité de franchissement d'une activité donnée. Deux utilités sont définies dans la fonction *reward*:

Rate reward : cette fonction est utilisée pour évaluer les paramètres dépendant du temps, comme par exemple la durée où une condition a été valide. Les conditions peuvent être des conditions sur les marquages des places. Par exemple si la présence d'un jeton dans une place P1 représente l'indisponibilité d'un composant, alors l'indisponibilité du composant peut être calculée à partir de la fonction *rate reward* suivante :

```
If (P1->Mark ()= =1)
return 1;
```

Ce code retourne la durée pendant laquelle la condition P1->Mark()= =1 était valide.

Impulse reward : cette fonction permet d'évaluer les paramètres dépendant des tirs des activités, comme le nombre de franchissement ou la probabilité de tir des activités.

```
/*transmission*/
return 1;
```

Si on considère que l'activité *transmission* représente une transmission sur le medium, ce code nous permet de calculer le nombre de transmissions effectuées durant une durée déterminée.

2.6.4 Study

Une des grandes facilités offertes par *Möbius* est l'utilisation des variables dans la construction des modèles, ce qui permet une modification des modèles sans revenir à l'intérieur de chaque modèle atomique. Par exemple le marquage initial d'une place peut être déclaré comme une variable, la durée d'une transition peut être déclarée en fonction des paramètres globaux, les probabilités sur les cas peuvent aussi être des variables. Toutes les variables déclarées dans les modèles atomiques sont

automatiquement ajoutées dans un tableau et c'est à cette étape qu'on attribue les valeurs de chaque variable.

2.6.5 Solveur

Möbius offre deux types de solutions pour obtenir des estimations sur les variables recherchées : solutions numériques exactes ou solutions par simulation. En général la solution par simulation peut être utilisée pour tous les modèles conçus dans *Möbius*, tandis que la solution numérique ne peut être utilisée que pour des modèles respectant les conditions suivantes :

1) toutes les activités ont une distribution exponentielle sur la durée d'activation,

2) toutes les activités ont ou bien une distribution exponentielle, ou bien une distribution déterministe sur la durée d'activation. En plus une seule action déterministe peut être activée au même instant.

Dans les deux solutions un nombre de traces peuvent être sauvegardées pour mieux comprendre le comportement du modèle, comme l'instant des franchissements des activités ou le marquage des places avant et après le franchissement.

Il est important de signaler que les solutions par simulation donnent des valeurs approximées et non pas les valeurs exactes des variables recherchées. La précision des valeurs dépend du nombre d'histoires lancées pour estimer les valeurs : c'est le principe même de la méthode de Monte Carlo décrite au paragraphe §2.2.3.5.

En l'absence de solution analytique, ces méthodes semblent être séduisantes par la grande variété des hypothèses qu'elles permettent de prendre en compte (pas de restriction sur les modèles). Cependant, elles présentent un inconvénient majeur, il s'agit du temps de simulation. *Möbius* permet l'utilisation de plusieurs machines pour simuler en parallèle plusieurs histoires et collecter les informations issues de chaque machine pour l'évaluation globale. C'est un atout non négligeable qui réduit énormément le temps de simulation.

2.7 Conclusion

Ce chapitre a présenté les différentes techniques de sûreté de fonctionnement ainsi que les différentes études concernant la sûreté de fonctionnement des systèmes commandés en réseau. Afin de bien préciser l'environnement, nous avons choisi de présenter tout d'abord les méthodes statiques, leurs possibilités, et également les extensions dynamiques qui existent pour certaines d'entre elles.
Après avoir posé la problématique particulière de l'étude des fautes transitoires sur les systèmes commandés en réseau, nous avons orienté notre choix aux approches de

type réseaux de Petri, et plus particulièrement les réseaux d'activités stochastiques (RAS), une variante des réseaux de Petri stochastiques colorés.
Parmi plusieurs logiciels permettant l'utilisation des réseaux de Petri, le choix s'est arrêté sur *Möbius tool*, un logiciel assez puissant permettant la modélisation et la simulation des RAS (*stochastic activity networks*). Ce choix est justifié par la facilité de modélisation, la conception hiérarchique et les fonctions de récompense qui aident énormément à la tâche d'évaluation des paramètres. De plus, *Möbius* est un outil particulièrement adapté aux simulations de Monte Carlo sur la base d'une simulation à événements discrets [Mounnin, 2007]. En effet, *Möbius* propose notamment un choix entre deux générateurs aléatoires et offre la possibilité de donner une séquence d'initiation du générateur, ce qui permet de jouer les mêmes histoires pour comparer des configurations différentes ou de changer l'initialisation et avoir ainsi des histoires différentes. L'ordonnanceur du simulateur évalue les délais associés à toutes les activités valides pour le marquage atteint et l'activité suivante, respectant le principe de causalité, sera réalisée. La simulation permet donc d'estimer l'ensemble des variables de performance sur la base des processus stochastiques définis au niveau des modèles atomiques des constituants.

Chapitre 3 Modélisation et simulation des SCR en vue de l'évaluation de la SdF

3.1 Introduction

Ce chapitre propose une démarche pour la modélisation des systèmes commandés en réseau dans le but d'évaluer l'influence sur les paramètres de la SdF des fautes qui peuvent apparaître dans le réseau d'un tel système.

Comme nous l'avons déjà dit, toute recherche de solution analytique en l'état actuel des connaissances est vaine. Nous recherchons donc un modèle de simulation qui permettra ultérieurement d'accéder aux paramètres de SdF par la méthode de Monte Carlo; le formalisme retenu étant celui des SAN.

La modélisation se fait en quatre étapes :

1) modélisation du réseau de communication : cela intègre le mode d'accès au medium, la transmission, les retards des messages et la réception des messages,

2) modélisation de l'occurrence des fautes sur le réseau : celle-ci consiste à injecter des perturbations sur le medium, permettant d'intégrer la réaction du réseau suite à des perturbations, la retransmission des messages affectés et le cas échéant les pertes des messages,

3) modélisation fonctionnelle des différents composants du système automatique traditionnel comme les capteurs, les régulateurs, les actionneurs et d'autres composants, ces éléments étant considérés comme étant non défaillants ; cette phase intègre l'activation périodique des composants et l'évolution des variables d'état du système automatique,

4) jonction des différents sous-modèles réalisés dans les étapes précédentes pour former un modèle composé représentant le système commandé en réseau ; une attention particulière doit être portée aux places partagées entre plusieurs composants, notamment celles qui relient les différents composants au medium de communication.

.

Le protocole choisi comme support pour notre étude est le protocole CAN, qui est largement utilisé dans le milieu industriel caractérisé par des fortes perturbations extérieures (automobile, train, métro, aéronautique…). Dans cette thèse, il sera le seul protocole utilisé comme support de l'étude. L'utilisation d'un autre protocole ne pose pas de problème particulier. Nous pourrons envisager ultérieurement de comparer l'influence du protocole sur la fiabilité d'un système particulier afin d'établir un critère de choix. Nous présentons maintenant la modélisation détaillée de chaque composant.

3.2 Modélisation du fonctionnement et du dysfonctionnement du réseau CAN

Un facteur important lorsqu'on modélise un système est le niveau de détail à prendre en compte. Un modèle peut contenir toutes les informations possibles et être très fidèle au système réel étudié, mais étant très détaillé il devient très grand et par conséquent difficile à exploiter ensuite pour l'analyse dans un temps raisonnable. Des abstractions sur le modèle peuvent minimiser la taille en gardant une fidélité au système : ceci peut rendre son exploitation plus facile. Le modèle idéal n'existe pas, c'est un compromis entre le degré de fidélité par rapport à la réalité et la possibilité ou facilité d'exploiter ce modèle.

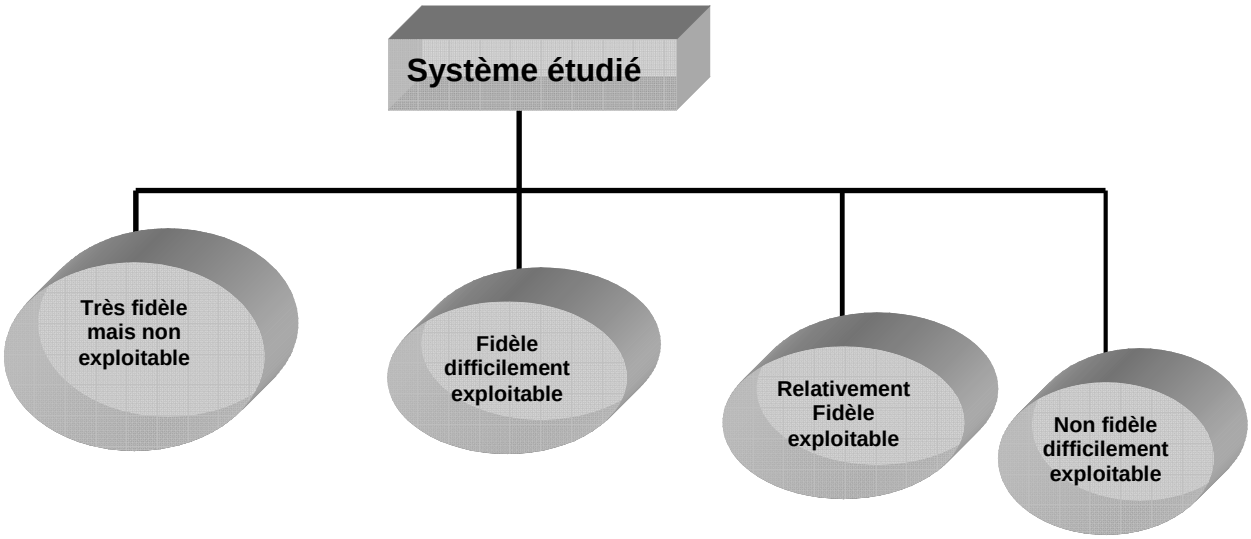


Figure 3.1 : Modélisation des systèmes [Sheldon et al, 2004]

Ce principe s'applique bien entendu à la modélisation des protocoles. Avant de commencer la modélisation d'un protocole réel, il faut préciser quels sont les couches (cf. modèle OSI) et les niveaux de détail que nous souhaitons prendre en compte.

La modélisation la plus fine est celle qui prend en compte toutes les couches, de la couche application jusqu'à la modélisation des valeurs des signaux électriques (tension et/ou courant) sur le câble assurant la liaison.

Vis-à-vis du système commandé en réseau une trame peut être :

- reçue dans les délais,
- reçue avec un grand retard,
- reçue avec modifications (trame altérée),
- perdue.

Il nous semble inutile d'entrer dans des détails comme par exemple le codage des messages et les niveaux des signaux électriques. Pour cela nous avons fait le choix de travailler avec des trames d'une manière symbolique. Dans le modèle SAN d'un protocole, une trame sera représentée par un jeton coloré qui contiendra des informations comme l'adresse du nœud émetteur, la taille du message, et la valeur de l'information envoyée sur le réseau.

3.2.1 Modélisation fonctionnelle du Protocole CAN

En prenant l'hypothèse de travailler avec des trames symboliques, le modèle présenté figure 3.2 représente le fonctionnement d'un réseau CAN (*Controller Area Network*) [CAN, 1990].

Une trame sera représentée par un jeton coloré dans une place étendue. Une trame est formée d'un ensemble d'attributs (id, taille, valeur) *id* représente l'adresse de la station émettrice, *taille* est la taille en bits de la trame, *valeur* représente la valeur à communiquer entre les stations (i.e mesure, commande …).

Sur la figure 3.2 on peut voir qu'à chaque composant lié au medium sont associées deux places étendues (contenant des jetons de type trames) : *componenti_input* et *componenti_buffer*.
Les messages à envoyer sont mis dans les places *componenti_input*. Les places *componenti_buff* représentent les files d'attentes des trois composants qui émettent sur le réseau. Toutes les places étendues de ce modèle sont de type trame avec trois attributs.

L'activité *begin_transmission* permet de commencer une nouvelle transmission. Cette activité est valide s'il y a au moins un message prêt a être envoyé c'est-à-dire un jeton dans l'une des places représentant les files d'attentes.

On rappelle que dans CAN chaque message porte un identificateur (*id*) unique. Grâce à cet identificateur, les stations, qui sont en permanence à l'écoute du réseau, reconnaissent et traitent les messages qui les concernent. En plus l'identificateur détermine la priorité des messages. Le plus petit identificateur représente la plus grande priorité. Notons que l'identificateur n'indique en rien la destination du message, ce qui implique que chaque nœud doit être capable de décider si le message présent sur le bus le concerne ou non.

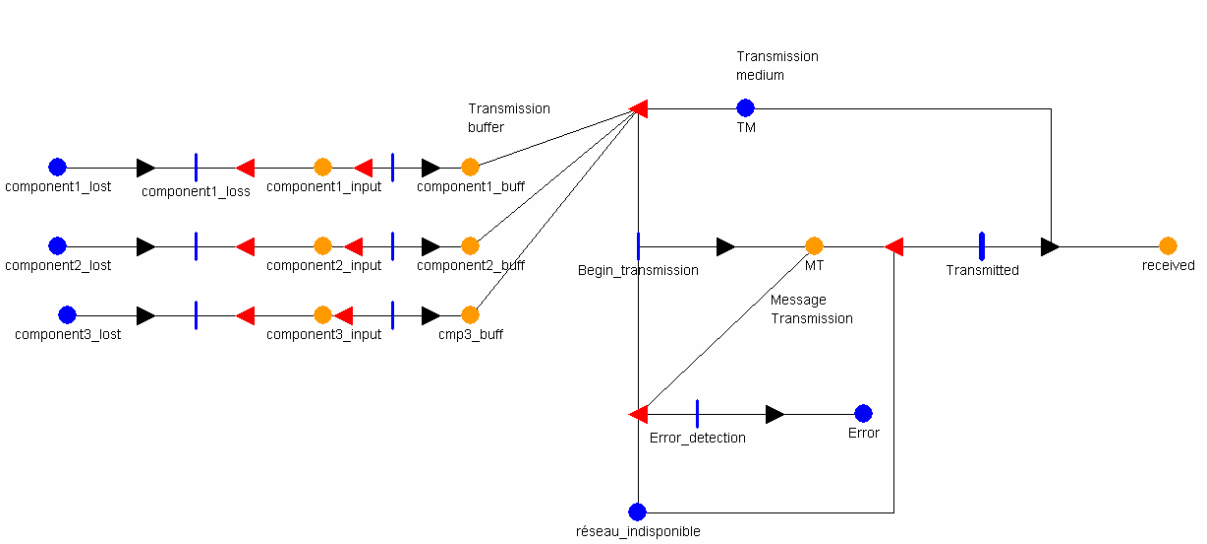


Figure 3.2 : Réseau de communication

La place *TM* représente la disponibilité du medium, la présence d'un jeton dans *TM* indique que le medium est libre. L'absence d'un jeton indique que le medium est occupé.

Quand plusieurs messages sont prêts à être envoyés, l'accès au medium est donné pour le message le plus prioritaire. Le code de la sortie de l'activité *begin_transmission* assure ce mode de fonctionnement.
Dans cet exemple les messages envoyés par la première station sont les plus prioritaires, tandis que ceux de la troisième station sont les moins prioritaires.

Quand un message gagne l'accès au medium i.e. un jeton dans la place *MT* la transmission commence. A cet instant un jeton est enlevé de la place *TM* pour indiquer que le medium n'est plus libre.

La durée de transmission varie selon la taille de chaque message et du débit de transmission, ce dernier paramètre peut être considéré comme paramètre global à définir dans la partie étude (voir chapitre 2 §2.6.4). Après le franchissement de l'activité *transmitted,* le message est mis dans la place *received* et il est prêt à être consommé par les stations intéressées. En même temps un jeton est mis dans la place *TM* pour indiquer que le medium est maintenant libre pour une nouvelle transmission.

La place *réseau_indisponible* est partagée avec le modèle *Perturbation* que nous présenterons ultérieurement. La présence d'un jeton dans cette place indique que le réseau est perturbé et qu'il est indisponible pour la transmission des messages. Si au cours d'une transmission, des perturbations affectent le medium, la transmission est arrêtée et une erreur est détectée. La place *error* est partagée avec la place *error* du modèle *Détection des erreurs.*

3.2.1.1 Validation croisée du fonctionnement du réseau CAN

Cette partie est consacrée à la validation de notre modèle proposé pour le fonctionnement du réseau CAN. *Truetime* est une boîte à outils qui permet la simulation des systèmes commandés en réseau [Anderson et al, 2005]. Une librairie comprenant plusieurs protocoles développés en C est prête à être utilisée. La partie suivante présente une comparaison entre les résultats de notre modèle de fonctionnement du protocole CAN développé en *Möbius*, et le modèle CAN de *Truetime*.

Dans ce paragraphe, un scénario de transmission est donné. On suppose que quatre composants sont liés au réseau (voir figure 3.3).

- Composant1 (périodique) : émet un message m1 périodiquement tous les 0.01*s*,
- Composant2 (événementiel): à la réception du message m1, Composant2 envoie un message m2,
- Composant3 (périodique) : émet un message m3 périodiquement tous les 0.01*s*,
- Composant4 (événementiel): à la réception du message m3, Composant4 envoie un message m2.

Nous nous intéressons à l'observation des instants d'envoi, les messages qui circulent sur le medium et la durée de transmission. La taille des messages circulant sur le medium est fixée à 64 bits, et le débit de transmission à 125kbit/s.

La figure 3.3 montre les actions suivantes pour chaque composant :

- message prêt à être envoyé (indiqué par ↑),
- transmission d'un message sur le medium (niveau haut),
- message bloqué par une transmission en cours (niveau moyen).

Les résultats obtenus par simulation sous *Möbius* correspondent exactement avec ceux obtenus avec *Truetime*.

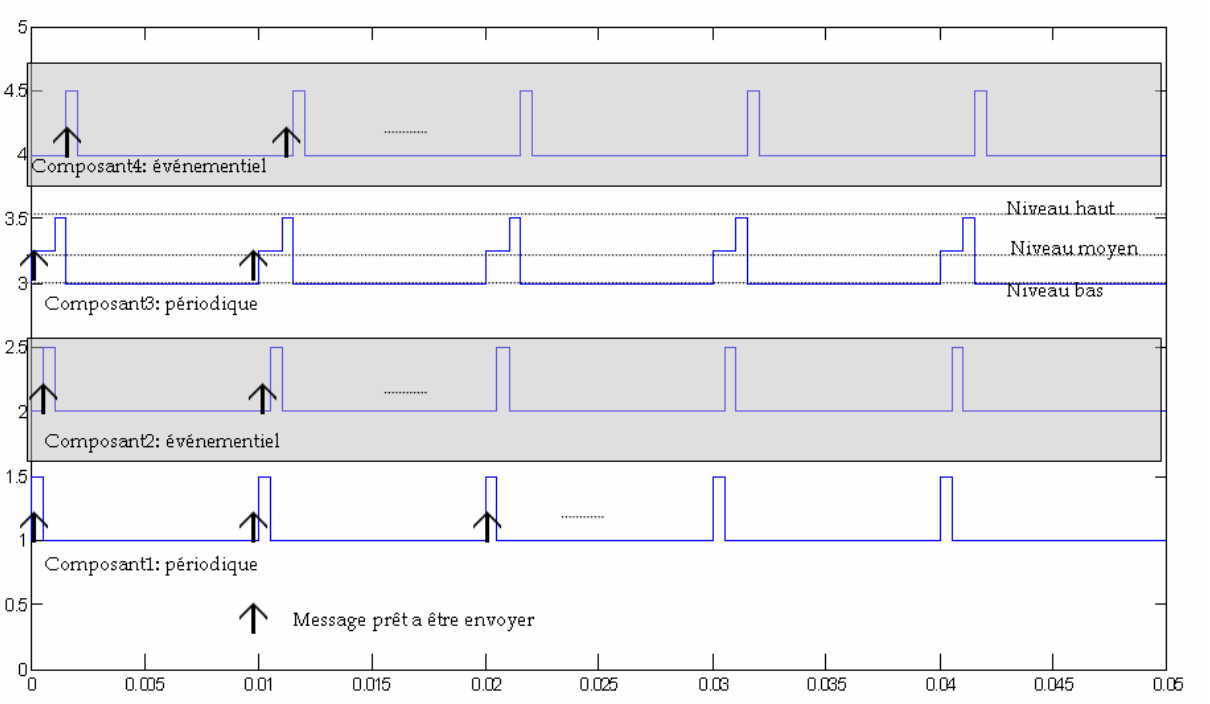


Figure 3.3: Exemple de scénario de communication (aussi bien avec Möbius que Truetime). Avec : niveau haut = envoi du message, niveau moyen = message en attente, niveau bas = inoccupé

Cette partie nous permet de vérifier que le modèle du fonctionnement du réseau CAN que nous avons développé sous *Möbius* est cohérent avec celui disponible dans la boîte à outils *Truetime*. Il ne s'agit pas d'une validation à proprement parler car la comparaison n'est pas effectuée avec un vrai réseau mais comparer ces deux modèles, développés séparément à partir des spécifications, permet d'avoir une certaine idée de leurs pertinences.
La partie suivante est consacrée à l'extension du réseau CAN pour prendre en compte sa réaction en présence des fautes de transmissions.

3.2.2 Modélisation dysfonctionnelle

Un message peut être caractérisé par le triplet (C_m, D_m, S_m) où C_m est le temps de transmission, D_m est la durée avant laquelle le message doit être envoyé qui détermine la date limite d'attente, et S_m est la taille en bits des données à envoyer.
En réalité, le réseau CAN est utilisé dans des environnements fortement pollués où les perturbations risquent d'altérer des messages en cours de transmission. La détection des messages altérés se fait par un ensemble de mécanismes [CAN, 1998] comme :

- le monitoring de bus,
- la détection d'erreurs (CRC : contrôle de redondance cyclique),
- la vérification du format du message,

- le *bit stuffing,*
- l'acquittement.

Comme tous les nœuds du réseau surveillent simultanément le bus, ils détectent des différences entre bits reçus et bits émis. Dès qu'une erreur est détectée, la transmission en cours est interrompue par l'émission d'un indicateur d'erreur ("*error flag*"). Par le biais de son contenu (des bits dominants consécutifs), la trame d'erreur prendra la priorité sur tous les autres messages. A la fin de la trame d'erreur, l'émetteur du message interrompu peut recommencer à émettre son message.

Le recouvrement des erreurs s'effectue obligatoirement par la retransmission automatique du message perturbé. Chaque message possède un nombre limité de retransmission N_{re}. Donc chaque message est maintenant caractérisé par les quatre attributs (C_m, D_m, S_m, N_{re}).
Si le nombre maximal de retransmissions est atteint, la station concernée arrête d'émettre et le message est considéré comme perdu. Un autre cas où le message est perdu se produit lorsqu'un message de même nature (i.e. envoyé par la même station) est prêt à être envoyé, dans ce cas on abandonne l'envoi de l'ancien message et on envoie le nouveau. Cet aspect est pris en compte par les activités *componenti_loss* du modèle du réseau (figure 3.2).

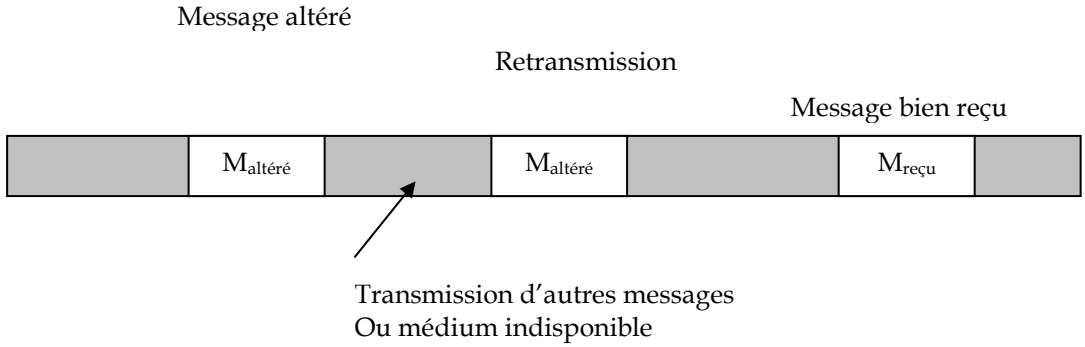


Figure 3.4 : Exemple d'une retransmission d'un message altéré

Comme cela a été justifié auparavant (§ 2.4.1.1), l'arrivée des perturbations est modélisée par un processus de Poisson. Un processus de Poisson de paramètre λ peut également être caractérisé à l'aide du comportement des durées entre événements, c'est-à-dire l'intervalle de temps s'étant écoulé entre deux événements successifs. Les durées ou intervalles de temps sont les variables aléatoires : *d1,d2,...* Il est montré que ces intervalles *d1,d2,…* sont indépendants et sont distribués selon une loi exponentielle de paramètre λ. En terme de modélisation, cela permet de représenter par une transition à taux de franchissement constant (on notera parfois cela transition ou activité exponentielle).

Notre modèle de perturbation est représenté par la figure 3.5, on distingue deux phases : la phase perturbée (jeton dans la place *Phase_perturbée*) et la phase normale (jeton dans la place *Phase_non_perturbée*). Le passage d'une phase à l'autre se fait par le franchissement des activités exponentielles *T1* et *T2*. La permutation des deux représente un changement de l'environnement.
Ceci peut être illustré par deux exemples :

- l'approche d'un radar puissant amène un véhicule équipé d'un réseau embarqué dans une phase perturbée (perturbations électro-magnétiques),
- un robot commandé par un réseau sans fil traversant des obstacles, entraînant des absences de communications (perturbations géographiques).

Une fois que le système est dans la phase perturbée, la transmission sur le medium risque d'être affectée. La présence d'un jeton dans la place *Phase_perturbée* sensibilise l'activité exponentielle *medium_affecté,* son franchissement indique que le medium est affecté par les perturbations extérieures (jeton dans la place *réseau_indisponible*).

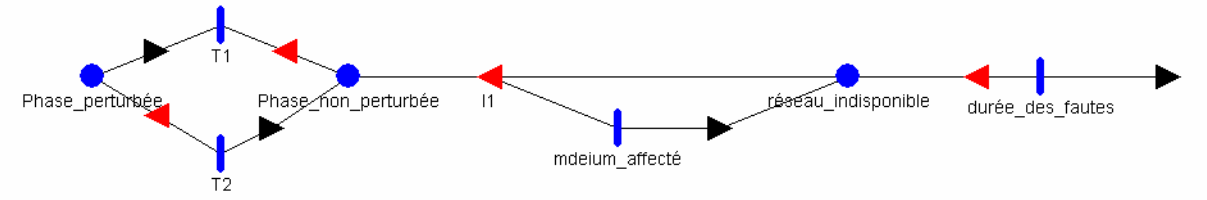


Figure 3.5 : Modèle de perturbation

Le taux associé à cette activité λ_{fa} représente le taux d'arrivée des fautes qui dépend de l'environnement extérieur. La durée de chaque perturbation est modélisée par l'activité déterministe *durée_des_fautes*. Durant l'existence d'un jeton dans la place *réseau_indisponible,* le medium est indisponible et aucune transmission ne peut débuter, un arc inhibiteur lie l'activité *begin_transmission* du modèle *réseau de communication* (Figure 3.2) avec la place partagée *réseau_indisponible (*l'arc inhibiteur est invisible sur le modèle SAN du *réseau de communication,* il est modélisé par un code dans l'entrée de l'activité *begin_transmission*).
Par contre si une transmission est déjà en cours et qu'un jeton est ajouté dans la place *réseau_indisponible,* une faute de transmission aura lieu et une erreur sera signalée (franchissement de l'activité *Error_detection* du modèle *réseau de communication*).

Lors d'un échange, plusieurs types d'erreurs peuvent se produire à différents niveaux.

- Au niveau de la couche physique : ces erreurs sont soit au niveau du bit, soit au niveau du codage du bit du fait que :
  - la valeur du bit lui-même est entâchée d'erreurs (parasite par exemple),
  - il existe une erreur de *bit stuffing* (insertion de bit opposé après une séquence de bits identiques pour aider à la synchronisation) pour des raisons quelconques involontaires (parasites, transmission, oublis).
- Au niveau de la violation de la structure de la trame :

  - nous pouvons avoir affaire à une non conformité structurelle du format de la trame (les informations ne sont pas à leur place) ou à des non cohérences des valeurs présentées dans la trame.
- Au niveau de la validité du contenu de la trame : ce sont les erreurs conventionnelles dues par exemple à des transferts s'effectuant en milieu électro-magnétiquement fortement pollués :
  - la trame n'a pas été acquittée provoquant une erreur,
  - la valeur du CRC[4] ne correspond pas à celle que l'on attendait.

La modélisation proposée ne fait pas la différence entre les différents types d'erreurs. Tout ce qui nous intéresse est de savoir si un message qui circule sur le medium est affecté ou non. On prend l'hypothèse qu'une perturbation affecte tous les messages en cours de transmission et que dans tous les cas, la présence d'erreurs sera signalée par une trame d'erreurs (*error frame*) qui sera générée sur le bus pour en informer qui de droit.
De plus, trois hypothèses ont été prises pour simplifier la modélisation :

- Un message altéré est toujours détecté et signalé directement par les mécanismes de détection d'erreur. Cette hypothèse est justifiée par l'étude de [Unruh, 1989] qui montre que la probabilité d'un message non altéré non détecté est de l'ordre de 10^{-12}.
- Un nouveau message écrase l'ancien par conséquent l'ancien message est perdu. Cette hypothèse est logique, puisque dans notre cas les messages contiennent des sorties et des commandes et il est préférable de prendre tout de suite les informations les plus récentes.
- Le temps de détection dépend du bit altéré, on prend l'hypothèse du pire cas, c'est-à-dire qu'on détecte toujours l'erreur au dernier bit, ce qui entraîne un temps maximum de détection. Un étude complète sur la variation du temps de détection en fonction de la position du bit affecté se trouve dans [Unruh 1989].

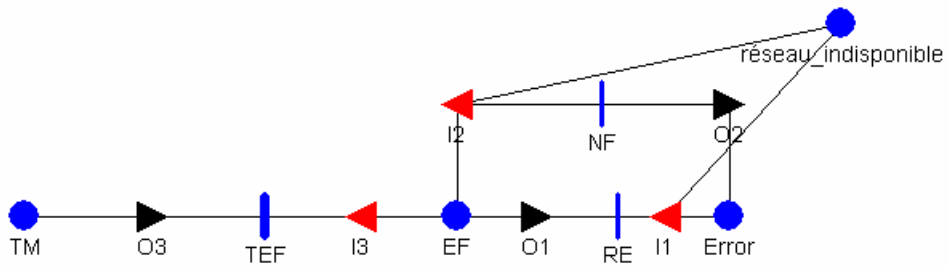


Figure 3.6 : Modèle de détection d'erreur

Ce modèle (Figure 3.6) représente la réaction du réseau CAN aux perturbations. La présence d'un jeton dans la place *Error* signifie qu'une faute de transmission est détectée. Les actions suivantes sont exécutées :

[4] Contrôle de redondance cyclique

- attente jusqu'à la fin de la faute qui a affecté le réseau (*réseau_indisponible->Mark()==0*). La fin de cette action se termine par le franchissement de l'activité *RE (recouvrement de l'erreur)*.
- envoi d'une trame d'erreurs (*Error frame*). A la fin de cette action, le marquage de la place *TM* est remis à un et le medium est à nouveau libre.
- Si durant l'envoi de la trame d'erreurs (présence d'un jeton dans *EF*), une nouvelle faute arrive sur le medium, l'activité *NF* est franchie et le jeton de la place *EF* est déplacé vers la place *Error*.

Le tableau 3.1 détaille les conditions de franchissements et le changement du marquage après le franchissement des activités du modèle *détection d'erreur*.

Tableau 3.1 : Conditions de franchissement et changement de marquage

Gate	type	Enabling predicate	Fonction
I1	entrée	(réseau_indisponible->Mark()==0)&& (Error->Mark()>=1)	;
I2	entrée	(réseau_indisponible->Mark()==1)&& (EF->Mark()==1)	;
I3	entrée	EF->Mark()==1	;
O1	sortie	;	Error->Mark()--; EF->Mark()++;
O2	sortie	;	Error->Mark()++; EF->Mark()- -;
O3	sortie	;	EF->Mark()- -; TM->Mark()=1;

3.3 Modélisation des composants classiques d'un système bouclé

La réalisation des systèmes de commandes numériques a rendu possible l'utilisation des réseaux de communication. Ceci signifie que les algorithmes de commandes sont implantés sous la forme de programmes et de flux de données distribuées sur une architecture de calculateurs et de réseaux gérés par des protocoles. Les systèmes à commander que nous envisageons sont, par nature, continus, alors que leur commande est informatisée et donc discrète. La connaissance des variables de sortie du système régulé (processus) par le système réglant (régulateur) ne se fait qu'à des instants donnés. On parle alors de système échantillonné. L'échantillonnage d'un système consiste à procéder périodiquement à l'acquisition des sorties du processus et à la mise à jour correspondante de ses variables d'entrée (les commandes). Le capteur est l'élément en charge de l'acquisition de l'état du processus, cette

information est envoyée au régulateur qui décide de l'action à produire (commande) pour changer les états du processus. Le régulateur n'agit en général pas directement sur le système, il envoie la commande à un actionneur qui va agir directement sur le système régulé (Figure 3.7).

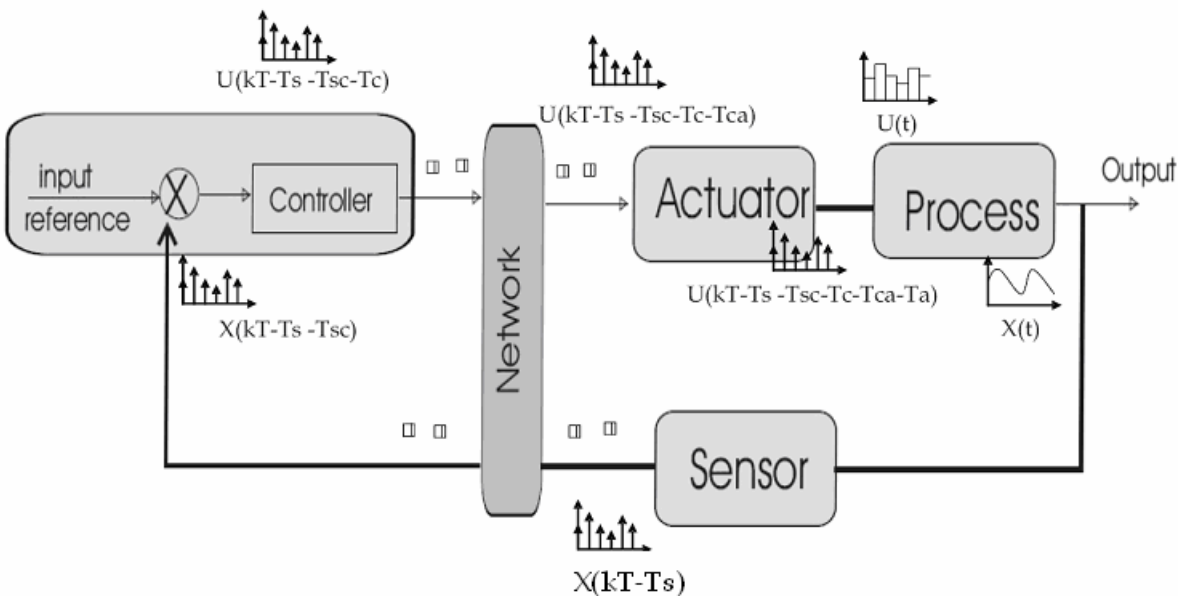


Figure 3.7 : Système commandé en réseau

Le réseau assure la communication entre les différents composants. Dans le cas idéal où l'échantillonnage se fait d'une manière périodique, il serait logique d'utiliser la représentation en z. L'utilisation du réseau ainsi que les ressources partagées (calculateurs exécutant plusieurs tâches) fait que les instants d'acquisition de l'état du système régulé et la mise à jour des commandes ne sont plus périodiques. La partie suivante montre comment on peut modéliser un tel système pour en simuler le comportement, en prenant en compte les retards variables.

Cette partie présente la modélisation des composants classiques d'un système bouclé. La partie suivante détaille les différents composants et la façon de les modéliser sous '*Möbius*'.

3.3.1 Capteur

Le capteur a pour rôle d'envoyer une image de l'état du processus au régulateur. Cela ce fait généralement d'une façon périodique. Le capteur se place à proximité du système à réguler pour permettre de prendre les mesures nécessaires. Dans le cas d'un *SCR*, il est lié au réseau par l'intermédiaire d'une interface de communication lui permettant d'envoyer les informations.

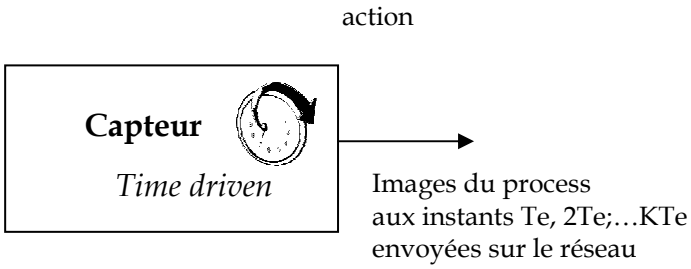


Figure 3.8 : Modèle du fonctionnement d'un capteur périodique

Le capteur est un composant périodique cadencé par une horloge. A chaque période (période d'échantillonnage du capteur *Te*), le capteur mesure les informations concernant le processus et prépare une trame pour l'envoyer au régulateur via le réseau (Figure 3.8).
La préparation des données et l'envoi des informations se fait avec un temps *Ts*, constant et petit que l'on peut négliger.
La périodicité du capteur est assurée par l'activité temporelle à temps fixe *period* (Figure 3.9), elle ajoute périodiquement toutes les *Te* unités de temps un jeton dans la *Place1*. La présence d'un jeton dans la *Place1* valide l'activation instantanée *sensor*. L'activité *sensor* prendra une image de l'état du système et la met dans la place *sensor_output*. Les temps de mesure et de codage de l'information sont supposés être très courts, pour cela l'activité *sensor* est de type instantanée. Cette activité peut être remplacée par une activité temporelle à temps fixe si on veut prendre en compte un petit retard. La place étendue *sensor_output* contient une structure de type *trame*, cette place sera partagée avec le modèle représentant le réseau. La présence d'un jeton dans cette place signifie qu'un message est prêt à être envoyé via le réseau.
La place partagée *process* représente l'état du système. Les équations décrivant l'évolution du processus sont détaillées dans la partie 3.3.4 Processus.

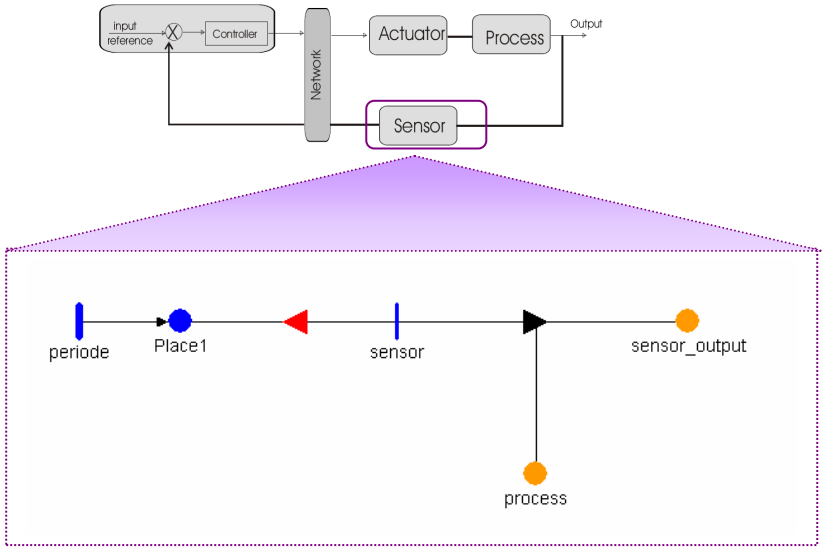


Figure 3.9 : Modèle de capteur

3.3.2 Régulateur

En général les régulateurs sont utilisés pour amener le système à répondre à des critères de performances décrits dans le cahier des charges. Parmi ces critères de performances on peut citer les critères classiques suivants [Aström et Hagglund, 1984], [Smith, 1972] :

- la précision, matérialisée par exemple par une valeur maximale de l'erreur de position : ε < seuil,
- la rapidité, matérialisée en général par une valeur maximale de temps de montée : t_m <seuil,
- la limitation de dépassement : d% <seuil.

Nous considérons un contrôleur PID connu comme étant le contrôleur le plus répandu dans le mode industriel.

Un contrôleur PID fonctionne de cette manière, à l'aide des trois règles correspondant aux trois lettres P-I-D.

- 'P' pour proportionnelle, on cherche donc à appliquer une consigne proportionnelle à l'erreur :

$$S(t) = K_p \times e(t).$$

K_P étant le coefficient de proportionnalité de la composante proportionnelle.

- 'I' comme intégrale, vient s'ajouter à la première composante en évitant cette fois l'accumulation de l'erreur au cours du temps :

$$S(t) = K_p \times e(t) + K_I \int_0^t e(t).dt.$$

Ce type de correcteur s'appelle PI, K_I étant le coefficient de la composante intégrale.

- 'D' pour dérivée permet d'anticiper et d'amortir le mouvement, en prenant en compte la variation de l'erreur au cours du temps :

$$S(t) = K_p \times e(t) + K_I \int_0^t e(t).dt + K_D \frac{de}{dt}$$

Il s'agit ici du correcteur PID, K_D étant le coefficient de proportionnalité de la composante dérivée.

L'équation de transfert d'un PID a donc la forme suivante :

$$H(s) = \frac{K_D s^2 + K_P s + K_I}{s}$$

Si les PID sont implémentés sur des microprocesseurs, des approximations sont faites pour assurer cette implémentation [Ravichandran et al, 2007]. La fonction de transfert est remplacée par un algorithme, appelé algorithme de commande. L'algorithme peut alors prendre la forme suivante :

$$p(i) = K_P(r(i) - y(i))$$
$$I(i) = I(i-1) + b_{i1}e(i) + b_{i2}e(i-1)$$
$$D(i) = a_d D(i-1) - b_d(y(i) - y(i-1))$$
$$\Delta u(k) = u(i) - u(i-1) = \Delta p(i) + \Delta I(i) + \Delta D(i)$$

$$\Delta p(i) = p(i) - p(i-1)$$
$$\Delta I(i) = I(i) - I(i-1)$$
$$\Delta D(i) = D(i) - D(i-1)$$

Avec :
r(i) la valeur de la consigne à l'instant i,
y(i) la sortie du système (sortie du capteur) à l'instant i,
e(i) l'erreur statique (différence entre la consigne et la sortie actuelle e(i)=r(i)-y(i)),
b_{i1}, b_{i2}, a_d et b_d sont des paramètres dépendant de K_I, K_D, des instants i et i-1, et des méthodes d'approximation.

Classiquement, une fonction régulation (sur un API par exemple) est exécutée périodiquement. Elle peut aussi travailler de façon événementielle (sur interruption par exemple). Ce dernier cas est utile pour la synchronisation dans les systèmes en réseau pour éviter les décalages d'horloge entre capteur et régulateur. Nous nous placerons dans ce cas. La réception par le régulateur d'un message en provenance du capteur déclenche l'exécution de son algorithme de régulation afin de générer une nouvelle commande (Figure 3.10).

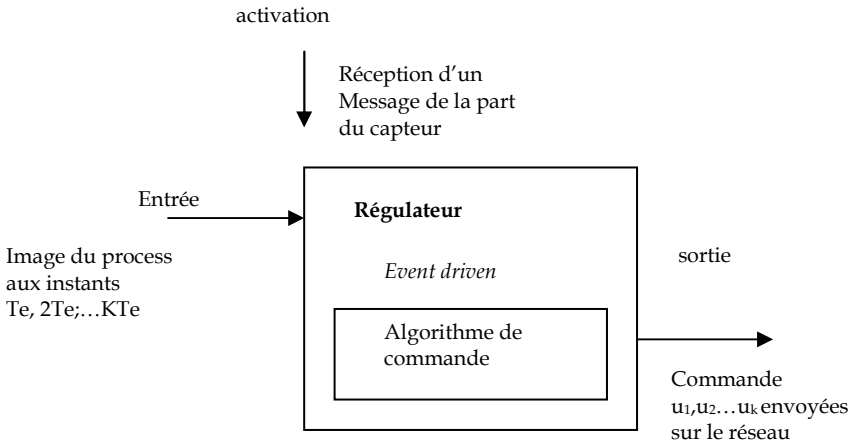


Figure 3.10 : Modèle du fonctionnement du régulateur

Rappelons que dans CAN un message n'est pas dédié à une station déterminée. Le message est vu par toutes les stations liées au medium. C'est aux stations de décider si le message circulant sur le medium les concerne ou non. Dans le cas du régulateur,

il s'intéresse seulement aux messages envoyés par le capteur, cet aspect est pris en compte par l'entrée de l'activité *controller_activity* (figure 3.11). L'activité *controller_activity* sera validée si le message reçu vient de la part du capteur. Dans ce cas, l'activité instantanée *controller_activity* sera franchie. Le franchissement de cette activité déclenche l'algorithme de commande codée dans la sortie *Output1*. La même hypothèse est prise pour le temps de calcul et du codage de l'information. Après le franchissement de l'activité instantanée *controller_activity,* un jeton de type *trame* est mis dans la place *controller_output.* Ce jeton contient trois attributs :

- *id* qui est égal à l'identificateur associé au régulateur, cela permet aux autres stations d'identifier les messages envoyés par le régulateur,
- *taille* qui est égal à la taille du message envoyé, ceci permet de calculer le temps nécessaire pour la propagation du message sur le medium de communication,
- *valeur* est la valeur de la commande.

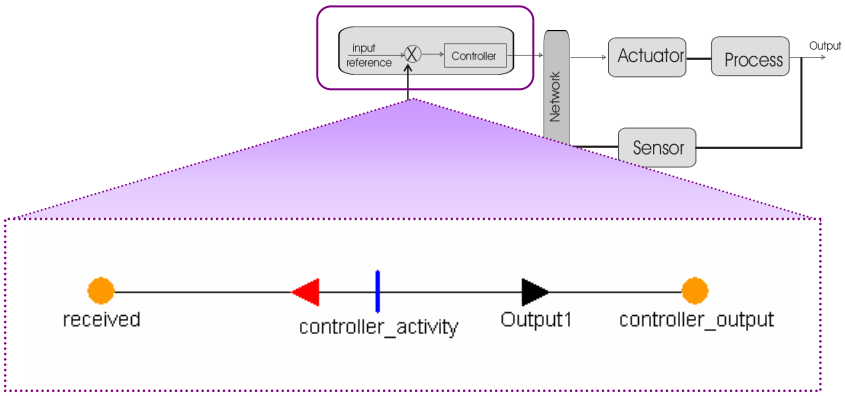


Figure 3.11 : Modèle du régulateur

3.3.3 Actionneur

Le modèle de l'actionneur ressemble au modèle du régulateur. L'actionneur fonctionne d'une façon événementielle. C'est à la réception du message envoyé par le régulateur qu'il va appliquer la commande au processus.

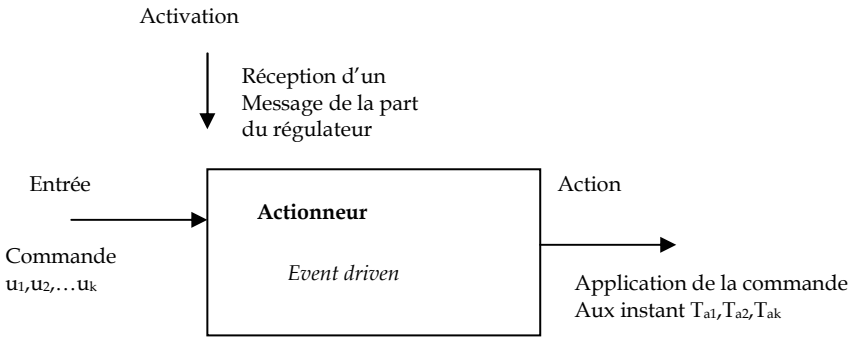


Figure3.12 : Modèle fonctionnel de l'actionneur

Les mêmes remarques sur l'identification des messages, sur les temps de calcul et du codage de l'information s'appliquent aussi à l'actionneur. Pour les besoins du calcul, les instants d'activation de l'actionneur sont sauvegardés dans une place étendue appelée *actuation*, ceci servira pour le calcul de l'évolution du processus dans le temps.

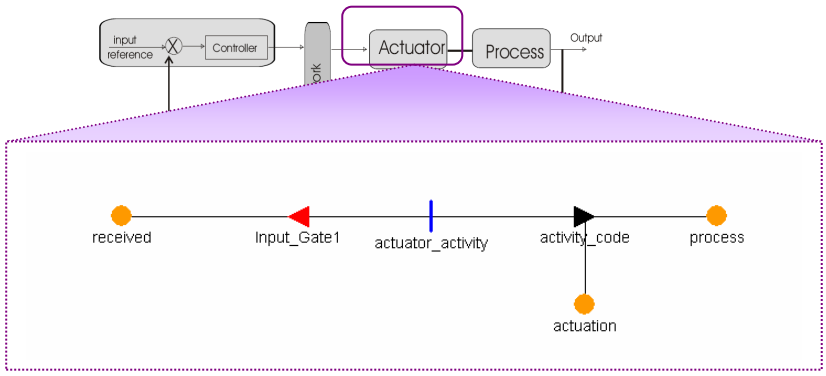


Figure 3.13 : Modèle SAN de l'actionneur

3.3.4 Processus

Comme nous l'avons déjà évoqué, la modélisation en z suppose un échantillonnage périodique strict et donc que toutes les variables ne sont définies qu'aux instants d'échantillonnage. Dans un contexte distribué en réseau, le capteur transmet l'information au régulateur qui enfin transmet la commande à l'actionneur. Le seul composant activé d'une manière périodique est le capteur. Il s'ensuit des délais de transmission qui peuvent être variables. Le régulateur envoie les commandes a_i aux instants Tr_i, l'actionneur les reçoit aux instants Ta_i. Les instants d'activation du régulateur et de l'actionneur ne sont plus périodiques et ne sont pas connus à l'avance. (Figure 3.14).

Dans la modélisation du processus, nous calculons directement la réponse (aux instants d'échantillonnage) à une succession d'échelons représentant les variations de la commande.
Nous supposons que l'actionneur maintient une valeur constante jusqu'à la réception d'une nouvelle commande de la part du régulateur. Le signal d'entrée du processus a la forme représentée en (figure 3.15). Chaque a_i représente une commande envoyée par le régulateur et Ta_i représente l'instant de l'application de la commande a_i sur le processus.

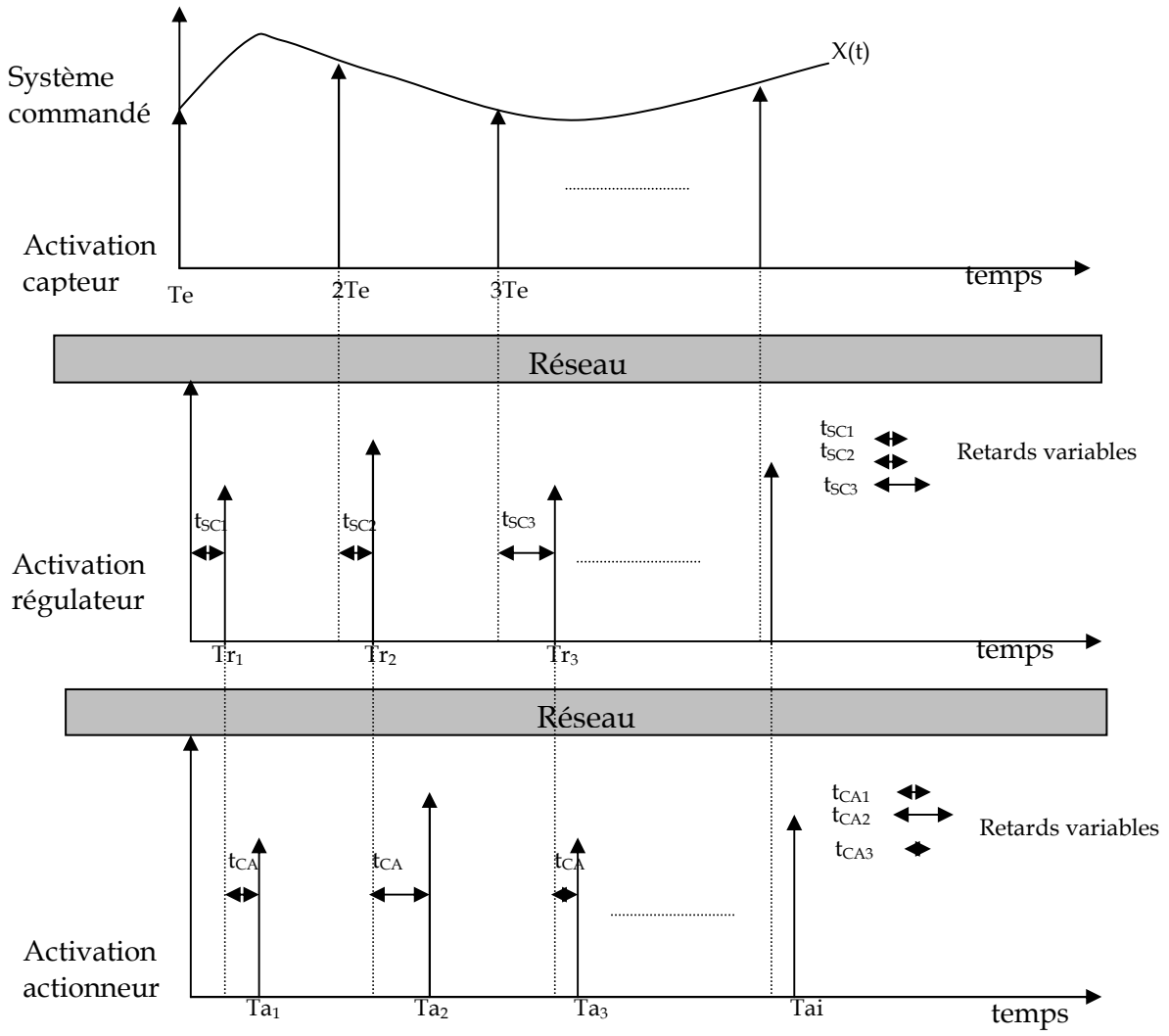


Figure 3.14 : Instants d'activations des composants

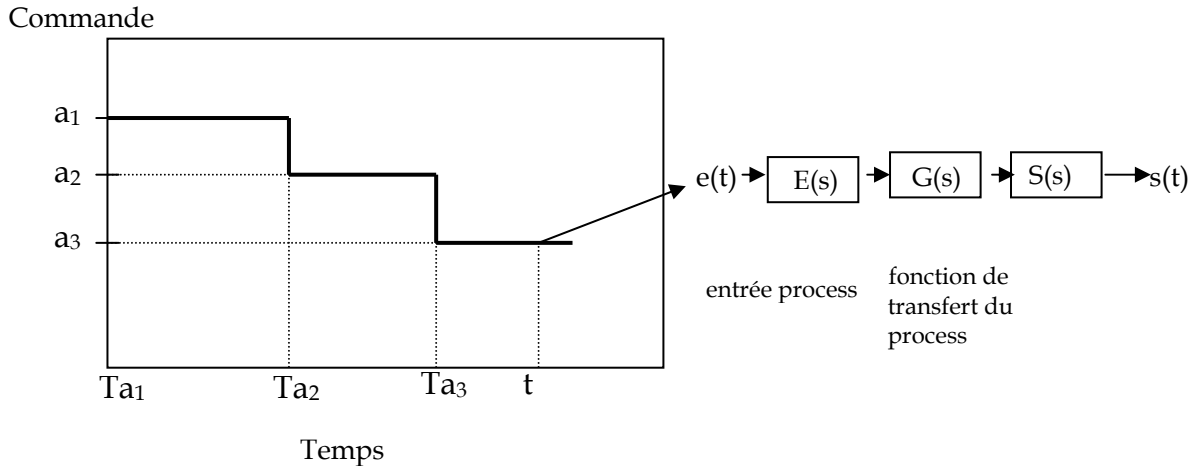


Figure 3.15 : Signal de commande

A un instant donné *t*, le signal à l'entrée du processus peut s'écrire comme une somme d'échelons :

$$e(t)=a_1U(t-T_{a_1})+(a_2-a_1)\times U(t-T_{a_2})+(a_3-a_2)\times U(t-T_{a_3})+... \quad (3.1)$$

En supposant que la commande a_1 est appliquée à l'instant zéro :

$$E(s)=\left(a_1.\frac{1}{s}\right)+\left[(a_2-a_1).\frac{e^{-sTa_2}}{s}\right]+\left[(a_3-a_2).\frac{e^{-sTa_3}}{s}\right]+... \quad (3.2)$$

a_i désigne la valeur de la commande aux instants Tr_i et Ta_i est l'instant de prise en compte de la commande a_i par l'actionneur.

On a :

$$S(s)=E(s)\times G(s) \quad (3.3)$$

La transformée de la sortie est calculée à partir de l'équation (3.3), où E(s) est la transformée de l'entrée et G(s) la fonction de transfert du processus.

G(s)= $\frac{K_1}{(s+x_1)}+\frac{K_2}{(s+x_2)}$

Ce qui donne :

$$S(s)=\left[\left(\frac{a_1}{s}\right)+\left[(a_2-a_1).\frac{e^{-sTa_2}}{s}\right]+\left[(a_3-a_2).\frac{e^{-sTa_3}}{s}\right]+...\right]\times\left[\frac{K_1}{(s+x_1)}+\frac{K_2}{(s+x_2)}\right] \quad (3.4)$$

De cette équation on peut déduire la sortie en fonction du temps :

On pose $\alpha_1=\frac{K_1}{x_1}$ et $\alpha_2=\frac{K_2}{x_2}$

$$s(t)=\begin{array}{l}\alpha_1a_1\left(1-e^{-x_1\times t}\right)+\alpha_1(a_2-a_1)\left(1-e^{-x_1\times(t-Ta_2)}\right)+\alpha_1(a_3-a_2)\left(1-e^{-x_1\times(t-Ta_3)}\right)+... \\ +\alpha_2a_1\left(1-e^{-x_2\times t}\right)+\alpha_2(a_2-a_1)\left(1-e^{-x_2\times(t-Ta_2)}\right)+\alpha_2(a_3-a_2)\left(1-e^{-x_2\times(t-Ta_3)}\right)+...\end{array} \quad (3.5)$$

Rappelons que le capteur mesure les informations périodiquement. Si T_e est la période d'échantillonnage du capteur, on considère qu'à l'instant zéro l'actionneur applique une commande initiale a_1, donc à l'instant T_e :

$$s(T_e)=\alpha_1a_1\left(1-e^{-x_1\times Te}\right)+\alpha_2a_1\left(1-e^{-x_2\times Te}\right) \quad (3.6)$$

On pose : $s_1 = \alpha_1 a_1 \left(1 - e^{-x_1 \times Te}\right)$ et $s_2 = \alpha_2 a_1 \left(1 - e^{-x_2 \times Te}\right)$

A l'instant 2*T_e, il y a deux possibilités :

1) L'actionneur ne reçoit pas de nouvelle commande avant l'instant 2*T_e unité de temps (le retard est supérieur à la période d'échantillonnage), l'équation 3.6 reste valable à conditions de remplacer Te par 2*Te.

2) L'actionneur reçoit une nouvelle commande avant l'instant 2*T_e. Dans ce cas la sortie peut être calculée à partir de l'équation 3.5:

$$s(2*Te) = \begin{matrix} \alpha_1 a_1 \left(1 - e^{-x_1 \times 2*Te}\right) + \alpha_1 (a_2 - a_1)\left(1 - e^{-x_1 \times (2*Te - Ta_2)}\right) \\ + \\ \alpha_2 a_1 \left(1 - e^{-x_2 \times 2*Te}\right) + \alpha_2 (a_2 - a_1)\left(1 - e^{-x_2 \times (2*Te - Ta_2)}\right) \end{matrix} \quad (3.6.1)$$

$$= \begin{matrix} \alpha_1 a_1 - \alpha_1 a_1 e^{-x_1 \times 2 \times Te} + \alpha_1 a_2 - \alpha_1 a_1 - \alpha_1 (a_2 - a_1) e^{-x_1 \times (2*Te - Ta_2)} \\ + \\ \alpha_2 a_1 - \alpha_2 a_1 e^{-x_2 \times 2 \times Te} + \alpha_2 a_2 - \alpha_2 a_1 - \alpha_2 (a_2 - a_1) e^{-x_2 \times (2*Te - Ta_2)} \end{matrix} \quad (3.6.2)$$

$$= \begin{matrix} \cancel{\alpha_1 a_1} - \alpha_1 a_1 e^{-x_1 \times 2 \times Te} + \alpha_1 a_2 - \cancel{\alpha_1 a_1} - \alpha_1 (a_2 - a_1) e^{-x_1 (2*Te - Ta_2)} \\ + \\ \cancel{\alpha_2 a_1} - \alpha_2 a_1 e^{-x_2 \times 2 \times Te} + \alpha_2 a_2 - \cancel{\alpha_2 a_1} - \alpha_2 (a_2 - a_1) e^{-x_2 \times (2*Te - Ta_2)} \end{matrix} \quad (3.6.3)$$

On a :

$$\alpha_1 a_1 e^{-x_1 \times 2 \times Te} = \alpha_1 a_1 e^{-x_1 \times Te}.e^{-x_1 \times Te} \quad , \quad \alpha_2 a_1 e^{-x_1 \times 2 \times Te} = \alpha_2 a_1 e^{-x_2 \times Te}.e^{-x_2 \times Te}$$

et

$$\alpha_1 a_1 e^{-x_1 \times Te} = s_1 - \alpha_1 a_1 \quad , \quad \alpha_2 a_2 e^{-x_2 \times Te} = s_2 - \alpha_2 a_1$$

On remplaçant ces termes dans l'équation 3.6.3 on obtient l'équation 3.7

$$s(2*Te) = \begin{matrix} (s_1 - a_1 \alpha_1) \times e^{-x_1 \times Te} + \alpha_1 a_2 - \alpha_1 (a_2 - a_1) e^{-(x_1 \times (2*Te - Ta_2))} \\ + \\ (s_2 - a_1 \alpha_2) \times e^{-x_2 \times Te} + \alpha_2 a_2 - \alpha_2 (a_2 - a_1) e^{-(x_2 \times (2*Te - Ta_2))} \end{matrix} \quad (3.7)$$

On pose :

$$s_1 = (s_1 - \alpha_1 a_1) e^{-(x_1 \times Te)} + \alpha_1 a_2 - \alpha_1 (a_2 - a_1) e^{-(x_1 \times (2*Te - Ta_2))}$$

et

$$s_2 = (s_2 - a_1\alpha_2)e^{-(x_2 \times Te)} + \alpha_2 a_2 - \alpha_2 (a_2 - a_1)e^{-(x_2 \times (2*Te - Ta_2))}$$

Si on suit le même raisonnement on trouvera que si à l'instant i*T_e l'actionneur reçoit une nouvelle commande entre les instants (i-1)*T_e et i*T_e la sortie vaut:

$$s(i*T_e) = \begin{matrix} (s_1 - \alpha_1 a_{i-1}) \times e^{-x_1 \times Te} + \alpha_1 a_i - \alpha_1 (a_i - a_{i-1}) e^{-(x_1 \times (i*Te - Ta_i))} \\ + \\ (s_2 - \alpha_2 a_{i-1}) \times e^{-x_2 \times Te} + \alpha_2 a_i - \alpha_2 (a_i - a_{i-1}) e^{-(x_2 \times (i*Te - Ta_i))} \end{matrix} \quad (3.8)$$

Ces équations sont implémentées dans le modèle représentant le capteur (sortie de l'activité *sensor*), dans ce cas nous avons jugé inutile d'ajouter un autre modèle pour représenter le processus, puisque généralement un capteur est associé à un seul processus. Si besoin cette séparation peut être facilement appliquée. Notons que ce calcul se fait une seule fois tous les T_e unités de temps (période du capteur), il ne s'agit pas d'un calcul continu de la sortie. Nous calculons la sortie de notre système à des instants donnés en tenant compte des retards variables qui varient d'une période d'échantillonnage à une autre.

3.4 Modèle composé

Comme nous l'avons vu auparavant, le modèle composé regroupe tous les autres sous-modèles pour former un seul modèle représentant le système global. La figure 3.16 montre le modèle composé d'une boucle fermée distribué autour d'un réseau.

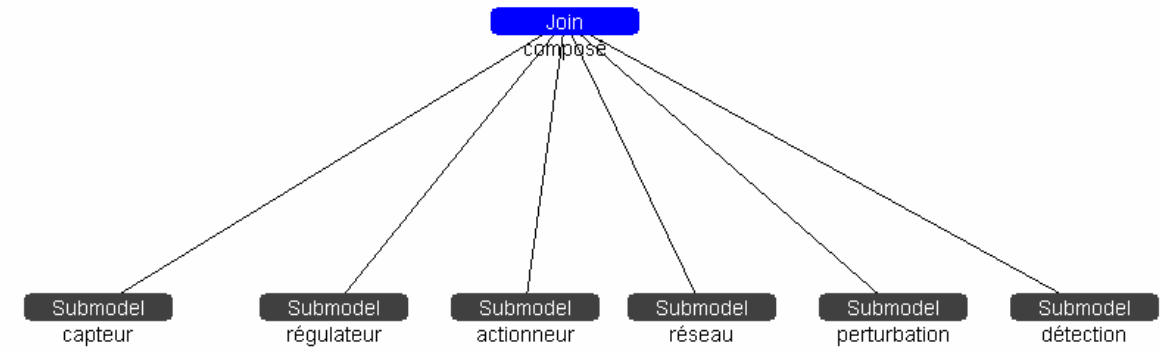


Figure 3.16 : Modèle composé

Ce modèle se compose d'un capteur, d'un contrôleur et d'un actionneur, auxquels s'ajoute le réseau. Le modèle intègre également un modèle de perturbation et un modèle de détection des erreurs.

3.5 Exemple de simulation

Nous nous intéressons maintenant à la simulation du modèle et à l'observation des variables, notamment la sortie du système. L'exemple est tiré de l'ouvrage [Aström et Wittenmark, 1990].
Le système étudié possède la fonction de transfert suivante :

$$G(s) = 1000/s(s+1)$$

Au début nous observons la sortie du système, sans perturbation.
Le contrôleur PD est choisi pour respecter deux critères : un dépassement de moins de 5% de la consigne et un coefficient d'amortissement $\varsigma = 0.7$. La période d'échantillonnage est fixée à 0.01 seconde.

La figure 3.17 montre deux sorties (sans retards et sans perturbations) pour deux consignes, 3.17 (a) la réponse du système pour un échelon et 3.17 (b) pour un signal en créneaux.

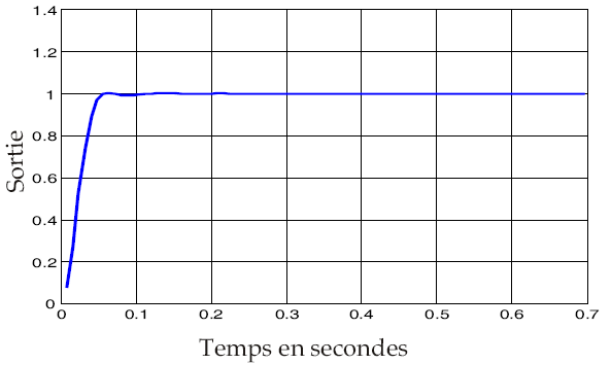


Figure 3.17 (a) réponse à un échelon

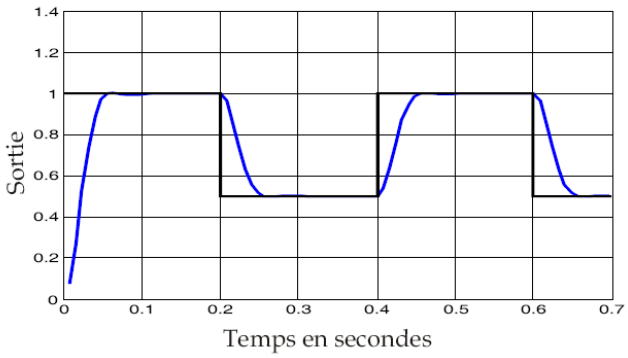


Figure 3.17 (b) réponse à un signal en créneaux

Afin d'illustrer l'influence du réseau de communication dans l'asservissement, nous présentons maintenant les résultats de plusieurs essais.

Influence du retard fixe :

Nous ajoutons un retard fixe t_D sur tous les messages circulant sur le réseau et nous observons la sortie du système.
La figure 3.18 montre qu'un retard trop élevé (c'est-à-dire une faible qualité de service du réseau) entraîne une dégradation significative de l'efficacité de la commande. Nous observons ainsi que la réponse du système est appropriée lorsque le retard est faible, par contre l'apparition d'un retard égal à 40%*T_e provoque une sortie très perturbée (3.18(c)). Le système perd sa stabilité pour un retard fixe égal à 48%*T_e (figure 3.18(d)). Les retards de transmission sur le réseau dégradent la

performance de la dynamique du système et affectent la stabilité du SCR. Les valeurs de retard sont ici propres à l'application et ne sont pas fondamentales.

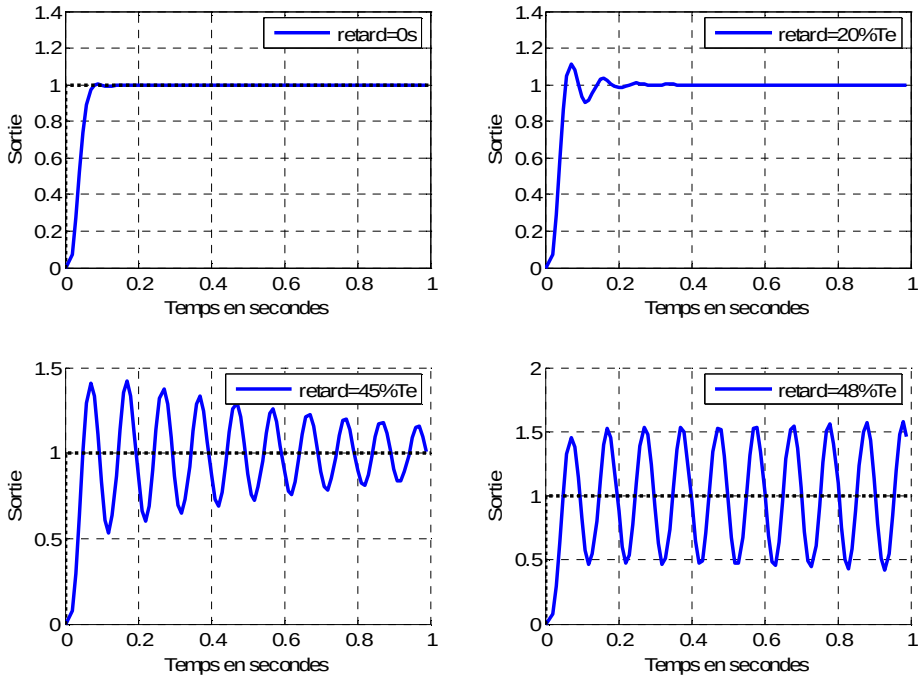


Fig. 3.18: Dépassement en sortie d'un SCR dû aux retards introduits par le réseau (la grandeur en abscisse est le temps en s).

De nombreuses études (systèmes à retard) ont traité de l'influence du retard fixe sur les performances d'un système commandé. Notre souci dans ce paragraphe a été simplement de montrer la capacité de nos modèles à traiter cet aspect. Dans la suite, nous allons focaliser notre étude sur l'influence des retards variables aléatoirement.

3.6 Conclusion

Ce chapitre a présenté un environnement pour la modélisation et la simulation des systèmes commandés en réseau. La modélisation se fait d'une manière hiérarchique. La première phase consiste à construire des modèles atomiques représentant chacun un composant du système. La présence des retards variables interdisant l'utilisation de la transformée en z pour modéliser le comportement de ces systèmes, le calcul de la sortie se fait à des instants donnés à partir de la représentation en s.
La deuxième phase consiste à regrouper les modèles atomiques pour former un modèle composé qui représente un système commandé en réseau. Le réseau utilisé dans cette étude est le réseau CAN largement utilisé industriellement dans des milieux perturbés.

Le but de ces modèles et de former un support de simulation pour l'évaluation de la sûreté de fonctionnement en présence des fautes transitoires. La possibilité d'utiliser des paramètres globaux et la jonction des sous-modèles en des modèles composés, permettent de créer des modèles paramétrés facilement utilisables pour modéliser d'autres systèmes.

Chapitre 4 Influence des fautes transitoires sur la fiabilité des SCR

4.1 Introduction

Comme cela a été étudié au deuxième chapitre, la fiabilité d'un système commandé ne peut pas être considérée comme une simple fonction booléenne sur les défaillances des composants. Nous proposons de la définir comme un ensemble de probabilités de satisfaction de différents critères. Ces critères sont par exemple des seuils sur les paramètres de performance fonctionnelle (dépassement, temps de réponse, stabilité…) comme l'ont proposé [Hongbin et al, 2005]. Ces paramètres caractérisent donc la 'qualité du service' délivré par le système. Pour être capable d'évaluer la fiabilité il faut tout d'abord disposer d'un moyen permettant d'évaluer l'influence sur ces paramètres, de retards variables et/ou de pertes dans les messages. Dans des cas particuliers des systèmes commandés en réseau (retard fixe ou petite perturbation sur le retard), il est possible de calculer analytiquement les paramètres de performances. Dans le cas général où le retard est variable (et surtout dans des proportions importantes), il est très difficile d'évaluer ces paramètres d'une manière analytique. Si on dispose d'un modèle adéquat, la simulation peut répondre à cette question. Dans ce but, au chapitre trois, nous avons proposé un environnement de simulation des systèmes commandés en réseau. Ce chapitre montre comment l'exploitation des modèles peut permettre l'évaluation de la fiabilité.

4.2 Influences des fautes

Le but de notre étude est d'analyser l'influence des fautes transitoires dans le réseau et leur conséquence sur la fiabilité du SCR. Les fautes permanentes ne sont pas prises en compte dans cette étude. En effet, une coupure permanente d'un réseau veut dire que tous les composants sont isolés et cela entraîne dans de brefs délais la défaillance du système. Pour cette raison, il nous semble judicieux de mettre l'accent sur l'influence des fautes transitoires. Il est important de préciser que ces fautes transitoires sont étudiées vis-à-vis du système commandé et non pas du réseau, en d'autres termes, on appelle faute sur le réseau tout événement capable d'entraîner la défaillance du SCR. Ces fautes peuvent être classées en trois catégories :

- retard variable : les retards peuvent être constants si tout le trafic circulant sur le réseau est connu à l'avance et si le protocole gérant la circulation des

messages est déterministe (chapitre 1 §1.4). Sans cette hypothèse, les retards sont généralement variables. Même dans le cas des protocoles déterministes et du trafic constant, les retards peuvent être variables suite à l'occurrence de fautes transitoires sur le réseau qui sont généralement de nature aléatoire (chapitre 2 §4.1.1.). Une autre source potentielle de retard variable est le partage de ressources, par exemple si l'algorithme du régulateur est implanté sur une machine qui exécute d'autres tâches dont l'arrivée et la durée d'exécution varient d'une façon aléatoire et dont la priorité est supérieure par exemple.

- Pertes des messages : cette perte peut être le résultat d'un rejet de paquets lorsque les ressources sont saturées ou indisponibles, ou lors d'un dépassement d'échéance.
- Message erroné : un message est erroné si l'information envoyée par une station émettrice est reçue par la station réceptrice avec des modifications de contenu. Par exemple un régulateur envoie un ordre d'ouverture pour une vanne, et à la réception, le contenu du message ayant été modifié, l'ordre n'est pas exécuté. Ceci peut être dû à des altérations des bits (*Bit-Flip*) durant la transmission du message, ou à des erreurs au niveau du mécanisme de décodage.

Dans notre étude nous nous limitons aux deux premiers types de fautes : retards variables et pertes de messages. Ceci peut être justifié par la haute capacité des protocoles de communication de détecter les messages erronés.

4.2.1 Critères de performances

La qualité d'un système se définit en caractérisant l'ensemble des comportements admissibles du système régulé : évolution des différentes grandeurs dans le cadre du changement de consigne, ou bien réaction du système face à un bruit. Ainsi, un système est dit en équilibre quand il n'y a plus de variation de ses grandeurs d'états (la sortie est alors constante). La qualité du système peut s'exprimer par sa capacité à atteindre l'équilibre. Plus précisément, dans le cas d'un changement de consigne, il s'agit de quantifier la capacité à atteindre l'équilibre désiré (régime permanent) au bout d'un temps fini. Généralement trois paramètres sont importants : le dépassement, le temps de réponse à une entrée en échelon et la stabilité. Il s'agit de concevoir un système avec un minimum de dépassement et un temps de réponse aussi court que possible et de prouver que le système reste stable. D'autre part, il s'agit de réduire l'écart entre la valeur de consigne et la sortie atteinte au point d'équilibre, c'est-à-dire l'erreur statique. Ces critères sont rappelés sur la figure 4.1.

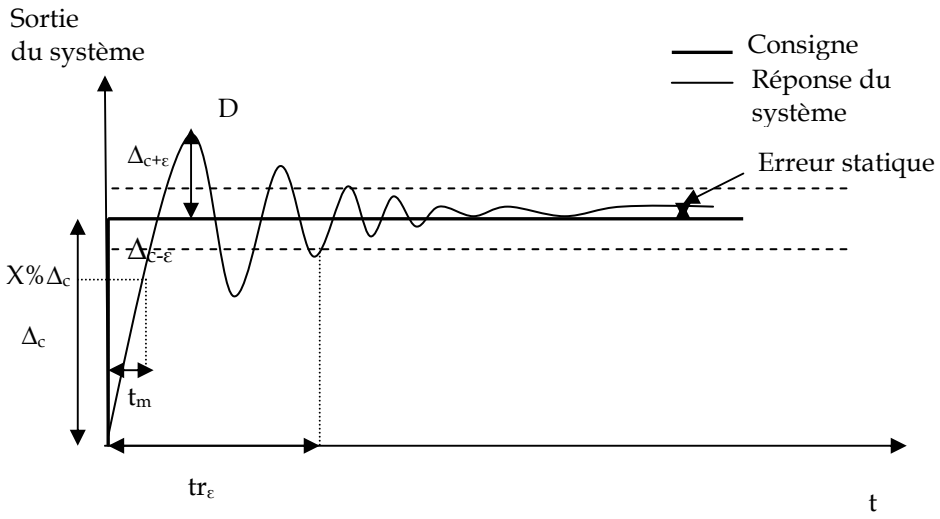


Figure 4.1 : Illustration des critères de performances utilisés pour caractériser un système lors d'un échelon de consigne.

La figure 4.1 illustre les paramètres de performances d'un système asservi. Le changement de consigne est noté Δ_c. D représente le maximum des écarts à la consigne attendue. Le temps de réponse tr_ε est le temps nécessaire entre le changement et la stabilisation à ε% du nouvel équilibre, le temps de montée t_m à X% est le temps nécessaire pour atteindre X% de la consigne. L'erreur statique représente l'écart entre la sortie au régime permanent et la consigne.

4.2.2 Influence du retard variable

Pour illustrer l'influence du retard variable, reprenons l'exemple du chapitre 3. Nous avons vu qu'un retard constant égal à 48% de la période d'échantillonnage sur chaque message transmis sur le réseau rend le système instable. Il est possible d'adapter la loi de commande à la connaissance a priori de ces temps de transmission d'une façon qui permet d'améliorer considérablement les performances du système ; ceci peut être réalisé par exemple avec un prédicteur de Smith [Aström et Wittenmark, 1990]. De telles techniques nécessitent donc que les retards soient constants. D'autres travaux ont été menés dans la communauté des automaticiens pour la recherche d'algorithmes de contrôle robustes aux retards variables ou tolérants les pertes de messages. Cependant, là n'est pas notre propos. Ce qui nous intéresse dans ce paragraphe est de donner un moyen pour caractériser l'influence des retards variables sur les performances d'un SCR.

Nous allons pour cela considérer que toutes les informations envoyées entre les composants subissent un retard aléatoire avant d'arriver à leurs destinations (Figure 4.2). Ce retard est tiré aléatoirement suivant une loi uniforme dans un intervalle de temps compris entre 0 et 40% de la période d'échantillonnage. Nous rappelons que

seul le capteur procède à un échantillonnage périodique, régulateur et actionneur fonctionnent sur réception des messages portant leur information d'entrée.

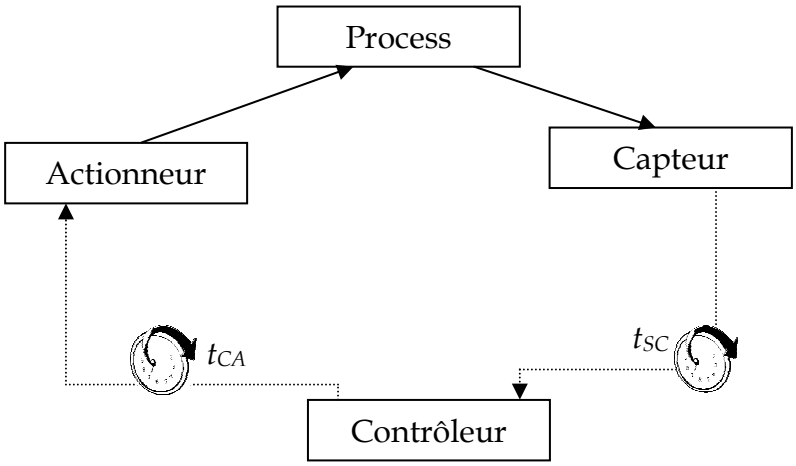


Figure 4.2 : Durées de transmission variables des informations envoyées

Le choix de 40% de la période est pris pour rester sous les limites de l'instabilité. Pour étudier l'influence du retard variable sur les paramètres de performances, nous avons répété 1000 essais de réponse à l'échelon de consigne (histoire) en introduisant simultanément un retard aléatoire sur la délivrance des informations entre capteur et régulateur d'une part et entre régulateur et actionneur d'autre part.

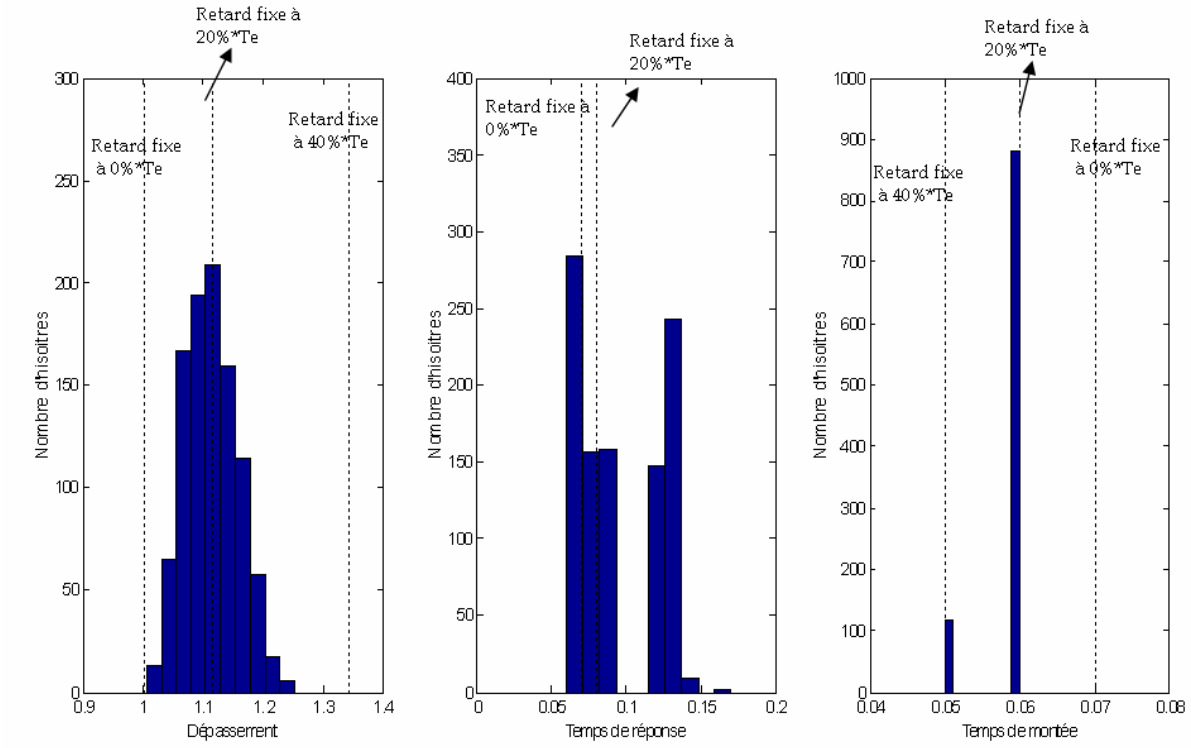


Figure 4.3 : Distribution du dépassement, du temps de réponse à 10% et du temps de montée en présence des retards variables (l'axe des abscisses représente la valeur du paramètre de performance et l'axe des ordonnées représente le nombre d'histoires qui ont évolué vers la valeur donnée en abscisse)

La figure 4.3 montre la distribution des paramètres de performances (dépassement, temps de réponse et temps de montée) en présence d'un retard variable. Les traits noirs représentent la valeur du paramètre concerné pour un retard constant (maximal soit 40% T_e, minimal soit 0% T_e, moyen soit 20% T_e).
On voit bien que l'influence du retard variable agit sur le dépassement qui reste encadré par deux valeurs minimale et maximale qui correspondent respectivement au retard constant minimal et maximal. On remarque encore qu'un grand nombre d'histoires (400 histoires) ont un dépassement qui se rapproche d'une valeur particulière 1.11474 qui correspond à la valeur du dépassement dans le cas d'un retard constant moyen (20%*Te). Le temps de réponse prend une valeur parmi cinq valeurs particulières (pour des raisons de clarté, la valeur du temps de réponse dans le cas d'un temps constant a 40%*Te sur la figure est égale à 0.42s).
Pour le temps de montée au contraire, les retards variables améliorent le temps de montée, près de 90% des histoires ont un temps de montée de 0.06 pour un temps de montée de 0.07 dans le cas d'un retard fixe minimal. Vu que le système est discrétisé, la sortie du système n'est mesuré qu'aux instants d'activation du capteur, ce qui explique que les valeurs du temps de montée sont des multiples de la période d'échantillonnage. Pour les 1000 histoires aucun cas d'instabilité n'est apparu. Ceci peut être expliqué par le fait que tous les retards sont inférieurs au pire cas (48% de la période d'échantillonnage).

L'essai suivant vise à voir l'influence sur la stabilité du système. Les retards sont maintenant tirés aléatoirement suivant une loi uniforme dans un intervalle de temps compris entre 20% et 60% de la période d'échantillonnage.

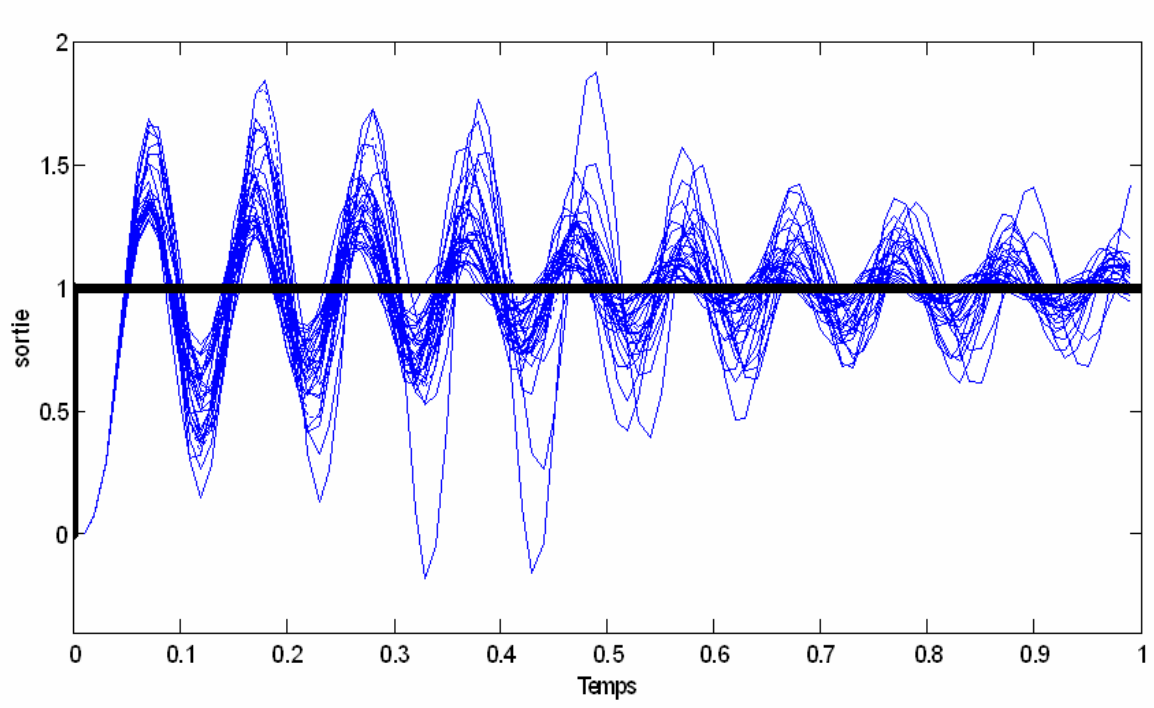


Figure 4.4 : Evolution de la sortie en présence d'un retard variable compris entre 20% et 60% de la période d'échantillonnage

La figure 4.4 superpose les sorties pour un essai de 1000 histoires. On remarque ici que la perturbation du système s'accentue très fortement avec l'augmentation de la plage du retard variable. On remarque aussi des cas où le système a tendance à l'instabilité. Il est intéressant de noter que bien qu'ayant une tendance à l'instabilité, le système ne quitte toutefois pas la zone de stabilité, même si certains retards dépassent la valeur fatidique de 48%.T_e. Cela indique que le système est robuste à des retards plus grands que le retard maximal, si la fréquence d'occurrence de ces retards est suffisamment faible.

4.2.3 Influence des pertes des messages

On considère maintenant que toute information échangée entre les composants peut être perdue avant d'être reçue par la station destinataire. On distingue deux types de messages : la mesure et la commande (Figure 4.5).

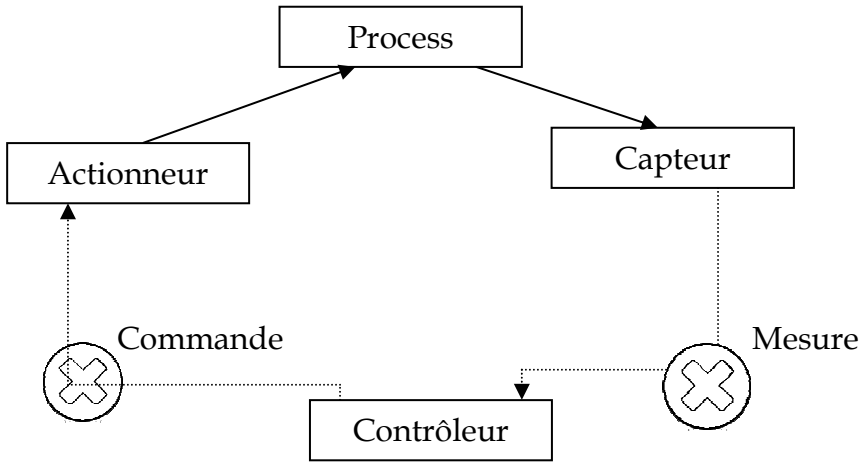


Figure 4.5 : Pertes des message circulant sur le réseau

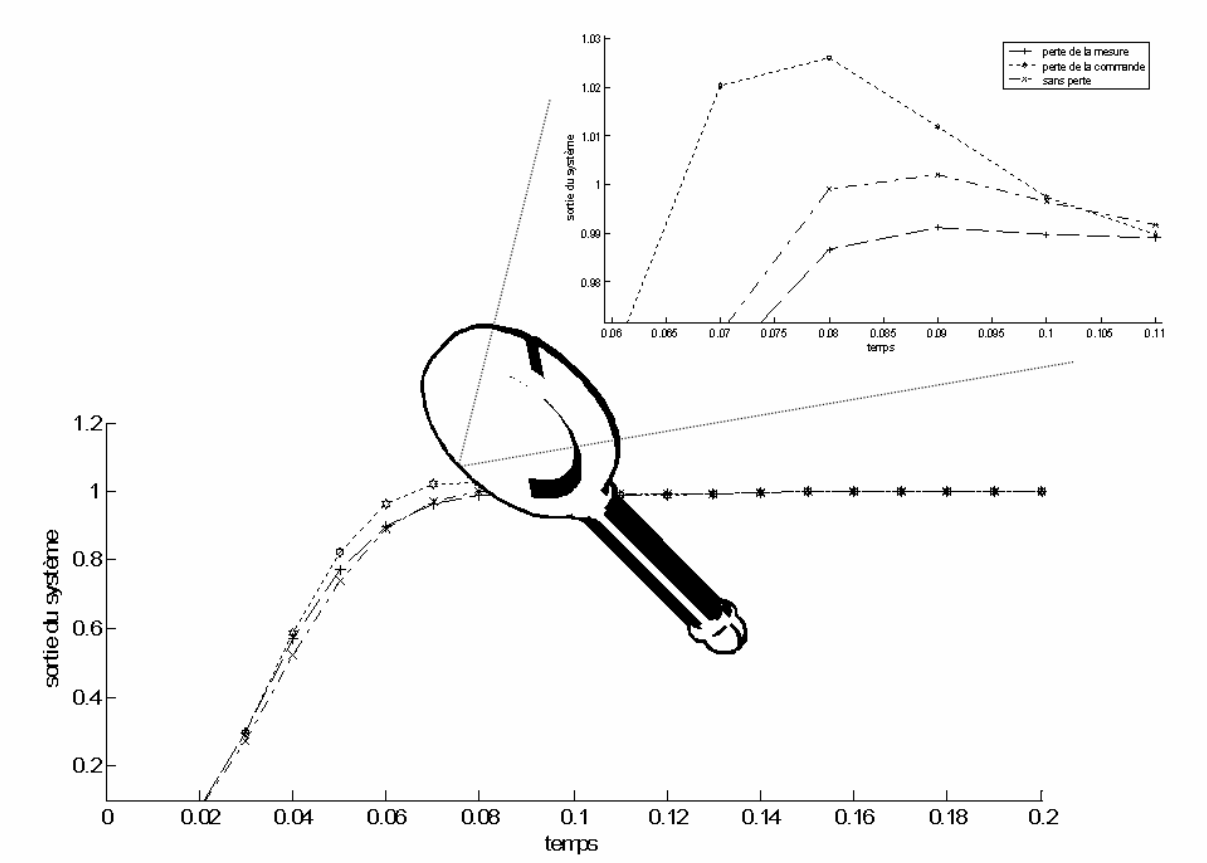


Figure 4.6 : Perte du troisième échantillon

La figure 4.6 montre la réponse du système à un échelon de référence, en présence de pertes sur l'information. On peut remarquer que l'influence d'une perte dans la même période de traitement n'est pas identique, selon qu'elle concerne la mesure ou la commande. Dans le premier cas, le contrôleur ne reçoit pas d'information et n'élabore pas une nouvelle commande. Dans le second cas, une commande est élaborée mais elle n'est pas acheminée à l'actionneur.

La figure 4.7 montre la distribution des paramètres de performance en présence des probabilités des pertes Pperte sur les messages (1000 histoires de réponse à un échelon pour chaque probabilité donnée de perte).

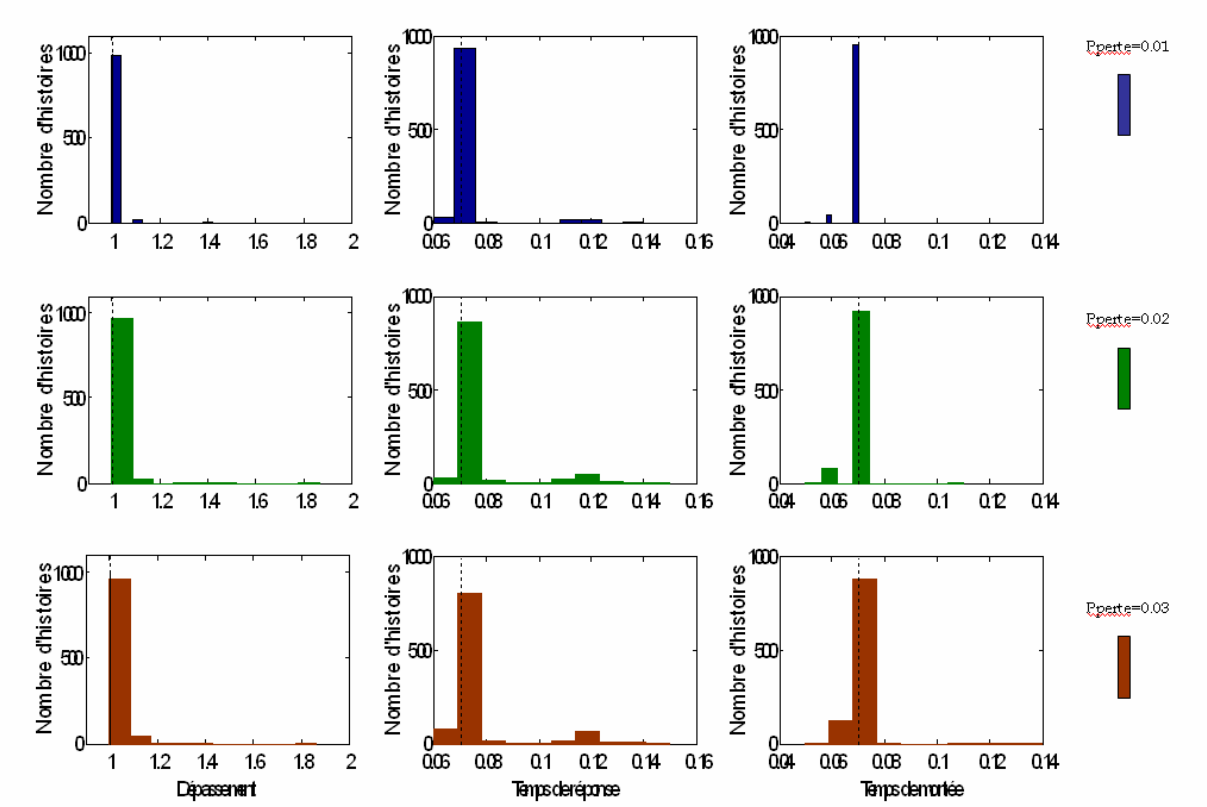


Figure 4.7 : Distribution du dépassement, du temps de réponse à 10% et du temps de montée pour des probabilités de perte de 0.01, 0.02, 0.03 (l'axe des abscisses représente la valeur du paramètre de performance et l'axe des ordonnées représente le nombre d'histoires qui ont évolué vers la valeur donnée en abscisse)

Les traits noirs représentent la valeur du paramètre concerné dans le cas où il n' y a pas de pertes. On remarque que dans la plupart des cas, les paramètres de performances ne sont pas affectés et que les valeurs se rapprochent des valeurs obtenues sans perte. Les résultats montrent que la qualité (notamment pour le temps de réponse et le temps de montée) en présence de pertes peut être meilleure que dans le cas idéal (pas de pertes). Bien que dans la plupart des cas le dépassement n'a pas changé, on observe des situations où le dépassement est à plus de 160% de la consigne ; ces situations correspondent aux pertes des messages dans la phase transitoire : cela peut être dangereux, lorsque ces dépassements sont non tolérables par le système.

4.2.4 Superposition de retards variables et pertes des messages

Dans ce paragraphe, on regardera l'influence simultanée des retards variables et des pertes des messages. Le retard est tiré aléatoirement suivant une loi uniforme dans un intervalle de temps compris entre 0 et 40% de la période d'échantillonnage. Et la probabilité de perte des messages est fixée à 0.01.

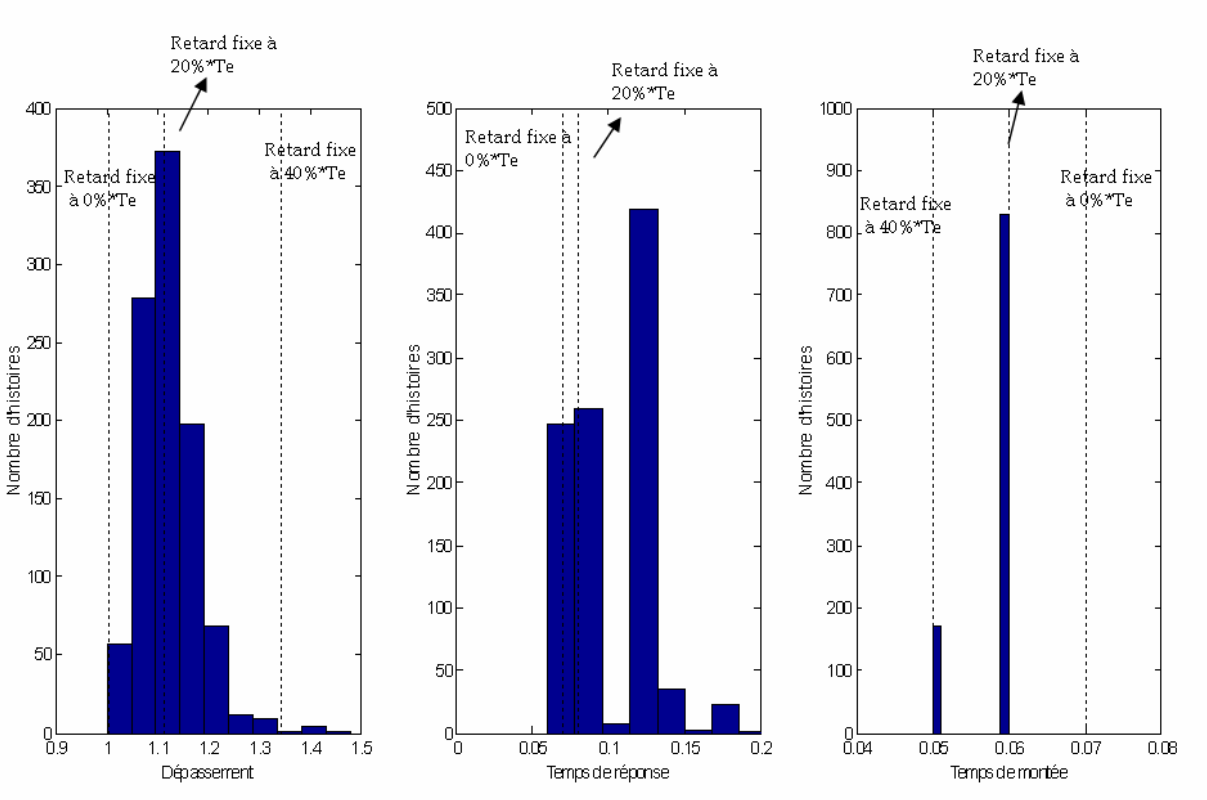


Figure 4.8 : Distribution du dépassement, du temps de réponse à 10% et du temps de montée en présence des retards variables et de perte des messages (l'axe des abscisses représente la valeur du paramètre de performance et l'axe des y représente le nombre d'histoires qui ont évolué vers la valeur donnée en abscisse)

Les résultats de la superposition des deux types de fautes ressemblent à ceux du retard variable avec une petite dégradation sur le dépassement et le temps de réponse. Pour le dépassement, on commence à avoir des dépassements plus grands que 1.34263 (valeur obtenue pour un retard constant maximal). Comme dans le cas d'un retard variable seul, un grand nombre d'histoires (370 histoires) ont un dépassement qui se rapproche de 1.11474 (dépassement obtenu pour un retard constant à 20%Te). Pour le temps de réponse, la superposition des deux types de fautes donne de nouvelles valeurs qui n'apparaissent pas dans chacun des cas considérés séparément. Ces résultats montrent la difficulté d'évaluer l'influence de la superposition des deux types de fautes à partir une étude séparée sur chaque type de faute, et justifie l'approche proposée par simulation.

4.3 Vers une étude de la fiabilité

Si on définit la mission du système comme étant la régulation d'une variable physique avec une qualité de service (ou qualité de contrôle) définie par un certain nombre de critères : dépassement borné, temps de montée et temps de réponse

minimaux, et non tendance à l'instabilité, alors tout comportement du système en dehors de ces spécifications doit être considéré comme une défaillance de celui-ci. Les sources de défaillances que nous avons envisagées étant probabilistes, cette défaillance du système revêt en conséquence un caractère probabiliste que l'on pourrait assimiler au concept de fiabilité. On définit la défaillance par critère comme étant la probabilité qu'un critère ne soit pas vérifié, par exemple la défaillance par dépassement aura lieu si à un moment donné la sortie dépasse la consigne d'un écart spécifié. La défaillance de la boucle aura lieu si au moins une défaillance sur un des critères se présente. Il serait alors idéal d'identifier la relation existant entre fiabilité du système et probabilité des défaillances aux multiples causes de défaillances évoquées.
Ces défaillances comme nous l'avons vu font que le calcul des paramètres de performance et à plus forte raison la détermination de non satisfaction des critères sont difficilement réalisables d'une manière analytique. C'est pourquoi nous proposons une approche d'estimation de la probabilité de défaillance sur chaque critère par simulation de Monte-Carlo. Nos modèles *SAN* serviront de support à cette simulation.

4.3.1 Critères d'évaluation

Afin d'obtenir des évaluations significatives, il est nécessaire de produire plusieurs simulations pour obtenir un niveau de confiance des variables estimées. Notons qu'avec l'approche de Monte-Carlo, on ne garantit pas d'avoir les valeurs exactes des paramètres à évaluer. En fait, on ne peut garantir que la probabilité pour que le nombre d'histoires N assure un résultat avec une précision donnée.
Pour avoir une confiance justifiée dans nos résultats on définit deux paramètres : un intervalle de précision et un pourcentage d'appartenance à cet intervalle. A chaque fois qu'on lance une nouvelle histoire i (i étant le numéro de l'histoire), on calcule la nouvelle valeur moyenne nommée V_i et on teste si la différence entre V_i et V_{i-1}, nommée d_i, appartient a l'intervalle de précision : si c'est le cas, on augmente le pourcentage de 1/N. Si le pourcentage d'appartenance atteint une limite donnée on arrête les simulations.

Par exemple une précision de 5% d'intervalle de précision et de 95% de pourcentage d'appartenance exige que 95% des d_i calculés soient inférieurs à ±5% de la dernière moyenne.

4.3.2 Défaillance par dépassement

La défaillance par dépassement consiste à fixer un écart maximum D_{dep} entre la sortie et la réponse idéale à un échelon de consigne. Si à un moment donné le dépassement en présence de fautes transitoires excède la consigne de D_{dep}, on détecte une défaillance par dépassement. On peut justifier physiquement ce choix car dans de nombreuses applications industrielles le dépassement de certaines variables même

transitoirement peut avoir un caractère dangereux. Citons par exemple le courant électrique dans un composant d'électronique de puissance (phénomène d'avalanche), la tension électrique aux bornes d'une charge (arc électrique), le couple mécanique sur un arbre de transmission (rupture de l'arbre), etc…

4.3.2.1 Influence du retard variable sur la défaillance par dépassement

Nous avons répété les essais de réponse à l'échelon de consigne (histoire) en introduisant simultanément un retard aléatoire sur la délivrance des informations entre capteur et régulateur d'une part et entre régulateur et actionneur d'autre part. Pour chaque simulation, on détecte si le critère de dépassement est satisfait ou non. A chaque nouvelle histoire, on calcule le nombre moyen de fois où le critère n'est pas satisfait (dépassement supérieur au seuil). La simulation est arrêtée lorsqu'on a obtenu une différence entre deux valeurs moyennes successives inférieure à ±5% de la dernière moyenne, et ceci au moins dans 90% des histoires.

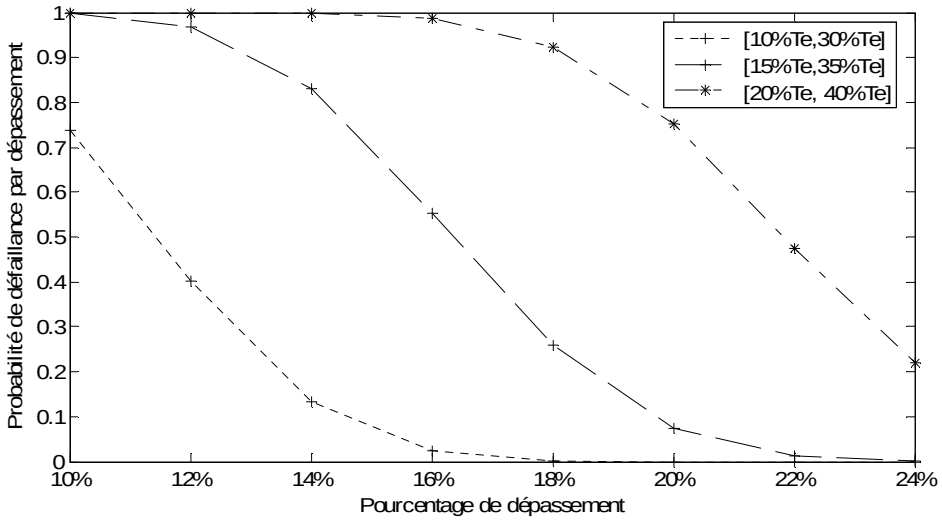


Figure 4.9 : Probabilité de défaillance par dépassement en présence des retards variables, l'axe des abscisses représente la contrainte de seuil D_{dep} (exprimé en % de la consigne), l'axe des ordonnées représente la valeur de la probabilité de défaillance

La figure 4.9 illustre comment on peut trouver la relation qui existe entre le retard variable qui peut être considéré comme un paramètre de la qualité de service du réseau d'une part et la défaillance par dépassement qui peut être une coupe d'ordre un ou bien un composant d'une coupe d'ordre supérieure vis-à-vis de la défaillance du système (système commandé). Les courbes montrent le changement de la probabilité de défaillance par dépassement en fonction de l'écart D_{dep}. Les trois courbes correspondent à trois classes de retard variable. Le retard est uniformément réparti entre 10% et 30% de la période d'échantillonnage pour le premier cas, entre 15% et 35% de la période d'échantillonnage pour le deuxième cas et entre 20% et 40%

pour le troisième cas. Ces valeurs sont prises pour illustrer l'influence du retard variable. Elles sont cependant réalistes, *pour le premier cas, si le protocole de* communication gère l'accès d'une manière prioritaire, on peut par exemple imaginer une situation ou :

- un message de haute priorité qui précède toujours l'envoi des messages relatifs à l'application, introduisant un retard de 10% de la période d'échantillonnage (125 bits pour une vitesse de débit de 125000bit/s)
- après ce retard le réseau sera momentanément indisponible pour une période aléatoire entre 0% et 20%.

D'autres causes peuvent engendrer ce type de retard : par exemple dans les réseaux ouverts non déterministes où le trafic peut varier d'une manière aléatoire. La meilleure façon de représenter ces retards est de représenter plus fidèlement le trafic et l'indisponibilité du réseau (défaillance transitoire), le paragraphe §4.4 de ce chapitre tente de répondre à cette question.

4.3.2.2 Influence des pertes sur la défaillance par dépassement

La même démarche pour l'évaluation des probabilités de défaillances est appliquée pour estimer l'influence des pertes des messages. On répète les essais de réponse à l'échelon de consigne (histoire) en introduisant simultanément des probabilités de pertes sur la délivrance des informations entre capteur et régulateur d'une part et entre régulateur et actionneur d'autre part. La simulation de Monte-Carlo est utilisée pour l'évaluation de la probabilité de défaillance par dépassement et le même critère d'arrêt est utilisé que dans le cas des retards variables (5% de précision et 90% de pourcentage d'appartenance)

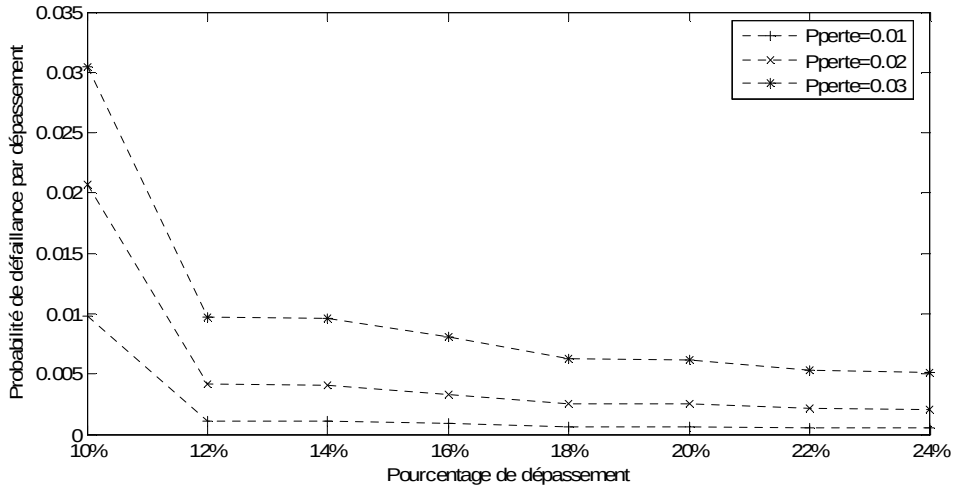


Figure 4.10 : Probabilité de défaillance par dépassement en présence des pertes sur les informations, l'axe des abscisses représente la contrainte de seuil D_{dep} (exprimé en % de la consigne), l'axe des ordonnées représente la valeur de la probabilité de défaillance

La figure 4.10 montre la probabilité de défaillance en fonction de la contrainte de seuil pour différentes probabilités de perte. Ces résultats montrent l'existence d'une relation entre la probabilité des pertes de message (un des paramètres de la qualité de service du réseau) et la défaillance par dépassement du système.
Comme on l'a fait remarquer à propos du retard variable, on peut s'interroger sur l'origine des pertes pour caractériser leur probabilité d'occurrence selon l'environnement et le trafic circulant sur le réseau. On pourrait prendre en compte par exemple la probabilité d'avoir deux messages consécutifs perdus (cette éventualité est plausible dans le cas des réseaux sans fil).

4.3.2.3 Influence de la superposition de retards variables et de la perte de messages sur la défaillance par dépassement

Les deux paragraphes précédents ont considéré une seule source de faute et ont permis de mettre en évidence la relation entre la source de faute et la défaillance par dépassement. Dans la suite, on cherchera à mettre en évidence l'influence simultanée de ces deux types de sources sur la défaillance par dépassement.

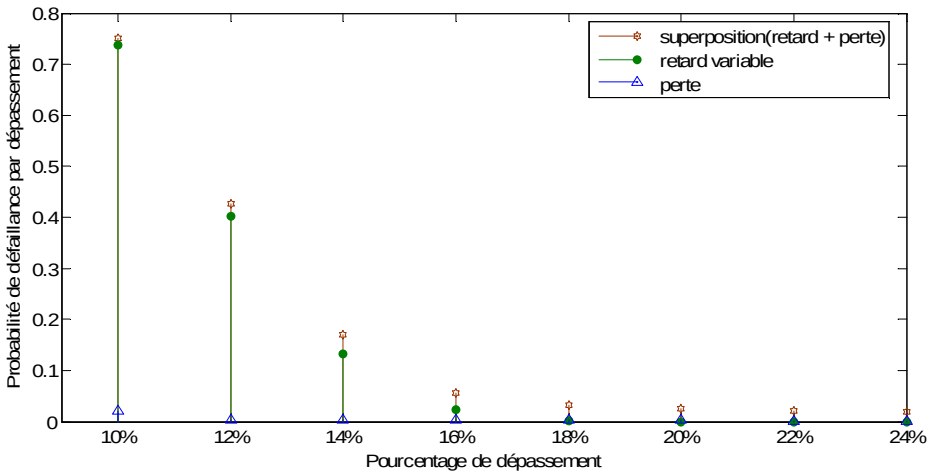


Figure 4.11 : Probabilité de défaillance par dépassement en présence des pertes et des retards variables, l'axe des abscisses représente la contrainte de seuil D_{dep} (exprimé en % de la consigne), l'axe des ordonnées représente la valeur de la probabilité de défaillance

Le retard est choisi uniformément entre 10% et 30% de la période d'échantillonnage et la probabilité de perte est fixée à 0.01. Sur la figure 4.11, les trois courbes correspondent aux trois cas : 1) retard variable 2) pertes des messages 3) superposition retard plus pertes des messages.
Une probabilité de perte de messages de 0.01 sur un réseau est une probabilité assez élevée, en pratique cette probabilité est plus petite ; tandis qu'un retard uniformément varié entre 10% et 30% est plus réaliste. Avec ces valeurs nous trouvons que le dépassement est beaucoup plus sensible à la variation du retard qu'à la perte des messages (la probabilité de dépasser de 10% la consigne est de l'ordre de

0.7 pour un retard variable et de 0.009 pour une probabilité de perte de 0.01). Les résultats de simulation montrent qu'il existe des cas où le système est fiable vis-à-vis d'une source unique de défaillance mais pas pour la superposition de deux sources différentes.

4.3.3 Défaillance par temps de réponse

La défaillance par temps de réponse est étudiée avec la même approche que la défaillance par dépassement : on fixe un écart maximum (allongement) $S_{réponse}$ entre le temps de réponse en présence de perturbations et le temps de réponse idéal à un échelon de consigne. Si à un moment donné l'écart en présence de fautes transitoires excède $S_{réponse}$, on détecte une défaillance par temps de réponse. On peut justifier ce choix puisque le temps de réponse caractérise la durée du phénomène transitoire qui dans de nombreux cas est source de perte énergétique. De telles pertes doivent être systématiquement évitées lorsqu'elles risquent par exemple de réduire dangereusement l'autonomie énergétique d'un système.

Comme dans le cas de la défaillance par dépassement, on étudie l'influence du retard variable, des pertes des messages et de la superposition des deux sur la probabilité de défaillance par temps de réponse. Par cette étude, on vise à établir expérimentalement comment la probabilité de défaillance par temps de réponse dépend de la probabilité de variation du retard et de la probabilité de perte de messages.

Les probabilités sont évaluées par simulation de Monte-Carlo. Le critère d'arrêt est le même que celui utilisé dans le cas du dépassement : intervalle de confiance de 5% et pourcentage d'appartenance de 90%.

4.3.3.1 Influence du retard variable sur la défaillance par temps de réponse

La figure 4.12 montre l'évolution de la probabilité de défaillance (probabilité que le temps de réponse dépasse le seuil spécifié) en fonction de la valeur de seuil choisi $S_{réponse}$ pour trois classes de retard variable (les mêmes classes de retard variable que celles utilisées au §4.3.2.1).

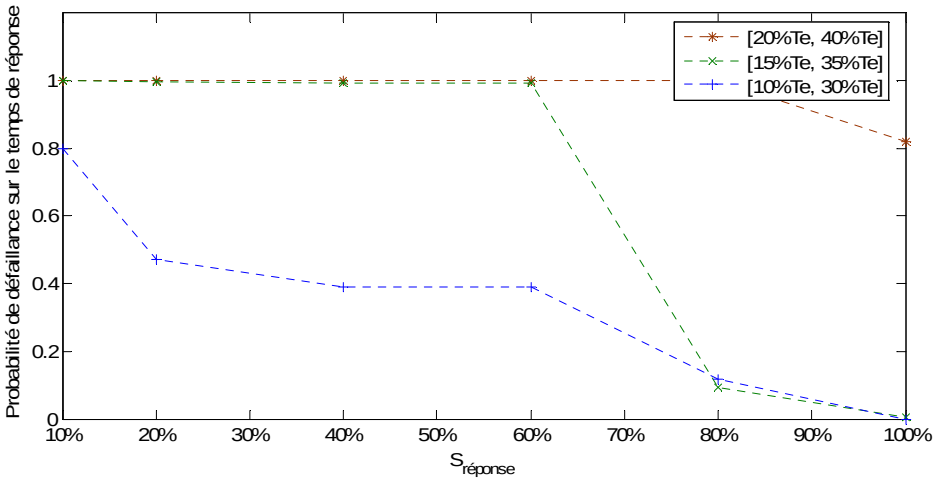


Figure 4.12 : Probabilité de défaillance par temps de réponse en fonction de la contrainte sur l'allongement $S_{réponse}$ (exprimé en pourcentage du temps de réponse normal) pour différentes classes de variation du retard

On peut remarquer que pour les valeurs de seuil 60% et 100%, on obtient les mêmes valeurs pour la probabilité de défaillance. En effet, lors de l'étude sur l'influence du retard variable sur le temps de réponse, on a remarqué que le temps du réponse prend des valeurs parmi cinq valeurs particulières seulement ce qui induit des effets de seuil.

4.3.3.2 Influence des pertes sur la défaillance par temps de réponse

La figure 4.13 montre la variation de la probabilité de défaillance par temps de réponse en fonction de la valeur du seuil $S_{réponse}$. On peut tirer la même remarque que dans le cas de la défaillance par dépassement, le temps de réponse est plus sensible aux retards qu'à la perte des messages (Malgré la grande probabilité prise pour la probabilité de pertes des messages, les probabilités de défaillances sont plus petites que dans le cas du retard variable).

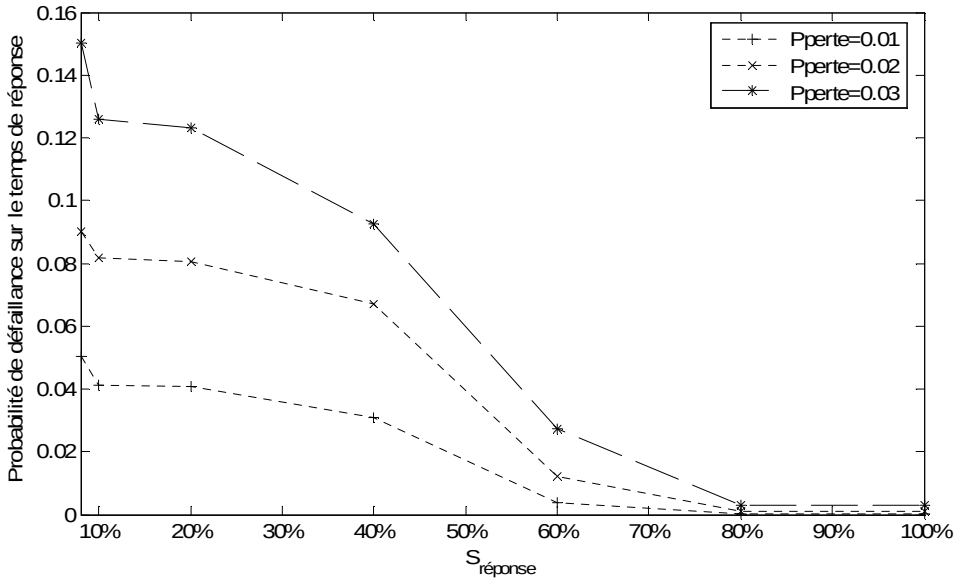


Figure 4.13 : Probabilité de défaillance par temps de réponse en fonction de la contrainte de seuil $S_{réponse}$ (exprimé en pourcentage du temps de réponse normal pour différentes probabilités de pertes des informations)

4.3.3.3 Influence de la superposition du retard variable et des pertes des messages sur la défaillance par temps de réponse

Ces essais nous permettent d'évaluer la probabilité de défaillance par temps de réponse. Dans les deux parties précédentes, on a mis en évidence l'influence de chaque source potentielle de cette défaillance ; ici nous allons superposer ces deux sources dans une même simulation afin d'en déduire leur effet cumulé.

Le retard est choisi uniformément entre 10% et 30% de la période d'échantillonnage et la probabilité de perte est fixée à 0.01.

De même que dans le cas de la défaillance par dépassement, les résultats montrent des cas où le système est fiable vis-à-vis d'une source unique de défaillance mais pas pour la superposition de deux sources différentes.

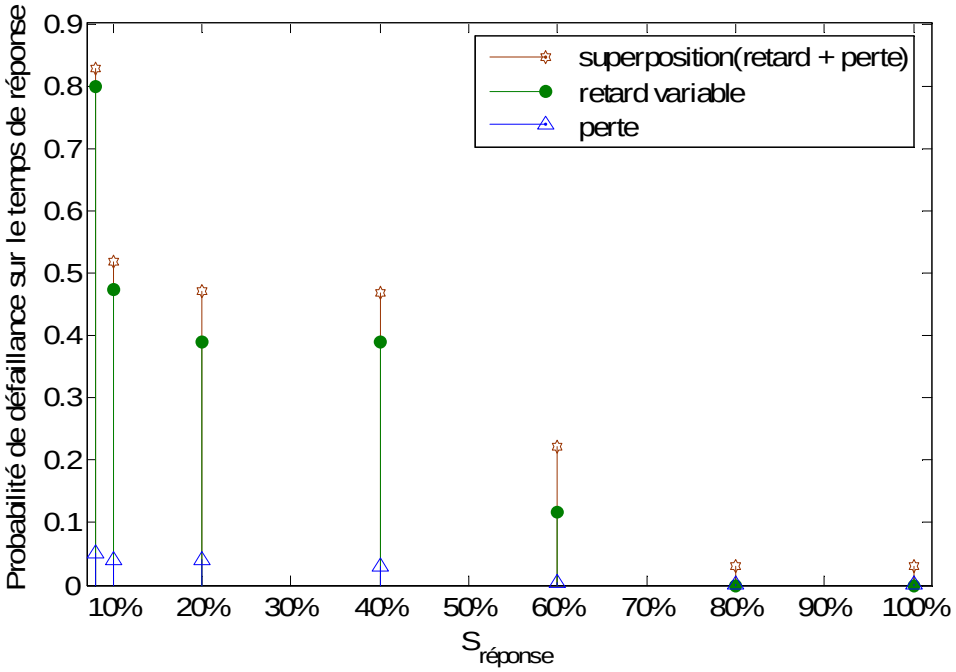


Figure 4.14 : Probabilité de défaillance par temps de réponse en fonction de la contrainte de seuil $S_{réponse}$ (exprimé en pourcentage du temps de réponse normal) en présence des pertes, du retard variable et de la superposition des deux.

4.3.4 Défaillance par stabilité

On cherche ici à identifier des situations où les perturbations ont une tendance à rendre le système instable. Pour définir la défaillance par stabilité, on surveille les pics successifs de la réponse à l'échelon en présence de ces fautes transitoires. Si on observe des pics d'amplitude croissante, on suppose que l'on est dans une situation de tendance à l'instabilité et on déclare une défaillance par stabilité.

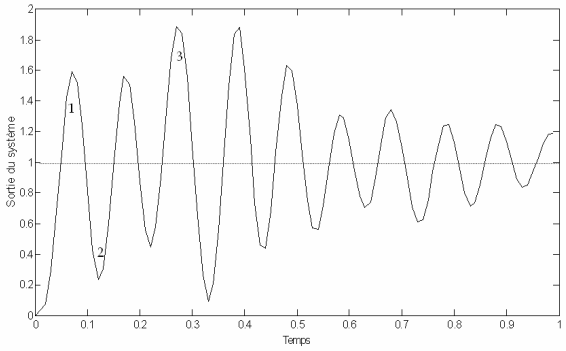


Figure 4.15 : Tendance à l'instabilité

Plus précisément, le critère est le suivant : si on mesure trois amplitudes successives de pics (en valeur absolue) croissantes, on identifie une tendance à l'instabilité. On déclare alors une défaillance par stabilité. Un tel comportement du système est

potentiellement inacceptable, il conduirait à des effets indésirables cumulant et amplifiant ceux déjà cités a propos du dépassement et du temps de réponse.

Tableau 4.1 Probabilité de défaillance par stabilité

Retard variable compris entre [20%Te, 60%Te]	Probabilité de défaillance par stabilité
Cas 1 : retard sans perte	0.0088
Cas 2 : retard + probabilité de perte 0.01	0.0496
Cas 3 : retard + probabilité de perte 0.02	0.1251
Cas 4 : retard + probabilité de perte 0.03	0.228

Le tableau 4.1 montre les résultats obtenus pour la probabilité de défaillance par stabilité. Les résultats sont donnés pour un retard variant uniformément entre 20% et 60% de la période d'échantillonnage. Cette partie a permis d'évaluer la probabilité de défaillance par stabilité. Sachant que le comportement « tendance à l'instabilité » est un événement indésirable, il est important d'évaluer sa probabilité d'occurrence. Dans le cas général, une étude antérieure doit précéder la phase d'évaluation de cette probabilité pour décider du nombre de pics inacceptable, ce paramètre dépend du contexte de l'application. Dans cette étude on a supposé que ce paramètre est une donnée pour l'étude de la sûreté de fonctionnement. Une suite de trois pics à amplitude croissante a été prise comme critère pour illustrer l'étude.

4.3.5 Evaluation de la fiabilité

D'après la définition de la fiabilité donnée précédemment, le système défaille si l'un des critères de performances n'est pas satisfait. Pour calculer la fiabilité, on procède de la manière suivante :
- la probabilité de défaillance par chaque critère est d'abord calculée, dans des conditions données (retard, probabilité de perte, ou bien la superposition des deux)
- la fiabilité suivant notre définition est ensuite déterminée : le calcul de la fiabilité se fait de deux manières : par le théorème de Poincaré connaissant les probabilités de défaillances par chaque critère ou par simulation directe sur le système.

Pour calculer la fiabilité il faut fixer les seuils pour chaque paramètre. Dans ce paragraphe on limite l'étude au dépassement et à la tendance à l'instabilité. On fixe le $D_{dép}$ à 70%. Pour la stabilité le même critère que dans le paragraphe précédent est utilisé (trois pics croissants).

La défaillance du système aura lieu si une défaillance par stabilité ou une défaillance par dépassement se présente. En terme de probabilité, cela se traduit par :

$$P_{defaillance\_système} = P(def\_dépassement \cup def\_stabilité)$$
$$= P_{def\_dépassement} + P_{def\_stabilité} - P(def\_dépassement \cap def\_stabilité) \qquad (4.1)$$

La probabilité de chacun de ces trois termes peut être calculée à partir des simulations sur les modèles SAN. Par exemple pour calculer la probabilité du troisième terme il suffit d'ajouter deux places *défaillance_dépassement* et *défaillance_stabilité* (figure 4.25) et de mettre un jeton dans la place *défaillance_dépassement* après une défaillance par dépassement et un jeton dans la place *défaillance_stabilité* après une défaillance par stabilité. La probabilité du terme $P(def\_dépassement \cap def\_stabilité)$ sera la probabilité d'avoir un jeton dans chacune des deux places.

Le tableau suivant donne l'évaluation de ces termes pour une précision de 5% et un pourcentage d'appartenance de 90%.

Tableau 4.2 : Evaluation des probabilités de défaillance

Retard variable compris entre [20%Te, 60%Te]	$P_{dépassement}$	$P_{def\_stabilité}$	$P(def\_dépassement \cap def\_stabilité)$
Cas 1 : retard sans perte	0.0151	0.0088	0.00026
Cas 2 : retard + probabilité de perte de 0.01	0.0287	0.0496	0.0017
Cas 3 : retard +probabilité de perte de 0.02	0.0415	0.1251	0.005
Cas 4 : retard +probabilité de perte 0.03	0.0551	0.231	0.01

A partir de ces probabilités déterminées, la fiabilité pourra être calculée avec l'équation 4.1 (théorème de Poincaré).

On peut aussi évaluer directement la fiabilité de la manière suivante : pour chaque histoire on teste si au moins un critère n'a pas été satisfait, si c'est le cas on augmente le compteur relatif à la défiabilité $n_{déf}(t)$ (cf chapitre 2 §2.2.3.5) qui représente le nombre d'histoires où le système s'est retrouvé dans un état défaillant. La fiabilité est estimée par la relation suivante :

$$R(t) = 1 - \frac{n_{déf}(t)}{N} \qquad (4.2)$$

Le tableau 4.3 compare les résultats de évaluation de la fiabilité par les deux méthodes.

Fiabilité	Equation 4.1	Equation 4.2	écart
Cas 1 :	0.0236	0.0235	0.04%
Cas 2 :	0.0766	0.0738	03.6%
Cas 3 :	0.1616	0.1589	1.6%
Cas 4 :	0.2761	0.2683	2.8%

On remarque que les valeurs constatées sur les écarts sont toujours inférieures à la précision que l'on s'est imposée sur la simulation (5%).

Pour obtenir les résultats du paragraphe 4.3, quatre campagnes de simulation (une campagne regroupe plusieurs études '*study*') ont été lancées :

- campagne_1 : pour l'évaluation de l'influence du retard variable.
- campagne_2 : pour l'évaluation de l'influence des pertes de messages.
- campagne-3 : pour l'évaluation de l'influence de la superposition du retard variable et des pertes de messages
- campagne-4 : pour l'évaluation directe de la fiabilité.

Les trois premières campagnes permettent d'évaluer les probabilités de défaillance par dépassement, par temps de réponse et par stabilité sous différentes conditions (retard variable, pertes, superposition). La quatrième campagne permet d'évaluer directement la fiabilité (§4.3.5). Toutes les campagnes peuvent s'exécuter en parallèle sur la même machine. Le temps de simulation est long mais acceptable, à titre d'exemple pour le calcul de la fiabilité dans le dernier paragraphe, 1158035 histoires ont été nécessaires avant l'arrêt des simulations (l'équivalent de 3 heures de simulation).

4.4 Intégration effective du réseau

Les modèles de retard variable et les pertes d'information suivant une loi de probabilité uniforme ont été utilisés pour calculer et évaluer la fiabilité dans la section précédente.

Si on tient compte maintenant du fait que l'origine des pertes ou retards est due à la présence du réseau, il faut l'introduire dans le modèle. Le caractère aléatoire n'est plus associé directement à la défaillance mais plutôt à la cause qui est susceptible de la produire. On introduit alors dans le modèle global la modélisation SAN d'un réseau CAN et l'injection des défaillances (chapitre 3 §3.2.2) dans celui-ci. On effectue alors le même type de simulation sur la réponse indicielle du système et l'enregistrement de ces dépassements des limites acceptables des performances. Dans ce qui suit, le modèle du fonctionnement et dysfonctionnement du réseau CAN que nous avons proposé (dans le chapitre 3) est utilisé pour évaluer en ligne les retards et les pertes. L'intégration du modèle du réseau représente plus fidèlement les retards et les pertes.

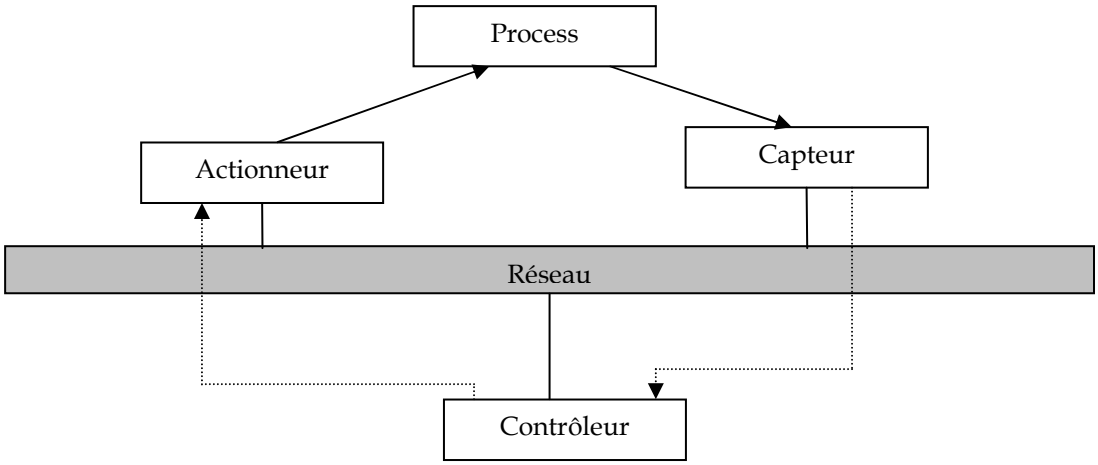


Figure 4.16 : Intégration effective du réseau

Dans le chapitre 3, nous avons présenté un modèle de perturbation basé sur un processus de Poisson pour représenter les perturbations électromagnétiques. On suppose que le système est immobile et que la source des fautes est présente tout au long de la mission, un jeton dans la place phase_perturbée (voir *Figure 3.5 : Modèle de perturbation*). La seule source de faute maintenant est l'arrivée des perturbations extérieures qui vont perturber le medium, l'effet de ces perturbations se traduit par des retards et des pertes sur les messages.

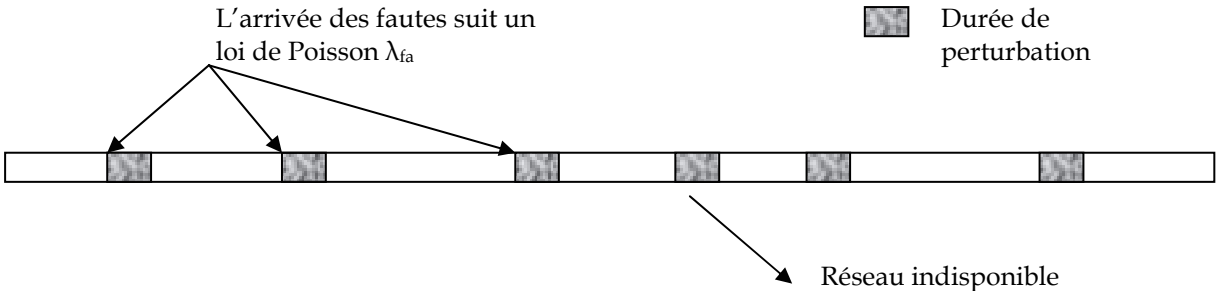


Figure 4.17 : Indisponibilité du réseau due aux perturbations

Pour les essais qui suivent, nous considérons que tous les messages circulant sur le réseau ont un champ de données de 3 octets (ce qui correspond à 85 bits)[5] et que le débit de transmission sur le medium est de 125Kbits/s. L'intégration de ces valeurs dans le modèle représentant se fait dans l'activité *transmitted* de la figure 3.2. La durée des perturbations est fixée à 3 fois la transmission d'un message normal. Cette durée est donnée pour illustrer l'étude, des durées variables peuvent être imaginées. Les durées des perturbations ainsi que leur taux d'arrivée sont généralement des données d'entrée pour l'étude de la sûreté de fonctionnement.

[5]La longueur en bits d'une trame de n octets de données dans CAN est donnée par la formule :

$$L = 47 + 8 \times n + \left\lfloor \frac{34 + 8 \times n - 1}{4} \right\rfloor$$

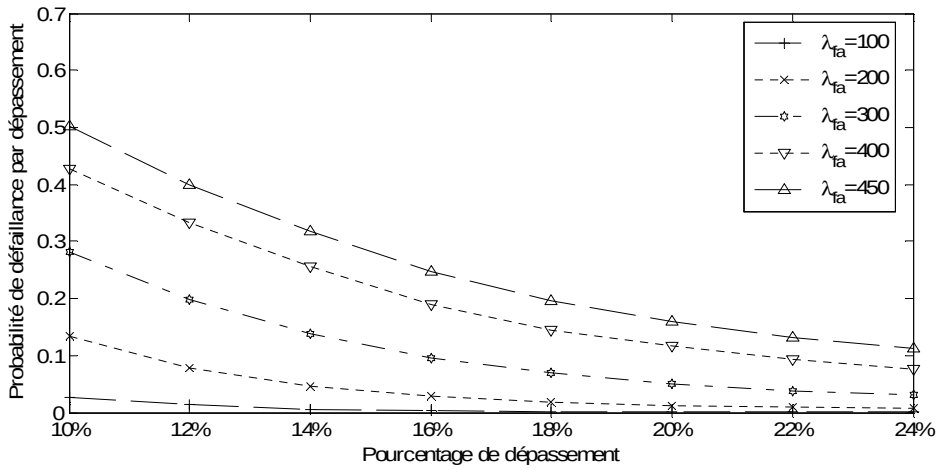


Figure 4.18 : Probabilité de défaillance par dépassement en fonction de la contrainte de seuil D_{dep} (exprimé en pourcentage de la consigne) pour plusieurs valeurs de λ_{fa} (taux d'arrivée des fautes).

La figure 4.18 montre les résultats de simulation (du modèle composé du chapitre 3 figure 3.16 sous les mêmes hypothèses d'arrêt de simulation, 5% de précision et 90% de pourcentage d'appartenance) mettant en évidence la défaillance par dépassement pour cinq valeurs différentes λ_{fa} qui représente le taux du processus de Poisson. Les valeurs de λ_{fa} reflètent l'état de l'environnement extérieur. La littérature propose de considérer des valeurs pouvant varier entre 75 et 500 fautes/secondes en fonction des milieux extérieurs [Portugal et Carvalho, 2004]. D'après le protocole CAN, un message affecté est automatiquement retransmis, le nombre de retransmissions dépend de la charge du réseau ; si la charge du réseau permet de le retransmettre plusieurs fois, le message finira par arriver à la station destinataire. Si au contraire la charge du réseau ne permet qu'un nombre limité de retransmissions et si les tentatives de retransmissions sont affectées, le message n'est plus retransmis et il est perdu. Dans notre étude, nous avons considéré que le réseau est dédié à la boucle et que chaque message a un nombre illimité de tentatives de retransmissions.

Le tableau 4.4 donne des paramètres relatifs à l'état du réseau, ces paramètres sont évalués pour chacune des valeurs de λ_{fa}.

	retard moyen	Efficacité	Utilisation
λ_{fa}=100f/s	0.0009	0.681	0.330
λ_{fa}=200f/s	0.0012	0.537	0.467
λ_{fa}=300f/s	0.0015	0.449	0.557
λ_{fa}=400f/s	0.0019	0.388	0.642
λ_{fa}=450f/s	0.0020	0.364	0.672

On remarque que pour λ_{fa} = 450, le retard moyen des messages circulant sur le réseau est de 0.002s. Ceci est très proche des conditions de la simulation avec un retard

variable uniformément réparti entre [0.001 et 0.003] (cf §4.3.2.1) où le retard moyen est égal à 0.002.

Une comparaison des résultats de la probabilité de défaillance par dépassement entre ces deux simulations montre que les résultats sont différents. Ceci est attendu puisque la distribution du retard n'est pas la même entre les deux cas (cas de l'intégration effective du réseau et cas du retard uniformément distribué).

4.4.1 Influence du trafic externe

On sait que les performances d'un réseau et notamment sa sensibilité aux sources de défaillance sont tributaires de sa charge et de la nature du protocole. Afin de mettre cela en évidence, nous simulons un système dans lequel le réseau est partagé avec d'autres applications.

Deux applications seront étudiées. Dans la première application on reprend le même exemple que précédemment et on ajoute un trafic périodique, on fait varier le URF[6] du trafic global circulant sur le réseau et on évalue l'influence sur la fiabilité. La seconde application contient trois boucles fermées distribuées autour un réseau de communication.

4.4.1.1 Trafic périodique

Comme le réseau peut être partagé avec d'autres composants (figure 4.19), on peut se poser la question suivante : quel trafic externe peut circuler sur le réseau en respectant un niveau donné de fiabilité ? Pour répondre à cette question il faut d'abord être capable d'évaluer la fiabilité en présence d'un trafic externe. On considère deux cas :
- Cas 1 : le trafic externe possède une priorité inférieure à celle de l'application,
- Cas 2 : le trafic externe possède une priorité supérieure à celle de l'application.

On suppose que tous les messages circulant sur le medium ont la même taille de 3 octets.

[6] URF : *User Request Factor* représente le taux de demande de l'utilisation du réseau

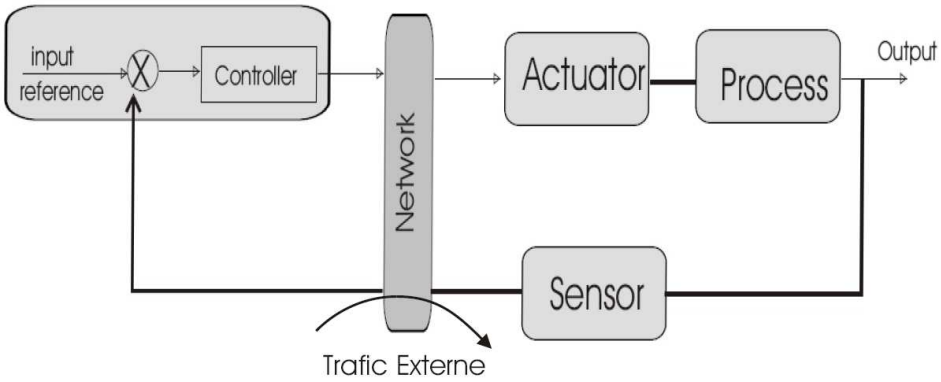


Figure 4.19 : SCR avec un trafic externe

4.4.1.1.1 Cas1 : $P_{trafic} < P_{application}$

Pour étudier l'influence du trafic, on fixe tous les autres paramètres : un débit de 125 kbits/s et un taux de perturbation de 200 fautes/s. Les cinq courbes correspondent aux différentes valeurs de URF représentant la probabilité de défaillance par dépassement pour différentes valeurs de URF.

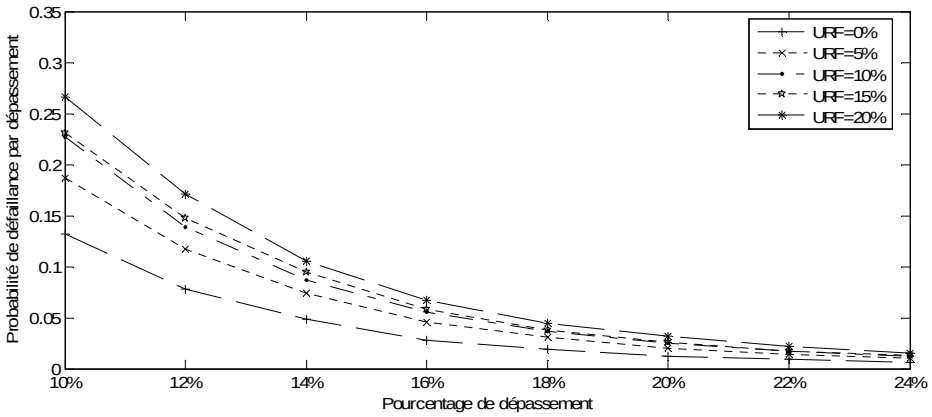


Figure 4.20 : Probabilité de défaillance par dépassement. L'axe horizontal représente la contrainte de seuil D_{dep} (exprimé en pourcentage de la consigne)

Bien que les messages de l'application aient une priorité supérieure à ceux du trafic externe, un retard s'ajoute aux messages de l'application. Ce retard dégrade la performance de la boucle et par conséquent la fiabilité.

4.4.1.1.2 Cas2 : Ptrafic > Papplication

Dans ce cas, le seul paramètre qui change est la priorité du trafic externe. Pour changer ce paramètre, il faut entrer dans le modèle représentant le réseau (ch.3, §6.2) plus précisément dans la sortie de l'activité *begin_transmission* et changer le code pour contrôler l'accès au medium de façon à favoriser l'accès aux messages du trafic externe.

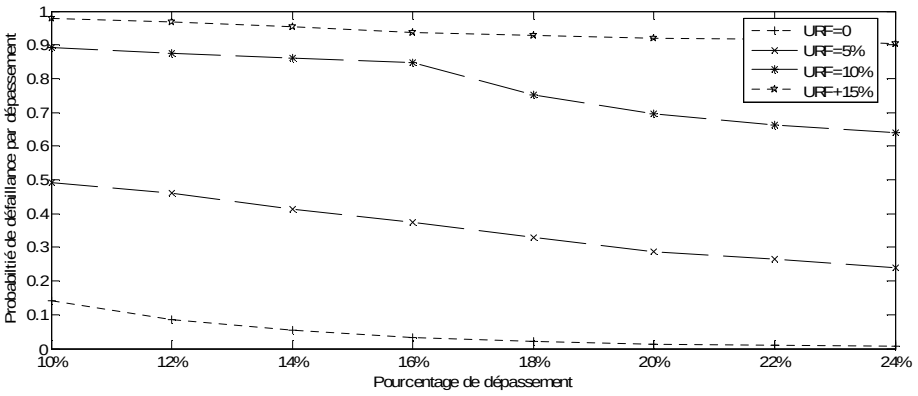


Figure 4.21 : Probabilité de défaillance par dépassement. L'axe horizontal représente la contrainte de seuil D_{dep} (exprimé en pourcentage de la consigne)

La probabilité de défaillance par dépassement se dégrade à cause de la grande priorité affectée au trafic externe. En effet, dans ce cas le retard subi par les messages de l'application est de plus en plus grand. On sait que sur CAN tous les messages prêts à être envoyés sont confrontés à un retard dû à :
- une transmission en cours,
- une station de priorité plus grande prête à émettre sur le réseau.

Dans le cas 1, les seuls retards sont ceux des transmissions en cours, tandis que dans le cas 2 les messages de priorité plus grande s'ajoutent pour augmenter le retard subi par les stations de l'application.
Cette partie permet de répondre à des questions comme :
- Etant donné un niveau de fiabilité à respecter quel est le trafic externe maximum qui peut circuler sur le réseau ?
- Si on imagine par exemple plusieurs trafics externes, quelles priorités doit-on attribuer aux messages de l'application pour respecter un niveau donné de fiabilité ?

Cette partie a permis l'étude de l'influence du trafic externe sur la défaillance par dépassement. Deux cas ont été considérés : trafic externe plus prioritaire et trafic externe moins prioritaire. D'autres types de trafic peuvent être pris en compte comme par exemple les deux cas ensemble, un trafic plus prioritaire et un trafic moins prioritaire en même temps. Cette partie a permis de montrer comment, connaissant l'information sur le trafic, la défaillance par dépassement puis la défaillance du système peuvent être évaluées.

4.4.1.2 Trois boucles partageant le même medium

L'application sur laquelle cette étude est présentée contient trois boucles fermées, distribuées autour un réseau de communication. L'exemple est tiré de l'article [Xia et al, 2006] (figure 4.22). Cette application est plus proche de la réalité, surtout dans le cas des systèmes commandés en réseau. Souvent on trouve plusieurs boucles qui partagent le même réseau.
La boucle étudiée au paragraphe précédent partage maintenant le réseau avec deux autres boucles. Chaque boucle est composée d'un capteur périodique S_i, un actionneur A_i, un contrôleur PID C_i et un système physique à commander représenté par une fonction de transfert. Tous les composants sont distribués autour d'un medium de communication.

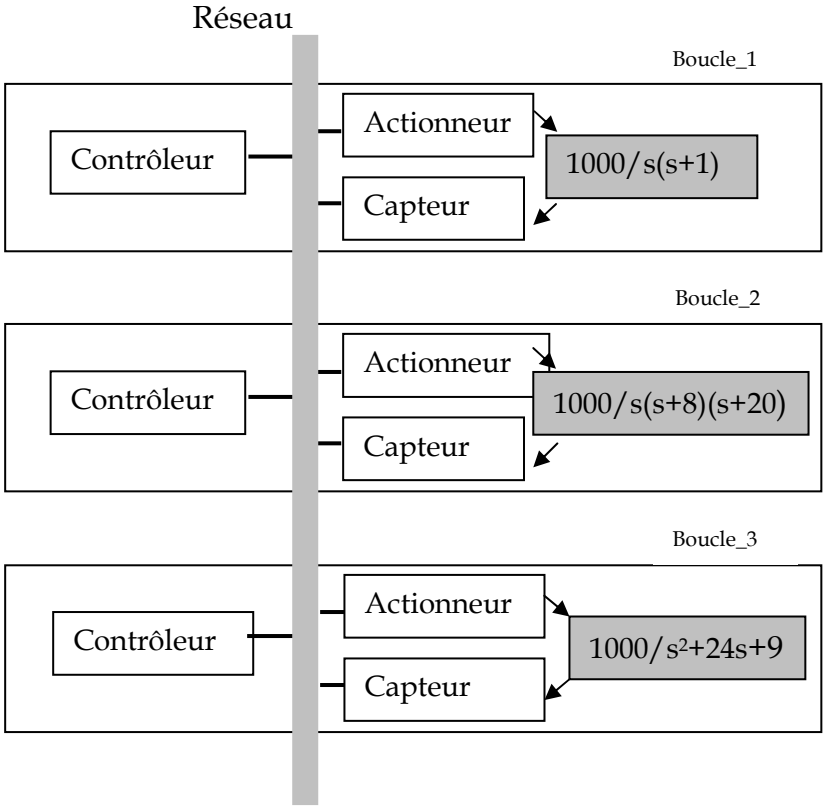


Figure 4.22 : Trois boucles partageant le même medium de communication

La fiabilité de la première boucle est étudiée (seul le critère de dépassement est étudié) en faisant varier la priorité des messages circulant sur le réseau.

La figure 4.23 montre le modèle composé correspondant à l'application.

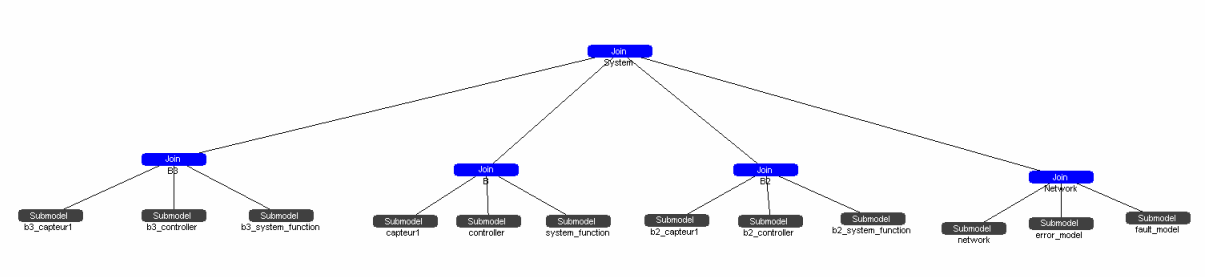


Figure 4.23 : Modèle composé des trois boucles

Le modèle du réseau ainsi que les modèles d'une boucle (capteur, régulateur, actionneur) se trouvent dans l'annexe B.
On distingue deux cas :
- la boucle numéro 1 possède la plus grande priorité,
- la boucle numéro 2 possède la plus grande priorité.

Le tableau 4.5 montre les priorités de tous les composants connectés sur le réseau.

Tableau 4.5 : Priorités des messages

Composant	Période d'activation	Priorité des messages (cas 1)	Priorité des messages (cas 2)
b1_sensor	TC1=0.01	1	3
b1_controller	événementiel	2	4
b2_sensor	TC2=0.008	3	1
b2_controller	événementiel	4	2
b3_sensor	TC3=0.009	5	5
b3_controller	événementiel	6	6

La méthode de Monte-Carlo et les critères d'arrêt définis précédemment sont utilisés pour le calcul des probabilités de défaillance par critères. Le taux de fautes est fixé à 200 fautes/s. Sur la durée de simulation, une évaluation des paramètres relatifs à la performance du réseau a été réalisée. Le tableau 4.6 récapitule ces différents paramètres.

Tableau 4.6 : Paramètre de performance du réseau

	retard moyen (boucle_1)	Efficacité du réseau	Utilisation du réseau
Cas 1	0.0014	0.44	0.764
Cas 2	0.0018	0.44	0.764

La courbe de la figure 4.24 montre la probabilité de défaillance par dépassement de la boucle_1. Pour les deux cas, la courbe au milieu correspond au cas où le réseau était dédié à la boucle_1 seule.

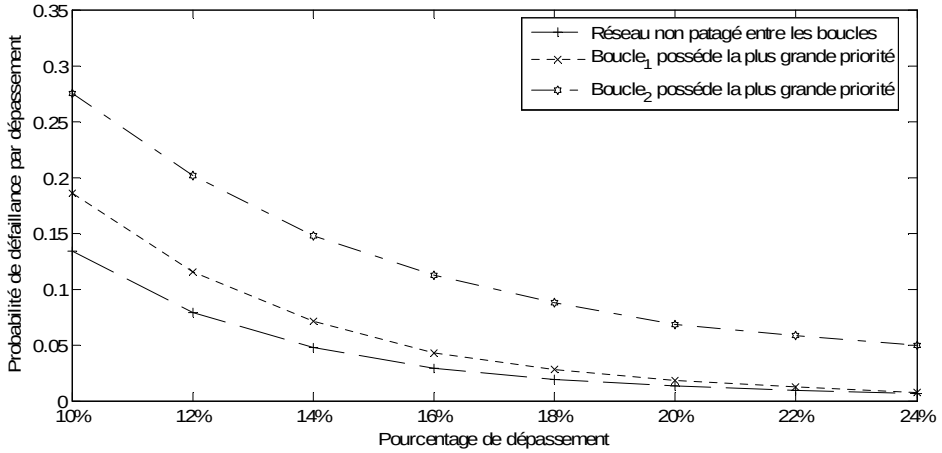


Figure 4.24 : Probabilité de défaillance par dépassement. L'axe horizontal représente l'écart D_{dep} (exprimé en pourcentage de la consigne)

L'existence d'autres boucles qui partagent le même réseau génère un trafic supplémentaire sur le réseau. Vis-à-vis de l'application (la première boucle), ce trafic est un trafic externe qui peut perturber les performances. Dans cette partie, l'influence de ce trafic sur la probabilité de défaillance de la première boucle a été étudiée. Il est possible que les trois boucles soient dépendantes, sous cette condition la définition de la défaillance dépendra des paramètres de performances des trois boucles. Ainsi le modèle de détection de la défaillance (figure 4.2) doit être modifié pour prendre en compte la nouvelle définition de la défaillance. Nous n'avons pas la prétention de résoudre le problème de l'évaluation de la fiabilité d'un SCR dans tous les cas possibles (plusieurs possibilités de trafics externes, plusieurs manières de définir la défaillance du système).

4.5 Vers un logiciel pour l'étude de la sûreté de fonctionnement des systèmes commandés en réseau

Cette étude nous a montré l'adéquation des réseaux d'activité stochastiques (SAN) pour mener des études de fiabilité dans un système commandé en réseau, par simulation de Monte Carlo. La possibilité de former un modèle composé dans Möbius et la possibilité d'utiliser des paramètres nous ont permis de créer des modèles de base paramétrés. Le but et de grouper ces modèles dans une bibliothèque facilement utilisable par n'importe quelle personne ayant une connaissance de base dans les réseaux de Petri. Actuellement un certain nombre de composants de bases pouvant se retrouver dans n'importe quel système commandé en réseau sont définis:

- capteur périodique, ayant la période comme paramètre,
- contrôleur PID, ayant comme paramètres les constantes P, I, et D,
- un réseau CAN, avec plusieurs paramètres concernant le débit de transmission, le taux de perturbation dû à l'environnement extérieur,
- un composant dérivateur,
- un composant intégrateur,
- des modèles qui correspondent aux fonctions de transfert du premier et du second ordre, les variables de ces modèles sont : la constante de temps et le gain statique pour les systèmes du premier ordre, la pulsation propre, le facteur d'amortissement et le gain statique pour les systèmes du second ordre.

Ces modèles sont prêts à être utilisés, il suffit de copier les modèles de base demandés pour l'exemple à modéliser, assigner les paramètres dans *study* et joindre les modèles de base dans un seul modèle composé. Cette dernière étape demande un peu d'attention, la personne qui se charge de cette tâche choisira les places partagées entre les différents modèles de base. Une fois que le modèle est prêt, des *reward function* peuvent être définies pour évaluer divers paramètres comme la probabilité d'être dans une place, les retards moyens, l'efficacité du réseau, etc…

Dans un souci de généralisation, un modèle de détection de la défaillance a également été créé, ce modèle étant également paramétré. La défaillance est définie en relation à des performances et des contraintes de la boucle comme par exemple :

- le temps de réponse,
- le temps de montée,
- la stabilité,
- l'erreur statique…

L'utilisateur fixera des limites sur le temps de réponse, le temps de montée et sur l'erreur statique. Si ces limites ne sont pas respectées une défaillance sera détectée.

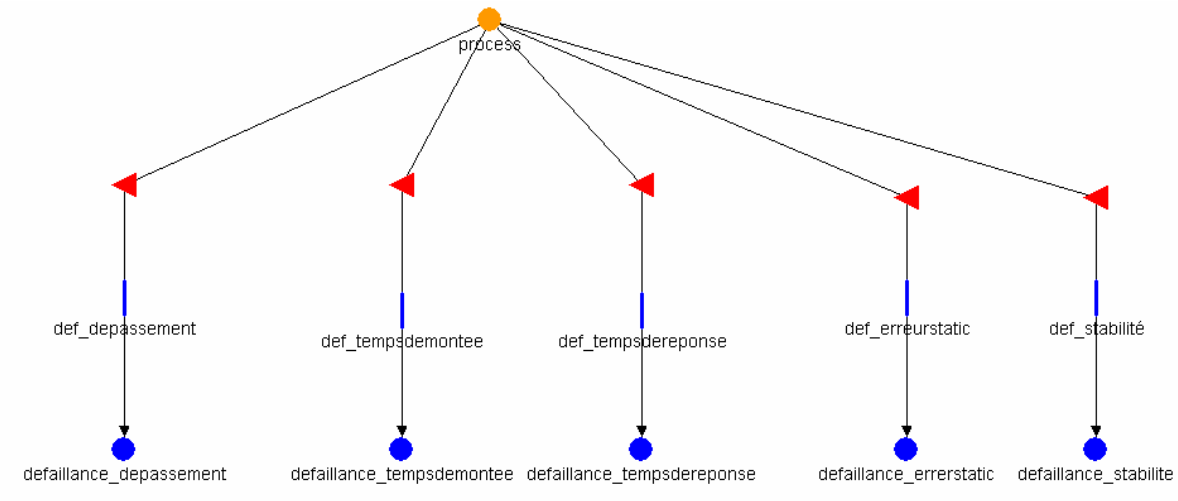


Figure 4.25 : Détection d'une défaillance en fonction des paramètres de performance de la boucle

Grâce à cette conception modulaire, l'outil d'évaluation pourra donc être constitué à la fois d'éléments de base (capteurs, actionneurs, contrôleurs de divers types, divers réseaux) mais également d'éléments qui sont eux-mêmes constitués d'éléments de base (systèmes en boucle fermée paramétrables…).
Chaque élément sera déterminé par ses paramètres fonctionnels (période d'échantillonnage, plage de mesure pour un capteur, type d'algorithme de commande et valeurs de ces paramètre pour le contrôleur…) et dysfonctionnels (taux de perturbation, type de perturbation, loi de probabilité…). Le réseau, élément particulier car il intègre plusieurs interfaces pour dialoguer avec les autres éléments du système, se caractérise par sa topologie et son protocole.

4.6 Conclusion

L'influence des délais sur les messages ainsi que les pertes des messages dues à des perturbations sur le réseau sont étudiées. Pour prendre en compte l'évaluation des paramètres de la sûreté de fonctionnement, la défaillance d'un SCR a été définie. Cette définition est basée sur un ensemble de paramètre de performances à vérifier. Cette définition de la défaillance et le test de détection des défaillances sont intégrés dans le modèle.

L'approche de Monte-Carlo est utilisée pour l'évaluation des paramètres de sûreté de fonctionnement. Grâce à la simulation et dans la mesure où ce type de système complexe intégrant différents aspects temporels et dynamiques est difficile à évaluer formellement, on est capable de prédire l'influence des perturbations sur la Sdf dans les limites classiques de ce type d'approche (explosion combinatoire, temps de simulation…). L'influence des priorités des messages ainsi que du trafic externe sur la fiabilité ont été aussi étudiés.

Finalement, pour profiter de nos modèles de base, nous avons développé une bibliothèque qui contient des modèles de base paramétrés prêts à être utilisés. Cette bibliothèque facilite énormément la tâche de modélisation et l'analyse d'un grand nombre de systèmes commandés en réseau. L'enrichissement de cette bibliothèque est une perspective de ce travail.

Nous n'avons pas la prétention de résoudre le problème de l'évaluation de la fiabilité d'un NCS. Ce problème est loin d'être résolu par des approches analytiques tant sa complexité est importante. Nous proposons une méthode et les outils associés pour approcher cette évaluation par simulation et ainsi apporter une aide à la conception des systèmes satisfaisant à des exigences critiques sur certains paramètres de performance.

Conclusion générale

Le travail a présenté une démarche pour l'évaluation de la fiabilité dans les systèmes commandés en réseau (SCR). L'introduction de boucles de commande dans les systèmes est souvent destinée à réduire leur sensibilité à certaines perturbations sur les signaux ou les données manipulés et cela leur confère assez souvent des propriétés de tolérance à certaines fautes de leurs composants. Nous pensons notamment aux fautes transitoires qui apparaissent en particulier dans les réseaux de transmission. On ne peut alors plus définir la fiabilité d'un système sur le modèle binaire marche ou panne. Si on reprend la définition de la fiabilité « aptitude d'un système à assumer une fonction requise sur un intervalle de temps donné », on voit la nécessité de définir la fonction requise. Lorsqu'on introduit une boucle de commande dans un système, l'objectif est d'obtenir un nouveau système défini par des objectifs de performances : stabilité, temps de réponse, dépassement, erreur statique… La présence de défaillances fugitives va entraîner des dégradations plus ou moins fortes sur ces performances risquant dans certains cas de provoquer des dégâts irréversibles (endommagement ou destruction de composants, rejets d'effluents, etc.). On doit donc imposer une limite à ne pas dépasser pour les perturbations temporaires de performances dues à ces défaillances fugitives. Le problème de la fiabilité est donc de calculer la probabilité pour que les performances du système restent dans des limites acceptables au-delà desquelles le risque est intolérable. A l'image de ce qui fut proposé par [Hongbin Li et Qing Zhao, 2005], la fiabilité devient un vecteur de probabilités pour que chaque indicateur de performance reste en deçà des limites tolérables. Pour chacune de ces probabilités, on peut définir les événements perturbateurs ou les combinaisons d'événements perturbateurs (coupes). Il n'est cependant pas possible de donner une expression analytique de ces probabilités car elles dépendent non seulement des probabilités d'occurrence des défaillances fugitives mais aussi des seuils fixant la limite d'acceptabilité et de la dynamique fonctionnelle du système. C'est la raison pour laquelle nous avons proposé une approche par simulation.

Dans un premier temps, nous avons cherché à identifier l'incidence de deux types de défaillances fugitives : la perte d'un échantillon d'une part et le retard d'un échantillon dans la boucle de régulation (sans considérer la présence d'un réseau) d'autre part. Nous réalisons une simulation de Monte Carlo sur la réponse à un échelon de consigne du système en injectant aléatoirement la perte du message ou en faisant varier aléatoirement le retard à la délivrance des informations entre capteur et contrôleur et contrôleur et actionneur simultanément [Ghostine et al, 2007b]. Nous plaçons des détecteurs sur les seuils de performances acceptables : temps de montée, temps de réponse, dépassement, tendance instable. Les moyennes sur les nombres de franchissement de ces seuils nous permettent d'approcher les probabilités recherchées. Le critère d'arrêt de la simulation est lié à une probabilité d'appartenance à un intervalle de confiance donné sur le résultat.

Enfin, nous simulons le comportement en présence des deux types de perturbations simultanément, mettant en évidence des effets cumulatifs. Cela peut être rapproché du concept de coupe. En effet, les défaillances injectées indépendamment peuvent ne pas avoir un effet inacceptable, par contre simultanément il est possible qu'elles produisent un effet inacceptable. Cependant, contrairement à ce que l'on a coutume de dire dans les systèmes statiques, la probabilité de non défaillance du système n'est pas le produit des probabilités de non occurrence des défaillances. Cette relation ne peut en l'état actuel des connaissances qu'être évaluée par simulation.

Si on tient compte maintenant du fait que l'origine des pertes ou retards est due à la présence du réseau, il faut l'introduire dans le modèle. Le caractère aléatoire n'est plus associé directement à la défaillance mais plutôt à la cause qui est susceptible de la produire. On introduit alors dans le modèle la modélisation SAN d'un réseau CAN et l'injection des défaillances dans celui-ci. Le même type de simulation sur la réponse indicielle du système a été effectué, ainsi que l'enregistrement des dépassements de limites acceptables des performances [Ghostine et al, 2006].

Les performances d'un réseau et notamment sa sensibilité aux sources de défaillance sont tributaires de sa charge et de la nature du protocole. Afin de mettre cela en évidence, nous avons simulé un système dans lequel est pris en compte le trafic circulant sur le réseau [Ghostine et al, 2008].

Nous n'avons pas la prétention de résoudre le problème de l'évaluation de la fiabilité d'un SCR. Ce problème est loin d'être résolu par des approches analytiques tant sa complexité est importante. Nous avons proposé une méthode et les outils associés pour approcher cette évaluation par simulation et ainsi apporter une aide à la conception des systèmes satisfaisant à des exigences critiques sur certains paramètres de performance.

Grâce à la simulation de Monte-Carlo, l'approche proposée a permis de quantifier l'influence des fautes transitoires sur la fiabilité. Cette quantification permet de répondre à diverses questions comme par exemple la satisfaction d'un niveau de fiabilité d'un SCR dans un milieu donné ou l'influence d'un trafic externe partageant le même réseau sur la fiabilité d'un SCR. Cette étape de quantification est une étape incontournable avant la certification des installations distribuées autour d'un réseau de communication.

Cette étude de la fiabilité dans un système commandé en réseau a pu être mené grâce à la puissance de modélisation des SAN. La possibilité de joindre plusieurs modèles de base pour former un modèle composé dans *Möbius* et la possibilité d'utiliser des paramètres nous ont permis de créer des modèles de base paramétrés. Actuellement, un certain nombre de composants de bases définis peuvent se retrouver dans n'importe quel système commandé en réseau. L'extension de cette bibliothèque en ajoutant d'autres types de réseaux et d'autres types de composants seront des perspectives de ce travail. Le but est d'aboutir à un outil permettant à un concepteur de :

- Comparer, vis-à-vis de la sûreté de fonctionnement, des architectures qui diffèrent par exemple par le type de réseau de communication, l'algorithme de commande ou les stratégies d'ordonnancement ou de priorités des messages et tâches.

- Donner des conseils sur le trafic externe maximum permis, tout en vérifiant des contraintes sur les paramètres de la sûreté de fonctionnement.
- Certifier des systèmes commandés en réseau fonctionnant dans des milieux fortement pollués ou sur des réseaux ouverts.

Les seuls types de fautes que nous avons considérées sont les fautes de transmissions engendrées par des perturbations extérieures. Quelques hypothèses sur la détection des messages altérés ont été prises pour simplifier la modélisation. Il sera intéressant de revoir ces hypothèses. Par exemple l'hypothèse que tous les messages altérés sont détectés par les mécanismes de détection de fautes mérite d'être approfondie. D'autres fautes transitoires sur d'autres composants peuvent être encore étudiées (par exemple, les fautes transitoires lors des reconfigurations de commande de systèmes en redondance passive, etc.).

Le partage de ressources, l'existence de système d'exploitation multi-tâches sur les processeurs peuvent encore engendrer des retards supplémentaires. Cet aspect ainsi que les influences des algorithmes d'ordonnancement, dans une approche de co-design, sont autant de perspectives de ces travaux.

Cependant, un des aspects les plus intéressants à développer pour l'aide à la conception de SCR est sans doute la recherche de facteurs d'importance spécifiques à ces systèmes. En fiabilité classique, les facteurs d'importances définissent la sensibilité de la fiabilité du système à certains facteurs liés au composant ou à la structure. Ici nous avons vu que la fiabilité dépend de paramètres fonctionnels (pôle et zéros de la fonction de transfert…), par exemple il sera intéressant de rechercher la sensibilité de la fiabilité à un coefficient de correcteur dont l'augmentation peut améliorer la dynamique du système mais dégrader sa fiabilité ; un compromis doit être recherché.

Il reste qu'un des points faibles de l'approche demeure le temps nécessaire pour la simulation de Monte Carlo. Nous n'avons pas tenté d'améliorer des outils standard disponibles. Il est vraisemblable que des efforts pourraient être faits dans ce sens (optimisation de logiciel, architectures parallèle…)…

Glossaire

AMDEC : Analyse des modes de défaillance, de leurs effets et de leur criticité

AN: Activity Network

CAN : Controller Area Network

CPN: Colored Petri Net

CRC : Contrôle de redondance cyclique

CSD : Control system deadline

CSMA/CD: Carrier Sense Multiple Access with Collision Detection

CSMA/CA : Carrier Sense Multiple Access with Collision Avoidance

MTTFF : Mean Time To First Failure

NCS : Networked Control system

PCSMA : Predictable Carrier Sense Multiple access

RA : Réseau d'activité

RAS : Réseau d'activité stochastique

RASC : Réseau d'activité stochastique coloré

RB : Réseaux Bayésiens

RDP PTD : Réseau de Petri Prédicats Transitions Différentiels

RPS : Réseau de Petri Stochastiques

RPSG : Réseau de Petri Stochastiques Généralisés

SAN : Stochastic Activity Network

SCR : Système commandé en réseau

SdF : Sûreté de fonctionnement

Bibliographie

[Ajmone et al, 1984] M. Ajmone, G. Conte, G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transaction on computer systems*. Vol (2), p:93-122 (1984)

[Alur et al, 1995] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.H. Ho, X. Nocollin, A. Olivero, J. Sifakis, S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer science*, n°138, (1995)

[Anderson et al, 2005] M. Andersson, D. Henriksson, A. Cervin. Truetime 1.3 Reference Manual, Department of automatic, Lund institute of technology, Sweden, (2005)

[Andref, 1994] J. Andref. Robustness to jitter in real-time systems. Technical Report ISRN LUTFD2/TFRT655076SE, Dept of Automatic Control, Lund Institute of Technology Sweden, (1994)

[Aström et Hagglund, 1984] K.J. Aström, and T. Hagglund. Automatic tuning of simple regulators. *In Preprints 9th IFAC WorId Congress*. p:267-272, Budapest, Hungary (1984)

[Aström et Wittenmark, 1990] K.J. Aström et B. Wittenmark. Computer-controlled systems, theory and design. Prentice hall, Englewood cliffs, New Jersey 07632 (1990)

[Aubry et Zanne, 1991] J-F. Aubry et C. Zanne. Intégration de la sûreté de fonctionnement dans la conception des systèmes de commande numérique des processus électroniques. *APII, AFCET- Sûreté de fonctionnement.* Vol (25), p:297-324. (1991)

[Babak, 2003] A.S. Babak. Stability of networked control systems in the presence of packet losses. *In: Proceedings of the IEEE Conference on Decision and Control.* p:676-681, USA, (2003)

[Barger, 2003] P.Barger. Evaluation et validation de la fiabilité et de la disponibilité des systèmes d'automatisation à intelligences distribuée en phase dynamique. Thèse de doctorat, université Henri Poincaré, Nancy 1, 15 décembre (2003)

[Berbra et al, 2007] C. Berbra, S Gentil, S Lesecq, J-M Thiriet. Co-design for a safe networked control DC motor. 3rd International IFAC workshop on networked control systems tolerant to faults, NeCST, Nancy France (2007).

[Blum et Juanole, 1999] I. Blum and G. Juanole. Comparing the networks CAN and ARINC 629 CP with respect to the quality of the service provided to an automatic control application. *In Fieldbus and technologie Conference*. September 1999.

[Bobio et al, 2003] A.Bobbio, E.Ciancamerla, G. Franceschinis, R.Gaeta, M.Minichino, L.Potrinale. Sequential application of heterogeneous models for the safety analysis of a control systems : a case study. *Reliability engineering and system safety*. Vol (81)-3, p: 269-280 (2003)

[Boudali et Dugan, 2005] H.Budali, J.B.Dugan. A new Bayesian network approach to solve dynamic fault trees. *IEEE reliability and maintainability Symposium*. p451-456, January 24-27,(2005).

[Bouissou et Bon, 2003] M.Bouissou, J.L. Bon. A new formalism that combines advantages of fault tree and Markov models : Boolean logic driven Markov processes. *Reliability Engineering and system safety*. Vol (82)-2, p:149-164 (2003)

[Bouissou et Dutuit, 2004] Reliability analysis of a dynamic phased mission system. *MMR 2004 Mathematical Methods in Reliability*. Santa Fe, (USA), June (2004)

[Bouissou et Dutuit, 1996] M. Bouissou et Y. Dutuit. Two Approaches to Analyse a Sequential, non Coherent and Looped System. *ESREL 96*. Crète, juin, (1996)

[Brahimi, 2007] B. Brahimi. Proposition d'une approche intégrée basée sur les réseaux de Petri de haut niveau pour simuler et évaluer les systèmes contrôlés en réseau. Thèse de doctorat, université Henri Poincaré, Nancy 1, 5 décembre (2007)

[Broster el al,2004] I. Broster, A. Burns, and G. Rodriguez-Navas. Comparing real-time communication under electromagnetic interference, *In Proceedings of the 16th Euromicro Conference on Real-Time systems*, *IEEE Computer Society*. p:45_52, Catania, Italy (2004)

[Buisson, 1993] J. Buisson. Analysis of switching devices with Bond Graphs. *Journal of the Franklin Institute*. Vol (330), n°6, p:1165-175, (1993)

[Cabarbaye et Laulheret, 2005] A. Cabarbaye et R. Laulheret. Evaluation dynamique de la sûreté de fonctionnement des systèmes dynamiques par modélisation récursive. *6ème congrès international pluridisciplinaire, qualité et sûreté de fonctionnement, Qualita.* Bordeaux, France (2005)

[CAN, 1990] Controller Area network CAN, an In-vehicle Serial Communication Protocol. SAE Handbook 1992 SAE Press. p:20.341-20.355.

[Carvalho et Portugal, 2001] A. Carvalho, P. Portugal. On dependability evaluation of fieldbus networks: a permanent fault analysis. *Industrial Electronics Society, 2001. IECON apos;01, The 27th Annual Conference of the IEEE.* Vol (1), p:299-304 (2001)

[Castillo et al, 1997] E.Castillo C. Solares, P.Gomez. Tail uncertainty analysis in complex systems. Artificiel intelligence. Vol (96), p:395-419, 1997.

[Cauffriez, 2004] L.Cauffriez, J.Ciccotelli, B.Conrard, M.Bayart . Design of intelligent distributed control systems: dependability point of view, *Reliability Engineering And System Safety.* Vol (4), p:19-32 (2004)

[Champagnat, 1998] R. Champagnat. Supervision des systèmes discontinus: définition d'un modèle hybride et pilotage en temps-réel, Thèse de Doctorat de l'Université Paul Sabatier de Toulouse, 1er octobre (1998)

[Chiueh 1994] Chiueh, T. et Venkatramani C. (1994). Supporting real-time traffic on Ethernet. *15th IEEE Real-Time Systems Symposium (RTSS'94)*.p:282–286, San Juan, Puerto Rico, (2003)

[CIAME, 1999] Réseaux de terrain. Description et critères de choix. CIAME. Editions Dunod. (1999)

[Ciardo et al, 1991]G. Ciardo, J. K. Muppala, and K. S. Trivedi. On the solution of GSPN reward models. *Performance Evaluation*. Vol (12)-4, p:237–253, (1991)

[Ciardo et al, 1991]G. Ciardo, J. K. Muppala, and K. S. Trivedi. A decomposition approcah for stochastic reward network models. *Performance evaluation.* Vol (18)-1, p:37-59, (1993)

[Cocozza et al, 2004]C. Cocozza-Thivent, R. Eymard, S. Mercier : A Numerical Scheme to Solve Integro-Diffrential Equations in the Dynamic Reliability Field, *PSAM7-ESREL'04.* Berlin, juin (2004)

[Conrard, 1999] B. Conrard, Contribution à l'évaluation quantitative de la sûreté de fonctionnement des systèmes d'automatisation en phase de conception, thèse de doctorat, CRAN, Université H. Poincaré, Nancy 1, (1999)

[Cruz, 1991] Cruz. A calculus for network delay, part I: network elements in isolation. *IEEE Transactions on Information Theory*. Vol (37)-1, p:114–141.

[David et Alla, 1989] R. David et H. Alla. Du Grafcet aux réseaux de Petri, Hermes, (1989)

[David et Allah, 1997] R. David, H Allah. Du Grafcet au réseau de Petri, deuxième édition revue augmentée, Editions Hermès, Paris, (1997)

[Devooght et Smidts, 1994] J. Devooght, C. Smidts. A theoretical analysis of DYLAM-type event tree sequences. *Proceedings of PSAM-II*. Vol (1), p:011.1-011.6, (1994)

[Doulliez et Jarnoulle, 1972] P.Doulliez, E.Jarnoulle. Transportation networks with random arc capacities. R.A.I.R.O. p:45-59 (1972)

[Dugan et al, 1992] J.B. Dugan, S.J. Bavuso and M.A. Boyd, Dynamic fault-tree models for computer systems. *IEEE Transactions on Reliability*. Vol (41), p:363-377, (1992)

[Dutuit et al, 1997] Y. Dutuit et al. Dependability modelling and evaluation by using stochastic Petri nets: application to two test cases. *Reliability Engineering and System Safety*. Vol (55), n°2, p:117-124, (1997)

[Florin et al, 1979] G. Florin, S. Natkin, P. Lonc. An evaluation CAD tool based on stochastic Petri nets, *IFIP conference on reliable computing*. Londres, (1979)

[Florin et Natkin, 1985] G. Florin et S. Natkin. Les réseaux de Petri Stochastiques. TSI. Vol. (4), No 1, Fév. (1985)

[Galdun et al, 2007] J.Galdun, R. Ghostine, J.M. Thiriet, J. Ligus, J. Sarnovsky. Definition and modelling of the communication architecture for the control of a helicopter-drone. *8th IFAC Symposium on cost oriented automation*, habana (cuba), February (2007)

[Gallara et Decotignie, 1984] D.Gallara,. et J.-D. Decotignie. Groupe de réflexion FIP : proposition d'un système de transmissions série multiplexée pour les échanges entre des

capteurs, des actionneurs et des automates réflexes. Livre blanc, Ministère de l'Industrie (1984)

[Georges, 2005] J.P. Georges. Systèmes contrôlés en réseau : Evaluation de performance d'architectures Ethernet commutées. Thèse de doctorat, université Henri Poincaré, Nancy 1, 4 novembre (2005)

[Georges et al, 2005] J.P. Georges, T.Divoux, E.Rondeau. Strict priority versus weighted fair queuing in switched ethernet networks for time critical applications. 19th IEEE international parallel and distributed processing symposium (IPDPS'05), (2005)

[Ghostine et al, 2006] R.Ghostine, J.M. Thiriet, J.F. Aubry. Dependability evaluation of networked control systems under transmission faults. *6th IFAC on fault detection, supervision and safety of technical processes, SafeProcess*. p:1129-1134, Beijing (china), August 29 September (2006)

[Ghostine et al, 2007a] R.Ghostine, J.M. Thiriet, J.F. Aubry. Influence of transmission faults on the dependability level of networked control systems: application to a control loop, *8th IFAC Symposium on cost oriented automation*, habana (cuba), February (2007)

[Ghostine et al, 2007b] R.Ghostine, J.M. Thiriet, J.F. Aubry. How lost messages can affect the dependability of a networked control system. Q*ualita.* Maroc Mars (2007)

[Ghostine et al, 2008**]** R.Ghostine, J.M. Thiriet, J.F. Aubry, M. Robert. A framework for the reliability evaluation of networked control systems. To appear in the *17th IFAC World Congress*. July 6-11, 2008, Seoul, Korea.

[Goodwin et al, 2001] G. Goodwin, S. Graebe, and M. Salgado. *Control System Design*. Prenctice Hall International (2001)

[Halevei et Ray, 1990] Y. Halevei and A.Ray, Performance Analysis of Integrated Communication and Control Systems Networks, *In ASME, Journal of Dynamic Systems, Measurement and Control*. Vol (112), p:365-632, (1990)

[Halevi et Ray, 1988] Y. Halevi and A. Ray, Integrated communication and control systems: part I- Analysis. *In Journal of Dynamic systems, Measurment and Control*. Vol (110), p:367-373. (1988)

[Hamidi, 2005] K. Hamidi. Contribution a un modèle d'évaluation quantitative des performances fiabiliste de fonctions électroniques et programmable dédiées à la sécurité. Thèse, Institut national polytechnique de lorraine, 27 Octobre, (2005).

[Hansson et al, 2002] H. Hansson, T. Nolte, C. Norstr om, and S. Punnekkat. Integrating Reliability and Timing Analysis of CAN-based Systems. *IEEE Transaction on Industrial Electronics*. Vol (49), n°6, December (2002)

[Hongbin Li et Qing Zhao, 2005] L. Hongbin, Z. Qing. A Cut/Tie Set Method for Reliability Evaluation of Control Systems. *American Control Conference*. USA (2005).

[Iung,2002] B. Iung. Contribution à l'Automatisation des Systèmes Intelligents de Production Interopérabilité des Processus de Contrôle, Maintenance et Gestion technique. Habilitation à diriger des recherches de l'université Henri Poincaré, Nancy, 17 Décembre (2002)

[Jensen, 1997] K. Jensen. Coloured Petri Nets. Basic concepts, Analysis methods and practical use. *Theoretical computer science, springer verlag*, 2nd corrected printing (1997)

[Juanole et Blum, 1999] G. juanole and I. blum. Influence des fonctions de base (communication ordonnancement) des systèmes distribués temps réel sur les performances d'application de contrôle-commande. *7émé colloque francophone sur l'ingénierie des Protocoles CFIP)*, (1999)

[Juanole et Moiney, 2005] G. Juanole and G. mouney. QOS in Real Time Distributed Systems and Process Control Applications. *In Proc. Of Workshop on Networked Control Systems (NeCST)*. European Project, Corsica, France, October (2005)

[Jumel, 2005] F. Jumel. Définition et gestion d'une qualité de service pour les applications temps réel. PhD thesis. Institut National Polytechnique de lorraine (2005)

[Kaufmann et al, 1975] Kaufmann A., Grouchko D., Cruon R., *Modèles mathématiques pour l'étude de la fiabilité des systèmes, Masson*, (1975)

[Kermisch et Labeau, 2002] C. Kermisch et P.E. Labeau. Approche dynamique de la fiabilité des systèmes. Rapport MNFD 2002-10, Service de Métrologie Nucléaire, Université Libre de Bruxelles (Belgique), novembre (2002)

[Khadiri, 1992] M. Khadiri. Evaluation directe et par simulation d'indices de fiabilité de réseaux de communication. Thèse de l'université de Rennes1, (1992)

[Khalfaoui, 2003] S. Khalfaoui. Méthode de recherche des scénarios redoutés pour l'évaluation de la sûreté de fonctionnement des systèmes mécatroniques du monde automobile. Thèse de Doctorat, Institut National Polytechnique, Toulouse, 26 septembre (2003)

[Kim et shin, 1993] H. Kim, K.G. Shin. Modelling of externally-induced/common-cause faults in fault-tolerant systems. *IEEE/AIAA Digital Avionics System Conference*. p:402-407,(1994)

[Kocick et sorel, 2000] R. Koicick and Y. Sorel. De la modélisation à la réalisation : réduction du cycle de développement des applications temps réel distribuées. *In Proceedings of the 8th conference on Real-time and embedded systems*. Paris, France, Mars 2000.

[Koptez, 1998] H. Koptez. A comparison of CAN and TTP. Technical Report, Techniste Universitat Wien, Austria, (1998)

[Kosgue et al. 1996] K. Kosuge, H.Murayama and K.Takeo. Bilateral Feedback Control of Telemanipulators via Computer Network. *Conf. on Intelligent Robots and Systems (IROS'96).* P:1380-1385, Osaka, Japon, November 4-8, (1996)

[Koubâa and Song 2004] A. KOUBAA, Y-Q. SONG, (m,k)-WFQ. Integrating (m,k)-Firm Real-Time Constraints into Guaranteed-Rate Networks. *Proceedings Real Time Systems Conference RTS'2004*. Paris (France) 30 Mars-1 Avril 2004.

[Koubâa, 2004] A. Koubaâ, Gestion de la Qualité de Service temporelle selon la contrainte (m,k)-firm dans les réseaux à commutation de paquets, LORIA-TRIO,thèse octobre 2004.

[Labeau, 1998] P.E. Labeau. A Survey on Monte Carlo Estimation of Small Failure Risks in Dynamic Reliability. *International Journal of Electronics and Communications*. Vol (52), p :205-211, (1998)

[Labeau, 2003] P.E. Labeau. Evolution de la modélisation en fiabilité dynamique. *Journée Fiabilité dynamique et simulation hybride*. ENSAM, PARIS, 12 mai (2003)

[Laprie et al, 1995] J-C. Laprie et al. Guide de la sûreté de fonctionnement. Toulouse : Cépaduès, (1995)

[Lee, 1994] Y. Lee, Y.Kim, and W. Kwon. Generalized predictive control for time-varying systems. In Proceedings of the Asian Control Conference, Tokyo, Japon, Juillet (1994)

[Lee, 2001] D. Lee, J. Alllan, H.A. Thompson, S. Bennett. PID control for a distributed system with a smart actuator. *Control Engineering Practice 9*. p:1235–1244, (2001)

[Lee, 2002] K.C. Lee, S. Lee. Performance evaluation of switched Ethernet for real-time industrial communications. *Computer Standards & Interfaces.* Vol (24), p : 411– 423, (2002)

[Lelevé 2000] Lelevé A. Contribution à la téléopération de robots en présence de délais de transmission variable. Thèse préparée à l'Université de Montpellier II, Science et Technique du Languedoc, décembre (2000)

[Lian et al, 2001] F.L. Lian,, R. James, M.Tilbury. Performance evaluation of control networks. Ethernet, ControlNet, and DeviceNet. *IEEE Control systems magazine February.* (2001)

[Lian, 2001] Feng-Li Lian. Analysis, Design, Modelling, and control of Networked control Systems. Ph.D dissertation University of Michigan, (2001)

[Locks et Satyanarayana, 1986] M.O. Locks and A. Satyanarayana. Network reliability the sate of the art, *IEEE Transaction on Reliability*. Vol (35), (1986)

[Lui et Lee, 2003] D.lui et al. AN efficient scheduling discipline for packet switching networks using earliest deadline first round robin. *12th conference on computer communications and networks ICCCN*.Vol (20)-22, p:5-10, (2003)

[Marsal et al, 2006] G.Marsal, B.Denis, J-M. Faure, G. Fery. Evaluation of response time in ethernet-based automation systems. *6th IEEE international workshop on factory communication systems, WFCS'06*. p:95-98, June (2006)

[Marti, 2001] P.Marti, R Villa, J.M. Fuertes, and G. Fohler. Control loop performance analysis over networked control. *In proceedings of the 28th Annual Conference of the IEEE Industrial Electronics society*. P:39-48, Sevilla, Espagne, Novembre (2002)

[Martin, 1995] M. Torngren. Modelling and design of distributed real-time control application. Master's thesis, Royal Institute of technology, KTH, Sweden, (1995)

[Mounnin, 2007] M. Mounnin. Approche unifiée défaillance/dommage dans la sûreté de fonctionnement pour la génération des matériels au combat, application aux systèmes d'armes terrestres. Thèse de doctorat, université de Valenciennes et du Hainaut Cambrésis, 10 octobre (2007)

[Medjoudj, 2006] M. Medjoudj. Contribution à l'analyse des systèmes pilotés par calculateurs: Extraction de scénarios redoutés et vérification de contraintes temporelles. Thèse de doctorat, université Paul Sabatier, Toulouse, Mars (2006)

[Mogavar et Meyer, 1984] A. Mogavar, J.F. Meyer. Performability modeling with stochastic activity network. *In proc of the 1984 real-time systems Symp.* p:215-224, Austin, TX, USA, (1984)

[Michaut, 2003] F. Michaut. Adaptation des applications distribuées à la Qualité de Service fournie par le réseau de communication. PhD thesis. Université Henri Poincaré, Nancy 1, Centre de Recherche en Automatique de Nancy (2003)
[Moitessier et al, 1991] F. Moitissier et al. Une méthode de conception des systèmes de commande temps réel repartis pour processus physiques rapides et mixtes. *ECC9,*. Grenoble. (1991)

[Moncelet, 1998] G. Moncelet. Application des réseaux de Petri à l'évaluation de la sûreté de fonctionnement des systèmes mécatroniques du monde automobile, Thèse de Doctorat, N°3076, Université Paul Sabatier, Toulouse, 9 octobre (1998)

[Moncelet et al, 1998] Moncelet, G., S. Christensen, H. Demmou, M. Pauldetto and J. Porras (1998). Dependability evaluation of a simple mechatronic system using coloured Petri nets *In: Workshop on Practical Use of Coloured Petri Nets and Design CPN*. P:189-198. Aarhus University, Aarhus, Denmark (1998)

[Moon, 1996]. Y.Moon and K.Kwon. Robust stability of linear systems with structured delayed perturbations. *In proceedings of Sensing instrument control engineering (SICE'96)*. Tottori, japan, juillet (1996)

[Navet et al, 2000] Navet, N., Y. Song and F. Simonot (2000). Worst-Case Deadline Probability in Real-Time Applications Distributed over Controller Area Network. *In: Journal of systems Architecture*. Vol (46), No. 1, p: 607-617, (2000)

[Nilsson 1998] J. Nilsson. Real-time control systems with delays. Ph.D. dissertation, Dept. automatic control, Lund Institute of Technology, Lund, Sweden, January (1998)

[Ogata, 1987] K. Ogata. *Discrete-time Control Systems*. Prenctise Hall, (1987)

[Overstreet 1999] J. W.O. overstreet, A Tzes. An Internet-based real-time control engineering laboratory. IEEE control systems Magazine. Vol (19)-5, p :19-34, (1999)

[Pérez et al, 2007] G.A. Perez Castaneda, J-F. Aubry, N. Brinzei. Etat de l'art en fiabilité dynamique. *JD-JN-MACS*. Reims, France, juillet (2007)

[Poledna, 1995a] S. Poledna. Tolerating sensor timing faults in highly responsive hard real-time systems. *IEEE Transaction on computers*. Vol (44), No 2, p:181-191, February (1995)

[Poledna, 1995b] S. Poledna. Fault tolerance in safety critical automotive applications: cost of agreement as a limiting factor, *25 international conference on a fault tolerant computing (FTCS-25)*. P:73-83, IEEE, june (1995)

[Portugal et Carvalho, 2001] P.J Portugal, A. Carvalho, On dependability evaluation of fieldbus Networks : a permanent fault analysis. *Proceedings of IECON'2001 27th Annual Conference of the IEEE Industrial Electronics Society*. P:299-306, (2001)

[Portugal et Carvalho 2004] P.J. Portugal, and A. Carvalho. A Stochastic Petri Net Framework for Dependability Evaluation of Fieldbus Networks – A Controller Area Network (CAN) Example. *International IEEE Conference in Mechatronics and Robotics – MECROB*, (2004)

[Poulard et al, 2004] G. Polard, B. Denis, J-M. Faure. Modélisation par réseau de Petri Colorés des architectures de commande distribuées sur réseau de terrain Ethernet et TCP/IP. *MOSIM*. Septembre Nantes-France (2004)

[Pritty et al, 1995] Pritty, D.W., J.R. Malone, D.N. Smeed, S.K. Banerjee et N.L. Lawrie. A real-time upgrade for Ethernet based factory networking. *In : 21st IEEE International Conference on Industrial Electronics (IECON'95)*. Vol (2), p:1631–1637, Orlando, (1995)

[Profibus, 1996] General Purpose field communication system – EN50170 Vol (2). PROFIBUS (1996)

[Ravichandran et al, 2007] C.S. Ravichandran, S. Subha Rani and T. Manikandan. Designing of PID Controller for Discrete Time Linear System Using Balanced Approach Reduced Order Model. *American Journal of Applied Sciences.* Vol (4)-3, p:155-159, (2007)

[Ray, 1994] A. Ray. Output feedback control under randomly varying distributed delays. *Journal of Guidance Control and dynamics*. Vol (17)-4, (1994)

[Richard, 2003] J.P. Richards, Time Delay Systems: An overview of some recent advances and open problems. *Automatica*. Vol (39), No. 10, p:1667-1694, October (2003)

[Rondeau et al, 2001] Rondeau, E., T. Divoux et H. Adoud (2001). Study and method of Ethernet architecture segmentation for Industrial applications. *In: 4th IFAC Conference on Fieldbus Systems and Their Applications (FET'2001)*. p: 165–172, Nancy, France, (20001)

[Rushby 2003] J. Rushby, A comparison of bus architectures for safety-critical embedded systems. Thecnical report, Computer Science Laboratory SRI International, (2003)

[Sanders et Meyer, 1991] H. Sanders and J. F. Meyer. Reduced base model construction methods for stochastic activity networks. *IEEE Journal on Selected Areas in Communications, special issue on Computer-Aided Modeling, Analysis, and Design of Communication Networks*. Vol (9)-1, p:25–36, January (1991)

[Sanders et Meryer, 2002] Sanders W.H., John F. Meyer. Stochastic activity networks: formal definitions and concepts. *Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science*. p:315- 343. Springer-Verlag New York, Inc., New York, NY, (2002)

[Sanders et Malhis, 1992] H.Sanders and M.Malhis. Dependability evaluation using composed SAN-based reward models. *Journal on parallel and distributed computing, special issue Petri net models of parallel computers*. Vol (15), No. 3, p. 238-254, July (1992)

[Schoenig, 2004] R. Schoenig. Définition d'une méthodologie de conception des systèmes mécatroniques sûrs de fonctionnement, Thèse de Doctorat de l'Institut National Polytechnique de Lorraine, 26 octobre (2004)

[Sheldon et jerath, 2004] F. Sheldon and K. Jerath. Assessing the Effect of Failure Severity, Coincident Failures and Usage-Profiles. *On the Reliability of Embedded Control Systems. SAC.* p :826-833, (2004)

[Shin et Guo, 1995] G. Shin and L. Guo. Incorporation of System Inertia into Reliability Evaluation of Real-Time Control Systems. *Proceedings of the 34th Conference on Decision & Control New Orleans*, LA - December (1995)

[Siu,1994] N. Siu. Risk assessment for dynamic systems : An overview. *Reliability engineering and system safety*. Vol *(*43), p:43-73 (1994)

[Smith, 1972] C.L. Smith. Digital Computer Process Control. *Intext Educational Publishers*, San Francisco, (1972)

[Tarn, 1998] T.J. Tarn, N. Xi. Planning and control of internet-based teleoperation. *Proceedings of SPIE: Telemanipulator and teleprence technologies*. Vol (3524) , p:189-193. Boston, (1998)

[Thomesse, 1998] J.P Thomesse. A review of the Fieldbus. *Annual reviews in Control*. Vol (22), p:35-45, (1998)

[Tipsuwan, 2001] Y.Tipsuwan, M.Y. Chow. Networked-based controller adaptation based on QoS negotiation and deterioration. *The 27th annual conference of the IEEE industrial electronics society (IECON 99)*. Vol (3), p:1271-1276, San Jos, CA, (2001)

[Tipsuwan, 2002] Y. Tipsuwan, & M.Y. Chow. Gain adaptation of networked mobile robot to compensate QoS deterioration. *The 28th annual conference of the IEEE industrial electronics society (IECON 02)*. Vol (4), p:3146–3151, Sevilla, Spain, (2002)

[Unruh, 1989] J. Unruh, H.J. Mathony and K.H. Kaiser. Error detection analysis of automotive communication networks, Technical report,Robert Bosh GmbH, (1989)

[Walsh et al, 1999] G.C. Walsh, H.Ye, L. Bushnell. Stability analysis of networked control systems. *Proceedings of the 1999 American control conference*. Vol (4), p:2876-2880, San Diego, CA, (1999)

[Wang et al, 2002] Z.Wang, Y-Q. Song, J-M. Chen and Y-X. Sun, Real-Time Characteristics of Ethernet and its Improvements. *The 4th world congress on Intelligent Control and Automation*. Shanghai, China, June 10-14, (2002)

[Weber et Jouffe, 2006] P. Weber, L. Joffe. Complex system reliability modelling with dynamic object oriented Bayesian networks (DOOBN). *Reliability Engineering and system safety*. Vol (91)-2, p:149-162 (2006)

[Wilwert, 2005] C.Wilwert. Influence des fautes transitoires et des performances temps réel sur la sûreté des systèmes X-by-Wire. Thèse de doctorat de L'institut national polytechnique de lorraine, 24 Mars (2005)

[Xia et al, 2006] F. Xia, L.Liu, S.Li,Y.Sun. Integrated feedback scheduling of networked control systems. *Journal of continuous, discrete and impulsive systems.* Series B, (2006)

[Yavatkar 1992] R. Yavatkar, P. Pai et R. Finkel. A reservation-based CSMA protocol for integrated manufacturing networks. Technical Report CS-216-92. Department of Computer Science, University of Kentucky, (1992)

[Yépez et al, 2002] J. Yépez, P. Marti, M. Fuertes. Control loop performance analysis over networked control systems. *IECON2002 28th Annual Conference of the Industrial Electronics Society,* Sevilla, Spain, November, (2002)

[Zhang et al, 2001] W. Zhang, M.S. Branicky and S. M. Philips. Stability of networked control systems. *IEEE Control systems Magazine*. Vol (21)-1, p.84-99, (2001)

[Zwingelstein, 1999] G. Zwingelstein. Sureté de fonctionnement des systèmes industriels complexes. Techniques de L'ingenieure, S(4), (1999)

ANNEXE A : Möbius tool et CPN tools

Dans cette partie on modélisera un système commandé en réseau avec deux outils différents :
- Möbius tool,
- CPN/tools.

Le but est de comparer ces deux outils du point de vue :
- puissance de modélisation,
- évaluation des paramètres de sûreté de fonctionnement,
- temps de calcul.

La figure A.1 montre le système étudié.

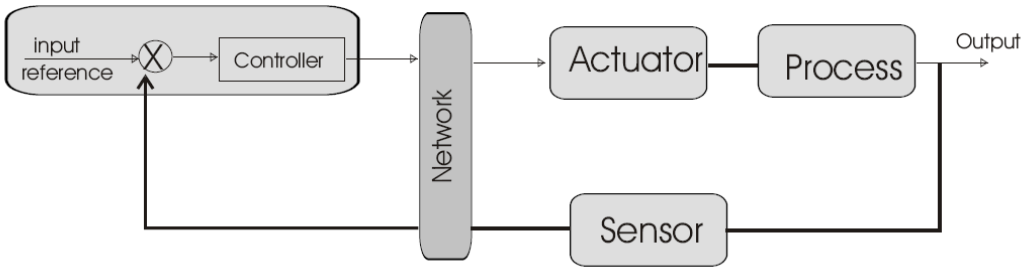


Figure A.1 Système étudié

Le système est composé d'un :
- capteur périodique de période Ts,
- régulateur proportionnel,
- actionneur,
- réseau de communication,
- processus.

Le processus est représenté par ces équations de récurrence :

$$x(k+1)=(9785x(k) +986\ u(k))/1000$$
$$y(k)=x(k)$$

A chaque période d'échantillonnage les actions suivantes se produisent :
- le capteur prélève les mesures du processus,
- le capteur envoie un message via le réseau pour le régulateur,
- à la réception du message envoyé par le capteur le régulateur élabore une nouvelle commande,
- le régulateur envoie un message via le réseau pour l'actionneur,
- l'actionneur reçoit le message et applique la commande sur le processus,

- le réseau se comporte comme une file FIFO, le premier message reçu sera le premier servi. On suppose qu'à chaque période d'échantillonnage Ts, chaque composant peut tomber en panne avec une probabilité P_{def} et que tous les composants ont la même probabilité de défaillance.

A.1 Modèles CPN/tools

A.1.1 Modèle du capteur

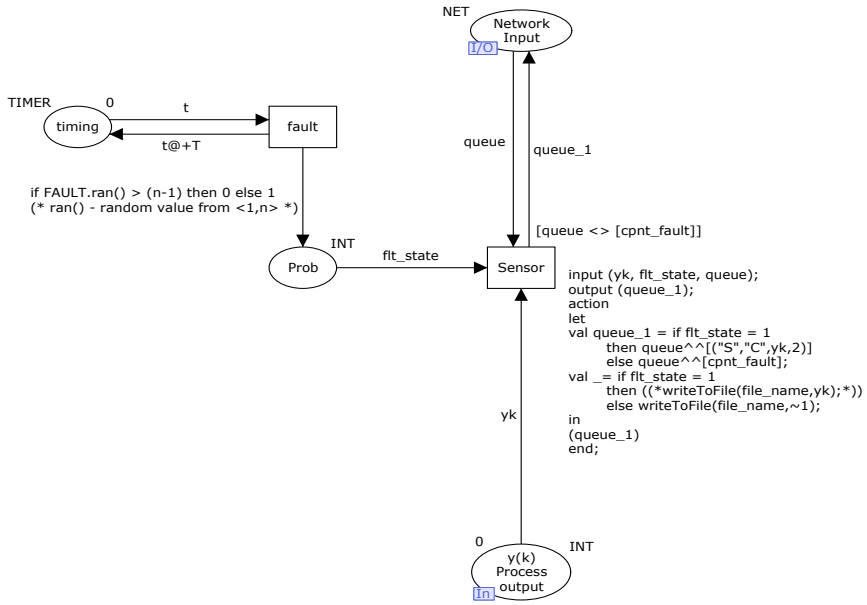


Figure A.2 Capteur CPN/tools

A.1.2 Modèle du régulateur

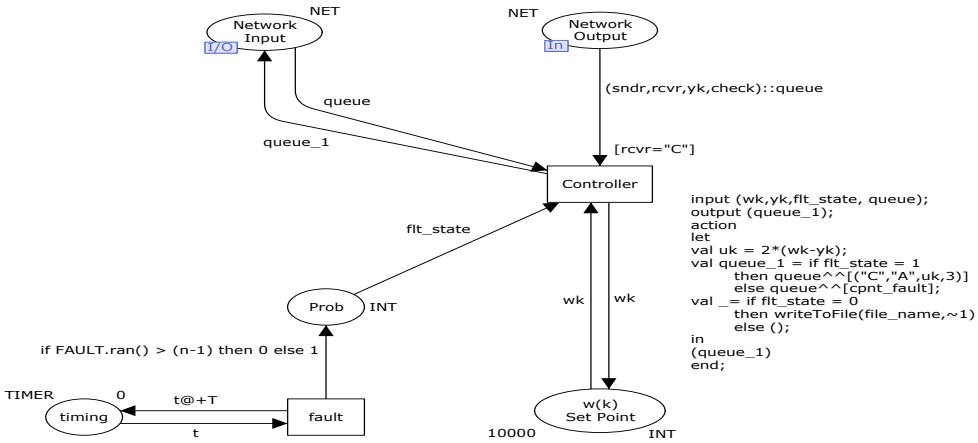


Figure A.3 Régulateur CPN/tools

A.1.3 Modèle de l'actionneur

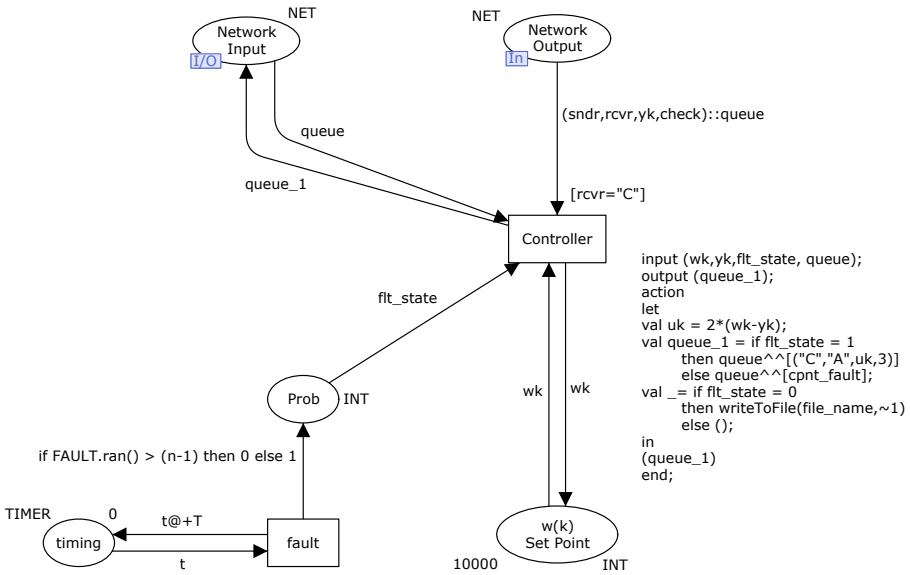


Figure A.4 Actionneur CPN/tools

A.1.4 Modèle du processus

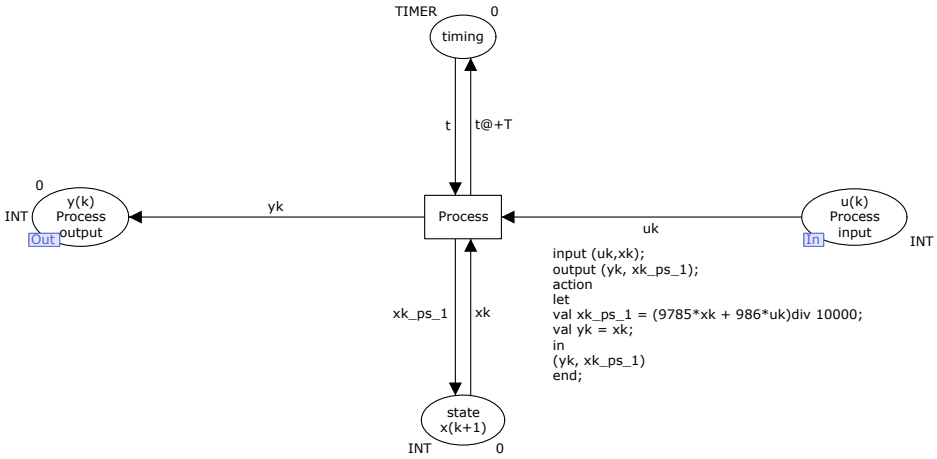


Figure A.5 Processus CPN/tools

A.1.5 Modèle du réseau

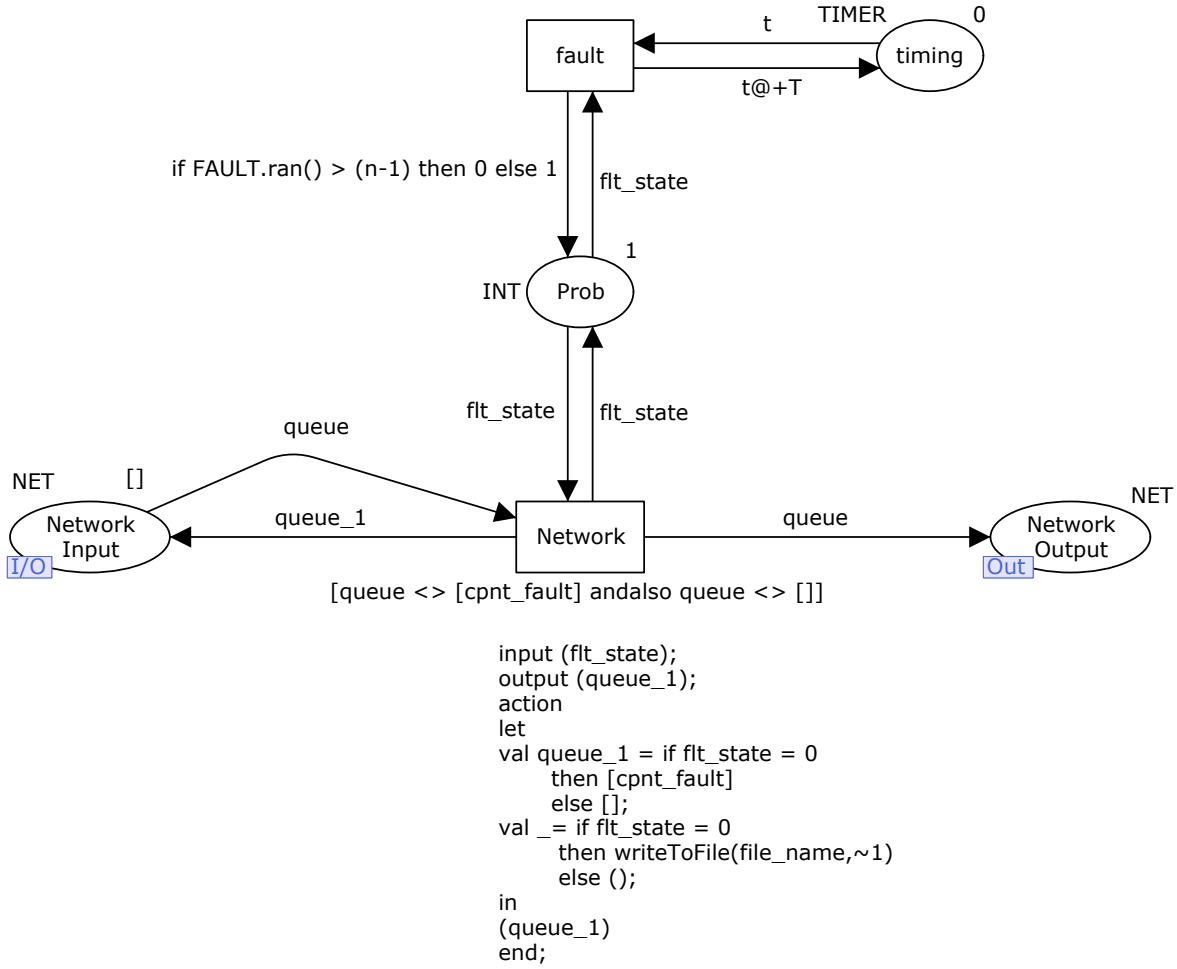


Figure A.6 Réseau CPN/tools

A.2 Modèles Möbius

A.2.1 Modèle du capteur

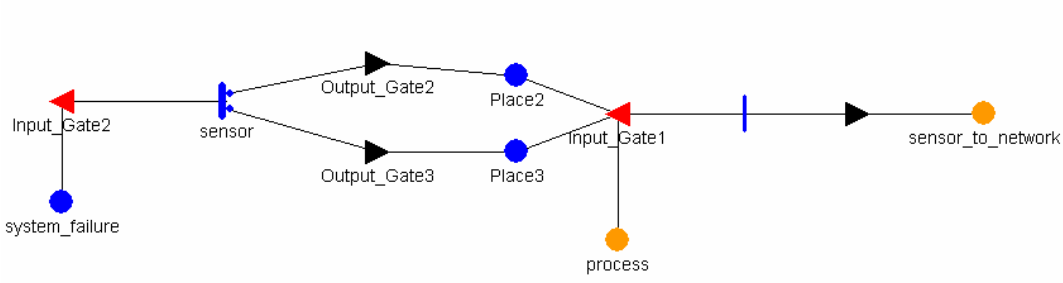


Figure a.7 Capteur Möbius tool

A.2.2 Modèle du régulateur

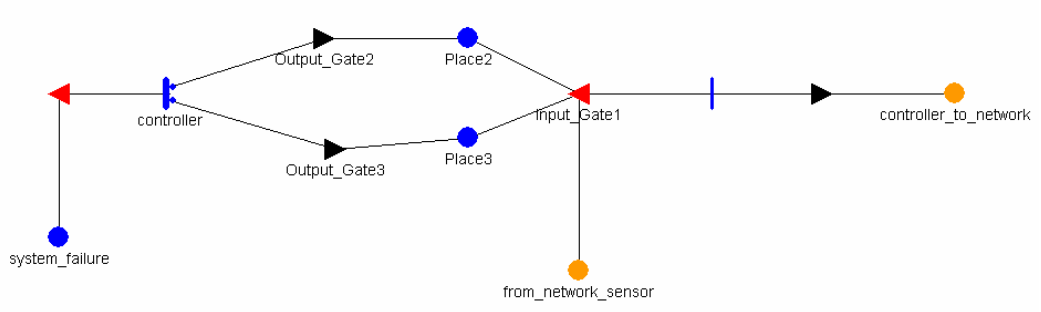


Figure A.8 Régulateur Möbius tool

A.2.3 Modèle de l'actionneur

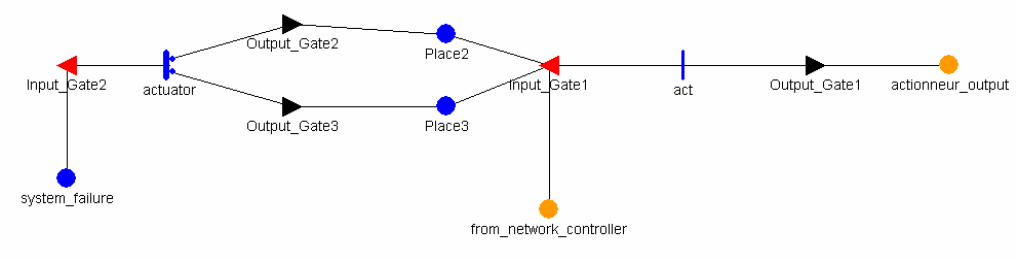


Figure A.9 Actionneur Möbius tool

A.2.4 Modèle du processus

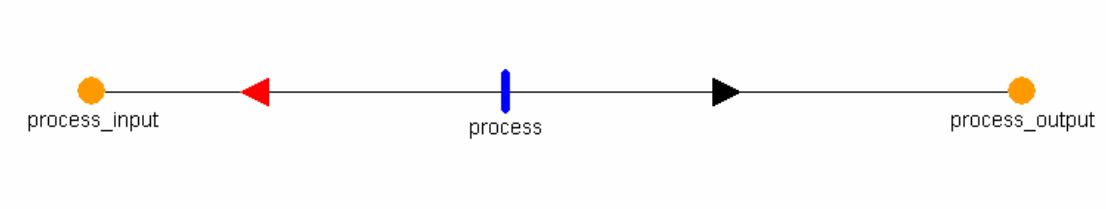


Figure A.10 Process Möbius tool

A.2.5 Modèle du réseau

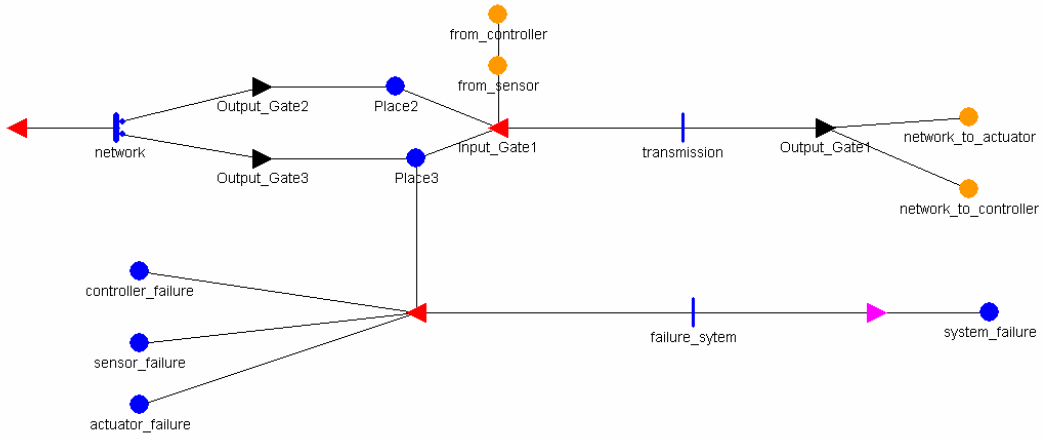


Figure A.11 Réseau Möbius tool

A.3 Comparaison

Le but de la modélisation est la simulation pour évaluer certains paramètres de sûreté de fonctionnement.
Bien que l'on puisse modéliser le même système avec les deux outils, la façon de construire les modèles est tout a fait différente.

Certains points méritent d'être signalés :

CPN/tools ne permet pas de travailler avec des nombres réels directement dans les jetons : pour contourner cette difficulté, une solution sera de transformer l'information contenue dans un jeton en une chaîne de caractère et de créer une équivalence entre les chaînes et des approximations des nombres réels. Dans Möbius, l'utilisation des réels est plus aisée.

CPN/tools est plus orienté vers l'utilisation des réseaux de Petri colorés, l'aspect stochastique est pris en compte par l'ajout d'un code ML.

Möbius/tool est conçu pour l'utilisation des réseaux d'activités stochastiques. Les places étendues permettent l'utilisation des structures définies en C++, jouant le rôle des jetons colorés dans CPN/tools.

L'utilisation des variables est permise dans les deux outils.

La simulation de Monte-Carlo est intégrée dans Möbius, le nombre de simulations à lancer et les critères d'arrêt sont définis grâce à l'option *solver*, et les paramètres à évaluer sont définis dans *reward*. Pour CPN/tools, cette tâche est un peu plus compliquée, il faut sauvegarder un nombre d'informations lors de la simulation dans un fichier et les analyser plus tard pour évaluer les résultats.

Pour les deux logiciels, une série de 1000 histoires a été générée pour évaluer deux paramètres : la fiabilité à t=200*Te et le MTTFF[7] du système.

Le tableau ci-dessous résume quelques points ainsi que les valeurs obtenues:

	Langage de program-mation	Aspect stochastique	aspect coloré	Monte-Carlo	Fiabilité à 200*Te	MTTFF	Temps de simula-tion
CPN/tools	ML	Besoin d'un code supplémentaire	Jeton coloré	Besoin d'un code supplémentaire	0.5531	249	60 s
Möbius	C++	Intégré directement dans la définition d'une transition	place étendue	Intégré directement dans la définition du solver	0.5541	250	24 s

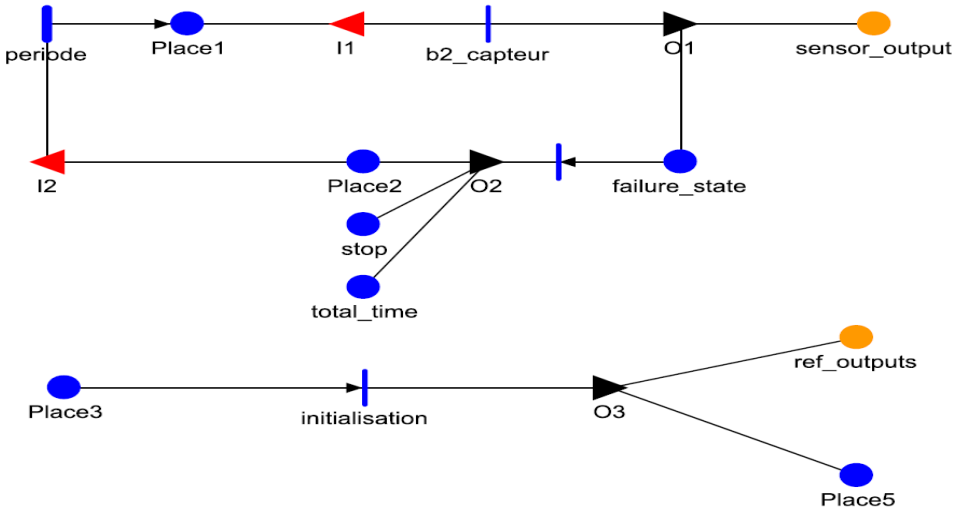
Les deux logiciels sont assez puissants et permettent bien la modélisation des systèmes commandés en réseau. Möbius présente quelques avantages relatifs à la facilité de Modélisation en C++, en particulier l'utilisation de l'approche de Monte-Carlo puisque son temps de calcul est un peu plus court que dans CPN/tools. Pour ces raisons, Möbius est l'outil de Modélisation utilisé dans nos travaux de thèse.

[7] MTTFF : Mean Time to first failure

ANNEXE B : Trois boucles partageant le même medium de communication

Dans cette annexe, on donne les modèles complets de la deuxième boucle (capteur, contrôleur, actionneur) ainsi que le modèle représentant le réseau.

Capteur (boucle 2) :



Place Names	Initial Markings
Place1	0
Place2	0
Place3	1
Place5	0
failure_state	0
ref_outputs	0
sensor_output	0
stop	0
total_time	1

Timed Activity:	période
Distribution Parameters	Value b2Tc1
Activation Predicate	(none)
Reactivation Predicate	(none)

Input Gate:	I1
Predicate	Place1->Mark()==1

Function	Place1->Mark()=0;

Input Gate:	**I2**
Predicate	Place2->Mark()==0
Function	;

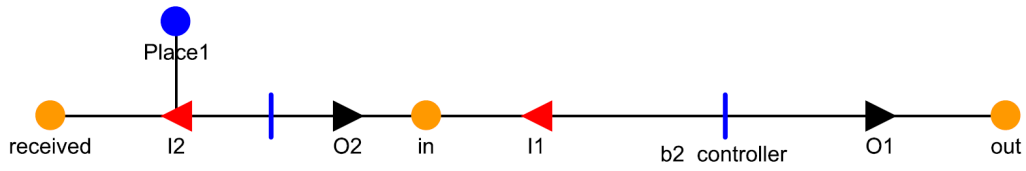
Output Gate:	**O1**

Function	b2_i++;
	if(b2_i==1)
	
	if(b2_i>1)
	{
	b2_xi= b2_ximoin1+ b2_uimoin1*b2_alpha*b2_a*b2_Te
	+(b2_ui-b2_uimoin1)*b2_alpha*(b2_a*(LastActionTime-b2_ti));
	b2_zi= (b2_zimoin1+b2_alpha*b2_uimoin1)*exp(-(b2_a)*b2_Te)-
	b2_ui*b2_alpha+
	b2_alpha*(b2_ui-b2_uimoin1)*exp(-b2_a*(/*b2_Te*/LastActionTime-b2_ti));
	b2_yi=b2_xi+b2_zi;
	
	}
	
	sensor_output->data->Mark()=b2_yi;
	sensor_output->dispo->Mark()++;
	sensor_output->id->Mark()=cmp1P;
	
	if(fabs(b2_yi-ref_outputs->Index(b2_i-1)->Mark())>bound)
	
	{
	failure_state->Mark()++;
	}
	
	b2_ximoin1=b2_xi;b2_zimoin1=b2_zi;b2_uimoin1=b2_ui;b2_yimoin1=b2_yi;

Output Gate:	**O3**
Function	Place3->Mark()--; Place5->Mark()++;

Output Gate:	O2
Function	Place2->Mark()++; stop->Mark()++; total_time->Mark()--;

Contrôleur (Boucle 1) :



Place Names	Initial Markings
Place1	1
in	0
Out	0
received	0

Instantaneous Activities Without Cases:
b2_controller
b2_controllerentree

Input Gate:	I1
Predicate	in->dispo->Mark()==1
Function	in->dispo->Mark()=0;

Input Gate:	I2
Predicate	((received->dispo->Mark()==1)&&(received->id->Mark()==cmp1P))||(Place1-

	>Mark()==1)

Function	if(received->dispo->Mark()!=0)received->dispo->Mark()=0;
	if(Place1->Mark()!=0)
	{
	Place1->Mark()=0;
	//mettre les variables a zero
	b2_i=0;
	b2_yi=0;
	b2_yimoin1=0;
	b2_xi=0;
	b2_zi=0;
	b2_ximoin1=0;
	b2_zimoin1=0;
	b2_ui=0;
	b2_uimoin1=0;
	b2_ti=0;
	b2_reference=0.5;
	b2_datau=0;
	b2_P=0;
	b2_dataDold=0;
	b2_datayold=0;
	}

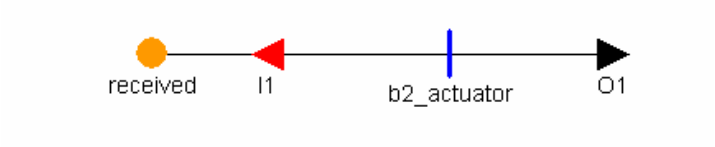
Output Gate:	**O1**

Function	b2_dataK = b2_K;
	b2_dataad = b2_Td/(b2_N*b2_h+b2_Td);
	b2_databd = b2_N*b2_K*b2_Td/(b2_N*b2_h+b2_Td);
	b2_P = b2_dataK*(b2_reference-received->data->Mark());
	b2_D = b2_dataad*b2_dataDold + b2_databd*(b2_datayold-received->data->Mark());
	b2_datau =(b2_P+b2_D);
	b2_dataDold = b2_D;
	b2_datayold = received->data->Mark();
	b2_commande=b2_datau;
	out->data->Mark()=b2_datau;
	out->id->Mark()=cmp2P;
	out->dispo->Mark()++;
	b2_ukmoin1=b2_datau;

Output Gate:	**O2**

Function	b2_temps=LastActionTime/0.2;
	b2_entier =int(b2_temps);
	if(b2_entier%2==1)
	b2_reference=1;
	else b2_reference=0.5;
	
	in->value->Mark()=(b2_reference-received->data->Mark());
	in->dispo->Mark()=1;

Actionneur :

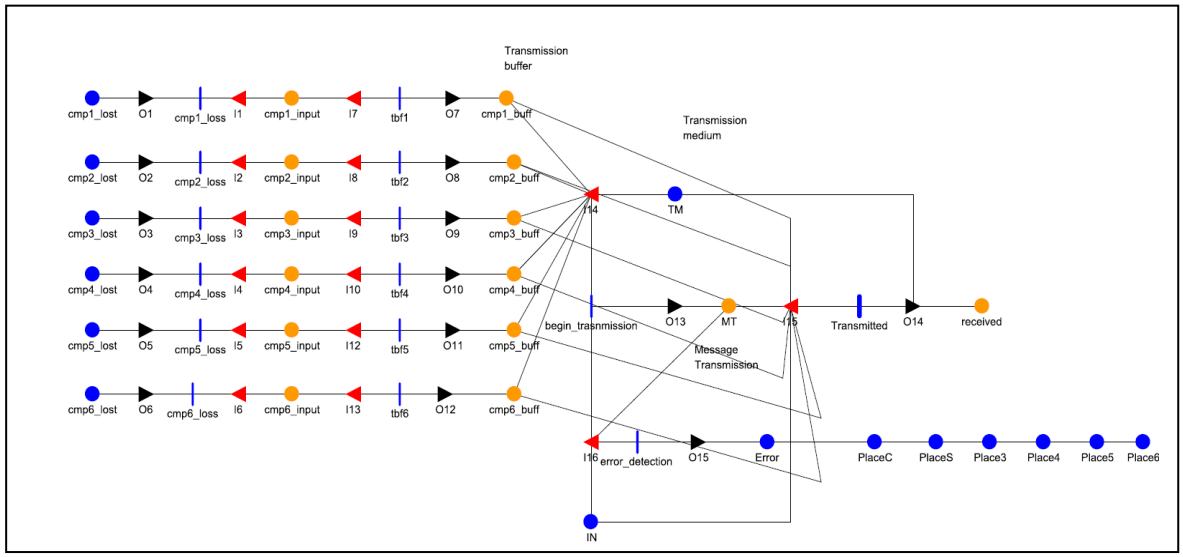


Place Names	Initial Markings
received	0

Input Gate:	I1
Predicate	(received->dispo->Mark()==1)&&(received->id->Mark()==cmp2P)
Function	received->dispo->Mark()=0;

Output Gate:	O1
Function	b2_ui=received->data->Mark(); b2_ti=LastActionTime;

Réseau :



Place Names	Initial Markings
Error	0
IN	0
MT	0
Place3	0
Place4	0
Place5	0
Place6	0
PlaceC	0
PlaceS	0
TM	1
cmp1_buff	0
cmp1_input	0
cmp1_lost	0
cmp2_buff	0
cmp2_input	0
cmp2_lost	0
cmp3_buff	0
cmp3_input	0
cmp3_lost	0
cmp4_buff	0
cmp4_input	0
cmp4_lost	0
cmp5_buff	0
cmp5_input	0
cmp5_lost	0

cmp6_buff	0
cmp6_input	0
cmp6_lost	0
received	0

Timed Activity:	**Transmitted**
Distribution Parameters	Value return TMTrate;
Activation Predicate	(none)
Reactivation Predicate	(none)

Input Gate:	**I1**
Predicate	((cmp1_buff->dispo->Mark()==1))||((cmp1_buff->dispo->Mark()==1)&&(cmp1_input->dispo->Mark()>0))
Function	;

Input Gate:	**I10**
Predicate	(cmp4_input->dispo->Mark()==1)
Function	;

Input Gate:	**I12**
Predicate	cmp5_input->dispo->Mark()==1
Function	;

Input Gate:	**I13**
Predicate	cmp6_input->dispo->Mark()==1
Function	;

Input Gate:	**I14**
Predicate	(((cmp1_buff->dispo->Mark()==1)&&(TM->Mark()==1)&&(IN->Mark()==0))|| ((cmp1_input->dispo->Mark()==0)&& (cmp2_buff->dispo->Mark()==1)&&(TM-

	>Mark()==1)&&(IN->Mark()==0)) || ((cmp1_input->dispo->Mark()==0)&& (cmp2_buff->dispo->Mark()==0)&&(cmp3_buff->dispo->Mark()==1)&&(TM->Mark()==1)&&(IN->Mark()==0)) || ((cmp1_input->dispo->Mark()==0)&& (cmp2_buff->dispo->Mark()==0)&&(cmp3_buff->dispo->Mark()==0)&&(cmp4_buff->dispo->Mark()==1)&&(TM->Mark()==1)&&(IN->Mark()==0)) || ((cmp1_input->dispo->Mark()==0)&& (cmp2_buff->dispo->Mark()==0)&&(cmp3_buff->dispo->Mark()==0)&&(cmp4_buff->dispo->Mark()==0)&&(cmp5_buff->dispo->Mark()==1)&&(TM->Mark()==1)&&(IN->Mark()==0)) || ((cmp1_input->dispo->Mark()==0)&& (cmp2_buff->dispo->Mark()==0)&&(cmp3_buff->dispo->Mark()==0)&&(cmp4_buff->dispo->Mark()==0)&&(cmp5_buff->dispo->Mark()==0)&&(cmp6_buff->dispo->Mark()==1)&&(TM->Mark()==1)&&(IN->Mark()==0))) && (received->dispo->Mark()==0)
Function	;

Input Gate:	**I15**
Predicate	((cmp1_buff->dispo->Mark()==1)&& (MT->dispo->Mark()==1)) || ((MT->dispo->Mark()==1)&& (cmp2_buff->dispo->Mark()==1)) || ((MT->dispo->Mark()==1)&& (cmp3_buff->dispo->Mark()==1)) || ((MT->dispo->Mark()==1)&& (cmp4_buff->dispo->Mark()==1)) || ((MT->dispo->Mark()==1)&& (cmp5_buff->dispo->Mark()==1))
Function	;

Input Gate:	**I16**
Predicate	(MT->dispo->Mark()==1)&&(IN->Mark()==1)
Function	;

Input Gate:	I2
Predicate	((cmp2_buff->dispo->Mark()==1))||((cmp2_buff->dispo->Mark()==1)&&(cmp2_input->dispo->Mark()>0))
Function	;

Input Gate:	I3
Predicate	((cmp3_buff->dispo->Mark()==1)&&(Place3->Mark()==1))||((cmp3_buff->dispo->Mark()==1)&&(cmp3_input->dispo->Mark()>0))
Function	;

Input Gate:	I5
Predicate	((cmp5_buff->dispo->Mark()==1)&&(Place5->Mark()==1))||((cmp5_buff->dispo->Mark()==1)&&(cmp5_input->dispo->Mark()>0))
Function	;

Input Gate:	I6
Predicate	((cmp6_buff->dispo->Mark()==1)&&(Place6->Mark()==1))||((cmp6_buff->dispo->Mark()==1)&&(cmp6_input->dispo->Mark()>0))
Function	;

Input Gate:	I7
Predicate	cmp1_input->dispo->Mark()==1
Function	;

Input Gate:	I8
Predicate	cmp2_input->dispo->Mark()==1
Function	;

Input Gate:	I9
Predicate	cmp3_input->dispo->Mark()==1
Function	;

Output Gate:	O1

Function	cmp1_buff->dispo->Mark()=0;
	cmp1_lost->Mark()++;
	retransmissioncmp1=0;
	PlaceS->Mark()--;

Output Gate:	O10

Function	cmp4_buff->id->Mark()=cmp4P;
	cmp4_buff->dispo->Mark()=1;
	cmp4_input->dispo->Mark()=0;

Output Gate:	O11

Function	cmp5_buff->id->Mark()=cmp5P;
	cmp5_buff->dispo->Mark()=1;
	cmp5_input->dispo->Mark()=0;

Output Gate:	O12

Function	cmp6_buff->id->Mark()=cmp6P;
	cmp6_buff->dispo->Mark()=1;
	cmp6_input->dispo->Mark()=0;

Output Gate:	O13

Function	if(cmp1_buff->dispo->Mark()==1)
	{
	MT->dispo->Mark()=cmp1P;
	MT->id->Mark()=1;
	MT->data->Mark()=cmp1_input->data->Mark();
	}
	else
	{
	if(cmp2_buff->dispo->Mark()==1)
	{
	MT->id->Mark()=cmp2P;
	MT->dispo->Mark()=1;
	MT->data->Mark()=cmp2_input->data->Mark();
	}
	
	else
	{
	if(cmp3_buff->dispo->Mark()==1)
	{
	MT->id->Mark()=cmp3P;
	MT->dispo->Mark()=1;
	MT->data->Mark()=cmp3_input->data->Mark();
	}
	else
	{

	if(cmp4_buff->dispo->Mark()==1)
	{
	MT->id->Mark()=cmp4P;
	MT->dispo->Mark()=1;
	MT->data->Mark()=cmp4_input->data->Mark();
	}
	else
	{
	if(cmp5_buff->dispo->Mark()==1)
	{
	MT->id->Mark()=cmp5P;
	MT->dispo->Mark()=1;
	MT->data->Mark()=cmp5_input->data->Mark();
	}
	else
	{
	if(cmp6_buff->dispo->Mark()==1)
	{
	MT->id->Mark()=cmp6P;
	MT->dispo->Mark()=1;
	MT->data->Mark()=cmp6_input->data->Mark();
	}
	}
	}
	
	}
	}
	}
	
	TM->Mark()--;

Output Gate:	**O13**

Function	if(cmp1_buff->dispo->Mark()==1)
	{
	MT->dispo->Mark()=cmp1P;
	MT->id->Mark()=1;
	MT->data->Mark()=cmp1_input->data->Mark();
	}
	else
	{
	if(cmp2_buff->dispo->Mark()==1)
	{
	MT->id->Mark()=cmp2P;
	MT->dispo->Mark()=1;
	MT->data->Mark()=cmp2_input->data->Mark();
	}
	
	else
	{
	if(cmp3_buff->dispo->Mark()==1)
	{
	MT->id->Mark()=cmp3P;
	MT->dispo->Mark()=1;
	MT->data->Mark()=cmp3_input->data->Mark();

	}
	else
	{
	if(cmp4_buff->dispo->Mark()==1)
	{
	MT->id->Mark()=cmp4P;
	MT->dispo->Mark()=1;
	MT->data->Mark()=cmp4_input->data->Mark();
	}
	else
	{
	if(cmp5_buff->dispo->Mark()==1)
	{
	MT->id->Mark()=cmp5P;
	MT->dispo->Mark()=1;
	MT->data->Mark()=cmp5_input->data->Mark();
	}
	else
	{
	if(cmp6_buff->dispo->Mark()==1)
	{
	MT->id->Mark()=cmp6P;
	MT->dispo->Mark()=1;
	MT->data->Mark()=cmp6_input->data->Mark();
	}
	}
	}
	
	}
	}
	}
	
	TM->Mark()--;

Output Gate:	**O14**

Function	if(MT->id->Mark()==cmp1P) {cmp1_buff->dispo->Mark()=0;retransmissioncmp1=0;}
	else
	{
	if(MT->id->Mark()==cmp2P) {cmp2_buff->dispo->Mark()=0;retransmissioncmp2=0;}
	
	else
	{
	if(MT->id->Mark()==cmp3P) {cmp3_buff->dispo->Mark()=0;retransmissioncmp3=0;}
	else
	{
	if(MT->id->Mark()==cmp4P) {cmp4_buff->dispo->Mark()=0;retransmissioncmp4=0;}
	
	else
	{
	if(MT->id->Mark()==cmp5P) {cmp5_buff->dispo->Mark()=0;retransmissioncmp5=0;}
	else
	{
	if(MT->id->Mark()==cmp6P) {cmp6_buff->dispo->Mark()=0;retransmissioncmp6=0;}
	}

	}
	}
	}
	}
	
	MT->dispo->Mark()=0;
	TM->Mark()++;
	received->dispo->Mark()++;
	received->id->Mark()=MT->id->Mark();
	received->data->Mark()=MT->data->Mark();

Output Gate:	**O15**

Function	Error->Mark()++;
	if(MT->id->Mark()==cmp1P){retransmissioncmp1++;
	if(retransmissioncmp1>nretransmission_permiscmp1)
	PlaceS->Mark()++;}
	else
	{
	if(MT->id->Mark()==cmp2P){retransmissioncmp2++;
	if(retransmissioncmp2>nretransmission_permiscmp2)
	PlaceC->Mark()++;
	}
	else
	{
	if(MT->id->Mark()==cmp3P){ retransmissioncmp3++;
	if(retransmissioncmp3>nretransmission_permiscmp3)
	Place3->Mark()++;
	}
	else
	{
	if(MT->id->Mark()==cmp4P){ retransmissioncmp4=retransmissioncmp4+1;
	if(retransmissioncmp4>nretransmission_permiscmp4)
	Place4->Mark()++;
	}
	else
	{
	if(MT->id->Mark()==cmp5P){ retransmissioncmp5++;
	if(retransmissioncmp5>nretransmission_permiscmp5)
	Place5->Mark()++;
	}
	else
	{
	if(MT->id->Mark()==cmp6P){ retransmissioncmp6++;
	if(retransmissioncmp6>nretransmission_permiscmp6)
	Place6->Mark()++; }
	}
	}
	}
	}
	
	}
	
	MT->dispo->Mark()=0;

Output Gate:	O2

Function	cmp2_buff->dispo->Mark()=0;
	cmp2_lost->Mark()++;
	retransmissioncmp2=0;
	PlaceC->Mark()--;

Output Gate:	O3

Function	cmp3_buff->dispo->Mark()=0;
	cmp3_lost->Mark()++;
	retransmissioncmp3=0;
	Place3->Mark()--;

Output Gate:	O4

Function	cmp4_buff->dispo->Mark()=0;
	cmp4_lost->Mark()++;
	retransmissioncmp4=0;
	Place4->Mark()--;

Output Gate:	O5

Function	cmp5_buff->dispo->Mark()=0;
	cmp5_lost->Mark()++;
	retransmissioncmp5=0;
	Place5->Mark()--;

Output Gate:	O6

Function	cmp6_buff->dispo->Mark()=0;
	cmp6_lost->Mark()++;
	retransmissioncmp6=0;
	Place6->Mark()--;

Output Gate:	O7

Function	cmp1_buff->id->Mark()=cmp1P;
	cmp1_buff->dispo->Mark()=1;
	cmp1_input->dispo->Mark()=0;

Output Gate:	O8

Function	cmp2_buff->id->Mark()=cmp2P;
	cmp2_buff->dispo->Mark()=1;
	cmp2_input->dispo->Mark()=0;

Output Gate:	O9

Function	cmp3_buff->id->Mark()=cmp3P;
	cmp3_buff->dispo->Mark()=1;
	cmp3_input->dispo->Mark()=0;

Ce travail s'inscrit dans le cadre de l'évaluation de la sûreté de fonctionnement des systèmes commandés en réseau (SCR). La capacité des systèmes de commandes à compenser les effets de certaines défaillances de composants amène à redéfinir le concept de défaillances du système. La conséquence est que l'évaluation de la fiabilité prévisionnelle du système est dépendante de l'évaluation fonctionnelle et devient impossible avec les méthodes traditionnelles de la sûreté de fonctionnement. Pour surmonter ces difficultés, une approche basée sur la modélisation en vue de la simulation est proposée. Nous avons choisi les Réseaux d'activités stochastiques (SAN) largement connus dans la modélisation des protocoles de communication ainsi que dans les études de la sûreté de fonctionnement. Dans un premier temps, nous avons cherché à identifier l'incidence de deux types de défaillances fugitives : la perte d'un échantillon d'une part et le retard d'un échantillon dans la boucle de régulation d'autre part. Après, nous simulons le comportement en présence des deux types de perturbations simultanément, mettant en évidence des effets cumulatifs. Si on tient compte maintenant du fait que l'origine des pertes ou retards est due à la présence du réseau, il faut l'introduire dans le modèle. On introduit alors dans le modèle global du système la représentation SAN d'un réseau CAN et l'injection des défaillances dans celui-ci. La méthode de Monte Carlo est utilisée pour estimer les indicateurs de sûreté de fonctionnement et on montre l'influence de certains facteurs comme la charge du réseau par exemple. Nous avons proposé une méthode et les outils associés pour approcher cette évaluation par simulation et ainsi apporter une aide à la conception des systèmes satisfaisant à des exigences critiques sur certains paramètres de performance.

Mots Clés : Fiabilité, systèmes commandés en réseau, réseau d'activité stochastique (SAN), méthode de Monte-Carlo, Controller Area Network (CAN), Modélisation et simulation.

Achieved work in this thesis deals with dependability evaluation of networked controlled system (NCS). The ability of control system to offset the effects of some components' failure leads to redefine the concept of system failure. Consequently the reliability evaluation is dependent on functional parameters and becomes impossible with traditional dependability methods. This work aims at bringing a contribution relative to this aspect. To overcome these difficulties, an approach based on both modelling and simulation is proposed. We choose to work with stochastic activity network (SAN) widely used in modelling communication protocols as well as in dependability studies. First we sought to identify the incidence of two types of transient faults: loss of samples and delay within the control loop. Next we simulate the behaviour in the presence of two types of disturbances at the same time highlighting the cumulative effects. In fact the origin of the loss or delay information inside the control loop is due to the presence of the network, this aspect must be taken into account, that is why we introduce a new model representing the Controller Area Network (CAN) and injection of possible perturbations. Monte-Carlo method is used to estimate dependability parameters showing the influence of some factors such as network load for example. We have proposed a method and associated tools to approach this evaluation by simulation and thus provide assistance in designing systems to meet requirements on certain performance parameters.

Key words: Reliability, networked controlled system, Stochastic Activity Network, Monte Carlo method, Controller Area Network (CAN), Modelling and simulation.