$N°$ d'ordre : 11/2017-D/INF

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Electronique et d'Informatique

Thèse présentée en vue de l'obtention du grade de Doctorat en Sciences

En : **Informatique**

Option : **Systèmes Informatique**

Par : **Monsieur Djamel MANSOURI**

Thème :

Analyse et Evaluation des Performances des Réseaux de Capteurs

Soutenue publiquement, le 21/12/2017, devant le jury composé de :

Mme	**AISSANI MOKHTARI**	**Professeur à l'USTHB**	**Président**
Mme	**Malika. BOUKALA**	**Professeur à l'USTHB**	**Directrice de Thèse**
Mme	**Lynda. MOKDAD**	**Professeur à l'UPEC France**	**Co-Directrice de Thèse**
M	**Larbi. SEKHRI**	**Professeur à l'Université d'Oran**	**Examinateur**
Mme	**Chafika. BENZAID**	**MCA à l'USTHB**	**Examinatrice**
M	**Kadda. BEGHDAD BEY**	**MCA à l'EMP/BEB**	**Examinateur**

Résumé

Au vu, de la limitation énergétique considéré comme une contrainte majeure dans les réseaux de capteurs, l'effort de la recherche porte sur la détermination des solutions permettant la réduction de la consommation de l'énergie. En règle générale, économiser de l'énergie se traduit par l'optimisation de la communication entre les nœuds du réseau (envoie et réception des paquets).

La solution la plus adaptée à cette problématique est le clustering adopté pour la conception des protocoles de communication. Le clustering est une méthode qui permet la construction des groupes de nœuds pour former une structure logique concernant une métrique donnée. L'objectif du regroupement est de prolonger la durée de vie d'un réseau en fournissant un équilibrage de charge et une connectivité élevée.

Sur un autre plan, les problèmes de sécurité liés au déploiement des réseaux de capteurs sont une exigence importante et fréquente. En effet, les informations sensibles produites par le réseau doivent être protégées et l'intégrité des données et l'authenticité doivent être garanties. En plus, la disponibilité du réseau est également un problème qui doit être pris en considération. En effet, le réseau ne remplit pas ses fonctions si ses services ne sont pas disponibles. Les réseaux de capteurs sont plus vulnérables à des attaques de Déni de Service. Ces attaques peuvent avoir de graves répercussions. En fait, ils réduisent ou éliminent la capacité du réseau pour l'exécution de ses tâches.

Plusieurs travaux de recherche ont été menés pour résoudre ce problème. Les solutions apportées sont généralement vérifiées par des simulations. Néanmoins, pour des applications réelles qui exigent une exactitude élevée dans les résultats, ces solutions peuvent être inadiquates. Cependant, le recours à des méthodes basées sur des principes mathématiques solides est nécessaire. Ces méthodes dites formelles, peuvent être utilisées comme des supports méthodologiques pour développer des solutions de manière efficace et pour assurer leur sûreté de fonctionnement. Parmi ces formalismes, les RdPs et les RASs constituent un bon support pour la modélisation et l'analyse de différents concepts dans les réseaux de capteurs.

Pour parvenir à ces buts, d'une part, deux nouvelles méthodes permettant la détection des attaques de Déni de Service (DoS) dans les réseaux de capteurs, basées sur des algorithmes de clustering dynamique, appelées R-Leach et R-FFUCA ont été proposées et décrites dans le cadre de cette thèse. Et d'autre part, nous avons proposé et évalué les performances d'une approche dynamique et adaptatif appelée Accordéon, pour la détection du flooding dans les réseaux de capteurs. Aussi, une modélisation de la méthode proposée par les Réseaux d'Automates Stochastique a été décrite dans cette thèse. A travers les résultats numériques, l'évaluation des performances de la méthode proposée a montré que son utilisation est très intéressante en termes de temps de détection et de durée de vie du réseau en comparaison avec d'autres méthodes de détection.

Enfin, nous avons présenté une autre contribution qui consiste en la modélisation par les réseaux de Petri Colorés Stochastiques Généralisés, d'une part, de la détection des attaques de Déni de Service (DoS). D'autre part, d'une stratégie de défense contre les attaques actives dans les RCSF.

Mots-clés : Modèle formel, Réseaux de Petri, Attaques DoS , Réseaux de capteurs sans fil, Le clustering, LEACH, FFUCA.

Abstract

In view of the energy limitation considered as a major constraint in sensor networks, the research effort is focused on the identification of solutions to reduce energy consumption. Saving energy can be resulted by optimizing the communications between network nodes (sending and receiving packets).

The most suitable solution for this problem is the clustering approach which is adopted for the design of communication protocols. Clustering is a method that allows the construction of node groups to form a logical structure for a given metric. The goal of clustering is to extend network lifetime by providing load balancing and high connectivity.

On another level, the security problems associated with the deployment of sensor networks are a prevalent and an important requirement. Indeed, the sensitive information produced by the network must be protected and the integrity of data and authenticity must be guaranteed. In addition, network availability is also a problem that needs to be taken into consideration. In fact, the network does not perform its functions if its services are not available. Sensor networks are more vulnerable to DoS attacks. These attacks can have serious repercussions. In fact, they reduce or eliminate the network's ability to perform its tasks.

Several research works have been conducted to solve this problem. The solutions provided are usually tested by simulation. However, for real applications that require high accuracy in the results, these solutions may not be adequate. Nevertheless, the resort to solid mathematical principals based solutions is necessary. These so-called formal methods can be used as methodological supports to develop solutions efficiently and to ensure their safe operation. Among these formalisms, Petri Nets and RASs constitute a good support for the modeling and analysis of different concepts in sensor networks.

To achieve these goals, on one hand, two new methods for detecting DoS attacks in sensor networks, based on dynamic clustering algorithms, called R-Leach and R-FFUCA, have been proposed and described in this thesis. On the other hand, we proposed and evaluated the performance of a dynamic and adaptive approach called Accordion for the detection of flooding in sensor networks. Also, a modeling of the proposed method by the Stochastic Automata Networks was proposed and described in this thesis. Through the numerical results, the evaluation of the performance of the proposed method shows that its use is very interesting in terms of detection time and network lifetime compared to other detection methods.

Finally, we presented another contribution, which consists of modeling by the Generalized Stochastic Petri Nets, first, the detection of denial of service attacks. Second, a defense strategy against actives attacks in WSN.

Keywords : Formal model, Petri nets, DoS attacks, Wireless sensor networks, Clustering, LEACH, FFUCA.

Remerciements

Je voudrais en premier lieu remercier tous les membres du jury pour m'avoir fait l'honneur d'accepter de juger ce travail de thèse :

- Mme AISSANI MOKHTARI Professeur à l'USTHB, Président président du jury.
- Mme Malika. BOUKALA Professeur à l'USTHB, Directreur de Thèse.
- Mme Lynda. MOKDAD Professeur à l'UPEC France, Co-Directreur de Thèse
- Mr Larbi. SEKHRI Professeur à l'Université d'Oran, Examinateur.
- Mme Chafika. BENZAID MCA à l'USTHB, Examinateur.
- Mr Kadda. BEGHDAD BEY MCA à l'EMP, Examinateur.

Je tiens à remercier mon directeur de thèse le professeur Malika. BOUKALA qui m'a proposé un sujet dans une thématique aussi intéressante, d'actualité et d'avenir. Je tiens à lui faire part également de toute ma reconnaissance pour avoir cru en mes capacités et avoir su me montrer la voie à suivre. Elle m'a prodigué des conseils précieux tout au long de mes recherches et qui ont été d'une grande aide. Qu'elle sache que je lui voue une vive estime.

Je ne saurai trop remercier mon co-directeur, le professeur Lynda MOKDAD pour son soutien, ses conseils précieux, sa sympathie et l'intérêt qu'elle a manifestée à mon travail. Qu'elle trouve ici ma profonde gratitude et mes remerciements les plus sincères. Je voudrais également la remercier pour m'avoir accueilli au sein de son équipe de recherche et fourni d'excellentes conditions au niveau du laboratoire LACL. De plus, je lui adresse de chaleureux remerciements pour le temps qu'elle m'a accordé.

Je suis reconnaissant envers le professeur Jalel Ben-othman pour la qualité de sa collaboration, ses nombreux conseils, et pour la façon amicale avec laquelle il a suivi ce travail, ainsi que pour ses remarques pertinentes qui ont permis d'approfondir et d'enrichir les problématiques abordées au cours de ce travail.

Mes remerciements vont bien sûr à tous ceux qui y ont participé de près ou de loin à la réussite de ce travail, aux membres des laboratoires MOVEP de l'USTHB Algerie et LACL de l'UPEC France.

Mes remerciements les plus profonds vont à mes parents, particulierement, ma défunte mere qui m'a soutenu motivé durant mes nombreuses années d'études. Merci à toute ma famille pour leur soutien et la confiance qu'ils n'ont jamais cessé de me témoigner. Je réserve une pensée particulière à mon épouse, mes poussins et ma raison de vivre : Lyna, Sarah et Mohamed Anes. Je vous aime !

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

1- Cadre général

La technologie des Réseaux de Capteurs Sans Fil (RCSF, ou WSN : Wireless Sensor Networks) a étè considérée comme l'une des technologies les plus importantes dans le 21^{ime} siècle. D'après MIT's Technology Review, c'est une nouvelle technologie qui va révolutionner le monde actuel et changer notre façon de vivre et de travailler [1], en particulier, grâce à la disponibilité des capteurs qui sont plus petits, plus économiques et plus intelligents.

La conception d'un RCSF dépend principalement des exigences des applications auxquelles il est dédié. Cependant, lors de la conception des capteurs, on doit considérer les facteurs tels que, l'environnement, les objectifs de conception pour lesquels ils sont déployés, le coût du matériel et les contraintes des systèmes formés par ces capteurs, particulièrement, la contrainte énergétique.

2- Problématique

Avant la mise en place d'un réseau de capteurs, son déploiement nécessite une phase de test et d'analyse des performances, afin de s'assurer du bon fonctionnement de tous ses composants. En effet, les capteurs peuvent tomber en panne à cause des défauts de construction industrielle, de l'influence et de l'hostilité de l'environnement ou bien de l'expiration de la batterie des capteurs, due principalement aux communications sans fil.

En pratique, la simulation reste la technique la plus utilisée pour analyser et évaluer les performances des réseaux de capteurs, et ce pour réduire le maximum d'erreurs de conception et assurer un fonctionnement correct de ces réseaux. Un environnement de simulation qui teste, par exemple, les protocoles avec une bonne précision serait très bénéfique en termes de temps et de coûts. Les simulateurs permettent ainsi d'anticiper sur plusieurs paramètres d'un réseau pour porter des modifications ou des ajustements.

Aussi, les modèles mathématiques formels peuvent servir pour la modélisation et l'évaluation des performances de ces réseaux. Généralement, la modélisation, l'analyse et la vérification formelle des systèmes, se basent sur des modèles qualitatifs et quantitatifs (les systèmes de transitions et les réseaux de Petri, les files d'attentes, les chaînes de Markov,...).

Dans le même ordre d'idée, nous nous concentrons dans ce document sur l'utilisation de ces modèles formels pour la modélisation et l'évaluation des performances des réseaux de capteurs.

Par ailleurs, dans les RCSF, d'une part, les nœuds de capteurs ont des ressources limitées en termes : d'énergie, de puissance de transmission, de mémoire, et de capacité de calcul. D'autre part, l'énergie consommée par les nœuds de capteurs pour la transmission des données est la cause principale de l'épuisement de l'énergie dans les nœuds de capteurs. Pour remédier à ces contraintes, plusieurs solutions existantes incluent souvent l'utilisation d'un algorithme de clustering. Le clustering joue un rôle important dans l'économie de l'énergie dans les réseaux de capteurs. Avec le regroupement des nœuds en clusters, la consommation de l'énergie, la durée de vie du réseau et l'évolutivité peuvent être améliorées.

Dans ce contexte, nous présentons, une approche qui consiste à appliquer sur les RCSF, l'algorithme de clustering général, basé sur des propriétés ultramétriques et appelé FFUCA (Fast and Flexible Unsupervised Clustering Algorithm) .

Pour évaluer cette approche par rapport à une mesure de performance qui représente la consommation d'énergie des nœuds de capteurs, nous avons comparé les résultats numériques obtenus de la simulation avec ceux résultant de l'appliquation d'un autre algorithme en l'occurrence LEACH.

En outre, en considérant l'aspect sécurité, les RCSF sont plus vulnérables aux attaques que les réseaux filaires. Cependant, les fréquences radios utilisées dans ces réseaux sont ouvertes, ce qui permet, par exemple à un attaquant d'intercepter et de recueillir des données sur le réseau, voire modifier le contenu des données échangées entre les nœuds ou, de mettre facilement les communications sous écoute-espionne. Mais parfois, l'attaquant essaie de ralentir ou de bloquer complètement le fonctionnement du réseau jusqu'à le détruire (attaques de Déni de Service, DoS).

Comme indiqué précédemment, l'un des principaux défis des réseaux de capteurs est le maintien du niveau d'énergie de leur batterie pour prolonger leur durée de vie. Le routage hiérarchique basé sur le clustering est une technique très efficace pour résoudre le problème de la consommation d'énergie. Ainsi, l'objectif du clustering est de prolonger la durée de vie d'un réseau en fournissant un bon équilibrage de charge entre les différents nœuds des RCSF.

Dans cette thèse, nous nous focalisons aussi, sur l'étude du problème des attaques DoS dans les RCSF, particulièrement l'évaluation des performances de ces réseaux sous ces attaques. L'objectif recherché consiste à définir et à analyser des méthodes et des approches permettant la détection des attaques de déni de service, tout en tenant compte des contraintes énergétiques liées à ce type de réseau. Dans ce cadre, nous présentons notre contribution par la proposition d'une nouvelle méthode de détection des attaques DoS, basée sur le clustering récursif, en l'occurrence les méthodes récursives FFUCA et LEACH. Aussi, par la proposition et la modélisation par les réseaux d'automates stochastiques, d'une autre méthode de détection des attaques DoS appelée Accordéon. Enfin, nous présentons l'analyse et l'évaluation de nos propositions par rapport à plusieurs métriques, tels que, le taux de détection, le temps de détection, la consommation d'énergie et la durée de vie.

3- Organisation du document

Cette thèse est composé de quatre chapitres. Nos contributions sont détaillées dans les trois derniers.

Initialement nous présentons une introduction générale, tout en mettant l'accent sur la problématique étudiée.

L'objet du premier chapitre est de présenter un état de l'art sur les réseaux de capteurs sans fil en axant particulièrement notre réflexion sur les composantes, les caractéristiques et les spécificités, ainsi que les contraintes liées à la conception de ces réseaux. Aussi, nous présentons une étude du problème des attaques DoS dans les RCSF, particulièrement l'évaluation des performances des méthodes et des techniques de détection de ces attaques, en tenant compte des contraintes énergétiques liées à ce type de réseau, afin de proposer dans la suite du document des modèles permettant l'évaluation des performances de ces réseaux, en termes de consommation d'énergie, en considérant leur exigences de conception et de bon fonctionnement.

Le deuxième chapitre porte sur les méthodes de détection des attaques DoS basées sur le clustering. Nous donnons d'abord, une vue d'ensemble sur le clustering et nous présentons, les méthodes de clustering appelées FFUCA et LEACH, avec une évaluation des performances des deux méthodes, en faisant une comparaison des résultats numériques obtenus par simulation. Nous enchainons, par notre contribution qui consiste en une nouvelle méthode de détection des attaques DoS basée sur le clustering, les méthodes récursives FFUCA et LEACH. Pour valider notre proposition à travers la comparaison des résultats numériques, nous avons utilisé la mesure de performance qui représente la consommation d'énergie des nœuds du réseau de capteurs.

En cherchant à trouver un compromis entre la gestion de la consommation d'énergie et le bon fonctionnement des RCSF sous des attaques DoS, nous abordons dans le chapitre trois notre autre contribution dans ce domaine. Elle consiste à la proposition et la modélisation d'une autre méthode de détection des attaques DoS, appelée la méthode Accordéon. Notre proposition est basée sur le clustering récursif, pour l'élection des nœuds spécifiques qui auront comme tâche principale le monitoring du réseau. Nous présentons aussi, une modélisation de la méthode proposée par les Réseaux d'Automates Stochastiques. Enfin, nous donnons l'évaluation des performances de nos propositions en termes, de taux de détection, de temps de détection, de consommation d'énergie et de durée de vie.

Le dernier chapitre est dédié à l'etude des différentes méthodes et techniques de modélisation de ces réseaux, ainsi qu'un aperçu sur les différents travaux utilisant les modèles formels de modélisation et d'évaluation des performances de ces réseaux. Dans ce contexte, en utilisant les réseaux de Petri Stochastiques Colorés Généralisés, nous présentons une modélisation d'une méthode de détection des attaques DoS dans les RCSF, et d'une stratégie de défense contre les attaques actives dans ces réseaux.

Enfin, la conclusion générale, synthétise les différents points abordés dans ce document, récapitule les résultats obtenus, et introduit également les perspectives de ce travail de thèse.

Chapitre 1

ETAT DE L'ART SUR LES RESEAUX DE CAPTEURS

Sommaire

1.1 Introduction

Les réseaux de capteurs sont un ensemble de petites entités autonomes interconnectées par des liaisons de communication sans fil. Ces entités coopèrent pour acquérir et transmettre des mesures prises dans une zône géographique donnée. Par rapport à leurs petites tailles, les capteurs ont de nombreuses limitations telles que, la capacité de traitement, de stockage et surtout d'énergie. Les nœuds de capteurs sont généralement dotés d'une source d'alimentation limitée. Par conséquent, inclure des outils et des mécanismes permettant d'économiser de l'énergie et d'augmenter la durée de vie du réseau s'avère nécessaire. Ainsi, la conception d'un RCSF doit tenir compte des limites et des différentes contraintes et exigences liées à ces réseaux, notamment la contrainte énergétique qui représente un paramètre clé pour évaluer la fiabilité et le bon fonctionnement d'un RCSF. Dans ce chapitre, nous présentons un état de l'art détaillé sur les réseaux de capteurs sans fil.

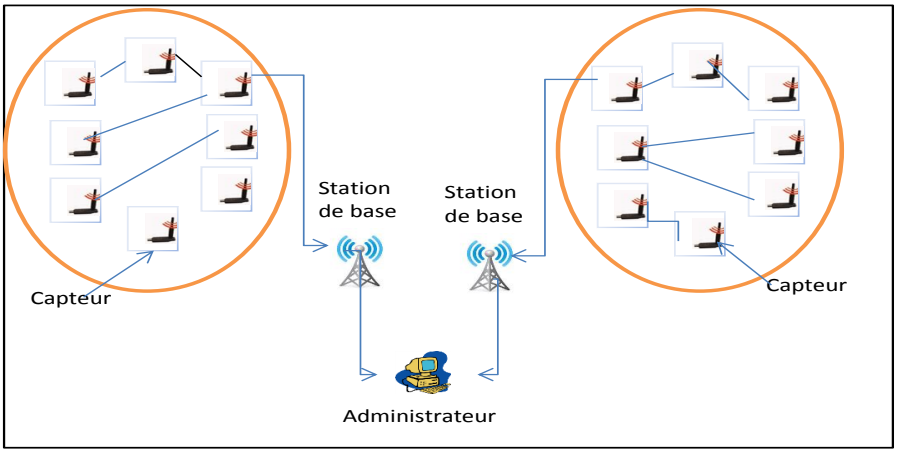


FIGURE *1.1* – Schéma d'un réseau de capteurs.

Qu'est-ce qu'un réseau de capteurs sans fil ?

Un réseau de capteurs est un ensemble de petits dispositifs appelés "nœuds capteurs", déployés dans un environnement pour répondre à un besoin spécifique tel que, la surveillance d'une zône donnée. Ces nœuds capteurs communiquent entre eux avec une station de base (nœud central ou nœud récepteur) à travers des liens de communication sans fil. Ces nœuds capteurs sont dit intelligents, car ils sont dotés de facultés leur permettant de relever une grandeur physique et la transformer en signaux électriques. Après traitements, ces signaux sont envoyés à la station de base (SB) pour exploitation. Ces réseaux sont utilisés généralement pour effectuer des mesures périodiques dans des zônes où l'être humain ne peut y accéder. Dans de tels cas, les capteurs sont répartis sur de larges zônes pour la surveillance constante de ces dernières. Par exemple, la détection des feux de forêt,

l'activité sismique, les gaz toxiques ou les menaces nucléaires, l'évaluation de la pollution de l'eau, ou la détection des cibles mobiles sur un terrain militaire.

1.2 Les applications des réseaux de capteurs

Dans la littérature, beaucoup d'auteurs ont présenté les applications des réseaux de capteurs selon les cinq domaines suivants :

1.2.1 Les applications environnementales

La surveillance environnementale est un vaste espace pour les applications des réseaux de capteurs. Leurs déploiements offrent la possibilité de la surveillance discrète des zônes qui sont difficiles d'accès pour l'homme. Nous pouvons trouver dans la littérature plusieurs travaux ayant traité et examiné ce type d'applications [2], [3], [4], [5]. Par exemple, la surveillance des feux de forêts [6], [7], la détection de la pollution et l'analyse de la qualité de l'air dans les milieux urbains, l'étude des écosystèmes des eaux de mer [8], [9], la surveillance de l'activité volcanique [10], l'observation de la faune, la prévention des catastrophes et la surveillance météorologique.

1.2.2 Les applications médicales

Dans le domaine médical, les RCSF peuvent servir pour surveiller les fonctions organiques assurant la survie d'un être humain à travers l'implantation de capteurs à l'intérieur de son corps (e.g. BAN - Body Area Network). Ces capteurs sont utilisés pour collecter et mesurer des indicateurs physiologiques (la tension artérielle, la glycémie, les pulsations cardiaques) ou détecter, diagnostiquer et traiter certaines maladies. Pour le suivi et la prise en charge d'un malade, les données physiologiques collectées peuvent être stockées pendant une longue durée [11],[12]. Aussi, ces réseaux peuvent servir pour la prise en charge des patients à mobilité réduite, permettant ainsi, de détecter et de déceler la présence d'un handicap ou d'un phénomène ayant un comportement étrange ou anormal chez ces malades [11]. Plusieurs applications ont été conçues dans ce domaine par exemple : la plate-forme Code Blue [13], MEDISN (Medical Emergency Detection in Sensor Networks) [14] et la plate-forme Mercury [15].

1.2.3 Les applications militaires

Plusieurs projets ont été initiés dans ce domaine, d'une part, pour servir les unités de combats dans un champ de bataille ou dans des endroits stratégiques et hostiles, afin de

surveiller l'activité des unités ennemies, ou d'analyser le terrain avant le déploiement des troupes. D'autre part, pour assurer la protection des villes contre des attaques nucléaires, biologiques ou chimiques (NBC). Dans [16], les auteurs ont cité quleques projets dans ce domaine, tels que :

- Le projet DSN (Distributed Sensor Network) [16] a été le premier projet utilisant les réseaux de capteurs dans le domaine militaire.
- Le projet intitulé "line in the sand" représente un réseau de capteurs conçu pour la détection, la classification et le suivi des mouvements des objets mobiles intrus déployés sur la base aérienne de Macdill en Floride [17].
- Le projet WATS (Wide Area Tracking System) [18] ayant pour objectif la détection et le dépistage des dispositifs nucléaires.
- Le projet VigilNet [19] a été conçu pour collecter les informations sur les capacités de l'ennemi et les positions des cibles hostiles.
- Le projet JBREWS (Joint Biological Remote Early Warning System) [20] utilisé pour avertir les troupes dans le champ de bataille sur des éventuelles attaques biologiques.
- Le projet PinPtr [21] est un RCSF permettant de localiser des tireurs d'élite et la trajectoire des balles.

1.2.4 Les applications commerciales

Dans le domaine commerciale, des micros capteurs peuvent être installés dans les produits commerciaux afin de traquer le processus de stockage et de livraison de ces derniers, de vérifier l'inventaire de chaque produit dans un entrepôt, ce qui permet au gestionnaire de localiser l'endroit exact du produit et de savoir le nombre de produits restants dans le stock. Aussi, des capteurs peuvent être utilisés dans une usine pour surveiller l'état d'un matériel, pour contrôler l'automatisation des processus d'usinage. A titre d'exemple, British Petroleum a développé un système appelé Cobis and Accenture Technology Labs [22] basé sur des capteurs permettant le contrôle des entrepôts et la gestion du stockage des barils de pétrole.

1.2.5 Les applications domotiques

On peut citer plusieurs applications dans ce domaine, par exemple dans les immeubles, le système de climatisation peut être conçu en intégrant plusieurs micro-capteurs dans les tuiles du plancher. Ainsi, la climatisation peut être déclenchée seulement aux endroits où il y a présence de personnes [23]. Les smart homes ou les maisons intelligentes sont

dotées de capteurs permettant d'une part, de contrôler, d'activer et d'arrêter les appareils électroménagers. D'autre part, de contrôler automatiquement l'ouverture et la fermeture des portes et des rideaux, le déclenchement de l'arrosage des plantes de jardins, l'allumage de la télé ou la lumière, etc. Ainsi, dans la domotique, les réseaux de capteurs permettent une manipulation aisée, une maintenance et un contrôle à distance des différents dispositifs ménagers [24]. Aussi, dans ce domaine, les réseaux de capteurs peuvent également être utilisés pour contrôler et surveiller les vibrations susceptibles d'endommager la structure d'un immeuble [25].

1.3 Architecture d'un nœud de capteur

En général, l'architecture d'un capteur sans fil est composée de quatre sous-systèmes [26] (Figure 1.2), à savoir :

1.3.1 Sous-système de communication

Le sous-système de communication permet des liaisons sans fil entre les nœuds de capteurs et le monde extérieur. Il reçoit des commandes provenant du sous-système de calcul et de traitement et les transmet à un autre nœud du réseau. Le sous-système se compose d'un émetteur-récepteur d'une courte portée qui a généralement un seul canal et un faible débit de données. Généralement, les capteurs utilisent des communications Radio Fréquence (RF). Il est à noter que, la communication radio fréquence joue un rôle important dans la conservation de l'énergie sachant qu'un capteur à communication radio fréquence peut fonctionner en quatre modes différents, le mode transmission, réception, inactivité et le mode veille. Pour économiser de l'énergie, il est nécessaire de mettre la radio en mode veille [27].

1.3.2 Sous-système de détection

Fondamentalement, un capteur sert à détecter des phénomènes physiques. Les capteurs peuvent être actifs ou passifs. Un capteur actif est un capteur qui détecte les phénomènes avec une manipulation active (par exemple un radar), alors qu'un capteur passif est un capteur qui détecte l'environnement sans manipulation active, par exemple, le thermomètre ou le microphone, etc. En raison de la diversité des capteurs, la consommation d'énergie n'est pas la même lors du fonctionnement de ce sous-système, en effet, elle varie en fonction de la puissance de ces capteurs.

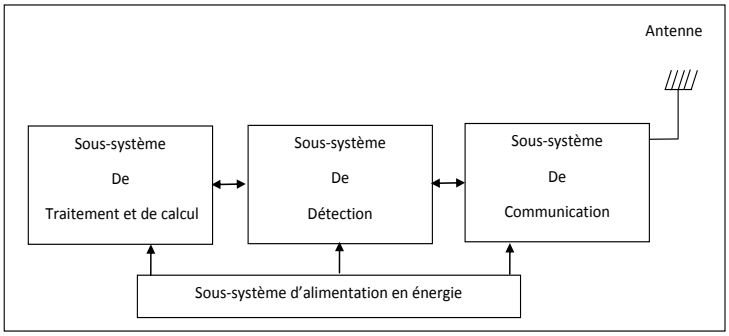


FIGURE *1.2* – Architecture d'un capteur.

1.3.3 Sous-système d'alimentation en énergie ou bloc d'alimentation

Ce sous-système est responsable de la fourniture de l'énergie au nœud de capteur. Le noyau du sous-système d'alimentation en énergie est la batterie qui est l'élément essentiel dans le nœud de capteur, car elle alimente tous ses composants. Par conséquent, la durée de vie des capteurs dépend totalement de sa batterie d'alimentation. Ainsi, la consommation d'énergie est un facteur fondamental à prendre en considération lors de la conception des capteurs.

1.3.4 Sous-système de traitement et de calcul

Il est responsable du traitement et du stockage des données provenant de diverses sources. Il est composé d'un processeur et d'une mémoire intégrant un système d'exploitation spécifique (TinyOS) [28].

En plus des sous-systèmes cités précédemment et selon le domaine d'application, un nœud peut être équipé de sous-systèmes supplémentaires ou optionnels, tels que, le système de localisation (GPS), qui permet de déterminer sa position, ou bien le sous-système générateur d'énergie (cellule photovoltaïque), ou encore un sous-système mobile pour lui permet de changer sa position ou sa configuration en cas de nécessité.

En général, nous pouvons trouver plusieurs types de capteurs avec des architectures différentes. Cette différence est due d'une part, à la spécificité et au rôle attribué à chaque capteur et d'autre part, aux applications pour lesquelles ils ont été conçu. L'objectif des concepteurs des architectures des capteurs étant de maximiser leur durée de vie par la réduction de la consommation énergétique qui dépend fortement du type du nœud. Dans

[29], les auteurs ont montré que la consommation prédominante de l'énergie dans un nœud capteur se fait généralement pendant la communication et le traitement des données. En règle général, nous pouvons définir trois niveaux dans lesquels se produit la consommation de l'énergie dans un capteur [26] à savoir au niveau de :

-) La détection : C'est l'énergie consommée par un nœud de capteur lors de l'activation de son unité de collecte des données. Le coût de cette énergie dépend du type du capteur (image, son, température, etc.).

-) Le traitement des données : C'est l'énergie consommée par un nœud lors de l'activation de son unité de traitement de données.

-) La communication : C'est l'énergie consommée par un nœud lors de l'activation de son unité de transmission. Cette énergie est beaucoup plus élevée que celle dissipée par l'unité de traitement.

1.4 La pile protocolaire des réseaux de capteurs sans fil

La plupart des architectures communes des RCSF, suit le modèle OSI (le standard de la communication entre les nœuds d'un réseau), pour permettre aux différents constructeurs de mettre au point des produits (logiciels ou matériels) compatibles. Principalement, la pile protocolaire d'un réseau de capteurs est composée de cinq couches tel que discuté et illustré dans la Figure 1.3 [30].

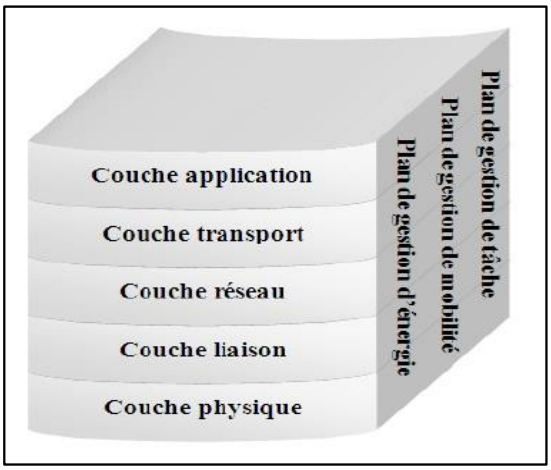


FIGURE *1.3* – Pile protocolaire d'un réseau de capteurs.

- **Couche physique** : Cette couche fournit une interface permettant de transmettre un flux de bits sur des supports physiques.

- **Couche liaison de données** : Cette couche est responsable du contrôle et de la correction des erreurs des dispositifs de détection.
- **Couche réseau** : La principale fonction de cette couche est l'adressage, le routage et l'acheminement des données via le réseau.

  A noter que les protocoles de routage peuvent être scindés en fonction des applications et du type de trafic des données, nous distinguons les applications orientées temps (time-driven), les applications orientées requêtes (query-driven), et les applications orientées évènements (event-driven) [31], [32].

- **Couche transport** : La fonction de cette couche est le transport et le découpage des données en paquets, le contrôle de flux, l'ordonnancement de l'ordre des paquets et la gestion des éventuelles erreurs de transmission. Cette couche fournit la fiabilité et garantit la non-congestion.
- **Couche application** : Cette couche assure l'interface avec les applications. Il s'agit donc du niveau le plus proche des utilisateurs, géré directement par les logiciels.

1.5 Contraintes, caractéristiques et défis de conception

La réalisation des réseaux de capteurs dédiés aux applications citées dans la section précédente, exige des techniques et des protocoles qui prennent en compte les spécificités et les exigences de ces réseaux. Dans ce qui suit, nous présentons les contraintes, les caractéristiques et les défis liés à la conception des capteurs d'un point de vue software et hardware. La conception des capteurs formant un réseau doit répondre aux exigences suivantes [33], [34], [35], [36] :

- Architecture évolutive et souple

Les RCSF sont généralement déployés avec un grand nombre de capteurs, ils peuvent atteindre le million. Ainsi, la conception de ces réseaux doit permettre d'entendre facilement la taille du réseau. Pour assurer une communication efficace intra et extra capteurs, la présence de plusieurs nœuds dans le réseau signifie que le nombre de messages échangés est trés important. Dans ce cas, un nombre aussi élevé de nœuds engendre beaucoup d'échanges de données et nécessite que la station de base soit équipée de mémoire importante pour stocker les informations reçues. Ceci peut engendrer des problèmes de communication et de contrôle qui nécessitent des protocoles capables de les gérer et de garantir un bon fonctionnement sans altérer les performances du réseau. Les protocoles de communication doivent

être conçu de manière à assurer le déploiement d'un grand nombre de nœuds dans le réseau sans affecter le routage et le clustering. En d'autres termes, le réseau doit préserver sa stabilité.

- Les erreurs liées à la communication sans fil

Les réseaux de capteurs peuvent être déployés dans des situations différentes, les exigences de chaque application peuvent varier de manière significative. Nous pouvons alors considérer que le médium sans fil peut être fortement affecté par les environnements bruyants. Un attaquant peut interférer délibérément et causer suffisamment de bruits pour affecter la communication.

- Tolérance aux pannes et adaptabilité

La tolérance aux pannes permet de maintenir les fonctionnalités du RCSF sans aucune interruption due à une défaillance ou un blocage causé par l'épuisement des batteries des capteurs, un préjudice physique, des problèmes de communication, ou d'interférence environnementale. Ces problèmes ne devraient pas affecter le reste du réseau.

- Déploiement ad hoc

Les réseaux de capteurs sont déployés de façon aléatoire, sans aucune infrastructure. Par exemple, pour la détection d'incendie dans une forêt, les nœuds sont généralement lâchés à partir d'un avion et ils peuvent communiquer directement avec la station de base. Ces réseaux utilisent un système radio multi-sauts et le nombre de stations de base dépend de la superficie couverte par les capteurs et de leur portée radio. Par conséquent, les protocoles devraient être capables de gérer le problème du déploiement ad hoc.

- Déploiement du nœud

Les réseaux de capteurs peuvent être déployés de façon aléatoire dans une zône donnée. Après le déploiement, ils peuvent être automatiquement maintenus sans présence humaine. Le déploiement des nœuds de capteurs se fait de deux manières, soit d'une manière dense ou dispersée. Dans le premier cas, un nombre relativement élevé de nœuds de capteurs sont déployés, alors que dans le second cas, moins de nœuds sont déployés. Ce type de déploiement est utilisé lorsque le coût des nœuds de capteurs est très élevé. Le déploiement dense est utilisé lorsqu'il est important de détecter à chaque instant (application de surveillance

sensible) ou lorsque l'on a plusieurs capteurs pour couvrir une superficie (application de surveillance de feu de forêt).

- Temps réel et qualité de service

Le maintien des caractéristiques temps réels et des critères de qualité de service dans les réseaux de capteurs est difficile à réaliser, car on doit assurer un maximum de la bande passante, un minimum de pertes de paquets, une réduction du délai de transfert de bout en bout entre les paquets d'un même flux de données, une diminution de l'atténuation du signal qui est fortement liée à la distance qui sépare les nœuds [33]. Ces problèmes peuvent affecter la fiabilité de certaines applications des réseaux de capteurs, particulièrement, les applications multimédias qui nécessitent une bonne qualité vidéo, audio et image, ou certaines applications dont le facteur temps est critique où la livraison des données détectées devrait être périodique. Par conséquent, les protocoles conçu pour de telles applications doivent gérer le problème du temps réel et la qualité de service [33].

- Changements dynamiques

Les nœuds du réseau de capteurs devraient s'adapter aux changements dûs à l'ajout de nouveaux nœuds ou à la défaillance des nœuds.

- Contraintes physiques

Lors de la conception physique des capteurs, on doit se focaliser sur l'économie de l'énergie. Les différentes unités composant l'architecture du capteur tels que le microcontrôleur, le contrôleur de l'énergie et de la puissance, et l'unité de communication devraient consommer moins d'énergie.

- Consommation d'énergie

La contrainte la plus importante dans le réseau de capteurs est la limitation de la puissance de la batterie des nœuds de capteurs qui sont déployés dans un environnement hostile, où la recharge et le remplacement de la batterie sont impossibles. Ainsi, les nœuds sont dépendants de la batterie pour leur alimentation et la durée de vie effective du réseau de capteurs est directement dépendante de la batterie. Par conséquent, la conservation et la gestion de l'alimentation sont des problèmes importants dans un RCSF, à prendre en charge lors de la conception des protocoles.

- Limitation de la puissance de calcul et la taille mémoire

C'est un autre facteur qui affecte les réseaux de capteurs dans la mesure, où chaque nœud mémorise les données et parfois plus d'un nœud stocke les mêmes données et les transfèrent à la station de base, ce qui engendre une consommation sans intérêt de l'énergie. Par conséquent, on devrait développer des schémas d'acheminement de données efficaces pour réduire au minimum la redondance dans le réseau.

- Transmission de courte portée

Dans les RCSF, on devrait envisager la transmission de courte portée, afin de réduire la possibilité d'être intercepté, ou écouté et de réduire aussi, le coût énergétique.

- Sécurité

Les problèmes de sécurité liés au déploiement des réseaux de capteurs sont une exigence importante et fréquente. En effet, les informations sensibles produites par le réseau doivent être protégées et l'intégrité des données et l'authenticité doivent être garanties. En plus, la disponibilité du réseau est également un problème qui doit être pris en considération. Par conséquent, le réseau ne remplit pas ses fonctions si ses services ne sont pas disponibles. En général, ces réseaux sont vulnérables à trois types d'attaques (les écoutes passives, les attaques actives, l'usurpation d'identité) [37].

1.6 La sécurité dans les réseaux de capteurs sans fil

Dans cette section, nous nous concentrons sur les problèmes des attaques DoS dans les RCSF, particulièrement l'évaluation et l'analyse des méthodes et techniques de détection des attaques DoS, en tenant compte des contraintes énergétiques liées à ce type de réseaux. D'une part, nous présentons une vue d'ensemble des différentes attaques DoS produites dans les différentes couches de la pile protocolaire et nous enchainons par un état de l'art sur les différents travaux ayant proposé des techniques et des méthodes permettant la détection de ces attaques.

1.6.1 Les différents types d'attaques DoS dans les différentes couches de la pile protocolaire

Les attaques DoS, impliquent la tentative de perturber, de corrompre ou de détruire un réseau. Principalement, tous les incidents dans lesquels un utilisateur est privé de services de ressources qui sont normalement en attente d'être disponibles [38]. Dans les RCSF, l'objectif d'une attaque DoS est de rendre les nœuds cibles inaccessibles aux utilisateurs légitimes [39]. Wood et Stankovic ont discuté dans [38] les différents types d'attaques DoS dans les différentes couches de la pile protocolaire des réseaux de capteurs sans fil.

— **Couche physique** : Sur la couche physique, il y a deux types d'attaques :

1. Le brouillage : Cette attaque représente un attaquant qui bloque la réception du canal radio du nœud en émettant sur la bande de fréquences afin de créer une interférence radio.
2. Le trempering : Il implique un accès physique au nœud, afin d'extraire toutes les informations sensibles telles que les clés cryptographiques.

— **Couche liaison** : Dans cette couche, les attaques DoS utilisent le mécanisme d'accès au réseau pour verrouiller le système. Les attaques qui visent cette couche provoquent [38] :

1. La collision : Elle se produit lorsqu'un nœud compromis et ses nœuds voisins transmettent simultanément en utilisant la même fréquence radio. S'il y a collision, une demande de retransmission des données est nécessaire, ce qui pourrait épuiser la batterie.
2. L'attaque par déni de sommeil : Cette attaque tente de pirater le système de gestion de l'énergie des nœuds capteurs pour réduire les opportunités de la transition vers l'état de faible consommation (état de veille).
3. L'attaque replay : Elle survient lorsque des messages échangés entre les nœuds de capteurs sont enregistrés et relayés pour épuiser l'énergie des nœuds de réception.

— **Couche réseau** : Cette couche peut être la cible de diverses attaques telles que :

1. Le renvoi sélectif : Dans ce cas, l'attaquant peut compromettre un nœud, afin qu'il envoie sélectivement des messages reçus de ses nœuds voisins [40]. Dans cette attaque, un nœud malveillant agira comme un nœud normal en transférant des messages, mais il ignore sélectivement certains d'entre eux.
2. L'attaque Sinkhole : Dans cette attaque, un nœud malveillant propose aux nœuds voisins l'utilisation d'une connexion plus puissante sur le chemin le plus court

pour atteindre la station de base [40], [41]. Ainsi, les nœuds voisins choisiront le nœud compromis comme le nœud suivant dans leur chemin de routage pour transmettre leurs données. Par conséquent, l'attaquant peut récupérer et modifier toutes les données reçues par le nœud compromis.

3. Le trou noir : Cette attaque est un cas particulier de l'attaque sinkhole. Elle implique la création d'une fausse station de base et d'une fausse topologie [40], [41]. L'attaquant crée un nœud malveillant dans le réseau, qui vise à modifier les tables de routage pour forcer un maximum de nœuds victimes à transmettre leurs données à partir de celui-ci. Une fois ces données reçues par l'attaquant, elles ne seront jamais retransmises.
4. L'attaque Sybil : Dans cette attaque, un nœud malveillant a plus d'une identité qui peut être générée aléatoirement, ou dupliquée à partir d'une identité légitime qui existe déjà [42]. Le nœud malveillant peut créer un grand nombre d'identités et peut modifier la table de routage des chemins des nœuds voisins.
5. L'attaque Wormhole : Elle est produite par un nœud compromis qui enregistre les paquets et les envoie à un autre nœud malveillant dans le réseau en utilisant une connexion puissante. L'objet de cette attaque est de tromper les nœuds voisins sur les distances réelles [38], [39]. En effet, un nœud victime cherche le chemin le plus court en nombre de sauts pour atteindre ses voisins. Dans ce cas, il sera obligé d'utiliser le chemin reliant les deux nœuds malveillants pour atteindre un endroit éloigné en un seul saut. Cette option conduira les nœuds voisins a passer par des nœuds malveillants pour faire circuler leurs informations.
6. L'attaque par inondation : Afin d'inonder une partie du réseau, un attaquant utilise un signal radio puissant pour envoyer des paquets Hello. L'attaquant induit en erreur de nombreux nœuds, en leur faisant croire qu'ils sont dans son voisinage [40]. Lorsque les nœuds de capteurs reçoivent les paquets Hello, ils tentent de transmettre leurs données au nœud attaquant, ce qui peut être hors de leur portée radio. Le but de cette attaque est de saturer le réseau et de consommer de l'énergie.
7. L'attaque misdirection : Dans cette attaque, un nœud malveillant présent dans le chemin de routage peut envoyer des paquets dans la mauvaise direction, tandis que les nœuds de destination sont inaccessibles. Au lieu d'envoyer des paquets dans la bonne direction, l'attaquant dirige ses paquets vers un nœud victime. Un attaquant pourrait également détourner le trafic du nœud victime [41]. En effet, un nœud malveillant peut également falsifier une adresse source, lors de l'envoi d'une requête, de sorte que la réponse retournée peut être déroutée.

— **Couche transport** : Cette couche est vulnérable à plusieurs attaques telles que :

1. L'attaque de désynchronisation : Elle est produite lorsqu'un attaquant interfère avec la connexion existante entre deux extrêmités. L'attaquant provoque une retransmission des données en envoyant des paquets entre les deux extrêmités de la connexion.
2. Le flooding : Cette attaque peut avoir deux formes, dans la première, un attaquant peut souvent faire des demandes pour établir une connexion jusqu'à ce que les ressources nécessaires à chaque connexion soient épuisées. Par conséquent, les demandes légitimes de connexion seront ignorées. En revanche, dans la seconde, l'attaquant envoie un grand nombre de paquets communs à une seule destination. Ainsi, le trafic réseau sera très intense, ce qui conduit à la non-distinction entre le trafic légitime et malveillant.

— **Couche application** : La couche application peut être ciblée par plusieurs attaques telles que :

1. L'attaque basée sur le chemin : Elle est produite par un attaquant qui injecte des paquets parasites dans le réseau, le long des chemins reliant les différents nœuds légitimes à la station de base [38].
2. La reprogrammation : C'est une attaque, qui peut se produire, par exemple lors de la reprogrammation des nœuds dans le cas d'un redéploiement du réseau. Dans ce cas, l'attaquant peut profiter de cette situation et prendre le contrôle de grandes parties d'un réseau.

1.6.2 Panorama des attaques DoS dans les RCSF

Ces dernières années, la sécurité des réseaux de capteurs sans fil a fait l'objet de nombreuses études dans la littérature [43], [44],[45]. Plusieurs solutions sont proposées pour sécuriser les différents aspects de ces réseaux, par exemple l'agrégation des données, les protocoles de routage et les protocoles de synchronisation de temps. La technique de clustering est généralement utilisée pour la conception des mécanismes de sécurité pour les RCSF [46],[47]. Dans cette section, nous présentons quelques techniques et protocoles proposés pour sécuriser les RCSF hiérarchiques, en particulier, ceux qui ont traité la détection des attaques de Déni de Service.

Perrig et al., ont présenté dans [48] le protocole SPINS qui est l'un des premiers protocoles à avoir fourni la confidentialité et l'authentification des données. SPINS inclu deux blocs de sécurité basés sur des clés symétriques : SNEP [49] et μTESLA. SNEP (Sensor

Network Encryption Protocol) est un protocole de chiffrement des réseaux de capteurs. Ce dernier utilise deux mécanismes de sécurité ; le premier sert à assurer la confidentialité en utilisant l'algorithme de cryptage DES (Data Encryption Standard), tandis que le second est utilisé pour calculer le code d'authentification de messages pour authentifier les données entre les nœuds. Le deuxième bloc μTESLA est une version du protocole TESLA adaptée aux RCSF qui utilise des clés symétriques. Il fournit une diffusion authentifiée, en utilisant des chaînes unidirectionnelles, construites avec des fonctions de hachage sécurisées. Il divise le temps en intervalles, que l'expéditeur associe à chaque clé de la chaîne pour effectuer des opérations cryptographiques. Toutefois, μTESLA prévoit l'envoi permanent des données aux capteurs, ce qui consomme plus d'énergie.

A noter que SPINS présente certaines limites, telles que la contrainte temps réel, car le processus de partage du code MAC accroit les délais de communication. Par exemple, dans μTESLA, le nœud récepteur doit attendre un certain temps avant d'authentifier les messages reçus. En outre, la contrainte liée à la diffusion de messages authentifiés par des nœuds de capteurs ne peut pas se faire sans l'aide de la station de base. Enfin, une limite importante du protocole SPINS est sa consommation d'énergie, qui est très élevée en raison des exigences de μTESLA comme mentionné ci-dessus.

TinySEC [50] est une architecture de sécurité de la couche liaison pour les réseaux de capteurs sans fil, implémentée dans le noyau de TinyOS [28]. TinySEC fournit l'authentification, l'intégrité et la confidentialité des messages. Il suppose que chaque nœud partage initialement avec la station de base une clé secrète, qui est utilisée pour dériver des clés de cryptage et d'authentification. TinySEC fournit deux services de sécurité, l'authentification avec (TinySec-Auth) où les paquets de données sont envoyés avec uniquement une authentification, sans chiffrement. Ce service, assure que les données reçues proviennent d'un nœud non compromis. Le service (TinySec-AE) assure le cryptage des messages échangés sur le réseau. En détectant les paquets non autorisés qui sont propagés dans le réseau de capteurs, TinySec évite la perte d'énergie causée par la transmission de ces paquets. Néanmoins, TinySec ne résout pas bien le problème de la sécurisation des données. Il fournit certainement l'authentification et le cryptage des paquets, mais présente certaines limitations : il n'assure pas une protèction contre les messages de relecture et ne fournit pas de protection spéciale contre les attaques de consommation de ressources.

Dans [51], les auteurs proposent le protocole Localized Encryption and Authentication Protocol (LEAP) qui est un protocole localisé de chiffrement et d'authentification gèrant des clés utilisées pour sécuriser les messages échangés dans les RCSF. Ce protocole est basé sur une approche de distribution de clé hybride. En effet, il utilise quatre types de clés différentes : une clé individuelle unique pour chaque nœud permettant de sécuriser les communications avec la SB ; une clé de groupe, qui est générée par la SB et diffusée dans tout le réseau pour sécuriser les communications entre la SB et tous les nœuds du réseau ; une clé du cluster, qui est utilisée pour sécuriser les communications entre le Cluster Head (CH) et tous ses nœuds membres, enfin, une clé partagée par paires, qui permet de sécuriser les données échangées entre les nœuds voisins. En utilisant plusieurs types de clés, le protocole LEAP garantit un haut niveau de sécurité. Néanmoins, la faiblesse de ce protocole réside dans la protection des clés initiales. La capture d'un nœud défaillant par un attaquant peut compromettre la clé initiale à tout moment, ce qui permet à cet attaquant de déduire toutes les paires de clés du réseau. Dans ce cas, l'attaquant peut toujours lancer une attaque de retransmission sélective, par exemple, sans être détecté. De plus, le protocole LEAP ne supporte pas la mobilité.

TESLA (Timed Efficient Stream Loss-tolerant Authentification) est un protocole de diffusion [52] qui utilise des propriétés temporelles pour assurer l'authentification. Il permet une gestion optimale de la consommation des ressources (temps de calcul du processeur) et réduit également les messages de congestion dans les réseaux. TESLA est l'un des protocoles qui permet d'optimiser la consommation d'énergie dans les réseaux de capteurs sans fil.

Aussi de nombreux chercheurs ont mené des études pour développer des solutions pour prévenir les attaques DoS dans les réseaux de capteurs avec des topologies spécifiques, particulièrement les topologies hiérarchiques.

Oliveira et al. proposent dans [53], le protocole SecLEACH qui est une extension de l'algorithme de clustering LEACH. Il utilise un mécanisme de pré-distribution de clé aléatoire avec une cryptographie à clé symétrique pour sécuriser les communications dans un réseau hiérarchique avec construction dynamique de clusters. SecLEACH prend en charge les exigences de la sécurité en termes d'authentification et de confidentialité des données. Dans SecLEACH, le nœud puits authentifie les nœuds CHs, et les CHs authentifient également les capteurs qui se joignent à eux. SecLEACH fournit l'authentification, la confidentialité et la fraîcheur des données et permet une bonne gestion de l'utilisation de la mémoire et de la consommation d'énergie. Néanmoins, d'une part, il ne peut pas prévenir des attaquants

internes des nœuds qui se déclarent CH et rejoignant d'autres CHs, seuls les attaquants externes sont identifiés [54]. D'autre part, il n'est pas recommandé pour les réseaux qui présentent des contraintes [55].

Kumar et al. présentent dans [56] une approche permettant de protéger les réseaux de capteurs des attaques DoS. La solution proposée s'opère sur deux étapes. La première est basée sur la construction des clusters à l'aide de l'algorithme HEED [57], alors que la deuxième étape utilise un serveur de distribution de clés (KDS) qui fournit les clés de session et des identifiants uniques pour chaque nœud du réseau. Après avoir formé les clusters et avant le déploiement des nœuds, chaque nœud met en place une clé secrète spéciale (Ks) et un identifiant unique (IDi) avec le serveur KDS. La coordination intra-cluster et la communication inter-cluster sont contrôlées par les cluster-head.

Lai et al ont proposé une nouvelle approche pour protéger les RCSF hiérarchiques contre les attaques DoS [58]. Les auteurs ont proposé d'élire des nœuds spéciaux appelés Cnodes pour surveiller le trafic dans ces réseaux. Dans un tel schéma, la partition du réseau en clusters fournit deux types de capteurs ; les nœuds normaux, qui collectent et transmettent des données, et les nœuds cluster-head (CH), qui agrègent et envoient les données à la SB. Pour surveiller et contrôler le trafic entrant et sortant des clusters, les auteurs proposent un autre type de capteurs nommés "Cnodes". Si un Cnode constate qu'un nœud envoie à son CH un nombre de paquets de données supérieur à un seuil donné, il considère ce nœud comme suspect ou compromis. Chaque Cnode ayant détecté un nœud compromis envoie un message d'avertissement au CH, qui commence immédiatement à ignorer tous les messages entrants à partir de ce nœud et prévient ses nœuds membres.

Hammal et al ont proposé dans [59] un mécanisme itératif pour élire les Cnodes en utilisant un processus d'élection démocratique basé sur les énergies résiduelles des nœuds. Ce mécanisme s'opère en deux phases : Une fois les clusters formés, la phase d'initialisation commence et chaque nœud envoie la valeur de son énergie résiduelle à son CH, et chaque nœud agit comme un Cnode et commence à contrôler ses voisins et continue à transmettre ses données collectées en même temps. La phase de l'élection démocratique commence à la fin d'une période donnée "d", où le CH demande à chaque nœud d'envoyer sa valeur d'énergie résiduelle et demande à chaque Cnode d'envoyer ses observations sur les taux de transmission de ses voisins. Le CH calcule l'énergie consommée pour chaque nœud "i" en considérant les taux moyens observés par les Cnodes pendant la période "d". Le CH compare la différence entre l'énergie consommée "annoncée" pour la période "d" et celle de la période précédente. Lorsque cette différence est inférieure à un seuil accepté, le nœud

"i" est déclaré "légitime" et il est ajouté à la liste des nœuds éligibles pour être sélectionné Cnode ; Sinon, le nœud est placé dans la liste des nœuds suspects. Il a été noté que dans cette méthode, les nœuds avec peu de voisins peuvent avoir moins de chances de devenir Cnodes parce qu'ils ont moins de voisins qui votent pour eux. Néanmoins, il n'est pas utile de les sélectionner comme Cnodes dans les deux cas : s'ils ont quelques voisins à surveiller et si l'espace géographique où ils se trouvent n'est pas couvert par des nœuds capteurs.

Une conception de sécurité adaptative (SecCBSN) pour sécuriser la communication dans les RCSF composés de clusters, est proposée dans [60]. Elle se compose de trois modules, qui peuvent détecter les nœuds malveillants en fournissant des protocoles sécurisés de communication et d'authentification entre les nœuds. Dans chaque cluster, un CH planifie les transmissions de ses nœuds capteurs et les surveille périodiquement. Le module de sécurité principal utilise le certificat TESLA (TCert) pour permettre aux nœuds existants d'authentifier de nouveaux nœuds entrants, déclenchant l'établissement de liens sécurisés et l'authentification de la diffusion entre les nœuds voisins. Dans SecCBSN, le module de détection d'intrusion bloque les nœuds compromis.

1.7 Conclusion

Dans les applications spécifiques aux réseaux de capteurs, les capteurs sont déployés dans des environnements généralement hostiles, difficiles d'accès. En cas d'épuisement de leur énergie, on ne peut pas leur changer de batterie. Aussi, ils doivent s'auto-organisés pour transmettre leur données recueillies à la station de base, ils doivent aussi être tolérants aux pannes pour permettre un fonctionnement sans aucune interruption. Ils doivent avoir une architecture simple et souple et faire face aux erreurs liées à la communication sans fil, satisfaire les critères liés à la qualité de service (maximum de la bande passante, un minimum de taux de pertes de paquets, etc.).

En outre, le déploiement de ces réseaux dans des fréquences radio ouvertes, rend l'écoute-espionne des communications assez faciles. Par conséquent, ils doivent considérer l'implémentation du service de sécurité pour assurer l'authentification, la confidentialité et l'intégrité. Dans le même ordre d'idée, sachant que la technique du clustering est l'une des méthodes utilisées pour économiser l'énergie dans les RCSF, dans le chapitre suivant, nous étudions l'intérêt de l'utilisation de cette technique à travers une description détaillée, et une évaluation des différents algorithmes utilisés dans ces réseaux. Aussi, nous présentons notre contribution dans ce contexte, par la proposition d'une nouvelle méthode de détection des attaques DoS basée sur le clustering.

Chapitre 2

METHODES DE DETECTION DES ATTAQUES DoS BASEES SUR LE CLUSTERING

Sommaire

2.1 Contexte et objectifs

Plusieurs problématiques dans les réseaux de capteurs ont été étudiées dans la littérature, telles que la sécurité, la tolérance aux pannes et la distribution de la charge d'énergie. Pour résoudre ces problèmes, les solutions existantes incluent souvent l'utilisation d'un algorithme de clustering.

Le clustering est une technique qui consiste à diviser selon un certain critère, un réseau de capteurs en un ensemble de petits groupes virtuels composé de plusieurs nœuds adjacents. Dans la plupart des cas, les groupes sont formés de telle sorte que la similitude entre les éléments d'un même groupe est maximale, tandis que la similitude entre les éléments provenant de différents groupes est minimale. Les groupes ainsi formés sont appelés "clusters" où chaque cluster est composé d'un coordinateur nommé "Cluster Head" (CH) et d'un ensemble de nœuds membres. La création des clusters assigne aux CHs des tâches spéciales telles que la collecte des données à partir de tous les nœuds du groupe, l'envoi des données à la station de base en utilisant des chemins directs ou des chemins multi-sauts. Les nœuds de chaque cluster peuvent communiquer entre eux, ou communiquer avec le CH.

L'objectif principal du clustering est de maintenir efficacement l'utilisation de l'énergie des nœuds de capteurs, en les impliquant dans des communications multi-sauts dans un groupe particulier. La formation des clusters est généralement basée sur l'énergie résiduelle des capteurs et la proximité des nœuds aux CHs. Le clustering joue un rôle important pour l'économie de l'énergie dans les réseaux de capteurs. Avec le regroupement des RCSF en clusters, la consommation de l'énergie, la durée de vie du réseau et l'évolutivité peuvent être améliorées.

Dans ce chapitre, nous présentons un aperçu sur les algorithmes de clustering spécifiques aux RCSF, proposés au cours des dernières décennies. Ensuite, nous abordons les méthodes de clustering appelées FFUCA et LEACH. En considérant la mesure de performance qui représente la consommation de l'énergie, une évaluation des performances des deux méthodes à travers une comparaison des résultats numériques obtenues de la simulation sera également présentée [61],[62]. Par la suite, nous présentons notre contribution qui représente une nouvelle méthode de détection des attaques DoS basée sur le clustering : les méthodes récursives FFUCA et LEACH [63],[64]. Enfin, nous présentons la comparaison des résultats numériques de la simulation des deux méthodes par rapport à la consommation de l'énergie des nœuds de capteurs.

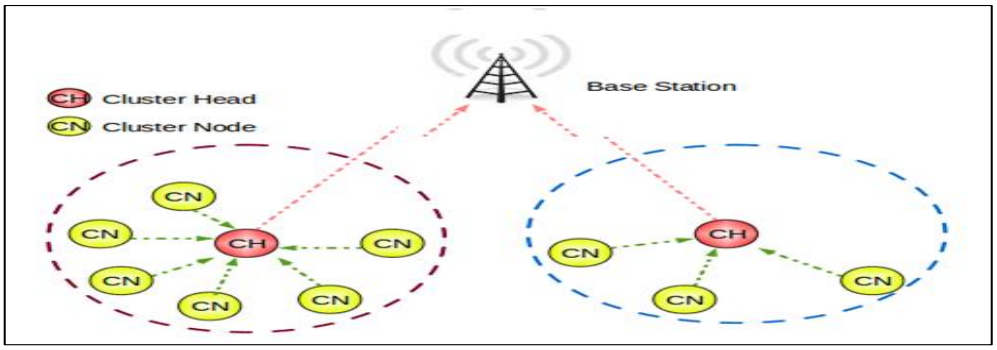


FIGURE *2.1* – Exemple de clustering

2.2 Aperçus des algorithmes de clustering dans les réseaux de capteurs

2.2.1 Energy Efficient Hierarchical Clustering (EEHC)

Kumar et al. ont proposé dans [65], Energy Efficient Hierarchical Clustering qui est un algorithme de clustering hiérarchique permettant de minimiser la consommation de l'énergie dans les RCSF. Initialement, chaque nœud de capteur est élu CH avec une probabilité "p" et s'annonce comme un CH volontaire à tous les nœuds voisins ayant la même portée radio. Cette annonce est diffusée à tous les nœuds qui se trouvent dans une distance de k-sauts du CH. Ainsi chaque nœud voisin du CH correspondant peut directement ou à travers des nœuds intermédiaires recevoir un tel message d'annonce et devient membre du cluster le plus proche. Par contre, s'il ne parvient pas à atteindre un cluster, il se considère CH (forcé). Si les messages d'annonce ne parviennent pas à un nœud dans un délai préétabli, alors le nœud devient un CH "forcé". L'algorithme EEHC est basé sur une structure de clustering multi-niveaux. Le processus initial de clustering est répété récursivement au niveau des CH pour permettre de construire de multiples niveaux de la hiérarchie des clusters.

En supposant qu'une hiérarchie à k-niveaux a été construite de cette manière, l'algorithme assure l'efficacité de la hiérarchie à h-niveaux (exemple, h est le niveau de hiérarchie le plus élevé) dans les communications entre les nœuds de capteurs et la station de base de la manière suivante : les nœuds de capteurs membres transmettent leurs données aux CHs correspondant au premier niveau (niveau 1). Ensuite, les CHs du premier niveau transmettent les données agrégées aux CHs du deuxième niveau et ainsi de suite jusqu'à ce que le dernier niveau de la hiérarchie de clustering (niveau h) soit atteint. Les CHs de ce dernier niveau transmettent les données agrégées à la station de base (SB). La consommation d'énergie relative à la collecte, l'agrégation et la transmission de données à la SB dépend

évidemment des paramètres "p" et "k" de l'algorithme. Les auteurs ont démontré à travers des simulations que la consommation d'énergie dans le réseau peut être considérablement réduite en utilisant ces paramètres optimaux et en faisant appel à plusieurs niveaux de la hiérarchie.

2.2.2 Hybrid Energy Efficient Distributed (HEED) Clustering

Hybrid Energy Efficient Distributed (HEED) Clustering est un algorithme de clustering multi-sauts proposé par Younes et Fahmy dans [57], l'objectif de HEED est d'assurer :

- La distribution de la consommation d'énergie pour prolonger la durée de vie du réseau.
- La minimisation de l'énergie pendant la phase de sélection des CHs.
- La réduction des coûts de contrôle du réseau.

HEED fournit une distribution uniforme de CH à travers le réseau et un meilleur équilibrage de charge. Les CHs sont sélectionnés périodiquement en fonction de deux paramètres : l'énergie résiduelle et le coût de la communication intra-cluster. L'énergie résiduelle de chaque nœud est utilisée pour sélectionner l'ensemble initial des CHs, en se basant sur des méthodes probabilistes. Ainsi, seuls les capteurs qui ont une énergie résiduelle élevée peuvent devenir des nœuds CHs. Le coût de la communication qui dépend des propriétés des clusters telles que la taille du cluster et aussi la portée de transmission sert à résoudre les conflits, lorsqu'un nœud se trouve à la portée de plusieurs CHs en même temps. Après la formation des clusters, les CHs attribuent des intervalles de temps aux membres de leurs clusters pendant lesquels ils peuvent transmettre.

2.2.3 Power-Efficient Gathering Sensor Information Systems (PEGASIS)

Dans [66], Lindsey et al. ont proposé le protocole PEGASIS qui a un double objectif. Le premier vise à prolonger la durée de vie d'un réseau en assurant une consommation d'énergie uniforme sur l'ensemble de ses nœuds. Le second vise à diminuer les retards. Le modèle de réseau considéré par PEGASIS suppose un ensemble homogène de nœuds déployés dans une zône géographique, où les nœuds sont supposés avoir une connaissance globale sur les positions des autres nœuds. L'objectif de ce protocole est de développer une structure de routage et un système d'agrégation pour réduire la consommation d'énergie et de livrer les données agrégées à la station de base avec un délai minimal, tout en équilibrant la consommation de l'énergie entre les nœuds de capteurs.

L'avantage principal de PEGASIS réside dans la diminution de la consommation en

énergie dûe au clustering. Néanmoins, il présente deux principales limites : d'une part, il exige toujours un ajustement dynamique de la topologie, ce qui pourrait causer un surcoût important. D'autre part, il suppose que tout les nœuds utilisent une liaison directe avec la station de base qui gère la topologie d'une manière centralisée. Or, cette supposition est loin de la réalité, car les capteurs communiquent généralement en mode multi-sauts pour atteindre la station de base.

2.2.4 Linked Cluster Algorithm (LCA)

L'algorithme présenté dans [67] est l'un des premiers algorithmes de clustering qui a été initialement développé pour les réseaux filaires. C'est un algorithme statique ayant pour objectif la maximisation de la connectivité du réseau. En effet, tous les nœuds du réseau sont organisés en un ensemble de clusters et chaque nœud appartient à au moins un cluster. Chaque cluster a son propre CH qui agit comme un contrôleur local des autres nœuds du cluster. Les clusters sont reliés via des nœuds passerelles pour connecter les clusters voisins et pour fournir une connectivité globale du réseau.

LCA est un algorithme à un seul saut, basé sur les identifiants (ID) des nœuds, où chaque nœud se déclare CH ou non en se basant sur son identifiant et ceux de ses voisins. Le nœud possédant le plus petit identifiant dans son voisinage à un saut, se déclare comme CH et ses voisins à un saut dont les identifiants sont supérieurs à celui du CH le joignent et deviennent des nœuds membres de ce cluster. Après la formation des clusters, et la détermination des CHs et des membres, si un nœud a parmi ses voisins à un saut plus qu'un CH, il se déclarera comme nœud passerelle. L'inconvénient majeur de cet algorithme réside dans son faible rendement en clusters, lorsque les nœuds sont disposés dans l'ordre de leurs identités, ou aucun nœud ne peut se déclarer CH. Aussi, sachant que les identifiants des nœuds ne changent pas, les nœuds ayant un petit identifiant sont les plus probables à être choisi comme CH, que les nœuds ayant un grand identifiant. Par conséquent, les nœuds CHs sont choisis pour une longue durée, ce qui induit une consommation rapide de leur énergie et par conséquent, ils provoquent des blocages dans leurs clusters. Une autre limitation de LCA est le coût relativement élevé des messages de contrôle que les CHs doivent utiliser pour diffuser leurs listes de nœuds. En outre, LCA ne considère pas la mobilité des nœuds et la maintenance de la structure des clusters s'avère coûteuse, car le mouvement d'un nœud peut engendrer des réactions en chaîne et nécessite la reconstruction totale de la structure. Enfin, les auteurs de cet algorithme n'ont pas traité le volet de l'efficacité énergétique, un des paramètres clés des réseaux de capteurs à prendre en considération.

2.2.5 Weighted Clustering Algorithm (WCA)

L'algorithme WCA [68] est un algorithme basé sur les poids des nœuds, qui à la différence des algorithmes basés sur des critères uniques pour la formation des clusters, combine plusieurs métriques telles que, la puissance de la transmission, la mobilité et l'énergie résiduelle des nœuds. Pour la construction des clusters, cet algorithme associe un poids à chaque nœud. Ce poids est représenté par une somme pondérée des différentes métriques impliquées dans son calcul. Sachant que dans les réseaux de capteurs, l'énergie est une ressource précieuse, il est nécessaire d'associer à la métrique énergie résiduelle, un coefficient de pondération très élevé. Chaque nœud calcul le poids combiné et le diffuse. Le nœud avec le plus petit poids dans son voisinage est choisi comme un CH.

Bien que cet algorithme présente de meilleures performances en termes de nombre de clusters produits et de nombre de changements de CHs, il présente un inconvénient majeur par rapport à la consommation de l'énergie induite pour connaitre le poids de tous les nœuds avant de commencer le processus du clustering, les capteurs épuisent, ainsi rapidement leurs batteries.

2.2.6 Low Energy Adaptive Clustering Hierarchy Protocol (LEACH)

Le protocole LEACH est l'un des protocoles de clustering le plus utilisé dans les réseaux de capteurs. Proposé par Heinzelman et al. dans [69], [70], ce protocole a pour but de prolonger la durée de vie du réseau en réduisant la consommation de l'énergie.

LEACH s'exécute sur plusieurs itérations où chaque itération représente l'intervalle entre la formation consécutive de deux clusters. Chaque itération est réalisée en deux phases qui sont réspectivement la phase de mise en place (set-up phase) et la phase de stabilité (steady-state phase).

1. La phase de mise en place (Election de CH et formation de clusters) : Au cours de cette phase, le processus d'élection des CHs est déclenché. Chaque nœud peut choisir par lui-même d'être promu CH ou non. Il choisit un nombre aléatoire entre [0, 1]. Si ce nombre est inférieur à un seuil T, alors il devient CH. Sinon, il joint le CH le plus proche.
2. La phase de stabilité : Durant cette phase, chaque nœud exécute les opérations suivantes :
   — Transmettre ses données à l'aide du protocole d'accès multiple à répartition de temps (TDMA est utilisé pour assurer la transmission des paquets sans collision) [69].
   — Couper sa transmission pour se mettre en état de veille.

De ce fait, aprés avoir effectué l'agrégation des données, ces dernières sont transmises à la station de base via le CH correspondant.

Principe de fonctionnement

Soit "P" le pourcentage moyen des clusters que nous voulons former pour notre réseau à un instant t. LEACH est composé de cycles de 1/p itérations.

Chaque itération "r" est organisée comme suit :

1. Chaque nœud n_i
   - calcule le seuil $T(i)$ tel que :

$$T(i) = \begin{cases} \dfrac{P}{1 - P \cdot \left(r \bmod \frac{1}{P}\right)} & \text{si } i \text{ n'est pas encore élu CH} \\ 0 & \text{si } i \text{ est élu déjà CH} \end{cases}$$

   - Choisir un nombre aléatoire x_i tel que : $0 \leq x_i \leq 1$.
   - Si $x_i \leq T(i)$ alors le nœud $'i'$ s'auto désigne comme CH pour l'itération en cours. $T(i)$ est calculé de sorte que chaque nœud devient CH une fois dans chaque cycle de $1/p$ itération : $T(i) = p$ lorsque $r = (1/p) - 1$.
2. Le CH auto-désigné informe les autres nœuds de sa décision en envoyant un message en broadcast, avec la même puissance de transmission (en utilisant le protocole Carrier Sense Multiple Access, CSMA MAC).
3. Le reste des nœuds choisissent de rejoindre le CH ayant l'intensité du signal la plus forte (signal reçu par les nœuds restants). Ils envoient un message de réponse pour informer le CH correspondant (en utilisant le même protocole que dans la précédente itération, CSMA MAC).
4. Les CHs effectuent un ordre de transmission (Time Division Multiple Access, TDMA) à tous les nœuds ayant choisi de les rejoindre et informent chaque nœud à quel moment ils envoient leurs données.
5. Les CHs restent à l'écoute et les nœuds membres effectuent des mesures de leur environnement et les envoient à leurs CHs respectifs. Quand ce n'est pas leur tour d'envoyer les données, ils restent en veille pour conserver leur énergie (les collisions de transmissions entre les nœuds des différents clusters sont réduites grâce à l'utilisation du protocole Code Division Multiple Access, CDMA).
6. Les CHs agrègent, et éventuellement compressent les données et les envoient à la station de base en une seule transmission. Cette transmission peut être directe, ou en multi-sauts, si elle est relayée par d'autres CHs.

7. Les étapes 5 et 6 sont répétées jusqu'à la dernière itération.

Il est à noter que chaque nœud décide de se désigner lui-même comme un CH ou non. Sa décision ne prend pas en compte le comportement des nœuds environnants. Pour cette raison, nous pouvons éventuellement avoir, pour une itération donnée, un certain nombre de CH très différent du pourcentage "p" sélectionné. En outre, tous les CHs élus peuvent être situés dans la même région du réseau, laissant des zônes "découvertes". Dans ce cas, nous pouvons seulement espérer que la répartition spatiale sera meilleure lors de la prochaine itération.

2.3 Application de l'algorithme Fast and Flexible Unsupervised Clustering Algorithm (FFUCA) dans les réseaux de capteurs

Fast and Flexible Unsupervised Clustering Algorithm (FFUCA) [71] est un algorithme basé sur les propriétés induites des espaces ultramétriques. Ces derniers sont des espaces ordonnés tels que les données d'un même cluster se trouvent à la même distance des données d'un autre cluster (par exemple en généalogie : deux espèces d'une même famille (frères) sont à équidistance de leur cousin qui appartient à une famille différente).

Les propriétés ultramétriques permettent d'éviter le calcul de l'intégralité des similarités (dissimilarités) mutuelles entre les éléments à traiter.

L'idée de la méthode FFUCA est de déduire, à partir d'un échantillon de taille "m" choisi uniformément de façon aléatoire, le comportement de l'ensemble des données à traiter par rapport à la distance utilisée. Pour ce faire, un espace ultramétrique est créé à partir de l'échantillon. Ensuite, le reste de l'ensemble est partitionné en utilisant le comportement déduit.

Dans la section suivante, nous proposons l'application de FFUCA dans les réseaux de capteurs, en définissant une judicieuse distance (métrique) entre les capteurs des RCSF [61],[62]. L'utilisation de l'algorithme FFUCA dans ce cas, permet de fournir une disposition optimale des nœuds, par rapport à la distance utilisée. Cette dernière est basée sur l'énergie nécessaire à une communication entre deux nœuds.

L'objectif est donc de disposer les nœuds de façon à ce que l'énergie consommée entre les nœuds soit la plus minimale possible, ce qui permet une durée de vie plus élevée.

Description de la méthode

Le processus de clustering est important dans l'organisation des RCSF. En effet, il affecte considérablement les performances du réseau, lorsqu'il s'agit d'utiliser une distance basée sur l'énergie nécessaire à la communication entre deux nœuds. La viabilité du réseau est directement liée à la disposition des nœuds constitués par le clustering. Ainsi, les approches de clustering devrait tenir compte des conditions suivantes [72] :

- L'énergie et la durée de vie du réseau limitées : l'énergie du capteur dans un réseau est limitée. La bonne gestion de cette ressource est vitale pour la durée de vie du réseau.
- La capacité du réseau : la performance des RCSF est liée à une organisation optimale des nœuds de capteurs et des CHs, ce qui améliore leur capacité en diminuant leur consommation d'énergie.
- La dépendance de l'application : le système de clustering doit être souple à la variété des besoins de l'application.

En considérant ces exigences, nous avons proposé l'application de la méthode de clustering FFUCA (Fast and Flexible Unsupervised Clustering Method) [71] dans les RCSF.

L'application de FFUCA permet (en matière de consommation d'énergie) une organisation rapide et optimale des clusters, des cluster-heads et des nœuds capteurs dans le réseau. La structure résultante est basée essentiellement sur une mesure de la consommation d'énergie d_e, ce qui assure une bonne économie d'énergie. Ainsi, il augmente la durée de vie du réseau et améliore ses performances. FFUCA est flexible aux données traitées (nœuds, énergie,...) et les paramètres de l'utilisateur (taille de l'échantillon ainsi que les seuils critiques de voisinage).

L'idée générale de la méthode FFCUA est basée sur les étapes suivantes :

1. A partir d'un échantillon de nœuds, déduire le comportement de l'ensemble des capteurs avec la mesure de la consommation d'énergie "d_e". L'échantillon est choisi aléatoirement (la taille de l'échantillon est donnée par l'utilisateur). Déduire le comportement consiste à construire un espace ultramétrique structuré de manière hiérarchique [71], [73].
2. Le comportement comprend les granularités des clusters et des groupes denses de capteurs (tailles des clusters).
3. L'intégration du reste des capteurs dans les clusters (sauf ceux choisis comme échantillon) se fait selon les informations de l'étape 1 (c'est-à-dire clusters et seuils).

Principe de fonctionnement

Considérons un RCSF de taille "n", doté d'une mesure de la consommation d'énergie "d_e" qui satisfait les propriétés de distance (métrique) pour trois capteurs $s_1, s_2, s_3 \in n$ (voir Figure 2.2).

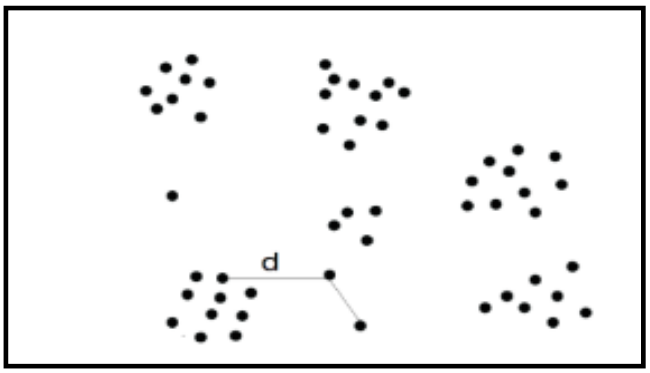


FIGURE *2.2* – Exemple d'espace métrique, un RCSF décrit par une distance d_e.

1. Symétrie $d(s_1, s_2) = d(s_2, s_1)$. L'énergie requise pour la transmission d'un bit de données de s_1 à s_2, est la même que pour la transmission d'un bit de s_2 à s_1.
2. Précision positive $d(s_1, s_2) \geqslant 0$, et $d_e(s_1, s_2) = 0 \Leftrightarrow s_1 = s_2$. L'énergie requise pour la transmission d'un bit de données est $\geqslant 0$. Elle est nulle pour la transmission d'un bit de données de s_1 à s_1.

3. Inégalité triangulaire $d(s_1, s_2) \leqslant d(s_1, s_3) + d(s_3, s_2)$. L'énergie requise pour la transmission d'un bit de données de s_1 à s_2 est inférieure ou égale à la somme de l'énergie nécessaire à la transmission d'un bit de données de s_1 à s_3 et de s_3 à s_2.

La méthode FFUCA est composée des étapes suivantes :

- **Etape 1** : Choisir au hasard un échantillon de nœud de taille "m" à partir du réseau "n" (exemple, $m = n/50$ nœuds si $n = 1000$) (voir la Figure 2.3).
  Remarque 1 : Le choix de l'échantillon des nœuds dépend du réseau traité et de l'expertise de son déploiement.
  Remarque 2 : Le nombre "m" de nœuds dans l'échantillon dépend du nombre de nœuds "n" dans l'ensemble du RCSF comparativement aux limites de la distance "d_e" [74].

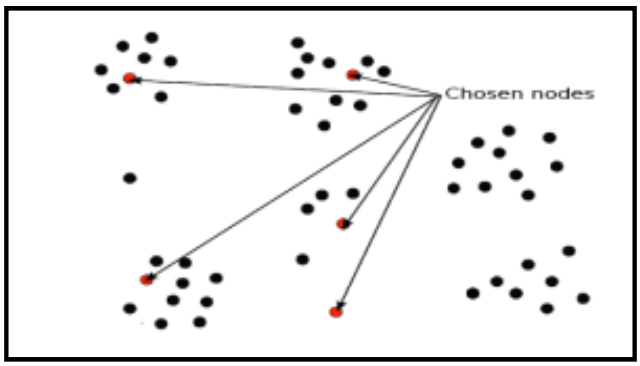


FIGURE *2.3* – Choix avec m = 20 nœuds.

Exemple 1 : Considérons un RCSF de taille $n = 100000$ nœuds et la distance $d_e \in [0, 0.5]$, si nous utilisons un échantillon de 50 nœuds, nous pouvons avoir une idée de l'organisation optimale des nœuds par rapport à la consommation d'énergie. Mais si $n = 500$ et "d_e" $\in [0, 300]$, le choix d'un échantillon de 5 nœuds ne fournit pas suffisamment d'informations sur la manière dont les nœuds "n" se comportent avec la distance "d_e".

- **Etape 2** : Exécuter un algorithme de clustering hiérarchique classique (WPGMA ou UPGMA, deux algorithmes connus pour la construction des espaces ultrametrique) sur l'échantillon de nœuds choisi en utilisant la distance "d_e".
- **Etape 3** : Représenter les distances dans le dendrogramme résultant, ainsi l'espace ultramétrique est construit (voir Figure 2.4).

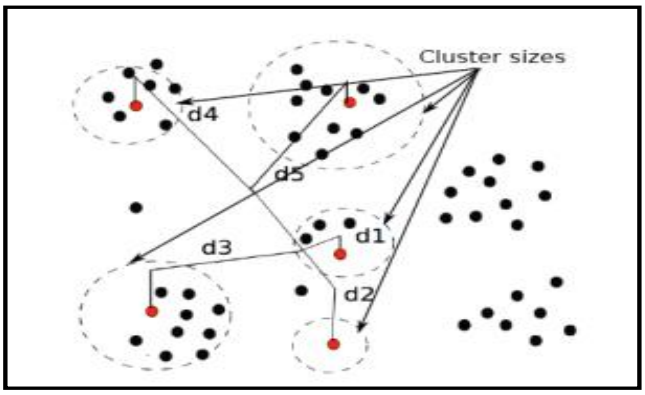


FIGURE *2.4* – Choix des tailles des clusters, représentés par des cercles.

- **Etape 4** : Déduire les intervalles de clusters (seuils) qui dépendent de la proximité recherchée et la distance "d_e". En fonction des proximités entre les nœuds, les clusters peuvent être fins ou larges (voir Figure 2.5).
  Remarque 3 : Puisqu'un espace ultramétrique est utilisé, les clusters seront distingués par des seuils.

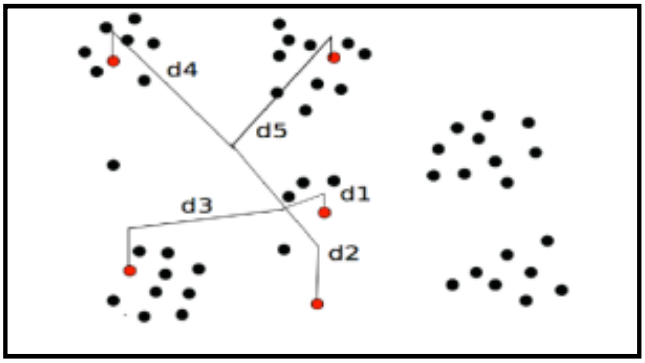


FIGURE *2.5* – La construction d'un espace ultramétrique à partir d'un échantillon de nœuds.

- **Etape 5** : Choisir d'une manière uniforme et aléatoire un représentant (nœud) par cluster à partir du résultat de la dernière étape (voir la Figure 2.6).
  Remarque 4 : Dans cet exemple, comme le nombre de nœuds est petit, nous avons utilisé les mêmes nœuds choisis pour l'échantillon comme des nœuds représentants de clusters.

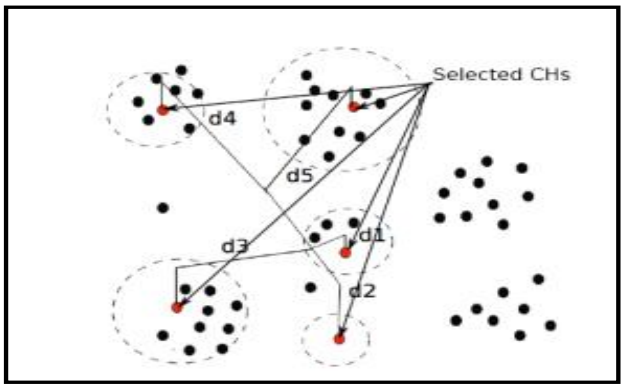


FIGURE *2.6* – Choix des CHs.

- **Etape 6** : Choisir le reste des nœuds un par un et les comparer (selon la distance "d_e") avec les nœuds du cluster représentant l'étape précédente tel que :
  — Si la comparaison des données est proche d'un (ou plusieurs) nœud(s) par rapport aux seuils définis, alors un nœud est ajouté au même cluster (s).
  — Sinon, créer un nouveau cluster, qui sera représenté par le nœud distant (voir la Figure 2.7).

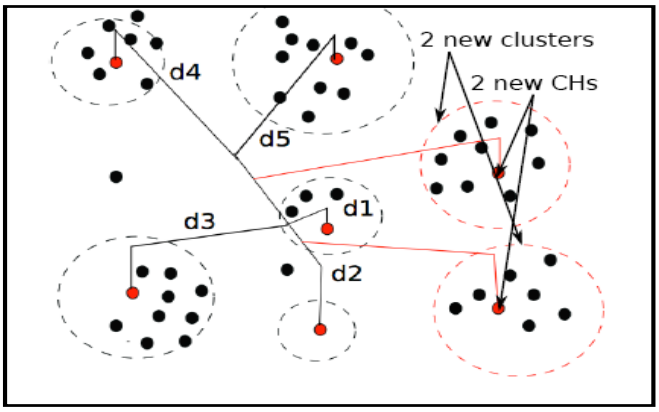


FIGURE *2.7* – L'agrégation de nœuds et émergence de nouveaux clusters

Remarque 5 : Si la comparaison des données est à proximité de plus d'un représentant, le nœud sera alors ajouté à plus d'un cluster et, par conséquent, cela permet de générer un chevauchement de clustering (voir [75], [76], pour plus de détails sur les chevauchements de clustering).
Remarque 6 : Le choix aléatoire de point initial n'affecte pas les clusters résultants [71], [73].

Remarque 7 : Les valeurs aberrantes sont plus éloignées des nœuds, elles sont considérées comme des bruits en général. FFUCA est sensible à ce genre de nœuds [71] (voir Figure 2.8).

Le coût de calcul de la méthode FFUCA est dans le cas moyen, égal à $O(n)$, et dans les rares cas défavorables, égal à $O(n^2)$. Le coût de l'échantillon des nœuds dans le clustering hiérarchique est négligé parce qu'il est sans importance par rapport à l'ensemble du nombre de nœuds dans le réseau traité. La complexité de FFUCA reste la même, même si la taille du réseau augmente, donc il peut traiter les RCSF dynamiques [71].

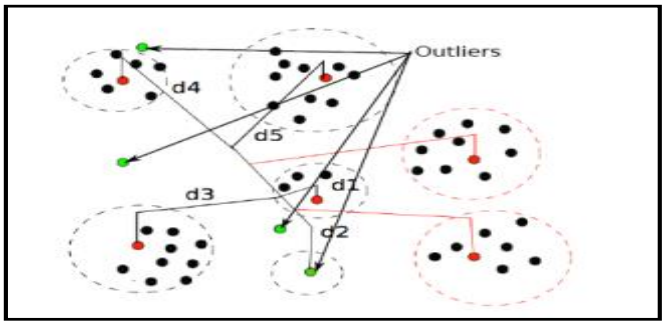


FIGURE *2.8* – Détection des valeurs aberrantes

Algorithme : FFUCA appliqué dans les RCSF

• Variables :
Espace métrique :
Réseau de capteurs sans fil de n nœuds ;
Mesure de la consommation d'énergie (métrique) d_e.

• Début :

1. choisir de façon aléatoire un échantillon de nœuds de taille m ;
2. créer un espace ultramétrique à partir de l'échantillon des nœuds, à l'aide de l'algorithme de clustering hiérarchique WPGMA (ou UPGMA) ; //complexité $O(m^2)$ // Ou m représente la taille de l'échantillon des nœuds.
3. déterminer les intervalles à partir de la hiérarchie de l'étape 2 ;
4. choisir uniformément de façon aléatoire un représentant par cluster, à partir des clusters de l'étape 3 ;
5. **Pour** i < n−k; i++ ; **faire** ou //k représente le nombre de nœuds représentant

```
Pour j<k; j++ ; faire
        Calculer d(i, j) ; // complexité dans le pire cas O(n²)
    Si d(i, j) < intervalle du cluster de j  Alors
        allouer i au cluster de j ;
      Si i ∈ plus d'un cluster Alors
        garder ce dernier que dans le cluster du plus proche représentant ;
          // un nœud pourrait appartenir à deux clusters ssi il est à équidistance
            des deux plus proches représentants
      Sinon créer un nouveau cluster qui sera représenté par cet élément i éloigné
          de tous les autres.
      Fin
   Sinon créer un nouveau cluster qui contiendra et sera représenté par ce
        nœud i éloigné de tous les autres.
  Fin
 Fin
Fin
```

- Proposition d'une comparaison entre LEACH et FFUCA

Nous avons effectué des simulations sur Matlab pour comparer la méthode FFUCA avec LEACH sur un RCSF de 100 capteurs. Nous avons concentré notre travail sur le déploiement des nœuds, notamment sur le regroupement (clustering) optimal des nœuds en matière de consommation d'énergie. Pour mesurer la consommation d'énergie entre les nœuds dans un cluster, nous avons utilisé la distance d_e, donnée par :

$$d_e = \sqrt{(X_{ch} - X_{n_i})^2 + (Y_{ch} - Y_{n_i})^2} \tag{2.1}$$

Où X_{ch}, Y_{ch} représente les coordonnées x, y du cluster-head et X_{ni}, Y_{ni} les coordonnées x, y du nœud "i".

En utilisant cette mesure de consommation d'énergie d_e, nous avons construit des clusters avec FFUCA et LEACH, à partir d'un réseau composé de 100 nœuds déployés aléatoirement dans un champ de $100m \times 100m$. Nous avons supposé que tous les nœuds ont une position fixe tout au long de la période de simulation. La station de base est située au centre du champ de déploiement.

Résultats numériques

Dans [61], [62], nous avons présenté les résultats numériques, où trois tests ont été effectués pour chaque algorithme avec diverses configurations. Les nœuds sont identifiés de 1 à 100. Les tableaux ci-dessous représentent les résultats obtenus. Ces derniers sont composés de trois colonnes. La colonne gauche représente les CHs et la droite représente la distance moyenne de consommation d'énergie entre le CH et les nœuds qui appartiennent à son cluster. La colonne du milieu représente les nœuds formant le cluster.

Dans les tests avec FFUCA, nous avons fixé des seuils de distances, representant la consommation d'énergie (c'est-à-dire la transmission d'un bit d'un nœud n_i à un nœud n_j appartenant au même cluster) respectivement à 10, 20 et 30 unités de d_e.

Dans le Tableau 2.1, avec le seuil 10, FFUCA fournit 17 clusters et 12 singletons. La consommation d'énergie moyenne dans les clusters générés est égale à $\frac{\sum d\_moy}{17} = 5,840$.

Dans le Tableau 2.2, avec le seuil 20, FFUCA génère moins de clusters que lors du premier test, mais les clusters sont plus grands. En effet, il fournit 13 clusters et 3 singletons. Malgré les différences de taille des clusters, la consommation énergétique moyenne entre les CHs et les nœuds est encore optimale, elle est égale à $10,791$.

Dans le Tableau 2.3, FFUCA confirme le déploiement optimal des nœuds en une itération quel que soit les paramètres initiaux (nœud et seuils). La distance moyenne d_moy est égale à $17,876$.

TABLE *2.1* – *Déploiement des nœuds avec un seuil* ≤ 10

CH	**Cluster**	d_moy
10	49	5,71
3	31, 51, 61, 75, 81, 93	2,02
8	20, 28, 63	5,01
6	79 80, 83	7,24
7	13, 30, 37	8,73
2	15, 70, 98	5,48
5	57, 60, 76	4,12
4	84	8,91
19	69	6,11
21	56, 67, 77	5,93
38	55	7,28
65	22, 60, 88	6,11
45	78	8,16
82	90	4,38
62	68, 96	7,91
94	95	0,67
64	42	5,51
Singletons	9, 47, 32, 40, 72, 58, 8, 73, 97, 91, 92, 89	/

TABLE *2.2* – *Déploiement des nœuds avec un seuil* ≤ 20

CH	*Cluster*	d_moy
10	24, 31, 34, 40, 42, 49,64, 79, 81	11,70
3	1, 11, 14, 19, 21, 51, 56,61, 69, 75, 77, 86, 93	10,16
8	16, 20, 25, 28, 55, 63,70, 82, 90	12,38
6	29, 72	14,40
9	22, 36, 65, 88, 94, 95,99	13,10
7	13, 30, 33, 37, 66, 80, 83	7,09
2	15, 18, 23, 38, 68, 91, 98	11,97
5	17, 6, 43, 57, 60, 74, 76,85, 87	12,73
4	84	8,34
32	45, 54, 78	13,95
41	46, 48, 73	13,32
44	50, 59, 97	4,62
52	62, 96	6,53
Singletons	89, 53, 92	

TABLE *2.3* – *Déploiement des nœuds avec un seuil* ≤ 30

CH	*Cluster*	*d_moy*
10	1, 19, 24, 28, 29, 31, 33, 34, 40, 48, 49, 51, 56, 58, 61, 64, 69, 72, 75, 79 81, 93	19,71
3	11, 14, 21, 22, 45, 47, 53, 57, 67, 77, 86, 92, 99	20,26
8	15, 16, 20, 23, 25, 35, 38, 55, 63, 70, 82, 90, 98	16,94
9	17, 36, 43, 60, 65, 66,74, 84, 85, 87, 88, 94, 95	21,13
2	18, 39, 44, 50, 52, 62, 68,91, 96, 97	21,52
7	13, 30, 37, 71, 80, 83, 89	9,86
5	12, 26, 76, 78	21,41
41	46, 73	16,78
54	59	13,28
Singletons	4, 5	

Pour les tests avec LEACH, nous avons utilisé la même mesure de consommation d'énergie que dans les tests avec FFUCA. Ainsi, nous avons regroupé les nœuds en effectuant trois tests (10, 100, 1000 itérations). Pour le premier test, nous avons regroupé les nœuds par l'exécution de LEACH en 10 itérations. Les résultats obtenus montrent que LEACH fournit 10 clusters et 13 singletons (voir Tableau 2.4). Même après 10 itérations, où certains nœuds ont consommé leurs énergie, LEACH ne fournit pas de clusters avec une *d_moy* optimal comme FFUCA. En fait, la distance moyenne obtenue avec LEACH est de $19,033$ alors qu'elle est égale à $5,840$ (seuil 10), $10,791$ (seuil 20) et $17,876$ (seuil 30) avec une itération de FFUCA.

TABLE *2.4* – *Déploiement des nœuds avec LEACH 10 itérations*

CH	*Cluster*	*d_moy*
9	4, 13, 30, 36, 37, 66, 67, 84, 99	26,54
23	2, 8, 16, 18, 20, 35, 44, 50, 30, 63, 70, 97, 98, 100	22,36
26	57, 78	8,87
32	12	5,20
47	11, 26, 32, 45, 53, 54,86, 92	12,81
49	6, 10, 23, 25, 28, 34, 40, 47, 52, 58, 82, 90	27,16
52	9, 15, 38, 39, 62, 68, 88, 91, 96	32,76
54	59	14, 78
64	7, 27, 29, 33, 41, 42, 46, 48, 71, 72, 73, 79, 80, 83, 21	26,23
88	5, 17, 21, 22, 43, 60, 65, 74, 76, 85, 87, 94, 95	13,62
Singletons	1, 3, 14, 19, 24, 29, 31, 40, 51, 61, 69, 75, 81, 93	

Dans le deuxième test avec LEACH, le clustering fournit après 100 itérations, 8 clusters et 13 singletons (voir Tableau 2.5). En dépit de l'énergie consommée, les clusters sont moins optimaux que ceux produits avec FFUCA dans une itération avec un seuil de 10. La distance moyenne d_moy avec LEACH est de $18,475$ alors qu'elle est égale à $5,84$ avec FFUCA.

TABLE *2.5 – Déploiement des nœuds avec LEACH 100 itérations*

CH	*Cluster*	*d_moy*
12	26, 32, 78	9,20
15	2, 39, 42, 52, 62, 68, 70, 91, 96, 98, 100	16,90
20	8, 16, 25, 28, 38, 55, 63, 82, 89, 90	14,90
35	15, 18, 20, 23, 35, 44, 50, 59, 97	15,17
42	6, 10, 27, 34, 41, 48, 49, 58, 64, 72, 79	19,11
47	11, 45, 53, 54, 57, 92	10,68
86	4, 5, 9, 12, 14, 17, 21, 22, 36, 43, 56, 60, 65, 67, 74, 76, 77, 85, 88, 94, 95, 99	26,09
89	7, 13, 30, 33, 37, 46, 66, 71, 73, 80, 83, 84	35,75
Singletons	1, 3, 19, 24, 29, 31, 40, 51, 61, 69, 75, 81, 93	

Un autre test avec LEACH, en considérant 1000 itérations a été effectué (voir Tableau 2.6). La consommation moyenne d'énergie dans les clusters est de $21,516$. Par rapport à l'énergie consommée pendant les 1000 itérations, la consommation moyenne d'énergie d_moy obtenue dans ce cas, est supérieure à celle de toute la consommation d'énergie trouvée avec FFUCA.

TABLE *2.6 – Déploiement des nœuds avec LEACH 1000 itérations*

CH	*Cluster*	*d_moy*
11	92	13,17
14	6, 13, 21, 29, 30, 33, 37,51, 56, 66, 67, 71, 72, 77,79, 80, 83, 84, 86, 89	26,01
20	100, 90, 82, 73, 63, 58, 55,52, 48, 46, 41, 38, 34, 28,27, 18, 16, 10	26,82
23	7, 35, 73, 98	45,33
45	3, 23, 47, 53, 57	18, 28
52	39, 60, 62, 65, 68, 91, 96	32,87
59	44, 45, 50, 59, 78, 97	22,18
60	4, 9, 14, 17, 43, 74, 76,85, 87	19,02
65	5, 22, 36, 88, 94, 95, 99	11,85
78	12, 15, 26, 32, 54	17,91
Singletons	1, 3, 14, 19, 24, 31, 51,56, 61, 69, 75, 77, 81, 93	

2.4 Proposition d'une nouvelle méthode de détection des attaques DoS basée sur le clustering

2.4.1 La méthode R-LEACH

Comme présenté dans l'état de l'art, les capteurs sans fil sont sensibles à de nombreux types d'attaques Ces attaques peuvent provenir de plusieurs sources tel que, des nœuds malveillants qui utilisent la technique d'inondation du réseau en envoyant un grand nombre de messages pour leurs voisins ou pour la station de base. Pour sécuriser les RCSF vulnérables à ces attaques, nous proposons une solution basée sur l'élection des nœuds de contrôle pour identifier un nombre maximum de nœuds compromis et bloquer leurs activités. Dans cette section, nous présentons la méthode proposée pour élire les nœuds de contrôle (Cnodes) [63]. Cette méthode est basée principalement sur l'utilisation de l'algorithme LEACH de façon récursive (R-Leach) pour l'élection des cluster-heads qui seront considérés comme des nœuds de contrôle (Cnodes).

Récursive LEACH

LEACH est un algorithme qui permet de fractionner un réseau en un ensemble de clusters où, chaque cluster est composé d'un cluster-head et d'un ensemble de nœuds membres. LEACH utilise une élection aléatoire et périodique pour choisir les CHs et construire les clusters. Cette élection permet un équilibrage de charge en termes de consommation d'énergie. Pour déterminer un premier ensemble de clusters, l'algorithme LEACH est exécuté, par la suite cet alghorithme est appliqué de nouveau sur chaque cluster obtenu. Ainsi, de cette façon nous pouvons obtenir k-clusters par l'application récursive de l'algorithme LEACH k fois. Ces itérations récursives seront appelées k-LEACH [63].

Exemple

Soit un réseau de capteurs composé de 100 nœuds avec les identifiants $[1, \ldots, 100]$. En exécutant l'algorithme LEACH, le résultat obtenu est 10 clusters répartis comme suit (voir Figure (a) 2.9) :

- **1-LEACH** première exécution de l'algorithme LEACH
  - $Cl_1 = [CH = 18; N = 44, 50]$, Ce cluster est composé des nœuds 44 et 50, avec le nœud 18 comme cluster-head.
  - Cl_2 = [CH = 21 ; N = 5, 9, 14, 17, 22, 43, 56, 60, 65,67, 74, 76, 77, 85, 86, 87, 88, 94, 95, 99].

Le second cluster a pour CH le nœud 21.

— ...

— Cl_{10} = [CH = 100 ; N = 2, 8, 15, 16, 20, 25, 28, 38, 55,63, 70, 82, 90, 91, 98], le 10^{ime} cluster comporte le nœud 100 qui représente son CH

- **2-LEACH** : Lorsque LEACH s'exécute une deuxième fois sur le cluster Cl_2, qui est un sous-réseau composé de 20 nœuds, le résultat obtenu est deux nouveaux clusters (voir Figure (b)-2.9) Le cluster Cl_{21} composé de 9 nœuds dont le nœud 60 est cluster-head, et le cluster Cl_{22} avec 8 nœuds membres et le CH de ce cluster est le nœud 65, le résultat obtenu est comme suit :

  — $Cl_{21} = [CH = 60; N = 5, 17, 43, 74, 76, 85, 86, 87]$.

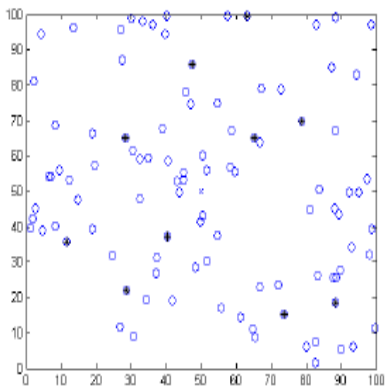
  — $Cl_{22} = [CH = 65; N = 9, 22, 67, 88, 94, 95, 99]$.

  En résumé, le cluster Cl_2 sera formé de deux sous-clusters Cl_{21}, Cl_{22} et le reste des nœuds auront le nœud 21 comme CH.

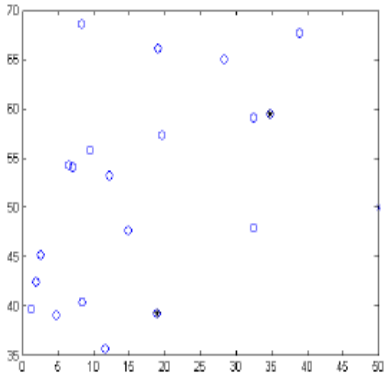
- **3-LEACH** : De la même manière, on a réexécuté récursivement l'algorithme LEACH pour la troisième fois sur les clusters Cl_{21} et Cl_{22}, le résultat obtenu est comme suit :

  — $Cl_{211} = [CH = 17; N = 43, 74, 76, 87]$.

  — $Cl_{221} = [CH = 22; N = 9, 67, 88, 94, 95, 99]$. A noter que le cluster Cl_{21} est formé par un seul sous-cluster.



(a) Exécution de LEACH pour la première fois.



(b) Exécution de LEACH pour la deuxième fois.

FIGURE *2.9* – Exemple de recursive LEACH.

2.4.2 Résultats numériques

Afin d'évaluer la solution proposée, des simulations sous Matlab ont été éxécutées en deux phases. Dans la première, nous avons exécuté récursivement l'algorithme LEACH [63]. Dans la deuxième phase, nous avons implémenté une méthode aléatoire pour construire de manière récursive des sous-ensembles (sous-clusters), et comparer les résultats de cette méthode avec la méthode proposée [64]. Initialement, nous avons généré aléatoirement 100 nœuds déployés dans un espace de dimension $100m \times 100m$. La position de chaque nœud est choisie aléatoirement entre [0, 100 m] sur chaque axe (horizontal et vertical). La station de base est placée au milieu de la zône de la simulation. Ensuite, nous avons exécuté l'algorithme LEACH avec un pourcentage de 10% de cluster-head et 10 rounds ("r=10"). Après l'exécution, nous avons obtenu Cl-clusters contenant K nœuds avec $K > 2$. Ces clusters sont eux-même divisés en sous-clusters en considérant les coordonnées de chacun de leurs nœuds, de la même façon en utilisant LEACH une seconde fois. Le sous-ensemble issu de la division de tout Cl-cluster s'appelle $Cl_i - cluster$. De plus, chaque cluster-head choisi dans cette étape est considéré comme nœud de contrôle (Cnode) pour chaque cluster Cl_i.

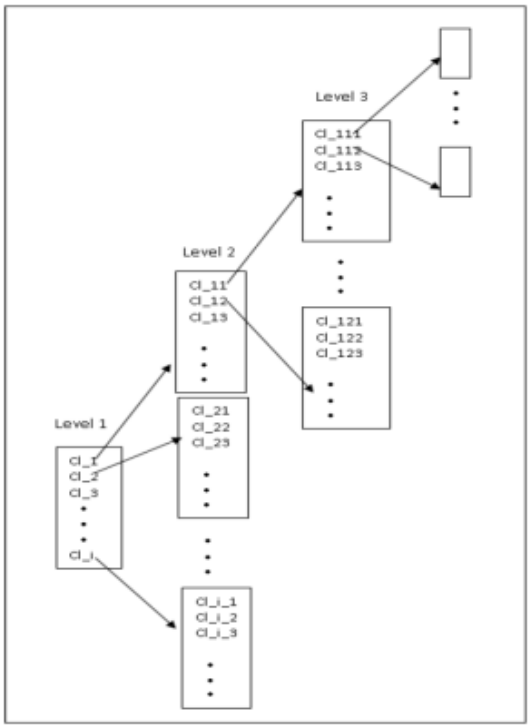


FIGURE *2.10* – Exécution de LEACH à plusieurs niveaux

En considérant le nombre de nœuds de chaque cluster (voir Figure 2.10), nous avons répété ce processus plusieurs fois jusqu'à ce que les clusters Cl_i formés dans ce niveau soient les mêmes clusters formés au niveau précédent Cl_{i-1}. A ce stade, l'algorithme atteint sa convergence. Pour chaque cluster obtenu de l'exécution récursive de l'algorithme LEACH, nous avons calculé la distance entre chaque cluster-head (nœud de contrôle) et ses nœuds voisins dans le cluster. Le résultat obtenu de cette première phase est Cl_i-cluster formé de k_j nœuds.

La deuxième phase consiste à choisir un nombre aléatoire de "Cnodes" en considérant la matrice de coordonnées des nœuds utilisés dans LEACH. Nous avons choisi aléatoirement un nombre de $Cl_i - clusters$ avec k_j Cnodes égal au nombre de Cnodes obtenu lors de l'exécution récursive de l'algorithme LEACH durant la première phase de la simulation. Ensuite, nous avons associé k_j nœuds pour chaque Cnode afin de construire des sous-ensembles équivalents aux clusters obtenus dans la première phase. Nous avons sélectionné les nœuds qui ont la plus courte distance du Cnode associé. Le résultat est un sous-ensemble S_i composé de k nœuds. En tenant compte de tous les sous-ensembles S_i ayant le même nombre de nœuds, nous avons comparé pour chaque cluster Cl_i obtenu lors de la première phase avec chaque sous-ensemble S_i résultant de la deuxième phase, les distances entre les nœuds membres et leurs Cnodes correspondants.

- Les résultats obtenus de la première phase (la méthode proposée) [64]. Le résultat est de 17 clusters répartis comme suit :

- 5 clusters chacun formé de 6 nœuds
- 3 clusters chacun formé de 4 nœuds
- 4 clusters chacun formé de 3 nœuds
- 5 clusters chacun formé de 2 nœuds.

Le cluster-head de chaque cluster est considéré comme un nœud de contrôle (Cnode), cette simulation a donnée 17 Cnodes distribués comme indiqué dans le Tableau 2.7.

- Les résultats obtenus de la deuxième phase (méthode aléatoire) de la simulation.

Dans cette phase, nous avons choisi au hasard 17 Cnodes parmi 100 nœuds. Le résultat est comme suit : Les identifiants des Cnodes sont : [3, 8, 13, 15, 20, 29, 33, 37, 50, 54, 59, 66, 83, 90, 93, 96, 98]. Par la suite, nous avons formé les sous-ensembles S_i pour chaque Cnode, en fonction du résultat obtenu de la simulation et en considérant la distance minimale entre chaque Cnode et ses nœuds les plus proches (voir Tableau 2.8). En se basant sur cette métrique, nous avons construit (17) dix-sept sous-ensembles composés de six, quatre, trois et deux nœuds.

TABLE *2.7 – Les résultats de l'exécution de la méthode proposée á trois niveaux*

Cl_i	Cl_{ij}	Cl_{ijk}	**Ensembles de nœuds**	*Cnode*	**Distance moyenne**
Cl_1			[44,50]		0,135
Cl_2	Cl_{21}	Cl_{211}	[43, 74, 76,87]	17	5,578
			[5,86]	60	13,400
	Cl_{22}	Cl_{221}	[9, 67, 88, 94, 95,99]	22	12,145
			[14, 56,77]	21	6,192
Cl_3	Cl_{31}	Cl_{311}	[27, 34, 42, 58, 64,73]	41	19,755
			[6, 10, 40,49]	24	8,472
Cl_4	Cl_{41}		[29,72]	79	8,580
			[51, 61,81]	31	7,110
Cl_5	Cl_{51}		[59,97]	23	15,939
Cl_6	Cl_{61}	Cl_{611}	[7, 30, 33, 71, 83,89]	80	12,711
	Cl_{62}		[4, 36,84]	66	22,444
Cl_7	Cl_{71}		[12, 26, 32, 45, 47,78]	57	16,018
Cl_8	Cl_{81}		[11,19]	92	10,403
Cl_9	Cl_{91}		[39, 52,68]	62	7,007
Cl_{10}	Cl_{101}	Cl_{1011}	[16, 38, 91,98]	15	13,313
	Cl_{102}	Cl_{1021}	[8, 20, 25, 55, 63,90]	28	10,518

- 5 sous-ensembles chacun formé par 6 nœuds.
- 3 sous-ensembles chacun formé par 4 nœuds.
- 4 sous-ensembles chacun formé par 3 nœuds.
- 5 sous-ensembles chacun formé par 2 nœuds.

Ensuite, nous avons calculé la distance moyenne entre chaque Cnode et ses nœuds dans le sous-ensemble.

TABLE *2.8 – Les résultats de l'exécution de la méthode aléatoire*

Sous-ensemble	**Ensembles de nœuds**	*Cnode*	**Distance moyenne**
S_1	[14, 31, 51, 61, 75, 22]	3	8,584
S_2	[22, 16, 25, 28, 63, 82]	8	12,479
S_3	[7, 30, 71, 72, 80, 8]	13	14,092
S_4	[2, 62, 68, 70, 91, 100]	15	10,791
S_5	[23, 34, 35, 38, 49, 58]	20	25,620
S_6	[6, 56, 64, 79]	29	13,215
S_7	[10, 42, 67, 89]	33	25,642
S_8	[4, 9, 36, 99]	37	32,735
S_9	[18, 44, 57]	50	6,449
S_{10}	[45, 53, 78]	54	9,363
S_{11}	[19, 47, 92]	59	19,349
S_{12}	[21, 22, 77]	66	28,398
S_{13}	[24, 40]	83	47,126
S_{14}	[41, 48]	90	31,388
S_{15}	[69, 86]	93	12,649
S_{16}	[39, 52]	96	8,343
S_{17}	[26, 11]	98	54,903

- Comparaison entre la méthode aléatoire et la méthode proposée

Dans le Tableau 2.9, nous avons considéré trois clusters, chacun étant composé de quatre nœuds et nous avons calculé la distance moyenne pour les deux méthodes aléatoire et proposée. Dans la méthode proposée, le Cnode ayant l'ID 17 contrôle les nœuds $[43, 74, 76, 87]$ et la distance moyenne entre ce Cnode et ses voisins est de $5,578$. Par contre, dans la méthode aléatoire, le Cnode avec l'ID 29 contrôle les nœuds $[6, 56, 64, 79]$ avec une distance moyenne égale à $13,215$pour tous les clusters représentés dans le Tableau 2.8.

La distance moyenne résultante de la méthode proposée (4.560) est significativement inférieure à celle déduite de la méthode aléatoire (11.932). En effet, dans le Tableau 2.9, nous avons considéré trois clusters, chacun formé de quatre nœuds.

TABLE *2.9 – Distance entre le Cnode et quatre nœuds avec trois clusters*

	Cl_1	Cl_2	Cl_3	Distance moyenne
Méthode aléatoire	13,215	25,642	32,735	11,932
Méthode proposée	5,578	8,472	13,313	4,560

De même, dans le Tableau 2.10, nous avons noté que la distance moyenne obtenue (11.055) par la méthode proposée est toujours meilleure par rapport à la méthode aléatoire (15.890).

TABLE *2.10 – Distance entre le Cnode et trois nœuds avec quatre clusters*

	Cl_1	Cl_2	Cl_3	Cl_4	Distance moyenne
Méthode aléatoire	6,449	9,363	19,349	28,398	15,890
Méthode proposée	6,192	7,007	8,580	22,444	11.055

Enfin, dans le Tableau 2.11, nous avons considéré également cinq clusters chacun formé de deux nœuds. Comme précédemment, il a été confirmé que la méthode proposée fournit de meilleurs résultats (distance moyenne égale à 10.714) par rapport à la méthode aléatoire (27.075). Par rapport aux résultats obtenus, la distance moyenne séparant chaque Cnode de ses nœuds voisins pour chaque cluster Cl_i formé en utilisant la méthode proposée est trés réduite par rapport à la distance séparant chaque Cnode de ses voisins pour chaque sous-ensemble S_i formé par la méthode aléatoire.

TABLE *2.11 – Distance entre le Cnode et deux nœuds avec cinq clusters*

	Cl_1	Cl_2	Cl_3	Cl_4	Cl_5	Distance moyenne
Méthode aléatoire	47,126	31,388	12,649	8,343	54,903	27,075
Méthode proposée	8,580	10,135	10,403	13,400	15,939	10,714

Généralement, le modèle radio utilisé dans les réseaux de capteurs sans fil considère la distance entre l'émetteur et le récepteur, comme un paramètre important pour déterminer la consommation de l'énergie pour les échanges de messages. En tenant compte de ce paramètre, le modèle de la consommation d'énergie de la radio utilisé dans [69] considère que l'énergie consommée dans les réseaux de capteurs est proportionnelle au carré de la distance. En effet, si la distance augmente, le coût énergétique augmente et devient très élevé. Ce critère a été utilisé pour évaluer la méthode proposée, en calculant la distance moyenne (dist_moy) entre les Cnodes et leurs nœuds voisins dans tout le réseau en utilisant cette formule

$$dist\_moy = \frac{\sum d\_moy^2}{N} \tag{2.2}$$

Où d_moy représente la distance moyenne dans chaque cluster et N représente le nombre de clusters générés. Nous avons considéré N égal à 17.

Le Tableau 2.12 indique que le carré de la distance moyenne $(159,209)$ calculé en utilisant la méthode proposée est largement inférieur à celui obtenu de la méthode aléatoire $(639,043)$, environ trois fois meilleure. Ainsi, nous pouvons conclure que l'utilisation de notre méthode assure une bonne économie d'énergie. Par conséquent, elle prolonge la durée de vie des réseaux de capteurs.

TABLE *2.12* – *Distance moyenne entre les Cnodes et leurs nœuds dans tout le réseau*

	Moyenne du carré de la distance
Méthode aléatoire	639,043
Méthode proposée	159,209

2.4.3 Résultats numériques de l'utilisation de R-LEACH pour la prévention des attaques DoS

- Environnement d'expérimentation

Afin de montrer, aussi l'avantage de la méthode proposée, d'autres simulations à l'aide de Simevents de Matlab ont été effectuées. Les simulations sont réalisées en deux phases [77]. La première concerne l'élection des nœuds de contrôle, tandis que la deuxième phase est consacrée à la détection et le blocage des nœuds compromis par les nœuds de contrôle. Dans la première phase de ces simulations, initialement, 100 nœuds ont été aléatoirement distribués dans un champ de dimension $100m \times 100m$. La station de base se trouve au milieu de la zône de la simulation. Par la suite, l'algorithme LEACH est exécuté avec un taux désiré de CH égal à 10 et 10 rounds, le résultat obtenu est un ensemble de clusters contenant K nœuds avec $K > 2$. En éxécutant LEACH une seconde fois, ces clusters sont eux-mêmes divisés en sous-clusters en considérant les coordonnées de chacun de leurs nœuds. Les sous-ensembles issus de la division de tout cluster représentent des sous-clusters. Chaque cluster-head des sous-clusters obtenus dans cette phase sera considéré comme nœud de contrôle (Cnode).

- Résultats obtenus

Après l'exécution de LEACH sur un réseau de capteurs composé de 100 nœuds avec les ID $[1 \ldots 100]$, les résultats sont 10 clusters répartis comme suit (Figure (a)- 2.11) :

1. 1-LEACH

— Cl_1= $[CH = 9; N = 4, 13, 30, 36, 37, 66, 67, 84, 99]$.
— Cl_2= $[CH = 23; N = 2, 8, 16, 18, 20, 35, 44, 50, 55, 63, 70, 98, 100]$.
— ...
— Cl_9 =$[CH = 64; N = 7, 27, 29, 33, 41, 42, 46, 48, 71, 72, 73, 79, 80, 83, 89]$.
— Cl_{10} = $[CH = 88; N = 5, 17, 21, 22, 43, 60, 74, 76, 85, 87, 94, 95]$.

2. 2-LEACH : Lorsque nous avons exécuté LEACH une deuxième fois sur le cluster Cl_1 (Figure (b)- 2.11), les résultats suivants ont été obtenus :

   Deux nouveaux clusters (Figure (c)- 2.11), qui representent deux sous-réseaux.

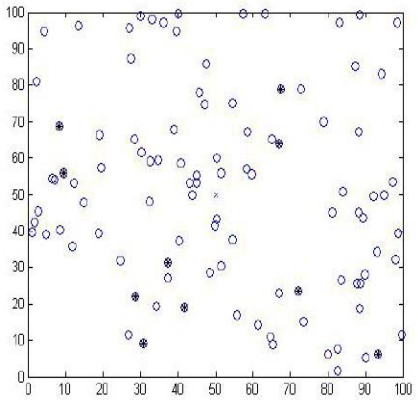
   - Le premier est Cl_{11} avec 5 nœuds et son cluster-head est le nœud 84. $Cl_{11} = [CH = 84; N = 4, 13, 30, 66, 99]$.
   - Le deuxième cluster est Cl_{12} avec 2 nœuds et son CH est le nœud 36, $Cl_{12} = [CH = 36; N = 37, 67]$.

Dans la seconde phase de la simulation, nous avons choisi le cluster Cl_{11} = [CH = 84 ; N = 4, 13, 30, 66, 99] qui est composé de 5 nœuds avec le nœud 84 comme cluster-head.

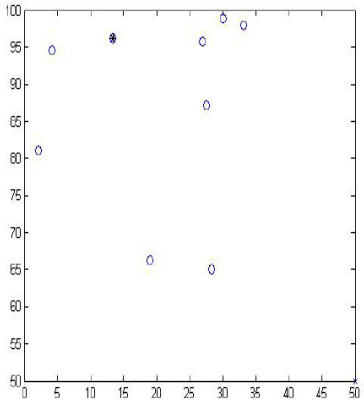
Dans cette simulation, les cluster-heads sont considérés comme des nœuds de contrôle (Cnodes) et le nœud 66 comme nœud compromis. La simulation consiste à échanger des messages entre différents nœuds du cluster. Le Cnode numéro 84 surveille l'activité du cluster et le nœud 66 (nœud compromis) inonde le cluster en envoyant un flux important de paquets.

Nous avons supposé que :

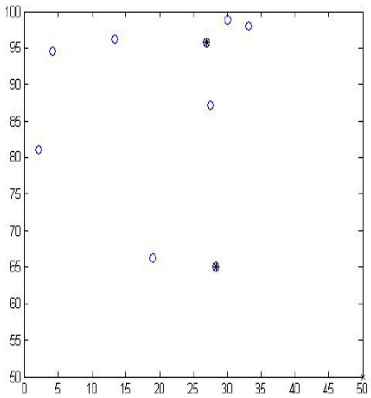
1. Le temps de la simulation est de 1000 unités de temps.
2. Le nœud compromis envoie ses paquets selon une distribution exponentielle avec les taux $\lambda \in [0.1, \ldots, 1]$.
3. Le Cnode détecte une activité anormale si le nombre de paquets envoyés dépasse un certain seuil variant dans l'intervalle $[100, \ldots, 600]$.
4. Les simulations consistent en un échange de messages entre différents nœuds du cluster. Le Cnode 84 surveille l'activité du cluster et le nœud 66 est un nœud compromis qui essaye d'inonder le cluster en envoyant un grand flux de paquets.
5. Lorsque la détection est observée, le Cnode bloque l'envoi des paquets du nœud compromis.



(a) Exécution de LEACH pour la première fois.



(b) Exécution de LEACH pour la deuxième fois.



(c) Exécution de LEACH pour la troisième fois.

FIGURE *2.11* – Simulation de LEACH récursive.

Dans la Figure 2.12, la courbe C_1 montre le nombre de paquets transmis par un nœud compromis sans considérer les seuils. Le nombre de paquets transmis diminue par rapport au taux de probabilité (suit une distribution exponentielle). La courbe C_2 représente le nombre de parquets transmis par le nœud compromis entre le début de la simulation et le processus de blocage par le Cnode.

Le nombre de paquets transmis diminue par rapport au taux de probabilité associé à la distribution exponentielle. En comparant les deux courbes C_1 et C_2, nous avons constaté que l'écart est considérable entre le nombre de paquets transmis par le nœud compromis avec et sans seuil.

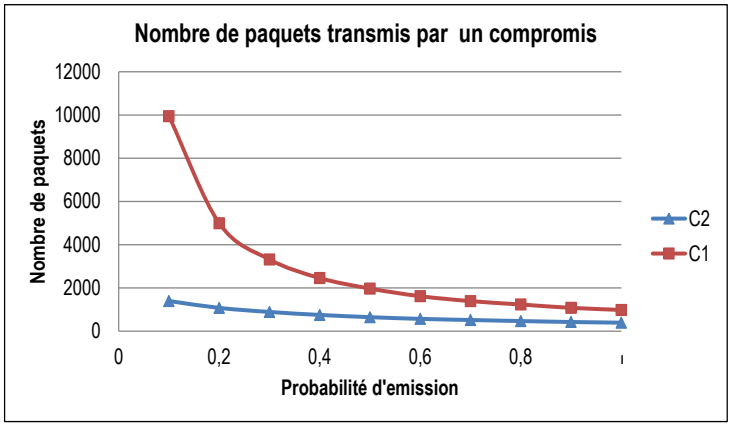


FIGURE *2.12* – Nombre de paquets transmis par un nœud compromis

La Figure 2.13 représente le temps de détection qui augmente avec la vitesse de probabilité d'émission et le seuil représentant le nombre de paquets tolérés et transmis par le nœud compromis. Par exemple avec un taux λ égal à $0,5$ et un seuil égal à 600, le nombre de paquets transmis est de 648. Cependant, les messages envoyés par le nœud compromis sont bloqués à un temps égal à 1000. Lorsque le taux λ est égal à 1 et le seuil est égal à 100, le nombre de parquets transmis est de 198. La différence entre le seuil et le nombre de paquets transmis entre la détection et le blocage $(600, 612)$ et $(100, 198)$ représente le nombre de paquets restants dans la file d'attente.

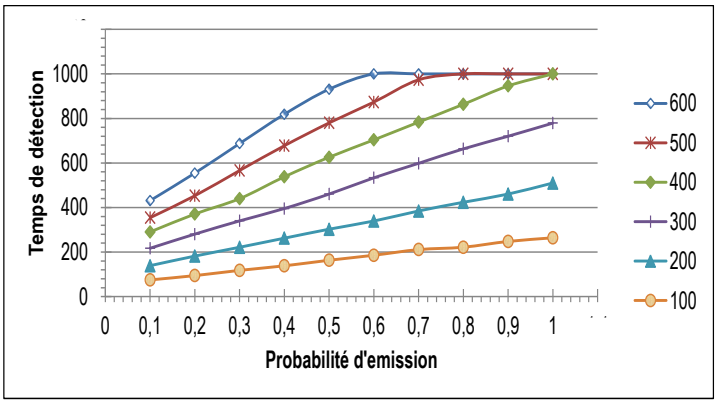


FIGURE *2.13* – Temps de la simulation

La Figure 2.14 représente le temps de détection en fonction du seuil. Le temps de détection de l'attaque de flooding par le nœud compromis augmente avec le seuil. Lorsque le seuil $\in [400, \ldots, 600]$ et la probabilité d'émission $\lambda \in [0.5, \ldots, 1]$: Le temps de blocage est environ de 1000 unités de temps : Pour le seuil $\in [100, \ldots, 400]$ et $\lambda \in [0.1, \ldots, 0.4]$ le temps de blocage est de $[264,048, \ldots, 945,993]$.

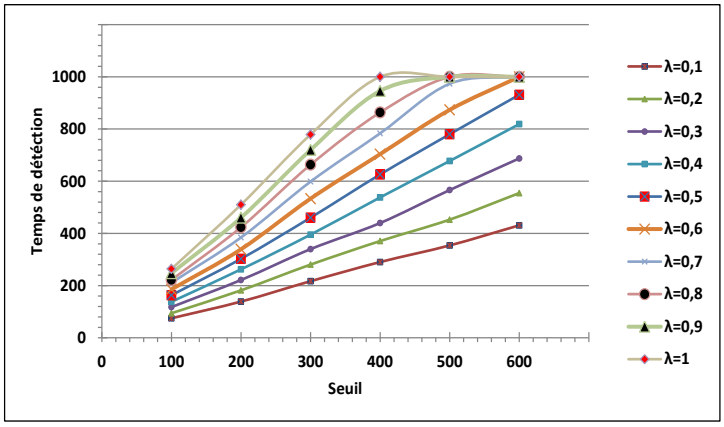


FIGURE *2.14* – Temps de détection en fonction du seuil

La Figure 2.15 représente le taux de paquets perdus transmis par le nœud compromis. Ce taux est égal au rapport entre le nombre de paquets transmis sans considérer le seuil et le nombre de paquets transmis en tenant compte du seuil. Cela représente le nombre de paquets qui doivent être transmis par le nœud compromis et le nombre de paquets transmis avant la détection et le blocage des émissions des paquets. Le taux de perte est très important, il est de [92%] pour un seuil de 100 et de [78%] pour un seuil de 300.

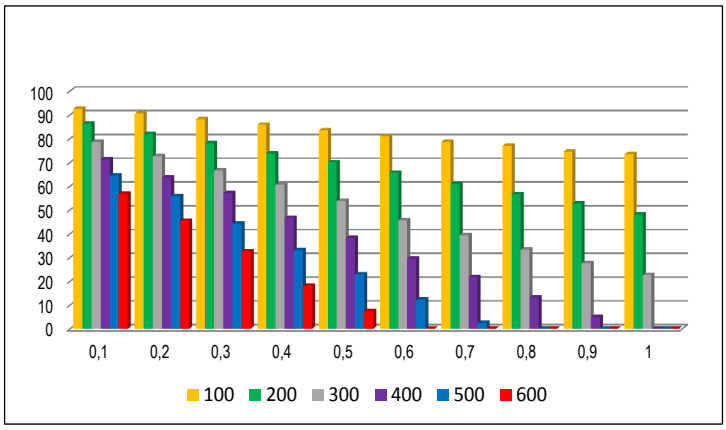


FIGURE *2.15* – Taux de perte des paquets.

2.4.4 La méthode R-FFUCA

Cette section présente une nouvelle approche pour détecter les attaques (DoS) [63]. Celle-ci repose sur le clustering qui consiste à regrouper récursivement les capteurs d'un RCSF jusqu'à obtenir une granularité requise (choisie par l'expert). Nous proposons d'appliquer cette approche avec l'algorithme FFUCA. Nous avons discuté le comportement et l'efficacité de l'approche proposée à travers une comparaison avec les résultats numériques obtenus par la méthode R-LEACH.

2.4.5 Méthode proposée

L'objectif de la méthode proposée est l'élection des nœuds de contrôle (Cnodes). L'approche proposée consiste à appliquer récursivement l'algorithme de clustering FFUCA pour détecter des nœuds stratégiques (CHs et Cnodes) en tenant compte de la consommation d'énergie et de l'emplacement des nœuds dans le réseau. Le but recherché est d'étudier la pertinence offerte par cet algorithme avec l'approche proposée. Ainsi, des tests ont été effectués sur un RCSF ayant 100 nœuds, où la même mesure de similarité est utilisée dans les simulations. Cette mesure représente la consommation d'énergie. Notre approche consiste à exécuter récursivement l'algorithme de clustering FFUCA dans le réseau. Nous divisons d'abord tout le réseau en clusters, ainsi, les premiers clusters et les CHs sont définis. Ensuite, la même opération est appliquée plusieurs fois, mais sur les clusters obtenus. Le but de la dernière exécution est d'obtenir les premiers nœuds (Cnodes). Le clustering récursif est appliqué jusqu'à obtention de la granularité souhaitée. Nous avons appelé cette approche R-FFUCA.

2.4.6 Résultats numériques

En appliquant, le clustering récursif sur l'algorithme FFUCA, sur un RCSF avec 100 nœuds. Le critère de proximité considéré (d_e) représente la métrique qui mesure l'énergie consommée pour transmettre un bit d'un nœud quelconque n_i à un autre n_j appartenant au même cluster. Cette métrique est donnée par la formule 2.1.

En utilisant la mesure d_e, nous avons généré des clusters avec FFUCA. Nous avons d'abord généré 100 nœuds déployés dans le champ $100m \times 100m$. Nous avons supposé que tous les nœuds ont une position fixe tout au long de la période de simulation. La station de base est située au centre du champ.

Dans nos simulations, nous avons exécuté trois itération pour l'algorithme LEACH et FFUCA (c'est-à-dire 3-LEACH et 3-FFUCA). Les résultats de chacun d'eux sont résumés dans les tableaux 2.13, 2.14, 2.15, 2.16, 2.17, 2.18. Chaque nœud est marqué par un identifiant $i \in [1, \ldots, 100]$. Chaque tableau est composé de trois colonnes. La colonne de gauche représente les CHs et la droite représente la distance moyenne de consommation d'énergie (d_moy) entre le CH et les nœuds qui appartiennent à son cluster. La colonne du milieu représente les nœuds qui forment le cluster.

1. **Résultats de l'approche basée sur FFUCA**

Dans les trois tests avec FFUCA, nous avons fixé respectivement des seuils de distances de consommation d'énergie d_e à $10, 20$ et 30. Nous avons noté que plus la consommation d'énergie entre le CH et les nœuds dans les clusters est faible, plus la durée de vie du réseau augmente. La réduction des distances à l'intérieur des clusters est un avantage majeur de l'algorithme FFUCA. Il est dû au grand seuil choisi (c'est-à-dire le seuil 30).

Le Tableau 2.13 présente les résultats de 1-FFUCA qui fournit 9 clusters. La distance moyenne d_moy_Totale est de $18, 92$. Elle est plus élevée que la d_moy_Totale trouvée dans 1-LEACH (c'est -à-dire $14, 29$).

Table *2.13 – 1-FFUCA (seuil* ≤ 30*)*

CH	*Cluster*	*d_moy*
10	1, 19, 24, 28, 29, 31, 33, 34,40, 48, 49, 51, 56, 58, 61, 64, 69, 72, 75, 79, 81, 93	20.71
3	11, 14, 21, 22, 45, 47, 53, 57, 67, 77, 86, 92, 99	21.26
8	15, 16, 20, 23, 25, 35,38, 55, 63, 70, 82, 90, 98	17.94
9	17, 36, 43, 60, 65, 66, 74, 84, 85, 87, 88, 94, 95	22.13
7	13, 30, 37, 71, 80, 83, 89	10.86
2	18, 39, 44, 50, 52, 62, 68, 91, 96, 97	22.52
5	12, 26, 76, 78	22.41
41	46, 73	17.78
54	59	14.28
d_moy_Totale		18.92

Pour le second test 2-FFUCA, avec un seuil égal à 20, FFUCA génère 24 Cnodes pour 9 CHs (voir Tableau 2.14). Le reste des nœuds sont des singletons, ils sont à une distance de plus de 20 unités de tous les nœuds. En conséquence, chaque cluster

a plus de nœuds à contrôler que dans 2-LEACH. En outre, la distance moyenne d_moy_Totale dans les clusters est de $9,47$ au lieu de 11,13 comme dans 2-LEACH.

TABLE *2.14 – 2-FFUCA (seuil*≤ 20*)*

CH	$\mathbf{Cnode}_1$	$Cluster$	d_moy
10	19	1, 28, 69	10.63
	40	49	10.89
	64	29, 34	17.06
	81	31, 51, 56, 61, 75, 93	9.67
	33	72, 79	10.00
	48	58	17.91
3	21	14, 67, 77, 24, 86	8.58
	57	11, 45, 47	14.23
	99	22	8.66
	53	92	3.64
	24	*singletons*	
8	23	*singletons*	
	35	*singletons*	
	38	15, 16, 55, 70, 98	11.32
	82	20, 25, 63, 90	5.99
9	43	94, 95	9.88
	66	84	16.77
	87	74	2.87
	9	36, 65, 88	14.21
	17	60, 85	8.58
7	37	13, 30, 80, 83	6.99
	71	89	5.64
2	39	18, 68, 91, 96	6.90
	52	62	3,42
	97	44, 50	6.67
5	26	12, 78	8.47
	76	5	8.47
41	46	*singletons*	
	73	*singletons*	
54	59	*singletons*	
d_moy_Totale			9.47

La simulation 3-FFUCA fournit 12 Cnodes (voir Tableau 2.15) et 22 clusters. La distance moyenne d_moy_Totale est égale à $5,51$ ce qui est inférieur à la d_moy_Totale dans 3-LEACH (c'est-à-dire $11,08$).

TABLE *2.15 – 3-FFUCA (seuil* ≤ 10*)*

CH	**Cnode**$_1$	**Cnode**$_2$	*Cluster*	*d_moy*
10	19	1	69	1.85
		28	*singletons*	
	40		*singletons*	
	64		*singletons*	
	81	51	31, 56, 61	8.25
		75	81	6.57
	33		*singletons*	
	48		*singletons*	
3	21	21	14, 67, 77	4.33
	57	45	47	4.19
	99		22	8.66
	53		92	3.64
	24		*singletons*	
8	23		*singletons*	
	35		*singletons*	
	38	55	16	7.76
		15	70, 98	8.17
	82	25	90	6.44
		63	20	1.97
9	43	95	94	0.67
	66		*singletons*	
	82			2.87
	9	88	65	3.79
	17		85	3.90
7	37		13, 30, 80, 83	6.99
	71		89	5.64
2	39		68, 91, 96	6.90
	52		62	3.42
	97		44, 50	6.67
5	26		78	6.30
	76		5	8.47
d_moy_Totale				5.51

2. Résultats de l'approche basée sur LEACH

Pour évaluer l'approche proposée basée sur FFUCA, nous avons comparé notre approche avec LEACH, des simulations ont été éffectuées sous Matlab. Dans un champ de dimension $100m \times 100m$, la position de chaque nœud est choisie aléatoirement entre 0 et $100m$ sur chaque axe (horizontal et vertical). Nous avons exécuté LEACH avec le pourcentage souhaité de CH égal à 10 et 10 rounds. Des résultats 1-LEACH qui contient K clusters ont été obtenus. Ces clusters sont divisés en sous-clusters de façon itérative (en utilisant LEACH, et en considérant les mêmes coordonnées des nœuds). Les sous-ensembles résultants sont étiquetés li-cluster. De plus, chaque CH choisi dans cette étape est considéré comme nœud de contrôle (Cnode) pour chaque cluster Cl_i. Ce processus a été répété plusieurs fois jusqu'à ce que les clusters Cl_i formés dans ce niveau soient les mêmes que ceux formés au niveau Cl_{i-1} précédent. Enfin, la distance (d_moy) entre chaque CH (Cnode) et ses voisins dans le même cluster a été calculée.

Les résultats de 1-LEACH sont représentés dans le Tableau 2.13. Le 1-LEACH fournit 10 clusters et la distance moyenne globale d_moy_Totale dans les clusters est égale à $14,29$.

TABLE *2.16 – 1-LEACH*

CH	$Cluster$	d_moy
18	44, 50	10.13
21	5, 9, 14, 17, 22, 43, 56, 60, 65, 67,74, 76, 77, 85, 86, 87, 88, 94, 95, 99	20.95
24	6, 10, 27, 34, 40, 41, 42, 46, 48,49, 58, 64, 73	29.32
31	29, 51, 61, 72, 79, 81	11.75
35	23, 59, 97	10.81
37	4, 7, 13, 30, 33, 36, 66, 71, 80, 83, 84, 89	17.04
54	12, 26, 32, 45, 47, 53, 57, 78	13.39
69	1, 11, 19, 92	7.33
96	39, 52, 62, 68	7.70
100	2, 8, 15, 16, 20, 25, 28, 38, 55, 63, 70, 82, 90, 91, 98	14.55
d_moy_Totale		14.29

Les résultats de 2-LEACH sont représentés dans le Tableau 2.17. En plus des colonnes du Tableau 2.16, nous avons ajouté une nouvelle colonne qui contient les premiers Cnodes (Cnode_1). La distance moyenne dans tous les sous-clusters résul-

tants est égale à 11,13. 2-LEACH fournit 13 Cnodes et 16 clusters (le cluster qui n'a pas de Cnode est contrôlé par le CH correspondant). Chaque groupe de 1-LEACH est contrôlé en moyenne par deux Cnodes.

TABLE *2.17 – 2-LEACH*

CH	**Cnode**$_1$	*Cluster*	*d_moy*
18	18	44, 50	10.13
21	60	5, 17, 43, 74, 76, 85, 86, 87	13.07
65	9	, 22, 67, 88, 94, 95, 99	10.53
	21	14, 56, 77	6.19
24	46	27, 34, 41, 42, 48, 58, 64, 73	19.47
	24	6, 10, 40, 49	8.47
31	79	29, 72	8.58
	31	51, 61, 81	7.11
35	23	59, 97	15.93
37	13	7, 30, 33, 71, 80, 83, 89	13.86
	66	4, 36, 84	11.22
54	57	12, 26, 32, 45, 47, 78	16.01
69	92	11, 19	10.40
96	62	39, 52, 68	7.00
100	70	2, 15, 16, 38, 91, 98	11.47
	82	8, 20, 25, 28, 55, 63, 90	8.72
d_moy_Totale			11.13

La dernière simulation avec LEACH (c'est-à-dire 3-LEACH) est représentée dans le Tableau 2.18. Nous avons ajouté une autre colonne pour représenter la deuxième sélection de Cnodes (Cnode$_2$). Cette simulation a donnée 7 nœuds de contrôle et 17 clusters.

Le cluster qui ne possède pas Cnode est contrôlé par le CH correspondant. La distance moyenne dans les clusters finaux (c'est-à-dire les sous-clusters) *d_moy_Totale* est égale à 11.08 (voir Tableau 2.18).

TABLE *2.18 – 3-LEACH*

CH	**Cnode**$_1$	**Cnode**$_2$	*Cluster*	*d_moy*
18	18		44, 50	10.13
21	60	17	43, 74, 76, 87	5.57
		60	5, 86	13.40
	65	22	9, 67, 88, 94, 95, 99	12.14
24	46	41	27, 34, 42, 58, 64, 73	19.75
		24	6, 10, 40, 49	8.47
31	79	79	29, 72	8.58
	31	31	51, 61, 81	7.11
35	23	23	59, 97	15.93
37	13	80	7, 30, 33, 71, 83, 89	12.71
	66	66	4, 36, 84	11.22
54	57	57	12, 26, 32, 45, 47, 78	16.01
69	92	92	11, 19	10.40
96	62	62	39, 52, 68	7.00
100	70	15	16, 38, 91, 98	13.31
	82	28	8, 20, 25, 55, 63, 90	10.51
d_moy_Totale				11.08

3. Comparaison entre les approches basées sur LEACH et FFUCA

Le modèle radio utilisé actuellement dans les capteurs sans fil considère de nombreux paramètres, tels que, le nombre de paquets envoyés et reçus et la distance entre l'émetteur et le récepteur. Ces paramètres sont trés importants pour déterminer la dissipation d'énergie. Concernant le paramètre de distance, dans le modèle de dissipation d'énergie utilisé dans [70], les auteurs considèrent que l'énergie de consommation dans le réseau est proportionnelle au carré de la distance.

En effet, si la distance augmente, le coût énergétique est très élevé. Nous utilisons donc ce critère pour évaluer la pertinence de LEACH et FFUCA. Ainsi, en utilisant les résultats de 3-LEACH et de 3-FFUCA, nous calculons la distance carrée moyenne entre les Cnodes et leurs voisins en utilisant la formule 2.2.

Le Tableau 2.19 montre que FFUCA fournit une structure organisationnelle plus optimale que LEACH. En fait, la distance moyenne dans 3-FFUCA est égale à $624,99$ alors qu'elle est égale à $1859,99$ dans 3-LEACH. Par conséquent, l'utilisation de FFUCA assure un meilleur déploiement des nœuds et une économie d'énergie.

TABLE *2.19 – Distance moyenne entre les Cnodes et leurs voisins dans le 3-clustering*

	Moyenne des carrés des distances
Approche basée sur LEACH	1859,99
Approche basée sur FFUCA	624,99

— **Faux positifs et faux négatifs des approches LEACH et FFUCA**

Le faux positif est une expression qui décrit une situation où les dispositifs déclenchent une alarme pour une éventuelle activité malveillante ou une attaque [78]. Sinon, le faux positif peut représenter une mauvaise attaque ou un mauvais nœud malveillant. Le taux de faux positifs, ici, définit la proportion de nœuds normaux qui sont considérés comme des nœuds compromis. Il s'agit du pourcentage de nœuds qui n'appartiennent à aucun cluster (nœuds qui ne sont pas sous contrôle). Ainsi, dans le cas d'une fausse alarme, ces nœuds sont considérés comme des nœuds compromis. Le faux négatif décrit l'échec de la détection des activités malveillantes [78]. Par exemple, lorsqu'une attaque réelle est considérée comme une connexion normale. Dans notre étude, le taux de faux négatifs est la proportion de nœuds compromis qui sont considérés comme normaux. Il représente le pourcentage de Cnodes (cluster head) dans chaque sous-cluster. Les taux de faux positifs (Fp) et faux négatifs (Fp) sont calculés comme suit :

$$Fn = \frac{N_c}{n} \tag{2.3}$$

$$Fp = \frac{N_s}{n} \tag{2.4}$$

Où : Ns et Nc : représentent respectivement, le nombre de nœuds qui n'appartiennent à aucun cluster (singletons), le nombre de Cnodes.

N représente le nombre total de nœuds dans l'ensemble du réseau.

Fn est le taux de faux négatifs et Fp est le taux de faux positifs.

Par rapport aux résultats de la simulation, le taux de faux positifs et le taux de faux négatifs sont représentés dans le Tableau 2.20 :

TABLE *2.20 – Faux positifs et faux négatifs de LEACH et de FFUCA*

	Fp	Fn
Approche basée sur LEACH	3%	16%
Approche basée sur FFUCA	8%	12%

Parmi les caractéristiques d'une efficience d'un système de détection, ses faibles quantités de faux positifs (c'est-à-dire de fausses alertes) et faux négatifs (c'est-à-dire les attaques indétectables) [78]. Nos résultats ont montré que FFUCA détecte plus de fausses alertes, et présente moins d'attaques indétectables que LEACH.

2.5 Conclusion

Dans ce chapitre, des méthodes de détection des attaques de déni de service basées sur la technique de clustering, qui a pour objectif principal le maintien efficace de l'utilisation de l'énergie des nœuds de capteurs, un des défis majeur des réseaux de capteurs, ont été proposées.

Dans ce contexte, une nouvelle solution de sécurisation des réseaux de capteurs sans fil, basée sur le clustering appelée R-LEACH est proposée. Cette méthode consiste à sélectionner des nœuds de contrôle afin d'améliorer la détection des attaques DoS tel que le brouillage ou le flooding. La méthode est basée sur l'application récursive de l'algorithme LEACH pour sélectionner les cluster-heads (Cnodes) avec des nœuds non loin [64]. Les nœuds "Cnodes" élus ont pour mission la surveillance des nœuds voisins. Les résultats de la simulation montrent que notre approche est beaucoup mieux comparée à la méthode aléatoire par rapport à la distance entre les Cnodes et leur voisins, ce qui implique implicitement la réduction de la consommation d'énergie dans le réseau.

Aussi, un ensemble d'expérimentations effectuées sur la méthode proposée, en utilisant le simulateur Simevents ont été données. Les résultats numériques obtenus par simulation, ont montré que notre approche donne des résultats significatifs en termes de taux de détection et de temps de détection [77].

Une nouvelle approche pour détecter les attaques de déni de service dans les réseaux de capteurs sans fil [77] a été également proposée. Cette approche est basée sur un clustering récursif appelée R-FFUCA. Les résultats numériques obtenus de l'implémentation de cette méthode induit une meilleure gestion de l'énergie et donc une plus longue durée de vie du réseau.

Dans le chapitre suivant, nous prétons également, une nouvelle approche dynamique et adaptive permettant la détection du flooding dans les réseaux de capteurs.

Chapitre 3

PROPOSITION ET MODELISATION DE LA METHODE ACCORDEON POUR LA DETECTION DES ATTAQUES DoS

Sommaire

3.1 Contexte et objectifs

Dans ce chapitre, nous avons proposé une nouvelle méthode appelée la méthode Accordéon pour détecter et appréhender les attaques de déni de service dans les RCSF. Cette approche est une méthode dynamique et adaptative basée sur l'utilisation de la technique de clustering qui permet d'élire des nœuds de contrôle ayant pour fonction principale l'analyse du trafic à l'intérieur d'un cluster et l'envoi des avertissements au cluster-head, chaque fois qu'un comportement anormal est suspecté ou détecté.

La méthode proposée [79] repose sur l'analyse de l'évolution des seuils (alertes) des messages envoyés dans le cluster. La méthode proposée a été évaluée et les résultats numériques obtenus montrent son avantage par rapport aux autres méthodes de détection .

3.2 Définition formelle

Dans les attaques par flooding, un nœud malveillant envoie un grand nombre de paquets communs vers une ou plusieurs destinations. Ainsi, le trafic dans le réseau sera très intense, ce qui conduit à confondre entre le trafic légitime et malveillant. Pour détecter les attaques de flooding, nous proposons une nouvelle méthode qui consiste à diviser le réseau en plusieurs zônes, pour élire des nœuds spéciaux appelés Cnodes, qui gèrent le suivi du trafic dans la zône sous leur couverture. Pour diviser le réseau en zônes, nous proposons d'utiliser une technique de clustering qui permet de former des clusters composés d'un certain nombre de nœuds et de CHs. Les Cnodes sont élus en utilisant le clustering récursif chaque fois qu'un comportement anormal est suspecté à l'intérieur d'un cluster, ce qui signifie qu'il existe une détection présumée de nœud malveillant.

Chaque Cnode élu contrôle et analyse le nombre de paquets envoyés par ses voisins dans le cluster. Après un intervalle de temps, si le nombre de paquets échangés dans le réseau augmente et dépasse un certain seuil cela indique qu'un nœud malveillant envoie plus de messages (une alerte est déclenchée), les Cnodes avisent le CH pour renforcer la surveillance en augmentant le nombre de Cnodes par l'élection de plus de Cnodes dans la zône concernée. Sinon, le CH diminue le nombre de Cnodes. La méthode Accordéon consiste à augmenter ou à diminuer le nombre de Cnodes lorsque le nombre d'alertes augmente ou diminue, respectivement.

Chaque Cnode surveille les nœuds pendant un temps fini. La transmission des paquets par un nœud "i" pour $1 \leq i \leq N$ est supposée avec un taux "λ_j" ; Sa valeur est dans l'intervalle $[\lambda_0, \lambda_h]$ pour $1 \leq i \leq h$. Le trafic ´λ_0ˇ est considéré comme un trafic normal ; Entre ´λ_jˇ et ´λ_hˇ, le trafic déclenche une alerte et lorsque le trafic atteint la valeur ´λ_hˇ, il est considéré comme un trafic anormal, car il est trop élevé. Lorsque le nombre de paquets transmis atteint un seuil, le nœud est considéré comme malveillant. La durée de la surveillance par le Cnode "k" est supposée suivre un taux μ_k, pour $1 \leq k \leq K$ avec L phase. On a résumé tous les paramètres par :

- N définit le nombre de nœuds normaux.
- K est le nombre maximum de Cnodes pouvant être élus dans une période. Il commence par un nombre égal à "m", et augmente quand les alertes augmentent jusqu'à atteindre un maximum de K.
- L représente le nombre de phases qui définit la période de surveillance d'un Cnode.
- S définit le maximum du nombre d'alertes.
- d_k peut prendre une valeur $\in [m, K]$; Il définit le nombre de Cnodes pouvant être élu au cours de la période. Plus le niveau d'alerte diminue, moins de Cnodes sont élus. Plus le niveau d'alerte augmente, plus de Cnodes sont élus.

3.3 Description de la méthode proposée

En général, pour réaliser des attaques DoS dans les RCSF, les nœuds malveillants utilisent la technique de flooding en envoyant un grand nombre de messages à leurs voisins ou à la SB. Pour sécuriser les RCSF vulnérables à ces attaques, nous proposons la solution basée sur l'élection des Cnodes dans les RCSF. Cette solution recouvre l'ensemble du réseau et identifie le nombre maximum de nœuds malveillants. Cette section présente la méthode proposée (méthode Accordéon) pour sélectionner progressivement les Cnodes en fonction de l'évolution du nombre de messages envoyés (alertes suspectes) par des nœuds malveillants. La méthode Accordéon consiste à augmenter ou à diminuer le nombre de Cnodes lorsque le nombre d'alertes augmente ou diminue, respectivement.

Dans ce qui suit, pour la méthode proposée, nous considérons les hypothèses suivantes :

Les premiers Cnodes sont élus par la méthode Recursive_clustering R_leach ou R_FFUCA, avec $R = 1$.

- Z_i représente les régions i surveillées par les i Cnodes.
- Le nombre de nœuds dans la zône Z_i est noté N_{Z_i}.

- S_j représente un seuil lorsque le nombre de messages envoyés est supérieur à S_j, avec $1 \leq j \leq K$.
- P_j représente le pourcentage de Cnodes désiré (croissant ou décroissant), où ´jˇ satisfait $0 \leq j \leq m$.

K et m représentent respectivement, le seuil maximal (alerte) et le pourcentage maximal de Cnodes souhaités.

On note que le clustering récursif [62] est une méthode qui exécute récursivement un algorithme de clustering pour choisir les CH et les Cnodes en considérant la consommation d'énergie des nœuds et leur position dans le réseau. Le clustering récursif repose sur deux étapes principales :

- Etape 1 : L'ensemble du réseau est divisé en clusters où les premiers clusters et les CH sont définis.
- Etape 2 : La même opération est effectuée sur les clusters obtenus pour élire les premiers Cnodes. Le clustering récursif est appliqué jusqu'à obtention de la granularité souhaitée.

Fonctionnement de la méthode

- Etape 1 : Diviser en régions données le réseau en appliquant le Recursive_Clustering_Level_1 avec un pourcentage de Cnodes désiré égal à P_0. Le résultat est "i" Cnodes tel que $i \leq P_0$. Chaque Cnode "i" surveille la zône "i" $[Cnode_i, Z_i]$.
- Etape 2 : Calculer le nombre de messages échangés M_i dans chaque zône $[Z_i]$ surveillée par $[Cnode_i]$.
- Etape 3 : Calculer la moyenne des messages M_g échangés dans la zône $[Z_i]$
  Où $M_g = somme(M_i)/(N_{Z_i})$.
- Etape 4 : Comparer la moyenne M_g avec le seuil S_j.
- Etape 5 : Exécuter l'algorithme ci-dessous.

```
Algorithme
m ← 0
Pour j ← 1 à k faire
    Si Mg > Sj alors
        Exécuter l'étape 1 avec un pourcentage souhaité Pm+1, ou Pm+1 > Pm
        Exécuter les étapes 2, 3 et 4
            Si Mg > Sj+1 alors
                Exécuter l'étape 1 avec un pourcentage souhaité Pm+2, ou Pm+2 > Pm+1
                Exécuter les étapes 2, 3 et 4
                    Si Mg > Sj+2 alors
                        Exécuter l'étape 1 avec un pourcentage souhaité Pm+3, ou Pm+3 > Pm+2
                        Exécuter les étapes 2, 3 et 4
                    Sinon
                        Exécuter l'étape 1 avec un pourcentage souhaité Pm+3, ou Pm+3 < Pm+2
                    Fin
            Sinon
                Exécuter l'étape 1 avec un pourcentage souhaité Pm+2, ou Pm+2 < Pm+1
            Fin
    Sinon
        Exécuter l'étape 1 avec un pourcentage souhaité Pm+1, ou Pm+1 < Pm
    Fin
Fin Pour
```

3.4 Modélisation par les Réseaux d'Automates Stochastiques

- Les Réseaux d'Automates Stochastiques

De nombreux outils ont été proposés pour modéliser les systèmes tels que les réseaux de file d'attente, les réseaux de Petri, etc. Cependant, les réseaux de file d'attente sont inefficaces lorsque de nombreux évènements de synchronisation complexes doivent être pris en compte dans les systèmes à modéliser. En outre, on peut dire que pour les réseaux de Petri stochastiques qui représentent bien la synchronisation, ne génèrent généralement pas de modèles compacts, car ils construisent la matrice de transition sans aucune connaissance de ses propriétés. Les Réseaux d'Automates Stochastiques (RAS) ont été introduits comme une solution pour des systèmes complexes avec un grand nombre d'états et des synchronisations complexes. Un autre avantage de ces réseaux est qu'ils fournissent une dérivation analytique de la matrice génératrice des chaînes de Markov en utilisant une algèbre de tenseur. Ce point est particulièrement important puisqu'il nous permet de traiter des systèmes qui peuvent avoir de très grands espaces d'états.

Les RAS permettent de construire chaque sous-système par un automate et les interactions peuvent être représentées par des arêtes dirigées. Ces arêtes peuvent être des taux de transition s'ils concernent uniquement le sous-système ou des syntonisations s'ils représentent des interactions entre d'autres sous-systèmes. Deux types de transitions sont définies : transitions locales et synchronisées. Une transition locale se produit uniquement dans l'automate alors que la transition synchronisée se produit dans plusieurs automates en même temps. Ainsi, on peut dire qu'un ensemble d'automates représente un RAS associé au système réel.

Un RAS est un ensemble d'automates A_i, $i \in [1, \ldots, n]$, où n représente le nombre d'automates dans le réseau. Un automate A_i est défini par :

$$(E_i, L, Q_i)$$

où :

- E_i est un ensemble d'états.
- L est l'ensemble des étiquettes. Une étiquette l est une liste ou une fonction de (j, τ_j, p_j) où j est le nom de l'événement synchronisé, τ_j est le taux de transition et p_j la probabilité de transition.
- Q_i est la fonction de transition de l'automate A_i et elle est définie sur $E_i \times E_i \rightarrow L$.

L'automate lui-même n'est pas Markovien. L'hypothèse Markovienne est valable uniquement pour le comportement global du RAS si nous supposons une distribution exponentielle pour le déclenchement des transitions. Comme indiqué précédemment, le RAS permet de dériver automatiquement le générateur Q (en temps continu) ou une matrice de transition P (en temps discret) par une formule compacte donnée dans [60]. Le générateur est donné par la formule suivante :

$$Q = \bigoplus_{i=1}^{n} F_i + \sum_{j=1}^{c} \bigotimes_{i=1}^{n} S_{i,j} + \sum_{j=1}^{c} \bigotimes_{i=1}^{n} R_{i,j} \tag{3.1}$$

Où :

— n, représente le nombre d'automates dans le système.
— c, représente le nombre de synchronisation.
— F_i est la matrice de transition de l'automate i sans synchronisation.
— $S_{i,j}$ est la matrice de transition de l'automate i considérant uniquement la synchronisation j.
— $R_{i,j}$ est une matrice représentant la normalisation associée à la synchronisation j

sur l'automate i.

— $\bigoplus$ et $\bigotimes$ dénotent la somme et le produit tensoriels [60], où la matrice de transition est définie par :

Définition 1

Soit A une matrice d'ordre $n \times n$, et B une matrice d'ordre $p \times p$. **le produit tensoriel** de A et B est une Matrice C d'ordre $np \times np$ tel que la matrice C peut être décomposée en n^2 blocks de taille P, est définie par :

$$C = A \bigotimes B = \begin{bmatrix} a_{11}B & ... & ... & a_{1n}B \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ a_{n1}B & ... & ... & a_{nn}B \end{bmatrix} \tag{3.2}$$

A et B sont des matrices de valeurs réelles, mais la définition est généralisée du produit tensoriel sur des matrices de valeurs fonctionnelles (c'est-à-dire les éléments de A et B sont des fonctions utilisant les états comme arguments).

Définition 2

Soit A une matrice d'ordre $n \times n$, et B une matrice d'ordre $p \times p$. **la somme tensorielle** de A et B est définie par :

$$C = A \bigoplus B = A \bigotimes I_B + I_A \bigotimes B \tag{3.3}$$

Dans ce qui suit, nous donnons le RAS associé à notre système. Nous montrons comment représenter les synchronisations et nous donnons les différents automates.

Le modèle considéré

Dans cette section, nous décrivons la modélisation de notre méthode par les RAS. Nous considérons "n" nœuds (nœuds normaux) et "m" Cnodes. Chaque Cnode surveille les nœuds pendant un temps fini. La transmission des paquets par un nœud i, pour $1 \leq i \leq n$ est supposé suivre un processus de Poisson de taux λ_i. Sa valeur est dans l'intervalle $[\lambda_0, \lambda_h]$. Pour un trafic avec un taux λ_0, il est considéré comme un trafic normal, entre λ_i et λ_h, le trafic déclenchera une alerte et lorsque le trafic atteindra la valeur λ_h, il est considéré comme un trafic anormal, car il est très élevé. Lorsque le nombre de paquets transmis atteint un seuil donné, le nœud est considéré comme malveillant.

La durée de la surveillance par le Cnode k est supposée suivre une distribution d'Erlang avec taux μ_k, pour $1 \leq k \leq K$ avec L phases. Les automates associés au modèle sont décrits comme suit :

- l'automate nommé A_i pour chaque nœud i décrit le trafic des données transmises. Ainsi, N automates sont considérés. La Figure 3.1 décrit l'automate pour un seul nœud.
- l'automate noté A_c est défini pour chaque nœud de contrôle. L'automate associé à un Cnode est donné dans la Figure 3.2.
- l'automate nommé A_m représente la durée de la surveillance par Cnode et il est donné dans la figure 3.3.
- l'automate nommé A_d représente l'application de la méthode proposée qui consiste à augmenter le nombre de Cnodes lorsque le nombre d'alertes augmente et à diminuer le nombre de Cnodes lorsque le nombre d'alertes diminue. Cet automate est donné dans la Figure 3.4.
- l'automate noté A_e représente la détection (si le nœud i est malveillant ou non). Cela donne n automates (le système est composé par n nœuds). Après la période de surveillance, si le nombre de paquets de données transmis est supérieur à un seuil fixe, ce qui se décrit par le déclenchement de la synchronisation S_s. La Figure 3.5 représente un automate qui détecte si le nœud i est malveillant ou non.

Dans ce qui suit, les différentes synchronisations nécessaires sont décrites :

- $S_1, S_2, \ldots, S_{s-1}$ sont des synchronisations qui interagissent entre l'automate A_i, A_c et A_d. Ceci est utilisé pour indiquer quand le Cnode doit déclencher une alerte. Plus le niveau d'alerte augmente, plus le nombre de Cnodes augmente.
- S_s est une synchronisation qui interagit entre l'automate A_c et A_d. Cette synchronisation se déclenche après s alertes. Ceci confirme que le nœud i est malveillant.
- S_f est une synchronisation qui interagit entre tous les automates. Elle est utilisée pour la réinitialisation lorsque la période de la surveillance du Cnode est terminée.
- $S_{d_1}, S_{d_2} \ldots, S_{d_s}$sont des synchronisations qui interagissent entre l'automate A_m et l'automate A_d. Elles sont utilisées pour diminuer le nombre de Cnodes en fonction du nombre d'alertes. Plus le niveau d'alerte diminue, moins de Cnodes sont élus.

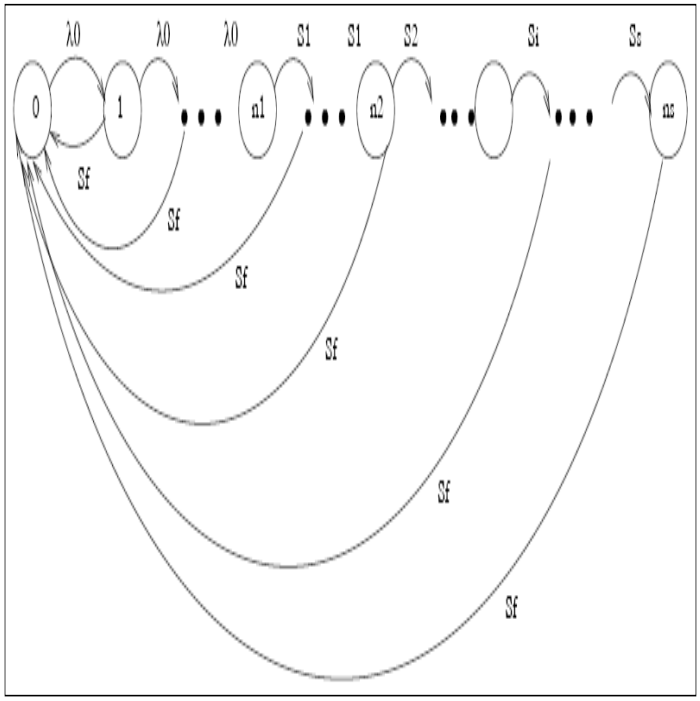


FIGURE *3.1* – Automate d'un seul nœud (transmission de données).

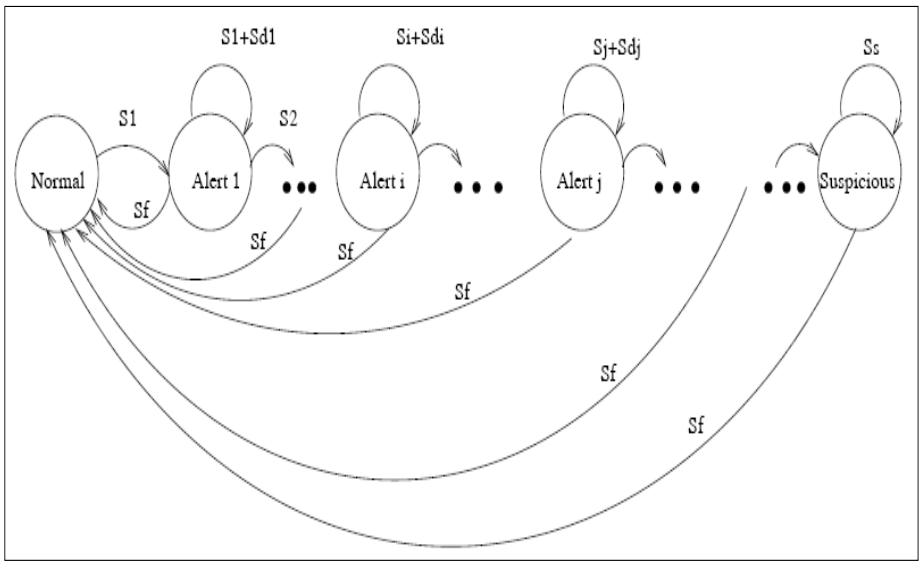


FIGURE *3.2* – Automate d'un Cnode.

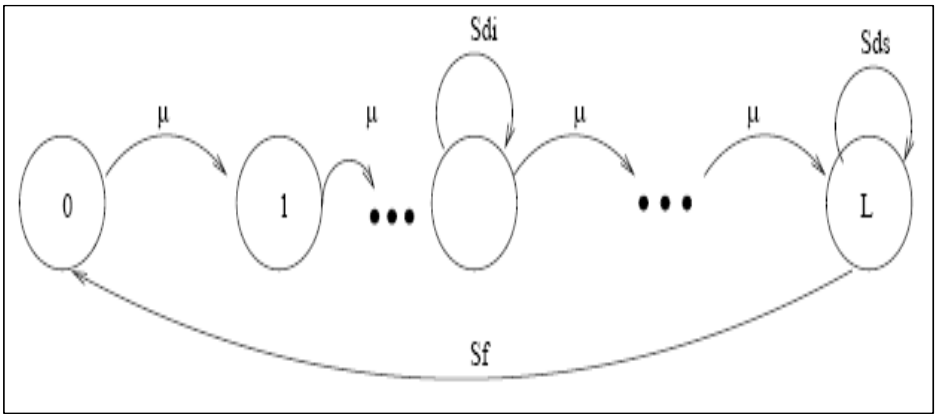


FIGURE 3.3 – Automate A_m.

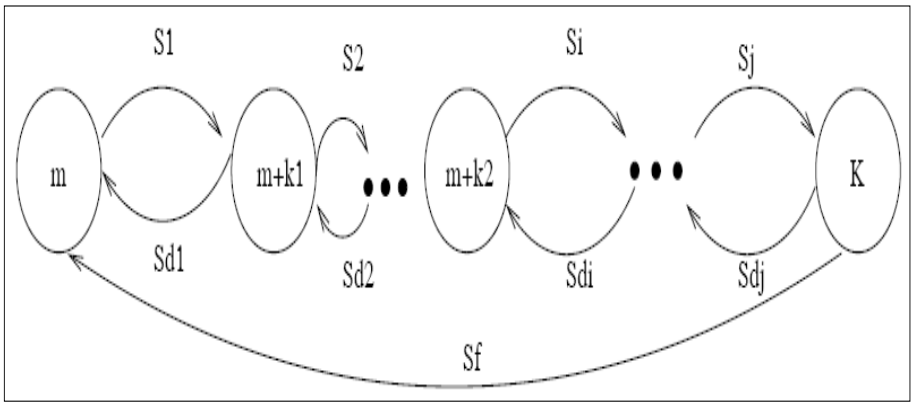


FIGURE 3.4 – Automate A_d.

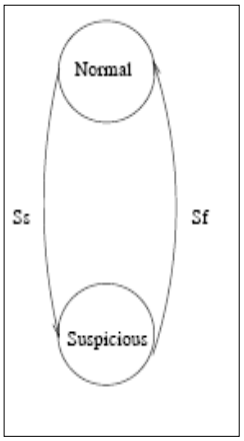


FIGURE 3.5 – Automate A_e.

Le résumé des paramètres est dans ce qui suit :

- n définit le nombre de nœuds normaux. Avant de détecter les nœuds malveillants, ils sont considérés normaux.
- K est le nombre maximum de Cnodes pouvant être élus dans une période. A l'état initial, ce nombre est fixé à m et il augmente quand les alertes augmentent jusqu'à atteindre un maximum de K.
- L représente le nombre de phases qui définissent la période de surveillance d'un Cnode.
- S définit le nombre maximum d'alertes.
- d_k définit le nombre de Cnodes qui peuvent être élu au cours de la période. Il peut prendre une valeur entre $[m, k]$. Plus le niveau d'alerte diminue, moins de Cnodes sont élus. Et inversement, l'augmentation du niveau d'alerte inclut l'élection de plus de Cnodes.

3.5 Résultats numériques

Pour évaluer et analyser les performances de la méthode proposée par rapport au taux et au temps de détection et à la durée de vie du réseau, des simulations ont été réalisées sous Matlab. En utilisant la fonction seedsrandomize, une moyenne de 30 exécutions pour chaque expérimentation a été obtenue. Dans cette étude, le modèle proposé traite le cas des attaques provenant d'un seul nœud compromis. Premièrement, nous présentons les résultats numériques de la méthode proposée par rapport au temps de détection et à la durée de vie. Deuxièmement, nous présentons une comparaison de la méthode proposée avec d'autres que nous avons proposées précédemment (les méthodes R_Leach et R_FFUCA).

3.5.1 Résultats de la méthode Accordéon

Pour l'évaluation de la méthode Accordéon, le taux de nombre de Cnodes a été varié à partir des valeurs $(5, 10, 15$ et $20\%)$, et le nombre de paquets échangés entre les nœuds a été varié, dans un intervalle de $(4000, 8000, 12000, 16000, 20000)$. Le nombre de paquets échangés représente le seuil d'alerte qui sera considéré pour augmenter ou diminuer le taux de Cnodes.

En analysant les résultats numériques représentés sur la Figure 3.6, il est montré que le temps de détection diminue sur toutes les courbes (du taux de Cnodes égal à 5% jusqu'à 20%). Par exemple, pour un seuil égal à 4000, le temps de détection de la première alerte est inférieur à 400 unités de temps, ce qui signifie qu'un nœud suspect a envoyé 4000

paquets pendant le temps de la surveillance. Lorsque ce nœud suspect augmente le nombre de paquets envoyés (8000), en utilisant 5% de taux de Cnodes, le temps de détection est inférieur à 600 unités de temps. Cependant, pour le même seuil (8000), lorsque le nombre de Cnodes augmente (10%), le temps de détection diminue (moins de 500 unités de temps). Ainsi, lorsque le seuil augmente, il est préférable d'augmenter le nombre de Cnodes pour la détection précoce des nœuds malveillants.

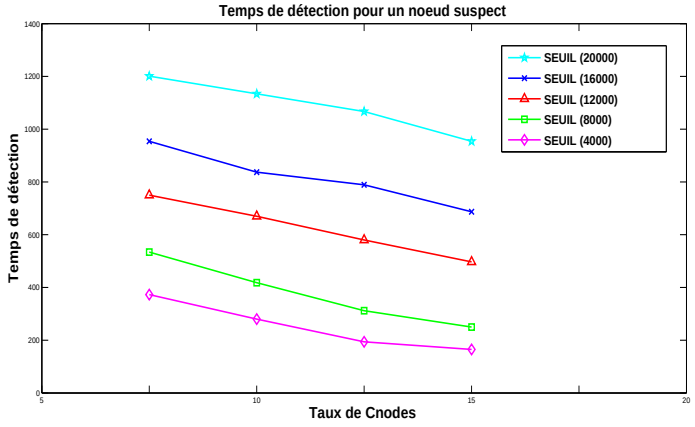


FIGURE *3.6* – Temps de détection pour un seul nœud.

En outre, les Figures 3.7 à 3.11 représentent le temps de détection pour différents seuils $(4000, 8000, 1200, 16000$ et $20000)$ et un taux de nœuds suspects différents $(1, 2, 3$ et $4\%)$. Nous avons noté que, par exemple, dans la Figure 3.7 (seuil 4000 paquets), en utilisant 5% des Cnodes (courbe 1 avec le taux Cnodes noté $ACC5\%$), le temps de détection du premier nœud suspect est inférieur à 400 unités de temps, alors qu'il est de 410 pour le deuxième nœud suspect. Cependant, le temps de détection augmente pour le troisième (493) et le quatrième nœud suspect (583). Lorsque le nombre de Cnodes est augmenté (voir la courbe 4 avec le taux de Cnodes $ACC20\%$), le temps de détection du premier nœud suspect est de 165 unités de temps, alors qu'il est de 180 unités de temps pour le deuxième nœud suspect. Ainsi, le temps de détection augmente pour le troisième et le quatrième nœud suspect.

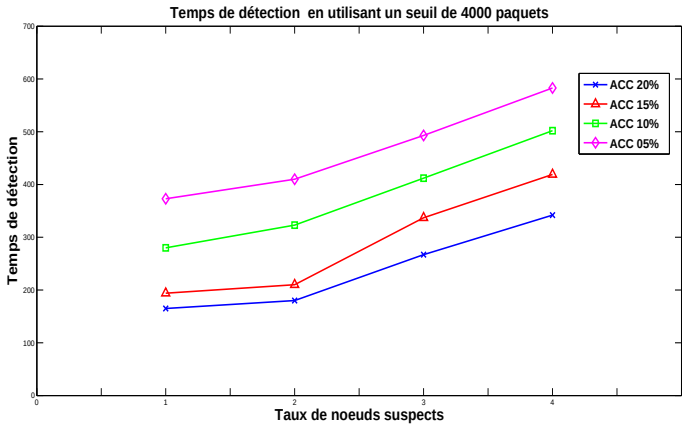


FIGURE *3.7* – Temps de détection pour un seuil de 4000 paquets.

Nous avons noté aussi que dans les Figures 3.8 à 3.11 qui représentent le temps de détection pour différents taux de Cnodes (5, 10, 15 et 20%) et différents seuils (8000, 1200, 16000 et 20000), nous avons observé le même comportement. Lorsque le nombre de Cnodes et le seuil d'alerte augmentent, le temps de détection diminue du premier au quatrième nœud suspect.

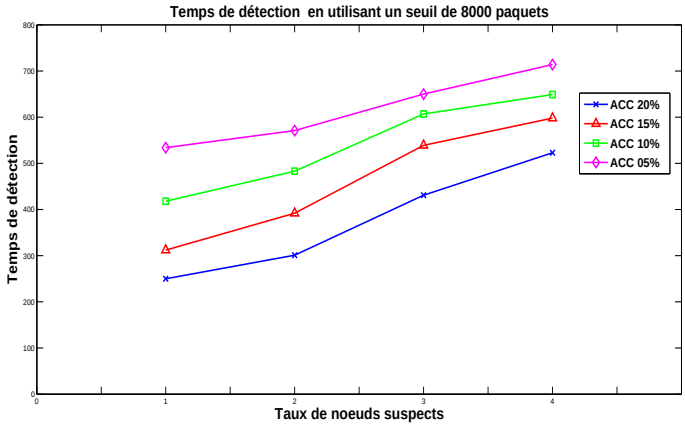


FIGURE *3.8* – Temps de détection pour un seuil de 8000 paquets.

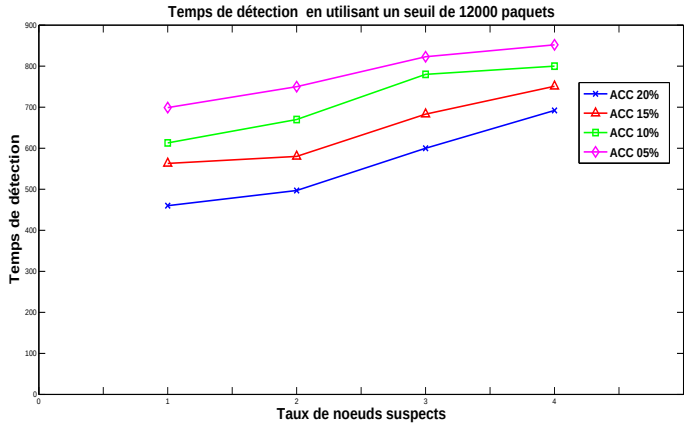


FIGURE *3.9* – Temps de détection pour un seuil de 12000 paquets.

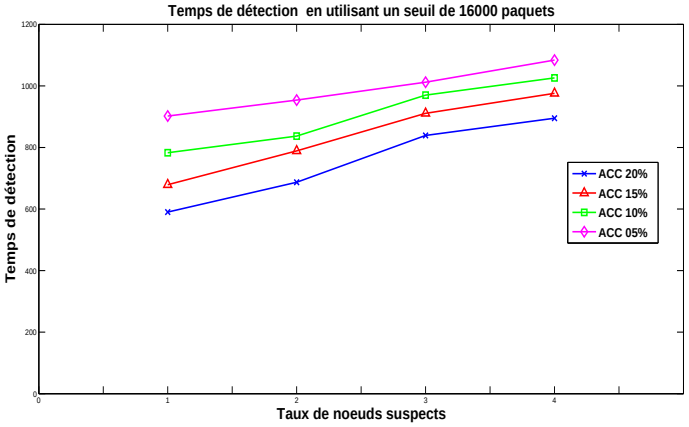


FIGURE *3.10* – Temps de détection pour un seuil de 16000 paquets.

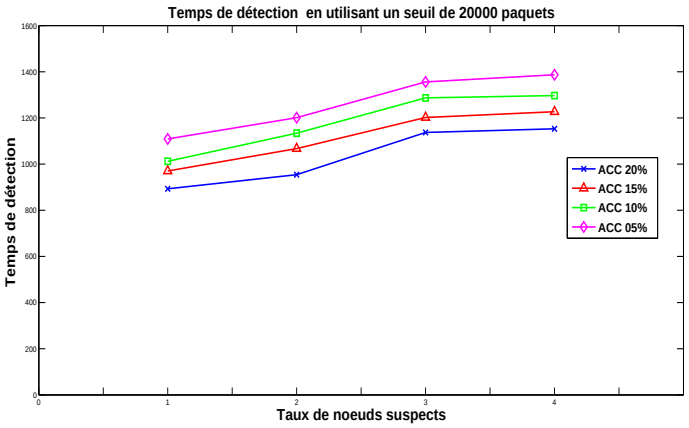


FIGURE *3.11* – Temps de détection pour un seuil de 20000 paquets.

La Figure 3.12 représente l'évolution de la durée de vie des nœuds. Nous avons constaté que pour un taux de Cnodes égal à 5%, le premier nœud perd toute son énergie (mort) dans un temps égal à 1294 unités de temps et le dernier nœud est mort dans un temps égal à 2213 unités de temps. Lorsque le nombre de Cnodes est augmenté à 10%, le premier nœud est mort dans un temps égal à 1068 unités de temps et le dernier nœud est mort en un temps égal à 2069 unités de temps. Ainsi, la durée de vie du nœud diminue lorsque le nombre de Cnodes augmente. Dans ce cas, nous avons clairement remarqué que lorsque le nombre de Cnodes diminue, la durée de vie augmente.

Par conséquent, pour des applications spécifiques, susceptible de subir des attaques DoS, il est judicieux de déployer des Cnodes de manière graduelle, en fonction de l'évolution du taux de messages envoyés par les nœuds malveillants. Ainsi, fixer à l'avance le nombre de Cnodes n'est pas très bénéfique pour prolonger la durée de vie des réseaux de capteurs.

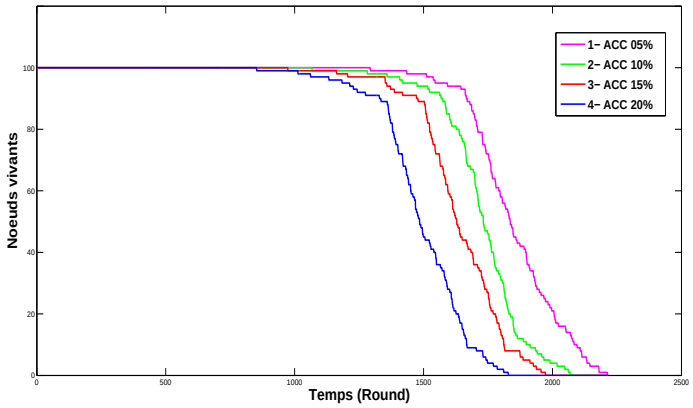


FIGURE *3.12* – Evolution de la durée de vie des nœuds.

3.5.2 Comparaisons des résultats

En considérant la durée de vie et le temps de détection des nœuds malveillants, les trois méthodes suivantes ont été comparées : la méthode Accordéon avec différents taux de Cnodes, la méthode R_Leach et la méthode R_FFUCA.

Dans la Figure 3.13 qui représente le temps de détection pour différents taux de Cnodes, en utilisant la méthode Accordéon et les méthodes R_Leach (niveau_1) et R_FFUCA (niveau_1), la méthode Accordéon avec 20% de Cnodes (voir la courbe notée $ACC20\%$) donne de très bons résultats en termes de temps de détection par rapport aux méthodes R_FFUCA et R_Leach. Par exemple, pour un seuil égal à 4000, $ACC20\%$ détecte la

première alerte en un temps égal à 165 unités de temps, tandis que pour R_FFUCA et R_Leach, les temps de détection sont respectivement 300 et 240 unités de temps . La même conclusion pour le temps de détection de la deuxième à la cinquième alerte peut être faite.

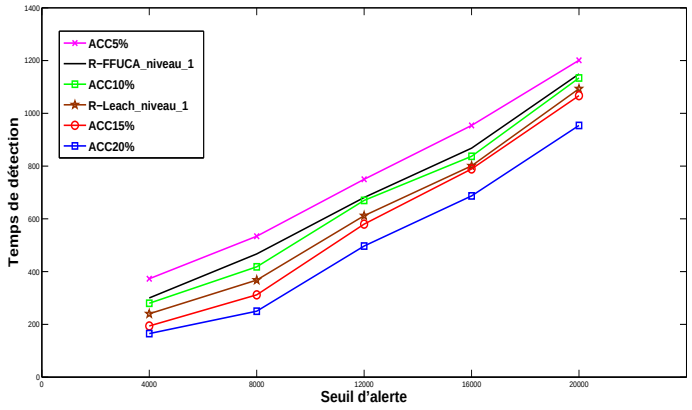


FIGURE *3.13* – Temps de détection niveau_1.

La Figure 3.14 représente le temps de détection pour différents taux de Cnodes en utilisant les méthodes Accordéon, R_Leach (niveau_2) et R_FFUCA (niveau_2). Il est montré que la méthode R_FFUCA est nettement meilleure en temps de détection que la méthode R_Leach et $ACC20\%$. Par exemple, pour un seuil égal à 4000, R_FFUCA détecte la première alerte en un temps égal à 78 unités de temps, tandis que dans R_Leach et $ACC20\%$, le temps de détection est respectivement 101 et 165 unités de temps. Ainsi, nous avons observé les mêmes résultats pour le temps de détection de la deuxième à la cinquième alerte.

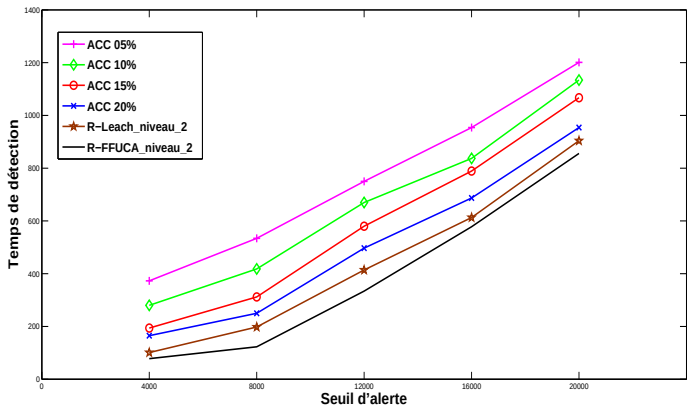


FIGURE *3.14* – Temps de détection niveau_2.

Aussi, la Figure 3.15 qui représente le temps de détection pour différents taux de Cnodes en utilisant les méthodes Accordéon, R_Leach (niveau_3) et R_FFUCA (niveau_3) donne les mêmes conclusions que la Figure 3.14. Cependant, il est clairement montré que la courbe qui représente R_FFUCA est au-dessous de celle représentant R_Leach et $ACC\%20$.

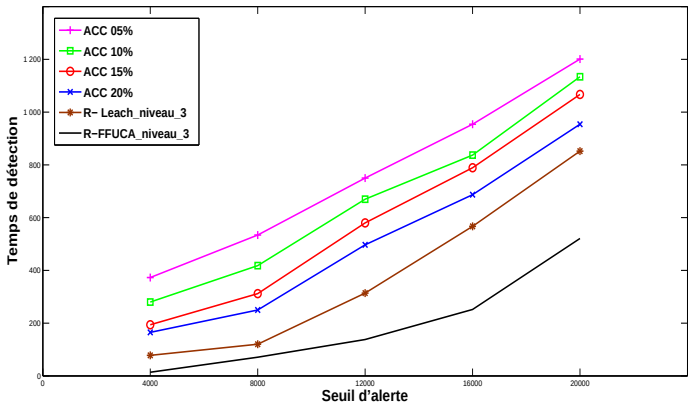


FIGURE *3.15* – Temps de détection niveau_3.

La Figure 3.16 représente l'évolution des nœuds vivants pour les différentes méthodes. Nous remarquons que lorsque le nombre de Cnodes augmente, le nombre de nœuds vivants diminue. Par conséquent, les Cnodes consomment plus d'énergie à chaque fois pour analyser les messages provenant de leurs voisins. Nous avons constaté que les nœuds meurent beaucoup plus rapidement dans la méthode Accordéon avec un taux de Cnodes égal à 20%($ACC20\%$) en comparaison avec les méthodes R_FFUCA et R_Leach.

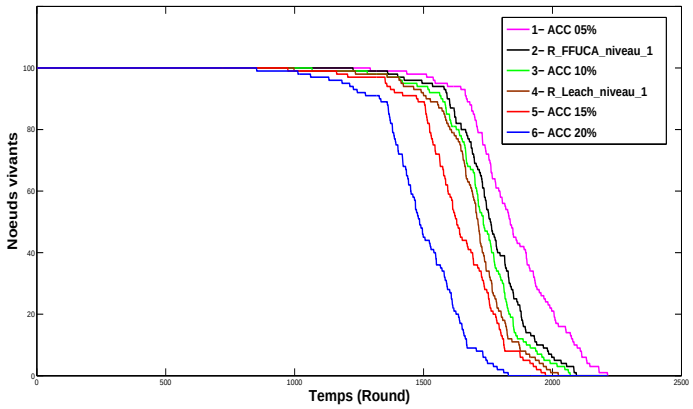


FIGURE *3.16* – Evolution de la durée de vie des nœuds niveau_1.

Ainsi, dans $ACC20\%$, le premier nœud meurt après 853 unités de temps et le dernier nœud périt après 1828 unités de temps. Alors que dans R_FFUCA, le premier nœud

meurt après 1227 et le dernier nœud meurt après 2093 unités de temps. Pour R_Leach, le premier nœud meurt après 974 unités de temps et le dernier nœud meurt après 2023 unités de temps.

Comme dans la Figure 3.16, la Figure 3.17, montre qu'en augmentant le nombre de Cnodes, le nombre de nœuds vivants diminue. Dans ce cas, dans la méthode R_FFUCA ($level\_2$), les nœuds meurent rapidement (le premier nœud meurt après 511 unités de temps et le dernier nœud meurt après 1521 unités de temps) que dans la méthode R_Leach (niveau_2) (le premier nœud meurt après 784 unités de temps et le dernier nœud périt après 1778 unités de temps) et la méthode Accordéon ($5, 10, 15$ et 20%). Ainsi, par rapport à la durée de vie des nœuds, la méthode proposée permet d'étendre la durée de vie.

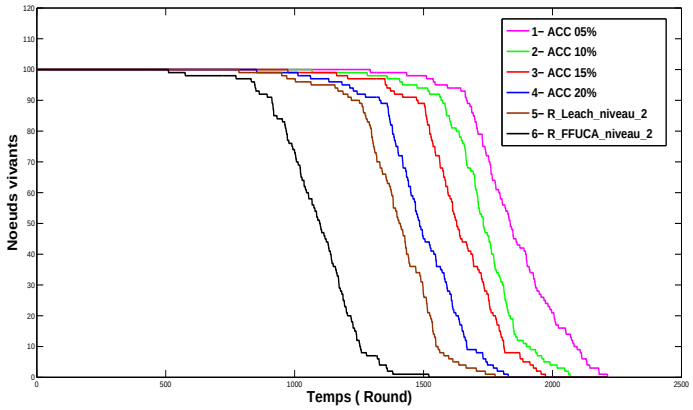


FIGURE *3.17* – Evolution de la durée de vie des nœuds niveau_2.

Dans la Figure 3.18 qui représente les méthodes R_Leach (niveau_3), R_FFUCA (niveau_3) et Accordéon, le comportement reste également similaire.

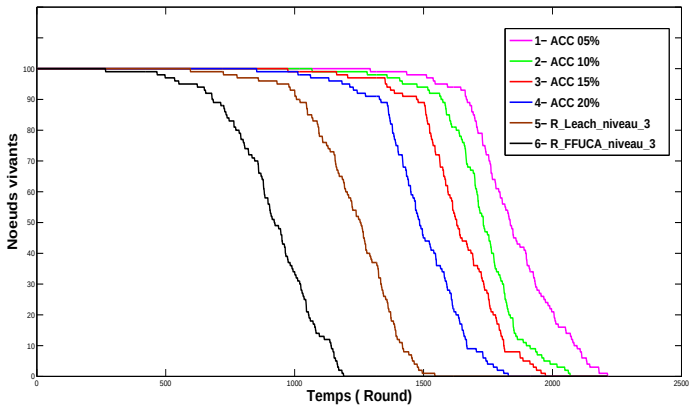


FIGURE *3.18* – Evolution de la durée de vie des nœuds niveau_3.

3.5.3 Discussion

Il est à noter que les méthodes R_FFUCA, R_Leach et Accordéon permettent l'élection de Cnodes pour détecter rapidement les attaques des nœuds compromis. Les résultats de la simulation ont montré que les méthodes R_FFUCA et R_Leach choisissent les Cnodes de façon excessive et sans tenir compte de l'évolution des alertes (le nombre de paquets transmis par un nœud suspect). Par conséquent, cela induit une augmentation de la consommation d'énergie dans les capteurs (chaque Cnode consomme plus d'énergie à chaque fois qu'il analyse les messages venant de ses voisins). En utilisant les méthodes R_FFUCA et R_Leach, la durée de vie du réseau diminue et les Cnodes élus peuvent mourir sans détecter des nœuds compromis. La méthode Accordéon qui est basée sur l'augmentation (ou la diminution) progressive du nombre de Cnodes en fonction de la croissance (ou décroissance) des alertes permet une détection plus rapide des nœuds suspects en assurant une consommation d'énergie rationnelle dans le réseau.

3.6 Conclusion

Dans ce chapitre, nous avons proposé une approche dynamique et adaptative de la détection du flooding dans les réseaux de capteurs. Cette méthode permet d'élire les nœuds de contrôle (Cnodes) qui sont utilisés pour détecter les nœuds compromis susceptibles d'inonder les réseaux de capteurs par des attaques DoS [79]. L'approche proposée permet d'élire progressivement les Cnodes, en fonction du taux de messages envoyés par les nœuds compromis. Lorsqu'un nœud transmet à son voisin un certain nombre de messages dépassant un certain seuil, il sera considéré comme un nœud suspect. Dans ce cas, d'autres Cnodes doivent être élus. Si le seuil des messages augmente également, un autre nombre de Cnodes sont également élus, et ainsi de suite. Dans le cas contraire, le nombre de Cnodes est réduit. Aussi, une modélisation de la méthode proposée par les Réseaux d'Automates Stochastique a été décrite dans ce chapitre. A travers les résultats numériques, l'évaluation des performances a montré que son utilisation est très intéressante en termes de temps de détection et de durée de vie du réseau en comparaison avec d'autres méthodes de détection.

En outre, comme indiqué précédemment, la conception des réseaux de capteurs nécessite par sa complexité des modèles formels qui permettent leur modélisation et leur analyse des performances, en vue de vérifier et de minimiser leur défauts de conception, d'évaluer l'influence de l'environnement dans lequel ils sont déployés et d'estimer leur consommation énergitique. Dans le chapitre suivant, nous présenterons un aperçu sur les différents travaux utilisant les modèles formels pour modéliser les réseaux de capteurs. En utilisant les réseaux

de Petri Colorés Stochastiques Généralisés, nous présenterons également une modélisation d'une méthode de détection des attaques DoS dans les RCSF, et d'une stratégie de défense contre les attaques actives dans ces réseaux.

Chapitre 4

MODELISATION D'UNE METHODE DE DETECTION DES ATTAQUES DoS ET D'UNE STRATEGIE DE DEFENSE CONTRE LES ATTAQUES ACTIVES DANS LES RCSF

Sommaire

4.1 Contexte et objectifs

La conception des réseaux de capteurs n'est pas aussi simple par rapport à leur spécificités particulières, tels que, la fiabilité des communications entre les nœuds (dues à l'utilisation de liaisons radio sans fil) et les ressources (calcul, mémoire) très limités parce qu'ils doivent respecter des contraintes de coût. Aussi, les concepteurs des RCSF doivent tester et analyser les performances de ces réseaux avant leur mise sur pied et leur déploiement, afin de s'assurer de leur bon fonctionnement et d'estimer de façon fiable la consommation d'énergie. Généralement, deux principales techniques sont utilisées pour tester et analyser les performances des réseaux sans fil, les techniques de simulation informatiques et les méthodes d'analyse formelle.

La simulation représente la technique la plus utilisée pour analyser et évaluer les réseaux de capteurs, et ce, en fournissant des informations sur leur comportements, leur évolutions et leur fonctionnements, ce qui permet d'améliorer et de diminuer au maximum les erreurs de conception. Aussi, la simulation permet d'économiser le coût, le temps et de vérifier les corrections apportées sur les RCSF, d'optimiser l'utilisation de leurs ressources et d'augmenter leur tolérance aux pannes et leur durée de vie.

Par contre, les techniques formelles permettent de modéliser et d'évaluer des propriétés quantitatives des réseaux de capteurs modélisés à l'aide de modèles formels tels que, les chaînes de Markov, les files d'attentes et les réseaux de Petri.

Dans ce chapitre, nous présentons, les exigences de la modélisation et de la simulation des résaux de capteurs avec quelque travaux de modélisation proposés par des chercheurs du demaine. Ensuite, nous introduisons notre contribution dans ce contexte à travers la modélisation par les réseaux de Petri Colorés Stochastiques Généralisés, d'une part, la détection des attaques de Déni de Service (DoS). D'autre part, une stratégie de défense contre les attaques actives dans ces résaux.

4.2 La simulation et la modélisation des RCSF

4.2.1 Exigences de la simulation et de la modélisation des RCSF

La simulation et la modélisation des réseaux de capteurs doivent considérer les caractéristiques particulières de ces réseaux introduites dans le premier chapitre et les exigences des différentes étapes de conception des RCSF (par exemple, la conception des protocoles de

communication et des applications). Dans ce cadre, on peut citer les principales exigences suivantes :

- **La fidélité** : Le but principal de la simulation est de reproduire fidèlement le système du monde réel et de prédire son comportement. Pour les RCSF, il faut avoir une précision des modèles de communications radio et de l'environnement physique. Une simulation imprécise peut conduire à des conclusions erronées.
- **L'évolutivité** : Généralement, les nœuds des RCSF sont souvent déployés en grandes quantités, le simulateur devrait bien supporter l'évolutivité avec un temps de simulation raisonnable.
- **La consommation d'énergie** : Sachant que l'alimentation énergétique des nœuds de capteurs est très limitée, les concepteurs de réseaux doivent obtenir des estimations précises sur la consommation de l'énergie, pour régler leurs applications avant leurs déploiements dans des environnements réels.
- **L'extensibilité** : Il est facile de modifier les modules existants ou d'intégrer de nouveaux modules dans un RCSF. Une structure avec une grande modularité permet aux utilisateurs d'ajouter ou de modifier facilement des fonctionnalités.
- **La prise en charge de l'hétérogénéité** : Les RCSF sont des systèmes hétérogènes, incorporant différents types de nœuds avec des capacités très variées. Toutefois, la modélisation de différents types de nœuds et la gestion des interconnexions sont nécessaires dans les simulations des RCSF.
- **La facilité d'utilisation** : Une interface utilisateur graphique peut faciliter et accélérer l'établissement de la topologie du réseau et la composition des modules de base. Elle peut également permettre la visualisation rapide des résultats de la simulation.

4.2.2 La simulation dans les réseaux de capteurs sans fil

Les simulateurs permettent de réaliser des expérimentations assez réalistes sur les RCSF pour étudier leur comportement en termes de déploiement, de routage, de topologie, de mobilité, d'environnement et de contraintes temps réel. Les simulateurs permettent ainsi, d'anticiper sur plusieurs paramètres d'un RCSF pour porter des modifications ou des ajustements. La communauté scientifique a conçu des simulateurs de RCSF de différents types qui intègrent un certain nombre d'outils permettant de simuler, de contrôler et d'étudier le comportement de plusieurs propriétés spécifiques liées aux RCSF. Ainsi, les simulateurs les plus célèbres et les plus utilisés pour la simulation des réseaux de capteurs sont :

* **NS-2** [80].
* **OMNeT**(Objective Modular Network Test in C++) [81].
* **GloMoSim** : Global Mobile Information system Simulate [82].
* **WSNet** [83].
* **J-Sim**[84].
* **TOSSIM** [85].

4.2.3 La modélisation des réseaux de capteurs sans fil

Les techniques de modélisation des réseaux de capteurs sont relativement récentes. Dans cette section, nous présentons un aperçu des travaux présentés dans la littérature, ayant traité la modélisation et la discription formelle des différents aspects des réseaux de capteurs.

1. **Modélisation et étude des performances d'un réseau de nœuds collecteurs dans un réseaux de capteurs à grande dimension**

   Les auteurs ont traité dans [86] la problématique du passage à l'échelle des RCSF qui ne doit pas dégrader les performances du réseau. En associant, la grande dimension et la taille de ce type de réseaux, il est trés important de prédire son bon fonctionnement et d'estimer ses performances de qualité de service avant son déploiement. Dans [86], les auteurs ont présenté la modélisation et l'étude de performances d'un réseau de nœuds collecteurs pour déterminer les limites de fonctionnement d'une architecture de communication pour les réseaux de capteurs sans fil de trés grande dimension (jusqu'à un million de nœuds). Cette modélisation permet aux concepteurs de ce type de réseaux, d'effectuer une configuration optimale des paramètres tels que, la fréquence de génération des données, la densité du réseau, la taille des paquets de données, le nombre de nœuds sources dans un cluster et le nombre de nœuds collecteurs. Une étude analytique a été effectuée pour différents scénarios afin de déterminer les valeurs limites des paramètres pour lesquelles le système reste stable. Les résultats analytiques obtenus ont été comparés à ceux trouvés par simulation. Les auteurs ont présenté une modélisation par file d'attente de l'architecture de communication à deux niveaux avec différents canaux. Cette modélisation a été développée pour déterminer les limites liées au mécanisme de collecte des données de cette architecture, où chaque nœuds collecteur collecte périodiquement les mesures des capteurs de son cluster et les achemine via ses voisins à un collecteur final ou à une station de base.

Chaque nœuds collecteur possède deux récepteurs, l'un représente le premier niveau du réseau, il est composé des nœuds de capteurs statiques regroupés en clusters, où les nœuds collecteurs sont représentés par des cluster-heads (liaison entre le CH et les différents nœuds appartenant au cluster considéré). Par contre, le second récepteur représente le second niveau qui correspond au réseau formé par l'ensemble des nœuds collecteurs (liaison entre les différents CHs et la station de base).

Pour évaluer les performances de ce type de réseaux, initialement, les auteurs [86] ont estimé la charge locale d'un nœud collecteur, en calculant le taux moyen de paquets générés par ces nœuds et le taux de service des nœuds collecteurs reliés directement à la station de base. Une étude analytique a été effectuée pour estimer la charge maximale supportable par les nœuds collecteurs pour différents scénarios (l'impact de la période de mesure, l'impact de la taille des paquets et l'impact du nombre de nœuds collecteurs). Enfin, les résultats de cette étude analytique ont été comparés aux résultats de la simulation réalisée sur la plate-forme OPNET. Ce travail a permis aux auteurs d'identifier les limites de l'architecture et de fixer les valeurs maximales des paramètres (la taille des paquets, leur fréquence d'envoi par les capteurs et la taille du réseau) pour lesquelles l'ensemble du trafic généré ne constitue pas une limite pour le bon fonctionnement du RCSF.

2. **Estimation de la consommation énergétique des réseaux de capteurs sans fil hiérarchiques en utilisant le modèle de file d'attente M/M/1**

Dans [87], les auteurs ont proposé un modèle analytique pour l'estimation de la consommation de l'énergie dans les RCSF hiérarchiques en utilisant les files d'attentes M/M/1. Le modèle proposé represente un schéma de réduction de la consommation de l'énergie des nœuds de capteurs en réduisant le nombre de transitions entre les états actifs et inactifs de ces nœuds en fonction du nombre de paquets de données dans la file d'attente. Le RCSF considéré est basé sur une architecture hiérarchique, où l'ensemble des nœuds sont regroupés en clusters et sont uniformément répartis dans un environnement donné. Le nœud Cluster-Head collecte les données des nœuds de capteurs, alors qu'un nœud puits positionné au centre du RCSF, reçoit les données transmises par les nœuds CH. Le modèle analytique proposé utilise une file d'attente M/M/1 pour estimer la consommation totale d'énergie des nœuds de capteurs. Ce modèle analytique est basé sur les principaux paramètres, tels que, la consommation moyenne en énergie et le nombre de paquets transmis permettant de réduire la consommation d'énergie.

Le modèle suppose que les membres du cluster collectent périodiquement des données de l'environnement et les envoient aux nœuds CHs. L'arrivée des paquets de données sur les capteurs est supposée suivre un processus de Poisson avec une fréquence d'arrivée moyenne λ_{C_M} et selon la politique FIFO. Pour calculer l'énergie consommée pour transmettre des données au nœud récepteur, chaque nœud membre, chaque nœud CH et chaque nœud puits est modélisé par une file d'attente M/M/1. Pour plus d'économie d'énergie, les auteurs proposent un schéma de réduction de la consommation d'énergie dans tous les nœuds de capteurs du réseau, en réduisant le nombre de transitions entre les états actifs et inactifs des nœuds de capteurs. L'objectif visé correspond à la réduction du nombre de transitions entre l'état actif et l'état inactif des capteurs en fonction du nombre de paquets de données dans la file d'attente, ce qui entraîne une réduction de la consommation de l'énergie.

- Le modèle analytique de la consommation d'énergie

Dans le modèle analytique, les transitions de l'état actif à l'état inactif (et inversement) consomment la majeure partie de l'énergie dans les RCSF. Si le nombre de commutateurs entre ces états est réduit, la consommation d'énergie est réduite aussi. L'organigramme de la Figure 4.1 [87] montre le processus de commutation entre les nœuds capteurs dans les deux états (actif et inactif). La variable compteur permet de calculer le nombre de paquets qui peuvent être présents dans la file d'attente des nœuds de capteurs. Ces derniers restent en état de repos jusqu'à ce qu'ils reçoivent B paquets et passent ensuite à l'état actif. dans cet 'état, les nœuds de capteurs transmettent ou reçoivent des paquets jusqu'à ce que le compteur soit nul. Lorsque le compteur est à zéro, les nœuds de capteurs passent à l'état de repos. Dans cette technique analytique, le nœud de capteur reste dans l'état inactif jusqu'à réception de B paquets et passe ensuite à l'état actif.

Pour valider leur modèle analytique, les auteurs ont effectué des simulations pour différents scénarios en changeant le taux moyen des arrivées des paquets et en modifiant également le nombre de cluster-head. En mesurant la consommation moyenne en énergie de tous les nœuds de capteurs et la consommation moyenne en énergie des CHs, les résultats de la simulation du modèle analytique ont montré que la réduction du nombre de transitions a un impact significatif sur l'économie d'énergie. Il est à noter que le modèle analytique présenté dans [87] convient aux applications tolérantes aux délais et aux architectures hiérarchiques, néanmoins, il n'est pas très efficace pour les architectures plates, ou pour des applications temps réels.

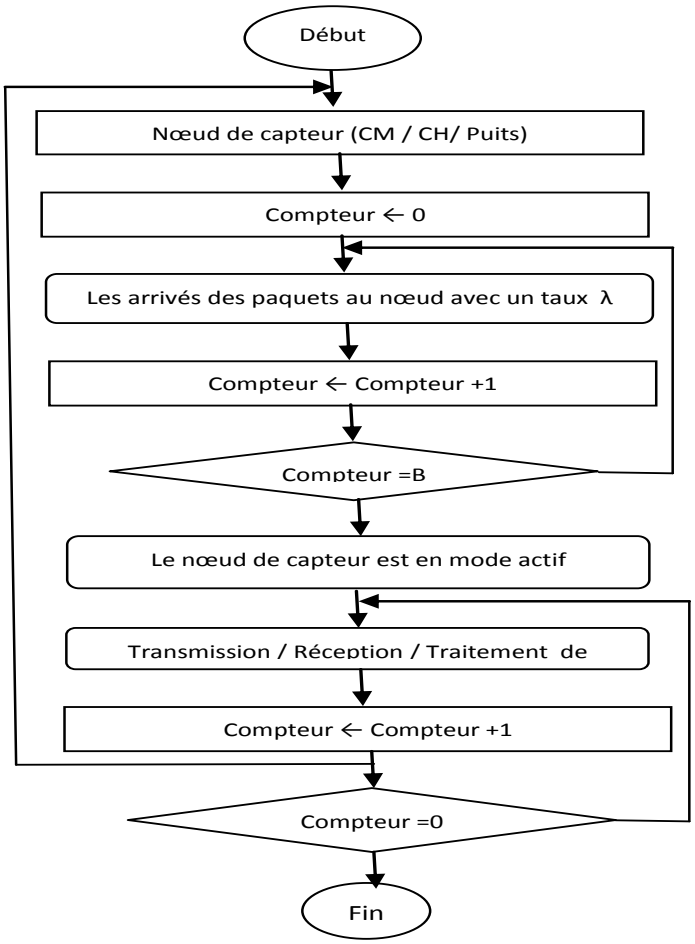


FIGURE *4.1* – Le processus de commutation du mode actif au mode inactif.

3. Validation des propriétés de qualité de service des réseaux de capteurs biomédicaux basée sur le modèle UPPAAL

Tschirner et al. [88] ont proposé la modélisation d'un réseau de capteurs sans fil biomédical en utilisant UPPAAL [89], [90]. Les auteurs ont considéré un réseau de capteurs sans fil biomédical qui fait la collecte de données, constitué de nœuds équipés d'un émetteur/récepteur utilisant le protocole de routage de diffusion contrôlée. La diffusion contrôlée consiste à rediffuser pour chaque nœud un message uniquement la première fois où il le reçoit, pour éviter les boucles de rediffusions infinies. La topologie du réseau est modélisée par une matrice A, où l'élément $A[i, j]$ ayant une valeur positive indique la force du signal entre le nœud "i" et "j", une valeur négative indique l'absence de connectivité.

Le RCSF est modélisé sous la forme d'un réseau d'automates temporisés, où chaque automate représente le comportement d'un nœud. Pour modéliser la diffusion contrôlée, une matrice contenant les identifiants des paquets déjà reçus est définie. La même matrice est utilisée pour indiquer si un acquittement a été reçu ou pas. Les collisions sont modélisées à l'aide d'un tableau représentant l'état du signal reçu par chaque nœud. Si le nœud commence une transmission alors qu'un de ses voisins reçoit un signal, l'élément du tableau correspondant est mis à une valeur négative signifiant la corruption du paquet.

Cette modélisation a permis à Tschirneret et al. de vérifier trois propriétés de qualité de service à savoir :

— La connectivité du réseau.
— Le taux de paquets reçus par la station de base qui représente la proportion entre le nombre de paquets envoyés par les nœuds, et le nombre de paquets qu'elle a reçus.
— Le délai de bout en bout qui représente le temps écoulé entre l'émission d'un paquet par un nœud et la réception de ce paquet par la station de base.

Bien que la modélisation présentée dans cette étude a permis de vérifier certaines propriétés de qualité de service pour les réseaux de capteurs biomédicaux sans fil, néanmoins, la consommation en énergie qui est un aspect très important dans les réseaux de capteurs n'a pas été modélisé.

4. **Evaluation de la perte de paquets en utilisant les réseaux de Petri Stochastiques Déterministes**

   Les auteurs ont proposé une approche d'évaluation des performances des RCSF hiérarchiques par une description formelle basée sur l'utilisation des Réseaux de Petri Stochastiques Déterministes (RdPSD) pour évaluer la perte de paquets dans ces réseaux [91]. En effet, avant d'implémenter les protocoles de routage, il est nécessaire de considérer cette métrique afin de minimiser le nombre de paquets de données perdus.

   Les auteurs ont opté pour les RdPSD, car, ces derniers modélisent bien quelques tâches effectuées par les nœuds de capteurs qui peuvent durer dans le temps (un temps prédéfini (temps de détection)) ou bien s'exécuter dans des temps aléatoires (comme les opérations d'émission / réception). A travers les RdPSD, les temps prédéfinis et aléatoires sont modélisés respectivement par des transitions déterministes

et exponentielles et les paquets envoyés et reçus sont modélisés par deux places distinctes. Ces deux places agissent comme des compteurs qui calculent le nombre de paquets envoyés et reçus. Le taux de perte de paquets représente le ratio des paquets envoyés/les paquets reçus.

Le modèle de réseaux de capteurs considéré dans [91] est basé sur une topologie hiérarchique, les nœuds de capteurs sont organisés en cluster où chaque cluster est géré par un seul nœud appelé Cluster-Head (CH). Les autres nœuds sont appelés membres. Pendant un intervalle de temps appelé round, chaque membre récupère des informations de son environnement et les envoient à son CH. Chaque CH agrège les données reçues et envoie le résultat final à la station de base.

- Modélisation de l'activité d'un cluster

Le modèle RdPSD associé à l'activité d'un cluster est représenté dans la Figure 4.2 [91]. Dans ce modèle, le début du round est déterminé par l'état initial "Init". La transition immédiate "StartRound" est activée (elle a la priorité la plus élevée), vu que l'action effectuée par le nœud capteur est déclanchée. Les deux nœuds membres $MBR1$ et $MBR2$ sont prêts pour détecter les données de leur environnement. Aprés détection, pour envoyer les données, les transitions $Sense1$ et $Sense2$ (qui suivent une distribution exponentielle avec un taux SenRate) sont tirées jusqu'à un temps "$1/SenRate$". Les transitions "S1CH" et "S2CH"(les nœuds membres 1 et 2 envoient leur propre données à leur CH) sont activées selon leur temporisation (SenRate1) et (SenRate2). Deux jetons sont introduits respectivement, un dans les places "RD1" et "RD2" (les données sont transmises) et un autre dans la place "SentD" (compteur de paquets envoyés). Ce dernier agit comme un compteur de messages envoyés par les deux membres. Ainsi, il gardera la trace du nombre de paquets envoyés des membres à leur CH.

Les transitions "TransitD1" et "TransitD2" (Le CH reçoit les données des nœuds membres 1 et 2) peuvent être déclenchées ou pas, cela dépend des paquets envoyés des membres 1 et 2, s'ils n'ont pas été perdus en transit. Par conséquent, la place "Data" (les données des membres ont probablement été reçues et sont en attente d'agrégation) peut contenir au maximum deux jetons. Le CH n'attendra pas un temps infini pour lancer l'opération d'agrégation des données. La transition "Aggr" est tirée après l'écoulement du temps qui lui est associé. Le tir de transition "Aggr" se produit quand il y'a au moins un jeton dans le place "Data". Pendant l'opération

d'agrégation, le CH incrémente un autre compteur representé par la place "AggrD". Ce compteur calcule le nombre de paquets reçus. S'il n'y a pas de jeton dans la place "Data", la transition déterministe "S-BS" est activée (présence d'un arc inhibiteur), ce qui implique l'envoie des données à la station de base. Par conséquent, les nœuds seront réinitialisés, ce qui indique la fin du round. Le même algorithme sera répété pour chaque round.

- Evaluation des taux de perte de paquets

Au vu de la non-bornitude du modèle de RdPSD (les deux places "SentD" et "AggrD" admettent un nombre infini de jetons), l'état stationnaire n'est pas atteint et l'analyse numérique du comportement stationnaire n'a pas été réalisée. Néanmoins une analyse numérique transitoire du RdPSD a été effectuée et des mesures du taux de perte de paquets ont montré qu'il n'y a pas de perte de paquets élevée pour ce type de topologie de réseaux de capteurs [91].

Bien que l'approche proposée permet de faire des évaluations de performances sur les réseaux de capteurs hiérarchiques, la prédiction du comportement de ce modèle pour d'autres topologies de réseaux de capteurs (topologie plate ou hybride) n'est pas possible. Aussi, le modèle proposé ne considère pas les facteurs qui influent sur l'augmentation du taux de perte des paquets tels que, les distances, le type d'application, l'impact de l'environnement.

5. **Analyse du flux de données par les Réseaux de Petri Stochastiques Généralisés**

Lacerda et al.[92] proposent l'utilisation des Réseaux de Petri Stochastiques Généralisés pour analyser plusieurs propriétés des réseaux de capteurs sans fil avec une architecture multi-sauts. Les auteurs ont présenté les modèles RdPSG représentant respectivement les nœuds capteurs et les communications.

Ainsi, chaque nœud est modélisé par un réseau de Petri (Figure 4.3 [92]), qui décrit les capacités du nœud à lire les données collectées de l'environnement et les traitements qu'il exécute, particulièrement le traitement relatif à l'agrégation des données. La place "Input_place" représente les données reçues des autres nœuds. Ces données sont immédiatement placées dans la place "agg_queu" où elles attendent l'opération d'agrégation. L'agrégation est représentée par la transition "agrégate_Data" qui prend deux jetons de la place "agg_queu" et introduit un jeton dans la même place. La fonction d'agrégation est modélisée par une transition

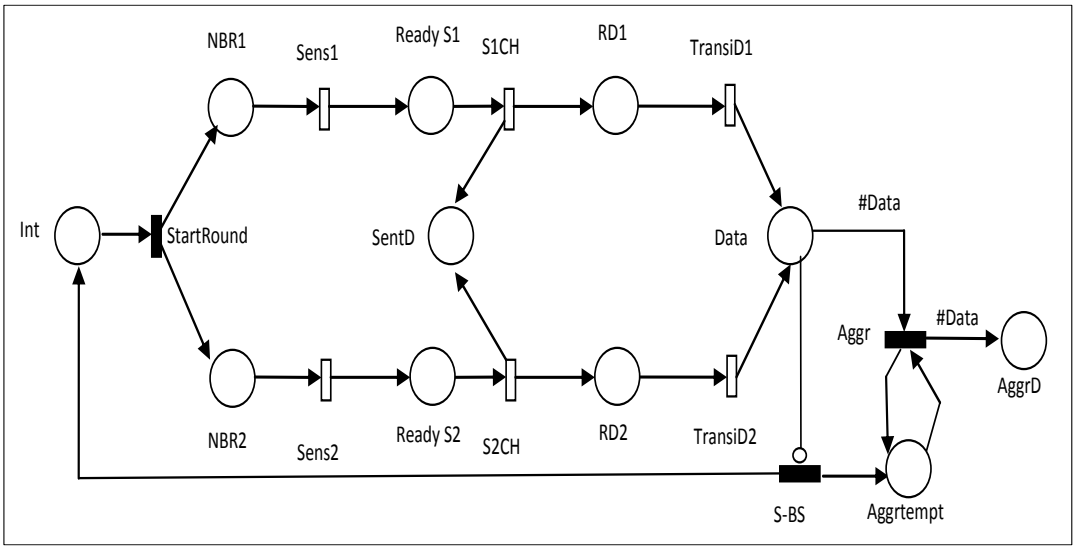


FIGURE *4.2* – Le modèle RdPSD d'un cluster.

immédiate parce que le temps de son exécution est négligeable. Après un certain temps donné, défini par la temporisation de la transition exponentielle "send_data", les données seront introduites dans la place "output_place" pour être envoyées aux nœuds cibles.

Les modèles pour les nœuds qui envoient seulement des messages et le nœud puits sont beaucoup plus simples. Pour ces nœuds, seulement les fonctions lecture et envoi des données de l'environnement sont modélisés. Les modèles contiennent une place appelée "events" (qui représente l'environnement). Le modèle complet de RCSF, comporte aussi une transition avec une temporisation donnée qui met un jeton dans chacune des places "events" de chaque modèle de nœud, de sorte que tous les nœuds collectent les données de l'environnement en même temps. Ces données vont être immédiatement introduites dans la place "agg_queu" pour être agrégées avec d'autres données et envoyées vers le nœud puits.

L'envoi des messages est également modélisé par un RdPSG représentant les canaux, la place "input_place" représente les messages à envoyer. Si le canal n'est pas occupé, les messages sont envoyés immédiatement, prenant un temps défini par la temporisation de la transition exponentielle "send_complete". Pendant l'envoi des messages, le canal est occupé et d'autres messages à envoyer sont en attente dans la place "input_place". Ce modèle ne tient pas compte des échecs possibles

dans l'envoi des messages, cependant les auteurs ont ajouté une autre transition exponentielle, représentant les temps d'attente de communication, avec en entrée la place "channel_busy" et une temporisation plus importante que celle de la transition "send_complete". La temporisation de cette transition est liée à la fiabilité du canal.

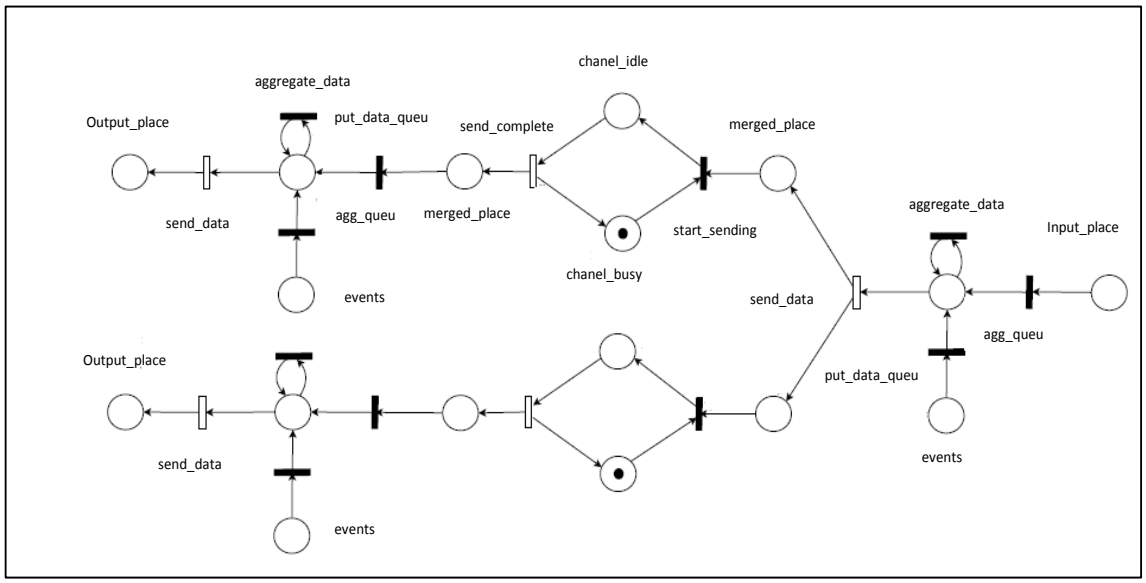


FIGURE *4.3* – Modèle de transmission des données depuis le nœud i.

Le modèle de RdPSG est composé des modèles de nœuds et des modèles de canal. Cette composition est obtenue par la fusion de la place "output_place" du nœud expéditeur avec la place "output_place" du canal et la place "input_place" du nœud récepteur, pour chaque canal entre deux nœuds. En utilisant cette méthode pour construire le modèle, la valeur prévue du nombre de jetons dans la place "channel_idle" peut être calculée pour chaque canal et la valeur prévue du nombre de jetons dans la place "agg_queue" peut être calculée aussi pour chaque nœud de capteur. Ces mesures indiquent, respectivement, l'utilisation moyenne des canaux et la probabilité d'attente des données pour l'arrivée de plus de données, afin d'exécuter l'agrégation.

Dans l'approche proposée dans [92], les auteurs ont utilisé les RdPSG pour modéliser une fonction très importante dans les réseaux de capteurs notamment l'agrégation des données. Le formalisme RdPSG fournit plusieurs méthodes d'analyse pour des mesures quantitatives. Néanmoins, des enrichissements de ces modèles peuvent être effectués pour analyser d'autres propriétés pour différentes topologies

des réseaux de capteurs.

6. **Modélisation des attaques Déni de Service dans les réseaux de capteurs sans fil**

Dans [93], les auteurs ont représenté sous un aspect formel le processus de détection des attaques de déni de service dans les réseaux de capteurs. En effet, ils ont proposé la modélisation d'un réseau de capteurs équipés des fonctionnalités de détection des attaques DoS, par respectivement, les Chaînes de Markov à Temps Continu (CMTC) et les Réseaux de Petri Stochastiques Généralisés.

(a) **Modélisation par les Chaînes de Markov à Temps Continu**

Le réseau de capteurs considéré dans cette modélisation est représenté par un seul cluster qui contient un cluster-head (CH), "n" capteurs chargés d'effectuer des mesures et "m" nœuds jouant le rôle de Cnodes. Dans le réseau, il existe exactement un nœud compromis "c" parmi les "n" capteurs. Le capteur "i" ($1 \leq i \leq n$) génère du trafic selon un processus de Poisson de taux λ_i. Le nœud compromis "c" génère du trafic selon un processus de Poisson de taux λ_c (tel que $\lambda_c \gg \lambda_i$). Chaque Cnode effectue périodiquement un contrôle du trafic de son environnement, selon une distribution exponentielle de paramètre μ, si un trafic anormal est observé, le Cnode transmet un rapport d'anomalie au CH. La topologie du cluster correspond à un graphe connexe où chaque nœud peut atteindre directement chacun des autres nœuds du cluster.

La modélisation du cluster est représentée par une CMTC multi-dimensionnelle de taille "m". Il est alors exprimé sous la forme d'un m-uplet x= (x_1, x_2,. . . , x_m) de macro-états x_k = (x_{k1}, x_{k2},. . . ,x_{kn}, x_{kd}). Chaque macro-état x_k représente le nombre de paquets détectés par le Cnode k. Plus exactement, chaque x_{ki} avec $1 \leq i \leq n$ est un compteur, initialisé à 0, qui est incrémenté à chaque fois que le Cnode k détecte un paquet en provenance du capteur "i". Lorsque le Cnode "k" détecte un trafic anormal, la variable de type booléen x_{kd} initialisée à 0 passe à 1.

Le Cnode utilise une fonction de seuil $F\{0, 1\}$ pour déterminer la nature normale ou non d'un nœud. Autrement dit, le Cnode "k", dans l'état x_k, reçoit périodiquement les messages de chaque capteur "i" appartenant au cluster. Ce sont des transmissions dites normales, qui arrivent avec une fréquence moyenne μ_i. Chaque message reçu laisse le Cnode k dans un état presque identique, à la

seule différence que le compteur x_{ki} est incrémenté de 1. La même chose se produit avec une fréquence μ_c pour les messages envoyés par le nœud compromis, qui provoque à chaque fois l'incrémentation du compteur x_{kc}.

La vérification du trafic enregistré par le Cnode k survient avec une fréquence μ. La fonction F est appliquée sur l'ensemble des x_{ki}, (tous les nœuds appartenant au cluster). Si l'un des capteurs a envoyé un nombre de paquets supérieur à la valeur du seuil tel que, $(F(x_k) \leq seuil)$, x_{kd} prend la valeur 1. Un message d'alerte est alors envoyé du Cnode k au CH. Si en revanche, aucune anomalie n'est à rapporter, la valeur x_{ki} est laissée à 0. Dans les deux cas, tous les compteurs x_{ki} sont réinitialisés à 0, afin de s'assurer que la vérification suivante soit effectuée sur des données "fraiches". Pour obtenir la probabilité de la détection des attaques, des mesures analytiques ont été effectutées à partir des états stationnaires. Par conséquent, la probabilité de détection du nœud compromis par le Cnode k, est calculé à la distribution stationnaire du macro-état x_k=(x_{k1}, x_{k2},. . .,x_{kn}, x_{kd}).

Il est à noter que dans la modélisation par les CMTC présentée dans la section précédente, la variable aléatoire qui représente la période entre deux contrôles consécutifs, suit une distribution exponentiellement. Cependant, en réalité le contrôle se produit à des intervalles fixes, ou continu sans interruption. Par conséquent la modélisation stochastique de la détection des attaques DoS exige un processus stochastique non Markovian, qui ne présente pas cette contrainte (par example les réseaux de Petri).

(b) **Modélisation par les Réseaux de Petri Stochastiques Généralisé (RdPSG)**

Dans [93], les auteurs considèrent un réseau constitué de plusieurs capteurs regroupés en cluster. Chaque nœud appartenant à un cluster peut être un nœud statique ou dynamique. Dans le premier cas (nœud statique), le nœud est doté de la fonctionnalité de détection ou de la fonctionnalité de surveillance du trafic sur une portion du réseau (nœud de contrôle ou Cnode). Dans le second cas (nœud dynamique), les nœuds peuvent basculer entre les deux fonctionnalités.

- Modèle RdPSG des nœuds de détection

Les auteurs dans [93], supposent que chaque capteur effectue les opérations de détection selon des intervalles de temps correspondant à une fonction exponentielle Exp (λ_i). Par conséquent, chaque nœud envoie les paquets de données

détectés selon une distribution exponentiellement avec un taux λ_i. La fonction de détection est modélisée par un RdPSG qui consiste en une simple transition temporisée exponentiellement, sans places d'entrée (c.-à-d. toujours sensibilisée) et avec autant d'arcs sortants conduisant au buffer d'entrée des nœuds voisins (Figure 4.4 [93]).

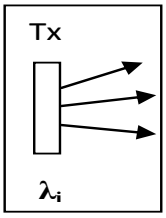


FIGURE *4.4* – Modèle RdPSG nœud normal.

- Modèle RdPSG des Cnodes

Le Cnode a pour fonction principale la surveillance du trafic d'une portion du réseau. Du point de vue de la modélisation, les auteurs dans [93] distinguent deux cas. Le premier correspond à un nœud Cnode doté uniquement de la fonction de contrôle (cas de Cnodes statiques Figure 4.5 [93]). Par contre, le second cas représente un Cnode doté de deux fonctionnalités, la détection et le contrôle (cas Cnodes dynamiques Figure 4.6 [93]).

i. **Modèle de Cnode statique**

Dans le modèle des réseaux de Petri Stochastiques Généralisés représentant les Cnodes statiques (Figure 4.5 [93]), un Cnode détecte une attaque lorsque le débit du trafic dépasse un seuil donné (c.-à-d. le nombre de paquets par période d'observation). La place $InBuffer$ représente l'entrée tampon d'un nœud, où sont placés les paquets transmis par les nœuds voisins. La place $InBuffer$ reçoit les jetons (correspondant aux paquets transmis) par les arcs d'entrée provenant du modèle des nœuds voisins (c.-à-d. les arcs d'entrée de la place $InBuffer$ sont les arcs de sortie de la transition temporisée, représentant l'activité de détection de chaque nœud voisin).

La modélisation de la fonctionnalité de surveillance du trafic est basée sur le principe de l'exclusion mutuelle ; elle est représentée par deux transitions temporisées déterministes $checkYes$ et $checkNo$. Elles correspondent à verifier périodiquement par le Cnode si la fréquence du trafic entrant a été anormale. A la fin de chaque intervalle fixe $[0, \Delta]$, la transition $checkYes$ est

activée, si aumoins S paquets ont été reçus (la place $InBuffer$ contient, au moins S jetons) ou $checkNo$ est activé, si moins de S paquets ont été reçus (la place $InBuffer$ contient moins de S jetons). Dans le premier cas ($checkYes$ activée), un jeton est ajouté dans la place det, représentant l'occurrence d'une détection d'attaques DoS. Sinon ($checkNo$ activée), aucun jeton n'est ajouté à place det.

Après le déclenchement de la transition $checkYes$ ou $checkNo$, un jeton est ajouté à la place $empty\_buffer$, cela permet à la transition immédiate $e-on$ de se déclencher itérativement jusqu'à ce que la place $empty\_buffer$ soit vide. Ainsi, la transition $e-end$ est déclanchée, ce qui représente la fin du cycle de processus de contrôle.

Il est à noter que vider la place $empty\_buffer$ est nécessaire pour mesurer correctement la fréquence du trafic à chaque intervalle $[0, \Delta]$.

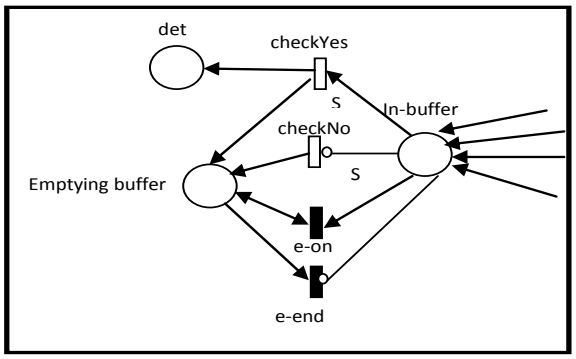


FIGURE *4.5* – Modèle RdPSG Cnode statique.

ii. Modèle de Cnode dynamique

Le RdPSG modélisant les Cnodes dynamiques (Figure 4.6 [93]), représente une extension simple du modèle des Cnodes statiques en ajoutant la place "Cnode" et la transition temporisée "TX" qui suit une distribution exponentielle. Chaque Cnode peut périodiquement passer de la fonction de détection à la fonctionnalité de contrôle. Si la place "Cnode" contient un jeton, alors la fonction "contrôle" est activée/ Dans ce cas, le RdPSG de la Figure 4.6 se comporte exactement comme celui de la Figure 4.5. Inversement, si la place Cnode est vide, alors la fonction de détection est activée

(la transition "Tx" est activée, l'arc inhibiteur entre la place "Cnode" et la transition "Tx"). Le réseau est désactivé.

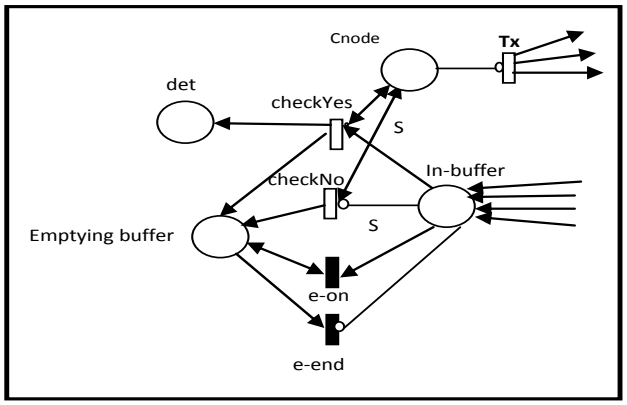


FIGURE *4.6* – Modèle RdPSG Cnode dynamique.

4.3 Description des Réseaux de Petri Colorés Stochastiques Généralisés

Avant de présenter une modélisation formelle des attaques de déni de service, par les réseaux de Petri Colorés Stochastiques Généralisés. Nous présentons une description détaillée de ces réseaux.

Les Réseaux de Petri Stochastiques Généralisés (RdPSG) [94] sont des modèles graphiques et mathématiques permettant de modéliser des systèmes séquentiels qui présentent les caractéristiques de simultané, asynchrone, de distribution non déterministe et/ou stochastique. Le comportement dynamique des RdPSG est décrit par le franchissement de certaines transitions qui peuvent être immédiates et/ou temporisées dont les temps d'exécution sont représentés par des variables aléatoires de distributions exponentielles. Le graphe d'accessibilité du modèle est isomorphe à une chaînes de Markov dans le cas où les transitions sont toutes temporisées, alors qu'il est semi-Markovien avec la présence de transitions immédiates. Une chaînes de Markov intégrée peut être extraite et une distribution à l'état stationnaire peut être calculée si le système est ergodique.

Parmi les différentes extensions du modèle des réseaux de Petri (RdP), nous citons les Réseaux de Petri Colorés (RdPC) [95]. Dans ces modèles, les couleurs sont ajoutées aux objets représentant les composants du système, ce qui permet de plier la représentation

sans perte d'information du composant qui exécute une action. L'identification de l'entité active est réalisée par l'intermédiaire de fonctions de couleurs permettant d'étiqueter les arcs du modèle.

Dans les RdPCs, les jetons sont typés par des couleurs représentées par n-uplets. Le nombre de classes (couleurs) de jetons est fini. A chaque transition sont associées différentes couleurs de franchissement et des fonctions sont associées aux arcs. La dynamique du réseau est décrite par le franchissement des transitions avec la disparition ou la création de couleurs.

Dans notre travail, nous nous sommes intéressés à une autre extension des RdP, appelée Réseaux de Petri Colorés Stochastiques Généralisés (RdPCSG) [96], [97]. Ces derniers conjuguent les caractéristiques des RdPC avec celles des RdPSG. Un RdPCSG est un graphe biparti dont les nœuds sont des places (représentées par des cercles) et les transitions qui peuvent être soit temporisées (représentées par des rectangles vides) ou immédiates (représentées par des rectangles pleins). Les transitions temporisées décrivent l'exécution des activités consommatrices de temps.

Les places avec des jetons décrivent l'état du système, tandis que les transitions représentent des évènements qui causent les changements d'état. Dans les RdPCSG, un jeton peut intégrer certaines informations, en effet, un jeton peut être considéré comme une instance d'une structure de données avec un certain nombre de champs, dont la sémantique dépend de la place à qui appartient le jeton.

La définition du type de données associées à chaque place est appelée domaine de couleur de la place. Les champs de type de données sont sélectionnés parmi un ensemble de types de base appelé classe de couleur de base. Les spécifications des classes de couleur de base font partie de la définition du réseau. Les classes de couleur peuvent être finies ou infinies. Nous ne considérons que les classes de couleurs de cardinal fini qui peut être défini par l'énumération des éléments.

Un état du système est représenté par une distribution de jetons de couleur dans les places, ce qui est appelé un marquage. La distribution initiale est notée M_0.

Les transitions dans RdPCSG peuvent être considérées comme des procédures avec des paramètres formels. Les types de paramètres formels définissent le domaine de la couleur de la transition ; leur déclaration fait partie de la description réseau et le type associé à

chaque paramètre doit être une classe de base de la couleur. Par conséquent, comme pour les places, le domaine de la couleur de transition est un produit cartésien des classes de base de couleur. Le domaine de la couleur de transition t est noté $C(t)$.

Une transition dont les paramètres formels ont été instanciés à des valeurs réelles est appelée instance de transition. Nous utilisons la notation $[t, c]$ pour un exemple de transition t, où $c \in C(t)$ représente l'affectation des valeurs réelles au paramètre de la transition.

Les arcs sont étiquetés par des fonctions de couleurs qui correspondent à un exemple de couleur d'une transition sur un ensemble de jetons de couleur. Les fonctions de couleur sont utilisées pour activer ou déactiver une transition, ce qui signifie que l'action qu'elle représente peut se dérouler dans l'état actuel du système.

- Définition formelle des réseaux de Petri Colorés Stochastiques Généralisés [97].

Un RdPCSG $= (P, T, CB, C, W^-, W^+, W^h, Pri, M_0, \theta)$ est composé de :

— P : un ensemble fini de places.
— T : un ensemble fini de transitions temporisées et immédiates, $P \cap T = \emptyset, P \cup T \neq \emptyset$.
— CB : famille de classes de couleurs de base : $CB = C_1, \ldots, C_n$ avec $C_i \cap C_j = \emptyset$.
— C est une fonction dans $P \cup T$ qui associe à chaque nœud r un domaine de couleur $C(r)$ qui représente le produit Cartésien des éléments de CB.
— W^-, W^+, W^h : $W^-(p,t), W^+(p,t), W^h(p,t) \in [C(t) \rightarrow Bag(C(p))]$ sont les fonctions couleurs qui représentent respectivement les étiquettes des inputs, outputs et les arcs inhibiteurs entre la transition t et la place p.
— Pri est la fonction de priorité définie comme suit $\forall t \in T, Pri(t) : C(t) \rightarrow N$. $Pri(t)(c)$ est la priorité de l'instance de couleur $[t, c]$.
— M_0 est le marquage initial décrivant l'état initial du système : $\forall p \in P, M_0(p) \in Bag(C(p))$.
— θ est une fonction définissant un ensemble de transitions T tel que $\theta(t) : C(t) \times (\prod_{p \in P} Bag(C(p))) \rightarrow R^+$ est la fonction de temporisation dans le modèle.

4.4 Modélisation des nœuds de capteurs et des nœuds de contrôles par les RdPCSG

Dans le modèle abordé dans [93], permettant la modélisation des attaques DoS par extensions des réseaux de Petri Stochastiques Généralisés, les auteurs ont présenté des modélisations des fonctions de détection par les modèles RdPSG pour représenter les nœuds de détection et l'élection statique et dynamique des nœuds de contrôle. Les auteurs ont opté pour ce type de RdP, afin de représenter la temporisation liée au temps d'exécution des transitions qui sont représentés par des variables aléatoires de distributions exponentielles, ce qui interprète très bien l'intervalle des délais relatifs à l'exécution de deux monitorings successifs par les nœuds de contrôle pour la détection des attaques DoS.

Toutefois, dans les modèles proposés, nous avons constaté d'une part, qu'ils permettent de détecter uniquement la présence de nœuds compromis sans pour autant les identifier et d'autre part, ils ne sont pas porteurs d'informations sur les différents nœuds du réseau de capteurs concernés tel que :

- les identifiants des nœuds compromis.
- l'identifiant du cluster-head de chaque nœud.
- l'énergie consommée par les nœuds compromis et les nœuds normaux.
- l'énergie résiduelle des nœuds compromis et normaux.

Pour remédier à ces insuffisances, nous avons enrichi les modèles présentés dans [93], en proposant des modèles basés sur les RdPCSG. Le domaine des couleurs des jetons défini dans ces réseaux nous a permis d'intégrer certaines informations. En effet, un jeton peut être considéré comme une instance d'une structure de données avec un certain nombre de champs, dont la sémantique dépend de la place à qui il appartient.

- Modélisation des nœuds capteurs

Pour ce type de nœuds, nous avons utilisé le modèle présenté dans[93] pour la modélisation des nœuds capteurs ; ce modèle est décrit par un RdPSG qui consiste en une transition source temporisée notée Tx qui suit une distribution exponentielle. Le délai de franchissement de cette transition est égal à $Ds = exp(n)$ alors que les arcs de sortie de cette transition sont dirigés vers la mémoire-tampon d'entrée des nœuds voisins. Nous avons enrichi ce modèle par l'introduction de la notion de couleur pour les jetons du modèle afin de porter toutes les données relatives aux nœuds (les identifiants des nœuds, les identifiants

du cluster-head, ...).

- Modélisation du processus de détection par les Cnodes

-) Les domaines des couleurs complexes sont :
C= <IDch_i, Ech_i, Pt_i, Pr_i> ; <IDn_j, IDch_i, E_j, Pt_j, Pr_j> ; (<IDn_j, Pt_j) tel que :

- IDch_i : identifiant du cluster-head i.
- IDn_j identifiant du nœud j.
- Ech_i : énergie initiale du nœud CH_i.
- E_j : énergie initiale du nœud j.
- Pt_i : nombre de paquets transmis par le CH_i.
- Pr_i : nombre de paquets reçus par le CH_i.
- Pt_j : nombre de paquets transmis par le nœud j.
- Pr_j : nombre de paquets reçus par le nœud j.

-) Les fonctions associées aux arcs d'entrée sont :

1. La fonction S qui permet de calculer le nombre de paquets transmis par un nœud j à ses nœuds voisins est donné par : S (<IDn_j, Pt_j) = S_j les arguments de cette fonction représentent la couleur <IDn_j, Pt_j>.
2. La fonction Id : la fonction identité telle que : Id(< IDn_j>) = < IDn_j>.

-) Les couleurs associées aux transitions "checkYes" et "checkNo" sont :

1. checkYes <IDn_j>.
2. checkNo <IDn_j, Pt_j>.

- Modèle RdPCSG des Cnodes statiques

Dans le Modèle de la figure 4.9,le franchissement des transitions temporisées Tp_j associées à la couleur <IDn_j, Pt_j> correspond à l'envoi de paquets aux nœuds voisins, produit un jeton de couleur <IDn_j, Pt_j> dans la place $In - buff$. La fonction de monitoring est décrite dans le modèle défini dans [93]. Les deux transitions temporelles *checkYes* et *checkNo* sont en exclusion mutuelle. Elles correspondent au contrôle périodique effectué par le Cnode en vérifiant si la fréquence du trafic entrant est anormale, à la

fin de chaque intervalle de temps (fixe) $[0, \Delta]$.

Si la place $Inbuffer$ contient au moins S_j (résultat de la fonction associée à l'arc de sortie) jetons de couleur <IDn_j, Pt_j>, ce qui correspond au nombre de paquets transmis par chaque nœud j, alors la transition $checkYes$ est tirée et un jeton de couleur <IDn_j> est introduit dans la place det, ce qui représente l'apparition d'une détection d'une attaque DoS, et le nœud j sera considéré comme un nœud compromis. Sinon, la transition $checkNo$, est tirée, ce qui signifie dans ce cas que le nombre de paquets reçus est inférieur à S_j paquets (la place $Inbuffer$ contient moins de S_j jetons de couleur <IDn_j, Pt_j>) et aucun jeton n'est ajouté à la place det.

Le tir de la transition $checkYes$ ou de la transition $checkNo$, permet de vider la mémoire tampon d'entrée en ajoutant un jeton dans la place $empty$. Ceci permet à la transition immédiate $e - on$ de se déclencher de manière itérative jusqu'à ce que le tampon d'entrée se vide. Lorsque la place $empty$ est vide, la transition $e - end$ est tirée, ce qui représente la fin du cycle pour vider la mémoire tampon. Cette opération est nécessaire, car elle permet de mesurer correctement la fréquence du trafic, à chaque intervalle successif $[0, \Delta]$. Nous notons qu'on ne consomme pas de temps pour vider de la mémoire tampon.

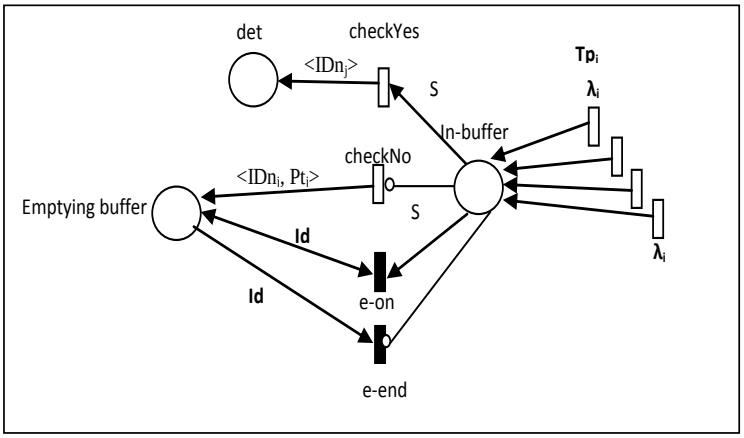


Figure 4.7 – Modèle RdPCSG d'un Cnode statique.

- Modèle RdPCSG des Cnodes dynamiques

Dans ce cas, la fonction des nœuds de contrôle est alternée entre la fonction de contrôle (monitoring) et la fonction de détection (recueillir les données du réseau). Nous avons

procédé de la même façon que pour le cas statique. Dans le modèle de la Figure 4.8, le franchissement des transitions Ts_c produit un jeton de couleur <IDn_j,> dans la place *Cnode*, ce qui déclenche le processus de contrôle et le réseau de Petri aura le même comportement que celui du modèle statique. Si la place *Cnode* est vide, alors la fonction de contrôle est désactivée et la fonction de détection est activée.

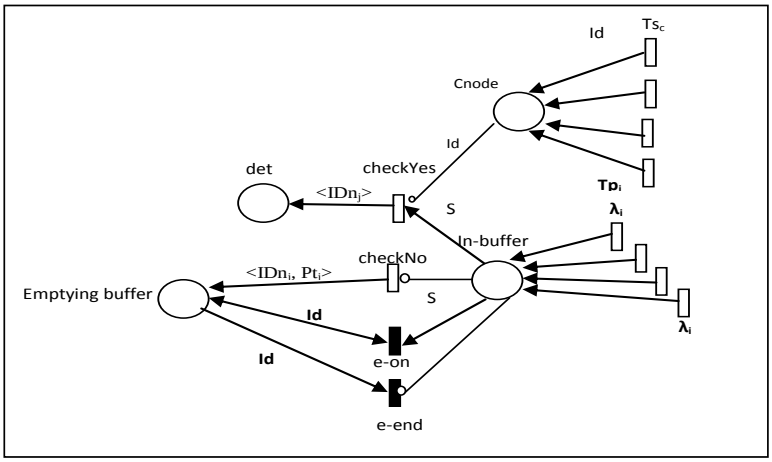


FIGURE *4.8* – Modèle RdPCSG d'un Cnode dynamique.

4.5 Modélisation d'une stratégie de défense contre les attaques actives dans les RCSF

Pour isoler et bloquer l'activité des nœuds compromis, l'utilisation des techniques de chiffrement pour sécuriser les communications entre les nœuds légitimes et les cluster-heads s'avère nécessaire. Les protocoles de groupe s'adaptent très bien à ce type de problème, car ils permettent le chiffrement des messages échangés entre tous les nœuds légitimes appartenant à un cluster. Les messages cryptés utilisent une clé de groupe qui n'est connue que par tous les nœuds légitimes et le cluster-head.

Il existe différentes approches pour générer cette clé [98]. Parmi elles :

— L'approche centralisée basée sur l'utilisation d'un serveur pour calculer et fournir la clé de groupe [99], [100].
— L'approche distribuée [101], [102], qui nécessite la contribution de chaque membre du groupe pour construire la clé de groupe.
— L'approche décentralisée [103] qui utilise des contrôleurs de sous-groupes pour générer une clé et la distribuer à chaque sous-groupe qu'il contrôle.

Dans le contexte de notre travail, nous avons opté pour une approche distribuée qui convient bien à la solution de notre problème, en particulier dans le cas des structures basées sur les clusters (groupes) qui forment le RCSF. En effet, le protocole distribué Asokan-Ginzboorg [104] décrit un échange de messages entre N nœuds et leur cluster-head (CH) pour générer une clé commune permettant de sécuriser leurs futures communications. Chaque nœud i génère une contribution (une partie de la clé de groupe) et une clé symétrique et les envoie à son CH, le tout chiffré par la clé publique de ce dernier. Une fois que le CH construit la clé de groupe en utilisant toutes les contributions reçues, il l'envoie à chaque nœud du cluster en la chiffrant en utilisant la clé correspondante au nœud en question.

4.5.1 Description du protocole Asokan-Ginzboorg

- Le CH génère une clé publique (e) et l'envoie à tous les nœuds (participants) qui lui appartiennent. Cette clé est utilisée pour chiffrer les informations échangées entre les différents nœuds.
- A la réception de cette clé, chaque nœud (participants) i, génère deux types d'informations, sa clé symétrique R_i et un nonce S_i, qui représente sa contribution pour calculer la clé de groupe.
- Chaque nœud envoie à la fois ses informations chiffrées par la clé (e) au CH.
- Après avoir reçu toutes les informations transmises par tous les nœuds du cluster, le CH génère sa propre contribution S_n et la concatène avec toutes les contributions envoyées par les différents nœuds.
- Le CH envoie à chaque nœud du cluster un message contenant sa propre contribution S_n et toutes les contributions des autres nœuds (S_i,..., S_n). Chaque message envoyé sera crypté par la clé symétrique R_i de chaque nœud.
- En recevant ce message, chaque nœud du cluster calcule la clé de groupe GK. Cette clé est égale à $f(S_i, \ldots, S_n)$.
- Pour confirmer son calcul, chaque nœud envoie un message de confirmation à son CH en chiffrant ce message avec la clé de groupe GK.

4.5.2 Complexité du protocole

Le protocole d'Asokan-Ginzboorg est mis en œuvre sur quatre phases indépendamment du nombre de participants.

-) Le leader (dans notre cas le leader représente le CH) envoie à l'ensemble des membres (nœuds) une clé de chiffrement e chiffrée par la clé publique p ; $L_n \rightarrow Id_i$, $\{e\}\, p$ et

$i \in [1, n-1]$;

-) Le CH recueille les contributions de ces différents membres. Avec $Id_i \rightarrow L_n, \{r_i, s_i\}\, e$ avec $i \in [1, n-1]$; r_i, et s_i sont respectivement la clé symétrique et le nonce cryptographique de chaque membre ;

-) Le leader envoie la clé de groupe concaténée et d'autres informations à chacun des membres. $L_n \rightarrow Id_i\ \{r_i, s_j\}\, e$ et $j \in [1, n]$ et $i \in [1, n-1]$;

-) Chaque membre envoie au leader la confirmation de la clé de groupe qu'il a correctement calculé.(S_i, $h(s_1, \ldots, s_n)$ k, i, $k = f(s_1, \ldots, s_n)$ et les fonctions h, f sont des fonctions de hachage.

La complexité de ce protocole est mesurée en termes de coût de communication. Cependant, comme nous l'avons présenté, ce protocole est principalement basé sur l'échange de messages en quatre phases entre le leader et ses membres. A chaque phase, (n-1) messages sont envoyés. Nous avons donc déduit que pour chaque round $4 * (n-1)$ messages sont échangés. La complexité globale calculée pour toute phase est de l'ordre de $n * 4 * (n-1)$.

4.5.3 Le modèle considéré

- Les domaines des couleurs complexes sont :

C= <ID_ch, Key_e, ID_n> ; <ID_n, K_n, N_n> ; <ID_n, ID_CH, K_n, N_n> ; <ID_n, ID_CH> ; <ID_n, K_G> ; <ID_CH, K_G> ; tel que :

— ID_ch : identifiant du cluster-head.
— ID_n : identifiant du nœud n.
— Key_e :clé symétrique du nœud CH.
— K_n : clé symétrique du nœud n.
— K_G : clé de groupe.

Le franchissement des transitions temporisées T_0 associées à la couleur $< ID\_CH >$ produit les n jetons de couleurs composées $< ID\_ch, Key\_e, ID\_n >$ dans la place intermédiaire S_m_1. Le franchissement de la transition T_{01} ajoute n jetons de couleurs composées à la place Add_nonce. Le franchissement de la transition T_{02} correspond à l'ajout à chaque nœud son nonce pour la construction de la clé du groupe et n jetons de couleur $< ID\_n, K\_n, N\_n >$ seront introduits dans la place intermédiaire S_m_2. La transition T_{03} sera sensibilisée, son tir produit un jeton de couleur $< ID\_n, ID\_CH, K\_n, N\_n >$ dans la place "$confirm\_cl$" et la place intermédiaire S_m_3 et un jeton de couleur $< ID\_n, ID\_CH, K\_n, N\_n >$ sera introduit dans la

place $key\_reception$, ce qui correspond au calcul de la clé de groupe par chaque nœud. Après le calcul de la clé de groupe, celle-ci sera comparée à la clé calculée par le CH, ce qui correspond au franchissement de la transition T_{05} et de la transition T_{03}. Un jeton de couleur $< ID\_n, K\_G >$ et un jeton de couleur $< ID\_n, ID\_CH, K\_n, N\_n >$; seront introduits dans la place $confirm\_Key$.

La partie inférieure du modèle modélise le processus de la fin de l'établissement de la clé du groupe, ce qui correspond au tir des transitions T_{06} et T_{07} et l'ajout d'un jeton respectivement dans les places $End\_session$ et $End\_key\_session$. Ceci permet à la transition immédiate $e-on$ de se déclencher de manière itérative jusqu'à ce que la place $End\_key\_session$ se vide. Lorsque la place $End\_session$ est vide, la transition $e-end$ est tirée, ce qui représente la fin du cycle de l'établissement de la clé de groupe.

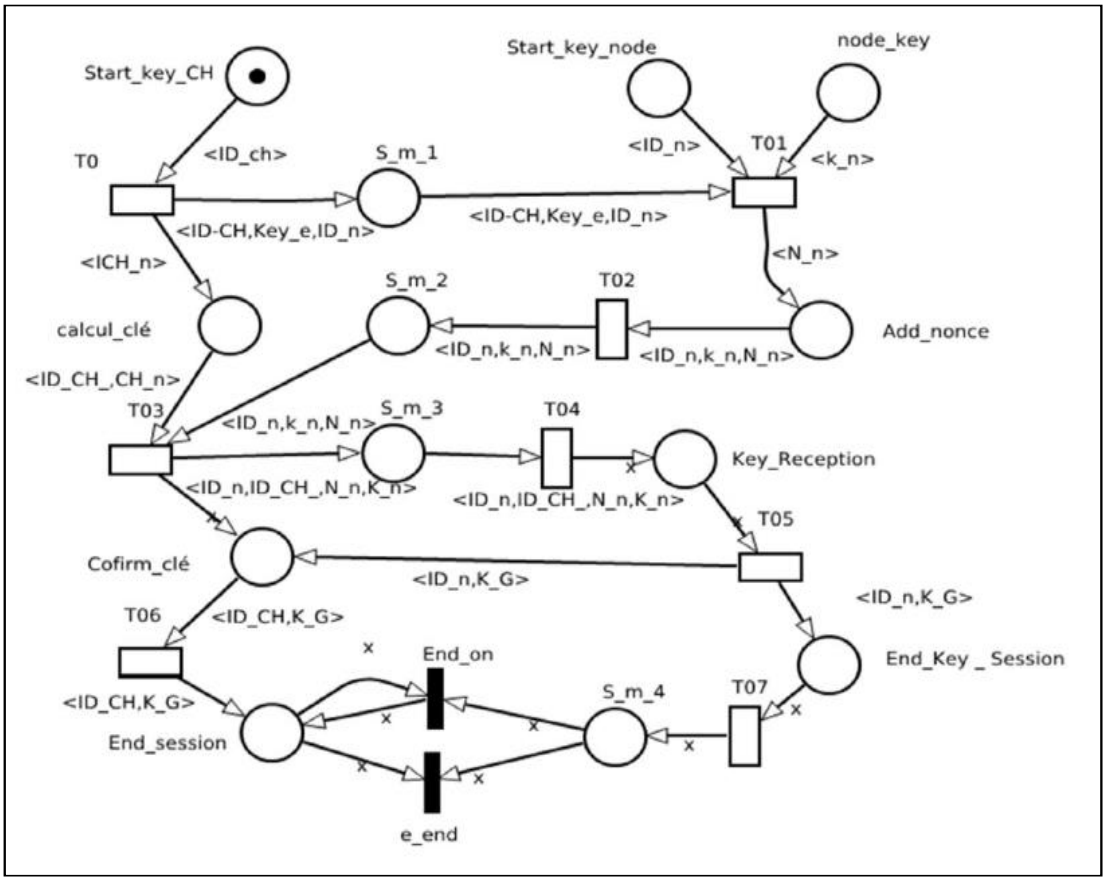


FIGURE *4.9* – Modèle réseau de Petri de la stratégie de défense.

4.6 Conclusion

Les réseaux de capteurs sans fil sont des systèmes complexes qui sont soumis à plusieurs contraintes, et imposent des exigences particulières, ce qui rend leur conception une tâche très délicate. Ainsi, avant la réalisation de ces réseaux, il est judicieux de prédire leur comportement, de s'assurer qu'ils répondent bien aux spécifications et aux exigences, pour lesquelles ils ont été conçus, particulièrement celles liées à la fiabilité et surtout d'économie d'énergie. Le recours aux méthodes de la simulation et aux techniques de modélisation de ces réseaux s'avère nécessaire. Dans ce contexte, nous avons proposé dans ce chapitre, des modèles à base de réseaux de Petri permettant d'analyser le comportement de ces réseaux, particulièrement sous des attaques de déni de service.

Conclusion générale

Dans cette thèse nous avons abordé le problème de la conception des réseaux de capteurs sous les différentes contraintes et les exigences de fiabilité liées à ces réseaux, particulièrement la contrainte liée à la limitation des ressources énergétiques. Il n'existe pas de méthodes permettant la conception, la vérification et l'analyse des performances des réseaux de capteurs, pour prédire un fonctionnement correct avec une fiabilité élevée, et assurer une solution optimale en termes de consommation énergie. Ainsi, le recours à des méthodes et des outils d'aide à la conception des réseaux de capteurs reste une alternative pour résoudre ce problème.

Généralement, nous pouvons utiliser les simulateurs et les méthodes formelles pour examiner l'exactitude et le bon fonctionnement de ces réseaux. Dans ce contexte, nous avons présenté quelques simulateurs utilisés dans les réseaux de capteurs et un état de l'art sur les différents travaux utilisant des modèles formels pour la modélisation des différents aspects liés à ces réseaux, notamment la gestion de la consommation de l'énergie. Aussi, nous avons proposé des modélisations à base de Réseaux de Petri permettant d'analyser le comportement de ses réseaux, particulièrement sous des attaques de déni de service.

En considérant la technique du clustering qui est l'une des méthodes utilisées pour économiser l'énergie dans les RCSF, nous avons aussi, présenté une description détaillée de cette technique et un panorama des différents algorithmes de clsustering utilisés dans ces réseaux. Par la suite, nous avons présenté la méthode basée sur l'application de l'algorithme de clustering général FFUCA dans le domaine des réseaux de capteurs. Après l'évaluation des performances de la méthode en considérant la métrique consommation d'énergie, nous avons constaté que FFUCA donne de résultats meilleurs que l'algorithme LEACH.

Dans le même ordre d'idée, en considérant l'aspect sécurité, nous avons montré la vulnérabilité de ses réseaux aux attaques de déni de service, ciblant les différents modèles des couches ISO, provoqués par des nœuds malveillants, par exemple. De ce fait, ces nœuds exploitent les liaisons radio sans fil pour déstabiliser le réseau, toucher l'intégrité et la

confidentialité des données. Ces attaques peuvent avoir plusieurs natures tels que, l'écoute espionne, le brouillage et le flooding. En tenant compte du défi énergétique caractérisant les réseaux de capteurs, nous avons étudié l'opportunité de l'utilisation de la technique de clustering dans les solutions proposées pour détecter et contrer les attaques DoS dans les réseaux de capteurs. Nous avons également présenté notre contribution à travers la proposition de plusieurs approches basées le clustering récursif.

La première repose sur l'application récursive de l'algorithme LEACH pour sélectionner les nœuds de contrôle permettant de surveiller le trafic dans un réseau de capteurs, voire détecter d'éventuelles attaques malveillantes. En adoptant cette méthode, nous avons évalué et analysé les performances de la méthode proposée par rapport à plusieurs critères tels que le temps de détection, le taux de détection, le taux de la consommation d'énergie.

Aussi, nous avons utilisé l'approche proposée pour prévenir et détecter les attaques de déni de service dans les RCSF. A travers, l'implémentation de la méthode proposée sur Simevents, nous avons obtenus des résultats numériques trés significatifs en termes de taux de détection et de temps de détection .

La deuxième approche proposée dans ce contexte, représente une approche basée aussi sur un clustering récursif appliqué à l'algorithme FFUCA. L'évaluation de la m éthode proposée a montré qu'en termes de détection des nœuds malveillants, les résultats sont très convaincants. Aussi, en termes de consommation d'énergie l'utilisation récursive de l'algorithme FFUCA procure une meilleure gestion de l'énergie, par conséquent, une plus longue durée de vie du réseau.

Enfin, une nouvelle approche dynamique et adaptive pour la détection du flooding dans les réseaux de capteurs a été présentée dans cette thèse. Elle consiste à élire progressivement des nœuds de contrôle (Cnodes), en fonction du taux de messages envoyés par les nœuds compromis.

Cette approche permet une détection plus rapide des nœuds suspects, en assurant une consommation d'énergie rationnelle. A travers les résultats de la simulation de la méthode proposée, nous avons montré que son utilisation est très intéressante en termes de temps de détection et de durée de vie du réseau en comparaison avec d'autres méthodes de détection.

Perspective et travaux futurs

Au terme de ces travaux, plusieurs perspectives de recherche se dégagent :

- Implantation d'un logiciel permettant la simulation et la validation des modèles proposés (RdPCSG), particulièrement la prise en charge des couleurs composées.
- Enrichissement de l'approche proposée dans [92] pour analyser d'autres propriétés pour différentes topologies des réseaux de capteurs sans fil.
- Proposition de l'amélioration des solutions proposées pour la détection des attaques DoS en considérant :
  1. Les attaques coopératives ;
  2. D'autres méthodes de clustering ;
  3. D'autres paramètres de performance (débit, latence, etc.) ;
  4. D'autres types d'attaques à savoir : l'attaque Sinkhole, le trou noir et l'attaque Wormhole.
- Analyse et évaluation des performances des réseaux de capteurs par rapport à d'autres métriques et d'autres aspects, tels que la fiabilité et la qualité de service.
- Etude et analyse de l'application des solutions proposées sur une application réelle des réseaux de capteurs (détection de la pollution par exemple).

Bibliographie

[1] W. Roush, "10 emerging technologies that will change your world," Technology Review **February**, 2 (2003).

[2] M. Bouaziz and A. Rachedi, "A survey on mobility management protocols in wireless sensor networks based on 6lowpan technology," Computer Communications **74**, 3–15 (2016).

[3] O. S. da Penha and E. F. Nakamura, "Fusing light and temperature data for fire detection," in "Computers and Communications (ISCC), 2010 IEEE Symposium on," (IEEE, 2010), pp. 107–112.

[4] L. Yu, N. Wang, and X. Meng, "Real-time forest fire detection with wireless sensor networks," in "Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on," , vol. 2 (IEEE, 2005), vol. 2, pp. 1214–1217.

[5] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in "Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications," (Acm, 2002), pp. 88–97.

[6] J. Lloret, M. Garcia, D. Bri, and S. Sendra, "A wireless sensor network deployment for rural and forest fire detection and verification," sensors **9**, 8722–8747 (2009).

[7] M. Asensio and L. Ferragut, "On a wildland fire model with radiation," International Journal for Numerical Methods in Engineering **54**, 137–157 (2002).

[8] G. Xu, W. Shen, and X. Wang, "Applications of wireless sensor networks in marine environment monitoring : A survey," Sensors **14**, 16932–16954 (2014).

[9] C. A. Pérez, M. Jimenez, F. Soto, R. Torres, J. López, and A. Iborra, "A system for monitoring marine environments based on wireless sensor networks," in "OCEANS, 2011 IEEE-Spain," (IEEE, 2011), pp. 1–6.

[10] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a wireless sensor network on an active volcano," IEEE internet computing **10**, 18–25 (2006).

[11] M. Lehsaini, "Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique," Ph.D. thesis, Université de Franche-comté. UFR des sciences et techniques (2009).

[12] R. Bader, M. Pinto, F. Spenrath, P. Wollmann, and F. Kargl, "Bignurse : A wireless ad hoc network for patient monitoring," in "Pervasive Health Conference and Workshops, 2006," (IEEE, 2006), pp. 1–4.

[13] V. Shnayder, B.-r. Chen, K. Lorincz, T. R. Fulford-Jones, and M. Welsh, "Sensor networks for medical care," (2005).

[14] J. Ko, J. H. Lim, Y. Chen, R. Musvaloiu-E, A. Terzis, G. M. Masson, T. Gao, W. Destler, L. Selavo, and R. P. Dutton, "Medisn : Medical emergency detection in sensor networks," ACM Transactions on Embedded Computing Systems (TECS) **10**, 11 (2010).

[15] K. Lorincz, B.-r. Chen, G. W. Challen, A. R. Chowdhury, S. Patel, P. Bonato, M. Welsh *et al.*, "Mercury : a wearable sensor network platform for high-fidelity motion analysis." in "SenSys," , vol. 9 (2009), vol. 9, pp. 183–196.

[16] C.-Y. Chong and S. P. Kumar, "Sensor networks : evolution, opportunities, and challenges," Proceedings of the IEEE **91**, 1247–1256 (2003).

[17] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda *et al.*, "A line in the sand : a wireless sensor network for target detection, classification, and tracking," Computer Networks **46**, 605–634 (2004).

[18] T. Gosnell, J. Hall, C. Jam, D. Knapp, Z. Koenig, S. Luke, B. Pohl, A. Schach von Wittenau, and J. Wolford, "Gamma-ray identification of nuclear weapon materials," Tech. rep., Lawrence Livermore National Lab., Livermore, CA (US) (1997).

[19] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic *et al.*, "Vigilnet : An integrated sensor network system for energy-efficient surveillance," ACM Transactions on Sensor Networks (TOSN) **2**, 1–38 (2006).

[20] M. Brown, "Users guide developed for the jbrews project," Los Alamos National Laboratory of California University (1999).

[21] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based countersniper system," in "Proceedings of the 2nd international conference on Embedded networked sensor systems," (ACM, 2004), pp. 1–12.

[22] T. KNot, "Bp frontiers magazine," Tech. Rep. 9, Lawrence Livermore National Lab., Livermore, CA (US) (2004).

[23] K. LAHMAR, "Etude et validation d'un environnement de simulation de la consommation dans les rcsf," Tech. rep., CESlab, National School of Engineers of Sfax, Tunisia (2012).

[24] Y. Miao, "Applications of sensor networks," in "Seminar on ireless Self-Organization Networks, http ://www7. informatik. uni-erlangen. de/ dressler/lectures/seminar-sensornetze-ss05/paperying-miao. pdf," (2005).

[25] K. Chintalapudi, T. Fu, J. Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, and S. Masri, "Monitoring civil structures with a wireless sensor network," IEEE Internet Computing **10**, 26–34 (2006).

[26] J. Feng, F. Koushanfar, and M. Potkonjak, "Sensor network architecture," Handbook of Sensor Networks (2004).

[27] J. Feng, F. Koushanfar, and M. Potkonjak, "System-architectures for sensor networks issues, alternatives, and directions," in "Computer Design : VLSI in Computers and Processors, 2002. Proceedings. 2002 IEEE International Conference on," (IEEE, 2002), pp. 226–231.

[28] P. S.J., *TinyOS* (2009).

[29] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks : a survey," Computer networks **38**, 393–422 (2002).

[30] A. A. A. Alkhatib and G. S. Baicher, "Wireless sensor network architecture," in "2012 International Conference on Computer Networks and Communication Systems (CNCS 2012)," (2012).

[31] K. C. Rahman, "A survey on sensor network," Journal of Computer and Information Technology **1**, 76–87 (2010).

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

[33] A. J. Swati and R. Priyanka, "Wireless sensor network (wsn) : Architectural design issues and challenges," International Journal on Computer Science and Engineering **2**, 3089–3094 (2010).

[34] K. Kaur, P. Kaur, and E. S. Singh, "Wireless sensor network : Architecture design issues and applications," International Journal of Scientific Engineering and Research (IJSER) **2** (2014).

[35] K. Maraiya, K. Kant, and N. Gupta, "Application based study on wireless sensor network," International Journal of Computer Applications **21**, 9–15 (2011).

[36] C.-Y. Chong and S. P. Kumar, "Sensor networks : evolution, opportunities, and challenges," Proceedings of the IEEE **91**, 1247–1256 (2003).

[37] K. V. Madhav, R. L. Selvaraj *et al.*, "A study of security challenges in wireless sensor networks." Journal of Theoretical & Applied Information Technology **20** (2010).

[38] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," computer **35**, 54–62 (2002).

[39] N. Farooq, I. Zahoor, S. Mandal, and T. Gulzar, "Systematic analysis of dos attacks in wireless sensor networks with wormhole injection," International Journal of Information and Computation Technology **4**, 173–182 (2014).

[40] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks : Attacks and countermeasures," Ad hoc networks **1**, 293–315 (2003).

[41] A. D. Wood and J. A. Stankovic, "A taxonomy for denial-of-service attacks in wireless sensor networks," Handbook of Sensor Networks : Compact Wireless and Wired Sensing Systems pp. 739–763 (2004).

[42] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks : analysis & defenses," in "Proceedings of the 3rd international symposium on Information processing in sensor networks," (ACM, 2004), pp. 259–268.

[43] S. Alam and D. De, "Analysis of security threats in wireless sensor network," arXiv preprint arXiv :1406.0298 (2014).

[44] K. Chelli, "Security issues in wireless sensor networks : attacks and countermeasures," in "Proceedings of the World Congress on Engineering," , vol. 1 (2015), vol. 1, pp. 1–3.

[45] D. Mansouri, L. Mokdad, J. Ben-othman, and M. Ioualalen, "Dynamic and adaptive detection method for flooding in wireless sensor networks," International Journal of Communication Systems (2017).

[46] J. Ibriq and I. Mahgoub, "Hikes : Hierarchical key establishment scheme for wireless sensor networks," International Journal of Communication Systems **27**, 1825–1856 (2014).

[47] M. Bala Krishna and M. Doja, "Deterministic k-means secure coverage clustering with periodic authentication for wireless sensor networks," International Journal of Communication Systems **30** (2017).

[48] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins : Security protocols for sensor networks," Wireless networks **8**, 521–534 (2002).

[49] D. Liu and P. Ning, "Multilevel μtesla : Broadcast authentication for distributed sensor networks," ACM Transactions on Embedded Computing Systems (TECS) **3**, 800–836 (2004).

[50] C. Karlof, N. Sastry, and D. Wagner, "Tinysec : a link layer security architecture for wireless sensor networks," in "Proceedings of the 2nd international conference on Embedded networked sensor systems," (ACM, 2004), pp. 162–175.

[51] S. Zhu, S. Setia, and S. Jajodia, "Leap+ : Efficient security mechanisms for large-scale distributed sensor networks," ACM Transactions on Sensor Networks (TOSN) **2**, 500–528 (2006).

[52] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The tesla broadcast authentication protocol," Rsa Cryptobytes **5** (2005).

[53] L. B. Oliveira, A. Ferreira, M. A. Vilaca, H. C. Wong, M. Bern, R. Dahab, and A. A. Loureiro, "Secleach, on the security of clustered sensor networks," Signal Processing **87**, 2882–2895 (2007).

[54] G. Wang and G. Cho, "Secure cluster head sensor elections using signal strength estimation and ordered transmissions," Sensors **9**, 4709–4727 (2009).

[55] M. Guechari, L. Mokdad, and S. Tan, "Dynamic solution for detecting denial of service attacks in wireless sensor networks," in "Communications (ICC), 2012 IEEE International Conference on," (IEEE, 2012), pp. 173–177.

[56] V. D. Kumar and C. Navaneethan, "Protection against denial of service (dos) attacks in wireless sensor networks," International Journal of Advanced Research in Computer Science & Technology **2**, 439–443 (2014).

[57] O. Younis and S. Fahmy, "Heed : a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," IEEE Transactions on mobile computing **3**, 366–379 (2004).

[58] G.-H. Lai, C.-M. Chen, and B. Jeng, "Dos detection in cluster-based sensor networks," in "Proceedings of the Fourth IASTED International Conference on Communication, Network and Information Security," (ACTA Press, 2007), pp. 109–115.

[59] Q. Monnet, Y. Hammal, L. Mokdad, and J. Ben-Othman, "Fair election of monitoring nodes in wsns," in "Global Communications Conference (GLOBECOM), 2015 IEEE," (IEEE, 2015), pp. 1–6.

[60] J.-M. Plateau, Brigitte ; Foutneau, "A methodology for solving markov models of parallel systems," Journal of parallel and distributed computing **12**, 370–387 (1991).

[61] S. Fouchal, D. Mansouri, L. Mokdad, J. Ben-Othman, and M. Ioualalen, "Clustering wireless sensors networks with ffuca," in "Communications (ICC), 2013 IEEE International Conference on," (IEEE, 2013), pp. 6438–6443.

[62] S. Fouchal, Q. Monnet, D. Mansouri, L. Mokdad, and M. Ioualalen, "A clustering method for wireless sensors networks," in "Computers and Communications (ISCC), 2012 IEEE Symposium on," (IEEE, 2012), pp. 000888–000892.

[63] S. Fouchal, D. Mansouri, L. Mokdad, and M. Iouallalen, "Recursive-clustering-based approach for denial of service (dos) attacks in wireless sensors networks," International Journal of Communication Systems **28**, 309–324 (2015).

[64] D. Mansouri, L. Mokdad, J. Ben-Othman, and M. Ioualalen, "Detecting dos attacks in wsn based on clustering technique," in "Wireless Communications and Networking Conference (WCNC), 2013 IEEE," (IEEE, 2013), pp. 2214–2219.

[65] D. Kumar, T. C. Aseri, and R. Patel, "Eehc : Energy efficient heterogeneous clustered scheme for wireless sensor networks," Computer Communications **32**, 662–667 (2009).

[66] S. Lindsey and C. S. Raghavendra, "Pegasis : Power-efficient gathering in sensor information systems," in "Aerospace conference proceedings, 2002. IEEE," , vol. 3 (IEEE, 2002), vol. 3, pp. 3–3.

[67] K. Beydoun, "Conception d'un protocole de routage hiérarchique pour les réseaux de capteurs et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique," Ph.D. thesis, Université de Franche-comté. UFR des sciences et techniques (2009).

[68] M. Chatterjee, S. K. Das, and D. Turgut, "Wca : A weighted clustering algorithm for mobile ad hoc networks," Cluster computing **5**, 193–204 (2002).

[69] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in "System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on," (IEEE, 2000), pp. 10–pp.

[70] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," IEEE Transactions on wireless communications **1**, 660–670 (2002).

[71] S. Fouchal and I. Lavallée, "Fast and flexible unsupervised custering algorithm based on ultrametric properties," in "Proceedings of the 7th ACM symposium on QoS and security for wireless and mobile networks," (ACM, 2011), pp. 35–42.

[72] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," Computer communications **30**, 2826–2841 (2007).

[73] S. Fouchal, M. Ahat, and I. Lavallée, "Optimal clustering method in ultrametric spaces," in "Computers and Communications (ISCC), 2011 IEEE Symposium on," (IEEE, 2011), pp. 123–128.

[74] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbor meaningful," in "International conference on database theory," (Springer, 1999), pp. 217–235.

[75] F. Brucker, "Modèles de classification en classes empiétantes," Ph.D. thesis, Paris, EHESS (2001).

[76] G. Cleuziou, "Une méthode de classification non-supervisée pour l'apprentissage de règles et la recherche d'information," Ph.D. thesis, Université d'Orléans (2004).

[77] D. Mansouri, L. Mokddad, J. Ben-Othman, and M. Ioualalen, "Preventing denial of service attacks in wireless sensor networks," in "Communications (ICC), 2015 IEEE International Conference on," (IEEE, 2015), pp. 3014–3019.

[78] N. Dagorn, "Détection et prévention d'intrusion : présentation et limites," (2006).

[79] D. Mansouri, L. Mokdad, J. Ben-othman, and M. Ioualalen, "Dynamic and adaptive detection method for flooding in wireless sensor networks," International Journal of Communication Systems (2017).

[80] T. Issariyakul and E. Hossain, *Introduction to network simulator NS2* (Springer Science & Business Media, 2011).

[81] A. Varga, "Discrete event simulation system," in "Proc. of the European Simulation Multiconference (ESM'2001)," (2001).

[82] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim : a library for parallel simulation of large-scale wireless networks," in "Parallel and Distributed Simulation, 1998. PADS 98. Proceedings. Twelfth Workshop on," (IEEE, 1998), pp. 154–161.

[83] G. Chelius, A. Fraboulet, and E. Fleury, "Wsnet : a modular event-driven wireless network simulator," (2006).

[84] A. Sobeih, W.-P. Chen, J. C. Hou, L.-C. Kung, N. Li, H. Lim, H.-Y. Tyan, and H. Zhang, "J-sim : A simulation environment for wireless sensor networks," in "Proceedings of the 38th annual Symposium on Simulation," (IEEE Computer Society, 2005), pp. 175–187.

[85] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim : Accurate and scalable simulation of entire tinyos applications," in "Proceedings of the 1st international conference on Embedded networked sensor systems," (ACM, 2003), pp. 126–137.

[86] T. Affouaaby, "Modèlisation et ètude de performances d'un rèseau de puits pour rèseaux de capteurs sans fil à grande dimension," Tech. rep., Centre de Recherche en automatique de Nancy, Universitè de Nancy (2012).

[87] R. Rasouli, M. Ahmadi *et al.*, "Energy consumption estimation in clustered wireless sensor networks using m/m/1 queuing model," International Journal of Wireless & Mobile Networks **5**, 15 (2013).

[88] S. Tschirner, L. Xuedong, and W. Yi, "Model-based validation of qos properties of biomedical sensor networks," in "Proceedings of the 8th ACM international conference on Embedded software," (ACM, 2008), pp. 69–78.

[89] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi, "Uppaal a tool suite for automatic verification of real-time systems," Hybrid Systems III pp. 232–243 (1996).

[90] R. Alur, C. Courcoubetis, and D. Dill, "Model-checking in dense real-time," Information and computation **104**, 2–34 (1993).

[91] A. Berrachedi and M. Boukala-Ioualalen, "Evaluation of the energy consumption and the packet loss in wsns using deterministic stochastic petri nets," in "Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference on," (IEEE, 2016), pp. 772–777.

[92] B. Lacerda and P. U. Lima, "Petri nets as an analysis tool for data flow in wireless sensor networks," in "1st Portuguese Conference on WSNs, Coimbra, Portugal," (2011), pp. 1–6.

[93] P. Ballarini, L. Mokdad, and Q. Monnet, "Modeling tools for detecting dos attacks in wsns," Security and Communication Networks **6**, 420–436 (2013).

[94] M. G. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, "Modelling with generalized stochastic petri nets," (1995).

[95] K. Jensen, "Coloured petri nets," in "Petri nets : central models and their properties," (Springer, 1987), pp. 248–299.

[96] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad, "Stochastic well-formed colored nets and symmetric modeling applications," IEEE Transactions on Computers **42**, 1343–1360 (1993).

[97] C. Dutheillet and S. Haddad, "Aggregation of states in colored stochastic petri nets : Application to a multiprocessor architecture," in "Petri Nets and Performance Models, 1989. PNPM89., Proceedings of the Third International Workshop on," (IEEE, 1989), pp. 40–49.

[98] N. Chridi, "Contributions à la vérification automatique de protocoles de groupes." Ph.D. thesis, Université Henri Poincaré-Nancy I (2009).

[99] A. T. Sherman and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," IEEE transactions on Software Engineering **29**, 444–458 (2003).

[100] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," IEEE/ACM transactions on networking **8**, 16–30 (2000).

[101] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," ACM Transactions on Information and System Security (TISSEC) **7**, 60–96 (2004).

[102] M. Steiner, G. Tsudik, and M. Waidner, "Cliques : A new approach to group key agreement," in "Distributed Computing Systems, 1998. Proceedings. 18th International Conference on," (IEEE, 1998), pp. 380–387.

[103] S. Mittra, "Iolus : A framework for scalable secure multicasting," in "ACM SIGCOMM Computer Communication Review," , vol. 27 (ACM, 1997), vol. 27, pp. 277–288.

[104] N. Asokan and P. Ginzboorg, "Key agreement in ad hoc networks," Computer communications **23**, 1627–1637 (2000).

Annexes

Liste de publications

— - **Articles de revue**

1. Mansouri, Djamel, Lynda Mokdad, Jalel Ben ?othman, and Malika Ioualalen. "Dynamic and adaptive detection method for flooding in wireless sensor networks." International Journal of Communication Systems (2017).

2. Fouchal, S., Mansouri, D., Mokdad, L. and Iouallalen, M., 2015. "Recursive ?clustering ?based approach for denial of service (DoS) attacks in wireless sensors networks". International Journal of Communication Systems, 28(2), pp.309-324.

— - **Communications Internationales**

1. D. Mansouri, L. Mokdad, J. Ben-Othman, M Ioualalen. "Detecting DoS attacks in WSN based on clustering technique". In Wireless Communications and Networking Conference (WCNC), pages 22142219. IEEE, 2013.

2. Mansouri, D, Mokdad, L, Ben-Othman, J., Ioualalen, M. "Preventing denial of service attacks in wireless sensor networks". In : Communications (ICC), 2015 IEEE International Conference on. IEEE, pages. 3014-3019, 2015.

3. Fouchal, S., Mansouri, D., Mokdad, L., Ben-Othman, J., Ioualalen, M. (2013). "Clustering wireless sensors networks with FFUCA". In Communications (ICC), International Conference on pages 6438-6443, 2013. IEEE.

4. Fouchal, S., Monnet, Q, Mansouri, D., Mokdad, L., Ioualalen, M. "A clustering method for wireless sensors networks". In Computers and Communications (ISCC) Symposium on pages 888-892, 2012, IEEE.