N° d'ordre : 02/2014-M/INF

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene

Faculté de l'Électronique et d'Informatique

MÉMOIRE

Présenté pour l'obtention du **diplôme** de **MAGISTER**

En : INFORMATIQUE

Spécialité : Informatique Mobile

Par : Yassine BOUFENNECHE

Sujet

Influence des messages répétés et des vacances sur les performances et la consommation d'énergie dans les réseaux de capteurs sans fil

Soutenu publiquement, le 27/11/2014, devant le jury composé de :

Mme	Samira MOUSSAOUI	Professeur	à l'USTHB	Présidente
Mme	Nawel GHARBI	Professeur	à l'USTHB	Directrice de mémoire
M	M. C. BOUKALA	Maître de conférences/A	à l'USTHB	Examinateur
M	M. A. RIAHLA	Maître de conférences/B	à l'UMBB	Invité

Résumé

Les technologies sans fil offrent de nouvelles perspectives dans le domaine des télécommunications et des réseaux informatiques. Grâce aux progrès réalisés, un nouveau type de réseaux est apparu sous le nom de réseaux de capteurs sans fil. Ce sont des réseaux sans infrastructure fixe, qui peuvent être déployés de façon rapide dans des zones difficilement accessibles. Ils consistent en des nœuds de petites tailles communicants entre eux et dont le but est de surveiller une zone en prenant des mesures et de faire remonter des alarmes vers certains nœuds particuliers capables de relayer l'information à grande échelle. En revanche, ces réseaux posent un certain nombre de défis scientifiques intéressants pour la communauté des chercheurs, dont certains d'entre eux sont liés directement aux contraintes physiques des nœuds capteurs ; ces derniers ont des ressources très restreintes en matières de mémoire, énergie, calcul et de communication.

À l'égard de ces contraintes, les réseaux de capteurs sans fil doivent être analysés et évalués au préalable. Cependant, l'analyse dans ce domaine se fait généralement par simulation. Dans ce mémoire, nous proposons une approche formelle permettant de faire une évaluation des performances de ces réseaux. Notre travail vise deux contraintes en parallèle, qui sont : la limitation du buffer et la limitation en ressources d'énergie. Dans un premier temps, nous proposons une modélisation qui prend en compte la limitation de mémoire, à l'aide du formalisme des files d'attente avec rappel, buffer limité et source finie. L'analyse du modèle et l'évaluation des performances sont mesurés grâce à un algorithme récursif que nous avons développé. Nous validons cet algorithme en utilisant le formalisme des réseaux de Petri stochastiques généralisés et l'outil GreatSPN.

Dans un second lieu, nous visons à établir un compromis entre la consommation d'énergie des nœuds et les performances du réseau. Pour ce faire, nous introduisons la notion de vacance dans le modèle précédent, de telle façon qu'elle soit combinée avec le rappel, la limitation du buffer et la source finie des messages. Deux modèles sont proposés et comparés, dont le premier consiste à une mise veille des nœuds, et le deuxième consiste à les mettre en off. Nous utilisons, là aussi, le formalisme des réseaux de Petri stochastiques généralisés pour la formalisation des modèles et le développement des formules des indices de performances.

Mots clés : Réseaux de capteurs sans fil, Modélisation, Files d'attente avec rappel et buffer, Source finie de messages, Indices de performance, Vacance du serveur, Réseaux de Petri stochastiques généralisés, Conservation d'énergie.

Abstract

Wireless technologies offer new opportunities in the field of telecommunications and computer networks. With recent advances, a new type of networks has emerged as Wireless Sensor Networks. These networks are without infrastructure and they can be deployed quickly in difficult to access areas. They consist of small size nodes communicating among them and whose purpose is to monitor an area by taking measurements and making up alarms to specific nodes that can relay information on a large scale. However, these networks present a number of interesting scientific challenges for the research community, some of which are related directly to the physical constraints of the sensor nodes ; they have very limited resources in memory, energy, computing and communication resources.

With respect to these constraints, wireless sensor networks should be analysed and assessed in advance. However, analysis in this area is generally made by simulation. In this brief, we propose a formal approach to make an accurate assessment of the performance of these networks. Our work has two parallel constraints, which are : the storage capacity and the limitation of energy resources. Initially, we propose a model that takes into account the limitation of memory, using the formalism of retrial queues with limited buffer and finite source. The model analysis and the performance evaluation are measured thanks to a recursive algorithm we developed. We validate our algorithm using the formalism of generalized stochastic Petri nets and GreatSPN tool.

Secondly, we aim to establish a compromise between the energy consumption of sensor nodes and network performance. At this end, we introduce the notion of server vacation in the previous model, so it will be combined with the retrial phenomenon, limitation of buffer and finite source of messages. Two models are proposed and compared. The first of which is to put sensor nodes in standby state, and the second is to turn them off. We use, again, the formalism of generalized stochastic Petri nets for models formalization and the development of performance indices formulas.

Keywords : Wireless Sensor Networks, Modelling, Retrial queues with buffers, Finite source of messages, Performance indices, Server vacation, Generalized stochastic Petri nets, Power saving.

À mes parents
À mes frères et sœurs
À tous ceux qui m'aiment

Remerciements

Je tiens à remercier Madame le professeur Samira MOUSSAOUI pour l'honneur qu'elle me fait en présidant le jury de mon mémoire.

J'adresse mes sincères remerciements à Messieurs les Docteurs Mohand Cherif BOUKALA et Mohamed Amine RIAHLA, d'avoir en dépit d'une intense activité, bien accepté de participer au jury de soutenance.

Je voudrais également exprimer mes remerciements à Madame le professeur Nawel GHARBI, qui, en tant que directrice de mémoire, s'est toujours montrée à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'elle a bien voulu me consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Je n'oublie pas à remercier tous les membres de ma famille pour leur contribution, leur soutien et leur patience.

Enfin, j'adresse mes plus sincères remerciements à tous mes proches et amis, qui m'ont toujours soutenu et encouragé au cours de la réalisation de ce mémoire.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRALE

Les progrès réalisés dans les technologies MEMS (*Micro-Electro-Mechanical Systems*) ont permis le développent des dispositifs électroniques miniatures dits nœuds micro-capteurs ou simplement nœuds capteurs. Ces derniers ont la capacité de mesurer des grandeurs physiques, telles que la chaleur, l'humidité, le mouvement, la pression, etc., et de les convertir en information numérique. Ils ont aussi des capacités de communication sans fil leur permettant de communiquer entre eux. Le déploiement d'un certain nombre de nœuds capteurs avec un ou plusieurs nœuds spéciaux dits *sinks* dans des zones qu'on veut surveiller forme un réseau de capteurs sans fil (WSN [1] : *Wireless Sensor Network*). Les nœuds détectent des évènements qui se produisent dans leur entourage et envoient des messages correspondants au sink, généralement d'une manière multi-sauts, qui à son tour, peut les relayer à grande échelle. Ces réseaux sont simples et rapides à déployer même dans des milieux difficilement accessibles, car ils ne nécessitent aucune infrastructure. De plus, leur coût est très réduit et ils peuvent par conséquent être à la disposition d'une grande communauté.

Grâce à ces avantages, de nombreuses applications de réseaux de capteurs voient actuellement le jour dans des domaines aussi variés que la défense, la sécurité, la santé, l'agriculture, etc. En revanche, ces réseaux confrontent plusieurs défis, que les protocoles doivent prendre en compte, comme la tolérance aux pannes, le passage à l'échelle, la gestion de la topologie du réseau. D'autres défis sont liés directement aux nœuds capteurs eux-mêmes. Comme ces nœuds sont de petite taille, ils ont par conséquent, des unités d'émission et de réception avec une faible portée du signal. Quant à l'énergie, ils utilisent le plus souvent des batteries irremplaçables, et donc la durée de vie du réseau est déterminée par l'autonomie des batteries. De plus, la capacité de stockage de chaque nœud est très restreinte en la comparant avec celle que nous pouvons trouver sur les autres appareils informatiques actuels.

1. Dans le reste de ce rapport, nous utilisons le terme WSN pour désigner un réseau de capteurs sans fil.

Vues ces contraintes, l'analyse des réseaux de capteurs sans fil avant leur déploiement est nécessaire et même indispensable. Ainsi, on aura une vision sur les performances et la consommation d'énergie du réseau. L'analyse et l'évaluation des réseaux de capteurs se font généralement par simulation grâce aux outils logiciels, tels que NS-2 [14], OMNet++ [15], OPNET [16], etc. Cependant, l'inconvénient de la simulation est qu'elle donne des résultats différents, en dépendance de l'outil utilisé. D'autre part, pour obtenir des résultats proches de la réalité, la simulation devrait prendre un temps d'exécution très long. Une autre alternative, qui reste encore jeune dans le domaine des réseaux de capteur, est l'utilisation des méthodes formelles basées sur des abstractions mathématiques du réseau, et qui ont l'avantage de fournir des résultats exactes.

Parmi les modèles formels, nous trouvons les files d'attente et les réseaux de Petri. Une file d'attente consiste en un espace d'attente, un espace de service composé d'un ou de plusieurs serveurs, un processus qui décrit l'arrivée des clients sollicitant un service, et un processus décrivant leur service. Un comportement naturel des clients arrivant et ne pouvant pas octroyer le service immédiatement est de quitter le système et répéter la demande ultérieurement au lieu d'attendre dans l'espace d'attente. Ceci est connu sous le nom de phénomène de rappel. Un système de file d'attente avec un tel phénomène est dit file d'attente avec rappel (FAR). La combinaison d'un espace d'attente et du phénomène de rappel conduit à un système de file d'attente avec rappel et buffer. Cependant, ces modèles sont plus complexes à analyser et les résultats analytiques n'existent que pour certains cas particuliers. En fait, le seul cas où des résultats analytiques ont été trouvés est le modèle avec un seul serveur et une seule position d'attente [17, 18, 19]. Quand la source d'où les clients arrivent est finie, on parle de file d'attente à source finie. Dans un tel système, le taux d'arrivée des clients diminue quand le nombre de clients dans le système augmente [20]. À cause de la complexité des systèmes caractérisé par le phénomène de rappel, les chercheurs dans ce domaine se sont orientés vers des méthodes numériques ou approximatives.

Par ailleurs, les réseaux de Petri (RdP) sont des modèles graphiques puissants qui permettent l'expression des caractéristiques des systèmes parallèles, telles que la synchronisation, la concurrence, l'exclusion mutuelle, le conflit, etc. L'incorporation de la notion de temps aléatoire dans les réseaux de Petri mène aux réseaux de Petri stochastiques généralisés (RdPSG). Ces derniers ont de plus, l'avantage de faire une évaluation des performances des systèmes, en déterminant leurs propriétés quantitatives (indices de performance) en plus des propriétés qualitatives.

L'analyse des réseaux de capteurs sans fil à l'aide des méthodes formelles a commencé à attirer l'attention des chercheurs ces dernières années, mais elle reste encore jouvencelle. Parmi les travaux existant, la plupart d'entre eux se focalisent sur la conservation d'éner-

gie des nœuds capteurs. C'est le cas des travaux cités dans [21, 22, 23, 24, 25]. Cependant, ces auteurs ignorent la limitation des tailles des buffers des nœuds capteurs, qui est une contrainte forte et qui ne devrait pas être écartée dans un WSN. Par contre, d'autres travaux étaient conscients de cette limitation [26, 27, 28, 29], mais ils restent insuffisants, car ils ne prennent pas en considération le débordement des buffers des capteurs et les retransmissions des messages.

Le travail réalisé dans ce mémoire est sur deux plans. Dans un premier plan, nous proposons une approche formelle de modélisation et d'analyse des WSN en prenant en compte la limitation des buffers des nœuds capteurs et les retransmissions dues à cette limitation (dans le but d'éviter la perte de messages). De plus, nous considérons une source de messages qui est finie. Ceci est dû au fait que le nombre d'évènements pouvant se produire dans la zone de déploiement du réseau est toujours limité aussi grand soit-il. Pour la modélisation, nous utilisons le formalisme des files d'attente avec rappel. Ainsi, notre modèle combine entre l'attente, le rappel et la source finie. Aucune solution analytique ou numérique n'existe dans la littérature pour un tel modèle. Pour cela, nous avons développé un algorithme récursif inspiré de la méthode numérique présentée dans [30]. Nous développons aussi les formules des principaux indices de performance relatifs à ce modèle. Pour vérifier la validité de notre algorithme et des formules des indices de performance, nous procédons à une modélisation du même système en utilisant le formalisme des RdPSG, et nous extrayons les formules des mêmes indices de performance à partir de ce modèle. L'utilisation de ce formalisme nous permet de tirer l'avantage de l'utilisation de l'outil software GreatSPN [31].

Dans un second plan, nous nous intéressons à la conservation d'énergie des nœuds capteur. Nous essayons de trouver le meilleur moyen d'exploiter les temps libres des nœuds. Pour cela, nous incorporons la notion de vacance du serveur [2] dans le modèle précédent, de deux manières différentes. La première consiste à mettre les nœuds capteurs en veille, quant à la deuxième, elle les met en off. À notre connaissance, ces deux modèles seraient les premiers à combiner le rappel, le buffer limité, la source finie et la vacance du serveur avec application dans les WSN. À cause de la difficulté, voir même l'impossibilité, d'analyse de ces modèles avec les FAR, ils sont formalisés à l'aide des RDPSG. Les deux modèles sont comparés en matière de qualité de service et de conservation d'énergie.

Le reste de ce document est organisé comme suit : Le premier chapitre est consacré aux réseaux de capteurs sans fil. Nous présentons leurs spécificités et nous citons quelques exemples de leurs domaines d'application. Dans le deuxième chapitre, nous décrivons les formalismes mathématiques utilisés dans ce mémoire, notamment, les chaînes de Markov, les files d'attente, les files d'attente avec rappel, les modèles avec vacances, les réseaux de

2. Les systèmes dans lesquels le service devient temporairement indisponible sont connus sous le nom de files d'attente avec vacances de serveur(s)

Petri et les réseaux de Petri stochastiques généralisés. Ensuite, dans le troisième chapitre, nous présentons les travaux connexes et nos approches d'analyse des WSN proposées dans ce travail. Les principaux résultats sont présentés et discutés dans le quatrième chapitre. Enfin, une conclusion générale résumera les différents points abordés dans ce mémoire, ainsi que quelques perspectives intéressantes à notre travail.

CHAPITRE 1

RÉSEAUX DE CAPTEURS SANS FIL

1.1 Introduction

Les réseaux de capteurs sans fil [13, 32], plus connus sous le nom de *Wireless Sensor Networks* (WSN), ont gagné beaucoup d'attention au cours de ces dernières années, grâce à leurs avantages multiples et à la possibilité de leur utilisation dans une large gamme d'applications. Un réseau de capteurs sans fil est composé d'un grand nombre de nœuds capteurs de petites tailles, qui communiquent entre eux d'une manière sans fil [33]. Ces nœuds collaborent entre eux afin d'accomplir une tâche commune, en détectant et envoyant les informations concernant les évènements potentiels qui se produisent dans leur entourage. Les réseaux de capteurs sans fil ont une variété d'applications dans le domaine civil autant que dans le domaine militaire [32], tel que la surveillance de l'environnement, la gestion du trafic, la surveillance des structures, etc.

Ce premier chapitre est consacré aux réseaux de capteurs sans fil. Nous allons présenter les points nécessaires à la compréhension d'un WSN, tels que : la définition d'un capteur, l'architecture d'un WSN, le modèle de communication en couches, les différentes topologies d'un WSN, les problèmes rencontrés lors de la conception d'un WSN, la nature du trafic dans un WSN, les techniques d'accès au support physique, le routage et les différents domaines d'application de ce type de réseaux.

1.2 Définition d'un capteur

Un capteur est un équipement qui permet de mesurer une grandeur physique dans l'environnement qui l'entoure, telle que la température, le taux d'humidité, les vibrations, et qui la transforme en une grandeur numérique. Un capteur seul peut permettre des applications locales. Par exemple, un capteur de présence va permettre d'éclairer une lampe lors du passage d'une personne. Lorsque le capteur est équipé d'une carte réseau,

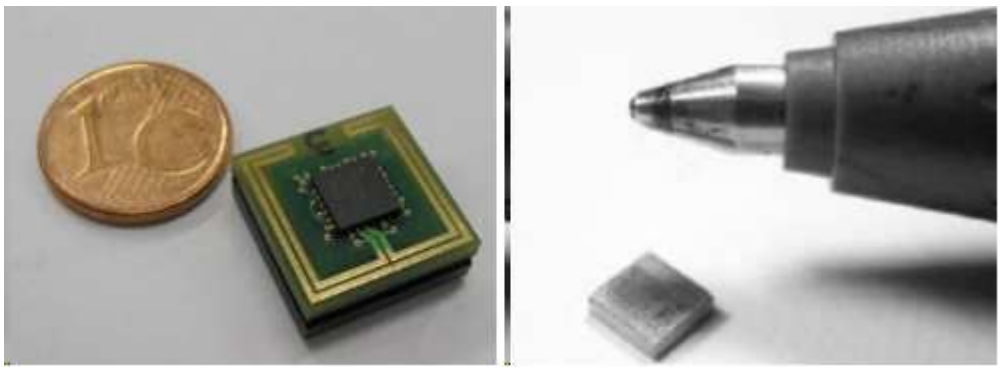

FIGURE 1.1 – Exemples de capteurs

il peut alors être mis en réseau, ce qui lui donne une autre dimension [34]. La figure 1.1 donne des exemples de capteurs.

1.3 Anatomie d'un nœud capteur

Comme indiqué dans la figure 1.2, un nœud capteur est composé essentiellement de quatre éléments de base : une unité de captage (*sensing unit*), une unité de traitement (*processing unit*), une unité de transmission (*transmission unit*) et une unité d'énergie (*power unit*). Un nœud capteur peut avoir des éléments complémentaires en fonction du type de l'application. Par exemple, un système de localisation, comme le système GPS (*Global Positioning System*), qui permet de localiser le capteur, ou encore un générateur d'énergie, comme les panneaux solaires, et un mobilisateur permettant le déplacement des nœuds capteurs [13]. Un noeud capteur est très contraint en matières de vitesse de calcul, capacité de stockage et de communication, volume et capacité en énergie [35].

Unité de captage : généralement composée de deux sous-unités : un capteur qui reçoit (collecte) des mesures sur les paramètres de l'environnement (la température, la pression, etc.) et un convertisseur analogique-numérique (*Analog to Digital Converter*) qui fait la conversion de l'information analogique relevée en information numérique et la transmet à l'unité de traitement [13].

Unité de traitement : cette unité est chargée de l'exécution des protocoles de communications qui permettent au nœud de collaborer avec les autres nœuds du réseau pour accomplir les tâches de captage [13]. Elle est composée d'un microprocesseur et d'une unité de stockage de capacité limitée, généralement de moins de 10 Ko de RAM (*Random Access Memory*) et de moins de 100 Ko de ROM (*Read Only Memory*) [36], nécessaire à l'implantation et à l'exécution d'un programme logiciel qui peut être un système d'exploitation spécialement conçu pour les micro-capteurs, comme le système d'exploitation TinyOS par exemple [37]. La mémoire d'un nœud capteur est donc très limitée par rapport à ce que nous pouvons trouver sur les ordinateurs personnels actuels.

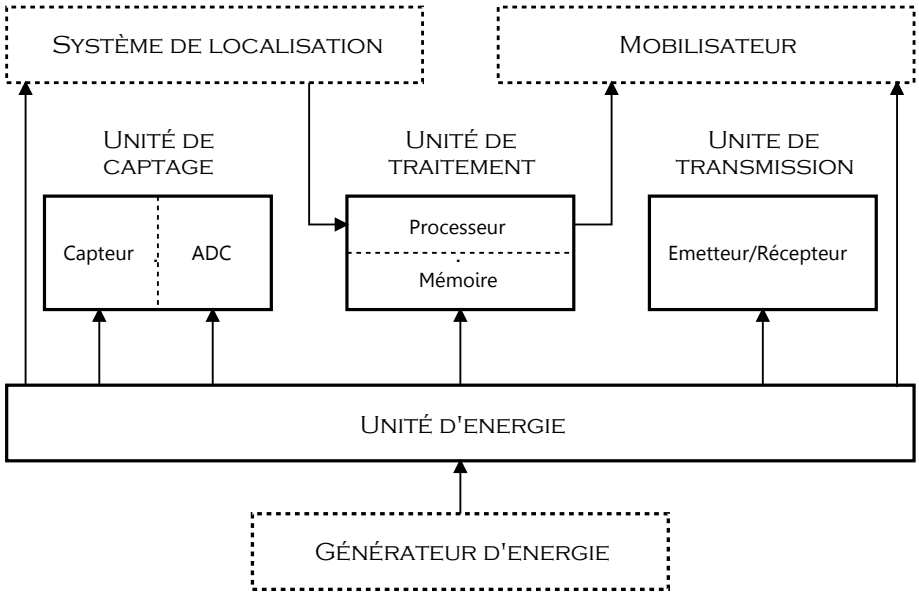

FIGURE 1.2 – Composants d'un nœud capteur [13]

Unité de transmission : elle est responsable de toutes les émissions et les réceptions de données, via un support de communication sans fil commun entre les nœuds.

Unité d'énergie : elle est considérée comme l'un des composants les plus importants d'un nœud capteur. Elle alimente les trois unités précédentes. L'unité d'énergie peut être soutenue par une source d'énergie tel qu'un panneau solaire [13].

1.4 Architecture

Un réseau de capteurs est constitué d'un très grand de nœuds capteurs, permettant de capter et collecter des évènements, d'analyser et de transmettre les informations recueillis dans différents environnements. Outre les nœuds capteurs eux-mêmes, les WSN possèdent le plus souvent des stations de contrôle appelées nœuds-puits (*sink*) ou stations de base [38]. La communication entre les nœuds et le sink se fait généralement d'une manière multi-sauts, à cause de la limitation de la portée du signal des nœuds [24].

Une station de base peut être un nœud fixe ou mobile capable de collecter et de stocker les informations issues des nœuds capteurs, et peut également relier le réseau à l'Internet ou à un autre réseau, où un utilisateur peut avoir accès aux données collectées [39]. Les stations de base possèdent beaucoup plus de capacité que les nœuds capteurs, tant au niveau de la mémoire que de la vitesse de traitement ou en matière d'énergie. Une architecture typique d'un réseau de capteurs sans fil est illustrée dans la figure 1.3.

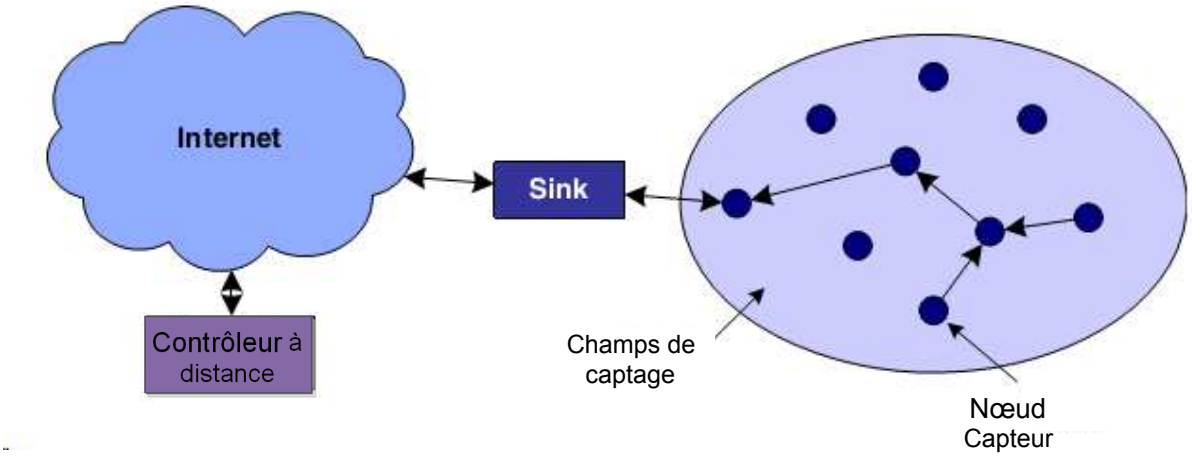

FIGURE 1.3 – Architecture de base d'un réseau de capteurs sans fil

1.5 Modèle en couches

La figure 1.4 représente le modèle de communication en couches, utilisé par le sink et tous les autres nœuds du réseau. Ce modèle fait la combinaison entre le routage et la gestion de l'énergie et favorise la coopération entre les nœuds capteurs [13]. Il comprend cinq couches : la couche application, la couche transport, la couche réseau, la couche liaison de données et la couche physique, ainsi que des plans de gestion d'énergie, de gestion de la mobilité et de gestion des tâches.

Selon les fonctionnalités des nœuds capteurs, différentes applications peuvent être bâties sur la couche application. Cette couche rend les autres couches transparentes à l'utilisateur final. La couche transport permet de diviser les données issues de la couche application en segments, ainsi elle réordonne et rassemble les segments venus de la couche réseau avant de les envoyer à la couche application. La couche réseau s'occupe du routage des données provenant de la couche transport. La couche liaison de données assure la gestion des liaisons entre les nœuds et les stations de base et le contrôle d'erreurs. Le protocole MAC (*Media Access Control*) de cette couche gère l'accès au support physique. La couche physique assure les techniques de modulation, de cryptage, de transmission et de réception de données.

En outre, le plan de gestion de l'énergie surveille la consommation d'énergie au niveau du nœud capteur. Il peut, par exemple, informer les nœuds voisins que ce dernier ne peut pas participer à l'opération de routage si le niveau d'énergie devient plus bas. Le plan de gestion de la mobilité détecte et enregistre tout les mouvements des nœuds capteurs, afin de garder continuellement une route vers l'utilisateur final, et maintenir une image récente sur les nœuds voisins. Le plan de gestion des tâches assure l'équilibrage et la distribution des tâches sur les différents nœuds du réseau, afin d'assurer un travail coopératif et efficace

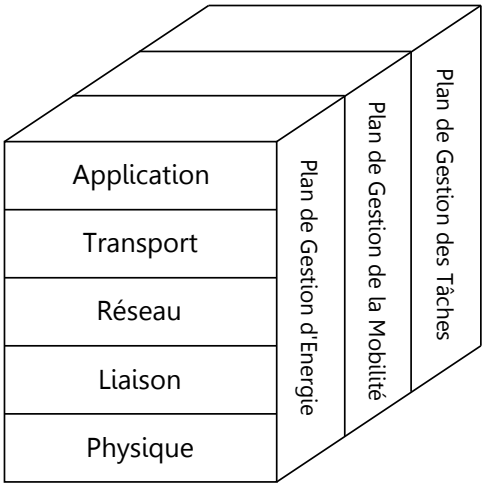

FIGURE 1.4 – Modèle en couches d'un réseau de capteurs sans fil

en matière de consommation d'énergie, et par conséquent, prolonger la durée de vie du réseau [13].

1.6 Structure d'un réseau de capteurs sans fil

Les nœuds composant un réseau de capteurs sans fil peuvent communiquer entre eux d'une manière sans fil selon différentes topologies de communication. Nous donnons ci-dessous une brève description des topologies qui peuvent être adoptées par un réseau de capteurs sans fil.

1.6.1 Topologie en étoile

Dans un réseau avec une topologie en étoile (voir Figure 1.5), une station de base unique peut envoyer et/ou recevoir des messages à un nombre de nœuds capteurs. Les nœuds capteurs ne sont pas autorisés à communiquer entre eux. L'avantage de ce type de réseau inclut la simplicité, la minimisation de la consommation d'énergie au niveau des nœuds capteurs. La latence des communications entre les nœuds capteurs et la station de base est relativement réduite. L'inconvénient d'un tel type de réseau est que la station de base doit être placée à l'intérieur de la portée du signal de chacun des nœuds capteurs [40], ce qui ne rend pas le réseau aussi robuste que d'autres types de réseaux. Un autre problème est aussi lié à la station de base dont la panne provoque l'arrêt de la totalité du réseau.

1.6.2 Réseau maillé

Un réseau maillé (voir Figure 1.6) autorise la transmission de données à partir d'un nœud à n'importe quel autre nœud du réseau se trouvant à l'intérieur de sa portée du signal. Ceci conduit à une communication multi-sauts. Autrement dit, si un nœud veut

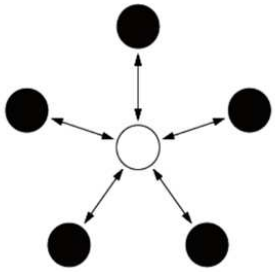

FIGURE 1.5 – Topologie en étoile

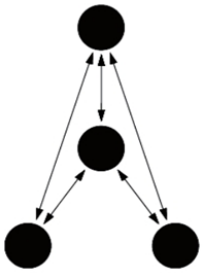

FIGURE 1.6 – Topologie maillée

envoyer un message à un autre nœud en dehors de sa portée du signal, il peut utiliser un nœud intermédiaire pour assurer la transmission du message au nœud désiré [40]. L'avantage de cette topologie est la redondance et l'évolutivité. Si un nœud tombe en panne, un nœud distant reste capable de communiquer avec n'importe quel autre nœud à l'intérieur de sa portée du signal, et qui, à son tour, peut transmettre le message à l'endroit souhaité. En outre, le domaine du réseau n'est pas nécessairement limité ; il peut être facilement étendu par l'ajout d'autres nœuds. L'inconvénient de ce type de réseau est que les nœuds qui mettent en place les communications multi-sauts consomment plus d'énergie. En plus, quand le nombre de sauts augmente, le temps nécessaire à un message pour qu'il arrive à sa destination augmente à son tour.

1.6.3 Topologie hybride étoile - maillée

Une topologie hybride entre la topologie en étoile et la topologie maillée (voir Figure 1.7) donne un réseau de communication robuste et souple, tout en maintenant la capacité de garder la consommation d'énergie des nœuds au minimum. Dans cette topologie, les nœuds capteurs ayant des niveaux d'énergie résiduels plus bas ne sont pas autorisés à faire des communications multi-sauts, ce qui permet une consommation minimum d'énergie. Toutefois, les autres nœuds sont autorisés à effectuer des communications multi-sauts et relayer les messages de ces derniers au reste du réseau. Généralement, les nœuds qui ont la possibilité des communications multi-sauts ont des niveaux supérieurs d'énergie, et si possible, sont branchés à des prises électriques [40].

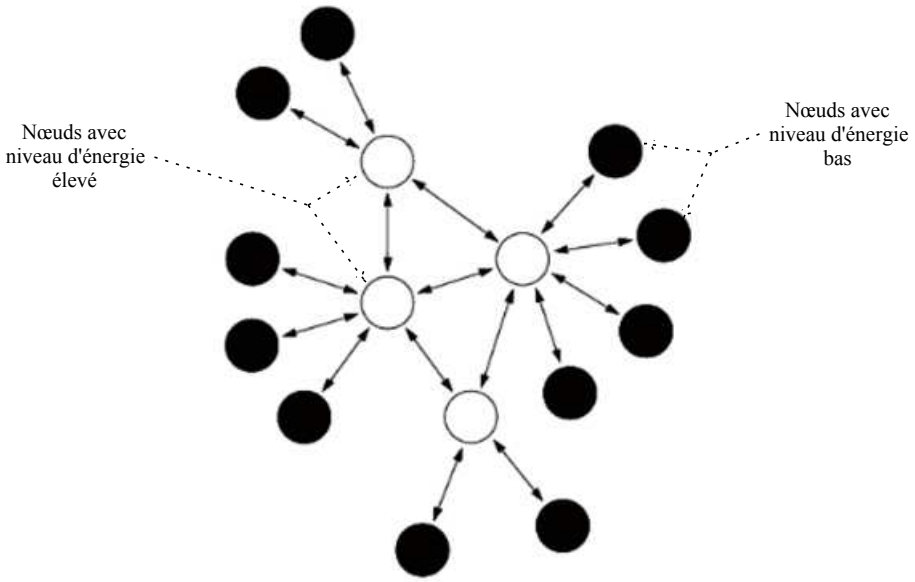

FIGURE 1.7 – Topologie hybride étoile - maillée

1.7 Problèmes de conception d'un WSN

Ils existent plusieurs défis liés au déploiement des WSN. Les nœuds capteurs communiquent généralement à travers des liaisons radio sans infrastructure. Un autre défi est lié au problème d'énergie des nœuds capteurs, qui est généralement non renouvelable [35]. Afin de prolonger la durée de vie des WSN, les protocoles doivent être conçus en vue d'une gestion efficace des sources d'énergie dès le début [13]. Les problèmes de conception ont été résumés dans plusieurs papiers et livres [13, 41, 42, 43]. Nous les mentionnons ci-dessous.

Tolérance aux pannes : les nœuds sont vulnérables et souvent déployés dans des environnements dangereux. Ils peuvent tomber en panne à cause des problèmes de hardware, des dommages physiques ou par l'épuisement d'énergie. Les protocoles déployés dans les nœuds capteurs doivent être capables de détecter ces pannes le plutôt possible et être robustes dans la gestion d'un nombre relativement élevé de pannes, en assurant la fonctionnalité globale du réseau.

Passage à l'échelle : le nombre de nœuds dans un réseau de capteurs sans fil varie de quelques uns à quelques centaines de milliers. En outre, la densité de déploiement de ces nœuds est aussi variable et peut atteindre un niveau très élevé, et par conséquent, un nœud pourrait avoir quelques milliers de voisins à l'intérieur de sa portée du signal. Les protocoles déployés dans le réseau doivent être capables de gérer cette situation et maintenir les meilleures performances.

Coûts de production : puisque beaucoup de modèles de déploiement des WSN considèrent les nœuds comme étant des dispositifs jetables, les réseaux de capteurs peuvent

continuer à récolter l'information seulement si les différents nœuds capteurs peuvent être produits à bon marché. Le prix d'un nœud capteur devrait idéalement être pas loin de $1. [13]

Contraintes du matériel : au minimum, chaque nœud doit être équipé d'une unité de détection, d'une unité de traitement, d'une unité de stockage, d'une unité de transmission, et d'une unité d'énergie. Optionnellement, les nœuds peuvent avoir plusieurs capteurs intégrés ou des dispositifs additionnels tels qu'un système de localisation. Cependant, chaque fonctionnalité additionnelle vient avec un coût additionnel et augmente la consommation d'énergie et la taille physique du nœud. Ainsi, les fonctionnalités additionnelles doivent être toujours équilibrées avec les contraintes du coût.

Topologie du réseau : le déploiement aléatoire, le fonctionnement autonome, et la fréquence élevée de pannes rendent la maintenance de la topologie d'un réseau de capteurs une tâche complexe. En effet, plusieurs centaines de capteurs sont déployés avec une densité pouvant être supérieure à 20 nœuds par m^3 [44], ceci exige une bonne gestion et maintenance de la topologie du réseau déployé.

Support de transmission : la communication entre les nœuds est généralement réalisée avec des liaisons radio au-dessus des bandes ISM [1] (*Industrial Scientific Medical bands*). Cependant, quelques réseaux de capteurs utilisent des communications optiques ou infrarouges, avec l'avantage d'être robustes, et ne soufrent pas des interférences [35].

Consommation d'énergie : beaucoup de défis des réseaux de capteurs tournent autour de la limitation des sources d'énergie. La taille des nœuds limite la taille des batteries [35]. La conception du software et du hardware doit considérer soigneusement le problème d'énergie. La politique de gestion d'énergie dépend aussi de l'application ; dans certaines applications, il pourrait être acceptable d'éteindre un sous-ensemble de nœuds, pendant que d'autres applications exigent que tous les nœuds soient opérationnels en même temps.

1.8 Nature du trafic

Dans les réseaux de capteurs sans fil, on distingue deux types de trafic, qui dépendent de l'application du réseau : *trafic périodique* et *trafic apériodique* [45].

1. Les bandes ISM sont des bandes de fréquences qui peuvent être utilisées pour des applications industrielles, scientifiques, médicales, domestiques ou similaires sans demande d'autorisation auprès des autorités.

1.8.1 Trafic périodique

Dans les réseaux à trafic périodique, les nœuds chargés de la collecte des données transmettent des messages à des intervalles réguliers vers leur(s) destinataire(s). Chaque nœud génère donc des flux intermittents de messages qui lui permettent de passer périodiquement en mode sommeil afin de conserver son énergie [46]. Utilisés dans des applications de supervision, les réseaux périodiques permettent de rassembler suffisamment d'informations sur une zone donnée pour en déduire les tendances et les comportements.

1.8.2 Trafic apériodique

Dans un réseau à trafic apériodique, la transmission de l'information à partir des nœuds sources peut être déclenchée de deux façons :

Transmission par évènement (*event-driven*) : une source ne transmet des données que lorsqu'un évènement est détecté dans ses environs. Un évènement est décrit comme un changement significatif de la valeur mesurée par le capteur dans le temps. La transmission par évènement est généralement utilisée dans des applications de surveillance, comme la détection des incendies et d'intrusion.

Transmission par requêtes (*on demand*) : dans ce cas, les nœuds capteurs ne transmettent de l'information que lorsqu'un autre nœud du réseau (potentiellement le sink) en fait explicitement la requête via un message d'intérêt [2].

1.9 Techniques d'accès au support

Les réseaux de capteurs utilisent des liens de communication sans fil à travers un support commun. Donc, il est indispensable qu'il y ait des méthodes qui permettent aux nœuds d'accéder à ce support commun. À cette fin, des techniques et protocoles ont été proposées. Ces techniques peuvent être classées en trois catégories principales : les *protocoles basés sur un schedule*, les *protocoles basés sur la contention* et les *protocoles hybrides*.

1.9.1 Protocoles basés sur un schedule

Cette famille de protocoles suppose l'existence d'un schedule qui organise l'accès aux ressources pour éviter la contention entre les nœuds. Les ressources sont le temps et la bande de fréquence. La plupart des protocoles basés sur un schedule utilisent la technique TDMA (*Time Division Multiple Access*), qui consiste à découper le temps en slots, chaque slot du temps est alloué à un seul nœud pour lui garantir l'accès au canal afin qu'il puisse

2. Le sink (on un autre nœud) envoie une requête au réseau, dans laquelle il exprime son intérêt (ex. les zones où l'humidité est supérieure de 65%).

transmettre ses données [47].

Ces protocoles sont généralement économes en énergie car ils évitent les collisions, ils limitent le *idle listening*[3] et le *overhearing*[4], et mettent les nœuds en mode sommeil durant les slots du temps réservés aux autres nœuds. De plus, ils garantissent un délai borne de bout-en-bout. En revanche, l'aspect centralisé et le besoin d'une synchronisation rendent ce type de protocoles rigide et non-adapté pour les topologies dynamiques et mobiles [48]. L-MAC [49] est un exemple de protocoles MAC basés sur un schedule.

1.9.2 Protocoles basés sur la contention

Les protocoles basés sur la contention ou protocoles basés sur un accès aléatoire, n'exigent aucune coordination entre les nœuds partageant le même support. Les nœuds qui se trouvent dans une situation de collision réessayent d'accéder au support après une durée de temps aléatoire. Cependant, ces protocoles ne sont pas bien assortis avec l'environnement des WSN [47]. L'amélioration de ces protocoles avec un mécanisme d'évitement de collision est nécessaire pour améliorer leurs performances. Le mécanisme le plus utilisé par cette famille de protocoles est la technique CSMA-CA (*Carrier Sense Multiple Access with Collision Avoidance*), qui impose à un émetteur de s'assurer que le canal est libre avant d'émettre des paquets. Comme exemples de protocoles basés sur la contention, nous citons les protocoles S-MAC [50] et T-MAC [51].

1.9.3 Protocoles hybrides

Une troisième famille de protocoles propose de combiner les deux méthodes décrites ci-dessus. Ainsi, ces protocoles essaient d'avoir les avantages des deux méthodes en alternant les deux dans le temps ou en les combinant d'une manière intelligente. C'est le cas du protocole Z-MAC [52].

1.10 Routage dans les WSN

L'acheminement de l'information dans les réseaux de capteurs est considéré comme un problème plus complexe que dans les autres types de réseau sans fil (ad hoc, ou cellulaires) à cause des caractéristiques inhérentes des nœuds capteurs [41].

Ils existent plusieurs manières pour classifier les protocoles de routage. En général, ces protocoles sont divisés, en se basant sur la structure du réseau, en trois classes : routage

3. L'*idle listening* est l'attente d'une trame par le module radio. Cela arrive quand il a été demandé à un nœud d'être éveillé mais qu'il ne reçoit aucune trame et n'en transmet aucune non plus.

4. L'*overhearing* est la réception par un nœud d'une trame qui ne lui est pas destinée. L'énergie consommée pour la réception et le traitement des données de cette trame est perdue et sans aucun intérêt.

à plat (*Flat-Based*), routage hiérarchique (*Hierarchical-Based*) et routage géographique (*Location-Based*). Dans le routage à plat, tous les nœuds ont typiquement des rôles et des fonctionnalités égaux. Toutefois, dans le routage hiérarchique, les nœuds jouent des rôles différents dans le réseau. Dans le routage géographique, les positions des nœuds sont utilisées pour router les données dans le réseau [53]. En outre, les protocoles de routage peuvent aussi être classés suivant d'autres facteurs, par exemple, en se basant sur la qualité de service (*QoS-Based*) ou en se basant sur la multiplicité de chemins (*Multipath-Based*), comme ils ont été classifiés dans [54].

1.10.1 Routage à plat

Dans les réseaux à plat, les nœuds capteurs collaborent entre eux afin d'accomplir une tâche de captage (détection) [53]. En raison du grand nombre de tels nœuds, il n'est pas faisable d'attribuer un identificateur à chaque nœud. Cette considération a conduit aux protocoles dits *Data-Centric*[5], où la station de base envoie des requêtes à certaines régions et attend les données à partir des nœuds situés dans cette région. Un exemple des protocoles data-centric est : SPIN [55] et Directed Diffusion (DD) [56].

1.10.2 Routage hiérarchique

Les algorithmes de routage hiérarchique se basent sur une organisation en arbre ou sur un partitionnement (*clustering*) du réseau en un ensemble de groupes disjoints afin de garantir le passage à l'échelle, la connexité et la prolongation de la durée de vie du réseau. Les communications entre ces différents groupes sont assurées grâce à un cœur de réseau constitué exclusivement par des super-nœuds (*cluster-heads* ou nœuds parents) désignés pour prendre en charge le routage intra- et extra-cluster. PEGASIS [57] et APTEEN [58] sont deux exemples de protocoles hiérarchiques.

1.10.3 Routage géographique

Le routage géographique se base sur la localisation relative ou absolue des nœuds, afin de trouver une route vers le(s) destinataire(s). Il s'agit d'utiliser les coordonnées du nœud source et celles de ses voisins afin de transmettre ou de retransmettre un paquet au plus près de sa destination. À titre d'exemple de protocoles géographiques, nous citons le protocole GEAR [59].

1.10.4 Protocoles basés sur la qualité de service (*QoS-based*)

Dans ce type de protocoles, des métriques de performance du réseau, telles que le délai de bout en bout, la consommation d'énergie, la bande passante, etc., sont prises en

5. Dans un protocole data-centric, on s'intéresse seulement aux données envoyées par les nœuds, mais pas à leur identifiant (l'information est centrée sur les données).

compte lors de la transmission de données à la station de base [39]. C'est surtout le cas des applications industrielles et militaires. SPEED [60] est l'un des protocoles de routage basés sur la qualité de service.

1.10.5 Protocoles à plusieurs chemins (*Multipath-Based*)

En considérant la transmission de données entre les nœuds capteurs et la station de base, il existe deux paradigmes : le routage en utilisant un seul chemin (*Single-Path*) et le routage en utilisant plusieurs chemins (*Multipath*). Dans le premier cas, chaque nœud capteur envoie ses données à la station de base via le chemin le plus court. Quant au deuxième cas, chaque nœud capteur cherche les k premiers chemins les plus courts et divise et envoie les données via ces différents chemins [54]. Un exemple de ce type de protocoles, est le protocole Disjoint Paths [61].

1.11 Applications des WSN

Les réseaux de capteurs ont gagné leur popularité grâce à leur efficacité dans différents domaines d'application [13, 32, 47, 62, 63, 64]. Nous résumons ci-dessous les principaux domaines :

Applications militaires : les réseaux de capteurs sans fil sont susceptibles d'être une partie intégrante du commandement militaire, contrôle, communications, renseignement, surveillance de champs de batailles, reconnaissance et systèmes de ciblage.

Surveillance de zones : pour la surveillance de zones, les nœuds capteurs sont déployés dans une zone où un phénomène doit être surveillé. Lorsque les capteurs détectent l'évènement surveillé (chaleur, pression, etc.), l'information est rapportée à la station de base, qui prend par la suite les mesures nécessaires.

Transport : des informations en temps réel sur le trafic sont collectées par les réseaux de capteurs pour suivre plus tard des modèles de transport et alerter les chauffeurs de la congestion et des problèmes de circulation.

Applications dans la santé : certaines des applications des réseaux de capteurs dans la santé sont : la surveillance intégrée des patients, les diagnostics des maladies et l'administration de médicaments dans les hôpitaux, la télésurveillance des données physiologiques de l'homme, le suivi et la surveillance des médecins ou des patients à l'intérieur des hôpitaux.

Détection de l'environnement : le terme réseaux de capteurs environnementaux [35] couvre de nombreuses applications des réseaux de capteurs à la recherche dans les sciences

de la terre. Cela comprend la détection des volcans, ou d'autres évènements dans les océans, les glaciers, les forêts, etc. [35]. Nous citons à titre d'exemple :

- la surveillance de la pollution de l'air ;
- la détection des feux de forêt ;
- et la détection des glissements de terrains.

Surveillance des structures : les réseaux de capteurs sans fil peuvent être utilisés pour surveiller les mouvements à l'intérieur des bâtiments et des infrastructures, tels que les ponts et les tunnels.

Secteur agricole : l'utilisation d'un réseau sans fil libère l'agriculteur de l'entretien du câblage dans un environnement difficile. L'automatisation de l'irrigation permet une utilisation plus efficace de l'eau et de réduire les déchets. D'autre part, les capteurs sont capables de surveiller le taux de pesticides dans l'eau potable, le degré d'érosion du sol, et le niveau de pollution de l'air en temps réel.

1.12 Conclusion

Flexibilité, tolérance aux pannes, haute capacité de captage, coût réduit, installation rapide sont les caractéristiques qui ont permis aux réseaux de capteurs de gagné leur popularité et une multiplicité de domaines d'applications. Ce large étendu d'applications fera sans aucun doute, de cette technologie émergente une partie intégrale de nos vies futures.

Cependant, la réalisation des réseaux de capteurs nécessite la satisfaction de certaines contraintes qui découlent d'un nombre de facteurs guidant la phase de conception, tel que la tolérance aux pannes, le passage à l'échelle, le coût, le matériel et la topologie. Une autre contrainte très importante, souvent ignorée par les chercheurs dans le domaine des réseaux de capteurs, est la limitation de la mémoire des nœuds capteurs. Cette limitation en mémoire pourrait entrainer une perte de messages en cas de saturation. Ainsi, l'analyse de ces réseaux avant leur déploiement est d'une grande importance.

Nous nous intéressons dans ce mémoire, à l'application des modèles formels pour l'obtention de résultats exactes. Nous nous intéressons particulièrement à l'analyse des WSN en utilisant les formalismes de files d'attente avec rappel, chaînes de Markov et les réseaux de Petri stochastiques généralisés pour déterminer ses performances. De ce fait, nous allons présenter dans le chapitre suivant, les ingrédients nécessaires à la compréhension de ces formalismes.

CHAPITRE 2

FORMALISMES MATHÉMATIQUES

2.1 Introduction

Aujourd'hui, la plupart des systèmes sont devenus de plus en plus complexes. Il est nécessaire, lors de la phase de conception, de vérifier si le futur système répond aux exigences définies dans le cahier des charges. Il est nécessaire aussi de modifier ou améliorer un système, qui est déjà opérationnel, en modifiant ou en améliorant certains paramètres de performance, tels que le nombre de processus en cours de traitement sur l'unité centrale et l'état de ces processus dans les systèmes informatiques, le temps qui sépare l'émission et la réception d'un message ou le nombre de messages perdus dans un système de communication, le débit et l'état des différents machines dans un système de production, etc.

Le processus permettant de déterminer les différents paramètres de performance des systèmes est connu sous le nom de *l'évaluation* ou *l'analyse des performances*. Mais, avant cela, on doit passer obligatoirement par une phase de *modélisation* permettant de déduire un modèle, qui représente une abstraction mathématique du système. On parle ainsi de l'analyse de performances du modèle et non celle du système, d'où l'importance de la phase de modélisation.

Notre étude des performances des nœuds capteurs sans fil, qui sera abordée dans le chapitre suivant, est basée essentiellement sur trois formalismes mathématiques : les *files d'attente avec rappel*, les *files d'attente avec vacance* et les *réseaux de Petri stochastiques généralisés*. Cependant, l'analyse de ces modèles nécessite la maîtrise des processus stochastiques et plus particulièrement des chaînes de Markov, qui feront l'objet de la section suivante. Ensuite, nous allons aborder le sujet des files d'attente classiques, avec rappel et avec vacance. Les réseaux de Petri et les réseaux de Petri stochastiques généralisés seront étudiés dans la section 2.4. Nous terminons le chapitre par une conclusion.

2.2 Notions mathématiques

2.2.1 Variable aléatoire [1]

Une variable aléatoire est une application X d'un espace de probabilité Ω dans un espace d'états E. L'espace d'états E est défini comme l'ensemble de valeurs que pourrait prendre la variable aléatoire X. Lorsque $E \subset \mathtt{Z}$, X est dite variable aléatoire *discrète*. Par contre, si $E \subset \mathtt{R}$, X est dite variable aléatoire *continue*.

$$\begin{aligned} X: \ & \Omega \longrightarrow E \\ & \omega \longrightarrow X(\omega) \end{aligned}$$

Une variable aléatoire *discrète* est définie par ses *probabilités d'états* : $p(n) = P[X = n]$ pour $n = -\infty, ..., +\infty$, telles que :

$$\sum_{n=-\infty}^{\infty} p\,(n) = 1$$

Une variable aléatoire *continue* est définie par sa fonction *densité de probabilité* $f_x(x)$ pour $x \in]-\infty, +\infty[$, telle que :

$$\int_{-\infty}^{+\infty} f_x(x)dx = 1$$

2.2.2 Loi exponentielle [2]

La *loi exponentielle* de paramètre $\lambda \geq 0$, est une variable aléatoire à temps continu, dont la densité de probabilité est définie par :

$$f(t) = \begin{cases} \lambda e^{-\lambda t} & \text{si } t \geq 0 \\ 0 & \text{sinon} \end{cases}$$

2.2.3 Processus stochastique

Un processus stochastique est une famille de variables aléatoires $\{X(t), t \in T\}$, indexées par l'ensemble T des temps, définies sur un même espace de probabilité, et à valeurs dans un espace d'état E. La variable aléatoire $X(t)$ décrit l'état du processus à l'instant t [1]. L'ensemble des temps T, de même que l'espace d'états E peuvent être discrets ou continus :

- Si $T \subset \mathtt{N}$ (ou dans $\mathtt{Z}$) et $E \subset \mathtt{N}$, on dit que le processus est à temps discret et à espace d'états discret.
- Si $T \subset \mathtt{N}$ (ou dans $\mathtt{Z}$) et $E \subset \mathtt{R}$, on dit que le processus est à temps discret et à espace d'états continu.
- Si $T \subset [0, +\infty[$ (ou dans $\mathtt{R}$) et $E \subset \mathtt{N}$, on dit que le processus est à temps continu et à espace d'états discret.

- Si $T \subset [0, +\infty[$ (ou dans $\mathtt{R}$) et $E \subset \mathtt{R}$, on dit que le processus est à temps continu et à espace d'états continu.

Un processus stochastique $\{X(t), t \in T\}$ est dit *markovien* ou *sans mémoire* si son comportement dans le futur (à l'instant t_{n+1}) ne dépend que de l'état courant (t_n) [65]. Autrement dit, s'il vérifie la propriété de Markov ou de *perte de mémoire* définie formellement comme suit :

$$P[X(t) \leq x \mid X(t_n) = x_n, ..., X(t_0) = x_0] = P[X(t) \leq x \mid X(t_n) = x_n],$$
$$\forall\, t > t_n > t_{n-1} > ... > t_0$$

2.2.4 Processus de Poisson

Un *processus de comptage* $\{N(t), t \geq 0\}$ est un processus qui permet de dénombrer les occurrences d'un évènement aléatoire donné en fonction du temps, par exemple, le temps d'arrivée des clients à un espace de service, le temps d'arrivée des requêtes à un ordinateur, etc. [1]. $N(t)$ est la variable aléatoire égale au nombre d'occurrences appartenant à l'intervalle de temps $[0, t]$; $(N(t) - N(s))$ désigne donc le nombre d'occurrences appartenant à l'intervalle de temps $]s, t]$.

Un processus de *Poisson* de paramètre λ est un processus de comptage $\{N(t), t \geq 0\}$ vérifiant les conditions suivantes :

i. Les nombres d'occurrences dans des intervalles de temps disjoints sont indépendants ;

ii. La probabilité d'une occurrence dans un petit intervalle est proportionnelle à ce dernier, le coefficient de proportionnalité étant λ ;

iii. La probabilité qu'il y ait plus d'une occurrence dans un petit intervalle est négligeable.

Remarque : En considérant un système où les arrivées suivent un processus de Poisson de taux λ, alors les inter-arrivées auront une distribution exponentielle de paramètre λ.

2.2.5 Chaînes de Markov

Une *chaîne de Markov* est un processus stochastique markovien $\{X(t), t \in T\}$ à espace d'état discret E ($E \subset \mathtt{N}$). Selon les valeurs que prend l'index t, une chaîne de Markov peut être à *temps discret* (CMTD : Chaîne de Markov à Temps Discret), ou bien *à temps continu* (CMTC : Chaîne de Markov à Temps Continu).

2.2.5.1 Chaînes de Markov à temps discret

Un processus stochastique $\{X_n, n \in \mathbb{N}\}$ à temps discret et à espace d'états discret E est une *chaîne de Markov à temps discret* si la proprité de Markov suivante est vérifiée :

$$P[X_{n+1} = j_{n+1} \mid X_n = j_n, ..., X_0 = j_0] = P[X_{n+1} = j_{n+1} \mid X_n = j_n], \forall\, n \in \mathbb{N}, j_k \in E$$

Interprétation : La mémoire du processus ne va pas au-delà du dernier état occupé, qui seul conditionne l'état suivant.

Une chaîne de Markov est *homogène* si $\forall\, s, t \in \mathbb{N}$ tels que $(s < t), \forall\, u \in \mathbb{N}$ et pour tout couple d'états (i, j) la condition d'*homogénéité* suivante est vérifiée :

$$P[X_{s+u} = j \mid X_s = i] = P[X_{t+u} = j \mid X_t = i] = P[X_u = j \mid X_0 = i]$$

Interprétation : la probabilité de transition $P[X_t = j \mid X_s = i]$ d'un état i à un état j ne dépend que de la durée $(t - s)$ et non du couple (s, t).

La *probabilité de transition* $P[X_{n+1} = j \mid X_n = i]$ de l'état i à l'état j en une unité de temps, notée p_{ij}, est donc indépendante du temps n.

Une CMTD dont le nombre d'états $|E| = s$ peut être définie soit par une matrice P carrée d'ordre s, dite *matrice de probabilités de transition*, dont les éléments sont les probabilités de transition p_{ij}, ou bien par un graphe orienté, dit *graphe de transition*, dont les sommets sont les états (e_i) de la chaîne, joints deux à deux par l'arc orienté $e_i \longrightarrow e_j$ et pondéré par la probabilité p_{ij}, si et seulement si $p_{ij} > 0$.

Ergodicité d'une CMTD

Une CMTD est dite *irréductible* si et seulement si de tout état i on peut atteindre n'importe quel autre état j de la chaîne en un nombre fini d'étapes. Ceci peut être décrit formellement comme suit :

$$\forall\, i, j \in E, \exists\, m > 1 \text{ tel que } p_{i,j}^{(m)} \neq 0$$

La probabilité $p_{i,j}^{(m)}$ représente la probabilité de transition de l'état i à l'état j en m étapes.

$$p_{i,j}^{(m)} = P[X_{n+m} = j \mid X_n = i], \forall\, n \in \mathbb{N}$$

La *période* d'un état j, d_j, est défini comme le plus grand commun diviseur de tout entier positif n, de telle sorte qu'il est possible d'atteindre l'état j à partir de lui-même

après n transitions, c'est-à-dire :

$$d_j = PGCD\{n \geq\ 1 \mid p_{j,j}^{(n)} > 0\}$$

La *période* d'une CMTC est égale au PGCD (Plus Grand Commun Diviseur) des périodes de chaque état. Elle est égale au PGCD des longueurs de tous les circuits du graphe de transitions correspondant. Une CMTD est dite *périodique* si sa période est supérieure à 1 et *apériodique* si sa période est égale à 1.

On dit qu'une chaîne de Markov à temps discret est *ergodique* si elle est *finie*, *irréductible* et *apériodique*.

Régime stationnaire d'une CMTD

Lors de l'analyse d'un système décrit par une chaîne de Markov à temps discret à s états, l'état dans lequel se trouve le système au début de l'analyse est dit *distribution initiale*. Elle est représentée par un vecteur de probabilité $\pi^{(0)} = [\pi_0^{(0)}, \pi_1^{(0)}, ..., \pi_s^{(0)}]$ où, $\pi_i^{(0)} = P[X_0 = i]$ pour $i = 1, \cdots, s$, correspond à la probabilité que la chaîne se trouve à l'instant initial dans l'état i. Supposons que le système se trouve initialement à l'état j, alors on aura $\pi_j^{(0)} = 1$ et $\pi_i^{(0)} = 0$ pour tout i différent de j.

Soit $\pi^{(n)} = \{\pi_i^{(n)}, i \in E\}$ le vecteur des probabilités d'états à l'instant n, où $\pi_i^{(n)} = P[X_n = i]$ est la probabilité que la chaîne se trouve à l'instant n à l'état i. L'analyse de la CMTD au *régime transitoire* consiste à déterminer le vecteur $\pi^{(n)}$. Quand une chaine de Markov à temps discret est en régime transitoire, on a l'égalité suivante :

$$\pi^{(n)} = \pi^{(n-1)}.P = \pi^{(n-2)}.P^2 = \cdots = \pi^{(0)}.P^n$$

où, P est la matrice des probabilités de transition correspondante à la chaîne.

Lorsque le système modélisé aura fonctionné suffisamment longtemps, c'est-à-dire, n devient très grand (tend vers l'infini), le vecteur des probabilités d'états converge vers un vecteur π ;

$$\pi = \lim_{n \to \infty} \pi^{(n)}$$

Dans ce cas, on dit que le système a atteint le *régime stationnaire* ou *régime permanent* et le vecteur $\pi = \{\pi_i, i = 1, \cdots, s\}$ est dit vecteur des *probabilités stationnaires*, et il est l'unique solution du système d'équations linéaires suivant :

$$\begin{cases} \pi \cdot P = P \quad (*) \\ \sum_{i \in E} \pi_i = 1 \end{cases}$$

$(*)$ est équivalent au système linéaire suivant :

$$\left(\pi_i = \sum_{j \in E} \pi_j p_{i,j}\right)_{i=1,2,\cdots,s}$$

En particulier, si la distribution initiale d'une CMTD ergodique est la distribution stationnaire, elle conserve cette distribution pour tous les temps [3].

2.2.5.2 Chaînes de Markov à temps continu [3, 4, 5]

Les chaînes de Markov à temps continu diffèrent des chaînes de Markov à temps discret en temps dans lequel une transition peut avoir lieu ; dans le cas d'une CMTC, une transition peut avoir lieu dans un temps arbitraire. Par contre, les transitions entre les états d'une CMTD se font dans des points discrets dans le temps. Autrement dit, le processus stochastique est observé à des instants arbitraires dans le cas d'une CMTC, et à des instants discrets dans le cas d'une CMTD.

Une *chaîne de Markov à temps continu* est un processus stochastique $\{X(t), t \geq 0\}$ si et seulement si la propriété de Markov définie dans le cas du temps continu comme suit est vérifiée :

$$P[X(t_{n+1}) = j_{n+1} \mid X(t_n) = j_n, ..., X(t_0) = j_0] = P[X(t_{n+1}) = j_{n+1} \mid X(t_n) = j_n]$$

Comme résultat de cette propriété, le temps de séjour dans un état de la chaîne (temps dans lequel le processus reste dans cet état) est distribué exponentiellement. La probabilité de transition $p_{ij}(t_a, t_b)$ d'un état i à un étai j dans l'intervalle de temps $[t_a, t_b)$, avec $t_a \leq t_b$, est donnée par :

$$p_{ij}(t_a, t_b) = P[X(t_b) = j \mid X(t_a) = i]$$

Dans le cas d'une chaîne de Markov à temps continu homogène, c'est-à-dire, les probabilités de transition dépendent de la valeur $t = t_b - t_a$ et pas des valeurs de t_a et t_a elles-mêmes, les probabilités de transition deviennent :

$$p_{ij}(t) = P[X(t_a + t) = j \mid X(t_a) = i] = P[X(t) = j \mid X(0) = i]$$

Générateur infinitésimal [4, 5]

Considérons une chaîne de Markov à temps continu. Quand le processus entre dans un état i, dans lequel il va rester une durée de temps aléatoire de distribution exponentielle de paramètre μ_i, ensuite, il saute instantanément vers un état $j \neq i$ avec une probabilité p_{ij}, la valeur $\mu_{ij} = \mu_i \times p_{ij}$ représente le *taux de transition* de l'état i à l'état j. Donc, le *temps de transition* de l'état i à l'état j est exponentiel de paramètre μ_{ij}.

Le générateur infinitésimal noté Q d'une chaine de Markov à temps continu ayant s états est une matrice carrée d'ordre s, dont les éléments q_{ij} $(i \neq j)$ correspondent aux taux de transitions μ_{ij} ; $q_{ij} = \mu_{ij}$. Par définition, les éléments de la diagonale $q_{ii} =$ sont égaux à l'opposée de la somme des autres éléments de la ligne i :

$$q_{ij} = \begin{cases} \mu_{ij} & \text{si } i \neq j \\ -\sum_{k=1,k\neq i}^{s} \mu_{ij} & \text{si } i = j \end{cases}$$

Ainsi, comme dans le cas des matrices des probabilités de transition pour les CMTD, les générateurs infinitésimaux permettent de décrire parfaitement le comportement des CMTC.

D'une manière semblable à une CMTD, on peut représenter une chaîne de Markov à temps continu par un graphe de transition, qui est un graphe orienté, où les sommets représentent les états, et les arcs représentent les transitions qui ont un taux non nul dans la matrice Q. Autrement dit, il y aura un arc pondéré par la valeur de q_{ij} allant de l'état i à l'état j si $q_{ij} \neq 0$. Les boucles [1] sont omises de la représentation dans le graphe de transition.

Chaîne de Markov incluse

À partir de chaque chaîne de Markov à temps continu définie par son générateur infinitésimal $Q = \|q_{ij}\|$, on peut obtenir une chaîne de Markov à temps discret dite *chaîne de Markov incluse* (CMI). Elle est définie par sa matrice de probabilité de transition $P = \|p_{ij}\|$, les éléments p_{ij} sont obtenus comme suit :

On sait que :

$$q_{ij} = \mu_{ij} = \mu_i . p_{ij}$$

On a donc :

$$p_{ij} = \frac{\mu_{ij}}{\mu_i} = \frac{\mu_{ij}}{\sum_{k\neq i} \mu_{ik}} = -\frac{q_{ij}}{q_{ii}}$$

Grâce ce résultat, l'étude des chaînes de Markov à temps continu devient beaucoup plus facile [5]. En effet, pour garantir l'existence d'un régime stationnaire, la CMTC doit être irréductible, et on a les résultats suivants [4] :

Résultat 1. Une CMTC est irréductible si et seulement si sa CMTD incluse est irréductible.

Résultat 2. Une CMTC finie et irréductible est *ergodique*.

1. Une boucle est un arc reliant un état i à lui-même.

Résultat 3. Par conséquent, une CMTC finie et irréductible tend vers une distribution stable π après l'écoulement d'un temps fini. Le vecteur π est l'unique solution du système d'équations matricielles suivant :

$$\begin{cases} \pi \cdot Q & = 0 \\ \sum_{i \in E} \pi_i & = 1 \end{cases}$$

Où : E est l'espace des états de la chaîne de Markov.

2.2.6 Processus de naissance et de mort

Les *processus de naissance et de mort* sont des chaînes de Markov à temps continu à valeurs dans l'espace d'états $\mathbb{N}$ et dont les seules transitions possibles à partir de l'état n se font vers les états contigus $(n-1)$ et $(n+1)$. Autrement dit, un processus de naissance et de mort est une CMTC dont le générateur infinitésimal $A = (a_{i,j})$ vérifie :

i. $a_{i,j} = 0$ si $|i - j| \geq 2$;

ii. $a_{i,i+1}$, noté λ_i, est le *taux de croissance* ou de *naissance* de l'état e_i ; λ_0 étant égal à $a_{0,1}$;

iii. $a_{i+1,i}$ noté μ_i est le *taux de décroissance* ou de *mort* de l'état e_i, pour tout i non nul.

Les processus de naissance et de mort sont utilisés pour modéliser les systèmes d'attente et l'évolution de populations [1]. Le processus de Poisson est un processus de naissance et de mort pour lequel la seule transition possible à partir de l'état n se fait vers l'état $(n + 1)$.

2.3 Files d'attente

Un système où des unités ou des clients arrivent à un espace d'attente et attendent afin d'acquérir un service auprès d'un moyen de service si le service n'est pas immédiatement disponible, et à la fin de leur service quittent ce système, est appelé *système de file d'attente*. La théorie des files d'attente est une forme de probabilité qui se rapporte aux files d'attente. Elle permet l'analyse des flux entrants et sortants dans un système de files d'attente, et de calculer les divers paramètres de performance du système, comme la probabilité que le service soit immédiatement disponible pour un nouveau client arrivant, le nombre moyen d'unités dans le système et en attente, le temps passé en attente et dans le système. Ainsi, il est possible de prendre des décisions sur la base des paramètres du système, comme le nombre de ressources composants le service par exemple.

Historiquement, la théorie des files d'attente revient au début du siècle précédent [6], quand Anger Kramp Erlang, un ingénieur danois qui travaillait pour la compagnie

de téléphone de Copenhague, a formulé une solution mathématique [66] qui a permis de déterminer le nombre de lignes nécessaires pour traiter un nombre donné d'appels téléphoniques. Par la suite, il a publié plusieurs articles qui représentaient la naissance de la théorie des files d'attente. Cette théorie est utilisée dans différents domaines, tels que le commerce, l'ingénierie, les finances, les systèmes et les réseaux informatiques, etc.

En fait, la théorie des files d'attente décrite ci-dessus est dite *théorie des files d'attente classique* ou *standard*, dans laquelle, un nouveau client qui arrive et trouve le(s) serveur(s) occupé(s) ou non disponible(s) se comporte selon l'un des deux scénarios suivants :

- Le client quitte le système définitivement sans être servi, ceci correspond au *modèle d'Erlang avec perte* [67] ;
- ou bien, il attend pour être servi après la libération d'un des serveurs, selon une certaine discipline (FIFO, LIFO, ...) [68, 69].

Il existe un autre scénario, qui correspond à une situation intermédiaire, dans laquelle le client rappelle ultérieurement pour avoir le service, autant de fois que nécessaire et à des intervalles de temps distribuées selon une certaine loi de probabilité. Ces systèmes sont appelés *systèmes avec rappel* ou bien *systèmes avec appels répétés*.

Dans certaines situations, le service devient temporairement non disponible au clients. Un tel système est dit *système d'attente avec vacance*. Dans les trois sections suivantes, nous allons aborder les modèles des files d'attente standards, files d'attente avec rappel et des files d'attente avec vacance.

2.3.1 Files d'attente standards

Le modèle de base des files d'attente standard peut être décrit comme suit : des clients arrivent suivant un certain processus pour avoir un service. À cet instant, si un serveur est libre le service commence immédiatement. Sinon, il se met en attente, et sera servi lorsque un serveur sera libre et que les clients arrivés avant lui auront été servis (à moins qu'un système de priorité plus complexe ne soit mis en place). Les temps de service suivent une certaine loi de probabilité. Dans le reste de ce mémoire, nous utilisons simplement le mot file d'attente pour désigner une file d'attente standard.

Un système de file d'attente, tel que schématisé dans la figure 2.1, possède cinq caractéristiques principales, qui sont : le *processus d'entrée* (des arrivées), le *processus de service* (temps de service), le *nombre de serveurs* composant le service, la *capacité du système* et la *discipline de la file d'attente*.

Processus des arrivées : les instants où les clients individuels s'introduisent dans le système, qui sont généralement inconnus et indépendants, s'appellent les *instants d'arrivée*. Les *inter-arrivées*, qui sont les intervalles de temps séparant deux arrivées successives,

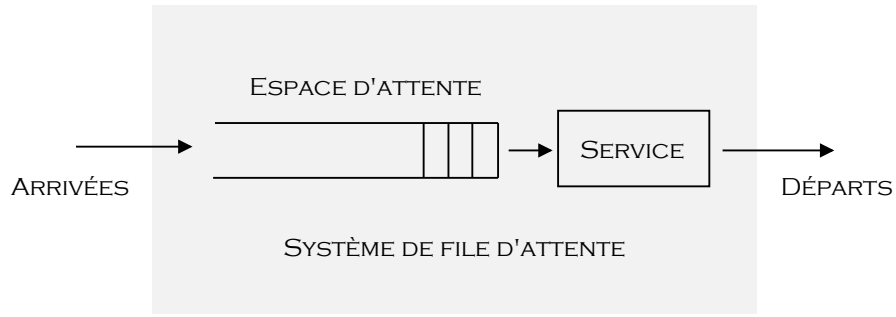

Figure 2.1 – Système de file d'attente

sont donc des variables aléatoires indépendantes de même loi. Le processus d'arrivée, qui décrit les instants d'arrivée, est alors le processus de comptage associé aux inter-arrivées. Si de plus, la loi est une loi exponentielle de paramètre λ, alors le processus des arrivées est un processus de Poisson de taux λ. D'autres lois peuvent être utilisées pour décrire les inter-arrivées, comme la loi générale, déterministe, géométrique, etc.

Temps de service : les durées de services sont toujours des variables aléatoires indépendantes qui suivent une même loi, qui est souvent, mais non nécessairement la loi exponentielle [1].

Nombre de serveurs : dans le cas le plus simple, le système contient un seul serveur. Par contre, il peut y avoir plusieurs serveurs qui travaillent en parallèle. Il y a aussi possibilité qu'un nombre infini de serveurs soient disponibles ; dans la pratique cela signifie que tous les clients reçoivent leur service en parallèle, et il n'y aura pas d'attente dans ce cas.

Capacité du système : la capacité du système correspond au nombre maximum de clients pouvant être admis par le système, y compris les clients en cours de service. Si la taille du système est infinie, il n'y aura aucun blocage de nouveaux clients arrivés. Par contre, si la taille du système est finie, ce qui correspond aux situations réelles, alors des clients peuvent être rejetés.

Discipline de la file d'attente : la sélection du prochain client à servir quand il y a un serveur libre est effectuée selon la discipline de la file d'attente. La règle la plus fréquemment utilisée s'exprime sous la forme « le premier arrivé est le premier servi », connue sous l'abréviation anglo-saxonne FIFO (*First In First Out*). D'autres disciplines sont évidemment possibles, comme « le dernier arrivé est le premier servi » (LIFO : *Last In First Out*) ou bien la discipline de service aléatoire (*Random*).

Une autre caractéristique très importante qui n'est pas présentée dans la figure 2.1 est la source des clients, qui peut être finie ou infinie. Dans le premier cas, le système est dit

fermé, dans l'autre cas le système est dit *ouvert*.

2.3.1.1 Système de notation [6, 7]

David G. Kendall a introduit une notation, qui porte son nom, de la forme $A/B/s$ pour décrire les caractéristiques des files d'attente. Elle est considérée comme un standard dans la théorie des files d'attente. Les symboles A et B représentent, respectivement, le processus des arrivées et le processus de service, qui peuvent être M pour Markovien, D pour Déterministe, E_k pour Erlang d'ordre k, ou G pour Générale. Le symbole s représente le nombre de serveurs. Cette notation a été étendue par la suite à la forme :

$$A/B/s/K/N/Q$$

pour représenter toutes les caractéristiques des files d'attente, où :

- K : représente la capacité du système. Si ce symbole est omis de la notation, la capacité du système est considérée infinie ;
- N : correspond à la taille de la source des clients. Si cette dernière est infinie, ce symbole peut être omis de la notation ;
- Q : représente la discipline de la file d'attente. La discipline par défaut est FIFO.

2.3.1.2 Paramètres de performance

Le but de l'analyse des systèmes est d'obtenir des mesures quantitatives, qui représentent les paramètres (indices) de performance du système. Ces paramètres diffèrent d'un système à l'autre. Les indices les plus importants sont : la probabilité qu'il y ait j unités dans le système, la probabilité de blocage, qui correspond à la probabilité qu'un nouveau client arrive et ne trouve pas de place dans le système, le débit, qui est le nombre moyen de clients servis par unité de temps, le taux d'utilisation du serveur, la taille moyenne de la file d'attente et le temps moyen de séjour, qui correspond au temps moyen passé dans le système. Le calcul de certains de ces paramètres utilise la loi de Little.

2.3.1.3 Loi de Little [6]

En 1961, John Little a publié un article [70] montrant que le nombre moyen d'unités dans le système, L, est lié au temps moyen passé dans le système, W, $L = \lambda.W$, où λ est le taux des arrivée des clients. De la même manière, les deux relations suivantes sont établies :

- $Ls = \lambda.Ws$, nombre moyen de clients en service.
- $Lq = \lambda.Wq$, nombre moyen de clients dans la file d'attente.

Ws et Wq représentent le temps moyen passé dans l'espace d'attente et le temps moyen passé en service respectivement.

2.3.2 Files d'attente avec rappel

Dans la théorie des files d'attente classiques vue dans la section précédente, nous avons constaté qu'un nouveau client qui arrive au système et ne peut pas être servi immédiatement rejoint l'espace d'attente et attend son tour, ou bien quitte le système définitivement. Mais, en réalité, ce n'est qu'une première approximation aux situations réelles, où généralement un tel client retourne au système après un temps aléatoire et essaye d'acquérir le service une autre fois. Ce phénomène est connu sous le nom de *phénomène de rappel*.

Falin et Templeton [7] ont affirmé que les modèles de files d'attente classiques qui ne prennent pas en compte le phénomène de rappel ne peuvent pas être utilisés pour résoudre certains problèmes pratiques importants [71]. D'autre part, il a été précisé dans le livre de Kosten [72] (page 33), que : «*tout résultat théorique qui ne prend pas en considération le phénomène de rappel doit être suspect.*». Les files d'attente avec rappel ont été introduites pour résoudre cette insuffisance, et ont été largement utilisées pour modéliser plusieurs problèmes dans les systèmes téléphoniques, systèmes et réseaux informatiques. Depuis les anciens travaux de Kosten [73], Cohen [74], Wilkinson [75], et Riordan[76], une variété de techniques et résultats ont été developpés pour resoudre quelques problèmes particuliers [77].

À cause de la complexité des files d'attente avec rappel due à la présence de deux flux d'appels, les résultats analytiques sont assez difficiles à obtenir, et existent seulement pour certains modèles particuliers, avec des hypothèses contraignantes sur certains paramètres, tels que le nombre de serveurs, leur fiabilité, etc.. À cet effet, les chercheurs se sont orientés vers des méthodes (algorithmes) numériques, des méthodes d'approximation et de simulation.

2.3.2.1 Modèle générale des files d'attente avec rappel

Un système de file d'attente avec rappel est composé de s ($s \geq 1$) serveurs parallèles et indépendants, disponibles pour le traitement des clients et d'un espace imaginaire dit *orbite*. Un client qui arrive au système pour la première fois est considéré comme un *client primaire* (*appel primaire*). Si un appel primaire trouve au moins un serveur libre, il l'occupe immédiatement, et quitte le système dès que son service est terminé. Si tous les serveurs sont occupés, alors ce client sera *bloqué*, et dans ce cas, il rejoint l'orbite et forme une source d'appels *secondaires* (*répétés*) et devient un *client en orbite*. Chaque client en orbite rappellera pour le service à des intervalles de temps aléatoires, jusqu'à ce qu'un serveur soit libre. Dans ce cas, le client est servi et ensuite quitte le système. Les clients secondaires sont traités de la même manière que les appels primaires. L'intervalle de temps entre deux tentatives consécutives faites par un même client de l'orbite est dit *temps de rappel*. Ce temps est indépendant de tous les temps de rappel précédents. La structure générale d'une file d'attente avec rappel est schématisée dans la figure 2.2.

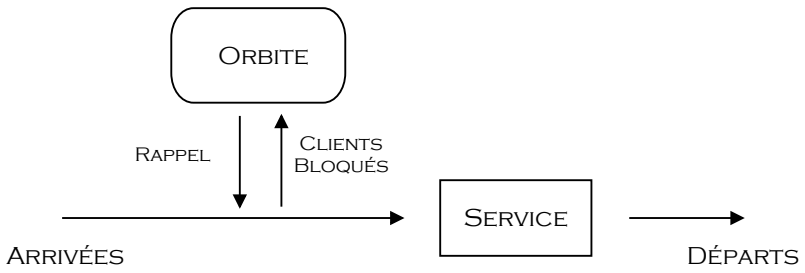

FIGURE 2.2 – Structure générale d'une file d'attente avec rappel

Caractéristiques des files d'attente avec rappel

Un système de file d'attente avec rappel est caractérisé par un mécanisme d'arrivée des clients, un mécanisme de service, un mécanisme de rappel, le nombre de serveurs, la capacité de l'orbite et la taille de la source des clients. Les temps des arrivées des clients ainsi que les temps de service et les temps de rappel sont aléatoires, ce qui rend le processus décrit par ce modèle un processus stochastique. Lorsque la station de service est formée d'un seul serveur, le modèle est dit *mono-serveur*. Dans le cas où elle est formée de deux ou plusieurs serveurs parallèles, le modèle est dit *multi-serveurs*. D'autre part, on distingue principalement deux types de systèmes de files d'attente avec rappel [12] :

- *Les systèmes ouverts* (*sources infinies*) : ils sont alimentés par une population infinie. Ainsi le nombre d'arrivées est illimité. Comme exemple, nous citons le nombre de programmes soumis à un ordinateur.
- *Les systèmes fermés* (*sources finies*) : ils sont plutôt alimentés par un nombre maximum d'unités fixé, correspondant par exemple, au nombre d'abonnés dans un réseau téléphonique.

Ce modèle possède plusieurs variantes, entre autres le modèle des files d'attente avec rappel et buffer.

2.3.2.2 Files d'attente avec rappel et buffer

La structure générale des files d'attente avec rappel et buffer est illustrée dans la figure 2.3. Dans un modèle de file d'attente avec rappel et buffer, le système contient, de plus, un espace d'attente dit *buffer* qui contient N positions d'attente (la taille du buffer est considérée limitée). Dans un tel système, un client primaire ne sera bloqué que si tous les serveurs et les positions d'attente sont occupés ; c'est-à-dire, dès l'arrivée d'un appel primaire, s'il y a au moins un serveur libre, le client reçoit le service immédiatement. Sinon, s'il y a des positions d'attente libres, il rejoint le buffer. Toutefois, si tous les serveurs et positions d'attente sont occupés, il quittera le système définitivement avec la probabilité $1 - H_0$, ou il rejoint l'orbite avec la probabilité H_0. Dans le cas d'une orbite à capacité finie, quand il est plein, n'importe quel client venant à l'orbite sera forcé de quitter le système définitivement. Un client en orbite rappelle plus tard pour le service. S'il trouve des serveurs ou des positions d'attente libres, il reçoit le service immédiatement ou rejoint

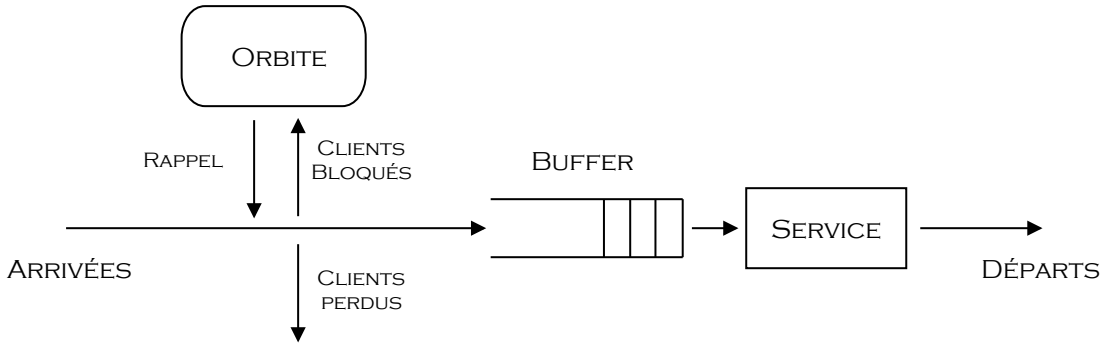

FIGURE 2.3 – Structure générale d'une file d'attente avec rappel et buffer

l'espace d'attente. Encore, si tous les serveurs et positions d'attente sont occupés, il quittera le système définitivement avec la probabilité $1 - H_p$ (le p^{eme} rappel sans succès) ou retourne en orbite avec la probabilité H_p.

La combinaison du buffer et du phénomène de rappel dans un système de file d'attente rend ce dernier plus complexe. Le seul cas de files d'attente avec rappel et buffer où des résultats analytiques ont été trouvés est le modèle avec un seul serveur et une seule position d'attente [17, 18, 19], dont les résultats sont assez similaires au modèle $M/M/2$ avec rappel [78].

2.3.2.3 Système de notation

Comme dans le cas des modèles des files d'attente standards, les modèles de files d'attente avec rappel, à leur tour, sont notés en utilisant la notation de KENDALL étendue, de la forme : $A/B/s/N/O/K$ où :

- A décrit la distribution des temps des inter-arrivées des clients ;
- B décrit la distribution du temps de service de chaque client ;
- s désigne le nombre de serveurs constituant la station de service ;
- N est le nombre de positions d'attente plus le nombre de serveurs dans le système ;
- O désigne la capacité de l'orbite ;
- K est la taille de la source (population) de clients.

La loi des inter-arrivées ou celle des temps de service peut être exponentielle (markovienne notée M), déterministe (D), générale (G), loi d'Erlang d'ordre k (E_k), géométrique (Geo) ou autre. Par contre, la distribution des temps de rappel est supposée généralement exponentielle de taux v (la durée moyenne des intervalles de rappel est de $1/v$). Cependant, il est important de préciser que plusieurs travaux récents considèrent des modèles avec une loi de rappel non-exponentielle [12].

2.3.2.4 Files d'attente avec rappel et source finie

Dans la théorie des files d'attente avec rappel, il est généralement supposé que le flux des clients primaires est poissonnien de taux λ, ce qui rend la probabilité d'arrivée d'un nouveau appel dans une durée dt égale à $\lambda.dt$ quelque soit le nombre de clients déjà présents dans le système à l'instant de son arrivée. Ceci signifie que les appels primaires sont générés par une source de clients suffisamment large [20]. Mais, en réalité, dans beaucoup de situations pratiques, il est important de prendre en compte le fait que le taux de génération des appels primaires diminue quand le nombre de clients dans le système augmente [20]. Ceci correspond à un modèle de file d'attente avec rappel à *source finie* d'appels primaires.

Le premier modèle de files d'attente avec rappel et source finie a été introduit par Kornyshev [79], qui a développé un algorithme numérique permettant le calcul des performances du modèle dans le cas des temps de service exponentiels. De Kok dans [80], a obtenu un schéma récursif pour le calcul de la distribution stationnaire d'un modèle de files d'attente avec rappel mono-serveur et avec une source finie d'appels primaires. Ohmura et Takahachi [81] ont obtenu un système de formules récursives pour le calcul de la distribution stationnaire de l'état du serveur et la longueur de la file d'attente à l'aide des transformations à l'état discret. Dragieva [82] a étudié le nombre d'appels dans le système et le processus de temps d'attente. Toutefois, seulement les indices les plus importants ont été calculés et certains résultats semblaient être incorrects [20]. Li et Yang [83] ont considéré un file d'attente avec rappel et vacances du serveur et ont dérivé un algorithme direct, sans passer à l'état discret, permettant de calculer la distribution de l'état du serveur et la longueur de la file d'attente en régime permanent. Dans [20] Falin et Artalejo ont utilisé la même approche de Ohmura et Takahachi pour améliorer les expressions des principaux indices de performance du modèle $M/G//1/K$.

2.3.3 Files d'attente avec vacance

Dans les modèles de files d'attente classiques, les serveurs sont considérés toujours disponibles. Toutefois, dans beaucoup de systèmes d'attente réels, les serveurs pourraient devenir non disponibles pour une période de temps à cause d'une variété de raisons. Cette période d'absence du serveur peut représenter le fait que le serveur travaille sur des tâches supplémentaires, une opération de maintenance, ou le serveur prend simplement une pause [84]. Pour analyser ces systèmes, la notion de *vacance du serveur* a été introduite dans les modèles d'attente pour représenter les périodes d'absence temporaire du serveur. Ces modèles sont appelés *modèles de files d'attente avec vacance.*

2.3.3.1 Politiques de vacance

Une politique de vacance peut être caractérisée par trois aspects, qui sont : *la règle du début de vacance*, *la règle de fin de vacance* et *la distribution de la durée de vacance*.

Règle du début de vacance

Cette règle indique quand le serveur prend une vacance. Elle est connue aussi sous le nom de *discipline de service*. Ils existent deux disciplines de service majores, à savoir, service *exhaustif* ou *non-exhaustif*. Avec un service exhaustif, le serveur ne peut prendre une vacance qu'une fois que le système devient vide. De l'autre coté, dans une discipline de service non-exhaustive, le serveur peut prendre une vacance même si le système n'est pas vide. Parmi les disciplines de service non-exhaustif, on trouve, par exemple, le *service limité* et le *service avec barrière*. Dans un système d'attente avec vacance à service limité, quand le serveur retourne d'une vacance, il sert un nombre k de clients au maximum, puis débutera une autre vacance. Avec une discipline de service avec barrière, quand le serveur retourne d'une vacance, il sert seulement les clients qui étaient dans le système à son arrivée.

Règle de fin de vacance

Cette règle indique quand le serveur reprend le service des clients. Il y a deux règles populaires, qui sont la politique de *vacance unique* et la politique de *vacance multiple*. Une politique de vacance multiple exige que le serveur prend des vacances successives jusqu'à ce qu'il trouve au moins un client attendant dans le système à la fin d'une période de vacance. Par contre, avec une politique de service unique, le serveur prend seulement une vacance à la fin de chaque période de service. Après cette vacance, le serveur commence le service des clients dans le système, ou il reste *libre* si aucun client n'est trouvé dans le système. Ils existent des règles générales telles que la politique de seuil (N-*vacance*) et la politique T-*vacance*. Dans le cas d'une politique N-*vacance*, le serveur reprend le service quand le nombre de clients dans le système atteint la valeur N. Quant à la politique T-*vacance*, le serveur reste en vacance pendant une période de longueur T. À la fin de cette période, il reprend le service.

Distribution de la durée de vacance

Les vacances du serveur sont généralement considérées comme des variables aléatoires indépendantes et identiquement distribuées avec une certaine loi de probabilité, selon les caractéristiques du système modélisé.

Nous notons que dans le cas d'un système à plusieurs serveurs, si tous les serveurs prennent des vacances et reprennent le service en même temps, on parle alors de *vacances synchrones* ou *vacance de la station*. Dans le cas contraire, quand chaque serveur prend

des vacances individuellement, on parle de *vacances asynchrones* ou *vacance du serveur*. Pour plus de détails et résultats relatifs aux files d'attente avec vacance, le lecteur intéressé peut consulter le papier synthèse de DOSHI [85] et le livre de TAKAGI [86].

2.4 Réseaux de Petri

Les réseaux de Petri (RdP en abrégé) représentent un outil graphique puissant permettant la description formelle des systèmes complexes dans des divers domaines. Il ont été introduits pour la première fois par CARL ADAM PETRI [87], d'où leur nom vient. Depuis cette époque, ils se sont avérés comme un excellent outil de modélisation des systèmes qui sont caractérisé par la concurrence, l'exclusion mutuelle, la synchronisation et le conflit, mais aucune notion de temps n'a été considérée dans le modèle de base.

La notion de temps dans les réseaux de Petri a été introduite par la suite par RAMCHANDANI [88], MERLIN [89], et SIFAKIS [90], et plusieurs autres propositions, avec des approches différentes et basées principalement sur des temps déterministes. Les premières définitions des temps aléatoires avaient été données par SYMONS [91], FLORIN et NATKIN [92], et MOLLOY [93], ce qui a permis de lier les réseaux de Petri au domaine de l'évaluation des performances. Cette approche des réseaux de Petri est connue sous le nom de *réseaux de Petri stochastiques*. Afin d'étendre la puissance de modélisation des réseaux de Petri stochastiques, une extension a été proposé dans [94], où des temps aléatoires sont combinés avec des temps déterministes *nuls*. Un tel formalisme est connu sous le nom de *réseaux de Petri stochastiques généralisés*.

2.4.1 Notions et définitions

Un réseau de Petri consiste de deux parties : une *structure* et un *marquage*. La structure du réseau est un *graphe biparti* orienté, qui représente la partie statique du système modélisé. Les deux types de nœuds sont les *places* et les *transitions*. Les places représentent les différents états possibles du système, tandis que les transitions décrivent les évènements qui peuvent modifier l'état du système. Les *arcs* relient les places et les transitions entre elles. Un arc allant d'une place à une transition exprime l'état local à partir duquel un évènement peut avoir lieu. Par contre, un arc allant d'une transition à une place décrit les changements de l'état local, causés par l'évènement. Les arcs sont libellés par des entiers positifs appelés *poids* de l'arc. Si le poids n'est pas indiqué sur un arc, alors il est égal à 1. Les poids représentent le nombre de ressources nécessaires pour déclencher un évènement dans le cas d'un arc joignant une place à une transition, et le nombre de ressources libérées suite au déclenchement de l'évènement dans le cas d'un arc joignant une transition à une place.

Une place peut contenir un nombre entier de *jetons* ou *marques*. Dans le cas où la place

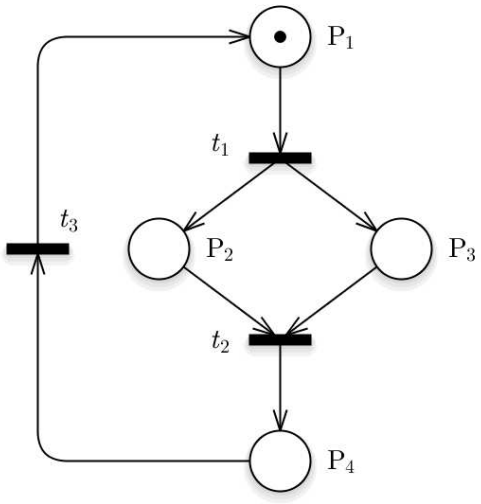

FIGURE 2.4 – Exemple d'un réseau de Petri

peut contenir un seul jeton au maximum, elle représente une *condition*. La condition est vérifiée si le jeton est présent dans la place. Une place représente une *situation* si elle peut contenir plus d'un jeton. L'ensemble des jetons présents dans les différentes places à un instant donné constitue le marquage, noté M. L'état initial du système modélisé par un réseau de Petri est décrit par un marquage dit *initial*, noté M_0, qui correspond aux marques initiaux des différents places.

Graphiquement, les places sont représentées par des cercles, les transitions par des bars ou des petits rectangles, les arcs par des flèches et les jetons par des points. Une représentation graphique d'un réseau de Petri est donnée dans la figure 2.4.

2.4.1.1 Définition formelle [8, 9]

Un réseau de Petri est défini formellement par un tuple $(P, T, Pré, Post)$, où :

- $P = (p_1, p_2, \ldots, p_n)$, est un ensemble fini de places ;
- $T = (t_1, t_2, \ldots, t_n)$, est un ensemble fini de transitions ;
- $Pré : P \times \mathtt{N} \longrightarrow \mathtt{N}$, est l'application d'incidence avant, correspondante aux arcs allant des places aux transitions ;
- $Post : P \times \mathtt{N} \longrightarrow \mathtt{N}$, est l'application d'incidence arrière, correspondante aux arcs allant des transitions aux places ;

$Pré(p, t)$ contient la valeur entière associée à l'arc joignant la place p à la transition t. S'il n'y a pas d'arc reliant p à t, alors $Pré(p, t)$ est égale à zéro, tandis que $Post(p, t)$ contient la valeur entière associée à l'arc joignant la transition t à la place p. S'il n'y a pas d'arc reliant t à p, alors $Post(p, t)$, à son tour, est égale à zéro. $\mathtt{N}$ est l'ensemble des entiers naturels.

Un réseau de Petri marqué se définit par un couple (R, M), dans lequel R est un réseau de Petri et $M : P \longrightarrow \mathbb{N}$ est l'application de marquage, telle que $M(p)$ désigne le nombre de marques présentes dans la place p. Si le réseau de Petri contient n places, alors $M(P)$ est un vecteur à n éléments.

Remarque : Dans le cas d'un arc allant d'une place à une transition, la place est dite *place d'entrée* de la transition. Dans le cas contraire (arc allant d'une transition à une place), elle est dite *place de sortie* de la transition.

2.4.2 Représentation Matricielle

Un réseau de Petri est représenté sous forme matricielle par un une matrice dite *matrice d'incidence*, telle que :

$$W = W^+ - W^-$$

W^- et W^+ sont *la matrice d'incidence* avant et *la matrice d'incidence arrière* associées, respectivement, à *Pré* et $Post$, ayant toutes les deux n lignes (nombre de places) et m colonnes (nombre de transitions). La matrice d'incidence avant W^- est égale à $[w_{ij}^-]$, où $w_{ij}^- = Pré(p_i, t_j)$, et la matrice d'incidence arrière W^+ est égale à $[w_{ij}^+]$, où $w_{ij}^+ = Post(p_i, t_j)$.

Exemple : soit le réseau de Petri marqué représenté dans la figure 2.4. Il contient quatre places et trois transitions :

$$P = \{p_1, p_2, p_3, p_4\}, T = \{t_1, t_2, t_3\}$$

La matrice d'incidence avant (W^-) et la matrice d'incidence arrière (W^+) correspondantes au réseau de Petri représenté dans la figure sont les suivantes :

$$W^- = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, W^+ = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

À partir de W^- et W^+, on obtient la matrice d'incidence (W) :

$$W = \begin{pmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}$$

Le marquage initial est : $M_0 = \{1, 0, 0, 0\}$.

2.4.3 Aspects dynamiques des réseaux de Petri

Les définitions et les notions des RdP vus précédemment représentent leur aspect *statique*, c'est à dire, la description instantanée du système. Cependant, les systèmes qu'on veut étudier évoluent avec le temps : il effectuent des actions et passent par différents états. Donc, il est nécessaire d'étudier leur évolution. Il est par conséquent nécessaire de disposer d'une description de leur fonctionnement, autrement dit d'une modélisation comportementale. Les notions suivantes permettent d'exprimer la dynamique des réseaux de Petri.

Nous utilisons les notations suivantes [95] :

- ${}^{\cdot}t = \{p \in P \mid Pré(p,t) \geq 0\}$, est l'ensemble des places d'entrée de t ;
- $t^{\cdot} = \{p \in P \mid Post(p,t) \geq 0\}$, est l'ensemble des places de sortie de t ;

2.4.3.1 Règles de sensibilisation et de franchissement [5, 10, 11]

Les règles de sensibilisation et de franchissement dans un réseau de Petri sont liées aux transitions. La règle de sensibilisation définit les *conditions* qui permettent à une transition t d'être franchie, et la règle de franchissement décrit le changement de l'état du système produit par la transition.

Une transition t est *sensibilisée* ou *tirable* ou *franchissable* depuis un marquage M si et seulement si chaque place d'entrée p contient un nombre de marques supérieur ou égal au poids associé à l'arc allant de p à t. D'une façon formelle on écrit :

$$\forall\, p \in {}^{\cdot} t, M(p) \geq Pré(p,t),$$

et on note : $M[t\rangle$.

Le *tir* d'une transition sensibilisée depuis un marquage M exclut de chaque place $p \in {}^{\cdot} t$ un nombre de marques égal au poids associé à l'arc allant de p à t, et dépose dans chaque place $p \in t^{\cdot}$ un nombre de marques égal au poids associé à l'arc allant de t à p. Ceci, conduit à un nouveau marquage M' défini par :

$$\forall\, p \in P, M'(p) = M(p) - Pré(p,t) + Post(p,t),$$

Dans ce cas, le marquage M' est dit *accessible* à partir du marquage M et on note : $M[t\rangle M'$.

Exemple : soient les deux réseaux de Petri représentés dans la figure 2.5. La transition t_1 n'est pas franchissable. Par contre, la transition t_2 est franchissable. Son franchissement conduit du marquage $M = (1,2,0)$ au marquage $M' = (0,0,1)$, et on écrit : $(1,2,0)[t_2\rangle(0,0,1)$.

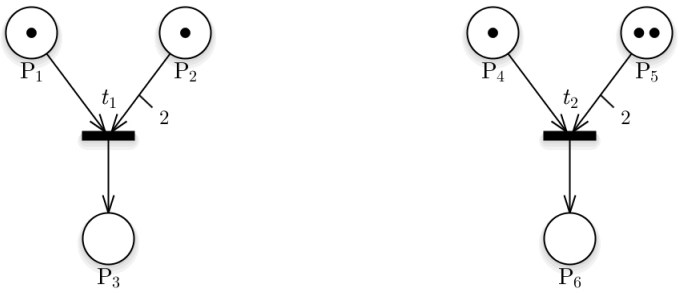

FIGURE 2.5 – Exemple de transitions

Une transition qui n'a aucune place d'entrée est dite *place source*, tandis qu'une transition qui n'a pas de places de sortie est appelée *transition puits*. On note qu'une transition source est toujours franchissable, et son tir produit des marques et ne consomme aucune marque. Par contre, le tir d'une transition puits consomme des marques sans en produire d'autres.

2.4.3.2 Ensemble d'accessibilité et graphe des marquages accessibles

À partir du marquage initial M_0, il est possible de calculer tous les marquages qui peuvent être atteints (espace d'états que pourrait prendre le système modélisé), ainsi que tous les chemins suivis par le système quand il passe d'un état à un autre. Soit (R, M_0) un réseau de Petri, et $\sigma = t_1, t_2 \cdots t_n \in T^*$ une *séquence de transitions*. La séquence σ est franchissable depuis un marquage M_0 si et seulement il existe un ensemble de marquages $\{M_1, M_2, \cdots, M_n\}$ tel que :

$$M_0[t_1\rangle M_1[t_2\rangle M_2 \cdots M_{n-1}[t_n\rangle M_n$$

Le tir de σ conduit alors au marquage M_n. On le note $M_0[\sigma\rangle M_n$. L'ensemble des marquages $A = \{M_0, M_1, M_2, \cdots M_n\}$ représente l'ensemble des *marquages accessibles* ou (*ensemble d'accessibilité*) à partir du marquage initial M_0 et on note $A(R, M_0)$.

Graphe des marquages accessibles

Soit (R, M_0) un réseau de Petri. Le *graphe d'accessibilité* ou *graphe des marquages accessibles*, noté $G(R, M_0)$, est un graphe dont les nœuds (sommets) représentent les marquages accessibles $A(R, M_0)$, et les arcs étiquetés par les noms des transitions qui ont causé les changements des marquages, sont définis par la relation de tir entre marquages. Donc, un arc étiqueté par t joint M à M' si et seulement si $M[t\rangle M'$.

Si on reprend le réseau de Petri décrit dans la figure 2.4, l'ensemble d'accessibilité est :
$M_0 = (1, 0, 0, 0)$, $A = \{M_1, M_2\}$, où $M_1 = (0, 1, 1, 0)$, $M_1 = (0, 0, 0, 1)$, et le graphe des marquages accessibles est représenté dans la figure 2.6.

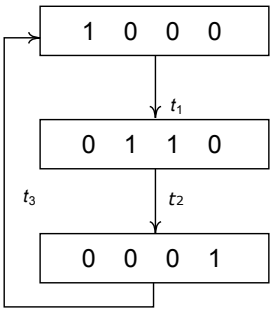

FIGURE 2.6 – Graphe de marquages accessibles d'un RdP

2.4.4 Situations possibles entre transitions

Lors de l'évolution d'un réseau de Petri, plusieurs situations sont possibles entre les transitions, dont les plus importantes sont : le conflit, la confusion, la concurrence et l'exclusion mutuelle.

Conflit : deux transitions t_1 et t_2 sont en *conflit structurel* si et seulement si elles ont au mois une place commune en entrée. Elles sont en *conflit effectif* pour un marquage M si elles sont en conflit structurel et :

$$M[t_1\rangle \text{ et } M[t_2\rangle \text{ et } \exists\, p \in P : M(p) < Pré(p, t_1) + Pré(p, t_2)$$

Un conflit effectif correspond donc à un choix exclusif entre deux franchissement [12]. Les transitions t_1 et t_2 de la figure 2.7 sont en conflit effectif.

Concurrence : contrairement au conflit, la concurrence est caractérisée par le parallélisme des activités. En conséquence, deux transition t_1 et t_2 sont *concurrentes* dans un marquage M si elles sont tous les deux sensibilisées dans M, et elles ne sont pas en conflit.

Confusion : une situation de confusion correspond à une situation où le conflit et la concurrence sont présents en même temps. Une situation de confusion est représentée dans la figure 2.7, où t_3 et t_5 sont concurrentes et chacune d'elles est en conflit avec t_4.

Exclusion mutuelle : deux transitions t_1 et t_2 sont *mutuellement exclusives* ou en situation d'*exclusion mutuelle* si elles ne peuvent jamais être sensibilisées en même temps dans un même marquage M. Autrement dit, t_1 et t_1 sont mutuellement exclusives si et seulement si :

$$\nexists M \in R(M_0) : t_1 \in E(M) \text{ et } t_2 \in E(M),$$

$E(M)$ désigne l'ensemble de transitions sensibilisées dans le marquage M.

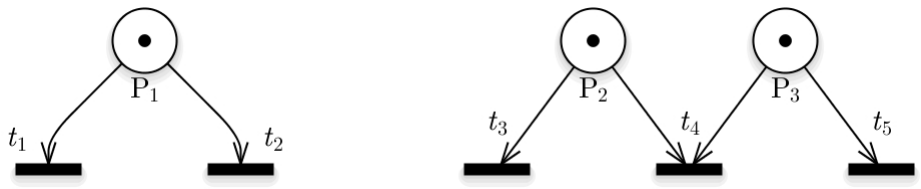

FIGURE 2.7 – Situations de conflit et de confusion

2.4.5 Propriétés des réseaux de Petri

Après la modélisation d'un système par un modèle formel, il est nécessaire de dériver et d'étudier des propriétés concernant ce système. Les réseaux de Petri ont une puissance majore dans l'analyse des propriétés et problèmes associés aux systèmes parallèles.

Dans ce qui suit, nous décrivons les propriétés les plus importantes des réseaux de Petri.

2.4.5.1 Bornitude

Un réseau de Petri (R, M_0) est dit *k-borné* ou simplement *borné* si et seulement si, pour tout marquage accessible à partir de M_0, le nombre de marques dans chaque place du réseau ne dépasse pas un nombre entier k. Dans ce cas, chaque place p est *bornée*. Formellement, on écrit :

$$(R, M_0) \text{ borné } \iff \exists\, k \in \mathtt{N}, \forall\, M \in A(R, M_0), \forall\, p \in P : M(p) \leq k$$

On note que si un réseau de Petri est borné, le nombre de marquages accessibles est fini, et par conséquent, le graphe des marquages accessibles existe. On note aussi que dans le cas particulier où $k = 1$, le réseau de Petri est dit *sauf*.

2.4.5.2 Absence de blocage

Un système modélisé par un réseau de Petri est en situation de blocage, s'il aboutit à un état (marquage) où aucune transition n'est franchissable [5]. Un réseau de Petri marqué est dit *sans blocage* pour un marquage initial M_0 si aucun marquage accessible ne représente une situation de blocage.

2.4.5.3 Vivacité [5, 8]

Une transition t dans un réseaux de Petri (R, M_0) est dite *vivante* pour le marquage initial M_0, si et seulement si pour chaque marquage accessible M, il existe un marquage M' et une séquence transitions σ contenant t tel que : $M[\sigma\rangle M'$. Autrement dit, il y a toujours une possibilité de franchissement de t à partir de tout marquage accessible de M_0. Une transition qui n'est pas vivante est dite *morte*. Si toutes les transitions sont vivantes, le

réseau de Petri est dit vivant. Une conséquence très importante de la propriété de vivacité, est que s'il existe au moins une transition vivante, aucune situation de blocage n'aurait lieu. Ainsi, on est sure que le système modélisé continue à fonctionner en permanence.

2.4.5.4 Réinitiabilité et état d'accueil

Un réseau de Petri (R, M_0) est dit *réinitiable* si pour chaque marquage $M \in A(R, M_0)$, M_0 est accessible à partir de M. Ainsi, dans un réseau de Petri réinitiable, on peut toujours retourner au marquage initial (état initial). Cependant, dans beaucoup d'applications, il n'est pas nécessaire de retourner à l'état initial [11], mais plutôt de revenir à un autre état dit *état d'accueil*. Un marquage M' est un *état d'accueil*, si pour chaque marquage M appartenant à l'ensemble d'accessibilité de (R, M_0), M' est accessible à partir de M.

2.4.5.5 Persistance

Un réseau de Petri (R, M_0) est dit *persistant*, si pour tout couple de transitions franchissables, si une d'entre elles est tirée, l'autre reste franchissable. Une fois une transition est sensibilisée dans un graphe persistant, elle reste franchissable jusqu'à ce qu'elle soit tirée.

2.4.6 Réseaux de Petri à arcs inhibiteurs

Dans certaines situations pratiques, il devient très important d'avoir la capacité de désactiver le tir d'une transition lorsque certaines conditions sont vérifiées. Le modèle de base des réseaux de Petri n'offre pas cette possibilité. Ainsi, *les réseaux de Petri à arcs inhibiteurs* ont été introduits. Un arc inhibiteur est un arc étiqueté particulier, qui relie une place à une transition et non l'inverse, avec un cercle du coté de la transition.

Ce modèle permet d'interdire le franchissement d'une transition si le marquage d'une place est supérieure à une valeur donnée. D'autre part, les matrices d'incidence sont complétées par une *matrice d'inhibition* qui impose que le marquage d'une place soit strictement inférieur à une valeur donnée qui correspond au poids de l'arc inhibiteur.

Définition [12]

Un réseau de Petri à arcs inhibiteurs est défini par un tuple $R = (P, T, Pré, Post, Inh)$ où :

- P est un ensemble fini de places et T un ensemble fini de transitions ;
- $Pré, Post : P \times T \longrightarrow \mathrm{N}$, sont les fonctions d'incidence avant et d'incidence arrière respectivement ;
- $Inh : P \times T \longrightarrow (\mathrm{N} \backslash \{0\})$, est la fonction d'inhibition.

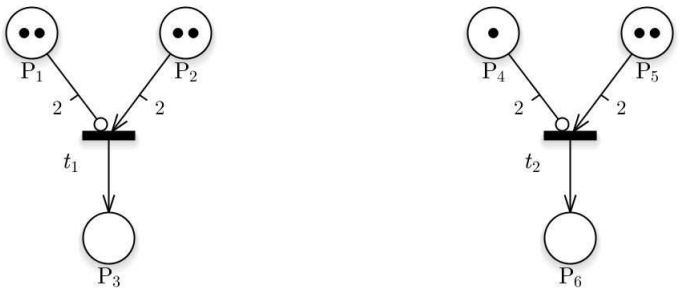

FIGURE 2.8 – Transitions à arcs inhibiteurs

Dans un réseau de Petri à arcs inhibiteurs, seule la condition de franchissabilité est modifiée par rapport au modèle de base.

Condition de franchissement

Soit (R, Inh) un réseau de Petri à arcs inhibiteurs et M un marquage. Une transition t est sensibilisée dans le marquage M si et seulement si : (i) chaque place d'entrée p contient un nombre de marques supérieur ou égal au poids associé à l'arc reliant p à t, et (ii) pour chaque place p' dont il existe un arc inhibiteur allant à t, le nombre de marques présentes dans la place p' doit être strictement inférieur au poids de l'arc inhibiteur. Formellement, on écrit :

$$t \text{ est franchissable dans } M \Longleftrightarrow \forall\, p \in P : M(p) \geq Pré(p,t) \text{ et } M(p) < Inh(p,t)$$

Si on prend les transitions de la figure 2.8, la transition t_1 n'est pas franchissable, tandis que la transition t_2 est franchissable.

2.4.7 Réseaux de Petri stochastiques généralisés

Le modèle de base des réseaux de Petri est un outil puissant de description, d'analyse et de vérification des systèmes complexes. Il offre des moyens de vérification des propriétés qualitatives du système modélisé, tel l'existence ou non d'une situation de blocage lors de l'évolution du système. Leur point faible réside dans le fait qu'ils sont incapables de déterminer des propriétés quantitatives, comme le nombre d'unités présentes dans un buffer à un instant donnée. Ceci revient à l'absence d'informations temporelles. Afin de palier ce problème et d'étendre leur capacité de modélisation, la notion de temps a été incorporée pour la première fois dans les réseaux de Petri par MERLIN[89] et RAMCHANDANI [88] en 1970. En fait, plusieurs manières d'incorporer le temps dans les RdP sont possibles ; on peut associer le temps à une transition, à une place, à un un arc, ou même à des jetons, mais la première approche reste la plus utilisée [96, 5].

Une transition à laquelle on associe un délai, est dite *transition temporisée*. Elle correspond à une activité qui nécessite un temps d'exécution, comme par exemple, l'exécution

d'une tâche par un processeur ou la transmission d'un message dans un réseau de communication. Une transition dans le modèle de base des RdP peut être franchie à n'importe quel moment une fois qu'elle est sensibilisée, tandis qu'une transition temporisée, une fois sensibilisée, elle ne peut être franchie qu'après l'écoulement du temps qui lui est associé. Ce temps est appelé *délai de franchissement*. On distingue principalement deux politiques de franchissement des transitions temporisées [96] :

- Le franchissement à trois phases (*three-phase firing*), dans lequel les jetons (marques) sont consommés à partir des places d'entrée quand la transition est sensibilisée, et ils ne seront produits dans les places de sortie qu'après l'écoulement du délai de franchissement.
- Le franchissement atomique (*atomic firing*), les marques restent dans les places d'entrée tout au long du délai de franchissement. Après l'écoulement de ce délai, elles seront consommées et déposées dans les places de sortie d'une manière atomique. Cette politique de franchissement peut conserver le comportement structurel des modèles de base des RdP [96].

Un réseau de Petri où toutes les transitions sont des transitions temporisées, atomiques et avec des délais de franchissement aléatoires est appelé *réseau de Petri stochastique* (RdPS en abrégé). Les délais de franchissement sont généralement des variables aléatoires de distribution exponentielle négative [96]. À cause de la propriété de perte de mémoire de la loi exponentielle, le graphe des marquages accessibles d'un RdPS borné est isomorphique à une CMTC à espace d'états fini, et ainsi, on peut bénéficier des méthodes d'analyse des CMTC dans les RdPS [11, 93]. Dans le cas où il y a deux ou plusieurs transitions qui sont sensibilisées simultanément, celle qui a le délai de franchissement le plus court sera franchie en premier.

En fait, une transition dans un réseau de Petri stochastique ne modélise pas forcément une activité qui nécessite un temps d'exécution. Elle peut correspondre à une opération de contrôle logique, comme la vérification de la disponibilité d'une ressource dans le système, ou à une activité qui a un temps d'exécution négligeable. Une telle transition est dite *transition immédiate*, son délai de franchissement est *nul*. Ainsi, après qu'elle soit sensibilisée, elle pourrait être franchie à n'importe quel moment.

Ainsi, les réseaux de Petri stochastiques ont été étendus à une classe connus sous le nom de *réseaux de Petri stochastiques généralisés* (RdPSG en abrégé), dans lesquels des transitions immédiates avec des délais de franchissement nuls coexistent avec des transitions temporisées avec des taux de franchissement distribués exponentiellement. L'existence des deux types de transitions conduit à la division de l'ensemble des marquages accessibles en deux sous ensembles :

- Les *marquages tangibles* (*tangible markings*) : où seules les transitions temporisées sont sensibilisées.

- Les *marquages évanescents* (*vanishing markings*) : dans lesquels au moins une transition immédiate est franchissable.

Dans un RdPSG, les transitions immédiates sont représentées graphiquement par des traits, tandis que les transitions temporisées sont représentées par des rectangles.

2.4.7.1 Définition formelle [12]

Un réseau de Petri stochastique généralisé (RdPSG) est un huit-uplet $< P, T, Pré, Post, Inh, pri, W, M_0 >$ où :

- P est l'ensemble des places ;
- T est l'ensemble des transitions temporisées et immédiates ;
- $Pré$, $Post$ et Inh : $P \times T \rightarrow \mathtt{N}$ sont les fonctions d'incidence avant, d'incidence arrière et d'inhibition respectivement ;
- $pri : T \rightarrow \{0, 1\}$ est la fonction de priorité qui associe à chaque transition temporisée la valeur 0 et à chaque transition immédiate la valeur 1 ;
- $W : T \rightarrow \mathtt{R}^+$ est une fonction qui associe à chaque transition temporisée un taux de franchissement et à chaque transition immédiate un poids ;
- $M_0 : P \rightarrow \mathtt{N}$ est le marquage initial du réseau.

Remarque : Les poids sont utilisés pour le calcul des probabilités de franchissement des transitions immédiates et éventuellement pour la résolution probabiliste des conflits entre plusieurs transitions immédiates sensibilisées dans un même marquage.

2.4.7.2 Conflit dans les RdPSG

Lorsque deux ou plusieurs transitions sont en conflit dans un réseau de Petri stochastique généralisé, deux politiques de sélection de la transition qui sera tirée sont envisageables :

- *Politique de présélection* : la transition qui va franchir est présélectionnée selon une certaine métrique, comme la priorité. C'est la politique la mieux adaptée pour les transitions immédiates.

- *Politique de course* : la transition qui a le délai de franchissement le plus court sera franchie. Cette politique est généralement adaptée pour résoudre les situations de conflit entre les transitions temporisées.

2.4.7.3 Politique de mémoire

Le délai de franchissement d'une transition dans un RdPSG correspond à un temporisateur. Quand la transition est sensibilisée, son temporisateur commence à décrémenter et la transition sera franchie dès qu'il atteint la valeur zéro. Lorsque plusieurs transitions temporisées sont franchissables dans un marquage M, le tir de l'une d'entre elles conduit à un autre marquage M', et par conséquent, certaines transitions peuvent devenir non franchissables avant l'écoulement de tous leurs délais de franchissement. La manière dont les temporisateurs sont affectés aux transitions dans le marquage M' sachant leurs valeurs atteintes dans le marquage M est connu sous le nom de *politique de mémoire*. Deux mécanismes de base sont considérés [96] :

- *Continuer* : les temporisateurs associés aux transitions gardent les mêmes valeurs et continuent à décrémenter dans le nouveau marquage.
- *Réinitialiser* : les temporisateurs associés aux transitions sont réinitialisés, c'est-à-dire, des nouvelles valeurs sont affectées à ces temporisateurs..

Dans le cas des RdPSG où les transitions temporisées ont des délais de franchissement distribués exponentiellement, c'est le mécanisme "*Réinitialiser*" qui est appliqué à cause de la perte de mémoire de la loi exponentielle.

2.4.7.4 Politique de service

Le *degré de franchissement* d'une transition t dans un marquage M, noté $ED(t, M)$ est égal au nombre de fois qu'elle puisse être franchie dans ce marquage. Formellement on écrit [12] :

$$ED(t,M) = k \text{ ssi } \begin{cases} \forall\, p \in^{\cdot} t, M(p) \geq k.Pré(p,t) \\ \text{et } \exists p \in^{\cdot} t, M(p) < (k+1).Pré(p,t). \end{cases}$$

Lorsque le degré de franchissement est supérieur à 1, les transitions peuvent se comporter selon trois politiques, dites *politiques de service*, qui sont [97, 12] :

1. **Serveur unique (single server) :** à chaque instant, un seul client est servi à la fois. Ainsi, pour un marquage donné M, le taux de franchissement d'une transition t de loi exponentielle est : $W(t, M) = W(t)$. C'est la sémantique par défaut des réseaux de Petri stochastiques généralisés.
2. **Serveurs multiples (K) (multiple servers) :** au plus, K clients peuvent être servis en parallèle. Le taux de franchissement dans ce cas est :

$$W(t,M) = min(K, ED(t,M)).W(t)$$

3. **Serveurs infinis (infinite servers) :** le nombre de clients pouvant être servis en parallèle est infini. Ainsi le taux de franchissement est : $W(t, M) = ED(t, M).W(t)$.

2.4.7.5 Processus stochastique associé à un RdPSG

L'introduction de la fonction W dans les réseaux de Petri permet la définition de l'aspect stochastique des RdPSG. La probabilité de franchissement d'une transition t_k sensibilisée dans un marquage M_i est définie par l'expression générale suivante [5, 96] :

$$P\{t_k|M_i\} = \frac{w_k}{q_i}$$

Où, q_i est donnée par :

$$q_i = \sum_{t_k \in E(M_i)} w_k$$

avec : $E(M_i)$ est l'ensemble des transitions sensibilisées dans le marquage M_i. Les paramètres w_k représentent les poids des transitions immédiates sensibilisées dans le marquage M_i si ce dernier est évanescent. Par contre, si le marquage M_i est tangible, les paramètres w_k représentent les taux de franchissement des transitions temporisées sensibilisées dans ce marquage.

Le *temps moyen de séjour* TS_i dans un marquage M_i correspond au temps passé par le réseau dans cet état, et il est donné par :

$$TS_i = \frac{1}{q_i}$$

En effet, le temps moyen de séjour dans un marquage correspond au délai de franchissement de la transition qui a provoqué le changement d'état (la transition qui a le délai de franchissement le plus court). Par conséquent, le temps moyen de séjour dans un marquage évanescent est nul.

2.4.7.6 Résolution numérique des RdPSG

Le processus stochastique associé à un RdPSG borné avec un marquage initial M_0 est un processus stochastique semi-markovien à temps continu, homogène, irréductible et à espace d'états fini [96]. Les processus semi-markoviens peuvent être analysés en déterminant les chaînes de Markov incluses qui décrivent les transitions entre les différents états du processus.

Dans le cas d'un RdPSG, la chaîne de Markov incluse (CMI) peut être reconnue en oubliant le concept du temps et en se focalisant sur l'ensemble des états du processus semi-markovien. Les spécifications du RdPSG sont suffisantes pour le calcul des probabilités de transition entre les états de la chaîne. L'ensemble d'accessibilité A est composé du sous-ensemble de marquages tangibles TS et du sous-ensemble de marquages évanescents ES tels que :

$$A = TS \cup ES,\ TS \cap ES = \emptyset$$

La matrice des probabilités de transition U de la chaîne de Markov incluse est obtenue en utilisant l'expression suivante :

$$u_{ij} = \frac{\sum_{t_k \in E_j(M_i)} w_k}{q_i}$$

De cette manière, à l'exception des élément diagonaux de la matrice U, toutes les probabilités de transition de la chaîne de Markov incluse peuvent être calculées indépendamment de la nature de la transition (immédiate ou temporisée).

En ordonnant les marquages de manière à ce que les marquages évanescent correspondent aux premières entrées de la matrice et les marquages tangibles correspondent aux dernières entrées, la matrice U peut être décomposée comme suit :

$$U = A + B = \begin{pmatrix} C & D \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ E & F \end{pmatrix}$$

Les éléments de la matrice A correspondent aux changements des marquages provoqués par le tir des transitions immédiates ; en particulier, ceux qui sont de la sous-matrice C sont les probabilités de transition d'un marquage évanescent à un autre marquage évanescent, quand à ceux qui sont de la sous-matrice D représentent les probabilités de transition d'un marquage évanescent à un marquage tangible. D'une manière similaire, les éléments de la matrice B correspondent aux changements des marquages provoqués par le tir des transitions temporisées : E est la sous matrice des probabilités de transition d'un marquage tangible à un marquage évanescent, et la sous-matrice F est celle des probabilités de passage d'un marquage tangible à un autre marquage tangible.

La distribution des probabilités d'états ψ de la CMI après l'étape n (après n transitions) peuvent être calculées en utilisant l'expression suivante [96] :

$$\psi(n) = \psi(0)U^n$$

Où, $\psi(o)$ est la distribution initiale des probabilités d'états de la CMI. La distribution stationnaire est obtenue après la résolution du système d'équations linéaires suivant :

$$\begin{cases} \psi \cdot U &= \psi \\ \psi \cdot 1^T &= 1 \end{cases}$$

où : 1^T est le vecteur de la même taille que ψ avec tous les composants égaux à 1.

La méthode présentée ci-dessus est efficace quand le nombre de marquages évanescents est petit. Toutefois, elle exige le calcul des probabilités stationnaires des marquages éva-

nescents qui sont connues à priori nulles (car le temps passé dans un tel marquage est nul).

Afin de limiter le calcul des probabilités transitoires et stationnaires aux marquages tangibles, les *chaînes de Markov incluses réduites* (CMIR) sont utilisées, en supprimant les états correspondants aux marquages évanescents. La matrice des probabilités de transition U' de la chaîne de Markov incluse est obtenue de la manière suivante [5] :

$$U' = F + E.H$$

telle que :

$$H = \begin{cases} \left(\sum_{k=0}^{\infty} C^k\right).D & \text{pas de boucles entre marquage évanecents,} \\ [I - C]^{-1}.D & \text{il existe des boucles entre les marquages évanescents.} \end{cases}$$

Quand la matrice des probabilités de transitions est obtenue, on utilise la technique standard de calcul des distributions des probabilités stationnaires (correspondants seulement aux marquages tangibles) et les différents paramètres de performances.

2.4.7.7 Calcul des paramètres de performance

L'analyse qualitative des RdPSG permet de vérifier la justesse du modèle et les propriétés qualitatives du système modélisé, telles que la vivacité et la bornitude, tandis que l'analyse quantitative consiste à calculer les probabilités stationnaires et les indices de performance. Néanmoins, le calcul des probabilités stationnaires et des indices de performances ne sont possibles que si la l'ergodicité du réseau est vérifiée, ce qui garantit l'existence du régime stationnaire. Les deux propriétés suivantes permettent de vérifier l'ergodicité :

- Un RdPSG borné et tel que son graphe des marquages accessibles est fortement connexe est ergodique [12, 98].
- Un RdPSG borné est ergodique s'il admet le marquage initial comme état d'accueil [12, 99].

Dans ce qui suit, nous allons présenter les principaux indices de performance des RdPSG, qui sont : la fréquence moyenne de franchissement des transitions, le nombre moyen de marques dans une place, le temps moyen de séjour d'une marque dans un sous-réseau, et la probabilité qu'un évènement particulier se produit.

Fréquence moyenne de franchissement d'une transition ($\bar{\lambda}\,(t_i)$) : Cet indice correspond au nombre moyen de tirs de la transition t_i en une unité de temps. Il est calculé par :

$$\bar{\lambda}\,(t_i) = \sum_{M_j \in E(t_i)} \lambda_i(M_j).\pi_j$$

Où :

- $E(t_i)$ est l'ensemble des marquages où t_i est franchissable ;
- $\lambda_i(M_j)$ est le taux de franchissement de t_i dans M_j.

Nombre moyen de marques dans une place ($n(p)$) : le nombre moyen de marques dans une place p est calculé à l'aide de la formule :

$$n(p) = \sum_{i:M_i \in E} M_i(p).\pi_i$$

où : $M_i(p)$ est le nombre de jetons dans la place p pour le marquage M_i, et E est l'ensemble des marquages accessibles.

Temps moyen de séjour d'une marque dans un sous-réseau ($E[T]$) : correspond au délai moyen qu'un jeton passe dans une partie S d'un RdPSG. Il peut être calculé en appliquant la formule suivante :

$$E[T] = \frac{E[N]}{E[\gamma]}$$

où : $E[N]$ est le nombre de jetons dans le sous-réseau S, et $E[\gamma]$ est le taux d'arrivée effectif[2] des jetons dans S.

Probabilité d'un évènement : la probabilité qu'un évènement particulier Υ est égale à la somme des probabilités de tous les marquages dans lesquels la condition correspondante à la définition de l'évènement est vérifiée. Ainsi, elle est calculée par :

$$P\{\Upsilon\} = \sum_{m_i \in X} \pi_i$$

où : $X = \{m_i \in A(R, m_0) : \Upsilon(m_i = true)\}$ est l'ensemble des marquages accessibles où la condition Υ est vérifiée.

2.5 Conclusion

Au cours de ce chapitre, nous avons présenté des notions mathématiques nécessaires à la compréhension des modèles de files d'attente et des réseaux de Petri. Nous avons abordé la théorie des files d'attente classiques, et nous avons vu que les modèles de files d'attente standards ne permettent pas de décrire le comportement réel des clients, d'où vient la nécessité de d'utilisation des files d'attente avec rappel pour la modélisation des systèmes d'attente. Nous avons vu que le phénomène de rappel rend l'analyse des files d'attente complexe. Ainsi les résultats analytiques n'existent que pour quelques modèles et avec

2. Le taux d'arrivée effectif correspond à la somme des débits moyens des transitions qui tirent dans S.

des hypothèses contraignantes sur certains paramètres, et par conséquent, les chercheurs dans ce domaine se sont orientés vers des méthodes numériques, des approximations ou des simulations. Le modèle des files d'attente avec vacance, où le service peut devenir non disponible temporairement, a été aussi abordé dans de chapitre.

Les modèles de réseaux de Petri sont très efficaces pour l'analyse des systèmes concurrents et la détermination de leurs propriétés qualitatives, telles que le blocage, la bornitude et la vivacité. Leur point faible était leur incapacité de calcul des propriétés quantitatives, et cela est dû à l'absence de l'information temporelle. L'introduction de la notion de temps aléatoire dans les modèles de réseaux de Petri a permis d'étendre leur puissance de modélisation, et de les lier au domaine de l'évaluation des performances. Leur puissance descriptive a été encore plus mise en valeur avec l'introduction des transitions immédiates en plus des transitions temporisées dans un même modèle, ce qui a donné naissance aux réseaux de Petri stochastiques généralisés.

Dans le chapitre suivant, nous allons proposer dans un premier temps une modélisation de capteurs sans fil avec les files d'attente avec rappel, ainsi qu'une approche d'analyse des performances. Dans un second temps, nous allons étudier le même système en utilisant les réseaux de Petri stochastiques généralisés, en introduisant la notion de vacance pour la réduction de la consommation d'énergie.

CHAPITRE 3

MODÉLISATION DES RÉSEAUX DE CAPTEURS

3.1 Introduction

Les réseaux de capteurs sans fil évoluent sans cesse et continuent à gagner leur place dans plusieurs domaines, aussi bien militaires que civils. Nous avons vu dans le premier chapitre qu'ils sont des réseaux très contraints en ressources. Leur portée du radio est très restreinte, ce qui exige une communication multi-sauts au lieu d'envoyer l'information directement au sink. De plus, dans la majorité des WSN, les nœuds capteurs opèrent avec des batteries irremplaçables, à cause de la nature des WSN. De ce fait, leurs sources d'énergie sont très limitées, et la durée de vie devient très courte à cause de l'épuisement d'énergie des nœuds capteurs. Une autre limitation concerne les buffers des nœuds capteurs. Chacun d'entre eux dispose d'un buffer de petite capacité, et par conséquent, des messages reçus par ce nœud risque d'être perdus à cause de l'overflow de ce buffer.

Vue l'importance de ces réseaux, l'évaluation de leurs performances est nécessaire. Cependant, l'approche générale pour l'analyse et l'évaluation de ces réseaux, est d'utiliser des outils de simulation, tels que NS-2, OMNet++, OPNET, etc. Une autre alternative aux outils de simulation serait d'utiliser des techniques de modélisation formelles. Ces dernières ont l'avantage de fournir des résultats exactes.

À l'inverse de la plupart des travaux dans les WSN, qui utilisent des modèles formels et sont concentrés sur le problème de l'énergie [21, 22, 23, 24, 25], nous nous intéressons en premier lieu au problème de perte de messages. Nous allons proposé une modélisation à l'aide des files d'attente avec rappel et buffer à la fois, avec une source finie de messages. Une autre modélisation possible est faite en utilisant le formalisme des réseaux de Petri stochastiques généralisés, afin de vérifier la validité de l'approche proposée en se basant sur le modèle des FAR. Cependant, suivant cette approche qui introduit les rappels pour remédier au problème de perte de messages, il y aura des retransmissions des messages,

ce qui entraine une consommation additionnelle de l'énergie. Afin d'essayer de trouver un équilibre entre le problème de perte de messages et la consommation de l'énergie, nous essayons de réduire l'énergie perdue durant les temps libres des capteurs en introduisant la notion de vacance. Ainsi, nous proposons deux modèles de RdPSG basés sur la théorie des files d'attente avec vacance des serveurs. Ces modèles correspondent à la mise en veille et à la mise en off des nœuds capteurs, respectivement.

Dans la section suivante, nous présentons les travaux connexes à notre contribution. Dans la section 3.3, nous décrivons en détail le modèle de FAR modélisant une portion du WSN. Nous donnons un algorithme récursif pour le calcul des probabilités stationnaires, et nous développons les formules des indices de performance les plus importants. Nous présentons par la suite, dans la section 3.4, le modèle de RdPSG correspondant, ainsi que les formules des mêmes indices de performance obtenues à partir de ce modèle. Dans la section 3.5, nous présentons les modèles de mise en veille et de mise en off d'un nœud capteur générique du réseau. Nous concluons ce chapitre dans la section 3.6.

3.2 Travaux connexes

La recherche dans le domaine des réseaux de capteurs sans fil en utilisant des modèles formels semble être encore à ses débuts. Nous avons trouvé peu de travaux dans ce domaine, appliquant des modèles de files d'attente, des FAR ou des réseaux de Petri.

Jiang *et al.* [21] ont utilisé la théorie des files d'attente standards M/G/1 avec une politique de vacance de type *N-vacance* pour optimiser la consommation d'énergie. Dans leur modèle, ils ont considéré des nœuds capteurs avec des buffers à capacité illimitée, et un schéma de communication plusieurs-à-un (*many-to-one*), dans lequel un groupe de nœuds communiquent seulement avec un nœud spécifique, avec un seul saut. Ce nœud peut être dans trois états différents : *libre*, si la fonction de transmission est désactivée, *occupé*, quand la transmission est activée, et *démarrage*, qui est un état transitoire entre l'état *occupé* et l'état *libre*. Initialement, le nœud est en état *libre*. Quand le nombre de paquets dans le buffer atteint le nombre N, le nœud passe à l'état *démarrage*, qui représente le temps nécessaire pour octroyer le canal radio. Après cette période, le nœud passe à l'état *occupé* et commence la transmission des paquets stockés dans le buffer. Dès que le buffer est vide[1], le nœud repasse à l'état *libre*. Les auteurs ont donné des fonctions mesurant la consommation d'énergie et la latence des messages.

Dans [22], les mêmes auteurs ont repris le même modèle avec quelques changements. Ils ont utilisé la politique de vacance $Min(N, T)$ plutôt que la politique de vacance *N-vacance*. Ils ont considéré seulement les états *libre* et *occupé*. Le nœud balance entre les

1. Ceci signifie que la politique de service utilisée est la politique de service exhaustive.

deux états selon les règles suivantes :

- Quand le buffer du capteur est vide, le nœud prend des vacances[2] successives de longueur fixe T.
- Si le nombre de paquets dans le buffer atteint la valeur N, le nœud se met dans l'état *occupé*.
- Après la m^{eme} vacance de longueur T, si le nombre de paquets dans le buffer n'atteint pas la valeur N, le nœud se met dans l'état *occupé*.

En se basant sur les résultats de la théorie des files d'attente M/G/1 avec la politique $Min(N, T)$ [100], et les résultats de Gakis [101], les auteurs ont donné des formules des longueurs moyennes des états *libre* et *occupé*, ainsi qu'une fonction mesurant la consommation d'énergie au niveau du nœud.

Récemment, Dimitriou dans [23], a appliqué un modèle de file d'attente avec rappel et vacance pour la conservation d'énergie des nœuds dans les réseaux sans fil. Chaque nœud dispose d'un buffer de capacité illimitée et d'une unité de transmission. Deux types de paquets sont considérés. Quand le nœud est occupé, les paquets de premier type sont stockés dans le buffer, tandis que les paquets du deuxième type rejoignent une orbite[3] de taille infinie. Dès que le buffer est vide, l'unité de transmission prend une vacance d'une certaine période et reprend le fonctionnement ensuite. À cet instant, si le buffer n'est pas vide, le nœud commence la transmission des messages stockés dans le buffer, d'une manière exhaustive. Sinon, il passe par une période d'écoute, durant laquelle il attend la réception d'éventuels paquets. Si aucun paquet n'est reçu durant cette période, il prend une autre période de vacance d'une durée différente de la précédente, et ainsi de suite. L'auteur a dérivé les formules de certaines métriques de performance et d'énergie à l'état stationnaire, telles que la probabilité d'état de l'unité de transmission : *occupé*, *en écoute*, *en vacance* ou *en* n^{eme} *vacance*, le nombre moyen de paquets en orbite, et le gain en énergie.

Lau *et al.* dans [24] ont proposé une modélisation d'un nœud d'un réseau de capteurs sans fil à l'aide d'une file d'attente M/G/1 avec vacances du serveur. Le serveur est le support de communication commun entre tous les nœuds capteurs, tandis que, l'agrégation de tous les buffers des nœuds est considérée comme un buffer centralisé de capacité infinie. Tous les nœuds sont synchronisés à prendre des vacances en même temps, en éteignant leur unité de transmission. Les auteurs ont donné des résultats analytiques de la latence des paquets et le pourcentage du temps passé dans l'état *occupé* par les nœuds, dans les cas d'une politique de service exhaustif et d'une politique de service avec barrière.

2. Le nœud prend une vacance signifie qu'il se met dans l'état *libre*.
3. Les paquets rejoignent l'orbite, ça veut dire qu'ils seront retransmis après un certain temps.

Afin d'alléger le problème du trou d'énergie[4] (*energy hole problem*) [102], JIANG *et al.* [103] ont proposé un modèle pour un réseau de capteurs sans fil, dans lequel la zone de déploiement des nœuds est divisée en plusieurs échelles par rapport à la distance au sink, qui est placé au milieu de la zone. Pour une échelle donnée, la charge moyenne du trafic autour de chaque nœud est analysée et calculée. Ensuite, chaque nœud est modélisé par une file d'attente M/M/1 avec une politique de vacance *N-vacance*, où la valeur du seuil N est choisie en fonction de la charge du trafic. De cette manière, la durée de vie du réseau peut être prolongée.

ZHANG et LI dans [25] ont proposé un modèle stochastique des WSN dans lequel chaque nœud capteur, avec un buffer à capacité illimitée, reste dans le mode *actif* ou dans le mode *inactif* aléatoirement et d'une manière alternative. Le mode *actif* consiste de deux phases, appelées *phase active* et *phase semi-active*. Quand un nœud est dans la *phase active*, il peut capter, transmettre, recevoir et relayer les paquets. Par contre, quand il est dans la *phase semi-active*, il peut seulement transmettre et relayer les données. Dans le mode *inactif*, le nœud est incapable d'interagir avec le monde extérieure (le reste du réseau). Le noeud en question passe entre les différents états selon les règles suivantes :

- La durée qu'un capteur reste dans le mode *inactif* est distribuée exponentiellement avec une moyenne de $1/\beta$. Après cette durée, il passe à la phase *active*.
- La durée pendant laquelle un capteur reste dans la phase active est un temps aléatoire avec une distribution exponentielle d'une moyenne de $1/\alpha$. Après cette durée, s'il y a au moins um message qui attend dans le buffer, le capteur passe à la phase *semi-active*. Dans le cas contraire, il repasse directement au mode *inactif*.
- Le capteur reste dans la phase *semi-active* jusqu'à ce qu'il transmet tous les paquets attendant dans le buffer, ensuite il passe au mode *inactif*.

L'expression explicite, à l'état stationnaire, du nombre de paquets dans le nœud, la latence des paquets, la consommation moyenne de l'énergie dans chacun des modes (ou phases), ainsi que l'énergie consommée par unité de temps pendant les transitions entre les différents modes (ou phases), sont donnés par les auteurs.

WÜCHNER *et al.* ont proposé dans [104] la modélisation des WSN par un modèle avec source finie, orbite non fiable et pannes des serveurs. Dans leur modèle, ils considèrent un nœud générique du réseau. L'orbite est le buffer local de ce nœud, et les serveurs sont les nœuds voisins[5]. Chaque nœud peut être soit *actif* soit *en panne*. Dans le cas *actif*, il peut être *occupé* s'il est en cours de transmission de message, ou *libre* dans le cas contraire. Seulement les voisins actifs et libres peuvent recevoir des messages. Quand le nœud en

4. Dans un WSN, les nœuds les plus proches du sink subissent à des trafics de paquets élevés, ce qui conduit à l'épuisement rapide de leur énergie par rapport aux nœuds qui sont loin du sink, et par conséquent la durée de vie du réseau est diminuée.

5. Les nœuds, dont le nœud en question peut en communiquer en un seul pas.

question reçoit un message, il essaye de le relayer à l'un de ses voisins (serveurs). Si aucun voisin n'est actif et libre, il le stocke localement dans son buffer (orbite) et essaye de le retransmettre ultérieurement. Les serveurs peuvent tomber en panne quand ils sont libres ou occupés. L'orbite est considérée non fiable aussi, car le nœud en question arrête de stocker des messages dans son buffer ou de retransmettre des messages quand il est en panne. Les auteurs ont représenté le modèle sous forme d'un RdPSG, et ont dérivé la CMTC sous-jacente. Les probabilités stationnaires et les indices de performance sont obtenus à l'aide de l'outil MOSEL (*Modeling, Specification and Evaluation Language*) [105].

En se basant sur le modèle étudié dans [104], Bérczes *et al.* dans [106] ont proposé un modèle de FAR avec priorité et source finie. Le serveur est l'unité de transmission et l'espace d'attente est le buffer d'un nœud particulier. Deux types de messages sont considérés. Les messages de priorité P_1 rejoignent directement le buffer, tandis que les messages de priorité P_2 rejoignent l'orbite et seront retransmis plus tard, si le serveur est *occupé* ou *inactif*. Le serveur est *occupé* s'il est dans l'état *actif* et au moins un message est dans le buffer, et *libre* s'il est actif et le buffer est vide. Le serveur commence par une *période d'écoute*. Après cette période, si aucun message n'est reçu, le serveur prend une vacance (état inactif). À la fin de cette période, s'il y a des messages de priorité P_1, il seront transmis. Dans le cas contraire, il reste dans l'état actif et commence une période d'écoute. Pendant cette période, les deux types de messages reçus seront servis. Si la période d'écoute expire sans avoir reçu aucun message, le serveur prend une autre vacance. Les auteurs ont utilisé l'outil MOSEL pour calculer plusieurs indices de performance.

Bérczes *et al.* dans [107] ont proposé un modèle très proche de celui analysé dans [106]. Quand le serveur est inactif, les messages reçus lancent un processus d'initialisation du serveur. La durée d'initialisation est distribuée exponentiellement. Deux cas sont considérés. Dans le premier cas, seuls les messages de priorité P_1 ont la capacité de lancer le processus d'initialisation, et dans cas, les messages de priorité P_2 ne peuvent être servis que durant la période d'écoute. Dans le deuxième cas, les messages de priorité P_2 peuvent aussi lancer le processus d'initialisation.

Wang *et al.* [108] ont proposé une modélisation des WSN multi-sink. Dans leur modèle, ils voient un réseau de capteurs comme un ensemble de sous-réseaux, dont chacun d'entre eux est composé d'un seul sink et plusieurs nœuds. Les nœuds envoient les informations collectées directement au sink. Chaque sous réseau est modélisé par une file d'attente $M/G/1/\infty$. Le service est le relai de l'information par le sink et le processus d'arrivée est l'envoi de paquets par les nœuds au sink. L'espace d'attente est le buffer du sink, qui est considéré infini. Les auteurs ont dérivé la taille moyenne du buffer, ainsi que le temps d'attente moyen au niveau du sink.

Tous les travaux cités ci-dessus ont considéré que la taille des buffers des nœuds capteurs est infinie. Par conséquent, les auteurs ne prennent pas en considération dans leurs modèles le phénomène de débordement des buffers et l'impact de la retransmission des paquets à cause de cette limitation de capacité, qui à une influence significative sur les performances du réseau. De plus, à l'exception des travaux cités dans [104, 106, 107], la source des messages est toujours considérée infinie. Cependant, en réalité, le nombre de messages captés, aussi important soit-il, est toujours fini.

Dans [26, 27, 28, 29] les auteurs ont pris en considération la limitation de la taille des buffers des nœuds capteurs, mais sans prendre en compte le phénomène de rappel ou de retransmission des paquets à cause du débordement des buffers des nœuds.

Luo *et al.* dans [26] ont modélisé chaque nœud par une file d'attente à capacité finie. Pour la conservation d'énergie, le nœud devient *inactif* périodiquement. L'environnement de ce nœud, qui représente l'ensemble de ses voisins, est dans l'état *off* quand tous les nœuds sont inactifs, et dans l'état *on* quand il y a au moins un voisin à l'état *actif*. Pour la résolution de leur modèle, ils ont utilisé une CMTC qui prend en considération l'état du nœud, l'état de l'environnement et le nombre de paquets dans le buffer. Les formules des probabilités d'état du nœud et de l'environnement, taux de perte et latence de paquets, débit de réception de paquets par le sink, et la consommation moyenne de l'énergie sont dérivées par le auteurs.

Pour réduire la probabilité de blocage des paquets reçus au niveau des nœuds à cause du débordement du buffer, Rustamov dans [27] a modélisé chaque nœud par une file d'attente avec deux serveurs en série[6]. Le premier serveur représente les couches *Application*, *Transport*, *Réseau* et *LLC*[7], ayant un taux de service μ_1. Tandis que le deuxième serveur est la sous-couche *MAC*, avec un taux de service μ_2. En guise de simplicité, et comme une première étude, l'auteur a considéré le système sans buffer. En analysant la chaîne de Markov correspondante au modèle, l'auteur a dérivé la probabilité de blocage en fonction du taux de réception des messages et des vitesses de traitement des deux serveurs. Dans ce modèle, l'auteur essaye de minimiser la probabilité de blocage, mais il ne prend pas en considération la retransmission des messages.

Martalò *et al.* dans [28] ont considéré des nœuds avec des buffers de taille limitée. Ils ont combiné la théorie des files d'attente et les CMTD pour déterminer la latence des paquets et le débit. Dans leur modèle, deux scénarios sont possibles. Les nœuds communiquent directement avec le sink dans le premier scénario. Dans le deuxième scénario, ils communiquent avec le sink par le biais d'un nœud intermédiaire.

6. Les paquets sont servis par le premier serveur, en suite par le deuxième serveur.
7. La couche liaison de données est divisée en deux sous-couches : LLC (*Logical Link Control*) et MAC (*Media Access Control*).

Le travail de Hu *et al.* [29] est consacré à l'analyse des buffers des nœuds et du processus de transmission des messages. Quand un nœud a des données à transmettre, il construit une table de routage, installe un chemin vers le sink et le maintient jusqu'à la fin de la transmission. L'arrivée des messages au buffer se fait selon le processus de Poisson, tandis que le temps de service est la somme des temps d'installation du chemin vers le sink, le temps d'attente dans le buffer et le temps de transmission. Les auteurs ont dérivé la taille des buffers.

Le problème d'intermittence dans la connectivité [8] du réseau dans les WSN à cause de la mobilité des nœuds est abordé par Lenin et Ramaswamy dans [109]. La totalité du réseau est modélisé par un réseau de files d'attente ouvert. Dans ce modèle, chaque nœud est vu comme un modèle de file d'attente avec un seul serveur, où le serveur est l'unité de traitement et le temps de service est le délai de routage. Le lien de communication entre les nœuds est vu comme un lien intermittent entre les files d'attente. La connectivité entre les nœuds est d'une durée aléatoire. Le réseau de files d'attente est ouvert, car les paquets peuvent arriver de l'extérieur (paquets captés). Après l'analyse de leur modèle, les auteurs ont obtenu les formules analytiques du délai moyen de bout-en-bout, la probabilité de perte de messages à cause de l'intermittence de la connectivité, le débit, et le nombre moyen de sauts.

Xiong *et al.* dans [110] se sont concentrés sur l'amélioration des performances des WSN pendant les situations de congestion. Pour améliorer le délai des messages, chaque nœud est modélisé par une file d'attente M/M/1, mais avec une politique de service LIFO au lieu de FIFO. Durant une situation de congestion, le sink reçoit beaucoup plus de messages à partir des capteurs les plus proches par rapport aux capteurs les plus loin. Pour assurer une équité de réception des messages, les auteurs ont divisé le buffer de chaque nœud en plusieurs files d'attente virtuelles, chacune est réservée à un sous arbre [9]. Quand le nœud en question veut transmettre un message, ce dernier est choisi à partir de la file d'attente virtuelle appropriée en se basant sur la taille du sous arbre.

Grâce à leur capacité de modélisation, les réseaux de Petri ont été utilisé par certains auteurs pour la modélisation des protocoles de contrôle d'accès au canal (MAC). Azgomi et Khalili [111] ont utilisé le formalisme de *réseaux de Petri colorés* [10] pour modéliser le protocole S-MAC [50]. Le même formalisme a été utilisé par Mokdad *et al.* [112] pour

8. L'intermittence dans la connectivité du réseau peut se produire à cause de plusieurs raisons. Par exemple, à cause de la mobilité des nœuds, de l'environnement où les nœuds sont déployés, et/ou dû à l'énergie des nœuds.

9. La topologie considérée est une topologie en arbre, par conséquent, chaque nœud est considéré comme le père d'un ensemble de sous-arbres.

10. Les réseaux de Petri colorés sont une extension des réseaux de Petri, où les ressources d'un même type sont identifiées par une même couleur.

modéliser et analyser le protocole EQ-MAC [113]. Dans ces deux travaux, les auteurs n'ont pas pris en compte ni les vacances ni le rappel.

3.3 Modélisation des WSN à l'aide des FAR

Nous proposons dans cette section une modélisation des réseaux de capteurs sans fil en utilisant le formalisme des files d'attente avec rappel. Notre modèle est général et ne concerne pas une topologie ou un protocole de routage spécifiques. De plus, il concerne un nœud capteur générique du réseau, de telle sorte que le modèle soit valable pour tous les nœuds du réseau.

Chaque nœud du réseau est équipé d'un buffer limité de taille $(N-1)$ et d'une unité de transmission qui peut contenir un seul message. Donc, la capacité totale de nœuds est égale à N. Pour simplifier le modèle, nous supposons que les messages ont des tailles égales, et correspondent à la longueur d'un mot mémoire dans l'espace de stockage. L'unité de transmission est supposée être bidirectionnelle, c'est-à-dire qu'elle peut émettre et recevoir des messages en même temps.

La nouveauté dans notre modèle est que nous prenons en considération la limitation des buffers des nœuds et la retransmission des messages à cause des débordements des buffers. De plus, nous considérons une source de messages à capacité limitée à R évènements (source finie). Le choix de la source finie de messages est motivé par le fait que le nombre d'événements susceptibles de se produire est toujours limité, aussi grand soit-il. Ainsi, notre modèle combine entre la limitation du buffer, le rappel (la retransmission) et la source finie de messages.

Prenons un nœud quelconque du réseau. Les autres nœuds qui sont capables de communiquer avec ce dernier, lui envoient des messages qu'ils ont capté eux-mêmes ou qu'ils ont reçu des autres nœuds. Dès la réception d'un message, s'il y un espace de stockage disponible, il sera mis en attente dans le buffer du nœud capteur et sera servi [11] (transmis) suivant la politique FIFO. Par contre, si le buffer est plein, le message doit être retransmis autant de fois que nécessaire, jusqu'à ce qu'il puisse être admis dans le buffer du nœud capteur. Les messages en attente de retransmission, c'est-à-dire, ceux qui doivent être retransmis, sont dits des messages *en orbite*.

Une représentation graphique d'un nœud générique du réseau est donné dans la figure 3.1. Les flèches continues représentent des transmissions, tandis que les flèches en pointillé représentent des retransmissions. Nous notons que, physiquement, l'orbite correspond aux

11. Dans notre modèle, le service d'un message stocké dans le buffer d'un nœud capteur correspond à sa transmission par ce dernier.

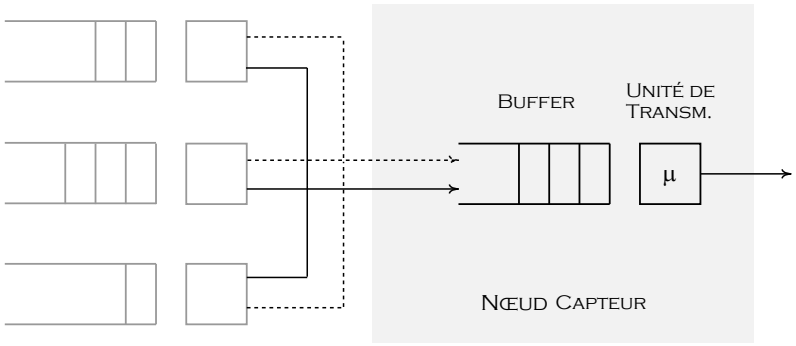

FIGURE 3.1 – Modélisation d'un nœud du réseau.

buffers des nœuds communicants avec le nœud en question.

Le nœud capteur est capable de transmettre un seul message à la fois. Le service d'un message correspond à sa transmission. Les temps de transmission sont supposés être exponentiels de paramètre $1/\mu$. De même, la retransmission des messages suit une loi exponentielle de paramètre $k.\xi$, où k est le nombre de messages en attente de retransmission. À cause de la source finie des messages, le débit de la réception des messages dépend de la taille de la source R, du nombre de messages stockés dans le buffer et du nombre de messages en attente de retransmission. Ainsi, la réception suit une loi exponentielle de paramètre $(R - j - k).\lambda$, où j est le nombre de messages dans le buffer (en attente de transmission), k est le nombre de messages en attente de retransmission et λ est le nombre moyen de messages reçus par unité de temps. Les temps de transmission, les intervalles de temps entre deux retransmissions successives et deux réceptions successives des messages sont considérés être indépendants les uns des autres.

3.3.1 Analyse du modèle

Soit j le nombre de messages au niveau du capteur en question (nombres de messages dans le buffer plus le message en cours de transmission) et k le nombre de messages qui n'ont pas été admis dans le capteur à cause du débordement du buffer (les messages en orbite). Le processus stochastique $X_t(j, k)$ qui décrit l'évolution du nombre de messages au niveau du capteur et celui en orbite est une chaîne de Markov à temps continu de taille égale à $N \times (R - N)$, où N est la capacité du capteur et $(R - N)$ est la taille de l'orbite.

Quatre évènements différents provoquent le changement d'état (transition) de la chaîne, qui sont : la réception d'un message, la fin de transmission d'un message, la réception d'un message quand le buffer est plein (ce que nécessite la retransmission du message, ie. sa mise en orbite et non sa perte), et enfin, la retransmission d'un message de l'orbite. Les différentes transitions, causées par chacun de ces évènements, avec leurs taux sont présentées dans le tableau 3.1.

La colonne "*Sens de flèche*" indique le sens de la flèche représentant une transition dans le diagramme de transition correspondant à la chaîne de Markov. Cette colonne a été ajoutée pour faciliter la lisibilité du diagramme donné dans la figure 3.2.

Évènement	**Transition**	**Taux**	**Sens de flèche**
Réception d'un message	$(j,k) \to (j+1,k)$ $0 \le j \le N-1$ et $0 \le k \le R-N$	$(R-(j+k))\,\lambda$	$\longrightarrow$
Fin trans. message	$(j,k) \to (j-1,k)$ $1 \le j \le N$ et $0 \le k \le R-N$	μ	$\longleftarrow$
Réception msg buf. plein	$(N,k) \to (N,k+1)$ $j = N$ et $0 \le k \le R-N-1$	$(R-(N+k))\,\lambda$	$\downarrow$
Retrans. d'un message	$(j,k) \to (j+1,k-1)$ $0 \le j \le N-1$ et $1 \le k \le R-N$	$k\,\xi$	$\nearrow$

TABLE 3.1 – Types de transitions de la chaîne de Markov

La figure 3.2 illustre le diagramme de transitions entre les états de la chaîne. Afin de dériver les différents indices de performance du modèle à l'état stationnaire, nous devons d'abord calculer les probabilités stationnaires. Il est clair que le graphe de transitions est fini et irréductible, donc, la chaîne de Markov est ergodique, et par conséquent, elle admet un régime stationnaire. À l'état stationnaire, le flux entrant à chaque état de la chaîne est égal au flux sortant de cet état. En appliquant cette règle, on obtient le système d'équations de (3.1) à (3.4), comme suit :

$$((R-k)\,\lambda + k\,\xi).P(0,k) = \mu.P(1,k) \quad \text{pour} \quad 0 \le k \le R-N \tag{3.1}$$

$$\begin{aligned}((R-(j+k))\,\lambda + k\,\xi + \mu).P(j,k) &= (R-(j+k-1))\,\lambda \\ &.P(j-1,k) + \mu.P(j+1,k) + (k+1)\,\xi.P(j-1,k+1) \\ &\text{pour} \quad 1 \le j < N \quad \text{et} \quad 0 \le k \le R-N\end{aligned} \tag{3.2}$$

$$\begin{aligned}((R-(N+k))\,\lambda + \mu).P(N,k) &= ((R-(N+k-1))\,\lambda).P(N-1,k) + \\ &(R-(N+k-1))\,\lambda.P(N,k-1) + (k+1)\,\xi.P(N-1,k+1) \\ &\text{pour} \quad 0 \le k \le R-N-1\end{aligned} \tag{3.3}$$

$$\mu.P(N,R-N) = \lambda.P(N,R-N-1) + \lambda.P(N-1,R-N) \tag{3.4}$$

avec $P(j,R-N+1) = 0$ pour $0 \le j \le N-2$ et $P(N,-1) = 0$ avec la condition de

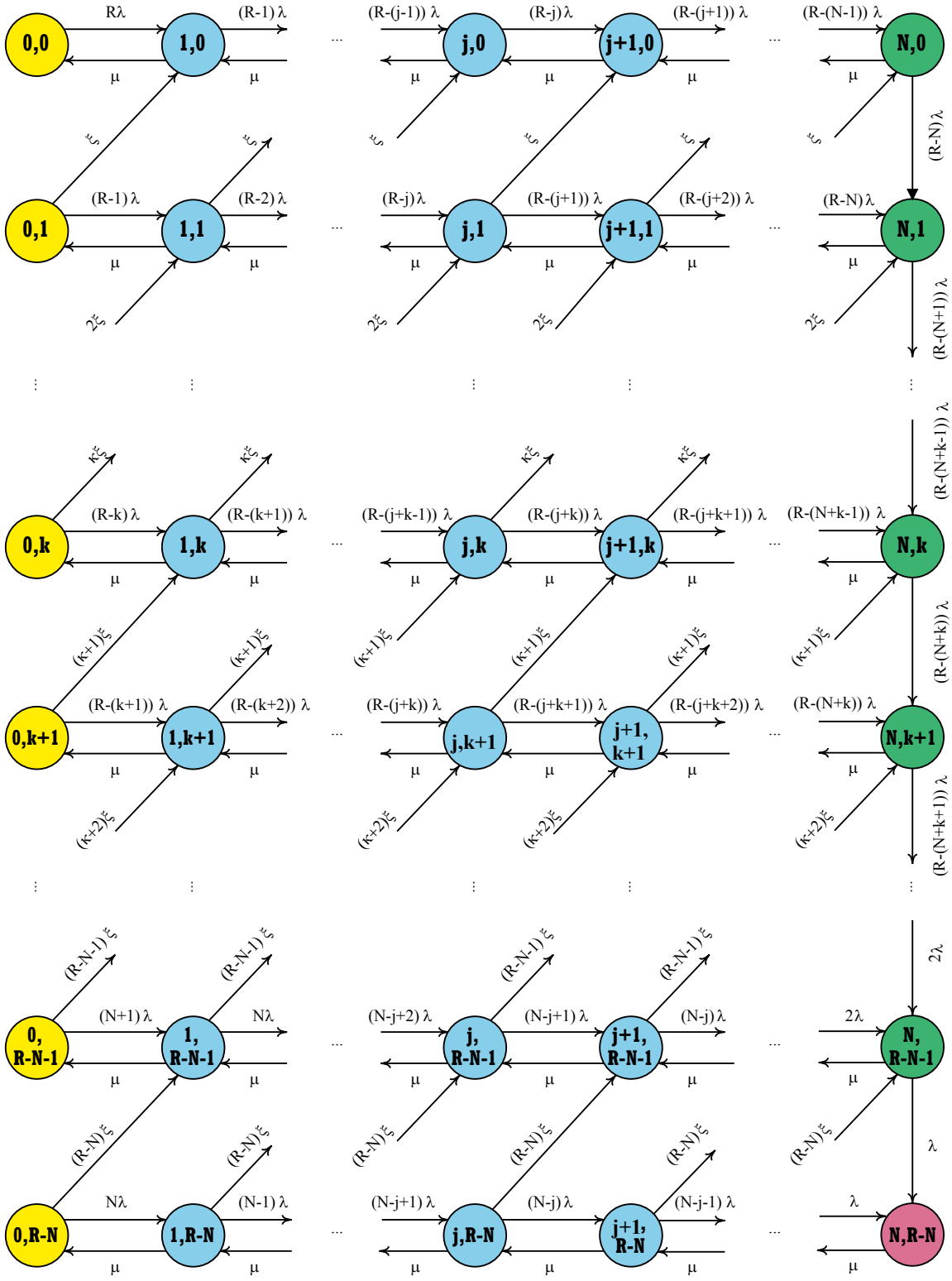

FIGURE 3.2 – Processus décrivant l'évolution de l'état du capteur

normalisation :

$$\sum_{j=0}^{N}\sum_{k=0}^{R-N} P(j,k) = 1 \tag{3.5}$$

3.3.2 Méthode de résolution

Notre modèle correspond à une file d'attente avec rappel, buffer à capacité limitée, un seul serveur et une source finie. Jusqu'à présent, aucune solution analytique n'existe pour ce modèle. Ainsi, afin d'obtenir les probabilités stationnaires, nous avons développé un algorithme récursif inspiré de la méthode numérique utilisée dans [30]. Notre algorithme est divisé en deux parties. La première partie est liée à une ligne générique (k) de la chaîne de Markov, dans laquelle nous calculons toutes les probabilités $P(j,k)$ à partir de la probabilité $P(N,k)$ d'une manière récursive. Quant à la deuxième partie, elle nous permet de calculer toutes les probabilités stationnaires à partir de la probabilité $P(N, R-N)$ en exploitant la première partie et les équations $(3.1) - (3.5)$.

3.3.2.1 Première partie

Considérons une ligne générique de la chaîne de Markov, qui correspond à la valeur k. À partir de la valeur de la probabilité $P(N,k)$, nous allons calculer toutes les probabilités $P(j,k)$ pour $0 \leq j \leq N-1$. Pour ce faire, nous allons suivre le raisonnement suivant :

En mettant,

$$S_1 = \begin{cases} x_j^{(k)} &= P(j,k) &;\quad 0 \leq j \leq N-1 \\ f &= P(N,k) & \\ O_j^{(k)} &= R-(j+k) &;\quad 0 \leq j \leq N-1 \\ l_j^{(k)} &= O_j^{(k)}\,\lambda + k\,\xi + \mu &;\quad 0 \leq j \leq N-1 \\ t_j^{(k)} &= (k+1)\,\xi.P(j-1,k+1) &;\quad 0 \leq j \leq N-1 \end{cases}$$

Ainsi, les équations d'équilibres relatives à la ligne (k) de la chaîne de Markov peuvent être réécrites de la manière suivante :

$$S_2 = \begin{cases} t_0^{(k)} &= (l_0^{(k)} - \mu)\,.\,x_0^{(k)} - \mu\, x_1^{(k)} \\ t_1^{(k)} &= -\,O_0^{(k)}\,\lambda\,.\,x_0^{(k)} + l_1^{(k)}\,.\,x_1^{(k)} - \mu\,.\,x_2^{(k)} \\ & \cdots \\ t_j^{(k)} &= -\,O_{j-1}^{(k)}\,\lambda\,.\,x_{j-1}^{(k)} + l_j^{(k)}\,.\,x_j^{(k)} - \mu\,.\,x_{j+1}^{(k)} \\ & \cdots \\ t_{N-1}^{(k)} + \mu\, f &= -\,O_{N-2}^{(k)}\,\lambda\,.\,x_{N-2}^{(k)} + l_{N-1}^{(k)}\,.\,x_{N-1}^{(k)} \end{cases}$$

En mettant maintenant, $\bar{l_j}^{(k)} = l_j^{(k)} \Big/ \mu$, $\bar{\lambda} = \lambda/\mu$ et $\bar{t_j}^{(k)} = t_j^{(k)} \Big/ \mu$. Ainsi, le système ci-dessus devient :

$$S_3 = \begin{cases} \bar{t}_0^{(k)} & = & (\bar{l}_0^{(k)} - 1) \,.\, x_0^{(k)} - x_1^{(k)} \\ \bar{t}_1^{(k)} & = & -O_0^{(k)}\, \bar{\lambda} \,.\, x_0^{(k)} + \bar{l}_1^{(k)} \,.\, x_1^{(k)} - x_2^{(k)} \\ & \cdots & \\ \bar{t}_j^{(k)} & = & -O_{j-1}^{(k)}\, \bar{\lambda} \,.\, x_{j-1}^{(k)} + \bar{l}_j^{(k)} \,.\, x_j^{(k)} - x_{j+1}^{(k)} \\ & \cdots & \\ \bar{t}_{N-1}^{(k)} + f & = & -O_{N-2}^{(k)}\, \bar{\lambda} \,.\, x_{N-2}^{(k)} + \bar{l}_{N-1}^{(k)} \,.\, x_{N-1}^{(k)} \end{cases}$$

Maintenant, nous introduisons les deux fonctions récursives $E_j^{(k)}$ et $F_j^{(k)}$ définies comme suit :

$$S_4 = \begin{cases} E_0^{(k)} & = & \bar{l}_0^{(k)} - 1 & \\ E_j^{(k)} & = & \bar{l}_j^{(k)} - O_{j-1}^{(k)}\, \bar{\lambda} \Big/ E_{j-1}^{(k)} & \text{si} \quad 1 \leq j \leq N-1 \\ F_0^{(k)} & = & \bar{t}_0^{(k)} & \\ F_j^{(k)} & = & \bar{t}_j^{(k)} + O_{j-1}^{(k)}\, F_{j-1}^{(k)}\, \bar{\lambda} \Big/ E_{j-1}^{(k)} & \text{si} \quad 1 \leq j \leq N-1 \end{cases}$$

Ainsi, le système d'équations S_3 peut être réécrit comme suit :

$$S_5 = \begin{cases} F_0^{(k)} & = & E_0^{(k)} \,.\, x_0^{(k)} - x_1^{(k)} \\ F_1^{(k)} & = & E_1^{(k)} \,.\, x_1^{(k)} - x_2^{(k)} \\ & \cdots & \\ F_j^{(k)} & = & E_j^{(k)} \,.\, x_j^{(k)} - x_{j+1}^{(k)} \\ & \cdots & \\ F_{N-1}^{(k)} & = & E_{N-1}^{(k)} \,.\, x_{N-1}^{(k)} - f \end{cases}$$

À partir du système S_5, nous pouvons déterminer les valeurs $x_j^{(k)}$ en fonction de la probabilité $P(N,k)$ (la valeur f), de la manière suivante :

$$S_6 = \begin{cases} x_{N-1}^{(k)} & = & (F_{N-1}^{(k)} + f) \Big/ E_{N-1}^{(k)} & \\ x_j^{(k)} & = & (F_j^{(k)} + x_{j+1}^{(k)}) \Big/ E_j^{(k)} & \text{pour} \quad 0 \leq j \leq N-2 \end{cases}$$

Les étapes de la première partie sont résumées dans l'algorithme 1.

3.3.2.2 Deuxième partie

La deuxième partie de notre méthode consiste à écrire un algorithme qui nous permet de calculer toutes les probabilités stationnaires en fonction de la probabilité $P(N, R-N)$, en exploitant les résultats de la première partie et les équations (3.1)-(3.4). Le fonctionnement de l'algorithme est décrit ci-dessous :

1. Nous mettons $P(N, R-N)$ égale à une constante f.

2. À partir de $P(N, R-N)$, nous calculons toutes les valeurs des $P(j, R-N) \quad (0 \leq j \leq N-1)$, en exploitant les équations du système S_6.

3. Nous calculons la valeur de $P(N, R-N-1)$ en utilisant l'équation (3.4).

Algorithm 1 Calcul des probabilités stationnaires pour une ligne générique (k)

input : λ, μ, ξ, f : réels ; N, R : entiers ;
output : $x^{(k)}[0..N]$: tableau des réels ;

▷ Calcul de $O_j^{(k)}$

function $O(j,k$:entier)
    return $R-(j+k)$;
end function

▷ Calcul de $\bar{l}_j^{(k)}$

function $\bar{l}(j,k$:entier)
    return $(\mathrm{O}(j,k)\,\lambda + k\,\xi + \mu)/\mu$;
end function

▷ Calcul de $\bar{t}_j^{(k)}$

function $\bar{t}(j,k$:entier)
    return $((k+1)\,\xi\,P(j-1,k+1))/\mu$;
end function

▷ Calcul de $E_j^{(k)}$

function $E(j,k$:entier)
    if $j=0$ **then**
        return $\bar{l}(0,k)-1$;
    else
        return $\bar{l}(j,k)-(O(j-1,k)\,\bar{\lambda})/E(j-1,k)$;
    end if
end function

▷ Calcul de $F_j^{(k)}$

function $F(j,k$:entier)
    if $j=0$ **then**
        return $\bar{t}(0,k)$;
    else
        return $\bar{t}(j,k)-(O(j-1,k)\,F(j-1,k)\,\bar{\lambda})/E(j-1,k)$;
    end if
end function

▷ Calcul de $x_j^{(k)}$

$x^{(k)}[N] \leftarrow f$;
for $(j=(N-1)$ to $0)$ **do**
    $x^{(k)}[j] \leftarrow (F(j,k)+x^{(k)}[j+1])/E(j,k)$;
end for

4. En utilisant les équations du système S_6, nous calculons toutes les probabilités $P(j, R-N-1) \quad (0 \leq j \leq N-1)$.

5. Les probabilités $P(N,k) \quad (0 \leq k \leq R-N-2)$ sont calculées à partir des probabilités $P(N,k+1)$, $P(N-1,k+1)$ et $P(N-1,k+2)$, en utilisant l'équation (3.3).

6. Nous calculons les probabilités $P(j,k)$ pour $(0 \leq j \leq N-1)$ et $(0 \leq k \leq R-N-2)$ à l'aide des équations du système S_6.

7. En appliquant la condition de normalisation (l'équation 3.5), la somme de toutes les probabilités doit être égale à un (1). Pour rendre cette condition vérifiée, nous divisons chaque $P(j,k)$ par la somme de toutes les $P(j,k)$.

Les étapes de la deuxième partie sont résumées dans l'algorithme 2.

3.3.3 Indices de performance

Une fois que nous avons les probabilités stationnaires, nous calculons les principaux indices de performance du modèle, qui sont : la probabilité de débordement du buffer, le nombre moyen de messages stockés dans le buffer, le nombre moyen de messages en cours de transmission, le nombre moyen de messages en attente de retransmission, le débit moyen de réception des messages, le débit moyen de retransmission des messages, le temps moyen d'attente des messages dans le buffer, le temps moyen de retransmission, la latence moyenne des messages, et le débit moyen de transmission.

- **La probabilité de débordement du buffer (P_{buf}) :** correspond à la probabilité que le buffer soit plein,

$$P_{buf} = \sum_{k=0}^{R-N} P(N,k)$$

- **Le nombre moyen de messages stockés dans le buffer du capteur (Q) :** représente le nombre moyen de messages en attente de transmission,

$$Q = \sum_{j=1}^{N} \sum_{k=0}^{R-N} (j-1).P(j,k)$$

- **Le nombre moyen de messages en cours de transmission (U) :**

$$U = 1 - \sum_{k=0}^{R-N} P(0,k)$$

- **Le nombre moyen de messages en attente de retransmission (K) :** représente le nombre moyen de messages qui n'ont pas été admis dans le capteur à la

Algorithm 2 Calcul des probabilités stationnaires

Input : λ, μ, ξ, f : réels ; N, R, M : entiers ;
Output : $P[0..N, 0..R-N]$: tableau des réels ;

$M \leftarrow R - N$;
$P(N, M) = f$; ①

▷ Calcul de P(j,M) pour j = 0..N-1

$P(N-1, M) = (F(N-1, M) + f)/E(N-1, M)$;
for $(j = (N-2)$ to $0)$ **do**
    $P(j, M) \leftarrow (F(j, M) + P(j+1, M))/E(j, M)$; ②
end for

▷ Calcul de P(N,M-1)

$P(N, M-1) \leftarrow (\mu\, P(N, M) - \lambda\, P(N-1, M))/\lambda$; ③

▷ Calcul de P(j,M-1) pour j= 0..N-1

for $(j = (N-1)$ to $0)$ **do**
    $P(j, M-1) \leftarrow (F(j, M-1) + P(j+1, M-1))/E(j, M-1)$; ④
end for

▷ Calcul de P(j,k) pour j = 0..N et k = 0..R-N-2

for $(k = (M-1)$ to $1)$ **do**

▷ Calcul de P(N,k) pour k = 0..R-N-2

    $A \leftarrow ((R - (N+k))\,\lambda + \mu)\,.\,P(N, k)$;
    $B \leftarrow ((R - (N+k-1))\,\lambda)\,.\,P(N-1, k)$;
    $C \leftarrow (k+1)\,\xi\,.\,P(N-1, k+1)$;
    $D \leftarrow (R - (N+k-1))\,\lambda$;
    $P(N, k-1) = (A - B - C)/D$; ⑤

▷ Calcul de P(j,k) pour j = 0..N-1 et k = 0..R-N-2

    for $(j = (N-1)$ to $0)$ **do**
        $P(j, k-1) \leftarrow (F(j, k-1) + P(j+1, k-1))/E(j, k-1)$; ⑥
    end for
end for

▷ Normalisation des probabilités (*somme* est la somme des P(j,k))

for $(j = 0$ to $N)$ **do**
    for $(k = 0$ to $M)$ **do**
        $P(j, k) \leftarrow P(j, k)/somme$; ⑦
    end for
end for

première réception,

$$K = \sum_{j=0}^{N} \sum_{k=0}^{R-N} k.P(j,k)$$

- **Le débit moyen de réception des nouveaux messages ($\bar{\lambda}$) :** correspond au débit moyen de réception des messages envoyés par les nœuds voisins pour la première fois,

$$\bar{\lambda} = (R - Q - U - K).\lambda$$

- **Le débit moyen de réception des messages retransmis ($\bar{\xi}$) :** correspond au débit de retransmission, par les nœuds voisins, des messages qui n'ont pas été admis dans le capteur à la première réception,

$$\bar{\xi} = K.\xi$$

- **Le débit global moyen de réception des messages (Λ) :**

$$\Lambda = \bar{\lambda} + \bar{\xi}$$

- **Le temps moyen d'attente des messages dans le buffer (W_q) :** calculé à partir du nombre moyen de messages stockés dans le buffer du capteur en utilisant la formule de Little,

$$W_q = \frac{Q}{\Lambda.(1 - P_{buf})}$$

- **Le temps moyen nécessaire à l'admission d'un message (W_o) :** représente le temps moyen nécessaire à l'admission d'un message qui n'a pas été stocké dans le capteur à la première réception,

$$W_o = \frac{K}{\Lambda.P_{buf}}$$

- **La latence moyenne des messages (W_l) :** correspond au temps moyen qui sépare l'instant de réception d'un message de l'instant de la fin de transmission du dernier bit de ce massage. Dans ce cas, il est égal à la somme du temps moyen d'attente des messages dans le buffer, du temps nécessaire à l'admission d'un message dans le capteur et du temps de transmission,

$$W_l = W_q + W_o + \frac{1}{\mu}$$

- **Le débit moyen de transmission (T) :** correspond au débit moyen de transmis-

sion des messages stockés dans le buffer,

$$T = \mu.U$$

- **Le taux d'oisiveté de l'unité de transmission (O) :** correspond au pourcentage de temps dans lequel l'unité de transmission soit oisive,

$$O = (1 - U) \times 100\%$$

3.4 Modélisation des WSN à l'aide des RdPSG

Dans le but de vérifier les résultats obtenus avec le modèle de FAR et la justesse de notre algorithme, nous avons proposé une autre modélisation du même système (un capteur générique du réseau) à l'aide d'un modèle de réseau de Petri stochastique généralisé. Nous avons choisi ce formalisme pour leur simplicité et leur capacité de modélisation, et surtout, pour bénéficier des outils développés dans ce domaine, notamment l'outil GreatSPN [114], qui nous permet de calculer facilement les différents indices de performance de notre modèle, une fois que nous développons et nous introduisons les formules correspondantes.

Nous présentons dans cette section le modèle de RdPSG décrivant un capteur générique du réseau. Nous donnons ensuite les formules des paramètres de performance dérivées à partir de ce modèle. La figure 3.3 présente notre modèle de RdPSG, dans lequel :

- La place *Source* représente la source des messages ;
- La place *Choix* représente la condition qu'un message reçu puisse être admis dans le capteur ou il devrait être retransmis. Cela dépend du nombre de messages dans le buffer du capteur ;
- La place *DispCap* représente le nombre de places disponibles pour le stockage des messages, y compris le message en cours de transmission ;
- La place $Buffer$ représente le buffer du capteur. Elle contient le nombre de messages en attente de transmission ;
- La place *CanLib* indique si l'unité de transmission du capteur est libre ou non, c'est-à-dire, s'il y a un message en cours de transmission ou non ;
- La place *EnTran* contient le message en cours de transmission ;
- La place *Orbite* contient les messages en attente de retransmission, c'est-à-dire, les message qui n'ont pas été admis dans le capteur à la première réception.

Le marquage initial du réseau est :
$M_0 = \{M(Source), M(Choix), M(DipCap), M(Buffer), M(CanLib), M(EnTran), M($

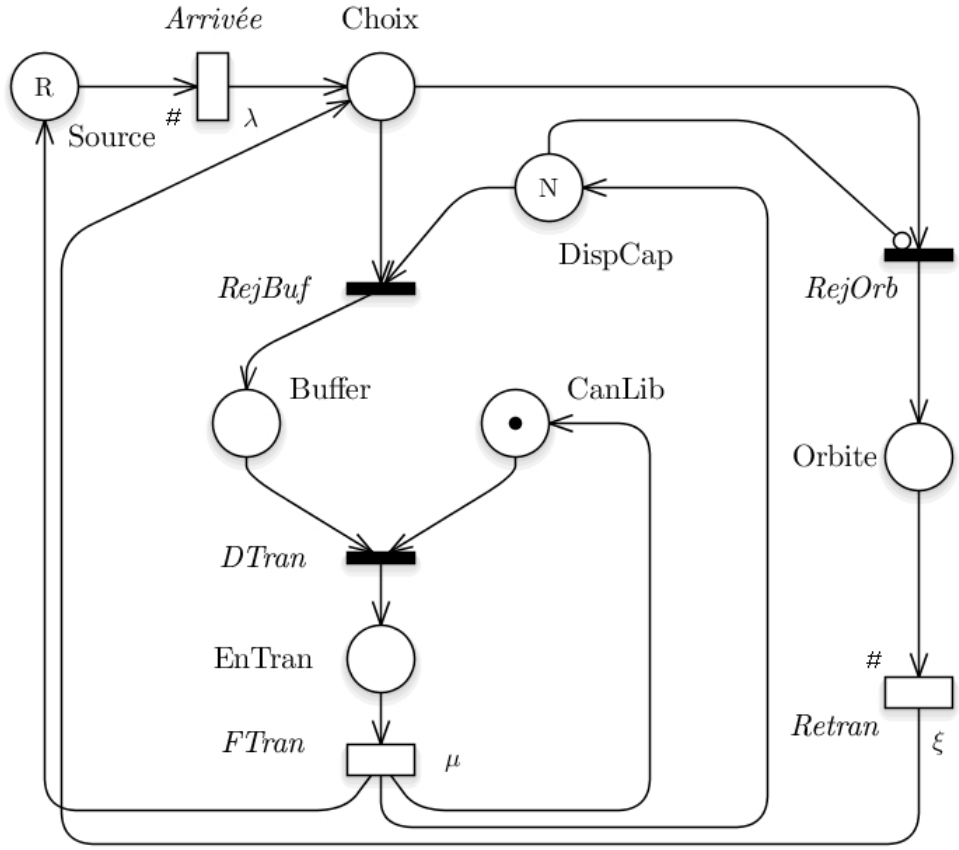

FIGURE 3.3 – Modèle de RdPSG d'un capteur du réseau

$Orbite)\} = \{R, 0, N, 0, 1, 0, 0\}$, ce qui signifie que, initialement, aucun message n'a encore été reçu, aucun message n'est en cours de transmission, aucun message n'est en attente de retransmission et que le buffer est vide.

Remarque : dans tous les modèles proposés, les transitions immédiates sont représentées par des traits et les transitions temporisées par des rectangles. D'autre part, nous supposons que le franchissement des transitions temporisées est atomique, c'est-à-dire, que les marques restent comptabilisées dans les places en entrée, tant que leur franchissement n'est pas terminé.

Le franchissement de la transition *Arrivée* représente la réception d'un message envoyé pour la première fois par un nœud voisin. Ainsi, la place *choix* reçoit un jeton, ce qui représente la condition qu'un message est prêt pour être transmis. La sémantique de service de cette transition est *à serveurs infinis*, ceci signifie que son taux de tir est égal à $ED(Arrivée, M).\lambda$, où $ED(Arrivée, M)$ est le degré de franchissement de la transition *Arrivée* dans un marquage M. Dans ce cas, le taux de franchissement de cette transition est dépendant du marquage. Ceci est indiqué par le symbole # placé à côté de la transition *Arrivée*.

À l'arrivée d'un message à la place *Choix*, si la place *DispCap* contient au moins

un jeton qui correspond à une place de stockage disponible dans le capteur, la transition immédiate $RejBuf$ sera tirée instantanément et le marquage de la place $Buffer$ sera incrémenté de 1, et celui de la place $DispCap$ sera décrémenté de 1. Si la place $CanLib$ contient un jeton, qui signifie que l'unité de transmission est libre, la transition $DTran$ sera tirée immédiatement. Son franchissement consomme le jeton de la place $CanLib$, décrémente le nombre de jetons de la place $Buffer$ de 1 et en dépose un dans la place $EnTran$, ce qui signifie que la transmission d'un nouveau message est commencée et que l'unité de transmission est passée de l'état oisif à l'état occupé. Par contre, si la place $DispCap$ est vide (le buffer est plein et il y a un message en cours de transmission), c'est plutôt la place $RejOrb$ qui sera tirée, et par conséquent, le message rejoint l'orbite (va être retransmis plus tard).

Une fois en orbite (en attente de retransmission), chaque message est retransmis après un délai de temps aléatoire, ce qui est matérialisé par le franchissement de la transition temporisée $Retran$. La sémantique de cette transition doit être à serveurs infinis, c'est-à-dire, le taux de tir est dépendant du marquage de la place $Orbite$, ce qui est représenté par le symbole # à côté de la transition $Retran$. Dès le franchissement de cette transition temporisée, un jeton est déposé dans la place $Choix$. Ceci matérialise l'évènement de retransmission d'un message. Ainsi, plusieurs messages peuvent être retransmis simultanément.

D'autre part, à la fin de la transmission d'un message (tir de la transition $FTran$), un jeton sera déposé dans la place $CanLib$ pour représenter le fait que l'unité de transmission est passée de l'état occupé à l'état libre. Un jeton est également déposé dans la place $DispCap$ car une place vient d'être libérée dans le buffer, et un autre jeton qui représente la fin de transmission d'un message, est déposé dans la place $Source$. Le franchissement de la transition $FTran$ est à une sémantique à *serveur unique*, car le nœud capteur dispose d'une seule unité de transmission qui est capable de transmettre un seul message à la fois.

3.4.1 Analyse du modèle et indices de performance

Le modèle de RdPSG présenté dans la figure 3.3 est borné, vivant, sans blocage et le marquage initial est un état d'accueil. Par conséquent, ce modèle admet un état stationnaire. Nous notons par $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ la distribution des probabilités de marquage à l'état stationnaire, où π_i est la probabilité que le processus soit à l'état M_i et $M_i(p)$ est le nombre de jetons dans la place p dans le marquage M_i. Nous notons par A l'ensemble des marquages tangibles accessibles et par $E(t)$ l'ensemble des marquages tangibles accessibles dans lesquels la transition t est franchissable.

En ayant les probabilités d'état stationnaires π, divers indices de performance intéressants peuvent être calculés. Nous donnons dans ce qui suit, les formules explicites des

paramètres d'un capteur générique d'un réseau de capteur sans fil, en se basant sur le modèle de RdPSG illustré dans la figure 3.3.

- **La probabilité de débordement du buffer (P_{buf}) :** correspond à la probabilité que la place *DispCap* ne contienne aucune marque,

$$P_{buf} = \sum_{i:M_i(DispCap)=0} \pi_i$$

- **Le nombre moyen de messages stockés dans le buffer du capteur (Q) :** correspond au nombre moyen de marques présentes dans la place *Buffer*,

$$Q = \sum_{i:M_i \in A} M_i(Buffer).\pi_i$$

- **Le nombre moyen de messages en cours de transmission (U) :** correspond au nombre moyen de marques présentes dans la place *EnTran*,

$$U = \sum_{i:M_i \in A} M_i(EnTran).\pi_i$$

- **Le nombre moyen de messages en attente de retransmission (K) :** correspond au nombre moyen de marques présentes dans la place *Orbite*,

$$K = \sum_{i:M_i \in A} M_i(Orbite).\pi_i$$

- **Le débit moyen de réception des nouveaux messages ($\bar{\lambda}$) :** correspond à la fréquence (débit) de la transition *Arrivée*,

$$\bar{\lambda} = \sum_{i:M_i \in E(Arrivée)} \lambda.M_i(Source).\pi_i$$

- **Le débit moyen de réception des messages retransmis ($\bar{\xi}$) :** correspond à la fréquence (débit) de la transition *Retran*,

$$\bar{\xi} = \sum_{i:M_i \in E(Retran)} \xi.M_i(Orbite).\pi_i$$

- **Le débit global moyen de réception des messages (Λ) :** correspond à la somme des fréquences moyennes des transitions *Arrivée* et *Retran*,

$$\Lambda = \bar{\lambda} + \bar{\xi}$$

- **Le temps moyen d'attente des messages dans le buffer (W_q) :** correspond au temps moyen de séjour d'une marque dans la place *Buffer*. Il est obtenu en utilisant la formule de LITTLE,

$$W_q = \frac{Q}{\Lambda.(1 - P_{buf})}$$

- **Le temps moyen nécessaire à l'admission d'un message (W_o) :** correspond au temps moyen de séjour d'une marque dans la place *Orbite*,

$$W_o = \frac{K}{\Lambda.P_{buf}}$$

- **Le débit moyen de transmission (T) :** correspond à la fréquence (débit) de la transition *FTran*,

$$T = \sum_{i:M_i \in E(FTran)} \mu.\pi_i$$

Les autres indices restent inchangés par rapport au modèle de FAR.

3.5 Modélisation avec vacances

Nous avons vu que, selon le modèle précédent, il n'y a aucune possibilité de perte de messages. Cependant l'inconvénient de ce modèle c'est qu'il ignore la consommation d'énergie des nœuds capteurs, qui augmente à cause des rappels (retransmission) des messages. L'unité de transmission est l'organe le plus gourmand en matière de consommation d'énergie qui est un critère très important dans les WSN [13]. Dans l'intention de contourner ce problème, nous proposons deux modèles de RdPSG basés des politiques de vacances du serveur.

Dans les deux modèles, nous essayons de profiter des périodes où l'unité de transmission est libre pour mettre le nœud capteur dans un état qui nécessite une consommation d'énergie plus basse. Le premier modèle consiste à mettre le nœud capteur *en veille*, c'est-à-dire, désactiver la fonction de transmission des messages. Donc, pendant une période de mise en veille, le nœud peut recevoir des messages mais ne peut pas en transmettre. Dans le deuxième modèle, le nœud est *mis en off*, c'est-à-dire qu'il ne peut ni recevoir ni transmettre des messages pendant une période de mise en off. À notre connaissance, c'est la première fois qu'un modèle formel dans les WSN combine le rappel, le buffer finie, la source finie et les vacances du serveur.

La politique de service dans les deux modèles est celle du *service exhaustif*, ce qui signifie que le nœud ne sera en vacance que s'il n'y a aucun message à transmettre. Nous avons opté pour cette politique de service, car nous ne voulons pas bloquer des messages qui sont déjà en attente de transmission tout au long d'une période de vacance. Nous

supposons aussi que le nœud capteur prend des vacances multiples, car notre but est de profiter le maximum des temps où l'unité de transmission est libre pour gagner un maximum d'énergie.

3.5.1 Modèle de mise en veille

Si l'unité de transmission est l'organe le plus gourmand en matière de consommation d'énergie, la fonction de transmission des messages domine cette consommation par rapport à la réception et le *idle listening* [21]. De ce fait, nous proposons dans ce modèle la désactivation de cette opération en mettant le nœud capteur en veille quand cela est possible et sans dégradation remarquable des performances du réseau.

À chaque fois que le nœud n'a aucun message à transmettre, c'est-à-dire qu'il n'a aucun message en cours de transmission et son buffer est vide, il sera mis immédiatement en veille. Durant la période où le nœud capteur est en veille, il continue à recevoir des messages des nœuds voisins et il les stocke dans son buffer, mais sans pouvoir les transmettre. Le nœud sera réveillé quand le nombre de messages stockés dans son buffer atteint un seuil n, et à partir de là, il reprend la transmission des messages.

Dans certaines situations où la densité de trafic est très basse, le nœud capteur une fois mis en veille, il pourrait rester longtemps avant qu'il soit réveillé. Pour éviter cette situation, nous proposons de réveiller le capteur après un certain délai moyen d. Dans ce cas, le nœud capteur est réveillé soit par l'atteinte du nombre de messages en attente le seuil n, soit par l'écoulement de ce délai.

Il est clair que la mise en veille du nœud affecte quelques indices de performance du capteur, et par conséquent, la performance totale du réseau, à cause du temps d'attente supplémentaire des messages quand la fonction de transmission est désactivée. C'est le cas de la latence des messages et le débit de transmission. Donc, les valeurs du seuil n et du délai d sont cruciales est doivent être choisies avec précaution.

La figure 3.4 présente le modèle de RdPSG correspondant à la mise en veille. Dans ce modèle, nous avons introduit une nouvelle place $CapVac$, qui représente l'état *en veille* du nœud capteur, et deux transitions immédiates $DVac$ et $FVac$ qui représentent le début et la fin d'une période de mise en veille, respectivement, et une transition temporisée $FVac_2$. La transition $FVac_2$ a comme taux de franchissement β, et par conséquent, un délai moyen de franchissement égal à $1/\beta$, qui correspond au délai d.

Quand la place $Buffer$ ne contient aucune marque, ce qui est modélisé par l'arc inhibiteur reliant la place $Buffer$ et la transition $DVac$, et la place $CanLib$ contient un jeton, ce qui exprime la disponibilité de l'unité de transmission, la transition $DVac$ sera

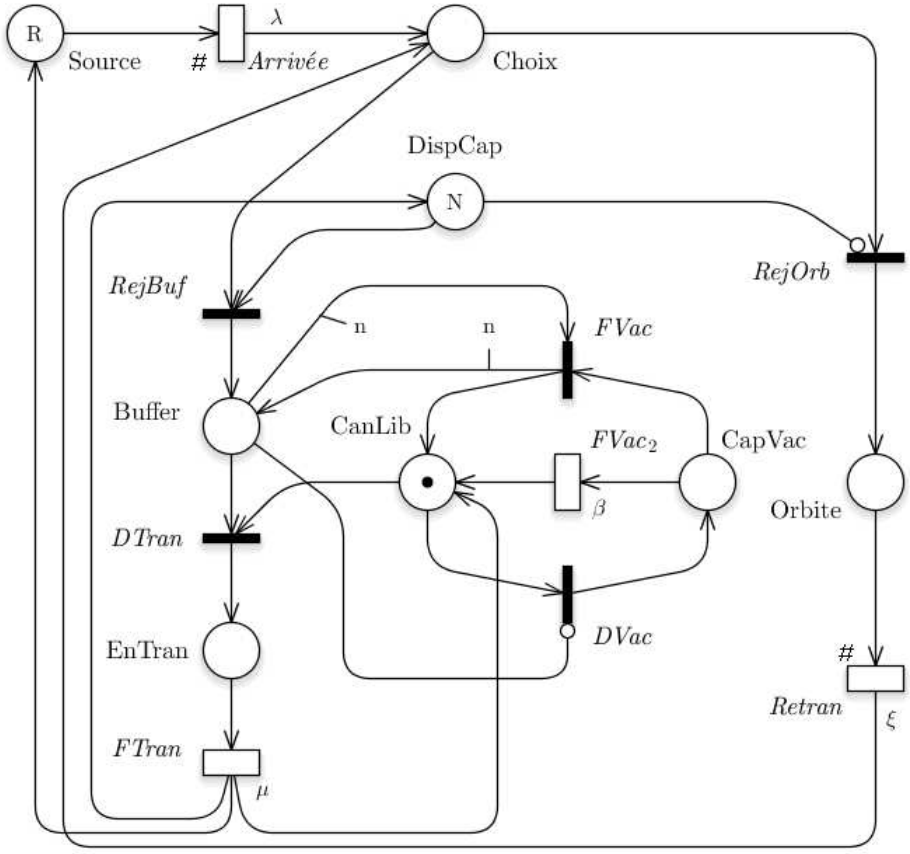

FIGURE 3.4 – Modèle de RdPSG correspondant à la mise en veille

tirée et le jeton de la place $CanLib$ sera consommé et mis dans la place $CapVac$. Ceci indique que le nœud capteur est passé à l'état *en veille*. Une fois le nœud capteur est en veille, la transition temporisée $FVac_2$ est sensibilisée et le nœud capteur reste dans cet état jusqu'à ce qu'elle soit tirée. Si par ailleurs, le nombre de jetons dans la place $Buffer$ atteint la valeur n, c'est la transition $FVac$ qui sera tirée en premier. Après le tir de $FVac$ ou celui de $FVac_2$, la marque de place $CapVac$ sera remise à la place $CanLib$, ce qui indique que le nœud capteur est réveillé ou remis en service.

Les règles de sensibilisation et de franchissement des autres transitions de ce modèle restent inchangées par rapport au modèle présenté dans la figure 3.3.

3.5.2 Modèle de mise en off

Contrairement à la mise en veille du nœud capteur, nous proposons plutôt, dans cette section, la mise en off du nœud capteur. De cette manière, le nœud capteur sera incapable de communiquer avec le reste du réseau. En conséquence, durant une période de vacance, chaque message envoyé à ce nœud doit être retransmis, ce qui est modélisé par la transition $RejOrb_2$ du modèle de RdPSG illustré dans la figure 3.5. Cette transition représente le fait que si le capteur est mis en off, les nouveaux messages reçus rejoignent l'orbite.

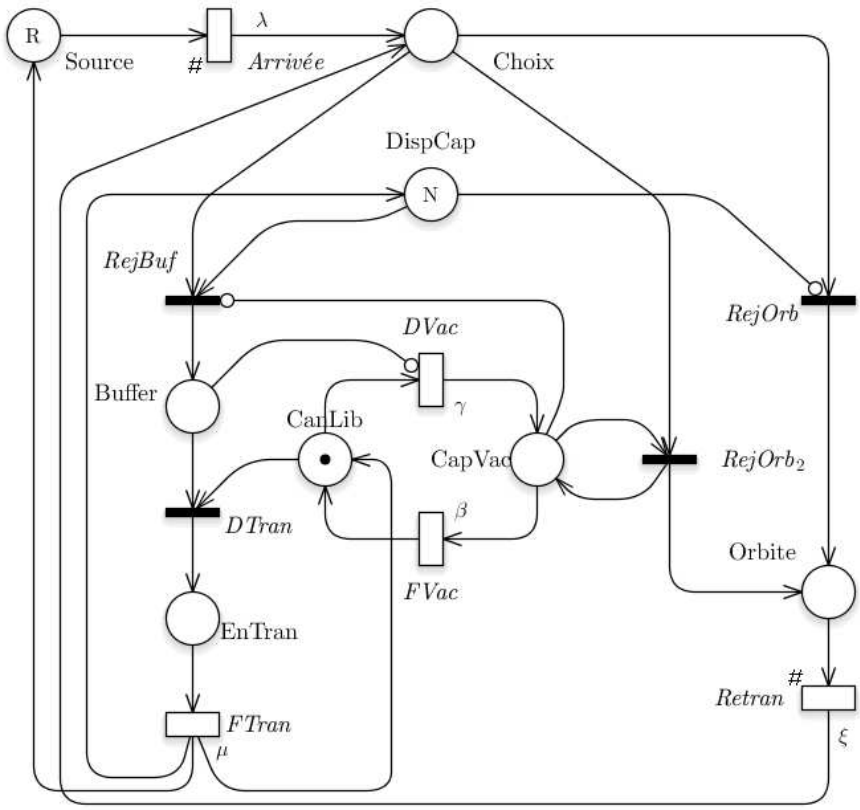

FIGURE 3.5 – Modèle de RdPSG correspondant à la mise en off

Contrairement au premier modèle, dans ce cas, une fois qu'il n'y a plus de messages à transmettre, le nœud capteur ne part pas directement en vacance (mise en off), mais plutôt, il passe une période de temps dite *période d'écoute*. À la fin de cette période, si aucun message ni reçu, le capteur sera mis en off. Dès le retour du nœud capteur d'une vacance, il passe une autre période d'écoute avant de prendre une autre vacance. Les durées des périodes d'écoute et de vacance sont des variables aléatoires distribuées exponentiellement avec les paramètres γ et β, respectivement, et qui sont mutuellement indépendantes. Nous notons que les valeurs de γ et β sont des valeurs décisives sur les performances du réseau, et un mauvais choix de ces valeurs pourrait entrainer des conséquences non souhaitables.

La figure 3.5 décrit une modélisation possible de la mise en off d'un nœud capteur par un RdPSG. Dans ce modèle, les transitions $DVac$ et $FVac$ sont des transitions temporisées sans mémoire et à sémantique de service à serveur unique.

À l'arrivée d'un message à la place $Choix$, si la place $CapVac$ contient une marque (le capteur est dans l'état *off*), le message rejoint l'orbite par le franchissement de la transition $RejOrb_2$. Sinon, il rejoint la place buffer, par le franchissement de la transition $RejBuf$, s'il y a encore des jetons dans la place $DispCap$. Dans le cas où le buffer est saturé, le message rejoint l'orbite par le franchissement de la transition $RejOrb$. Nous notons que les transitions $RejOrb_2$ et $RejBuf$ ne peuvent pas être franchissables à la fois. Ceci est garanti grâce à l'arc inhibiteur allant de la place $CapVac$ à la transition $RejBuf$.

Dès que la place $Buffer$ est vide, ce qui est modélisé par l'arc inhibiteur reliant la place $Buffer$ et la transition $DVac$, et la place $CanLib$ contient un jeton, ce qui exprime la disponibilité de l'unité de transmission, la transition $DVac$ devient franchissable et ne sera tirée qu'après une durée de temps moyenne égale à $1/\gamma$, qui correspond à la période d'écoute. Pendant cette période, si la place buffer reçoit un jeton, ce qui correspond à la réception d'un message, la transition $DVac$ devient non franchissable. Par contre, si la transition $DVac$ est tirée, le jeton de la place $CanLib$ sera consommé et déposé dans la place $CapVac$, ce qui indique que le capteur est passé à l'état *off*. La transition $FVac$ devient franchissable et ne sera tirée qu'après l'écoulement d'une durée de temps moyenne égale à $1/\beta$, qui correspond à la durée de mise en off. Après cette période, la marque de la place $CapVac$ sera remise à la place $CanLib$, ce qui indique que le nœud capteur est allumé.

Les sémantiques des autres transitions de ce modèle restent inchangées par rapport au premier modèle.

3.5.3 Mesure de gain en énergie

La quantité de gain en matière d'énergie dans les deux modèles représentés dans les figures 3.4 et 3.5, est en fonction du temps passé dans la place $CapVac$. Plus le temps passé dans cet état augmente plus de gain en énergie est important. Soient CE_o la consommation d'énergie dans l'état ordinaire du nœud capteur, et CE_v la consommation d'énergie dans l'état de vacance (mise en veille ou mise en off). La relation suivante est vérifiée :

$$CE_t = CE_o \,.\, P_o \,+\, CE_v \,.\, P_v \tag{3.6}$$

CE_t est la consommation totale d'énergie. P_o et P_v sont les probabilités que le nœud capteur soit dans l'état ordinaire et dans l'état de vacance, respectivement. Il est clair que la somme des probabilités P_o et P_v est égale à 1, car le capteur ne peut être dans un état autre que l'état ordinaire ou l'état de vacance. P_o et P_v sont calculées comme suit :

$$\begin{cases} P_v &= \displaystyle\sum_{i:M_i(CapVac)=1} \pi_i \\ P_o &= 1 - P_v \end{cases}$$

Ainsi, nous pouvons déterminer que le gain d'énergie est égal à la différence entre CE_o et CE_t, et peut être exprimé en pourcentage de la manière suivante :

$$gain = \frac{CE_o - CE_t}{CE_o} \times 100\% \tag{3.7}$$

En remplaçant CE_t dans l'équation (3.7), la formule du gain en énergie peut être réécrite comme suit :

$$gain = \frac{P_v.(CE_o - CE_v)}{CE_o} \times 100\% \tag{3.8}$$

3.6 Conclusion

L'objectif de ce chapitre était la présentation d'une approche de modélisation et d'évaluation des performances des réseaux de capteurs sans fil. Nous avons présenté dans un premier temps, une modélisation avec les files d'attente avec rappel et buffer et source finie de messages, ce qui a rendu le modèle plus complexe. Pour l'analyse du modèle à l'état stationnaire, nous avons utilisé le processus sous-jacent des chaînes de Markov, et nous avons développé une méthode numérique qui nous a permis de calculer les probabilités stationnaires, et ensuite les indices de performances les plus importants du réseau.

Afin de valider les résultats obtenus à l'aide des FAR, nous avons proposé une autre modélisation du même système avec les RdPSG. Nous avons dérivé les mêmes indices de performance que ceux obtenus à partir du modèle des FAR.

Grâce à la simplicité et à la grande puissance descriptive des RdPSG, nous avons pu introduire la notion de vacance dans notre modèle. Deux modèles de RdPSG, qui consistent à mettre les nœuds capteurs en veille et en off, respectivement, et qui considèrent à la fois le rappel des messages non acceptés, un buffer de capacité limitée, une source finie et les vacances du capteur, ont été proposés. Ces deux modèles visent essentiellement la réduction de la consommation d'énergie des nœuds capteurs. À partir de ces deux modèles, nous avons dérivé une formule explicite qui donne le pourcentage de gain en énergie par rapport au modèle de RdPSG sans vacance.

Dans le prochain chapitre, nous nous intéressons à la mise en ouvre de notre algorithme et de nos modèles et aux résultats expérimentaux qui montent l'impact des différents paramètres des modèles proposés sur les performances du réseau.

CHAPITRE 4

TESTS ET RÉSULTATS

4.1 Introduction

Nous nous intéressons dans ce chapitre à l'implémentation de notre algorithme de résolution d'un modèle basé sur les FAR avec buffer limité et source finie, ainsi qu'aux principaux résultats issus de nos expérimentations sur les trois modèles proposés dans ce mémoire. Notre étude va porter essentiellement sur l'influence du taux de réception des messages, du phénomène de rappel (retransmission) et les deux politiques de vacance proposées, sur les mesures de performance d'un réseau de capteurs sans fil.

En effet, les différentes mesures de performance obtenues concernent les nœuds capteurs un par un et non les performances du réseau dans sa totalité. Les indices de performance étudiés sont : la probabilité de débordement du buffer d'un capteur, le nombre moyen de messages en attente dans le buffer, le nombre moyen de messages en attente de retransmission, la latence moyenne des messages et le gain en énergie.

Deux études expérimentales différentes ont été effectuées. Dans la première étude, les expérimentations ont porté sur le modèle de FAR correspondant à un capteur du réseau. Quant à la deuxième étude, les expérimentations ont porté sur les deux modèles de vacance proposés (mise en veille et mise en off des nœuds capteurs).

Dans le cadre de ces expérimentations, les résultats numériques ont été obtenus en utilisant d'une part, une application, que nous avons développée en langage de programmation Java, implémentant notre algorithme décrit dans le chapitre précédent, et d'autre part l'outil software GreatSPN version 2.0.2, qui est un outil très performant principalement dédié à l'évaluation des performances.

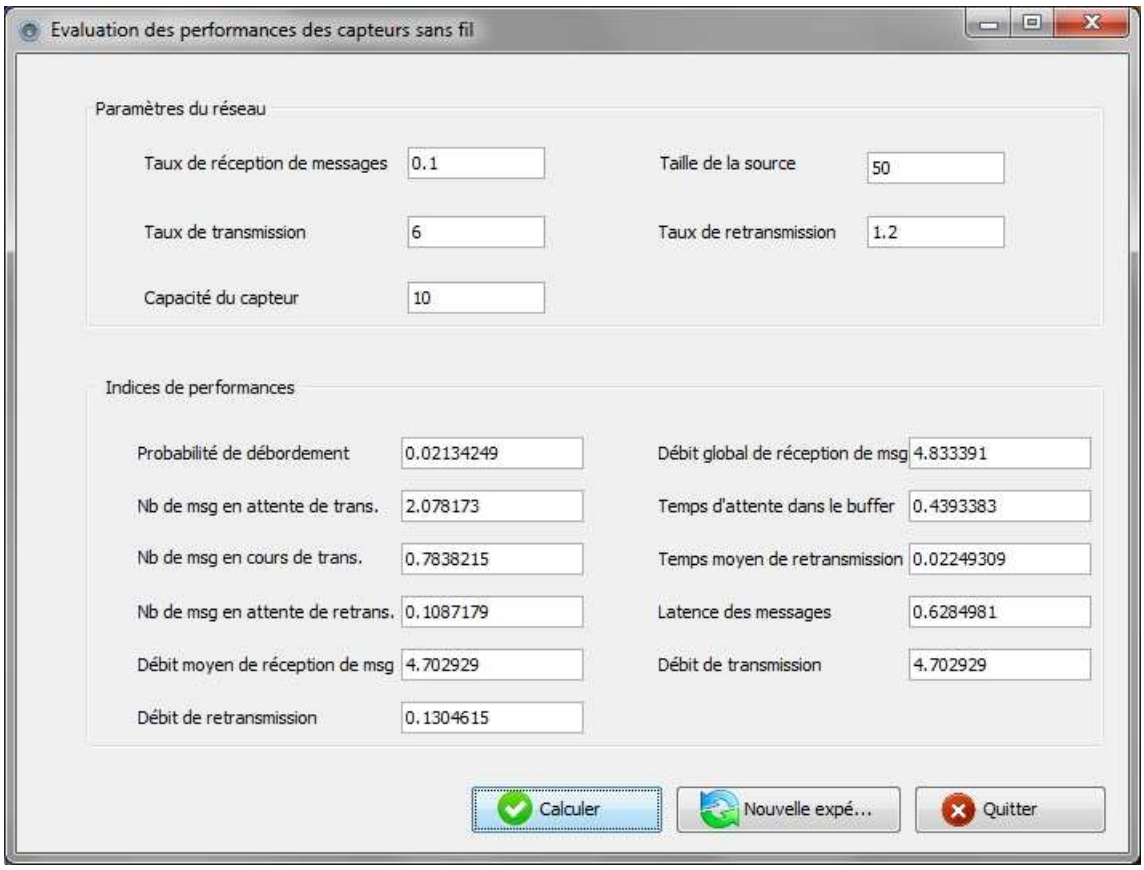

FIGURE 4.1 – Application pour l'évaluation des performances des WSN

4.2 Description de notre application

L'application que nous avons développée est une implémentation de l'algorithme récursif proposé dans le chapitre précédent. Elle a été développée sous Windows avec le langage de programmation Java. L'utilisateur entre les paramètres du réseau qui sont : le taux de réception de messages, le taux de transmission, la capacité du capteur, la taille de la source des messages et le taux de retransmission, et l'application lui fournit les principaux indices de performance. L'interface graphique de cette application est présentée dans la figure 4.1.

4.3 Description du GreatSPN

GreatSPN (*GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets*) [31], est un outil software qui permet la spécification, la validation et l'évaluation des performances des systèmes parallèles, en utilisant le formalisme des RdP stochastiques généralisés et leurs extensions colorés. Il comprend des algorithmes d'analyse efficaces, qui permettent son utilisation dans des applications complexes.

Développé à l'université de Turin en Italie, par un groupe de chercheurs qui travaillaient sur les RdPSG et l'évaluation des performances, il est distribué gratuitement à

d'autres universités pour des buts pédagogiques et des expérimentations dans le cadre de la recherche. La première version de cet outil software a été développée en 1984 suivie de plusieurs autres versions plus améliorées, dans lesquelles de nouveaux algorithmes ont été ajoutés, ainsi la correction des bugs des versions précédentes. Actuellement la version la plus récente disponibles est la version 2.0.2. [12]

Le package GreatSPN est composé de plusieurs programmes séparés qui coopèrent pour la construction et l'analyse des modèles de RdP en partageant des fichiers. D'autre part, en utilisant les systèmes de réseaux, différents modules d'analyse peuvent s'exécuter sur différentes machines dans un environnement de calcul distribué. Tous les modules sont écrits en langage de programmation C, pour garantir la portabilité et l'efficacité sur différentes machines. La version 2.0.2 est disponible pour les systèmes Solaris, SunOS et Linux.

Par ailleurs, l'outil GreatSPN permet l'analyse de systèmes complexes et volumineux ayant un espace d'états important. Ceci est possible du fait que tous les modules de résolution utilisent des techniques de stockage particulières pour le gain de mémoire. À titre d'exemple, la limite pratique de la version 1.7 sur des machines SUN SPARC avec une mémoire de 12-16 Mo s'élève à plusieurs centaines de milliers de marquages [31].

Les principales fonctions et techniques d'analyse implémentées dans l'outil GreatSPN sont :

- Une interface graphique simple à manipuler qui permet à l'utilisateur la saisie du modèle correspondant au système à analyser. Ce choix s'explique par la nature graphique des représentations du formalisme RdP ;
- Vérification des propriétés structurelles ;
- Génération et analyse du graphe d'accessibilité ;
- Résolution markovienne qui permet l'évaluation des performances stationnaires aussi bien que transitoire, en exploitant des techniques numériques matricielles efficaces ;
- La simulation : dans ce cas, plusieurs distributions de probabilité sont admises pour les transitions temporisées, telles que : Erlang, Cox, loi uniforme, discrète, etc.

Par ailleurs, plusieurs autre outils software ont été développés pour l'analyse des modèles de RdPSG. Nous citons essentiellement : SPNP (*Stochastic Petri Net Package*) [115], TOMSPIN (*TOol for Modelling with Stachastic PetrI Nets*) [116], TimeNET, UltraSan [117], DSPNExpress [118], etc. Cependant, les points faibles de certaines de ces outils étaient la faible portabilité, le manque de flexibilité des programmes et surtout l'absence d'une interface graphique qui est le mode le plus naturel pour la description des modèles basés sur le formalisme des RdP. Ainsi, des efforts ont été fournis pour la conception de l'outil GreatSPN, qui est caractérisé par la facilité d'utilisation, la portabilité, la modu-

larité et l'efficacité des modules d'analyse. [12]

En effet, comparé à d'autres outils, la particularité de GreatSPN est le fait qu'il offre un environnement de modélisation et d'analyse plus complet, dans lequel un variété de techniques et d'algorithmes d'analyse sont disponibles. En effet, les autres outils offrent de bonnes implémentations de certains de ces algorithmes, mais aucun d'entre eux ne les offre tous ensemble dans un environnement uniforme tel que GreatSPN [31].

Une fois que le modèle est complètement spécifié, nous pouvons commencer son analyse. L'analyse structurelle est habituellement accomplie en premier lieu, car elle permet de vérifier des propriétés intéressantes du système avant de construire son espace d'état. GreatSPN nous permet vérifier que le modèle est borné. Si c'est le cas, le modèle a un espace d'états fini. D'autre part, GreatSPN nous permet aussi de vérifier la vivacité du modèle. Ainsi, après la vérification que le modèle est fini, le graphe d'accessibilité tangible sera généré en se basant sur les résultats théoriques présentés dans [119, 120]. Si l'examen de ce graphe fini permet de déduire que le marquage initial est un état d'accueil, alors on est sûr que la chaîne de Markov sous-jacente est ergodique. Ainsi, elle sera dérivée et résolue pour permettre l'analyse quantitative. [12]

Le graphe des marquages accessibles peut être généré sans aucune information temporelle. Cependant, si nous nous intéressons au calcul des indices de performance, nous avons besoin de définir les poids des transitions immédiates, les taux des transitions temporisées, ainsi que leurs sémantique d'exécution : serveur-unique, multiple ou infini.

Après la construction du graphe des marquages accessibles, le calcul de la distribution de probabilité de marquage à l'état stationnaire ou transitoire devient possible. L'outil permet de calculer automatiquement et implicitement les fréquences de franchissement des transitions. De plus il permet le calcul d'autres indices de performance qui peuvent être explicitement définis par l'utilisateur, tel que : le nombre moyen de marques dans une place donnée, la probabilité d'un certain évènement. Une grammaire formelle est utilisée pour la définition des résultats désirés. les résultats obtenus peuvent être visualisés sur l'interface graphique du logiciel, comme on peut les récupérer à partir du fichier *file_ name.sta*.

4.4 Études expérimentales

4.4.1 Première étude

Le but de cette étude expérimentale est d'observer l'impact du taux de réception et du taux de retransmission des messages sur les mesures de performances des capteurs sans fil. Les résultats de cette étude sont obtenus à l'aide de l'application développée en Java.

Dans un premier temps, nous présentons dans la table 4.1 une comparaison entre les résultats concernant certains descripteurs de performance, obtenus avec notre application et l'outil GreatSPN, respectivement. À partir du cette table, nous voyons clairement que les valeurs des indices de performances sont presque identiques pour les deux cas, ce qui prouve la validité de l'approche d'analyse du modèle de FAR et notre algorithme, présentés dans le chapitre précédent.

TABLE 4.1 – Validations

	Algorithme (FAR)	GreatSPN (RdPSG)
Taille de la source des messages	50	50
Taille du capteur	10	10
Taux de réception des messages	0.1	0.1
Taux de transmission des messages	6	6
Taux de retransmission des messages	1.2	1.2
Probabilité de débordement du buffer	0.021342	0.021342
Nombre moyen de messages en attente de transmission	2.078173	2.078173
Nombre moyen de messages en cours de transmission	0.783821	0.783821
Nombre moyen de messages en attente de retransmission	0.108717	0.108718
Débit moyen de réception des nouveaux messages	4.702929	4.702929
Débit moyen de réception des messages retransmis	0.130461	0.130462
Temps moyen d'attente des messages dans le buffer	0.439338	$0,439338$
Débit moyen de transmission	4.702929	4.702929

Nous présentons dans ce qui suit, les résultats numériques de l'étude des performances des nœuds capteurs sans fil, modélisés par des FAR avec des buffers de tailles limitées et source de messages finie. Nous illustrons l'effet des taux de réception et de retransmission des messages sur les principaux indices de performance.

Les paramètres du réseau utilisés dans cette étude, ainsi que dans la deuxième étude sont résumés dans la table 4.2.

TABLE 4.2 – Paramètres du réseau

	R	N	λ	μ	ξ	β	n	γ
Figure 4.2	100	$10-30$	Axe abs.	30	2.0	-	-	-
Figure 4.3	100	$10-30$	Axe abs.	30	2.0	-	-	-
Figure 4.4	100	$10-30$	Axe abs.	30	2.0	-	-	-
Figure 4.5	100	$10-30$	Axe abs.	30	2.0	-	-	-
Figure 4.6	100	$10-30$	0.4	30	Axe abs.	-	-	-
Figure 4.7	100	$10-30$	0.4	30	Axe abs.	-	-	-
Figure 4.8	100	$10-30$	0.4	30	Axe abs.	-	-	-
Figure 4.9	100	$10-30$	0.4	30	Axe abs.	-	-	-
Table 4.3	100	20	0.2	30	0.5	Axe abs.	$1-25$	-
Table 4.4	100	20	0.2	30	0.5	Axe abs.	-	$0.05-0.5$
Table 4.5	100	20	0.2	30	0.5	Axe abs.	$1-25$	-
Table 4.6	100	20	0.2	30	0.5	Axe abs.	-	$0.05-0.5$
Table 4.7	100	20	0.2	30	0.5	Axe abs.	$1-25$	$0.05-0.5$
Table 4.8	100	20	Axe abs.	30	0.5	0.01	10	0.5

Nous donnons dans les figures 4.2-4.5 les graphes montrant l'influence du taux de réception des messages sur la probabilité de débordement du buffer, le nombre moyen de messages en attente de transmission et en attente de retransmission, et la latence moyenne des messages, avec la capacité du capteur comme paramètre.

- Les courbes de la figure 4.2 montrent l'effet du taux de réception des messages sur la probabilité de débordement du buffer. On voit bien que cette probabilité croît avec l'augmentation du flux des messages reçus, ce qui est intuitivement logique. De plus, on remarque qu'elle est relativement inférieur quand la taille du buffer du capteur est plus importante.

- À partir de la figure 4.3, on voit clairement que le taux de réception des messages influe considérablement sur la taille du buffer (nombre de messages en attente de transmission) ; quand le taux de réception des messages croît, la taille du buffer croît elle aussi.

- Les graphes de la figure 4.4 mettent en évidence l'influence du taux de réception des messages sur le nombre moyen de messages en attente de retransmission. Ce dernier croit significativement avec la croissance du taux de réception des messages. On remarque qui il devient de plus en plus inférieur pour les grandes valeurs de N par rapport aux petites valeurs de N. Ceci est dû au fait qu'à chaque fois qu'on augmente la capacité du stockage du capteur, les messages auront plus de chance à être admis dans le buffer sans être retransmis.

- Dans la figure 4.5, nous illustrons l'impact du taux de réception des messages sur la latence moyenne des messages. La latence croît rapidement, ensuite, à partir

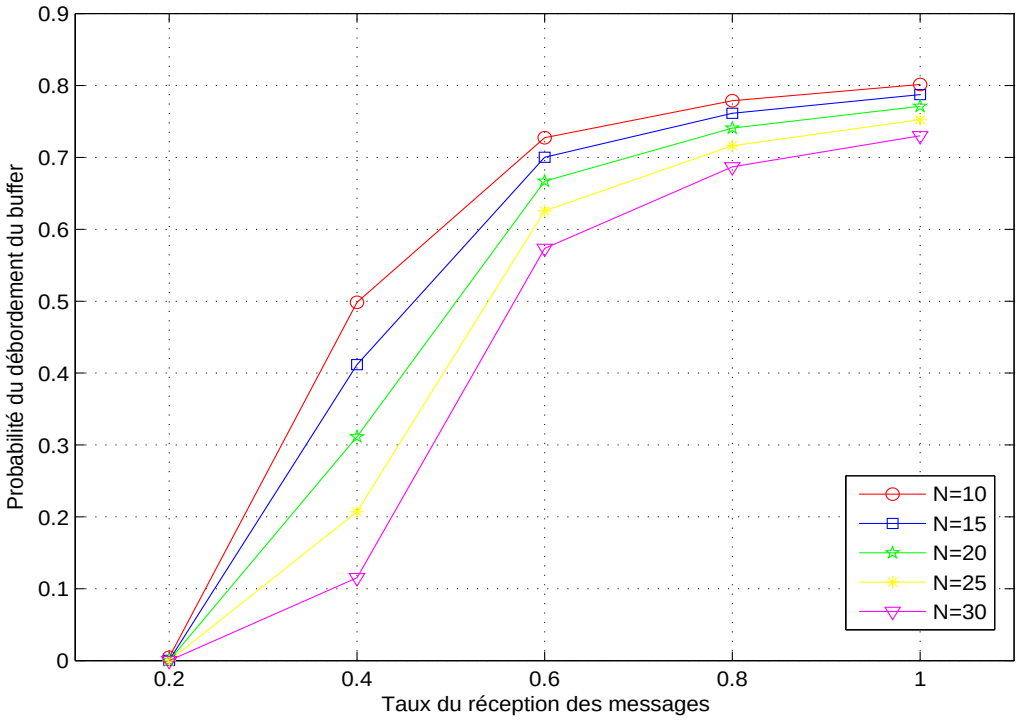

FIGURE 4.2 – Probabilité de débordement du buffer en fonction du taux de réception des messages

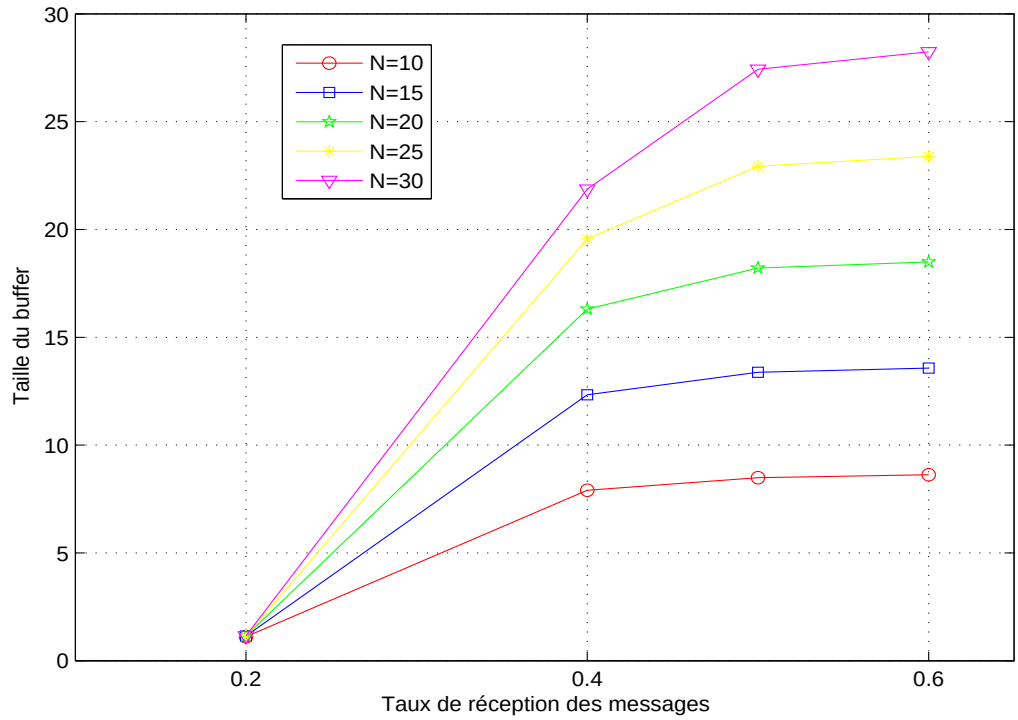

FIGURE 4.3 – Taille du buffer en fonction du taux de réception de messages

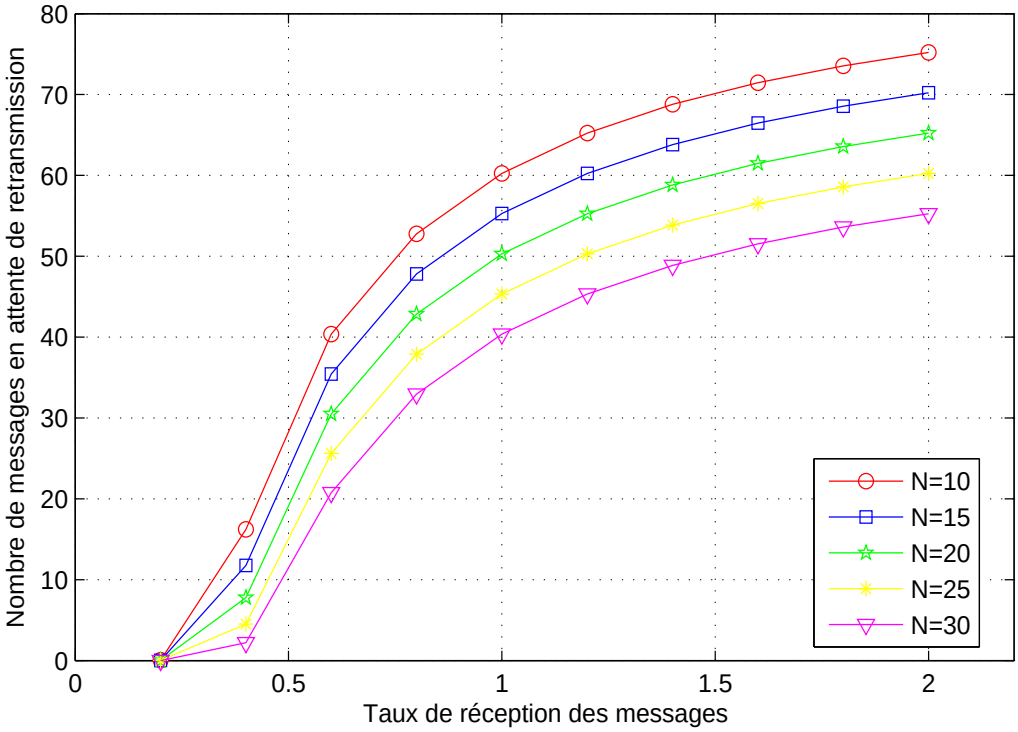

FIGURE 4.4 – Nombre de messages en attente de retransmission en fonction du taux de réception de messages

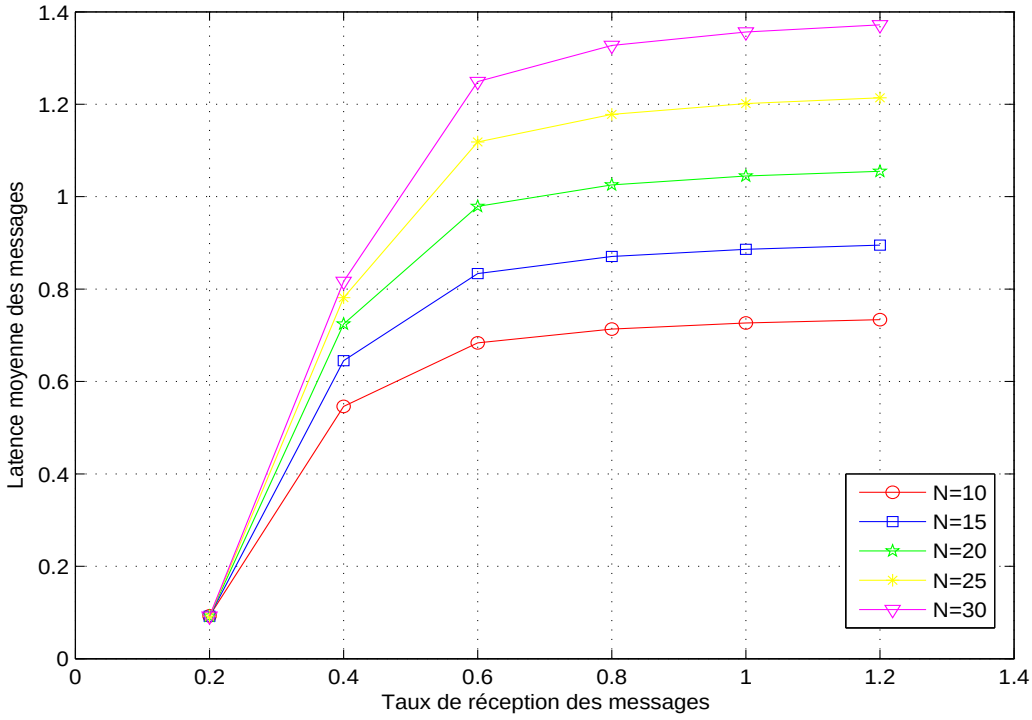

FIGURE 4.5 – Latence moyenne en fonction du taux de réception de messages

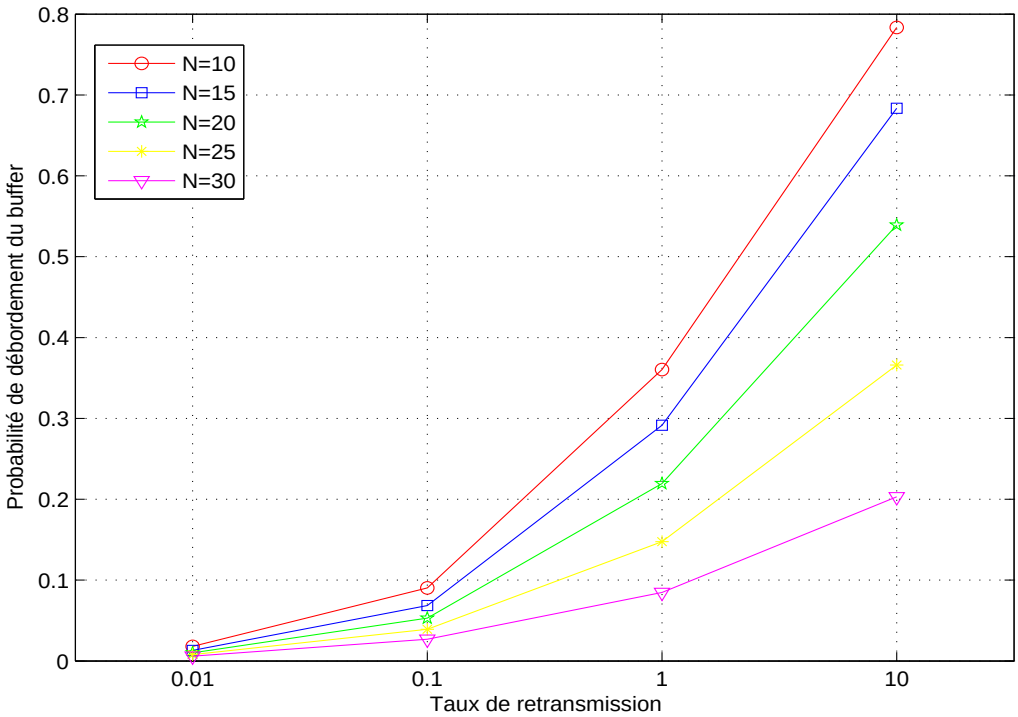

FIGURE 4.6 – Probabilité de débordement en fonction du taux de retransmission

d'une valeur proche de 0.6 du taux de réception des messages, la croissance se fait lentement. Nous avons pu constater aussi que la latence augmente avec la capacité du buffer.

Dans les figures 4.6-4.9, nous présentons l'effet du taux de retransmission des messages sur les mêmes métriques de performance étudiés ci-dessus. Pour chaque figure, nous utilisons des courbes correspondant à des capacités différentes du capteur.

- Dans la figure 4.6, la probabilité de débordement du buffer du capteur est représentée en fonction du taux de retransmission. À partir de cette figure, on constate que la probabilité de débordement croît en fonction du taux de retransmission. De plus, elle devient plus petite quand la capacité de stockage du capteur augmente.

- La figure 4.7 montre le nombre du messages en attente de transmission en fonction du taux de retransmission des messages. Les courbes de cette figure indiquent que le nombre de messages en attente de retransmission croit en fonction du taux de retransmission. Cette croissance se fait doucement quand les messages retransmis s'intensifient.

- La figure 4.8 montre l'impact du taux de retransmission sur le nombre moyen de messages en attente de retransmission. Ce dernier décroit d'une manière significative pour les taux de retransmission bas. Par contre, cette décroissance devient douce quand le taux de retransmission devient important. Là aussi, nous voyons bien

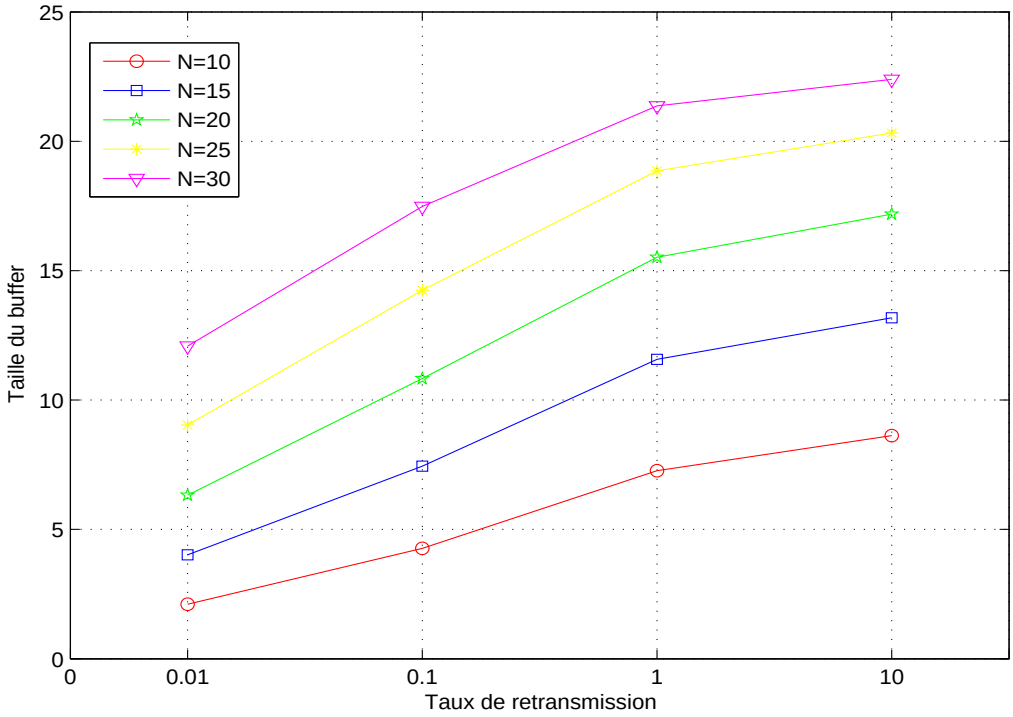

FIGURE 4.7 – Taille du buffer en fonction du taux de retransmission

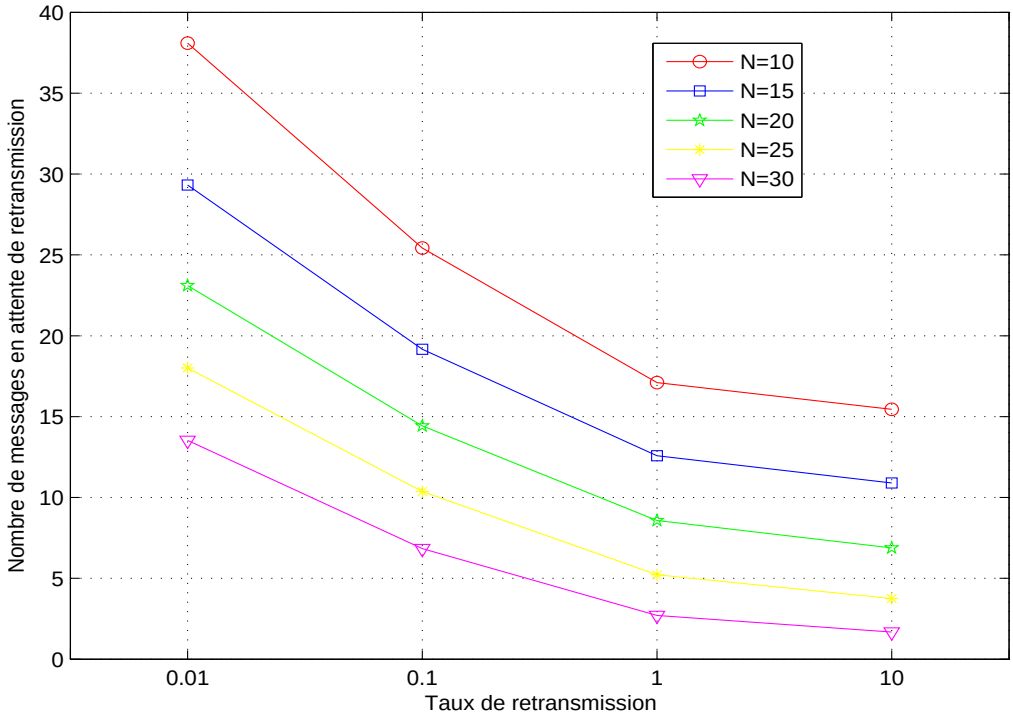

FIGURE 4.8 – Nombre de messages en attente de retransmission en fonction du taux de retransmission

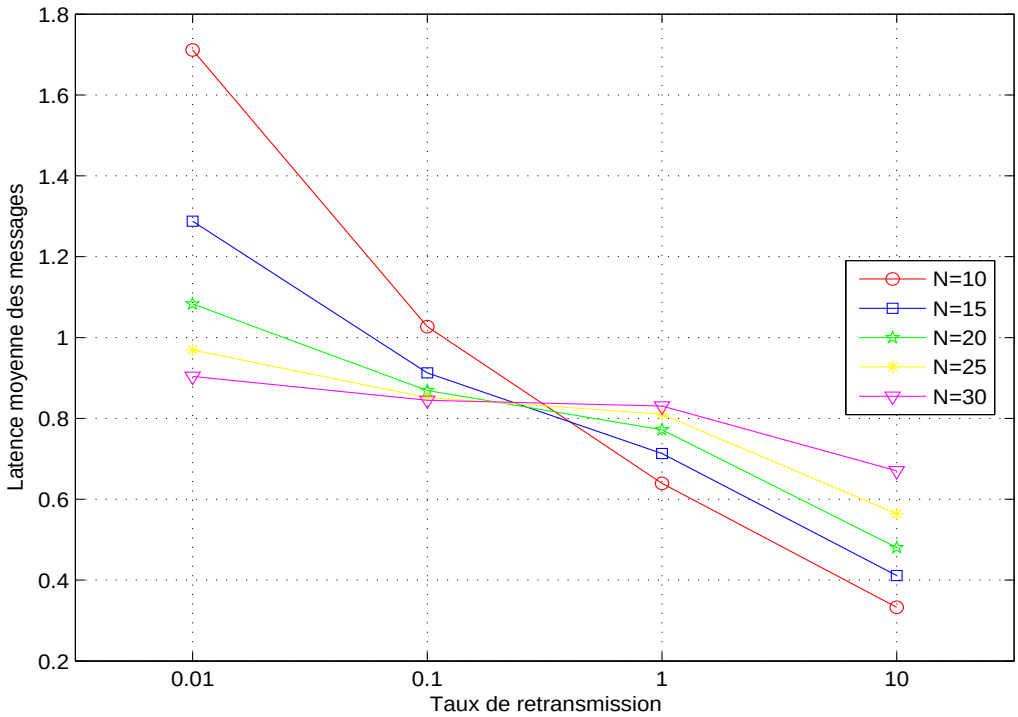

FIGURE 4.9 – Latence moyenne des messages en fonction du taux de retransmission

qu'une incrémentation de la capacité du capteur permet de réduire le nombre moyen de messages en attente de retransmission.

- La figure 4.9 montre l'effet du taux de retransmission sur la latence moyenne des messages. À partir de cette figure, nous constatons que la latence de messages décroit avec la croissance du taux de retransmission. De plus, nous voyons clairement qu'elle décroit aussi en fonction de la capacité du capteur pour des taux de retransmission bas. Par contre, elle devient meilleure avec des capacités de capteur réduites, pour des taux de retransmission élevés.

- Finalement, nous pouvons conclure que le taux de réception des messages, le taux de retransmission des messages et la taille du buffer du capteur, sont des facteurs essentielles qui affectent les performances des capteurs sans fil, et par conséquent, les performances du réseau complet.

4.4.2 Deuxième étude

Dans cette étude, nous avons réalisé des expérimentations sur les deux modèles avec vacance (mise en veille et mise en off des capteurs) que nous avons proposés. Nous avons étudié l'impact des deux politiques de vacance sur le gain en énergie des capteurs, ainsi que la latence moyenne des messages.

Le schéma de consommation d'énergie utilisé dans cette étude est relatif aux capteurs *Mica Mote Sensor*. Les niveaux de consommation d'énergie pour ce capteur sont : $81mW$ pour la transmission, $30mW$ pour la réception et l'*idle listening*, et $0.003mW$ quand le capteur est mis en off [21].

Les tables 4.3 et 4.4 montrent les mesures du gain en énergie en fonction du temps moyen autorisé d'une période de vacance [1], pour les modèles de mise en veille et de mise en off des capteurs. Les tables 4.5 et 4.6 donnent les résultats relatifs à la latence moyenne des messages, pour les modèles avec mise en veille et mise en off des capteurs, respectivement. Tandis que la table 4.7 nous montre les rapports *gain énergie/latence* relatives aux deux schémas de vacance. Enfin, nous présentons dans la table 4.8 l'effet du taux de réception des messages sur le gain en énergie.

TABLE 4.3 – Effet du temps moyen de vacance sur le gain en énergie (mise en veille)

	Gain en énergie (%)				
β	$n = 1$	$n = 5$	$n = 10$	$n = 19$	$n = 25$
0.01	**21.745518**	22.540237	23.529951	25.450700	**60.585418**
0.05	**21.745518**	22.538600	23.522900	25.420918	**53.633426**
0.1	**21.745518**	22.536585	23.514211	25.383959	**48.163140**
1.0	**21.745518**	22.500633	23.362344	24.762263	**28.802848**
5.0	**21.745518**	22.362240	22.827348	23.137063	**23.215955**
10	**21.745518**	22.232914	22.444407	22.494211	**22.496037**

TABLE 4.4 – Effet du temps moyen de vacance sur le gain en énergie (mise en off)

	Gain en énergie (%)		
β	$\gamma = 0.05$	$\gamma = 0.1$	$\gamma = 0.5$
0.01	61.53692	74.90042	90.83063
0.05	24.43889	37.84379	67.87428
0.1	14.03718	23.66472	52.88584
1.0	**1.671938**	**3.241379**	**13.06121**
5.0	**0.342887**	**0.681074**	**3.225780**
10	**0.172093**	**0.342887**	**1.667038**

- Dans la table 4.3, nous avons utilisé différentes valeurs du seuil [2] $n = 1, 5, 10, 19$, et 25. Pour la valeur $n = 1$, nous remarquons que la mesure du gain en énergie reste inchangée et égale à 21.745518 %. Ceci peut être justifié par le fait que le nœud capteur se remet toujours en état opérationnel normal (revient d'une vacance), quand

1. Le temps moyen autorisé d'une période vacance est égal à $1/\beta$.
2. Le nombre de messages dans le buffer n'atteint jamais la valeur 25, car la capacité du buffer est limitée à 20. Par conséquent, le capteur ne revient d'une vacance qu'après l'écoulement du temps moyen autorisé d'une période de vacance.

le nombre de messages stockés atteint le seuil ($n = 1$), et non après l'écoulement du temps moyen autorisé d'une période vacance. Pour $n = 5, 10$, et 19, le gain en énergie augmente légèrement. Par contre, quand $n = 25$, ce qui signifie que le capteur ne revient d'une vacance qu'après l'écoulement du temps moyen autorisé d'une période de vacance, le gain en énergie devient considérable, et atteint la valeur 60.585418 % pour $\beta = 0.01$. De plus, il décroît quand le temps moyen autorisé d'une période de vacance décroît (β croît).

- La table 4.4 montre l'effet du temps moyen d'une période de vacance [3] sur le gain en énergie. Nous avons fait des expérimentations avec différentes durées moyennes de période d'écoute [4], pour $\gamma = 0.05, 0.1$ et 0.5. Nous remarquons que, plus la période d'écoute est courte, plus le gain en énergie est important, car quand la période d'écoute est longue, le capteur n'aura pas trop de chances pour prendre une vacance. De plus, comme pour la mise en veille des capteurs, le gain en énergie décroît quand le temps moyen d'une période de vacance décroît.

- En comparant les résultats illustrés dans la table 4.3 par ceux illustrés dans la table 4.4, nous pouvons constater que la mise en off des capteurs nous offre un meilleur gain en énergie pour des longue périodes de vacance. Par contre, pour des courtes périodes de vacance, la mise en veille des capteurs est bien meilleure.

TABLE 4.5 – Effet du temps moyen de vacance sur la latence (mise en veille)

	Latence			Latence sans vac.
β	$n = 1$	$n = 10$	$n = 25$	
0.01	**0.091983**	0.322421	**12.93151**	0.091972
0.05	**0.091983**	0.321477	**4.671178**	
0.1	**0.091983**	0.320301	**3.438885**	
1.0	**0.091983**	0.299895	1.037407	
10	**0.091983**	0.179817	0.186410	

3. Dans le modèle de mise en off des capteurs, le temps moyen d'une période de vacance est lui-même temps moyen autorisé d'une période de vacance, et est égale à $1/\beta$.

4. La durée moyenne d'une période d'écoute est égale à $1/\gamma$.

TABLE 4.6 – Effet du temps moyen de vacance sur la latence (mise en off)

	Latence			Latence sans vac.
β	$\gamma = 0.05$	$\gamma = 0.1$	$\gamma = 0.5$	
0.01	1.635314	1.793260	1.947266	0.091972
0.05	0.905975	1.201846	1.665237	
0.1	0.580335	0.840959	1.410382	
1.0	0.134899	0.174567	0.410855	
5.0	0.099458	0.106825	0.162120	
10	0.095631	0.099255	0.127335	

TABLE 4.7 – Rapport *gain énergie/latence*

	Mise en veille			Mise en off		
beta	$n = 1$	$n = 10$	$n = 25$	$\gamma = 0.05$	$\gamma = 0.1$	$\gamma = 0.5$
0.01	236.408	72.9789	4.68510	37.6300	41.7677	46.6452
0.05	236.408	73.1713	11.4817	26.9752	31.4880	40.7595
0.1	236.408	73.4128	14.0054	24.1880	28.1401	37.4975
1.0	236.408	77.9017	27.7642	12.3939	18.5681	31.7903
5.0	236.408	99.5757	82.9070	3.44755	6.37560	19.8974
10	236.408	124.818	120.680	1.79955	3.45460	13.0917

TABLE 4.8 – Impact du taux de réception des messages sur le gain en énergie (%)

	$\lambda = 0.10$	$\lambda = 0.20$	$\lambda = 0.30$	$\lambda = 0.30$	$\lambda = 0.40$
Mise en veille	43.02422	23.52995	6.635288	1.955314	0.398492
Mise en off	94.26540	90.83063	74.96532	42.91621	10.32221

- En comparant les valeurs des tables 4.5 et 4.6 avec celles des tables 4.3 et 4.5, respectivement, nous pouvons constater que plus le gain en énergie est important, la latence moyenne des messages augmente. Nous remarquons qu'elle est grande d'une manière significative dans le cas de la mise en veille des capteurs avec la valeur $n = 25$. D'autre part, elle est toujours supérieure à la latence moyenne des messages dans le cas du modèle sans vacance des capteurs, qui est égale à 0.091972.

- À partir du tableau 4.7, nous pouvons constater que la mise en veille des capteurs avec un seuil $n = 1$, est la meilleure solution pour gagner l'énergie tout en ne pas augmentant la latence des messages. Ce modèle permet d'exploitation optimale du temps où les capteurs ne faites aucune tâche.

- Dans la table 4.8, nous illustrons l'effet du taux de réception des messages sur le gain en énergie. D'après cette table, nous voyons que le taux de réception influe

considérablement sur le gain en énergie pour les deux variantes de vacance. Quand le taux de réception de messages est bas, les chances que le capteur prend une vacance sont importantes, et vice-versa. Ceci est dû politique de service exhaustive utilisée par les deux variante.

- En fin, nous pouvons conclure et insister que les durées moyenne des périodes d'écoute et de vacance, ainsi que le seuil (n) sont des paramètres décisifs sur les performances et le gain en énergie des capteurs sans fil.

4.5 Conclusion

Au cours de ce chapitre, nous avons réalisé des calculs numériques en utilisant notre application et l'outil GreatSPN 2.0.2, pour montrer l'effet des paramètres du réseau et les disciplines de vacance des capteurs sur les performance du réseau.

Nous avons pu montrer dans un premier temps, grâce aux résultats obtenus par notre application et l'outil GreatSPN, la validité de la méthode d'analyse et de résolution proposée, qui est basé sur à un modèle de file d'attente avec rappel, buffer de capacité limitée, et source finie de messages. Dans un second lieu, la série d'expérimentations réalisées sur ce modèle appliqué aux nœuds capteurs sans fil, ont montré l'importance d'un bon choix des paramètres du réseau. Cette importance revient à leur grande influence sur les performances des capteurs, et par conséquent, sur les performances du réseau dans sa totalité.

Nous avons pu montrer aussi l'importance et le grand impact de la manière d'exploiter les temps libres des capteurs pour gagner de l'énergie. Les résultats ont montré que dans certains cas, la mise en off des capteurs sans fil nous offre un meilleur gain en énergie, mais avec une grande augmentation de latence des messages, ce qui n'est pas souhaitable. En prenant en compte le rapport *gain énergie/latence des message*, nous avons trouvé que la mise en veille des capteur avec un seuil $n = 1$, est la meilleure façon d'exploiter les temps libres des nœuds capteurs dans le cas des paramètres considérés.

CONCLUSION GÉNÉRALE

Les réseaux de capteurs sans fil sont devenus de plus en plus importants et continuent à gagner leur place dans des domaines variés. Cependant, ces réseaux ont leurs spécificités résultant de leur nature. Ils ont des ressources matérielles très restreintes, comme la capacité de communication, l'espace de stockage, la vitesse de calcul et la source d'énergie. Ainsi, leur analyse et évaluation est indispensable. Mis à part les outils de simulation qui sont la méthode traditionnelle utilisée dans les réseaux de capteur, une excellente alternative pour l'analyse et l'évaluation des performances de ces réseaux, est l'utilisation des méthodes formelles ayant l'avantage de fournir des résultats exacts. Cependant, l'obtention de ces résultats n'est pas toujours facile, ceci dépend de la complexité du modèle.

Nous avons vu que parmi les méthodes formelles existantes, il y a les files d'attente avec rappel, qui permettent l'introduction du phénomène de rappel dans un système de file d'attente. En se basant sur ce formalisme, nous avons proposé une modélisation des réseaux de capteurs sans fil. C'est un modèle qui concerne les nœuds un par un et pas la totalité du réseau. Nous avons considéré une source de messages finie, un buffer limité et la retransmission des messages à cause du débordement des buffers. Un tel modèle n'a pas de solution numérique ou analytique. Pour cela, nous avons développé d'une part, un algorithme récursif qui nous permet d'obtenir la distribution stationnaire, et nous avons d'autre part, développé les formules des principaux indices de performance.

Par ailleurs, les réseaux de Petri stochastiques généralisés sont très efficaces pour la modélisation des systèmes parallèles caractérisés par la concurrence, l'exclusion mutuelle, la synchronisation et le conflit, etc. Ce formalisme graphique et mathématique a une forte capacité de description, de modélisation et d'évaluation des performances des systèmes complexes. En nous basant sur ce formalisme, nous avons pu valider le modèle développé avec une FAR, ainsi que les formules des indices de performance, tout en utilisant l'outil GreatSPN.

D'un autre coté, l'utilisation des réseaux de Petri stochastiques généralisés nous a per-

mis d'introduire la notion de vacance du serveur dans le modèle précédent, et ce dans le but de conserver le maximum d'énergie des nœuds sans trop affecter les performances du réseau. Nous avons introduit cette notion de vacance de deux manières différentes et nous avons obtenu deux modèles différents. Le premier consiste à mettre les nœuds en veille quand ils sont libres, et le deuxième consiste à les mettre en off. Dans les deux modèles, nous avons combiné, pour la première fois, le buffer limité, le rappel, la source finie et la vacance dans les réseaux de capteurs sans fil.

L'impact des taux de réception et de retransmission des messages et les modes de vacance sur les performances du réseau et la conservation d'énergie sont étudiés à l'aide d'une application implémentant l'algorithme que nous avons proposé et l'outil GreatSPN. D'après les résultats obtenus, nous avons pu constater que la mise en veille des nœuds capteurs est la meilleure solution pour gagner de l'énergie et maintenir de bonnes performances du réseau.

Enfin, comme perspectives à ce travail, il serait intéressant de considérer des temps de transmission de loi générale au lieu des temps exponentiels. Il serait aussi bien intéressant de considérer la retransmission des messages à cause des problèmes de réseau, tels que la congestion et les problèmes de support de communication. Une autre perspective serait de prendre en considération plusieurs types de messages avec des priorités différentes, en utilisant le formalisme des RdPSG colorés. Nous proposons aussi de considérer d'autres disciplines de vacance. Une autre perspective intéressante est de proposer une approche d'évaluation des performances de la totalité du réseau en utilisant un réseau de files d'attente avec rappel et buffer.

BIBLIOGRAPHIE

[1] Y. Caumel, *Probabilités et processus stochastiques*. Springer-Verlag France, 2011.

[2] E. Pardoux, *Processus de Markov et applications : Algorithmes, réseaux, génome et finance, cours et exercices corrigés*. Dunod, 2007.

[3] P. Brémaud, *Initiation aux Probabilités et aux chaînes de Markov*. Springer-Verlag Berlin Heidelberg, 2009.

[4] B. Baynat, *Théorie des files d'attente, des chaînes de Markov aux réseaux à forme produit*. Paris : Hermès Sciences Publications, 2000.

[5] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modeling with Generalized stochastic Petri Nets*. John Wiley & Sons, 1995.

[6] N. T. Thomopoulos, *Fundamentals of Queuing Systems : Statistical Methods for Analyzing Queuing Models*. New York : Springer Science+Business Media, 2012.

[7] G. I. Falin and J. G. C. Templeton, *Retrial Queues*. London : Chapman and Hall, 1997.

[8] M. Diaz, *Les réseaux de Petri : Modèles fondamentaux*. Paris : Hermès Science Puclications, 2001.

[9] G. W. Brams, *Réseaux de Petri : Théorie et Pratique, volume 1*. Paris : Masson, 1983.

[10] A. Choquet-Geniet, *Les réseaux de Petri : Un outil de modélisation*. Paris : Dunod, 2006.

[11] T. Murata, "Petri nets : Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[12] N. Gharbi, *Evaluation des performances et de la fiabilité des systèmes Multi-classes avec Rappel à l'aide des réseaux de Petri Stochastiques colorés*. PhD thesis, USTHB, Algiers, 2007.

[13] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks : a survey," *Computer Networks*, vol. 38, no. 8, pp. 393–422, 2002.

[14] [Online], "The Network Simulator -ns-2. http ://www.isi.edu/nsnam/ns/, date de consultation : Janvier 2014."

[15] A. Varga, "OMNeT++," in *Modeling and Tools for Network Simulation* (K. Wehrle, M. Güneş, and J. Gross, eds.), pp. 35–59, Springer Berlin Heidelberg, 2010.

[16] X. Chang, "Network Simulations with OPNET," in *Proceedings of the 1st Conference on Winter Simulation : Simulation—a Bridge to the Future - Volume 1*, WSC '99, (Phoenix, Arizona, USA), pp. 307–314, 1999.

[17] G. I. Falin, "A probabilistic model for investigation of load of subscriber's lines with waiting places." in : Probability Theory, Stochastic Processes and Functional Analysis, Moscow State University, Moscow, 1985.

[18] G. I. Falin, "Single-line repeated orders queueing systems," *Optimization*, vol. 17, no. 5, pp. 649–667, 1986.

[19] G. E. Ridout, *A study of retrial queueing systems with buffers, MA Sc.* PhD thesis, Department of Industrial Engineering, University of Toronto, 1984.

[20] G. I. Falin and J. R. Artalejo, "A finite source retrial queue," *European Journal of Operational Research*, vol. 108, no. 2, pp. 409–424, 1998.

[21] F. C. Jiang, D. C. Huang, and K. H. Wang, "Design Approaches for Optimizing Power Consumption of Sensor Node with N-policy M/G/1 Queuing Model," in *Proceedings of the 4th International Conference on Queueing Theory and Network Applications*, QTNA '09, (Fusionopolis, Singapore), pp. 31–38, 2009.

[22] F. C. Jiang, C. T. Yang, K. H. Wang, and D. C. Huang, "Design Framework to Optimize Power Consumption and Latency Delay for Sensor Nodes Using Min(N, T) Policy M/G/1 Queuing Models," in *5th International Conference on Future Information Technology*, FutureTech' 2010, pp. 1–8, May 2010.

[23] I. Dimitriou, "Analysis of a Priority Retrial Queue with Dependent Vacation Scheme and Application to Power Saving in Wireless Communication Systems," *The Computer Journal*, vol. 56, no. 11, pp. 1363–1380, 2013.

[24] K. M. Lau and O. Yue, "Modeling synchronous wakeup patterns in wireless sensor networks : server vacation model analysis," in *2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications. PIMRC*, vol. 2, pp. 1312–1316, Sept. 2005.

[25] Y. Zhang and W. Li, "Modeling and energy consumption evaluation of a stochastic wireless sensor network," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–11, 2012.

[26] J. Luo, L. Jiang, and C. He, "Finite Queuing Model Analysis for Energy and QoS Tradeoff in Contention-Based Wireless Sensor Networks," in *IEEE International Conference on Communications*, ICC '07, pp. 3901–3906, June 2007.

[27] A. Rustamov, "Determination of QoS metrics in wireless sensor networks by using queuing theory," in *6th International Conference on Application of Information and Communication Technologies*, AICT '12, pp. 1–5, Oct. 2012.

[28] M. Martalo, S. Busanelli, and G. Ferrari, "Multihop IEEE 802.15.4 Wireless Networks With Finite Node Buffers : Markov Chain-Based Analysis," in *IEEE 10th International Symposium on Spread Spectrum Techniques and Applications*, ISSSTA '08, pp. 644–648, Aug. 2008.

[29] Z. Hu, Y. Y. Wen, and H. Zhao, "Message buffer evaluation for wireless sensor networks," in *2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation*, IMSNA '13, pp. 792–795, Dec. 2013.

[30] O. Hachida and K. Kawashima, "Buffer Behaviour with Repeated Calls," *Electronics and Communications in Japan*, vol. 62-B, no. 3, pp. 27–35, 1979.

[31] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo, "GreatSPN 1.7 : GRaphical Editor and Analyser for Timed and Stochastic Petri Nets," *Performance Evaluation*, vol. 24, no. 1-2, pp. 47–68, 1995.

[32] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, pp. 2292–2330, 2008.

[33] S. Abdel Hamid, H. S. Hassanein, and G. Takahara, *Routing for Wireless Multi-Hop Networks*. SpringerBriefs in Computer Science, 2013.

[34] N. Mitton, *Réseaux de capteurs sans fil : De nouveayx défis*. Techniques de l'Ingénieur, 2012.

[35] M. A. Martin, *Wireless Sensor Networks : Technology and Protocols*. InTech, 2012.

[36] D. Culler, D. Estrin, and M. Srivastava, "Overview of Sensor Networks," *IEEE Computer*, vol. 37, pp. 41–49, Aug. 2004.

[37] M. Horton and J. Suh, "A Vision for Wireless Sensor Networks," *Crossbow Technology*, pp. 361–364, 2005.

[38] Y. Busnel, "Systèmes d'informations pair-à-pair pour les réseaux de capteurs larges échelles," *La lettre scientifique de la Fondation Michel Métivier*, Oct. 2006.

[39] J. N. Al-Karaki and A. E. Kamal, "Routing Techniques in Wireless Sensor Networks : a Survey," *IEEE Wireless Communications*, vol. 11, pp. 06–28, Dec. 2004.

[40] J. Wilson, *Sensor Technology Handbook*. Burlington, MA, USA : Elsevier/Newnes, 2005.

[41] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, pp. 325–349, May 2005.

[42] O. Younis and S. Fahmy, "HEED : a Hybryd, Energy-Efficient, Distributed clustering approach for ad hoc sensor networks," *IEEE Transaction on Mobile Computers*, vol. 3, no. 4, pp. 366–379, 2004.

[43] J. Pan, Y. T. Hou, L. Cai, Y. Shi, and S. X. Shen, "Topology Control for Wireless Sensor Networks," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, MobiCom '03, (San Diego, CA, USA), pp. 286–299, 2003.

[44] E. Shih, S. H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical Layer Driven Protocol and Algorithm Design for Energy-efficient Wireless Sensor Networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, MobiCom '01, (Rome, Italy), pp. 272–287, 2001.

[45] I. Salhi, *Un codage réseau contraint pour les réseaux de capteurs sans fil*. PhD thesis, Paris-Est, Apr. 2012.

[46] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, (Baltimore, MD, USA), pp. 95–107, 2004.

[47] K. Sohraby, D. Minoli, and T. Znati, *Wireless Sensor Networks : Technology, Protocols and Applications*. John Wiley & Sons, Inc., 2007.

[48] G. Chalhoub, *MaCARI : Une méthode d'accès déterministe et économe en énergie pour les réseaux de capteurs sans fil*. PhD thesis, Université Balise Pascal, 2009.

[49] L. F. W. van Hoesel and P. J. M. Havinga, "A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks," in *Proc. First Int'l Workshop Networked Sensing Systems*, INSS '04, June 2004.

[50] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3 of *INFOCOM '02*, pp. 1567–1576, 2002.

[51] T. van Dam and K. Langendoen, "An Adaptive Energy-efficient MAC Protocol for Wireless Sensor Networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, (Los Angeles, California, USA), pp. 171–180, 2003.

[52] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-MAC : A Hybrid MAC for Wireless Sensor Networks," *IEEE/ACM Trans. Netw.*, vol. 16, pp. 511–524, June 2008.

[53] J. N. Al-Karaki and A. E. Kamal, "Routing Techniques in Wireless Sensor Networks : a Survey." Department of Electrical and Computer Engineering, Iowa State University, Dec. 2004.

[54] D. Goyal and M. R. Tripathy, "Routing Protocols in Wireless Sensor Networks : A Survey," in *Proceedings of the 2012 Second International Conference on Advanced Computing & Communication Technologies*, ACCT '12, pp. 474–480, 2012.

[55] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, (Seattle, Washington, USA), pp. 174–185, 1999.

[56] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion : A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, (Boston, Massachusetts, USA), pp. 56–67, 2000.

[57] S. Lindsey and C. Raghavendra, "PEGASIS : Power-Efficient Gathering in Sensor Information Systems," *IEEE Aerospace Conference Proceedings*, vol. 3, pp. 1125–1130, 2002.

[58] A. Manjeshwar and D. P. Agrawal, "APTEEN : A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks," *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)*, pp. 195–202, 2002.

[59] Y. Yu, R. Govindan, and D. Estrin, "Geographical and Energy Aware Routing : A recursive data dissemination protocol for wireless sensor networks," Tech. Rep. UCLA/CSD-TR-010023, UCLA Computer Science Department, May 2001.

[60] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher, "SPEED : A Stateless Protocol for Real-Time Communication in Sensor Networks," in *Proceedings of the 23rd International Conference on Distributed Computing Systems*, ICDCS '03, 2003.

[61] S. Lindsay, C. S. Raghavendra, and K. M. Sivalingam, "Data Gathering in Sensor Networks Using the Energy Delay Metric," in *Proceedings of the 15th International Parallel & Distributed Processing Symposium*, IPDPS '01, pp. 188–, 2001.

[62] A. Boukerche, *Algorithms and Protocols for Wireless and Mobile Ad Hoc Networks*. John Wiley & Sons, Inc., 2009.

[63] C. Buratti, A. Conti, D. Dardari, and R. Verdone, "An Overview on Wireless Sensor Networks Technology and Evolution," *Sensors*, vol. 9, no. 9, pp. 6869–6896, 2009.

[64] R. Verdone, D. Dardari, G. Mazzini, and A. Conti, *Wireless Sensor and Actuator Networks : Technologies, Analysis and Design*. Elsevier, 2008.

[65] A. A. Markov, "Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain." Notes of the Imperial Academy of sciences of St. Petersberg VIII Series, Physio-Mathematical College, Vol. XXII, N.9, Dec. 1907.

[66] A. K. Erlang, "Probability and Telephone Calls," *Nyt Tidsskrift Matematik Series B*, vol. 20, pp. 33–39, 1909.

[67] E. Brockmeyer, H. Halstrøm, A. Erlang, and A. Jensen, *The life and works of A.K. Erlang*. Transactions (Akademiet for de tekniske videnskaber (Denmark))), Akademiet for de Tekniske Videnskaber, 1948.

[68] L. Kleinrock, *Queuing systems. Volume 1 : Theory*. New York : John Wiley, 1975.

[69] J. H. Dshalalow, *Advances in Queueing : Theory, Methods and Open Problems*. Boca Raton : CRC Press, Inc., 1995.

[70] J. D. C. Little, "A proof of the queuing formula $L = kW$," *Operations Research*, vol. 9, no. 3, pp. 383–387, 1961.

[71] J. R. Artalejo and A. Gómez-Corral, *Retrial Queuing Systems : A Computational Approach*. Springer-Verlag Berlin Heidelberg, 2008.

[72] L. Kosten, *Stochastic Theory of Service Systems*. Ox-ford : Pergamon Press, 1973.

[73] L. Kosten, "On the influence of repeated calls in the theory of probabilities of blocking," *De Ingenieur*, vol. 59, no. 1, 1947.

[74] J. W. Cohen, "Basic problems of telephone traffic theory and the influence of repeated calls," *Philips Telecommunication Review*, vol. 18, no. 2, pp. 49–100, 1957.

[75] R. I. Wilkinson, "Theories for Toll Traffic Engineering in the U. S. A.*," *Bell System Technical Journal*, vol. 35, no. 2, pp. 421–514, 1956.

[76] J. Riordan, "Stochastic Service Systems." Wiley, New York, 1962.

[77] T. Yang and J. G. C. Templeton, "A survey on retrial queues," *Queueing systems*, vol. 2, pp. 201–233, 1987.

[78] G. I. Falin, "A survey of retrial queues," *Queueing systems*, vol. 7, pp. 127–168, 1990.

[79] Y. N. Kornyshev, "Design of a fully accessible switching system with repeated calls," *Telecommunications*, vol. 23, pp. 46–52, 1969.

[80] A. G. de Kok, "Algorithmic methods for single server systems with repeated attempts," *Statistica Neerlandica*, vol. 38, no. 1, pp. 23–32, 1984.

[81] H. Ohmura and Y. Takahashi, "An analysis of repeated call model with a finite number of sources," *Electronics and Communications in Japan*, vol. 68, no. 6, pp. 112–121, 1985.

[82] V. I. Dragieva, "Single-line queue with finite source and repeated calls," *Problems of Information Transmission*, vol. 30, no. 3, pp. 283–289, 1994.

[83] H. Li and T. Yang, "A single-server retrial queue with server vacations and a finite number of input sources," *European Journal of Operational Research*, vol. 85, no. 1, pp. 149–160, 1995.

[84] N. Tian and Z. G. Zhang, *Vacation Queueing Models : Theory and Applications (International Series in Operations Research & Management Science)*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2006.

[85] B. Doshi, "Queueing systems with vacations - A survey," *Queueing Systems*, vol. 1, no. 1, pp. 29–66, 1986.

[86] H. Takagi, "Queueing Analysis, Vol. 1 : Vacation and Priority Systems," *NorthHolland, Amsterdam*, 1991.

[87] C. A. Petri, "Communication with automata," Tech. Rep. RADC-TR-65-377, Rome Air Dev. Center, New York, 1966.

[88] C. Ramchandani, *Analysis of Asynchronous Concurrent Systems by Time Petri Nets*. PhD thesis, MIT, Cambridge, MA, 1974.

[89] P. M. Merlin, *A study of the recoverability of computer systems*. PhD thesis, Computer Science, University of California, Irvine, CA, 1974.

[90] J. Sifakis, "Use of Petri Nets for Performance Evaluation," in *H Beilner E Gelenbe Eds Measuring Modelling and Evaluating Computer Systems Amsterdam*, pp. 75–93, 1977.

[91] F. J. W. Symons, *Modeling and Analysis of Communication Protocols Using Numerical Petri Nets*. PhD thesis, University of Essex, 1978.

[92] G. Florin and S. Natkin, "Les réseaux de Petri stochastiques," *Technique et Science Informatiques*, vol. 4, no. 1, 1985.

[93] M. K. Molloy, "Performance Analysis Using Stochastic Petri Nets," *IEEE Transaction on Computers*, vol. 31, pp. 913–917, Sept. 1982.

[94] M. A. Marsan, G. Conte, and G. Balbo, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems," *ACM Transactions on Computer Systems*, vol. 2, pp. 93–122, May 1984.

[95] W. Khansa, *Réseaux de Petri P-temporels : contribution à l'étude des systèmes à évènements discrets*. PhD thesis, Université de Savoie, 1997.

[96] G. Balbo, "Introduction to generalized stochastic Petri nets," in *Formal methods for performance evaluation*, pp. 83–131, Springer, 2007.

[97] S. Haddad, P. Moreaux, and G. Chiola, "Distributions de Cox et Phase-type dans les réseaux de Petri stochastiques : Une méthode efficace de résolution," *Recherche opérationnelle*, vol. 32, no. 3, pp. 289–323, 1998.

[98] A. Zenie, *Les réseaux de Petri stochastiques colorés : Application à l'analyse des systèmes répartis en temps réel*. PhD thesis, Université Pierre et Marie Curie, Paris VI, Oct. 1987.

[99] C. Dutheillet, *Symétrie dans les réseaux colorés : Définition, analyse et application à l'évaluation des performances*. PhD thesis, Université Pierre et Marie Curie, Paris VI, Mar. 1992.

[100] S. Hur, J. Kim, and C. Kang, "An analysis of the M/G/1 system with N and T policy," *Applied Mathematical Modelling*, vol. 27, no. 8, pp. 665 – 675, 2003.

[101] K. G. Gakis, H. K. Rhee, and B. D. Sivazlian, "Distributions and first moments of the busy and idle periods in controllable M/G/1 Queueing Models with Simple and Dyadic Policies," *Stochastic Analysis and Applications*, vol. 13, no. 1, pp. 47–81, 1995.

[102] J. Li and P. Mohapatra, "An analytical model for the energy hole problem in many-to-one sensor networks," in *Vehicular Technology Conference, VTC-Fall 2005*, vol. 4, pp. 2721–2725, Sept. 2005.

[103] F. C. Jiang, D. C. Huang, C. T. Yang, and F. Y. Leu, "Lifetime elongation for wireless sensor network using queue-based approaches," *The Journal of Supercomputing*, vol. 59, no. 3, pp. 1312–1335, 2012.

[104] P. Wüchner, J. Sztrik, and H. Meer, "Modeling Wireless Sensor Networks Using Finite-Source Retrial Queues with Unreliable Orbit," in *Performance Evaluation of Computer and Communication Systems. Milestones and Future Challenges* (K. Hummel, H. Hlavacs, and W. Gansterer, eds.), vol. 6821 of *Lecture Notes in Computer Science*, pp. 73–86, Springer Berlin Heidelberg, 2011.

[105] K. Begain, G. Bolch, and H. Herold, *Practical Performance Modeling : Application of the Mosel Language*. Norwell, MA, USA : Kluwer Academic Publishers, 2001.

[106] T. Bérczes, J. Sztrik, P. Orosz, P. Moyal, N. Limnios, and S. Georgiadis, "Tool supported modeling of sensor communication networks by using finite-source priority retrial queues.," *Carpathian Journal of Electronic & Computer Engineering*, vol. 5, no. 1, pp. 13–18, 2012.

[107] T. Bérczes, B. Almási, A. Kuki, J. Sztrik, and R. Kakubava, "Modeling the Performance and the Energy Usage of Wireless Sensor Networks by Retrial Queueing

Systems," in *Proceedings of the 8th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, PM2HW2N '13, (Barcelona, Spain), pp. 133–138, 2013.

[108] Z. Wang, K. Yang, and D. K. Hunter, "Modelling and analysis of multi-sink wireless sensor networks using queuing theory," in *4th Computer Science and Electronic Engineering Conference*, CEEC '12, pp. 169–174, Sept. 2012.

[109] R. B. Lenin and S. Ramaswamy, "Performance analysis of wireless sensor networks using queuing networks," *Annals of Operations Research*, pp. 1–25, 2013.

[110] J. Xiong, M. R. Lyu, and K. W. Ng, "Congestion performance improvement in wireless sensor networks," in *IEEE Aerospace Conference, 2012*, pp. 1–9, Mar. 2012.

[111] M. A. Azgomi and A. Khalili, "Performance Evaluation of Sensor Medium Access Control Protocol Using Coloured Petri Nets," *Electronic Notes in Theoretical Computer Science*, vol. 242, no. 2, pp. 31–42, 2009.

[112] L. Mokdad, J. Ben-Othman, B. Yahya, and S. Niagne, "Performance evaluation tools for QoS MAC protocol for wireless sensor networks," *Ad Hoc Networks*, vol. 12, no. 2014, pp. 86–99, 2014.

[113] B. Yahya and J. Ben-Othman, "An Energy Efficient Hybrid Medium Access Control Scheme for Wireless Sensor Networks with Quality of Service Guarantees," in *Global Telecommunications Conference*, GLOBECOM '08, pp. 1–5, Nov. 2008.

[114] S. Baarir, M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, and G. Franceschinis, "The GreatSPN Tool : Recent Enhancements," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, pp. 4–9, Mar. 2009.

[115] G. Ciardo, J. Muppala, and K. S. Trivedi, "SPNP : Stochastic Peti Net Package," in *Proceedings 3rd International Workshop on Petri Nets and Performance Models* (I. C. Society, ed.), PNPM '89, (Kyoto, Japan), pp. 142–151, Dec. 1989.

[116] E. M. Thurner, "TOMSPIN - A tool for modelling with stochastic Petri nets and performance models," in *Proceedings of International Workshop on Petri Nets and Performance Models* (I. C. S. Press, ed.), PNPM '95, pp. 218–219, Dec. 1995.

[117] W. H. Sanders, W. D. Obal, M. A. Qureshi, and F. K. Widjanarko, "The UltraSAN : Modeling Environment," *Performance Evaluation*, vol. 24, no. 1, pp. 89 – 115, 1995.

[118] C. Lindemann, "DSPNexpress : a software package for the efficient solution of deterministic and stochastic Petri nets," *Performance Evaluation*, vol. 22, no. 1, pp. 3–21, 1995.

[119] G. Chiola, M. A. Marsan, G. Balbo, and G. Conte, "Generalyzed stochastic Petri nets : A definition at the net level and its implications," *IEEE Transactions on Software Engineering*, vol. 19, no. 2, pp. 89–107, 1993.

[120] S. Donatelli, G. Chiola, and G. Franceschinis, "GSPN versus SPN : What is the actual role of immediate transitions ?," in *Proceedings 4th International Workshop on Petri Nets and Performance Models* (I. C. Society, ed.), PNPM '91, (Melbourne, Australia), pp. 20–31, 1991.