BURKINA FASO
Université Nazi BONI

Année académique : 2021-2022

Ecole Doctorale Sciences et Techniques
(ED-ST)

Thèse de doctorat unique en informatique
Mention : Informatique

THEME :

Utilisation des techniques de clustering et de SDN pour optimiser la consommation d'énergie dans l'IoT

Soutenue par :
Mahamadi BOULOU

Directeur de thèse :
Professeur Tiguiane YELEMOU

Membres du jury :

- ***Président :*** M. Théodore Marie Yves TAPSOBA, Professeur Titulaire, Université Nazi BONI/Bobo-Dioulasso ;
- ***Rapporteur :*** M. Moussa DIALLO, Professeur Titulaire, Université Cheikh Anta Diop de Dakar/Sénégal ;
- ***Rapporteur :*** M. Tounde Mesmin DANDJINOU, Maître de Conférences, Université Nazi BONI/Bobo-Dioulasso ;
- ***Rapporteur :*** M. Boureima ZERBO, Maître de Conférences, Université Thomas SANKARA/Ouagadougou ;
- ***Examinateur :*** M. Cheikh Ahmadou Bamba GUEYE, Professeur Titulaire, Université Cheikh Anta Diop de Dakar/Sénégal.

DÉDICACE

À papa et maman, je vous aime

Remerciements

Le temps passe vite, mais les souvenirs sont éternels. Durant cette thèse, j'ai eu beaucoup d'expériences mémorables, aussi bien des joies que des difficultés. J'ai appris non seulement comment résoudre un problème, mais aussi à trouver un équilibre entre une vie académique et une vie sociale. Il y a beaucoup de défis et d'obstacles, mais je suis très reconnaissant pour toutes les opportunités et les défis.

Par le biais de ces remerciements, je voudrais saluer toutes les personnes qui ont contribué au succès de mes travaux et qui m'ont aidé lors de la rédaction de cette thèse. L'achèvement de cette thèse ne serait possible sans la patience, le soutien et l'encouragement de ces derniers. Je voudrais saisir cette occasion pour exprimer ma gratitude à ceux qui m'ont aidé de mille et une façons.

Je commence ces remerciements par mon Directeur de thèse, Monsieur Tiguiane YELEMOU, Professeur titulaire à l'Université Nazi BONI de m'avoir donné l'opportunité de poursuivre mes études de troisième cycle et aussi pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je tiens aussi à adresser mes vifs remerciements à Monsieur TAPSOBA Théodore Marie Yves, Professeur titulaire à l'Université Nazi BONI qui a bien voulu m'accueillir dans son laboratoire de recherches. Aussi, je lui adresse mes sincères reconnaissances pour les activités qu'il a initiées à l'endroit des doctorants que nous sommes pour faire avancer nos travaux.

J'adresse également mes sincères remerciements à Dr Toundé Mesmin DANDJINOU pour son soutien, ses précieux conseils et encouragements. Je lui suis reconnaissant pour tout ce qu'il a eu à m'apporter afin de faciliter la réalisation de cette thèse suite à la demande de mon directeur.

J'adresse mes remerciements à tous les membres de l'équipe REC (Réseaux Émergents et Cybersécurité) avec qui j'ai été jour et nuit durant ces années de thèse.

J'adresse mes remerciements aux enseignants de l'Université Nazi BONI et aussi les enseignants de l'Université Joseph KI-ZERBO qui m'ont fourni les outils nécessaires

à la réussite de mes études universitaires.

J'adresse mes remerciements au Directeur des Services Informatique (DSI) de l'UNB, Docteur Telesphore TIENDREBEOGO, mon chef de service, pour m'avoir déchargé de certaines tâches afin que je puisse mieux me concentrer sur mes travaux. J'adresse également mes remerciements à mes collègues de travail pour leur compréhension et leur sympathie.

J'adresse mes remerciements à M. SANOU Bakary Herman Magloire qui est un collègue dans la recherche pour son soutien et ses suggestions pertinentes.

J'adresse mes remerciements aux membres de jury qui m'ont fait l'honneur de juger ce travail.

Je voudrais adresser mes remerciements et ma reconnaissance à mes chers et précieux parents qui ont été d'un grand soutien sur tous les plans. Ils m'ont déchargé de certaines tâches domestiques qui m'incombaient afin que je puisse mieux me concentrer sur mes travaux ; grandement merci.

J'adresse mes remerciements à ma tendre épouse Fatoumata pour son soutien moral et ses précieux conseils.

Enfin, je n'oublie pas mes amis et tous ceux qui m'ont aidé de loin ou de près. MERCI à toutes et à tous.

Sigles et abréviations

ADV	Data Description
API	Application Programming Interface
APTEEN	Adaptive Threshold sensitive Energy Efficient sensor Network protocol
ARM	Advanced RISC Machine
BMN-LEACH-S	Balance Member Node-LEACH-S
BVGF	Bounded Voronoi Greedy Forwarding
CE	Control Elements
CH	Cluster Head
CMFR-CMQ	Congestion Management Flow Request Control Message Quenching
CSMA	Carrier Sense Multiple Access
DD	Directed Diffusion
DEARP	Dynamic Energy Aware Routing Protocol
EBCA	Energy-Balanced Clustering Algorithm
EECS	Energy Efficient Clustering Scheme
EEHC	Energy Efficient Heterogeneous Clustered
EEUC	Energy-Efficient Unequal Clustering Mechanism
FE	Forwarding Element
GAF	Geographic Adaptive Fidelity
GEAR	Geographic and Energy-Aware Routing (GEAR)
GeRaF	Geographic Random Forwarding
GPRS	General Packet Radio Service
GPS	Global Positioning System
HEED	Hybrid Energy-Efficient Distributed clustering
IoT	Internet of Things
INPP	In-Network Packet Processing
ISP	Internet Service Provider
LAN	Local Area Network

LEACH	Low-energy adaptive clustering hierarchy
LEACH-S	Low-energy Adaptive Clustering Hierarchy for Sensor
LFB	Logical Function Block
LTE	Long Term Evolution
MAC	Media Access Control
MECN	Minimum Energy Communication Network
MRPUC	Multihop Routing Protocol with Unequal Clustering
MOS	Mantis Operating System
NC	Noeud Collecteur de données
NE	Network Element
NEC	Nippon Electronic Corporation
NFV	Network Function Virtualization
NS	Noeud Source
NR	Noeud Rélais
ODL	OpenDayLight
ONE	Open Network Environment
OSI	Open Systems Interconnection
PAN	Personal Areas Network
PEACH	Power Efficient and Adaptive Clustering Hierarchy Protocol
PC	Personal Computer
PEGASIS	Power Efficient GAthering in Sensor Information Systems
QoS	Quality of Service
QSDN-WISE	Quality of services SDN-WISE
RAM	Random Access Memory
RCSF	Réseaux de Capteurs Sans Fil
RF	Radio-Fréquence
RMI	Remote Method Invocation
ROM	Read Only Memory

RR	Rumour Routing
RSSI	Received Signal Strength Indication
SDN	Software Defined Networking
SDN-WISE	Software Defined Network Wireless Sensor
SDK	Software Development Kit
SDP	Semidefinite Programming
SDWSN	Software Defined Wireless Sensor Network
SMECN	Small Minimum-Energy Communication Network
SNR	Signal to Noise Ratio
SOAP	Simple Object Access Protocol
SPIN	Sensor Protocols for Information Negotiation
STAR	Skills Knowledge Thinking Application and Relationships
TBF	Trajectory- Based Forwarding
TD	Topology Discovery
TDMA	Time Division Multiple Access
TM	Topology Management
UMTS	Universal Mobile Telecommunication System
WAN	Wide Area Network
WiFi	Wireless Fidelity
WSN	Wireless Sensor Network

Résumé

Du fait de la miniaturisation des nœuds capteurs, de la facilité et du faible coût de déploiement, l'utilisation des réseaux de capteurs sans fil (RCSF), a connu un grand essor. Un WSN peut comporter des centaines de milliers de nœuds capteurs. Une fois déployés, les WSN sont généralement confrontés à des difficultés de reconfiguration et de gestion de l'ensemble des nœuds capteurs. Face à ces difficultés, des solutions telles que les réseaux logiciels et les méthodes de regroupement par "cluster" ont été proposées.

Dans nos travaux, nous avons abordé dans un premier temps les problèmes d'optimisation de l'énergie dans les WSN à regroupement par "cluster". Dans un second temps, nous nous sommes intéréssés aux problématiques liées à la gestion d'énergie dans les réseaux de capteurs définis par logiciel.

Pour apporter une solution à la prémière problématique, nous avons proposé une solution de gestion d'énergie basée sur LEACH que nous avons nommée BMN-LEACH-S. Cette proposition permet d'une part d'équilibrer le nombre de nœuds membre entre les clusters d'un WSN en se basant sur un système à logique floue. Elle permet d'autre part, une élection de têtes de cluster qui se fait à tour de rôle. Nous avons montré à travers une étude analytique que cette solution pourrait augmenter la durée de vie du réseau.

Face à la deuxième problématique, nous proposons deux mécanismes que nous nommons : DEARP et CMFR-CMQ. DEARP utilise un seuil de basculement pour assurer l'équilibrage de la consommation d'énergie entre les nœuds. Pour la construction de routes, il utilise les noeuds qui ont une quantité d'énergie supérieure au seuil. Lorsque l'énergie résiduelle d'un nœud appartenant au chemin utilisé se trouve en dessous du seuil, il recherche à nouveau un autre chemin dont l'énergie minimale des nœuds est supérieure au seuil. Lorqu'il n'y a plus de chemin vers la destination respectant la contrainte de seuil, alors la valeur de ce seuil est diminuée. Ce nouveau processus permet ainsi aux nœuds de se décharger au même rythme. Les résultats de simulation ont montré que DEARP réduit le taux de perte de paquets et augmente la durée de vie du réseau. Nous avons proposé une trosième approche appelée CMFR-CMQ. Cette proposition permet de réduire l'envoi des doublons de message de type "*Request*" et évite la saturation des mémoires tampons des nœuds impliqués dans l'acheminement des données. A travers une étude analytique, nous montrons que CMFR-CMQ pourrait réduire la congestion des nœuds et par conséquent réduire leur consommation d'énergie.

Mots clés : *IoT, WSN, SDN-WISE, LEACH, Consommation d'énergie, Durée de vie des WSN.*

Abstract

Due to the miniaturisation of sensor nodes and the ease and low cost of deployment, the use of Wireless Sensor Networks (WSN) has grown rapidly. A sensor network can have hundreds of thousands of sensor nodes. Once deployed, WSN are usually faced with the difficulty of reconfiguring and managing all the sensor nodes. Therefore, solutions such as soft networks and clustering methods have been proposed to improve WSN management. In this work, we first addressed the energy optimisation problems in clustered WSN. We proposed a LEACH-based energy management solution that we named BMN-LEACH-S. This proposal firstly allows balancing the number of member nodes between clusters in a fuzzy logic based WSN. It also allows a selection of cluster heads which is done in turn. We have also shown through an analytical study that this solution could increase the lifetime of the network. In a second step, we worked on the problems related to energy management in software-defined sensor networks (SDWSN). In this case, we proposed two mechanisms, namely DEARP and CMFR-CMQ. DEARP, by means of a defined switchover threshold, combines residual energy control with the dijkstra algorithm to balance the energy consumption among the WSN nodes. Simulation results showed that DEARP reduces the packet loss rate and increases the network lifetime. As for CMFR-CMQ, this proposal reduces the sending of duplicate "*Request*" type messages and also avoids the saturation of the nodes' buffers during data routing in SDN-WISE. Based on an analytical study, we showed that CMFR-CMQ could reduce the congestion of SDN-WISE nodes and consequently also reduce the energy consumption.

Keywords : *IoT, WSN, SDN-WISE, LEACH, Energy consumption, lifetime.*

Sommaire

Liste des tableaux

Table des figures

Introduction générale

Contexte et problématiques

Avec l'avènement du paradigme de l'Internet des objets, en anglais *Internet of Things* (IoT) [11] [12], le réseau de capteurs sans fil en anglais *Wireless Sensor Network* (WSN) est une catégorie de réseau IoT qui ont été inclus parmi les technologies capables en tant que systèmes flexibles et peu coûteux pour construire des infrastructures de surveillance denses et rentables. Les WSN ont des coûts de matériels et de déploiement relativement faibles, car ils n'ont pas besoin de câblage, peuvent s'autoconfigurer et peuvent fonctionner pendant une longue période.

Les WSN sont également considérés comme l'une des meilleures techniques de collecte de données environnementales pour diverses applications. Les WSN sont généralement constitués de plusieurs nœuds. Ces nœuds sont des systèmes informatiques intégrés, petits et portables, reliés à des transducteurs spécialisés et disposant d'antennes radio pour la communication des données récoltées. Ils devraient être capables de surveiller, de traiter et de transmettre de manière autonome divers paramètres à divers endroits pendant une longue période, en utilisant une énergie limitée et souvent sans aucune maintenance au cours de leur vie.

Plus de deux décennies après la vision des WSN, les experts des domaines d'applications s'attendent à ce que la technologie résolve le problème de déploiement de nouvelles applications à moindre coût et avec peu d'efforts. La programmation des nœuds nécessite souvent des ingénieurs logiciels expérimentés. Le coût élevé et la faible fiabilité de la reprogrammation des nœuds constituent toujours des obstacles importants. Les coûts de maintenance peuvent augmenter de manière significative dans le cas de déploiements à distance ou dans des endroits difficiles d'accès.

Pour venir à bout des problèmes d'extensibilité du réseau et de difficultés de reconfiguration des nœuds une fois déployés, une nouvelle technologie est apparue, le Software Defined Networking (SDN) [13]. Le principe de cette technologie est la séparation du plan de contrôle du plan de données. Le contrôle des données est donc transféré à une entité logicielle appelée contrôleur. Ainsi, avec une vue centralisée du réseau, ce dernier permet une reconfiguration automatique, flexible et dynamique. Cette technologie était essentiellement destinée aux réseaux dotés d'une

infrastructure pour faciliter l'administration des équipements du réseau, notamment simplifier leur configuration.

Afin de résoudre le problème de la complexité de reconfiguration dans les WSN, les chercheurs ont proposé d'adapter cette nouvelle technologie SDN à ces réseaux. Cela donne naissance à un autre paradigme de réseau appelé réseau de capteurs défini par logiciel, en anglais Sofware Defined Wireless Sensor Network (SDWSN) [14]. Le SDWSN facilite la gestion de configuration et de maintenance des nœuds capteurs. Il facilite aussi le déploiement de nouvelles applications dans le réseau de capteurs à travers son contrôleur qui contient toutes les applications définissant le comportement des nœuds du réseau. Cependant, la gestion centralisée nécessite la production de nombreux messages de contrôle pour maintenir une vue globale du réseau afin de prendre des décisions efficaces concernant le comportement des nœuds. La production de messages de contrôle entraîne une consommation supplémentaire d'énergie, ce qui réduit la durée de vie du SDWSN.

Objectifs

Pour résoudre cette problématique de consommation d'énergie, plusieurs chercheurs ont proposé différentes techniques dont, le clustering, l'équilibrage et l'optimisation de charges de routage. Durant cette thèse, nous nous sommes fixé pour objectif général d'optimiser la consommation d'énergie dans les réseaux de capteurs à regroupement par cluster et aussi dans les réseaux de capteurs sans fil définis par logiciel. Pour atteindre cet objectif général, il s'agira pour nous de façon spécifique de :

- réduire la production des messages de contrôle ;
- répartir équitablement la charges de routage entre les noeuds ;
- éviter la congestion des nœuds lors des processus de routage.

Contributions

Comme contribution, nous avons dans un premier temps travaillé à améliorer le protocole de clustering LEACH-S [15] qui est une amélioration du protocole LEACH [16]. Le mécanisme que nous avons proposé à ce niveau est BMN-LEACH-S (Balance Member's node in LEACH-S). BMN-LEACH-S offre deux mécanismes. Le premier est le processus d'adhésion d'un nœud à un cluster afin d'équilibrer le nombre de nœuds entre les clusters. Cela, grâce à un système à logique floue prenant en paramètres l'énergie résiduelle du nœud et la puissance du signal. Le deuxième mécanisme permet une élection de tête de cluster à tour de rôle afin d'équilibrer la charge de gestion entre tous les nœuds du réseau. A travers une étude expérimentale, nous montrons que BMN-LEACH-S pourrait améliorer la durée de vie du réseau.

Dans un deuxième temps, nous avons travaillé à améliorer le processus de routage de SDN-WISE [10]. Ce dernier se base sur l'algorithme de Djisktra pour le choix de route, c'est-à-dire le plus court chemin. Le problème avec l'algorithme de Dijkstra est que les nœuds se trouvant sur les plus courts chemins pour aller vers le puits ont tendance à s'épuiser plus rapidement que les nœuds qui se trouvent dans les périphériques du réseau. Pour résoudre cette problématique, nous proposons un mécanisme de routage qui permet un équilibrage de la consommation d'énergie entre les nœuds du réseau. Pour ce faire, nous utilisons une variable seuil pour contrôler l'énergie des chemins les plus fréquentés. Lorsque l'énergie d'un chemin se trouve en dessous du seuil, le processus de sélection d'un nouveau chemin dont l'énergie résiduelle est supérieure au seuil est relancé. Ainsi, tous les nœuds du réseau participent à l'envoi des données. Des résultats de simulation montrent que notre solution appelée DEARP [17] permet une meilleure répartition de la charge du routage dans le réseau et augmente la durée de vie du réseau comparativement à l'algorithme de Djisktra de SDN-WISE.

Dans un troisième temps, nous avons travaillé sur les problématiques de charge de routage dues à la recherche de chemin vers le puits et de congestion dans l'architecture SDN-WISE. En effet, lorsqu'un nœud reçoit plusieurs paquets pour la même destination alors qu'il n'a pas de règle de flux correspondant aux paquets dans sa table de flux, il envoie un message de type "*Request*" pour chacun de ces paquets en transit au contrôleur et enregistre ces paquets dans sa mémoire tampon. Une fois que la mémoire tampon est saturée alors que le contrôleur n'a toujours pas répondu, il y a une perte de paquets au niveau du nœud. La création de doublons de messages de type "*Request*" entraîne une consommation d'énergie supplémentaire. Aussi, la saturation du nœud engendrerait des pertes de paquets, entraînant ainsi une consommation d'énergie supplémentaire du fait des nécessaires retransmissions. Pour faire face à ces problématiques, nous proposons une solution CMFR-CMQ [18] qui élimine l'envoi de doublons de messages de types "*Request*" et évite la congestion des nœuds du réseau. A travers une étude expérimentale, nous avons montré que CMFR-CMQ économiserait de l'énergie par rapport à SDN-WISE standard.

Plan de la thèse

Cette thèse est structurée en trois (03) chapitres. Dans le premier chapitre, nous présentons un état de l'art sur les réseaux de capteurs à cluster et ceux définis par logiciel. Le concept SDN et son application dans les WSN y sont introduits. Dans le deuxième chapitre, nous exposons notre première contribution BMN-LEACH-S pour résoudre problème de construction équilibrée des clusters et aussi pour résoudre le problème d'élection de têtes de cluster adéquat. Dans le troisième chapitre, nous présentons notre deuxième contribution appelée DEARP pour résoudre le problème

de la mauvaise répartition des charges lors du routage basé sur l'algorithme de dijkstra dans l'architecture SDN-WISE. Nous y exposons aussi une troisième contribution CMFR-CMQ qui résout le problème d'envoi des messages de types "*Request*" en doublon et aussi la congestion des nœuds due à la saturation de la mémoire tampon. Nous terminons enfin par une conclusion générale où nous présentons la synthèse des travaux et aussi nous émettons des perspectives de recherche pour des travaux futurs.

Chapitre 1

Etat de l'art

1.1 Introduction

Ce chapitre est consacré à une revue de la littéraire des techniques de gestion d'énergie dans les réseaux de capteurs. Afin de mieux cerner la problématique de consommation d'énergie dans les réseaux de capteurs sans fil (RCSF) en anglais *Wireless sensor networks* (WSN), nous présentons dans un premier temps le concept du WSN à cluster. Ensuite, nous faisons un état de l'art sur les différentes techniques proposées dans la littérature pour optimiser la consommation d'énergie. Aussi, nous établissons un état des lieux sur les WSN définis par logiciels et les solutions permettant l'optimisation de la consommation d'énergie. En dernier lieu, nous relevons quelques limites des propositions existantes et nous en dégageons des perspectives.

1.2 Réseaux de capteurs sans fil

Dans cette section, nous décrivons le principe de fonctionnement des réseaux de capteurs, ensuite nous examinons les facteurs de conception des nœuds et la pile protocolaire dans ce type de réseau. Nous exposons aussi quelques protocoles de routage dans les WSN et enfin nous abordons les domaines d'applications.

1.2.1 Définitions des concepts de base

Dans cette section, nous explorons les concepts fondamentaux des réseaux de capteurs sans fil. Nous commençons par définir les termes clés utilisés dans ce domaine, puis nous abordons les composants d'un nœud capteur, le principe de fonctionnement des WSN, et les facteurs à prendre en compte lors de la conception d'un tel réseau. Nous examinons également les différents domaines d'application des WSN, qui vont des applications militaires à la surveillance de l'environnement en passant par l'industrie. Enfin, nous étudions les systèmes d'exploitation pour les nœuds capteurs, qui sont des éléments cruciaux pour le fonctionnement efficace et stable d'un réseau de capteurs sans fil.

Un capteur peut être défini comme étant un dispositif [19] transformant

l'état d'une grandeur physique observée (telle que la température, l'humidité, la lumière, le son, la pression, la vitesse, l'accélération, le champ acoustique et magnétique, etc.), en une grandeur utilisable, généralement électrique, qui sera à son tour traduite en une donnée binaire utilisable par un système informatique. Autrement dit, les signaux analogiques produits par un capteur sur la base du phénomène observé sont convertis en signaux numériques par le convertisseur analogique numérique. Nous pouvons citer quelques types de capteurs. Par exemple un capteur qui permet de détecter le niveau d'un liquide ; ce type de capteur peut être utilisé dans un réservoir d'essence pour connaître son niveau ou observer la variation du volume d'eau dans un lac selon les saisons. Il y a aussi des capteurs qui permettent de détecter le taux d'humidité. Ils peuvent être utilisés pour contrôler les aliments dans une pièce susceptible de moisir à cause de l'humidité.

FIGURE 1.1 – capteurs

Un nœud capteur est un dispositif [19] de taille très réduite avec des ressources extrêmement limitées, autonome, capable de traiter des informations et de les transmettre, via les ondes radio, à une autre entité sur une distance limitée à quelques mètres. C'est un équipement composé de plusieurs éléments ou modules correspondant chacun à une tâche particulière ; acquisition, traitement ou transmission de données. Aussi, il dispose d'un module de gestion de l'énergie.

1.2.2 Composants d'un nœud capteur

Un nœud capteur est fondamentalement composé de quatre modules de base : le module de captage, le module de traitement, le module de transmission et le module de contrôle d'énergie. Des modules supplémentaires peuvent être greffés selon le domaine d'application, tels qu'un système de géolocalisation (GPS) ou bien un système générateur d'énergie (cellule solaire) et souvent d'un système mobilisateur chargé de les déplacer en cas de nécessité. La structure d'un nœud capteur basique est présentée dans la Figure 1.2.

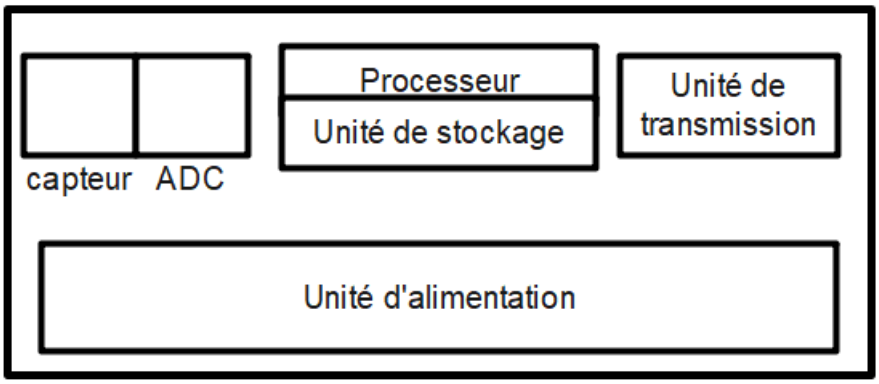

FIGURE 1.2 – Structure d'un nœud capteur

Le module de captage est chargé de mesurer divers paramètres environnementaux (température, pression, humidité) [19]. L'information est ensuite transmise sous forme numérique au module de traitement à l'aide d'un convertisseur analogique numérique. Le capteur est généralement composé de deux sous-unités : le récepteur et le transducteur (convertissant le signal du récepteur en signal électrique). Le capteur est responsable de fournir des signaux analogiques basés sur le phénomène observé au convertisseur analogique numérique. Ce dernier transforme ces signaux en des signaux numérique compréhensible par l'unité de traitement.

Le module de traitement ou l'unité de traitement [19] est la carte physique regroupant le processeur, la mémoire RAM et flash. Le système d'exploitation (Contiki par exemple) est hébergé sur ce module qui est à la base du calcul binaire et du stockage temporaire pour les données. Ce module est chargé d'exécuter les protocoles de communications qui permettent de faire collaborer le nœud avec les autres nœuds du réseau. Il peut aussi analyser les données captées pour alléger la tâche du nœud puits. Sur certains modèles, le module de traitement peut effectuer certains calculs sur les données pour optimiser les échanges.

L'unité de transmission [19] est responsable de toutes les émissions et réceptions de données via un support de communication radio. Les nœuds capteurs sont donc généralement équipés d'une radio ainsi que d'une antenne. Ce module est responsable

d'effectuer toutes les émissions et réceptions des données sur un médium sans fil. Elle peut être de type optique ou de type radiofréquence. Les communications de type optique sont robustes vis-à-vis des interférences électriques. Néanmoins, elles présentent l'inconvénient d'exiger une ligne de vue permanente entre les entités communicantes. Par conséquent, elles ne peuvent pas établir de liaisons à travers des obstacles. Les communications de type radio sont sensibles aux interférences et coûtent cher en énergie due de l'utilisation de divers circuit électronique. Par contre, elles sont moins sensibles aux obstacles.

L'unité d'alimentation [19] d'un nœud capteur fournit de l'énergie pour alimenter le capteur pour que ce dernier puisse traiter et transmettre les données capturées à un nœud central ou à un réseau de capteurs. Les capteurs peuvent être alimentés par différentes sources d'énergie, telles que des batteries, des panneaux solaires ou des sources d'énergie cinétique. La durée de vie de la batterie ou la quantité d'énergie disponible dépend de nombreux facteurs, tels que la fréquence de mesure, la quantité de données transmises et la qualité de la transmission sans fil. Dans les réseaux de capteurs sans fil, l'unité d'alimentation peut également inclure des mécanismes d'économie d'énergie pour prolonger la durée de vie de la batterie, tels que la mise en veille des capteurs lorsqu'ils ne sont pas utilisés ou la réduction de la puissance de transmission des données.

1.2.3 Principe de fonctionnement

Dans les réseaux de capteurs, la communication est faite de la façon suivante ; des nœuds équipés de capteurs récoltent des informations sur l'état du monde réel tels que la température de l'environnement, l'humidité du sol, la pluviométrie, etc. Ensuite, ils estiment ces états sous forme de données numériques. Ces données sont transportées par multi saut à un nœud puits (sink) considéré comme point de collecte. Ce nœud puits est généralement connecté à l'utilisateur du réseau par le biais d'un réseau WAN en occurrence Internet. L'usager peut adresser des requêtes aux autres nœuds du réseau, précisant le type de données requises et récolter les données environnementales captées par le biais du nœud puits. Il peut aussi agir sur l'environnement grâce à des nœuds appelés effecteurs. Dans cette architecture, il y a différents types de nœuds qui jouent différents rôles. Nous classons les nœuds selon leurs fonctions dans les WSN.

- *nœud source ou nœud capteur (NS)* : dont le rôle principal est de détecter les phénomènes physiques ou physiologiques se produisant dans son environnement immédiat afin de les transmettre, directement ou par multiples sauts, à un utilisateur final.
- *nœud relais (NR)* : il a pour rôle d'agréger et de retransmettre les mesures provenant des NS afin que celles-ci parviennent à un utilisateur final. Dans une architecture à plat, quelques travaux considèrent généralement un NS

comme un NR. Dans une architecture à 2 niveaux, un nœud passerelle joue le rôle de NR pour un ou plusieurs nœuds sources. Dans une telle configuration réseau, la capacité de transmission du NR est supposée généralement plus grande que celle du NS.

- *nœud collecteur (NC) de données* : il a pour rôle de collecter les mesures provenant des nœuds sources et éventuellement de les agréger. Généralement, un "Cluster-Head " ou chef de cluster est utilisé comme NC dans une architecture hiérarchique où les NS sont partitionnés en plusieurs groupes.

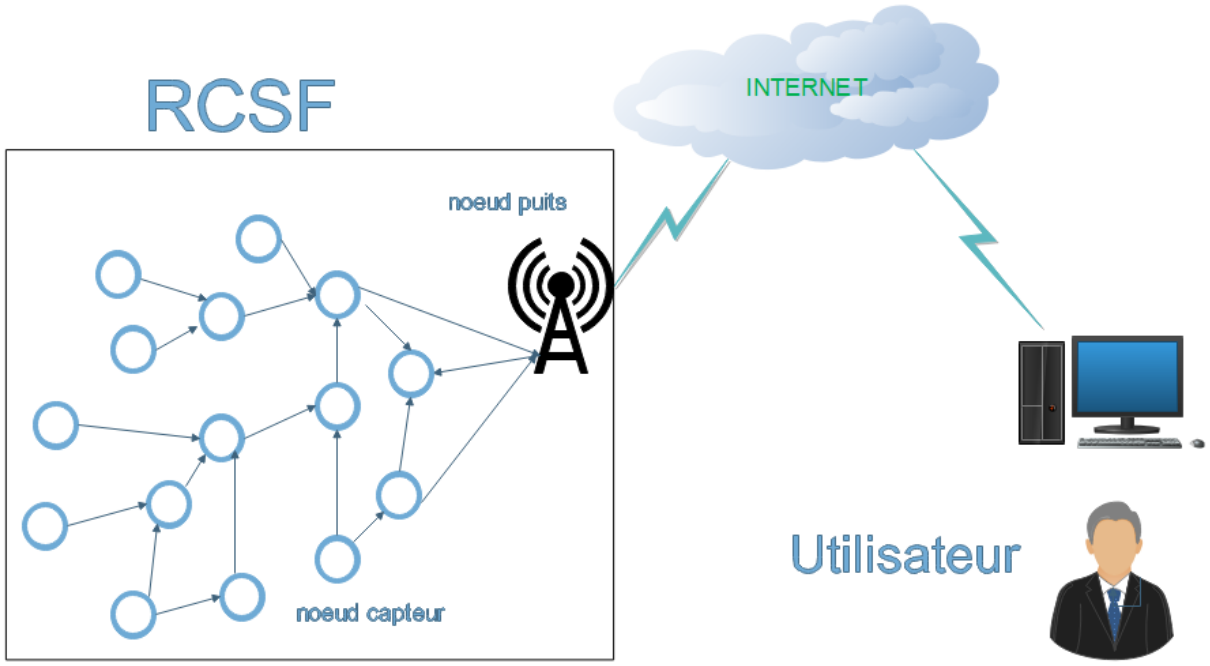

FIGURE 1.3 – Réseau de capteurs

1.2.4 Facteurs de conception d'un réseau de capteurs

Les facteurs de conception sont importants, car ils servent de ligne directrice pour la conception d'un protocole ou d'un algorithme pour les réseaux de capteurs. La conception et la réalisation des réseaux de capteurs sans fil sont influencées par plusieurs paramètres telles que l'environnement, la topologie du réseau, la tolérance aux pannes, l'évolutivité et la durée de vie du réseau etc [20].

1.2.4.1 Environnement

Les nœuds capteurs sont déployés de manière dense, soit très près, soit directement à l'intérieur du phénomène à observer. Par conséquent, ils fonctionnent généralement sans surveillance dans des zones géographiques éloignées. Ils peuvent

travailler à l'intérieur de grandes machines au fond d'un océan, dans un champ contaminé biologiquement ou chimiquement, sur un champ de bataille au-delà des lignes ennemies et dans une maison ou un grand bâtiment.

1.2.4.2 Topologie du réseau de capteurs

En général, il est possible de déployer des centaines de nœuds capteurs dans l'environnement d'étude pour la surveillance. Ils sont installés à quelques dizaines de mètres, les uns des autres. Le déploiement d'un nombre élevé de nœuds en densité exige une gestion minutieuse de l'entretien de la topologie. Nous examinons les questions liées à la topologie, l'entretien et le changement en trois phases [20] :

- Prédéploiement et phase de déploiement : les nœuds peuvent être jetés en masse ou placés un à un dans le champ. Ils peuvent être déployés en tombant d'un avion ou placés par un missile humain ou un robot.
- Phase de post-déploiement : après le déploiement, les changements de topologie sont dus à un changement de la position d'un ou des nœuds capteurs et aussi de leur accessibilité en raison du brouillage, du bruit, des obstacles en mouvement, de l'énergie, des dysfonctionnements et des détails de tâches.
- Phase de redéploiement de nœuds supplémentaires : des nœuds capteurs supplémentaires peuvent être redéployés à tout moment pour remplacer les nœuds défaillants ou en raison de changements dans la dynamique des tâches.

1.2.4.3 Tolérance aux pannes

La tolérance aux pannes [20] est la capacité à maintenir les fonctionnalités du WSN sans aucune interruption due à des défaillances de nœuds capteurs. Certains nœuds capteurs peuvent tomber en panne ou être bloqués en raison d'un manque d'énergie ou subir des dommages physiques ou des interférences environnementales. La défaillance des nœuds capteurs ne doit pas affecter la tâche globale du réseau de capteurs. C'est la question de la fiabilité ou de la tolérance aux pannes.

1.2.4.4 Évolutivité

Le nombre de nœuds capteurs déployés pour étudier un phénomène peut être de l'ordre de centaines ou de milliers [20]. Les nouveaux systèmes doivent pouvoir fonctionner avec ce nombre de nœuds. Ils doivent également utiliser la haute densité des réseaux de capteurs. La densité peut aller de quelques nœuds capteurs à des centaines de nœuds capteurs dans une région de moins de 10 m de diamètre [21].

1.2.4.5 Durée de vie

Le nœud capteur sans fil étant un dispositif microélectronique ne peut être équipé que d'une source d'énergie limitée. Dans certains scénarios d'application,

le réapprovisionnement en énergie peut être impossible. La durée de vie du nœud capteur montre donc une forte dépendance à la durée de vie de la batterie [20]. Dans un réseau de capteurs ad hoc, chaque nœud joue le double rôle d'initiateur de données et de routeur de données.

Le dysfonctionnement de quelques nœuds peut entraîner des changements topologiques importants et peut nécessiter le réacheminement de paquets et la réorganisation du réseau. C'est pourquoi la conservation et la gestion de l'énergie revêtent d'une importance supplémentaire. C'est pour ces raisons que les chercheurs se concentrent actuellement sur la conception de protocoles et d'algorithmes tenant compte de la consommation d'énergie pour les réseaux de capteurs. La tâche principale d'un nœud capteur dans un champ est de détecter des événements, d'effectuer un traitement local rapide des données, puis de transmettre les données. La consommation d'énergie peut donc être divisée en trois domaines : la détection, le traitement et la communication des données.

1.2.5 Domaines d'application des WSN

Les réseaux de capteurs ont connu un très grand succès, car ils détiennent un potentiel qui révolutionne de nombreux secteurs de notre vie quotidienne et notre économie. L'utilisation d'un réseau de capteurs dans un processus industriel ou domotique facilite son contrôle et le suivi de ses paramètres. Les applications tactiques, notamment les opérations de secours, militaires ou d'exploration, trouvent au WSN le réseau idéal. La technologie WSN intéresse également la recherche. Ainsi, de nombreuses applications civiles de ces types de réseaux ont vu le jour. On distingue la surveillance et la préservation de l'environnement, la fabrication industrielle, l'automatisation dans les secteurs du transport et de la santé, de l'agriculture, de la télématique et de la logistique.

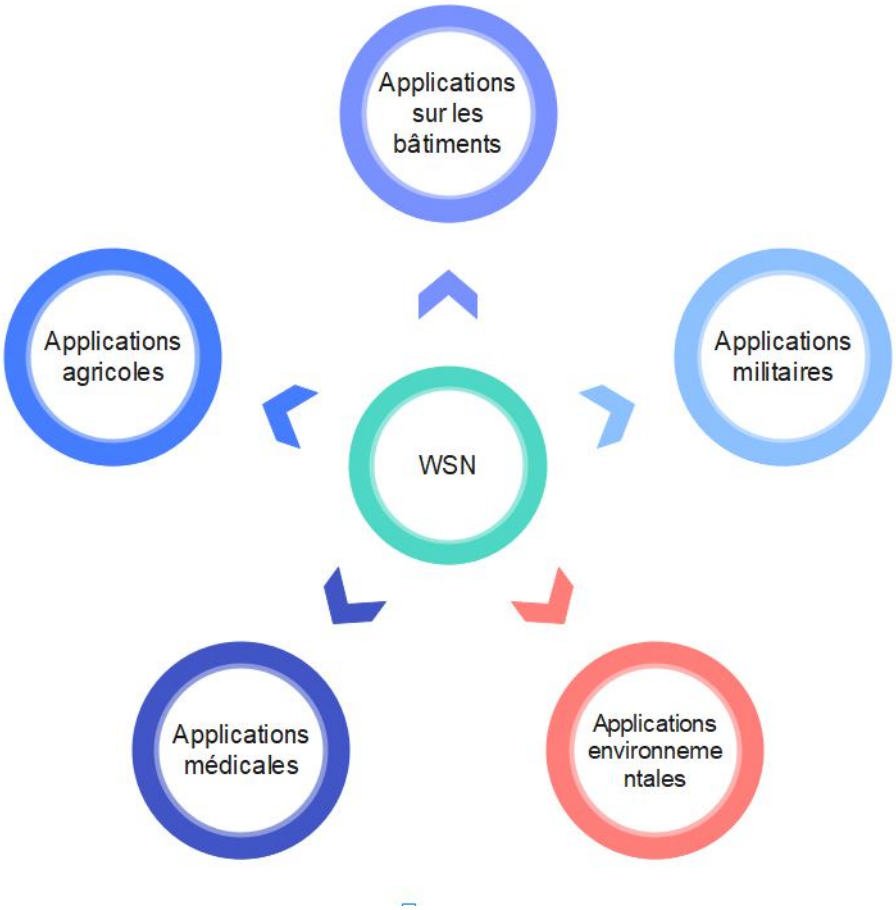

FIGURE 1.4 – Domaines d'application des WSN

1.2.6 Systèmes d'exploitation pour nœud capteur

Plusieurs systèmes d'exploitation ont été développés pour faire fonctionner les nœuds capteurs. Ces systèmes ont été développés dans différents langages et sont très légers pour être supportés par ces nœuds à faible capacité. Dans les sections suivantes, nous présentons quelques-uns de ces systèmes d'exploitation.

1.2.6.1 TinyOS

TinyOS est un système d'exploitation de petite taille environ de 400 octets [22]. TinyOS est flexible et construit à partir d'un ensemble de composants réutilisables qui sont assemblés en un système spécifique à une application. Il supporte un modèle de concurrence événementiel basé sur des interfaces à phases séparées, des événements asynchrones et des calculs différés appelés tâches.

TinyOS est implémenté dans le langage NesC, qui supporte le composant TinyOS et le modèle de concurrence ainsi que des optimisations étendues entre composants et la détection des courses au moment de la compilation.

TinyOS a permis à la fois des innovations dans les systèmes de réseaux de cap-

teurs et une grande variété d'applications. Un programme TinyOS est un graphe de composants. Chacun de ces composants étant une entité informatique indépendante qui expose une ou plusieurs interfaces. Les composants ont trois abstractions computationnelles : les commandes, les événements et les tâches. Les commandes et les événements sont des mécanismes de communication entre les composants, tandis que les tâches sont utilisées pour exprimer la communication intra-composant.

TinyOS peut être compilé pour fonctionner sur n'importe laquelle de ces plateformes sans modification. Des travaux sont en cours (par d'autres) pour porter TinyOS sur les processeurs ARM, Intel 8051 et Hitachi et pour supporter les radios Bluetooth.

1.2.6.2 Contiki

Contiki est un système configurable modulaire pour les WSN. C'est un système d'exploitation conçu [23] pour prendre le moins de place possible et le code est écrit en C. Le système complet est supposé pouvoir tourner sans problème avec 2 ko de RAM et 40 ko de ROM. Contiki possède une gestion de la programmation parallèle sous forme de «*proto-threads*» qui sont des processus légers développés spécialement pour l'occasion. Un système Contiki est constitué en deux parties : le noyau et les programmes chargés. Le partitionnement est effectué au moment de la compilation et est spécifique au déploiement dans lequel Contiki est utilisé.

1.2.6.3 MANTIS

Le système multimodal MANTIS pour les réseaux de capteurs sans fil fournit un nouveau système d'exploitation intégré multithread et multi-plateforme pour les réseaux de capteurs sans fil [24]. A mesure que les réseaux de capteurs s'adaptent à des tâches de plus en plus complexes, telles que la compression/agrégation et le traitement des signaux, le multithreading préemptif du système d'exploitation pour capteurs MANTIS (MOS) permet aux nœuds de microcapteurs d'entrelacer nativement des tâches complexes avec des tâches sensibles au temps, atténuant ainsi le problème du producteur-consommateur de tampon limité. L'une des principales caractéristiques de la conception MOS est la flexibilité, sous la forme d'un support multi-plateforme et de tests sur des PC et différentes plateformes de microcapteurs.

1.2.6.4 LiteOS

LiteOS fournit un environnement comparable à UNIX, adapté aux nœuds capteurs en réseau. LiteOS possède un système de gestion des fichiers et des commandes en mode terminal distantes semblable aux commandes Unix pour gérer les nœuds capteurs. Le noyau supporte le chargement dynamique et l'exécution multitâche d'applications. LiteOS possède une faible empreinte mémoire et peut fonctionner avec 128Ko de mémoire flash et 4ko de RAM.

L'architecture globale du système d'exploitation LiteOS est divisée en trois sous-systèmes [25] : LiteShell, LiteFS et le noyau. Mis en œuvre du côté du PC de la station de base, le sous-système LiteShell interagit avec les nœuds capteurs uniquement lorsqu'un utilisateur est présent. LiteOS fournit un mécanisme de montage de nœuds sans fil à travers un système de fichiers appelé LiteFS.

Tout comme la connexion d'un lecteur USB, un nœud LiteOS se monte sans fil sur le système de fichiers racines d'une station de base proche. De plus, de manière analogue à la connexion d'un dispositif, le montage sans fil supporte actuellement les dispositifs à porter sans fil. Le mécanisme de montage est pratique, par exemple, en laboratoire, lorsqu'un développeur souhaite interagir temporairement avec un ensemble de nœuds sur une table avant le déploiement. Bien que cela ne fasse pas partie de la version actuelle, il n'est pas difficile, d'un point de vue conceptuel, d'étendre ce mécanisme à un "service de montage à distance" pour permettre un montage réseau. Idéalement, un montage en réseau permettrait de monter un dispositif tant qu'il existe un chemin de réseau, soit via Internet, soit via une communication sans fil à bonds multiples à travers le réseau de capteurs.

1.3 WSN à cluster

Dans cette section, nous présentons tout d'abord le concept de clustering. Ensuite, nous citons les défis rencontrés dans les algorithmes de clustering. Enfin, nous détaillons quelques mécanismes de gestion d'énergie dans le WSN à cluster.

1.3.1 Concept de clustering

Le *clustering* ou la mise en cluster du réseau est une méthode utilisée pour surmonter les problèmes de consommation d'énergie. Dans les réseaux en cluster, certains capteurs sont élus en tant que têtes de cluster (CH) pour chaque cluster créé. Les nœuds capteurs dans chaque cluster transmettent leurs données au CH respectif, qui les regroupe et les transmet à une station de base [26]. La mise en cluster facilite l'utilisation efficace de l'énergie limitée des nœuds capteurs et prolonge ainsi la durée de vie du réseau.

Bien que les nœuds capteurs dans les clusters transmettent des messages sur une courte distance au sein des clusters, les CH perdent plus d'énergie en raison de la transmission de messages sur de longues distances (des CH à la BS) par rapport aux autres nœuds capteurs dans le cluster. La réélection périodique des CH au sein des clusters sur la base de leur énergie résiduelle est une solution possible pour équilibrer la consommation d'énergie de chaque cluster [27]. La mise en cluster augmente l'efficacité de la transmission des données en réduisant le nombre de capteurs qui tentent de transmettre des données à la station de base. L'agrégation de données au niveau des CH à travers la communication intra-clusters permet également d'éliminer

les problèmes de transmission de données. La mise en cluster est proposée en raison de l'évolutivité du réseau, d'économie d'énergie et de stabilité de la topologie du réseau.

1.3.2 Défis à relever dans les algorithmes de clustering

Les schémas de clustering jouent un rôle important dans les WSN. Ils peuvent améliorer efficacement les performances du réseau. Les schémas de clustering doivent tenir compte de plusieurs limitations essentielles des WSN [28].

- ***Énergie limitée :*** [28] les nœuds capteurs sans fil sont des capteurs de petite taille fonctionnant sur batterie, ils ont donc un stockage d'énergie limité. Il n'est pas toujours possible de recharger ou de remplacer leurs batteries après qu'elles soient épuisées.

  Les algorithmes de clustering sont plus efficaces sur le plan de la gestion énergétique que celui des algorithmes de routage direct. Ils peuvent être réalisés pour équilibrer la consommation d'énergie dans les nœuds capteurs en optimisant la formation des clusters, en réélisant périodiquement les CH en fonction de leur énergie résiduelle et en assurant une communication intracluster et intercluster.
- ***Faible durée de vie du réseau :*** [28] la limitation de l'énergie des nœuds réduit la durée de vie des nœuds d'un réseau. Les schémas de clustering aident à prolonger la durée de vie du réseau WSN en réduisant l'énergie utilisée dans la communication à l'intérieur et à l'extérieur des clusters.
- ***Capacités limitées :*** [28] la petite taille physique d'un nœud capteur limite les capacités des nœuds en termes de traitement, de mémoire, de stockage et de communication.
- ***Communication sécurisée :*** [28] la capacité d'un WSN à effectuer une communication sécurisée est de plus en plus requise. L'établissement de communications intra-cluster et intercluster sécurisées est l'un des défis importants dans la conception d'algorithmes de clustering puisque ces minuscules nœuds, lorsqu'ils sont déployés, ne sont pas sécurisés dans la plupart des cas.
- ***Formation des clusters et sélection du CH :*** [28] la formation de clusters et la sélection des CH sont deux des opérations importantes dans les algorithmes de clustering. La sélection de la taille optimale du cluster, l'élection, la réélection des CH et la maintenance des clusters sont les principaux problèmes à résoudre lors de la conception d'algorithmes de mise en cluster. Les critères de sélection pour isoler les clusters et pour choisir les CH doivent maximiser l'utilisation de l'énergie.
- ***Synchronisation :*** [28] lorsqu'on envisage un schéma de mise en clustering, la synchronisation et l'ordonnancement auront un effet considérable sur les performances globales du réseau. Les schémas de transmission par

créneaux (TDMA) programment régulièrement des intervalles de sommeil pour minimiser l'énergie utilisée. Ces schémas nécessitent des mécanismes de synchronisation pour établir et maintenir la transmission.

- ***Agrégation des données :*** [28] elle permet d'éliminer la duplication des données. Dans un grand réseau, il y a souvent plusieurs nœuds détectant des informations similaires. L'agrégation des données permet de différencier les données captées et les données utiles. De nombreux schémas de clustering fournissent ces capacités d'agrégation de données.
- ***Mécanismes de réparation :*** [28] en raison de leur nature, les WSN sont souvent sujets à la mobilité des nœuds, à leur mort, aux retards et aux interférences. Toutes ces situations peuvent entraîner une défaillance de la liaison. Lors de la conception de schémas de clustering, il est important de rechercher des mécanismes qui garantissent la récupération des liens et la fiabilité de la communication des données.
- ***Qualité de service (QoS) :*** [28] d'un point de vue global du réseau, nous pouvons examiner les exigences de qualité de service dans les WSN. Un grand nombre de ces exigences dépendent de l'application, comme par exemple le délai acceptable et la tolérance à la perte de paquets. Les algorithmes de clustering existants pour les WSN se concentrent principalement sur l'utilisation efficace de l'énergie du réseau, mais accordent moins d'attention à la prise en charge de la QoS dans les WSN. Les mesures de QoS doivent être prises en compte dans le processus de conception.

1.3.3 Gestion d'énergie dans le WSN à cluster

La gestion d'énergie est un aspect crucial dans les réseaux de capteurs sans fil (WSN), en particulier dans les WSN à cluster. Différentes stratégies ont été proposées pour la gestion d'énergie dans les WSN à cluster, telles que le clustering hiérarchique, le clustering basé sur l'emplacement et le clustering basé sur les performances. Ces approches ont pour objectif de réduire la consommation d'énergie en évitant les transmissions inutiles et en limitant la distance de transmission. Parmi les différentes approches proposées pour la gestion d'énergie dans les WSN à cluster, nous avons LEACH, HEED, EEHC, EECS, EEUC etc...

LEACH (Low-energy adaptive clustering hierarchy) [16] est un protocole qui forme des clusters en utilisant un algorithme distribué où les nœuds prennent des décisions autonomes sans aucun contrôle centralisé. Pour ce faire, un nœud décide d'être un CH avec une probabilité p et diffuse sa décision. Chaque nœud non CH détermine son cluster en choisissant le CH qui peut être atteint en utilisant le moins d'énergie de communication. L'algorithme permet d'équilibrer l'utilisation de l'énergie par une rotation aléatoire des CH. Il forme des clusters en fonction de la force du signal reçu et utilise les nœuds CH comme routeurs vers la station de base.

Tous les traitements de données, tels que la fusion et l'agrégation des données, sont effectués au sein du cluster.

Bien que LEACH ait été conçu pour réduire la consommation d'énergie des nœuds dans les réseaux de capteurs sans fil, il présente également des limites dans la gestion de l'énergie, telles que la variabilité de l'énergie des nœuds, l'utilisation inefficace de l'énergie, l'impact de la distance de communication et l'impact de la taille du cluster. Il est donc important de prendre en compte ces limitations lors du choix d'un protocole de clustering pour un réseau de capteurs sans fil.

EEHC (Energy Efficient Heterogeneous Clustered) [29] est un algorithme de clustering distribué et aléatoire pour les WSN. Les CH collectent les données des nœuds non-CH dans différents clusters et envoient un rapport agrégé à la station de base. Cette opération se déroule en deux phases : initiale et étendue. Dans la première phase, également appelée clustering à niveau unique, chaque nœud capteur s'annonce comme un CH avec une probabilité p aux nœuds voisins dans sa portée de communication. Ces CH sont appelés les CH volontaires. Tous les nœuds qui se trouvent dans un rayon de k sauts d'un CH reçoivent cette annonce soit par communication directe, soit par transfert. Tout nœud qui reçoit de telles annonces et qui n'est pas lui-même un CH devient membre du cluster le plus proche.

Les CH forcés sont des nœuds qui ne sont pas des CH et n'appartiennent pas à un cluster. Si l'annonce ne parvient pas à un nœud dans un intervalle de temps t prédéfini, calculé sur la base de la durée nécessaire à un paquet pour atteindre un nœud situé à k sauts, le nœud devient un CH forcé en supposant qu'il ne se trouve pas à moins de k sauts de tous les CH volontaires. La deuxième phase, appelée clustering multi-niveaux, construit h niveaux de hiérarchie de cluster. L'algorithme assure une connectivité à h sauts entre les CH et la BS.

Dans la communication inter-clusters, cet algorithme garantit que l'énergie dissipée par les CH éloignés de la station de base est réduite, car ces CH n'ont pas besoin de transmettre à la station de base(BS). Les CH les plus proches de la BS sont désavantagés, car ils sont des relais pour les autres CH. Bien que EEHC soit un protocole de clustering efficace pour réduire la consommation d'énergie des nœuds dans les réseaux de capteurs sans fil, il présente également des limites telles que la complexité de la gestion de cluster, l'impact de la taille du cluster, l'instabilité du réseau, l'impact de la densité du réseau et l'utilisation inefficace de l'énergie. Il est donc important de prendre en compte ces limitations lors du choix d'un protocole de clustering.

HEED (Hybrid, Energy-Efficient, Distributed clustering) [30] : est un algorithme de clustering multi-saut pour les WSN. Les CH sont choisis en fonction de deux paramètres importants : l'énergie résiduelle et le coût de communication intra-cluster. L'énergie résiduelle de chaque nœud est utilisée pour choisir de manière probabiliste, l'ensemble initial de CH, comme cela se fait couramment dans d'autres schémas de

clustering.

Dans HEED, le coût de communication intra-cluster ou la densité du cluster reflète le degré du nœud ou la proximité des nœuds avec le voisin est utilisé par les nœuds pour décider de rejoindre le cluster. Les faibles niveaux de densité des clusters favorisent une augmentation de la réutilisation spatiale tandis que les niveaux de densité élevés des clusters sont nécessaires pour la communication inter-clusters car ils couvrent deux zones de clusters ou plus.

HEED fournit une distribution uniforme des CH à travers le réseau et un meilleur équilibrage de la charge. Cependant, la connaissance de l'ensemble du réseau est nécessaire pour déterminer le coût de la communication intra-cluster. HEED est un protocole efficace pour réduire la consommation d'énergie des nœuds dans les réseaux de capteurs sans fil. Cependant,il présente également des limites telles que la sensibilité à la densité du réseau, la durée de vie de la batterie, le délai de transmission des données, la sensibilité aux paramètres et l'inégalité de l'énergie. Il est donc important de prendre en compte ces limitations pour améliorer le protocole dans sa gestion efficace de l'énergie.

EECS (Energy Efficient Clustering Scheme) [31] est un algorithme de mise en cluster dans lequel les candidats au poste de chef de cluster (CH) sont en concurrence pour la possibilité d'accéder à ce poste pour un tour donné. Cette compétition implique que les candidats diffusent leur énergie résiduelle aux candidats voisins. Si un nœud donné ne trouve pas de nœud avec plus d'énergie résiduelle, il devient un CH. La formation des clusters est différente de celle de LEACH. LEACH forme des clusters en fonction de la distance minimale des nœuds à leur CH correspondant.

EECS étend cet algorithme en dimensionnant dynamiquement les clusters en fonction de leur distance par rapport à la station de base. Le résultat est un algorithme qui résout le problème suivant : les clusters situées à une plus grande distance de la station de base nécessitent plus d'énergie pour la transmission que celles qui sont plus proches. En fin de compte, cela améliore la distribution de l'énergie dans le réseau, ce qui se traduit par une meilleure utilisation des ressources et une durée de vie prolongée du réseau.

Cependant, les clusters plus proches de la station de base peuvent devenir congestionnés, ce qui peut entraîner la mort précoce du CH. Bien que EECS soit un protocole de clustering efficace pour améliorer la durée de vie de la batterie des nœuds dans les réseaux de capteurs sans fil, il présente également des limites telles que la sensibilité à la densité du réseau, les problèmes de convergence, l'équilibre de l'énergie et la sensibilité aux paramètres.

Dans les WSN multi-sauts, il existe un problème de point chaud : les CH plus proches de la BS ont tendance à mourir plus rapidement, car ils relaient beaucoup plus de trafic que les nœuds distants. EEUC (Energy-efficient unequal clustering) dans [26], propose d'équilibrer la consommation d'énergie entre les clusters. Il

préconise ainsi, pour plus d'économie d'énergie dans les communications intra-cluster et intercluster, que la taille des clusters proches du nœud récepteur soit beaucoup plus petite que celle des clusters éloignés du nœud récepteur. En fait, EEUC est un système basé sur la distance similaire à EECS et il exige également que chaque nœud ait une connaissance globale de sa position et de sa distance par rapport au nœud récepteur. Il tente de prolonger la durée de vie du réseau et d'équilibrer la charge entre les nœuds. Il résout le problème des points chauds ; la taille des clusters est proportionnelle à la distance à la station de base.

Cependant, l'agrégation globale supplémentaire des données ajoute des frais généraux à tous les nœuds capteurs et dégrade les performances du réseau, en particulier pour un réseau à bonds multiples.

la plupart des protocoles de mise en cluster existants consomment de grandes quantités d'énergies, en raison de la surcharge liée à la formation des clusters et de la mise en cluster à niveau fixe, en particulier lorsque les nœuds capteurs sont déployés de manière dense dans les WSN. Pour résoudre ce problème, le protocole PEACH(Power-efficient and adaptive clustering hierarchy) est proposé [32] pour les WSN afin de minimiser la consommation d'énergie de chaque nœud et de maximiser la durée de vie du réseau. Dans le protocole PEACH, la formation de clusters est effectuée en utilisant les caractéristiques de sur-écoutes de la communication sans fil pour prendre en charge le regroupement adaptatif à plusieurs niveaux et éviter les frais généraux supplémentaires. Dans les WSN, le fait de surprendre un nœud peut permettre de reconnaître la source et la destination des paquets transmis par les nœuds voisins. PEACH est applicable aux réseaux de capteurs non conscients de l'emplacement et conscients de l'emplacement. Grâce à ses caractéristiques de sur-écoute, PEACH économise la consommation d'énergie de chaque nœud et prolonge ainsi la durée de vie du réseau.

Cependant, PEACH présente également des limites telles que la sensibilité à la densité du réseau, la sensibilité aux nœuds défectueux, la sensibilité aux changements de topologie et la complexité. Il est donc important de prendre en compte ces limitations pour proposer d'autres protocoles qui repondent efficacement au besoin de gestion d'énergie dans les réseaux de capteurs.

Il est traité dans [33] MRPUC (Multihop Routing Protocol with Unequal Clustering), un schéma de clustering distribué qui fonctionne par tours. Chaque tour est séparé en trois phases : configuration du cluster, formation du routage multi-sauts inter-clusters et transmission des données. Chaque nœud rassemble les informations corrélatives de ses nœuds voisins et élit un nœud avec une énergie résiduelle maximale comme CH. Les CH les plus proches de la station de base ont des clusters de plus petite taille afin d'économiser l'énergie nécessaire à la lourde tâche de transmission inter-clusters. Les nœuds réguliers rejoignent les clusters où les CH ont plus d'énergie résiduelle et sont plus proches d'eux. Un arbre de routage inter-clusters est

construit comme épine dorsale du réseau et les données sont transmises à la station de base par multi-sauts.

Cet algorithme empêche la mort précoce des CH car la communication inter-clusters dépend aussi de l'énergie résiduelle. Les CH se dirigent vers le CH voisin ayant l'énergie résiduelle la plus élevée. La formation du routage multi-sauts inter-clusters peut entraîner une surcharge supplémentaire. MRPUC présente également des limites dans la gestion de l'énergie, telles que la sensibilité à la taille du cluster, la sensibilité aux nœuds défectueux, la sensibilité aux interférences et la nécessité d'un recalibrage périodique. le MRPUC est un mécanisme de routage multi-sauts efficace pour économiser l'énergie dans les réseaux de capteurs sans fil. Cependant, il comporte également des limites en termes de gestion de l'énergie, de complexité de mise en œuvre et de limitation de la couverture.

Afin de minimiser la consommation d'énergie et de prolonger la durée de vie du réseau de capteurs, il a été proposé dans [34] LEACH-B, est une amélioration du protocole LEACH pour les réseaux de capteurs sans fil. Le protocole doit s'assurer que la répartition des clusters est équilibrée et uniforme. Pour atteindre cet objectif, le nombre de CH doit être dominant et le réseau a besoin d'un nombre optimal de CH. A chaque tour, après la première sélection de la tête de cluster conformément au protocole LEACH, une seconde sélection est introduite pour modifier le nombre de têtes de cluster en tenant compte de l'énergie résiduelle du nœud. Cependant, ce protocole présente plusieurs limites dans sa gestion d'énergie, telles que l'inégalité de la durée de vie de la batterie entre les nœuds, une charge de communication élevée, l'absence de prise en compte des défaillances potentielles des nœuds, et l'impossibilité d'ajuster dynamiquement la taille des clusters.

Les auteurs de [27], contrairement à LEACH, au lieu d'une transmission entre chaque CH et le nœud récepteur (sink), propose une solution qui consiste à élire un chef de cluster principal pour la transmission des données des différents chefs de cluster à la station de base. Elle se dessine suivant trois principales étapes. Premièrement, il y a un déploiement aléatoire des nœuds et la formation des clusters. Deuxièment, la sélection des têtes de clusters et pour terminer, et une communication intra-cluster pour choisir la tête de cluster principale parmi les différentes têtes de cluster afin de minimiser la puissance nécessaire à la transmission entre les CH et le nœud sink.

Pour la sélection des têtes de cluster principales, chaque tête de cluster choisit d'abord sa tête de cluster maître et envoie ses données à ce nœud particulier. Ces nœuds principaux acceptent les paquets de données des têtes de cluster et envoient ensuite les paquets de données agrégés au nœud récepteur (sink). Dans les réseaux de capteurs, la consommation d'énergie est due en majorité à la communication longue portée. Leur principale motivation est donc de réduire la communication longue distance entre les têtes de cluster et le nœud récepteur (sink). Selon eux, cette technique permet d'améliorer considérablement la durée de vie du réseau.

Le protocole de routage optimisé adopté dans le [35] par ses auteurs, PDORP (PEGASIS-DSR) utilise de manière optimale les caractéristiques du modèle de routage proactif et réactif. Si un nœud devient plus agressif au moment du transfert et qu'auparavant il n'était pas dans la mémoire cache, le nœud rélais va forcément recevoir un paquet de sa part et de cette manière, il peut endommager les routes existantes. Une solution à ce problème pourrait être la vérification de n'importe quel nœud au moment de la réception d'un paquet de données, mais cela entraînerait un retard inutile. Par conséquent, la solution proposée crée une liste de confiance pour la première fois à chaque tour sur la base des paramètres attribués aux nœuds. Après chaque tour, la liste de confiance est mise à jour et après un certain nombre de tours, la confiance ne sera pas vérifiée pour éviter les retards.

Lorsqu'un nœud source veut transmettre des données au nœud de destination, il calcule la distance de tous les voisins et transmet les données au nœud dont la distance est inférieure ou égale à la distance seuil et uniquement dans la direction des nœuds de destination. Il s'assure également que le nœud voisin de distance minimale est dans la direction du nœud de destination. Après ce processus, tous les nœuds en direction de la destination sont ajoutés à la liste de confiance uniquement lors du premier tour de simulation. Chaque fois qu'une nouvelle transmission de données est nécessaire, la liste de confiance est mise à jour au cours du premier tour de simulation et les données sont transférées uniquement via les nœuds qui se trouvent dans le vecteur de confiance. Cependant, bien que PDORP intègre des mécanismes pour limiter la consommation d'énergie des nœuds, les limites de gestion d'énergie peuvent encore exister en raison des caractéristiques matérielles des nœuds et de l'environnement dans lequel ils sont déployés.

Le modèle cross layer-LEACH [36] efficient en énergie pour un réseau de capteurs sans fil est proposé dans cet article. Le problème de la collecte de données de capteur corrélées à partir d'un nœud récepteur dans un WSN y est traité. La plupart des approches existantes mettent l'accent sur l'optimisation de la couche de routage uniquement. Mais, dans [36] les auteurs tente de maximiser la durée de vie du réseau en considérant la couche de routage, la couche physique et l'accès au lien dans la couche MAC. De plus, l'énergie résiduelle et la distance entre le nœud et la station de base sont prises en compte pour la sélection de la tête de cluster. L'énergie consommée lors de la transmission des données entre la tête de cluster et la station de base est directement proportionnelle à la distance qui les sépare. Après le mécanisme de routage, le modèle CL-MAC est traité en prenant la valeur seuil, l'énergie restante et le nœud comme entré.

L'algorithme proposé par Gupta et al dans [37], comprend quatre phases. Une phase de déploiement du capteur, suivit d'une d'une phase de configuration et de détermination du chemin, puis, la phase d'optimisation des données et enfin, la phase de transfert des données. Les nœuds capteurs sont déployés dans une topologie de

grille triangulaire. La zone entière est divisée en n carrés. Le nombre de nœuds capteurs dans chaque cluster est calculé comme m/n. La mise en cluster est déployée de deux façons, c'est-à-dire la mise en cluster régulière et irrégulière. Dans la mise en cluster régulière, la zone/rayon de chaque cluster reste la même. Dans la mise en cluster irrégulière, la zone ou rayon de chaque cluster diminue suivant une valeur constante au fur et à mesure qu'elle s'approche de la station de base.

Pour mettre en place le clustering et déterminer le chemin, initialement, le nœud au centre du cluster est choisi comme tête de cluster. En utilisant la formule du point médian, le centre de chaque cluster (qui est carré) est calculé et le nœud correspondant à ce point est assigné comme CH1 par la station de base. Le CH1 diffuse à tous les nœuds capteurs qu'il est la tête de cluster (l'énergie consommée à ce stade est très faible). Tous les nœuds capteurs dans le cluster auront l'ID de CH1 comme adresse pour transmettre leurs données détectées. Les données qui sont détectées par les nœuds capteurs sont transmises au CH1 via un seul saut. Toutes les données reçues par le CH sont traitées, filtrées et transmises à la station de base (BS) via un saut multiple en utilisant l'algorithme du plus court chemin.

Après le premier tour, la station de base n'est plus responsable de l'attribution du nouveau CH. Le CH actuel diffuse à nouveau un message (les signaux de diffusion ne sont envoyés que dans la moitié du rayon de la zone totale) informant de son niveau d'énergie. Cette énergie du CH est comparée à celle de tous les autres nœuds du cluster et le nœud ayant l'énergie la plus élevée est désigné comme nouveau CH. Sinon le CH1 reste le CH et la procédure se répète. Pour l'optimisation des données, le même niveau d'énergie et un identifiant unique sont attribués à chaque nœud capteur.

Toutes les solutions de clustering proposées plus haut cherchent certes à resoudre la problématique de consommation d'énergie à travers différentes techniques. Cepandant aucune de ces solutions n'attribue un quatum d'énergie à leur CH pour son reigne de commandement et au bout duquel il devra se faire remplacé dans son rôle de CH. Aussi, les mecanismes d'equilibrage des nœuds entre les clusters contribuent à une équilibrage de charge dans le réseau, alors que peu auteurs exploitent cette technique et même les approches proposées allant dans ce sens peuvent être améliorées. Nous pensons que l'exploitation de ses mécanismes pourraient apporter un plus à la gestion de consommation d'énergie dans les réseaux de capteurs.

1.4 Paradigme Software Defined Networking

dans le fonctionnement traditionnel des réseaux avant l'avènement de SDN, la gestion des nœuds était faite de façon distribuée, où le plan de contrôle et le plan de données étaient sur le même équipement réseau. Le plan de contrôle prenait toutes

les décisions concernant les tables de routage et l'appareil devait être configuré manuellement et séparément, ce qui pouvait s'avérer fastidieux dans des déploiements avec des centaines d'appareils à gérer [38]. De plus, chaque appareil travaillait de manière autonome et partageait des informations avec ses voisins pour obtenir une vue du réseau, et les nouveaux protocoles de routage ne pouvaient pas être mis en œuvre facilement. L'intégration d'appareils de marques différentes était difficile en raison de l'utilisation de logiciels propriétaires.

SDN a rapidement abandonné cette façon de mettre les deux plans sur le même équipement et a implémenté un plan [39] de contrôle commun sur un contrôleur distant pour toutes les entités du réseau. SDN est un nouveau modèle de réseau qui sépare le plan de contrôle du plan de données d'un dispositif, et le plan de contrôle est rendu programmable à l'aide des API pour agir sur le plan de données à distance.

cette approche innovante de la gestion des réseaux offre ainsi des avantages significatifs en termes de flexibilité, de performances et de sécurité. Dans cette sous section, nous explorons les différents aspects de la technologie SDN, y compris son architecture, ses avantages et ses défis, ainsi que les différents modèles et protocoles liés à cette technologie. Nous examinons également de plus près les protocoles OpenFlow, ForCES et OnePK et leur rôle dans la mise en œuvre de la technologie SDN.

1.4.1 Architecture, avantages et défis de la technologie SDN

1.4.1.1 Architecture SDN

SDN (Software-Defined Networking) ou réseau défini par logiciel désigne un ensemble de technologies innovantes qui a pour but de fournir un contrôle centralisé des ressources réseau, une meilleure programmabilité et combinaison d'un ensemble de ces ressources, ainsi que la virtualisation de ces ressources en les dissociant des éléments physiques du réseau. SDN (Software Defined Network) peut être vu aussi comme une méthode d'architecture de réseau qui permet d'utiliser des applications logicielles pour contrôler ou "programmer" le réseau de manière intelligente et centralisée. Cela permet aux opérateurs de gérer l'ensemble du réseau de manière cohérente et complète, quelle que soit la technologie de réseau sous-jacente.

Une architecture SDN [38] comporte trois couches décrites ci-après :

- une couche application : elle prend en charge les applications qui communiquent avec le contrôleur et lui ordonnent d'exécuter les fonctions souhaitées sur l'infrastructure de réseau physique sous-jacente. Ces applications utilisent également les données fournies par le contrôleur pour créer une vue logique de l'ensemble du réseau. Cela aide les administrateurs de réseau à prendre des décisions concernant la gestion du réseau. Ces applications peuvent également être utilisées pour effectuer des analyses de données. Les applications orientées

métier sont utilisées pour gérer les grands centres de données ou pour détecter toute activité réseau suspecte au sein du centre de données à des fins de sécurité.

- une couche contrôle : elle contient le contrôleur de réseau qui est la principale entité qui fait le lien entre la couche application et la couche infrastructure. Le contrôleur est responsable de la gestion de la communication entre les deux couches. Il transmet les instructions reçues des applications aux dispositifs physiques ou virtualisés sous-jacents et collecte les données de ces dispositifs et les renvoie aux applications.
- une couche infrastructure : elle est constituée de dispositifs physiques de mise en réseau qui exécutent le transfert réel des données. Elle comprend également les éléments virtualisés.

L'architecture SDN est généralement décrite par deux interfaces, à savoir l'interface northbound et l'interface southbound. La connexion entre le contrôleur et les applications est appelée interface northbound, tandis que la connexion entre le contrôleur et le matériel de réseau physique est appelée interface southbound.

1.4.1.2 Avantages du SDN

Le SDN possède des fonctions de découplage inhérentes au plan de contrôle et au plan de données et peut mieux contrôler le réseau grâce à la programmation. Ces fonctions apporteront des avantages potentiels d'une configuration améliorée, de meilleures performances et encouragera l'innovation dans l'architecture et le fonctionnement du réseau. Par exemple, le contrôle adopté par SDN peut inclure non seulement la transmission de paquets de données au niveau de l'échange, mais également le protocole de liaison au niveau de la liaison de données, brisant ainsi les barrières qui se chevauchent.

De plus, le SDN a pour fonction d'obtenir l'état instantané du réseau et peut fournir un contrôle réseau centralisé en temps réel en fonction de l'état instantané du réseau et des politiques définies par l'utilisateur. Il peut également optimiser la configuration du réseau et améliorer les performances du réseau.

SDN fournit une plate-forme pratique pour expérimenter de nouvelles technologies et encourager de nouvelles conceptions de réseaux. Ce qui prouve encore les avantages potentiels de SDN en raison de sa programmabilité réseau et de sa capacité à définir des réseaux virtuels isolés grâce au plan de contrôle. Dans cette section, nous mettons en évidence ces avantages de SDN [40].

1.4.1.3 Défis du SDN

Avec la promesse d'une configuration améliorée, de performances améliorées et d'encourager l'innovation, SDN en est encore à ses balbutiements. De nombreux problèmes fondamentaux n'ont pas encore été complètement résolus, parmi lesquels

le besoin le plus urgent de normalisation et d'adoption [40]. Bien que la définition SDN de l'ONF soit la plus populaire, OpenFlow, parrainé par l'ONF, n'est en aucun cas la seule norme SDN, ni une solution mature. L'implémentation des contrôleurs SDN manque toujours du pilote OpenFlow open source et le développement des applications SDN manque toujours de l'API Nord standard ou du langage de programmation de haut niveau.

Le SDN fournit une plate-forme pour les technologies de réseau innovantes, mais la transition des réseaux traditionnels vers la technologie SDN peut être perturbatrice et douloureuse. Les préoccupations communes incluent l'interopérabilité de la technologie SDN et des équipements réseau antérieurs, les performances et la confidentialité du contrôle centralisé et le manque d'experts en assistance technique. Les déploiements SDN existants sont généralement limités aux petites plates-formes de test utilisées pour la recherche de prototypes.

1.4.2 Différents modèles de la technologie SDN

La technologie SDN englobe toutes les solutions permettant une programmation du réseau, afin de mieux interagir avec les applications. Diverses solutions coexistent, adaptées selon les besoins des utilisateurs. On comprend aisément que les solutions permettant de simplifier le déploiement d'une application dans un "datacenter" ne sont pas les mêmes que celles qui permettront de mieux contrôler l'éclairage d'une ville à travers le réseau. On peut noter différents modèles de programmabilité [1], la programmabilité à travers un contrôleur et les "*overlay*".

1.4.2.1 Programmabilité à travers un contrôleur

Le modèle de programmabilité de contrôleur est le plus connu en tant que solution SDN. Dans ce modèle, une application donne une commande abstrait à un contrôleur, qui à son tour translate cette commande en une suite d'ordres auprès des dispositifs du réseau en question. Le rôle de la couche de contrôle est d'implémenter des règles sur le commutateur SDN. Ces règles peuvent impliquer une allocation de VLAN (ports d'accès), le routage et la gestion spécifique des services réseau (équilibrage de charge, haute disponibilité, QoS, etc.). Ce modèle est le modèle le plus connu et le plus utilisé, car il simplifie le réseau. Le contrôleur masque la complexité du réseau. On peut distinguer plusieurs cas en fonction du type de commande échangée entre le contrôleur et le nœud réseau. Au début, le problème était d'avoir la programmabilité du plan de données à travers une API, mais le nouveau modèle va implémenter un modèle qui donne au périphérique un ordre plus abstrait, tandis que ce dernier peut les implémenter librement et de façon convenable.

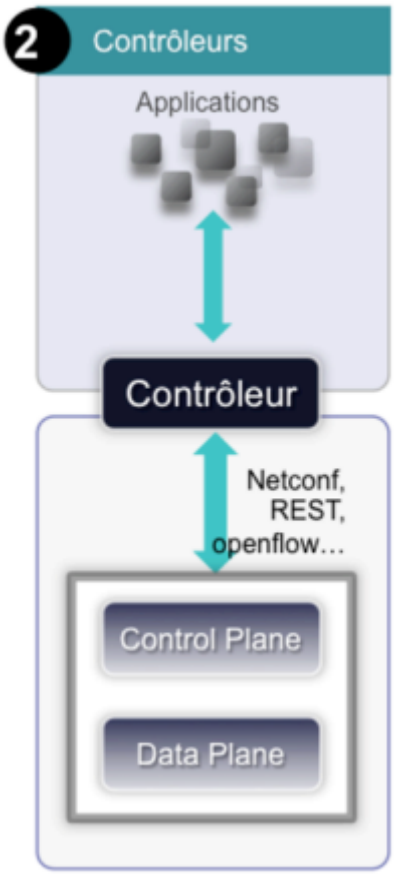

FIGURE 1.5 – Modèle SDN avec programmabilité via un contrôleur [1]

1.4.2.2 Overlay

L'overlay est un modèle SDN qui permet la création d'un réseau virtuel au-dessus du réseau physique. Dans ce modèle, les applications créent leur propre réseau, surmontant ainsi les limites du réseau physique sous-jacent. Ce dernier n'a pour mission que la simple connectivité entre les nœuds d'extrémité des tunnels et le réseau d'overlay assure l'intégralité des services. On parle également de virtualisation des fonctions réseau en anglais Network Function Virtualization (NFV) quand les équipements d'interconnections (routeurs, commutateurs, parefeu) sont des éléments virtualisés.

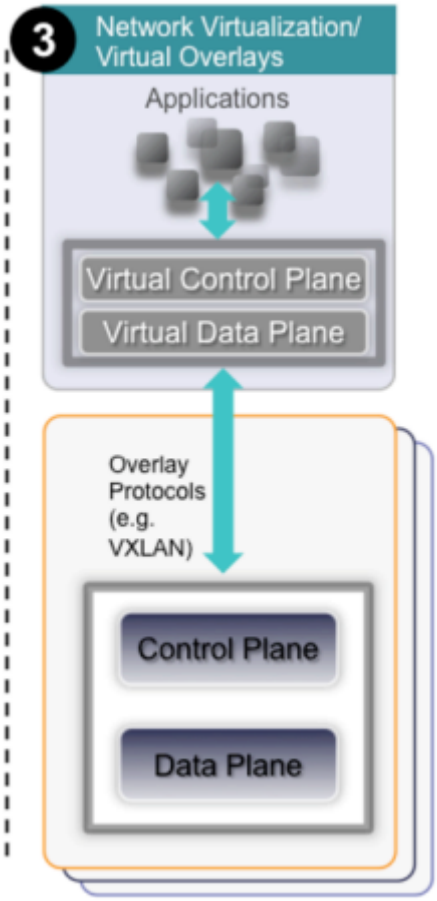

FIGURE 1.6 – Modèle SDN avec overlay [1]

1.4.3 Protocoles liés à la technologie SDN

Différents protocoles sont utilisés par le contrôleur. Parmi ces protocoles, nous pouvons citer ForCes [2], OpenFlow [41] et OnePK [42]. ForCes est un protocole open source mais très peu utilisé. OpenFlow est le protocole le plus connu et le plus utilisé dans les contrôleurs, il est également open source. Nous avons aussi OnePK de Cisco qui est une solution propriétaire. Dans cette section, nous présentons plus en détail ces trois protocoles.

1.4.4 OpenFlow

OpenFlow est actuellement le système le plus couramment déployé comme technologie de mise en réseau défini par logiciel (SDN) [41]. Le SDN se compose en découplant les plans de contrôle et de données d'un réseau. Un contrôleur logiciel est chargé de gérer les informations de transmission d'un ou de plusieurs commutateurs. Le matériel ne gère que l'acheminement du trafic selon les règles fixées par le contrôleur.

OpenFlow est une technologie SDN proposée pour normaliser la façon dont un

contrôleur communique avec les appareils du réseau dans une architecture SDN. Elle a été proposée pour permettre aux chercheurs de tester de nouvelles idées dans un environnement de production. OpenFlow fournit une spécification permettant de faire migrer la logique de contrôle d'un commutateur vers le contrôleur. Il définit également un protocole pour la communication entre le contrôleur et les commutateurs. Les architectures basées sur OpenFlow ont des capacités spécifiques qui peuvent être exploitées par les chercheurs pour expérimenter de nouvelles idées et tester de nouvelles applications. Ces capacités comprennent l'analyse du trafic basée sur des logiciels, le contrôle centralisé, la mise à jour dynamique des règles d'acheminement et l'abstraction de flux.

Les applications basées sur OpenFlow ont été proposées pour faciliter la configuration d'un réseau, pour simplifier la gestion des réseaux et l'ajout de fonctions de sécurité, la virtualisation des réseaux et des centres de données et de déployer des systèmes mobiles. Ces applications fonctionnent sur des systèmes d'exploitation de réseau tels que Nox, Beacon, Maestro, Floodlight ou Trema.

1.4.5 Protocole ForCes

Le groupe de travail ForCES de l'IETF (GT ForCES) a été créé en 2001, au moment de l'émergence des processeurs de réseau spécialisés. A l'origine, le principal moteur de la normalisation dans ce domaine était le besoin d'interfaces programmables ouvertes et normalisées pour les processeurs de réseau disponibles sur le marché. En pratique, la normalisation de ForCES a été conduite à l'époque par les besoins d'un forum et a été influencée par les concepts de réseaux programmables et l'initiative P1520.

Le cadre ForCES fournit la vue architecturale pour l'approche des réseaux programmables [2]. Le cadre est basé sur les éléments présentés dans la Figure 1.7. L'objectif du groupe de travail ForCES était de séparer les plans de transmission et de contrôle, ce qui est réalisé par la définition d'un ensemble de protocoles et d'un modèle nécessaire pour séparer les fonctionnalités du plan de contrôle telles que les protocoles de routage, les protocoles de signalisation et le contrôle d'admission, des activités du plan de transmission de données par paquet tel que la transmission de paquets, la mise en file d'attente et la modification de l'en-tête.

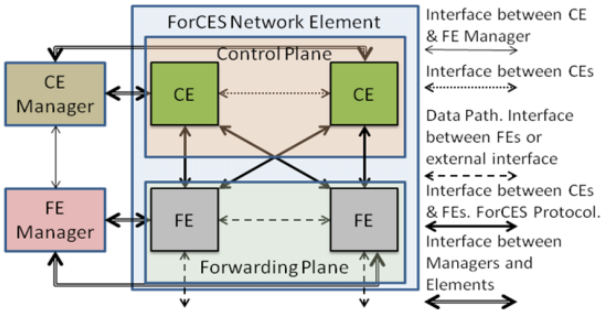

FIGURE 1.7 – Cadre de travail de ForCes [2]

Dans ForCES, un élément de réseau (NE) est composé d'éléments de contrôle (EC) et d'éléments de transmission (FE). Pour qu'un EC puisse contrôler un FE, le groupe de travail a décidé de développer un modèle d'abstraction pour les FE en se concentrant sur le plan de transmission. Pour faire face aux différentes fonctionnalités du plan de transport, le WG ForCES a normalisé un langage de modélisation qui permet aux développeurs de définir leurs propres modèles d'abstraction FE.

1.4.6 OnePK

Cisco Open Network Environment (ONE) est une solution complète qui permet aux réseaux de devenir plus ouverts, programmables et orientés applications. Il s'agit d'un portefeuille de technologies complémentaires qui comprend des API de plate-forme, des technologies de contrôleur et d'agent et des technologies de réseau de superposition [42]. Les API de plate-forme sont un élément clé de la stratégie Cisco ONE et font partie d'un kit de développement logiciel (SDK) appelé ONEPlatform Kit (onePK).

OnePK est un ensemble complet d'API de plate-forme, fournissant un accès programmatique en duplex intégral aux appareils Cisco. Il s'agit d'un kit d'outils qui permettent aux développeurs de créer des applications personnalisées qui interagissent avec les appareils du réseau Cisco d'une manière qui n'a jamais été possible auparavant. OnePK permet aux développeurs d'intégrer l'automatisation et l'orchestration dans le réseau. Les avantages de onePK peuvent être obtenus grâce à une mise à jour logicielle des routeurs et des commutateurs existants qui assure la protection des investissements et un chemin évolutif vers un réseau défini par logiciel. OnePK simplifie les opérations en permettant l'automatisation des fonctions, créant ainsi l'efficacité. Il fournit un cadre pour l'innovation, en donnant accès aux dispositifs de réseau, qui permettent de créer de nouveaux services. Enfin, OnePK augmente l'agilité en permettant de nouvelles fonctionnalités et en étendant les fonctionnalités des dispositifs, ce qui permet d'aller plus vite. Il en résulte une riche interaction entre les applications et l'infrastructure sous-jacente.

Les fonctionnalités de onePK représentent un modèle SDN hybride. Toutefois, ces fonctionnalités peuvent être utilisées pour faciliter la construction d'un système basé sur un SDN classique. Par exemple, un onePK peut être utilisé pour mettre en œuvre des agents OpenFlow ou être utilisé par un contrôleur réseau, chacun de ces éléments représentant des éléments typiques d'une architecture SDN classique.

1.5 Réseaux de capteurs définis par logiciel

Initialement, la technologie SDN était faite pour les réseaux avec infrastructures. Dans la décennie passée, les chercheurs se sont lancés pour l'application de cette technologie dans les réseaux de capteurs sans fil. Dans cette section, nous faisons cas des motivations de l'avènement de SDN dans le WSN. Ensuite, nous présentons les architectures et topologies de SDWSN. Aussi, nous exposons les apports de SDN dans les WSN. Enfin, nous décrivons quelques architectures SDWSN.

1.5.1 Motivations de l'avènement SDN dans les WSN

La nature spécifique de l'application des WSN les empêche d'utiliser tous leurs potentiels. De multiples WSN sont déployés pour de multiples applications dans la même zone [5]. De même, les fournisseurs ne parviennent pas à utiliser les fonctionnalités communes, car ils développent les WSN de manière isolée. En outre, la nature du déploiement à distance des WSN nécessite des dispositifs hautement autonomes et autoconfigurables, ce qui n'est pas réalisable en raison de la limitation des ressources de ces dispositifs [43]. Certains des problèmes communs aux WSN sont entre autres de l'économie d'énergie, la mobilité des nœuds capteurs, la gestion du réseau, la précision de la localisation et les WSN virtualisés. Tous les défis susmentionnés peuvent être relevés efficacement en utilisant le SDN.

Le SDN encourage le développement de protocoles rentables qui peuvent conduire à une augmentation considérable de la productivité du WSN. La séparation du plan d'acheminement de la logique de contrôle facilite la gestion du réseau et permet sa virtualisation. En outre, le récent regain de popularité de l'Internet des objets (IoT) a entraîné la production et le déploiement à grande échelle des WSN. Au cours de la prochaine décennie, des milliards de nœuds capteurs interconnectés pourraient être reliés par Internet. Le SDN peut fournir une plate-forme solide pour gérer un grand nombre de dispositifs en réseau et résoudre certains des principaux problèmes rencontrés par les WSN. En particulier, les caractéristiques les plus significatives qui peuvent être réalisées en utilisant des nœuds WSN compatibles SDN sont la gestion des nœuds et la gestion des ressources [7]. Un contrôleur peut prendre en compte l'énergie disponible pour les différents nœuds lors des décisions de routage afin d'améliorer la durée de vie du réseau.

Habituellement, les nœuds WSN sont considérés comme des dispositifs jetables

et spécifiques à une application. Mais compte tenu de leur utilisation dans divers domaines où les nœuds capteurs doivent collecter, traiter et transmettre différents types de données pour différentes applications, ils ont besoin d'un cadre solide dans lequel une bien meilleure utilisation de l'infrastructure sous-jacente peut être réalisée grâce au déploiement du SDN.

Un autre avantage clé de l'utilisation du SDN est que si une prise dans un réseau indique au contrôleur qu'un dispositif montre des signes de détournement, alors le contrôleur peut diriger le trafic de ce dispositif vers un système de détection d'intrusion (IDS) pour une analyse plus approfondie [44]. Cette approche peut s'avérer très utile pour le domaine WSN.

1.5.2 Architecture générale d'un SDWSN

En fonction des principales caractéristiques de la technologie SDN, une architecture intégrée basée sur la combinaison de SDN et WSN est présentée [3]. Le SDWSN comprend trois couches différentes : la couche application, la couche contrôle et la couche de transmission [45]. La couche d'application comprend le contrôle de la topologie, l'optimisation du routage, la simulation et les applications spéciales. La couche contrôle réceptionne les règles ou commandes venant de la couche application, les traduit en des règles de flux grâce à une API sud et ensuite les envois à la couche inférieure où elles seront enregistrées dans de tables de flux. La couche de transfert a pour principal rôle d'interpréter les règles de flux enregistrées dans la table de flux afin de prendre une décision d'acheminement des données reçues. S'il n'y a pas de règle de flux correspondante pour une donnée reçue. Un message "*Request*" est envoyé au contrôleur demandant la règle appropriée. La Figure 1.8 représente une architecture générale d'un SDWSN.

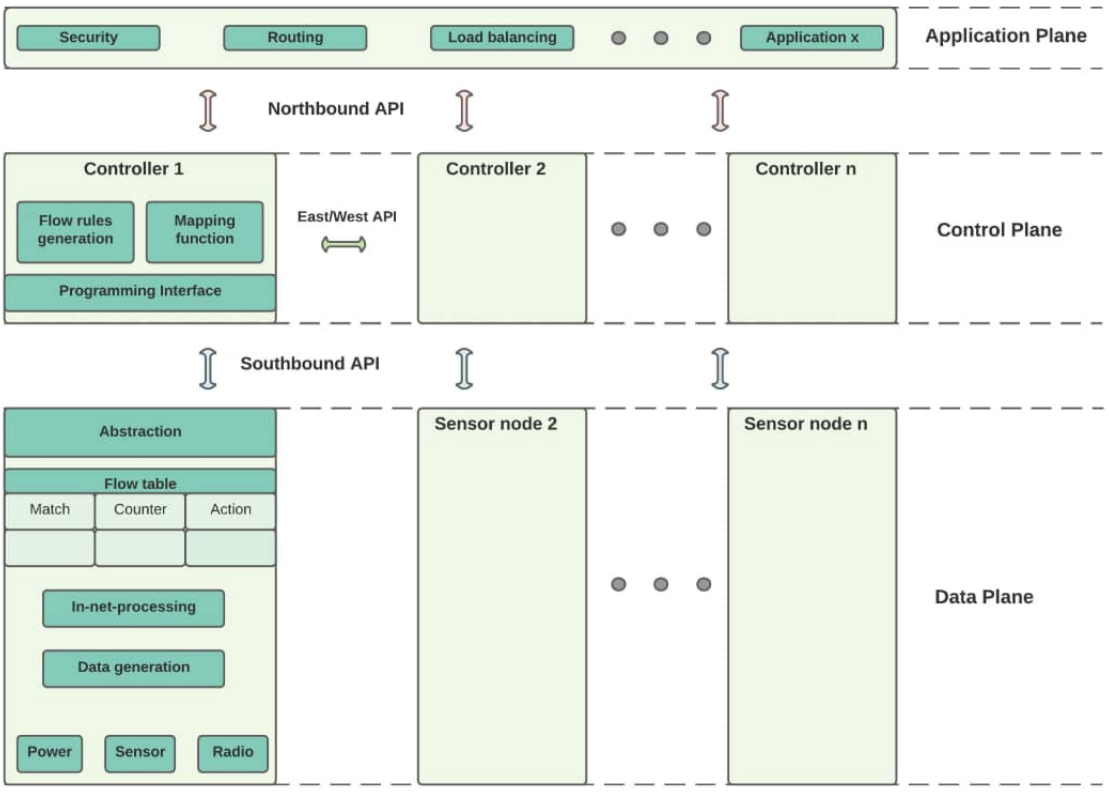

FIGURE 1.8 – Architecture générale d'un SDWSN [3]

Selon la dimension et la structure du réseau, la couche de contrôle peut être configurée dans la station de base ou dans la passerelle ou dans un nœud intermédiaire ou la tête de cluster. Pour un WSN à petite échelle, il est judicieux d'utiliser une structure de contrôle centralisée où la couche de contrôle peut être placée dans la station de base ou dans la passerelle. Par contre, pour un WSN à grande échelle, le contrôleur principal pourrait être configuré dans le nœud de passerelle et le contrôleur secondaire configuré dans le nœud de cluster en fonction de l'état du clustering. La transmission des messages entre le contrôleur et les nœuds repose sur le protocole Sensor Open Flow (SOF) [5]. Le SDWSN centralisé définit les types de messages de transmission spécifiques suivants : gestion, instruction et retour. Les messages de gestion font référence à l'état spécifique du nœud (énergie, taux d'utilisation) et à sa position (emplacement du nœud et nœuds voisins). Les messages d'instruction sont principalement utilisés pour mettre à jour la table des flux dont la structure comprend différents champs tels que la correspondance, la priorité, le compteur, l'action, le délai d'attente et le cookie. Les messages de retour sont utilisés pour prendre en charge la communication entre les nœuds communs et les nœuds de contrôle.

1.5.3 Topologie du SDWSN

La topologie du réseau désigne la disposition physique d'une variété de dispositifs interconnectés par des moyens de transmission. La topologie du réseau SDWSN peut être fixe ou mobile. Les SDWSN fixes peuvent être classés en deux catégories : les SDWSN centralisés utilisés dans les réseaux à petite échelle et les SDWSN distribués déployés dans les réseaux à grande échelle [46].

1.5.3.1 SDWSN centralisé

Dans le réseau SDWSN centralisé [4], les contrôleurs SDN passent par la passerelle, pour gérer l'ensemble du réseau de capteurs, y compris les comportements de travail et de sommeil des nœuds, le choix de routage et la transmission des données. Le travail dans [43] a proposé une station gérée qui utilise un intergiciel pour concevoir des informations de contrôle contenant une table de flux, une fonction de mappage et des informations. Le contrôleur SDN pour gérer tous les nœuds et l'unité de gestion du réseau sont déployés dans la station de base [47]. La couche d'application prend principalement en compte l'algorithme de localisation et de suivi. Les avantages du mode centralisé du SDWSN sont détaillés comme suit :

- Le contrôleur SDN réalise une gestion centralisée de l'ensemble du réseau. Il est adapté aux réseaux de petite taille ;
- Il utilise un mécanisme de rotation des nœuds qui permet d'économiser l'énergie des nœuds ;
- Il utilise la table de flux pour réduire la charge du réseau afin de réaliser un équilibrage de la charge du réseau autant que possible.

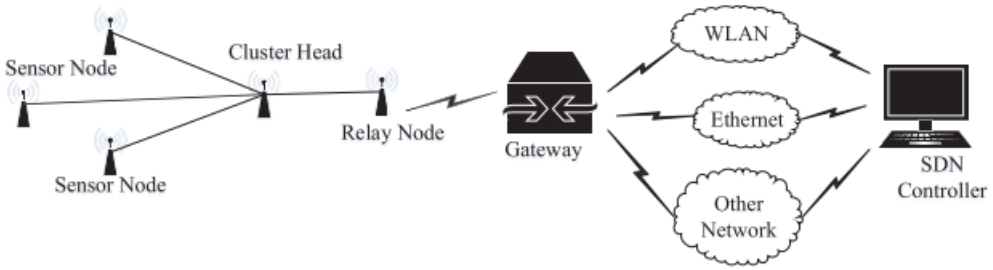

FIGURE 1.9 – Architecture centralisée d'un SDWSN [4]

1.5.3.2 SDWSN distribué

Dans un modèle SDWSN distribué [4], les nœuds de l'ensemble du réseau sont divisés en plusieurs clusters. Le contrôleur principale SDN, déployée au niveau de la passerelle ou passe via la passerelle, pour gérer et coordonner les sous-contrôleurs.

Les têtes de cluster sont des sous-contrôleurs SDN qui gèrent les nœuds capteurs dans chaque cluster (voir Figure 1.10). Le SDWSN distribué est adapté aux WSN à grande échelle en raison de la transmission de données importantes et de la charge importante du réseau. Les avantages du SDWSN distribué sont les suivants :

- La topologie du réseau est facilement extensible. Les nœuds sont ajoutés ou supprimés par région pour rendre plus pratique la gestion par la tête de cluster ;
- La consommation d'énergie est réduite. Dans le SDWSN centralisé, la consommation d'énergie des nœuds proches du puits est trop rapide. Au contraire, le SDWSN distribué a adopté une méthode coordonnée pour gérer le contrôleur principal et le contrôleur secondaire, de sorte que les nœuds normaux réduisent considérablement leur consommation d'énergie en profitant de la communication à un seul saut avec le nœud de cluster.
- La charge est plus équilibrée. Le (sous-)contrôleur régional gère les nœuds de sa région. La transmission des données de l'ensemble du réseau repose sur la tête de cluster qui transmet les données au puits par l'intermédiaire du contrôleur principal.

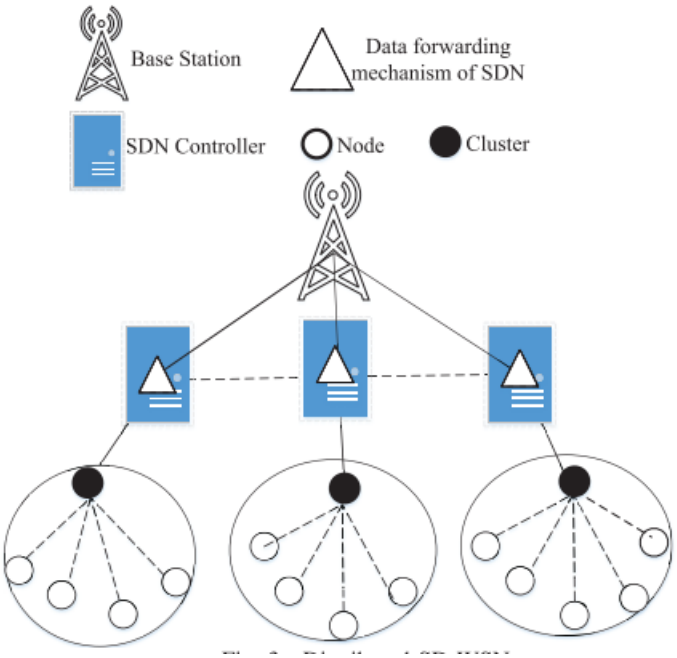

FIGURE 1.10 – Architecture distribuée d'un SDWSN [4]

1.5.4 Apports du SDN dans les WSN

Le SDN transforme le problème de l'administration du réseau en un problème de programmation, ce qui augmente la flexibilité et les autres avantages potentiels de la technologie SDN dans les WSN sont :

- Gestion de configuration : la gestion des réseaux de capteurs sans fil est très compliquée et fastidieuse. Les défis de la gestion des WSN sont pour la plupart inhérents aux réseaux d'infrastructure traditionnels, qui comprennent entre autres l'approvisionnement, la configuration et la maintenance [43]. L'approche SDN simplifie considérablement la gestion du réseau grâce à sa simplicité et à sa capacité d'évolution. [6] [43] Dans les WSN, la reconfiguration et la maintenance des nœuds capteurs tendent à être un processus compliqué et fastidieux si la gestion n'est pas flexible.
  Ce problème pourrait être exacerbé si l'environnement de déploiement du WSN n'est pas trop accessible. Ces difficultés sont atténuées en supprimant la logique de contrôle des nœuds capteurs, qui deviennent de simples éléments d'acheminement. Ces éléments d'acheminement seraient désormais contrôlés et manipulés à partir du contrôleur centralisé, ce qui permettrait la programmabilité des nœuds d'infrastructure physique [43]. Grâce à la flexibilité apportée par la technologie SDN, de nouveaux protocoles de routage peuvent être déployés facilement sans reconfigurer les nœuds et cela réduit également le besoin de compiler différentes versions pour les mêmes applications réseau pour différents nœuds capteurs. Ainsi, si une nouvelle méthode de gestion et de contrôle devient disponible, elle peut facilement être déployée, ce qui se traduit par une efficacité de fonctionnement. La technologie SDN permet également une configuration de mappage dynamique entre les nœuds capteurs et les contrôleurs si plus d'un contrôleur est utilisé comme illustré avec TinySDN [48].
- Au niveau de la gestion de l'énergie : les nœuds WSN sont soumis à des contraintes énergétiques, ce qui nécessite l'utilisation de protocoles économes en énergie dans les réseaux de capteurs. Le SDN peut fournir un moyen efficace de gérer la consommation d'énergie du WSN. Cela est possible, car un plan de contrôle logiquement centralisé maintient une vue complète de l'ensemble du WSN et peut donc réduire l'énergie consommée en appliquant les protocoles adéquats. Les fonctions du plan de contrôle peuvent gérer tous les protocoles de routage, ce qui évite aux nœuds de suivre des protocoles spécifiques à l'application qui pourraient consommer de l'énergie en fonction du trafic. Costanzo et al. [48] affirment que le SDN dans les WSN devrait prendre en charge des mesures communes de conservation de l'énergie, telles qu'elles sont actuellement étudiées dans les WSN traditionnels, comme le cycle de fonctionnement, l'agrégation des données dans le réseau et l'optimisation des couches croisées. Le paradigme SDN est pratique, car en vertu du découplage, les nœuds d'acheminement sont libérés d'une grande partie des fonctionnalités de calcul à forte intensité énergétique [49].
  Un algorithme d'ordonnancement du sommeil économe en énergie basé sur la

technologie SDN est proposé dans [50]. La plupart de ces fonctions consommatrices d'énergie résideront désormais dans le contrôleur, qui dispose de suffisamment de ressources énergétiques. Cela permet d'économiser une quantité considérable d'énergie et pourrait potentiellement prolonger la durée de vie du réseau. D'un point de vue heuristique, c'est le cas, mais le degré de cette affirmation doit encore être quantifié et reste ouvert aux recherches futures.

- Au niveau de la gestion de la sécurité : SDWSN souffre des mêmes problèmes de sécurité que les WSN traditionnels [44]. Cependant les solutions de sécurité implémentées de façon distribuée dans les WSN pourraient encore jouer un rôle vital en étant implémentées sur le plan de contrôle ou d'application. La centralisation de la gestion de la sécurité simplifie la mise en œuvre et la configuration des mécanismes de sécurité [51]. Cette perspective globale permet également une surveillance et une évaluation proactives, ce qui conduit à une contre-réaction rapide en cas d'attaque.
- Interopérabilité : les WSN sont connus pour leurs applications spécifiques. De ce fait, un nœud ne peut pas remplir plusieurs fonctions à la fois. Cela entraîne une sous-utilisation des ressources. Cette lacune peut être résolue en utilisant l'approche SDN. Le SDN atténue la dépendance vis-à-vis des fournisseurs en permettant aux éléments d'infrastructure d'être contrôlés à partir d'un point central, ce qui permet d'exécuter un seul protocole sur les éléments, même s'ils proviennent de différents fabricants [3].

1.5.5 Quelques architectures spécifiques de SDWSN

Plusieurs architectures SDWSN ont vu le jour. Chacune de ces architectures présente des avantages selon leur architecture et topologie de déploiement. Dans cette section, nous décrivons plus en détail quelques-unes de ces architectures. Le Tableau 1.1 présente une synthèse de quelques architectures spécifiques SDWSN.

1.5.5.1 Architecture SDWSN et Sensor OpenFlow

Dans cet article [5], les auteurs ont proposé un WSN défini par logiciel (SDWSN), une architecture présentant une séparation claire entre un plan de données et un plan de contrôle et Sensor OpenFlow (SOF), le composant central du SDWSN comme protocole de communication standard entre les deux plans. Le plan de données est constitué de capteurs qui transmettent les paquets en fonction du flux, tandis que le plan de contrôle est constitué d'un ou de plusieurs contrôleurs qui centralisent toute l'intelligence du réseau et assure le contrôle du réseau, comme le routage et le contrôle de la qualité de service. Cette architecture est présentée à la Figure 1.11. L'idée générale est de rendre le réseau sous-jacent (c'est-à-dire le plan de données) programmable en manipulant une table de flux personnalisable par l'utilisateur sur chaque capteur à travers SOF. Leur proposition s'appuie sur le pa-

radigme des réseaux définis par logiciel (SDN) Open-Flow, proposé pour les réseaux d'entreprise et de transporteur.

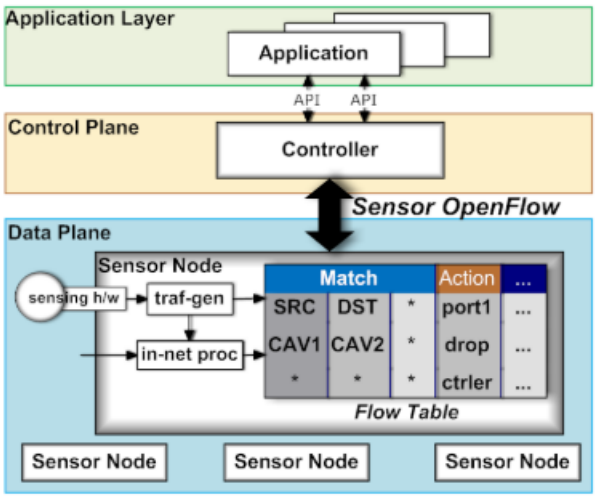

FIGURE 1.11 – Architecture SDWSN Sensor OpenFlow [5]

1.5.5.2 Architecture de SDWN

L'architecture du protocole SDWN est proposée dans [52] un réseau de capteurs basé sur l'utilisation de nœuds de communication IEEE 802.15.4. En plus de l'émetteur-récepteur, une unité de micro contrôle est déployée dans ces nœuds aux capacités de calcul et d'énergie limitées. Le réseau comprend également un puits qui est aussi le nœud où le contrôleur du réseau est exécuté. L'émetteur-récepteur IEEE 802.15.4 du puits est connecté à un système embarqué fonctionnant avec un système d'exploitation Linux. Les capacités de calcul et de communication de ce système embarqué sont considérablement élevées par rapport aux autres nœuds du réseau. La fonctionnalité de contrôleur de réseau sera exécutée par ce système embarqué. La Figure 1.12 présente l'architecture de protocole proposée pour les nœuds SDWN. Plus précisément, dans la Figure 1.12 à gauche, se trouve l'architecture pour le nœud générique, tandis que dans la Figure 1.12 à droite, se trouve l'architecture pour le puits.

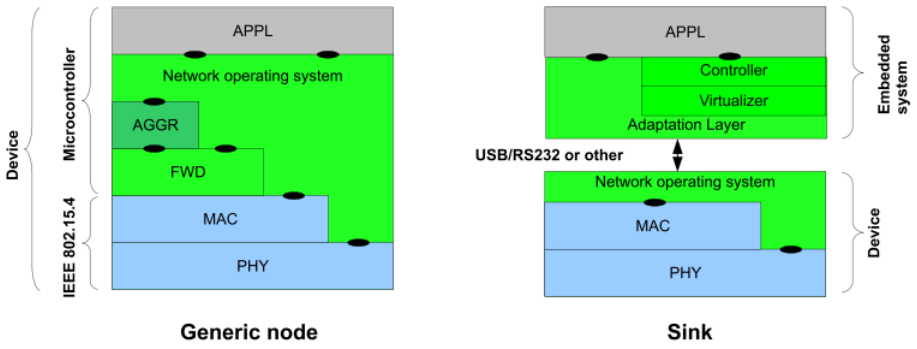

FIGURE 1.12 – Architecture de SDWN [6]

1.5.5.3 Architecture de TinySDN

TinySDN est un cadre SDN basé sur TinyOS [7]. Le TinySDN introduit des contrôleurs multiples dans les WSN et a deux composants principaux qui sont le nœud capteur SDN et le nœud contrôleur SDN (voir Figure 1.13). La conception du TinySDN se concentre sur les questions clés de l'approvisionnement en énergie, de la latence de communication et des trames de couche de liaison plus petites. La plupart de ces questions n'ont pas été abordées par les architectures précédemment proposées pour le SDN dans les WSN. Il s'agit également de la première conception basée sur le SDN pour des appareils fonctionnant sous TinyOS. Les dispositifs WSN typiques n'ont qu'un seul module radio qui transmet ou reçoit des signaux à un moment donné. Par conséquent, les plans de données et de contrôle doivent partager le même lien de communication et la bande passante disponible. Ce contrôle en bande provoque des retards dans le réseau. Le TinySDN propose un nouveau modèle dans lequel plusieurs contrôleurs sont utilisés dans les WSN et l'un d'entre eux est placé plus près des nœuds d'extrémité.

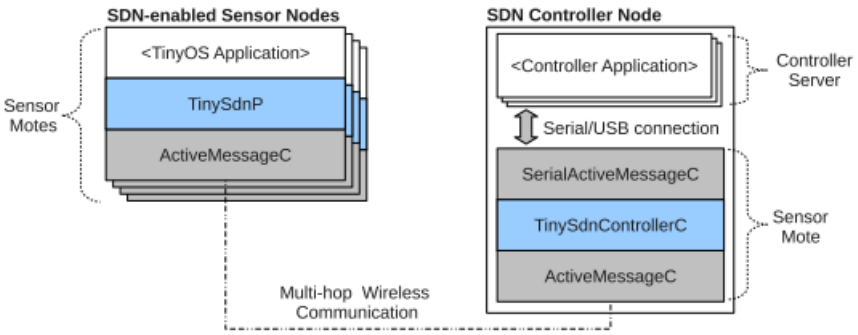

FIGURE 1.13 – Architecture TinySDN [7]

Les dispositifs d'extrémité sont considérés comme périphériques au SDN, ils sont hors de portée d'OpenFlow. Mais d'un autre côté, les nœuds capteurs dans les WSN se comportent comme des équipements finaux en générant des paquets de données. Le TinySDN peut déployer un nœud compatible SDN qui joue à la fois le rôle d'un commutateur SDN et d'un dispositif final SDN. Chaque nœud SDN doit trouver un nœud contrôleur SDN à rejoindre et recevoir des spécifications de flux. Un nœud capteur activé par SDN est divisé en trois parties à savoir l'application TinyOS, TinySdnP, ActiveMessageC. L'application TinyOS est l'équivalent de l'appareil final. Elle génère des paquets de données et les place ensuite sur le réseau en utilisant l'interface de programmation fournie par le composant TinySDN. TinySdnP est le composant principal de TinySDN qui vérifie si le paquet reçu correspond à une entrée de flux dans la table de flux et effectue ensuite l'action correspondante. ActiveMessageC est le composant TinyOS qui gère et fournit une interface de programmation pour interagir avec le module radio du nœud capteur.

Les nœuds contrôleurs SDN qui sont responsables de la création des flux de réseau avec deux modules. Le premier module nœud capteur s'exécute sur le nœud capteur et communique avec le nœud capteur compatible SDN en utilisant ActiveMessageC. Il agit comme un intermédiaire entre le module serveur du contrôleur et le réseau. Il transmet les messages reçus au serveur du contrôleur et reçoit les messages du réseau pour le module du serveur du contrôleur. Le deuxième est le module du serveur du contrôleur. Il contient la logique du plan de contrôle et est responsable de l'hébergement des applications du contrôleur et de la gestion des informations sur le flux et la topologie du réseau.

1.5.5.4 Architecture de Jaccobson

La Figure 1.14 montre l'architecture SDN-WSN proposée par Jacobsson et al [8]. La couche infrastructure contient les nœuds capteurs où chaque nœud est équipé d'un contrôleur local et la couche contrôle contient un ou plusieurs contrôleurs centraux. Au-dessus de la couche de contrôle se trouve la couche d'application, qui peut contenir des applications automatisées telles que le routage et la gestion de la topologie du réseau. Chaque capteur est équipé d'un contrôleur local. Le rôle du contrôleur local est de recevoir et d'exécuter les commandes du contrôleur central telles que la configuration, la reconfiguration et les mises à jour nécessaires des applications.

Le contrôleur local exécute ses programmes en installant un nouveau code ou script, qui étend ou modifie la fonctionnalité d'une application donnée. Chaque application est mise en œuvre à travers une machine virtuelle (VM) intégrée au capteur et la modification de cette application nécessite une modification du script déjà en place. Le contrôleur central doit découvrir la topologie réelle du réseau, mais aussi connaître la qualité des liens. La méthode LinkQuality Estimation (LQE) est basée sur des informations de trace de paquets. Une trace de paquets contient des

informations détaillées sur le comportement de la perte de paquets sur un lien.

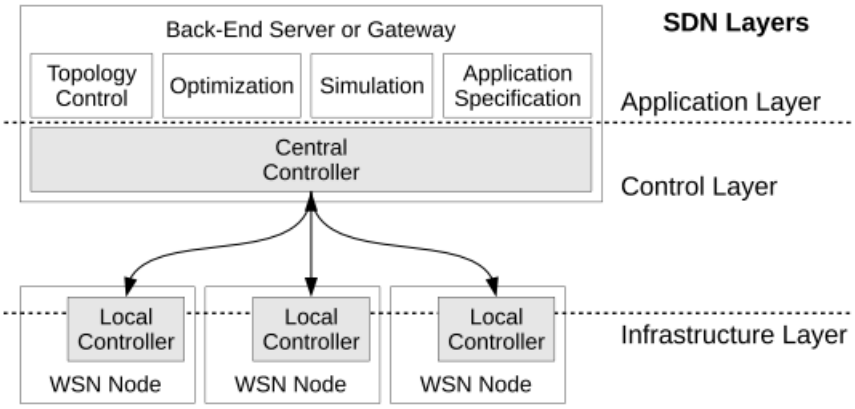

FIGURE 1.14 – Architecture de Jacobsson [8]

1.5.5.5 Architecture de SDN-Wise

Bien qu'il ait été démontré que les architectures mentionnées précédemment offrent un certain nombre d'avantages par rapport aux WSN traditionnels, il existe quelques lacunes [10] à savoir, les détails des protocoles ne sont pas toujours fournis, ce qui est fondamental pour le bon fonctionnement du réseau. Aussi, certaines des architectures ne sont pas mises en œuvre dans la pratique. Par conséquent, aucune évaluation des performances de ces solutions proposées n'a été effectuée. SDN-WISE [10] est la première mise en œuvre pratique d'une solution SDN de type OpenFlow conçue spécifiquement pour les WSN. Contrairement aux autres architectures, le SDN-WISE vise à limiter l'échange d'informations entre les nœuds et le contrôleur et à rendre les nœuds capteurs directement programmables.

En outre, le SDN-WISE offre la facilité d'implémentation de la logique du contrôleur SDN. Cela représente un avantage majeur par rapport aux solutions proposées précédemment, car cela augmente la flexibilité et la simplicité de la programmation du réseau. Les SDN-WISE offrent également la possibilité d'exécuter leur contrôleur dans un environnement simulé. Des logiciels de simulation tels que OMNET++ et COOJA peuvent être utilisés pour tester sa fonctionnalité. Le SDN-WISE s'efforce d'être efficace dans l'utilisation des ressources des nœuds capteurs, même si cette productivité peut entraîner une baisse du débit de données. Pour être économe en énergie, il encourage l'utilisation du cycle de service pour allumer et éteindre périodiquement le module radio, ce qui permet de conserver l'énergie. De plus, les WSN étant par nature axés sur l'information, le SDN-WISE rend le système plus conscient du contenu des paquets.

1.5.5.6 Architecture de IT-SDN

L'architecture IT-SDN [9] contient trois composants, comme le montre la figure 1, à savoir un protocole SB (southbound), un protocole ND (neighbor discovery) et un protocole CD (controller discovery). L'objectif du protocole SB est de définir la procédure de communication entre le contrôleur et les dispositifs SDN, y compris la définition du format des paquets, la manière de traiter ces paquets et le déroulement des opérations. Un protocole SB personnalisé est conçu et basé sur TinySDN [48] dans le sens où le concept de "flow id" est utilisé, c'est-à-dire que les paquets sont acheminés selon une étiquette prédéfinie. Certains paquets de contrôle sont acheminés par adresse et aussi un protocole sous-jacent pour effectuer la découverte du voisinage ND et celui du contrôleur CD. Cependant, l'architecture n'exige pas un protocole prédéfini et ne l'utilise pas pour transmettre les paquets de contrôle.

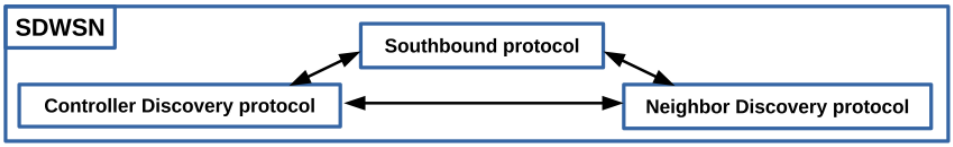

FIGURE 1.15 – Architecture IT-SDN [9]

Un protocole ND obtient et maintient les informations de voisinage des nœuds, tandis qu'un protocole CD identifie un candidat au prochain saut pour atteindre le contrôleur. Dans le cas de réseaux statiques, le premier protocole peut être exécuté uniquement au moment du démarrage du réseau ; à l'inverse, il doit être exécuté en continu dans les réseaux mobiles et dont les liens changent. Le second est utilisé principalement au démarrage du réseau, avant que le nœud ne reçoive les informations de routage du contrôleur. Les protocoles ND et CD sont conçus comme des entités séparées en raison de l'existence de nombreuses alternatives appropriées à ces tâches.

Grâce à cette approche, un chercheur pourrait facilement modifier et comparer les performances de différents algorithmes ou pourrait concevoir et mettre en œuvre un nouvel algorithme et l'intégrer sur l'architecture IT-SDN. De même, un développeur peut choisir la meilleure alternative en fonction du scénario d'application. Pour offrir cette flexibilité, un protocole ND doit se conformer à une interface prédéfinie, qui contient deux fonctions (procédure d'initialisation et réception de paquets) et génère des événements lors des changements de voisinage. De même, il existe une interface pour les protocoles CD qui fournit une fonction permettant d'obtenir le meilleur candidat voisin actuel pour atteindre le contrôleur, l'initialisation, la réception de paquets et la génération d'événements (en cas de changement de candidat).

Tableau 1.1 – Recapitulation des architectures SDWSN

Architecture	Scalabilité	Implémentation et environnement	Apports spécifiques	Contrôleur
Sensor OpenFlow [5]	NON	SIMULATION NS3	Protocole de support pour SDN (API southbound)	Centralisé
SDWN [6]	NON	PROPOSITION	Agrégation Routage Duty cycle	Centralisé
TinySDN [48]	OUI	PROPOSITION	Cadre multi contrôleurs Contrôle de trafic IN-band	Distribué
SDN-WISE [10]	OUI	SIMULATION COOJA	Reduction de la surcharge Agrégation Facilité de programmation	Distribué
Architecture de Jacobsson [8]	NON	PROPOSITION	Gestion de la mobilité et de la localisation	Centralisé
IT-SDN [9]	NON	SIMULATION COOJA	Définition de procédure de communication Un mécanisme de routage	Distribué

1.6 Gestion d'énergie dans le SDWSN

Dans la gestion du WSN basée sur le SDN, un contrôleur logiquement centralisé maintient une vue globale de tout le réseau impliquant une grande production de messages de contrôle. Cela entraîne une consommation d'énergie supplémentaire. Pour venir à bout de ce problème de consommation d'énergie, de nombreuses solutions ont été proposées. Différents mécanismes ont été implémentés au niveau du plan de contrôle pour réduire la consommation d'énergie dans les SDWSN parmi lesquelles nous pouvons citer : des techniques de réduction de message de contrôle [53], de routage [54] [55], d'agrégation de données et l'élection des têtes de cluster en fonction de l'énergie [56] et bien d'autres. Dans cette section, nous étudions plus en détail ces différentes techniques. Le Tableau 1.2 classe ces techniques en fonction de la technique d'optimisation d'énergie utilisée.

1.6.1 Utilisation du clustering dans SDWSN

La formation de cluster dans le SDWSN est une technique de réduction des charges du réseau. Dans ce type de réseau, les nœuds se mettent ensemble de façon

logique pour former des groupuscules appelés cluster. Dans chaque cluster, un nœud est choisi comme chef de cluster pour diriger la communication : ce chef est appelé tête de cluster. La formation de cluster a pour but de réduire la surcharge du réseau. En effet, les autres membres du cluster envoient leurs données à la tête de cluster qui les expédie à son tour vers la destination finale. Les têtes pourraient aussi agréger les données avant de les envoyer. Cela réduit la consommation d'énergie et améliore la durée de vie du réseau. Plusieurs variantes de l'application cette approche existent.

Flauzac et al [56] ont proposé de faire des regroupements des nœuds du réseau et ont supposé que chaque tête du cluster est un contrôleur appelé le SDNCH. Dans chaque domaine SDN, le SDNCH est chargé de gérer le fonctionnement des nœuds capteurs. Avec cette approche de regroupement, les informations recueillies sur l'environnement sont traitées sur le domaine par des nœuds et seront acheminées au SDNCH. De plus, le contrôleur a plus de ressources (énergie, puissance de calcul CPU et taille de la mémoire) que les autres nœuds dans le cluster. En utilisant l'interaction entre les SDNCH, il aura un accès complet aux commutateurs et mêmes aux règles de flux.

Les limites de la topologie SDNCH pour les réseaux de capteurs sans fil incluent la complexité de la mise en place et de l'administration, la consommation d'énergie élevée des commutateurs SDN, l'augmentation de la latence dans certaines applications, la fiabilité réduite en cas de défaillance du contrôleur centralisé et le coût élevé de mise en place. Cependant, ces limites ne sont pas insurmontables et peuvent être atténuées par l'utilisation d'approches intelligentes de routage, de mécanismes de tolérance de panne et bien d'autres techniques.

Dans [57], les auteurs ont étudié la possibilité d'introduire les concepts de réseaux définis par logiciel dans les réseaux de capteurs sans fil. Ils ont fait valoir que l'introduction de SDN dans les WSN peut aider à résoudre certains des problèmes difficiles des WSN, tels que les économies d'énergie et la gestion des réseaux. Ils ont également proposé une architecture SDN-WSN où le contrôleur à partir des informations sur l'état des nœuds capteurs structure le réseau en des clusters. Dans l'architecture proposée, le contrôleur est intégré à la station de base pour générer la table de routage pour les têtes de cluster. Dans leurs travaux, les nœuds capteurs simples font le filtrage de données pendant la collecte selon les règles définies par le contrôleur. Les données envoyées associent les informations d'état des nœuds.

Tan et al dans [55], ont proposé une architecture de réseau hiérarchique définie par logiciel pour les réseaux de capteurs sans fil. Cette architecture rend possible la complexe gestion du réseau et rend le système plus adaptable. Ils proposent en outre, un protocole de routage basé sur la QoS appelé QSDN-WISE. C'est à la fois un algorithme de regroupement, de routage et de maintenance du réseau local. Pour évite le phénomène de trou d'énergie et réduit la charge de travail d'une seule tête

de cluster, il lui on adjoint un algorithme de mise en cluster inégale basé sur une double tête de cluster, appelé DCHUC.

Une architecture SDN intégrée au WSN est proposée [49], dans laquelle le nœud réception du WSN est remplacé par le contrôleur et la tête de cluster par un commutateur de flux. Ce faisant, une séparation entre le plan d'acheminement et le plan de contrôle est effectuée. Ainsi, les nœuds capteurs, y compris les nœuds de tête, acheminent les paquets reçus avec un seul processus de consultation. Aussi, il ne participent pas à la décision d'acheminement conduisant autrement à des économies d'énergie.

Le commutateur openflow, qui fait office de tête de cluster a une énergie plus élevée que les nœuds capteurs. Le reste des nœuds détectent les données et les envoient à la tête de cluster. Le contrôleur est responsable de l'identification du voisinage et de la découverte de la topologie. Cette architecture proposée permet une économie d'énergie globale du réseau.

Les auteurs [58] ont proposé un nouveau modèle de décision d'acheminement sur les réseaux à grande échelle appelés SD-MHC-RPL, qui satisfait à la qualité de service requise, en termes de latence, de nombre de sauts et de consommation d'énergie, pour le réseau IoT. Pour faire face à un grand nombre d'appareils, leur approche consiste à organiser les nœuds en un scénario de cluster à saut multiple. Chaque grappe est gérée par un nœud de tête de cluster mettant en œuvre un protocole à faible consommation d'énergie.

Haosen et al [59] proposent un nouvel algorithme de clustering appelé EBCA basé sur SDWSN. Le contrôleur du SDWSN qui est également appelé nœud récepteur, redéfinit le rayon de communication de chaque nœud capteur en fonction de la distance par rapport au nœud récepteur et du nombre de nœuds voisins et propose une nouvelle méthode de regroupement basée sur l'énergie résiduelle du nœud lui-même et de ses voisins. Les nœuds capteurs choisissent les têtes de cluster en fonction de la distance par rapport à la tête de cluster et de l'énergie résiduelle de la tête de cluster.

Un nouvel algorithme de clustering appelé EDSS-LEACH Extending Dynamic Subnetwork Scheme-Low Energy Adaptive Clustering Hierarchy (EDSS-LEACH) basé sur un réseau de capteurs sans fil défini par logiciel (SDWSN) et du protocole LEACH, est proposé par Shen et al dans [60]. Ce dernier prend en compte les voisins, l'énergie résiduelle et les trois facteurs de saut vers le nœud puits. Grâce à un modèle de logique floue, il calcule les scores de chaque nœud à chaque tour d'expansion dynamique du sous-réseau.

En fonction des scores, l'algorithme obtient finalement un ensemble de têtes de cluster bien réparties où chaque tête de cluster détermine son prochain saut au moyen de l'algorithme de Dijkstra. Les résultats de simulation démontrent que l'algorithme EDSS-LEACH peut prolonger la durée de vie du réseau et améliorer l'équilibre de

la consommation d'énergie.

1.6.2 Routage

Afin d'optimiser la consommation d'énergie dans le SDWSN, plusieurs auteurs ont proposé des protocoles de routage efficient en énergie.

FTDP est une solution de routage proposée par Abdolmaleki dans [54] et implémentée sur l'architecture SDN-WISE. Il utilise le système flou pour calculer le coût associé aux nœuds voisins afin de choisir le nœud qui a le meilleur coût comme prochain saut. Le FTDP en fonction des informations (la quantité d'énergie restant du nœud, le nombre de paquets dans la file d'attente et le nombre de voisins) exécute le système flou qui consiste en la fuzzification, le système d'inférence et la défuzzification pour calculer le coût associé aux nœuds voisins. Le nœud ayant le coût le plus élevé est choisi comme nœud intermédiaire le plus approprié. Des résultats de simulation prouvent que FTDP améliore la durée de vie du réseau et augmente le taux de perte de paquets comparativement à SDN-WISE.

Bello et al dans le [61] ont proposé FTS-SDN. C'est une implémentation d'un gestionnaire de routage au niveau du contrôleur SDN-WISE. Ce gestionnaire de routage est chargé de créer les réponses aux demandes provenant du réseau. Dans la couche Application, l'application FTS-SDN Routing Manager décide donc de la manière dont le nœud demandeur doit traiter les paquets non appariés. Elle envoie ainsi donc, une ou plusieurs règles de flux pour permettre à un nœud demandeur de faire correspondre le paquet entrant actuel et les paquets similaires futurs. Cette procédure réeduisant ainsi la consommation d'énergie des nœuds demandeurs.

Tan et al [55] ont proposé l'algorithme de routage centralisé QSDN-WISE. Il construit deux chemins d'acheminement hétérogènes pour les nœuds et répond aux exigences des différents niveaux de données. La maintenance du réseau local réduit le nombre de messages de contrôle dans le réseau. Les résultats de la simulation indiquent que le QSDN-WISE peut assurer la prise en charge de la qualité de service pour des données ayant des exigences différentes, équilibrer la consommation d'énergie du réseau et prolonger la durée de vie du réseau.

Pour améliorer la distribution du trafic, Schaerer et al [13] ont proposé le protocole de routage dynamique conscient du trafic DTARP qui tient compte de la centralité et des statistiques de trafic dynamique des nœuds lors du calcul du chemin. Avec ces informations supplémentaires, les nœuds les plus actifs et les plus centraux sont reconnus et sont moins éligibles pour les retransmissions. Par conséquent, les itinéraires passant par des nœuds moins actifs sont choisis même s'ils ont un nombre de sauts légèrement plus élevé.

Un nouvel algorithme de routage pour le contrôleur SDN qui prolonge la durée de vie du WSN est proposé par Tomovic et al dans [62]. Il est rendu possible, lorsque le contrôleur SDN connaît les positions et les capacités des SN et même en cas de

déployement d'un faible pourcentage de nœuds SDN. C'est un scénario hybride dans lequel les nœuds SDN coexistent avec les nœuds capteurs traditionnels.

Al-Hubaishi et al [63] ont proposé une nouvelle approche de décision de routage utilisant un algorithme de Dijkstra basé sur le flou. Cette approche prend en compte non seulement les distances entre les nœuds, mais aussi l'énergie restante des nœuds sur le chemin, pour prolonger la durée de vie du réseau. Les résultats indiquent que la structure SDN proposée avec l'algorithme flou de Dijkstra est plus performante que celle utilisant l'algorithme normal de Dijkstra et son homologue basé sur ZigBee, en termes de consommation d'énergie. Leur approche fournir un routage efficace tout en prolongeant la durée de vie du réseau.

Banerjee et al [64] ont proposé SD-EAR, le routage conscient de l'énergie défini par logiciel. SD-EAR divise la structure entièrement distribuée d'un réseau de capteurs en cluster où chaque contrôleur est responsable d'une zone. Les contrôleurs connaissent la topologie de la zone associée et gardent également la trace des nœuds périphériques. Les nœuds périphériques dans une zone sont des nœuds qui ont une certaine portion de cercle radio en dehors de la zone. SD-EAR vise notamment à proposer un protocole de routage économe en énergie ainsi qu'une stratégie de veille pour favoriser la préservation de l'énergie dans le réseau.

Les techniques de routage proposées se basent sur la densité ou la centralité d'un noeud ou encore sur l'algorithme de dijkstra pour choisir le meilleur prochain saut pour aller vers le sink. Très peu de solutions tiennent compte de l'énergie résiduelle. Nous pensons que la surveillance directe de l'énergie résiduelle des noeuds permettra de connaître les noeuds les plus sollicités qui vont se décharger rapidement de leur énergie résiduelle. De ce fait, nous pourrions décrire un mécanisme pour équilibrer la fréquente de sollicitation entre les noeuds pour le routage de données.

1.6.3 Techniques de réduction de messages de contrôle

Afin d'optimiser les performances du réseau un nouveau mécanisme pour le SDN-WISE appelé Trickle timer, a été développé par Hieu et al dans [53]. SDN-Wise Trickle Timer est une méthode de synchronisation et de mise à jour des paramètres de contrôle dans les réseaux de communication logicielle définis par logiciel (SDN). Il permet d'assurer la cohérence des informations de contrôle en envoyant périodiquement des messages de mise à jour entre les dispositifs du réseau. La méthode SDN-Wise Trickle Timer utilise une approche de communication intermittente où les dispositifs du réseau envoient des messages de mise à jour à intervalles réguliers ou à chaque fois qu'un événement important se produit. Cela permet de réduire la quantité de trafic de contrôle dans le réseau, ce qui améliore les performances et réduit la consommation d'énergie.

Cependant, il y a des limites à l'utilisation de la méthode SDN-Wise Trickle Timer. Tout d'abord, la méthode repose sur des horloges internes relativement précises

dans les dispositifs du réseau pour maintenir la synchronisation. Si ces horloges ne sont pas synchronisées avec précision, cela peut entraîner des incohérences dans les mises à jour de contrôle, ce qui peut affecter les performances du réseau.

Ndiaye et al [65] ont proposé une solution appelée FR-CMQ fonctionnant sur l'architecture IT-SDN. Cette solution permet de réduire le nombre de messages de type demande de règles de flux à destination du contrôleur. Elle accomplit ceci en envoyant que la demande du premier paquet. Pour les autres paquets venant de la même source et partant vers la même destination seront mis en attente au niveau du nœud jusqu'à ce qu'il reçoive une instruction à suivre pour les traiter. Cela permet d'éviter les doublons et permet ainsi au nœud d'économiser de l'énergie.

Cependant FR-CMQ ne gère pas la congestion due à la mise en attente de paquets de données dans la mémoire tampon du noeud.

Afin de réduire le problème de surcharge du message de contrôle, Nagarathna et al [66] proposent le mécanisme de temporisation par le contrôleur SDN pour éviter le problème de surcharge et pour obtenir une réponse unique avec des informations appropriées sur ses nœuds adjacents situés dans la direction du nœud de destination lors de l'opération de transfert de paquets.

Buzura et al [67] ont mis en place un composant de contrôle bien informé efficacement pouvant gérer un WSN et déterminer si les nœuds capteurs doivent transmettre des données ou non. En outre, le contrôleur peut également décider de transmettre les données de manière proactive pour chaque capteur. Cela réduit le volume du trafic de données dans la couche WSN.

Dans cette proposition, pour que le contrôleur puisse réaliser les fonctions proposées, il met en œuvre deux composants. Tout d'abord, il a besoin d'une composante d'apprentissage de base qui apprend le comportement de chaque capteur, c'est-à-dire la fréquence de transmission des données pour chaque capteur afin de déterminer un modèle. Deuxièmement, il met en œuvre un mécanisme de diffusion basé sur les modèles d'intervalle de temps établis auparavant. Plusieurs simulations ont été effectuées sur des conditions météorologiques historiques et les résultats montrent une diminution significative des trafics du réseau réduisant la consommation d'énergie et augmentent la durée de vie du réseau.

1.6.4 Autres mécanismes d'optimisation d'énergie

Afin de pouvoir gérer l'allocation des ressources radio par des contrôleurs centraux dotés de puissance de traitement et de capacités de stockage et de calcul importantes, un algorithme d'allocation de ressources économe en énergie dans les SDWSN appelé CABPA à été proposé par Zhang et al [68]. Dans cet algorithme, ils formulent un problème d'optimisation pour minimiser la consommation d'énergie, sous la contrainte individuelle de qualité de service. Ensuite, le problème d'optimisation initial est transformé à l'aide d'une relaxation semi-définie, afin d'obtenir une

allocation adaptative centralisée de la bande passante et de la puissance.

pour apporter une solution au problème au problème spécifique de la consommation d'énergie dans les SDWSN, Zhang et al [69] ont proposé un algorithme centralisé basé sur le SDP (une programmation semi-définie). Le problème original est relaxé en un SDP, il sert de borne inférieure, qui permet d'obtenir facilement une allocation optimale globale de la puissance et de la bande passante. En outre, l'étanchéité de la limite inférieure est analysée par deux cas particuliers proposés. Aussi, une approche distribuée alternative est développée pour fournir une référence de performance de l'approche centralisée. Les résultats de la simulation révèlent que l'algorithme centralisé proposé est plus performant en ce qui concerne la consommation d'énergie et l'utilisation de la bande passante.

Afin de réaliser des économies d'énergie en réglant la puissance des nœuds d'envoi en fonction de la distance entre les nœuds voisins, Pradeepa et Pushpalatha [70] ont proposé le contrôleur SDNSPIN. Avec cette proposition, les informations sur la topologie du réseau de capteurs compatibles SDN sont transmises en deux étapes : le nœud contrôleur SDN, à travers un protocole SPIN, génère la table des voisins et parmi ces voisins, le meilleur est sélectionné en fonction de son niveau d'énergie et de la qualité de la liaison. Afin de distinguer le voisinage, le nœud contrôleur SDN diffuse des paquets ADV et attend les messages de demande des nœuds. A la réception d'une demande, chaque émetteur de demande est inclus dans la table de voisinage ajoutée avec son RSSI (Received Signal Strengh Indication) et le niveau d'énergie. Sur la base de sa table de voisinage, le meilleur voisin est suggéré au nœud du capteur pour transmettre l'information.

Dans le soucis d'apporter une architecture SDWSN économe en énergie, Les travaux de [71] proposent de mettre en œuvre trois catégories de nœuds WSN différentes : les nœuds gérants, les nœuds centraux et les nœuds normaux. Les nœuds gérants sont le gestionnaire du réseau en tant que contrôleur SDN. Les nœuds centraux agissent en tant que transpondeurs pour transmettre les données tandis que les nœuds normaux sont uniquement utilisés pour recevoir le flux de données. Enfin, un système d'exploitation de réseau, qui est responsable des informations de base de l'ensemble du réseau telles que la gestion de la topologie, est également indiqué.

Les auteurs [72] ont implémenté un algorithme de surveillance du trafic appelé SDN-TAP. SDN-TAP notifie la situation de congestion en envoyant un message d'alarme au contrôleur afin de recréer les règles de flux pour le nœud congestionné, le(s) nœud(s) source(s) et les nouveaux noeuds de transfert. Les évaluations ont révélé que le SDN-TAP surpasse les protocoles de routage classiques en réduisant le taux de perte de paquets et la consommation d'énergie.

Tableau 1.2 – Mécanismes d'optimisation d'énergie dans les SDWSN

Solutions	clustering	Aggregation	Réduction de message de contrôle	Duty Cycle	Routage
EDSS-LEACH [60]	OUI	OUI	OUI	NON	OUI
SDN-TAP [72]	NON	OUI	NON	OUI	OUI
FR-CMQ [65]	NON	NON	OUI	NON	OUI
DTARP [13]	NON	OUI	NON	OUI	OUI
QSDN-WISE [55]	OUI	OUI	OUI	OUI	OUI
FTDP [54]	NON	OUI	NON	OUI	OUI
Integrating Trickle Timing [53]	NON	OUI	OUI	OUI	NON
SD-EAR [64]	OUI	NON	OUI	OUI	OUI

1.7 Discussions et hypothèses

Les techniques de routage basées sur SDN-WISE faisant la répartition de charge [13] prennent pour paramètre de basculement la centralité des nœuds ou les statistiques des trafics effectués par le nœud. Cependant, la consommation d'énergie dans le réseau de capteurs n'est pas uniquement liée à ces facteurs. Ainsi, les techniques qui se basent uniquement sur ces derniers pour décider de l'instant de basculement vers un autre nœud pourraient ne pas être optimales en terme énergétique. C'est pourquoi, dans nos travaux, nous nous orienterons vers la surveillance directe de l'énergie résiduelle des nœuds pour la prise de décision afin de basculer vers un

autre nœud lorsque les nœuds les plus occupés se déchargent.

Les techniques utilisant la formation de cluster comportent plusieurs avantages tels que l'agrégation de données [60] et la réduction de messages de contrôle [55] dans le cluster. Cela favorise la réduction de la consommation d'énergie des nœuds. Cependant, tous ces traitements sont à la charge des têtes de cluster qui disposent des mêmes caractéristiques que les nœuds ordinaires, leurs niveaux d'énergie diminuent plus rapidement réduisant par la même occasion la durée de vie du réseau. Dans les clusters où il y a peu de nœuds, nous remarquons que la consommation d'énergie est accrue, car la fréquence de transmission de données agrégées est élevée. C'est pourquoi, nous explorons la piste d'un nombre minimum fixé de nœuds pour chaque cluster. Ainsi, plus il y a d'éléments par cluster, mieux l'agrégation de données optimise la consommation d'énergie.

1.8 Conclusion

Dans ce chapitre, nous avons dans un premier temps dressé l'état de l'art concernant les différentes architectures et solutions innovantes visant à résoudre le problème de la consommation d'énergie dans les SDWSN. Dans un second temps, nous avons présenté de manière générale les réseaux WSN à cluster et quelques mécanismes que des auteurs ont proposés pour la gestion optimale de la consommation d'énergie. Dans un troisième temps, nous avons analysé et relevé les limites de quelques propositions. Puis, nous avons proposé des pistes de travail pour combler les insuffisances observées. Dans le chapitre à venir, nous présenterons nos solutions pour répondre à quelques problématiques.

Chapitre 2

BMN-LEACH-S : une solution d'équilibrage de nœuds entre les clusters et d'élection efficace d'une tête de cluster

2.1 Introduction

Pour faire face au problème d'équilibrage de charge de trafics dans le WSN et exploiter efficacement les ressources énergétiques des nœuds, nous présentons une solution basée sur le protocole LEACH [16] et LEACH-S [15] appelée BMN-LEACH-S. Dans ce chapitre, nous présentons d'abord le fonctionnement de LEACH et celui de LEACH-S. Ensuite, nous exposons les limites de chacun de ces protocoles. Aussi, nous présentons notre solution BMN-LEACH-S. Enfin, nous présentons les resultats d'une évaluation analytique des performances de cette solution comparée à LEACH-S.

2.2 Présentation et fonctionnement de LEACH

Dans cette section, nous faisons un aperçu du protocole LEACH. Ensuite, nous exposons les détails de l'algorithme LEACH. Enfin, nous présentons les limites de LEACH.

2.2.1 Aperçu du protocole

LEACH est un protocole de mise en cluster auto-organisé et adaptatif qui utilise la randomisation pour répartir la charge énergétique de manière égale entre les nœuds capteurs du réseau. Dans LEACH, les nœuds s'organisent en clusters, avec un nœud agissant comme station de base locale ou tête de cluster. Si les têtes de clusters sont choisies a priori et fixées pendant toute la durée de vie du système, comme dans les algorithmes de mise en cluster classiques, les nœuds capteurs malchanceux choisis

pour être des têtes de cluster épuiseraient rapidement leurs énergies, réduisant ainsi la durée de vie de tous les nœuds appartenant à ces clusters. Ainsi, LEACH inclut une rotation aléatoire de la position de la tête de cluster, de sorte que ce rôle tourne entre les différents nœuds capteurs afin de ne pas épuiser la batterie d'un seul nœud capteur.

En outre, LEACH effectue une fusion et une compression des données au sein de chaque cluster pour ensuite les envoyer à la station de base. Cela réduit encore la consommation d'énergie et améliore la durée de vie du système. Les nœuds capteurs s'élisent eux-mêmes comme têtes de clusters locales à tout moment avec une certaine probabilité. Ces nœuds têtes de cluster diffusent leur statut aux autres nœuds du réseau. Chaque nœud capteur détermine à quel cluster il va appartenir en choisissant la tête de cluster qui nécessite le minimum d'énergie de communication. Une fois que tous les nœuds sont organisés en cluster, chaque tête de cluster crée un programme pour les nœuds de son cluster. Cela permet aux composantes radio de chaque nœud non tête de cluster d'être éteintes à tout moment, sauf pendant la transmission, ce qui minimise l'énergie dissipée dans les nœuds capteurs. Une fois que le nœud tête de cluster a reçu toutes les données des nœuds de sa cluster, il les agrège et transmet ensuite les données compressées à la station de base. Cependant, il n'y a que quelques têtes de clusters, cela ne concerne qu'un petit nombre de nœuds.

Comme nous l'avons vu précédemment, le fait d'être une tête de cluster épuise la batterie du nœud. Afin de répartir cette consommation d'énergie sur plusieurs nœuds, les nœuds têtes de clusters ne sont pas fixes. Ce changement de statut est fait automatiquement à différents intervalles de temps. Ainsi, un ensemble C de nœuds peut s'élire tête de cluster à l'instant t1, mais à l'instant t + d, un nouvel ensemble C1 de nœuds s'élit comme tête de cluster. La décision de devenir une tête de cluster dépend de la quantité d'énergie restante du nœud. De cette façon, les nœuds ayant le plus d'énergie restante se chargeront des fonctions énergivores du réseau. Chaque nœud prend sa décision de devenir une tête de cluster indépendamment des autres nœuds du réseau et aucune négociation supplémentaire n'est donc nécessaire pour déterminer les têtes de groupe. Le système peut déterminer, a priori, le nombre optimal de clusters à constituer. Cela dépendra de plusieurs paramètres, tels que la topologie du réseau et les coûts relatifs à la communication.

2.2.2 Détails de l'algorithme LEACH

Le fonctionnement de LEACH est décomposé en cycles, où chaque tour commence par une phase de mise en place des clusters, suivie d'une phase d'état stable durant laquelle les transferts de données vers la station de base ont lieu. Afin de minimiser les charges du réseau, la phase d'état stable est longue par rapport à la phase de mise en place.

2.2.2.1 Phase d'annonce

Au départ de la création des clusters, chaque nœud décide de devenir une tête de cluster pour le tour en cours. Cette décision est basée sur le pourcentage suggéré d'être une tête de cluster pour le réseau (déterminé a priori) et le nombre de fois où le nœud a été une tête de cluster jusqu'à présent. Cette décision est prise par le nœud n qui choisit un nombre aléatoire entre 0 et 1. Si le nombre est inférieur à un seuil T, le nœud devient une tête de cluster pour le tour actuel. Le seuil est défini comme suit :

$$T = \frac{P}{1 - P * (rmod\frac{1}{P})} \tag{2.1}$$

P est le pourcentage souhaité de têtes de cluster par rapport au nombre total de nœuds du réseau (par ex, $P = 0,05$) ;

r est le tour actuel et G est l'ensemble des nœuds qui n'ont pas été des têtes de cluster dans les derniers rounds $1/P$.

En utilisant ce seuil, chaque nœud sera une tête de cluster à un moment donné dans $1/P$ tours. Pendant le tour 0 ($r = 0$), chaque nœud a une probabilité P de devenir une tête de cluster après $1/P - 1$ tours. $T = 1$ pour tous les nœuds qui n'ont pas encore été têtes de clusters, après $1/P$ tours, tous les nœuds sont à nouveau éligibles pour devenir des têtes de cluster. Chaque nœud qui s'est élu tête de cluster pour le tour actuel diffuse un message d'annonce au reste des nœuds. Pour cette phase de "publicité de tête de cluster", ces têtes de clusters utilisent un protocole CSMA MAC.

Toutes les têtes de clusters transmettent leur publicité en utilisant la même énergie de transmission. Les nœuds qui ne sont pas des têtes de cluster doivent garder leurs récepteurs allumés pendant cette phase de mise en place pour entendre les annonces de tous les nœuds têtes de cluster. Une fois cette phase terminée, chaque nœud non-tête de cluster décide du cluster auquel il appartient pour ce tour. Cette décision est basée sur la force du signal reçu de l'annonce. En supposant que les canaux de propagation soient symétriques, l'annonce de tête de cluster entendue avec la plus grande force de signal est la tête de cluster à laquelle la quantité minimale de données transmises est destinée. En cas d'égalité, une tête de cluster est choisie aléatoirement.

2.2.2.2 Phase de mise en place des clusters

Après que chaque nœud ait décidé à quel cluster il appartient, il doit informer le nœud tête de cluster qu'il sera membre du cluster. Chaque nœud retransmet cette information à la tête de cluster en utilisant un protocole CSMA MAC. Pendant cette phase, tous les nœuds têtes de clusters doivent garder leurs récepteurs allumés.

Création du programme

Le nœud tête de cluster reçoit tous les messages pour les nœuds qui souhaitent être inclus dans le cluster. En fonction du nombre de nœuds dans le cluster, le nœud tête de cluster crée un programme TDMA indiquant à chaque nœud quand il peut transmettre. Ce programme est diffusé aux nœuds de la cluster.

Transmission des données

Une fois que les clusters sont créés et que le programme TDMA(Time Division Multiple Access) est fixé, la transmission des données peut commencer. En supposant que les nœuds ont des données à envoyer, ils les envoient à la tête de cluster pendant le temps de transmission qui leur est alloué. Cette transmission utilise une quantité minimale d'énergie. En rappel, le cluster est choisi en fonction de la force du signal reçu en guise d'annonce de la tête de cluster. La radio de chaque nœud non-tête de cluster peut être éteinte jusqu'à ce que le temps de transmission alloué au nœud soit écoulé, ce qui minimise la dissipation d'énergie dans ces nœuds. Le nœud tête de cluster doit garder son récepteur allumé pour recevoir toutes les données des nœuds de la cluster. Lorsque toutes les données ont été reçues, le nœud tête de cluster exécute des fonctions de traitement du signal afin de compresser les données en un seul signal.

Après un certain temps, qui est déterminé à priori, le tour suivant commence, chaque nœud déterminant s'il doit être une tête de cluster pour ce tour et annonçant cette information. La Figure 2.1 décrit le fonctionnement de LEACH.

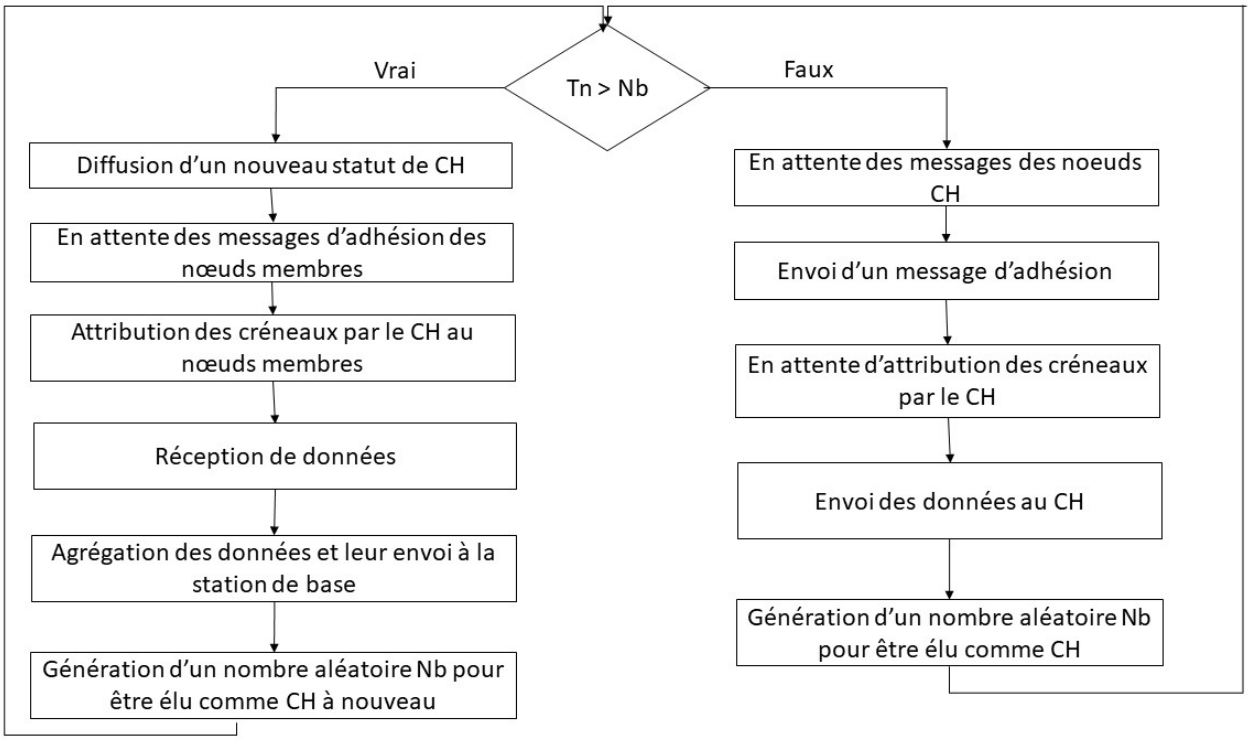

FIGURE 2.1 – Fonctionnement de LEACH

2.2.3 Limites de LEACH

Dans le protocole LEACH, la tête de cluster est sélectionnée à chaque tour pour éviter qu'un seul nœud ne supporte les charges routage. Cependant, la sélection périodique de la tête de cluster nécessite l'échange de beaucoup de messages de contrôle entre les nœuds. Ces derniers consomme beaucoup d'énergie par la génèration des surcharges dans le réseau. LEACH-S a été conçu pour atteindre les objectifs tels que minimiser l'énergie consommée par chaque nœud du réseau, prolonger la durée de vie du réseau et minimiser la surcharge générée par les messages de contrôle. Pour atteindre ces objectifs, LEACH-S réduit la sélection périodique des têtes de cluster afin de minimiser la production des messages de contrôle échangés entre les nœuds.

2.3 Protocole LEACH-S

Dans cette section, nous présentons d'une part le fonctionnement du protocole LEACH-S, une variante améliorée de LEACH. D'autre part, nous exposons les limites de ce protocole.

2.3.1 Présentation de l'algorithme

L'algorithme LEACH-S se déroule en tour comme le protocole LEACH avec deux phases : la phase d'installation et la phase de stabilité. La contribution de LEACH-S par rapport à LEACH se situe dans la phase d'installation, plus précisément à partir du deuxième tour. LEACH-S fonctionne comme suit :

- le tour initial commence par une phase d'installation suivie d'une phase de stabilité (les mêmes phases de LEACH).
- à partir du 2e tour, l'élection d'un nouveau CH est faite selon les critères suivants : si l'énergie résiduelle actuelle (E_r) du CH est supérieure à l'énergie moyenne (E_{moy}) de son cluster, alors le CH actuel reste un CH. E_{moy} est calculé à l'aide de la formule présentées dans l'équation (5.2) :

$$E_{moy} = \frac{\sum_{i=1}^{n} E_i}{n} \tag{2.2}$$

  E_i : l'énergie résiduelle du nœud i dans le cluster, n : nombre de nœuds dans le cluster.
- sinon, un nouveau CH est élu par le CH du tour précédent. En fonction de l'énergie résiduelle de chaque nœud dans le cluster, le nœud avec l'énergie maximale sera choisi.
- l'ancien CH informe le nouveau CH de son rôle dans le réseau et transmet la liste des identifiants de tous les nœuds membres.

Ces étapes sont répétées jusqu'à ce que tous les nœuds soient déchargés de leur énergie. La Figure 2.2 décrit l'organigramme de LEACH-S. La partie bleue de l'organigramme représente la principale contribution apportée par LEACH-S par rapport à LEACH :

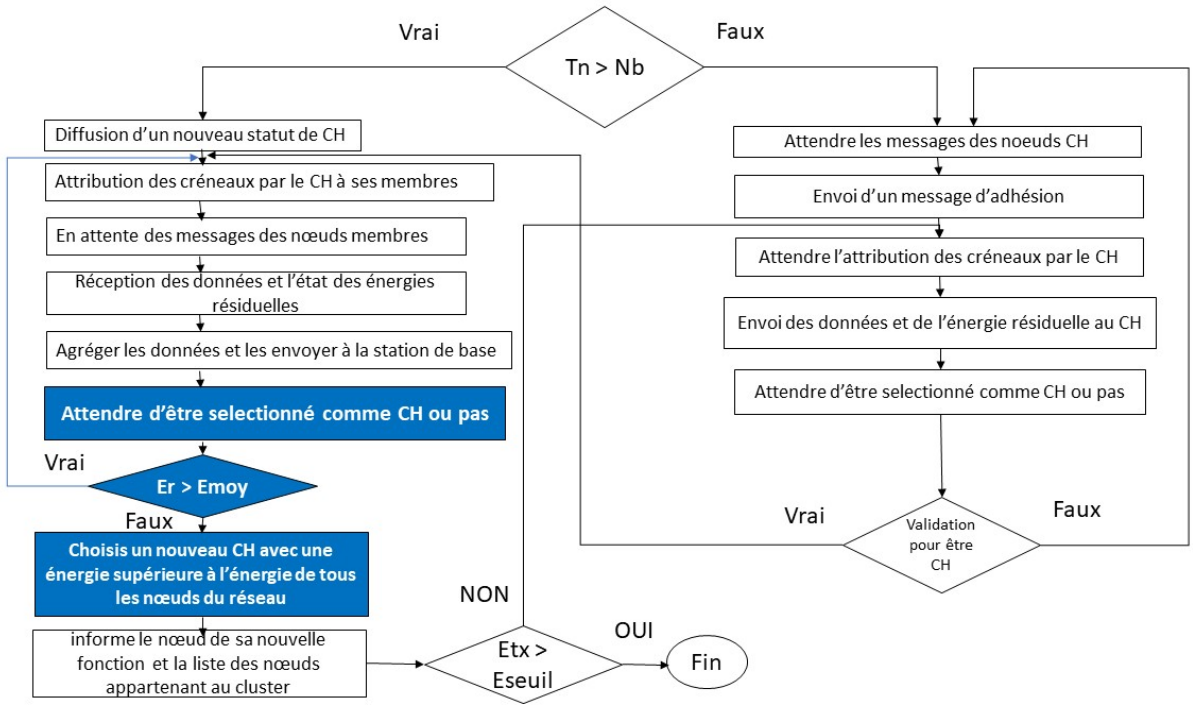

FIGURE 2.2 – Fonctionnement de LEACH-S

2.3.2 Limites de LEACH-S

LEACH-S est une amélioration de LEACH supprimant les cycles de reconstruction des clusters qui est source de consommation d'énergie dans le réseau. Cependant, dans LEACH-S pour élire une nouvelle tête de cluster, le CH sortant compare l'énergie résiduelle à la moyenne des énergies résiduelles du cluster afin de prendre une décision. Cela pourrait entraîner une instabilité dans le réseau parce que l'énergie résiduelle du CH ne tarde pas à se retrouver en dessous de la moyenne. Cela va entraîner un changement très fréquent des têtes de cluster occasionnant des instabilités. Aussi dans LEACH-S après le cycle initial, les clusters sont constitués. Un nœud membre demeure dans le même cluster indéfiniment. Par conséquent, les nœuds jouant le rôle de CH dans les clusters ayant plus de nœuds que la moyenne se déchargeront plus rapidement que les autres. Ce qui provoquera une instabilité dans le cluster due aux changements rapides de ces têtes de gros clusters.

2.4 Notre approche BMN-LEACH-S

Pour faire face au changement rapide de têtes de clusters et ses corollaires, nous proposons une nouvelle approche appelée *Balance Member's node LEACH-S* (BMN-LEACH-S). Cette solution d'une part propose une technique d'équilibrage

du nombre de nœuds membres des clusters en se basant sur un système à logique floue exploitant les métriques de base : nombre de nœuds déjà gérés par le CH et le RSSI du nœud cible avec le CH. D'autre part, elle propose une technique simple et efficace de changement de tête de CH en se basant sur la détermination d'un quantum d'énergie à épuiser par chaque CH avant que celui-ci passe la main à un autre nœud (nouvelle tête de cluster). Ces nouveaux mécanismes contribuent à équilibrer la charge des CH et ainsi à diminuer l'instabilité de certains clusters et de par là augmenter la durée de vie du réseau. Notre algorithme BMN-LEACH-S se déroule en cycles et chaque cycle est constitué de deux phases : la phase de configuration et la phase d'état stable. La phase de configuration du cycle initial est constituée du processus d'élection de la tête de cluster et du processus d'adhésion des nœuds à un cluster basé sur la prise en compte des métriques du système flou. Pour les autres cycles, le choix d'une nouvelle tête de cluster est effectué par le CH sortant.

2.4.1 Élection des têtes de cluster au cycle initial

Tout comme dans LEACH-S, dans BMN-LEACH-S, le cycle initial commence par une phase de configuration où chaque nœud capteur décide d'agir ou non en tant que tête de cluster pour ce tour particulier. Cette décision est prise par le nœud capteur en sélectionnant aléatoirement un nombre entre 0 et 1. Un nœud devient tête de cluster pour ce premier tour si le nombre tiré est inférieur à une valeur seuil prédéfinie. Il diffuse alors un message de contrôle annonçant son statut de CH. A la différence de LEACH-S, dans notre approche, les messages d'annonce du CH comportent le nombre de nœuds adhérant au cluster.

2.4.2 Processus d'adhésion des nœuds à un cluster

Dans cette section, nous décrivons dans un premier temps le principe du processus d'adhésion à un cluster. Dans un second temps, nous expliquons plus en détails le processus de calcul de la fonction du coût d'adhésion.

2.4.2.1 Principe du processus d'adhésion

Lorsqu'un nœud non CH reçoit un message d'annonce de la part d'un CH, il envoie son message de demande d'adhésion au CH. Dans le cas où ce nœud reçoit un message d'annonce de la part de plusieurs CH, il calcule le coût associé à chaque CH à l'aide du système flou. Le système flou prend en paramètre la qualité du lien (RSSI) et le nombre de nœuds membres déjà contenus dans le cluster du CH. Le processus de calcul du coût du CH par notre système à logique floue est présenté dans la section 2.4.2.2. Le nœud choisira le cluster dont le CH aura le meilleur coût. Après, il envoie un message d'adhésion au CH concerné. Ce message contient toutes ses informations d'identité. La partie colorée en vert de l'organigramme (voir

Figure 2.3) représente notre principale contribution par rapport à LEACH-S. Pour la comprehension l'organigramme, nous expliquons les variable suivants :

E : l'énergie du noeud tête de cluster au moment de son élection.

$E(t)$: l'énergie du noeud tête de cluster avec l'évolution du temps.

K : le quantum d'énergie à épuiser par une tête de cluster avant de céder sa place à un autre nœud.

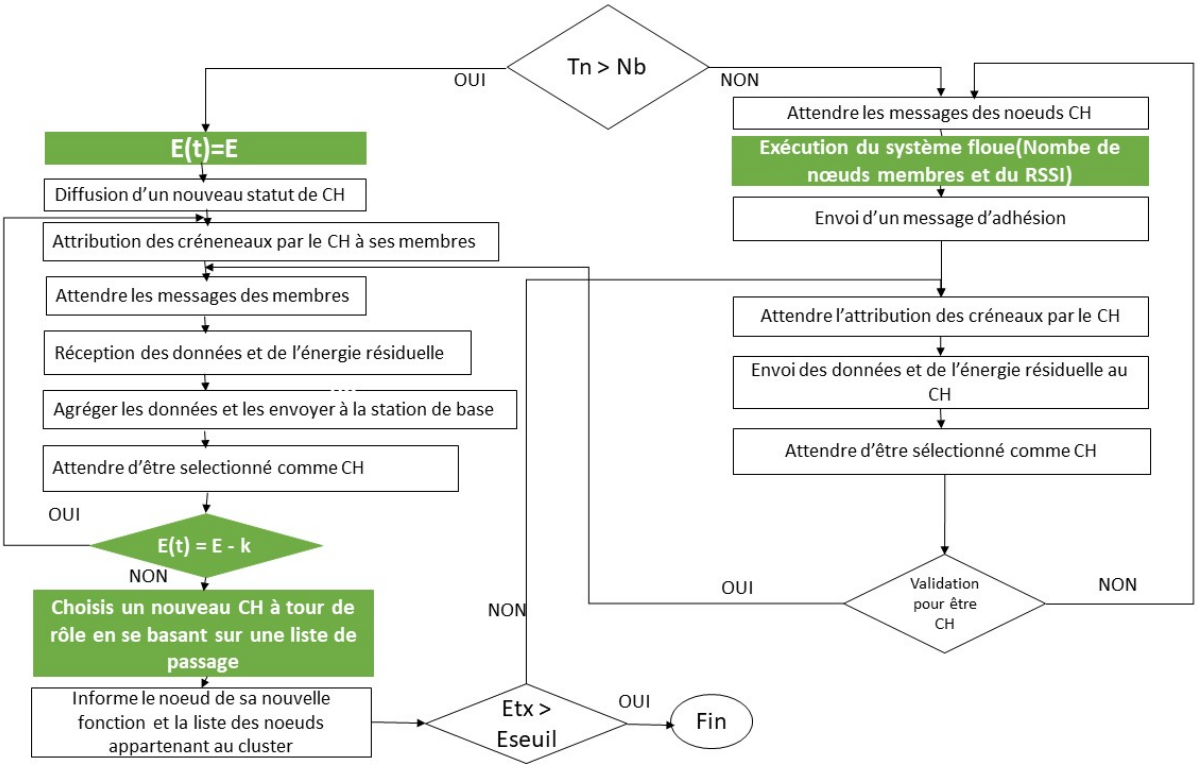

FIGURE 2.3 – Fonctionnement de BMN-LEACH-S

2.4.2.2 Fonction floue pour le calcul du coût d'adhésion

Notre solution d'équilibrage de nœuds utilisant une fonction floue fonctionne en trois étapes. La fuzzification des valeurs des paramètres (nombre de nœuds et RSSI), le système d'inférence et la défuzzification.

Phase de fuzzification

Dans cette phase, nous traduisons les paramètres d'entrées (la puissance du signal RSSI et le nombre de nœuds du cluster) en des valeurs discrètes entre 0 et 1. Le premier paramètre est le nombre de nœuds dans le cluster. Vu que notre objectif est d'équilibrer la taille des clusters, il est nécessaire de prendre en compte le nombre de nœuds (NN) que chaque cluster possède pour que le cluster ayant plus de nœuds ait moins de chance de recevoir un nouveau nœud. Aussi, le deuxième paramètre est

la puissance du signal ($RSSI$) entre le nœud et le CH. Nous avons pris en compte la puissance du signal pour que le cluster ayant une bonne liaison de communication ait plus de chance de recevoir le nœud.

Pour ce faire, nous utilisons une fonction d'appartenance pour traduire ces valeurs. Il faut noter qu'il y a plusieurs types de fonctions d'appartenance dont la fonction d'appartenance triangulaire et la fonction sinusoïdale. Mais, nous avons choisi la fonction d'appartenance triangulaire. Aussi, nous choisissons les mêmes variables linguistiques pour délimiter nos ensembles flous (très faible, faible, moyen, élevé, très élevé) pour les deux paramètres. Par exemple, si un nœud a le choix entre deux clusters nommés I et J. Le nombre de nœuds du cluster I est 8 et son $RSSI$ est 70 et le nombre de nœuds du $clusterJ$ est 3 et son $RSSI$ est 50. $ClusterI$ $(NN = 8; RSSI = 70)$ $ClusterJ(NN = 3; RSSI = 50)$.

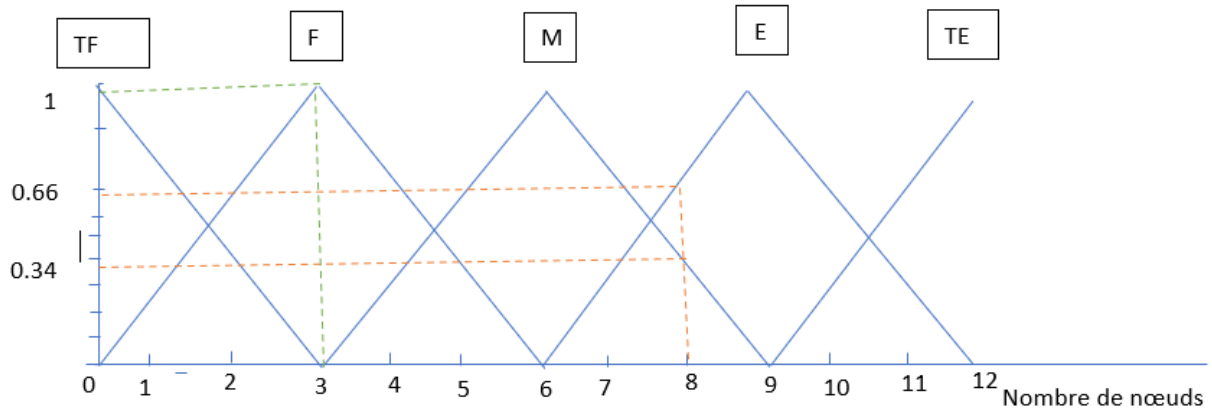

FIGURE 2.4 – Fuzzification du nombre de nœuds

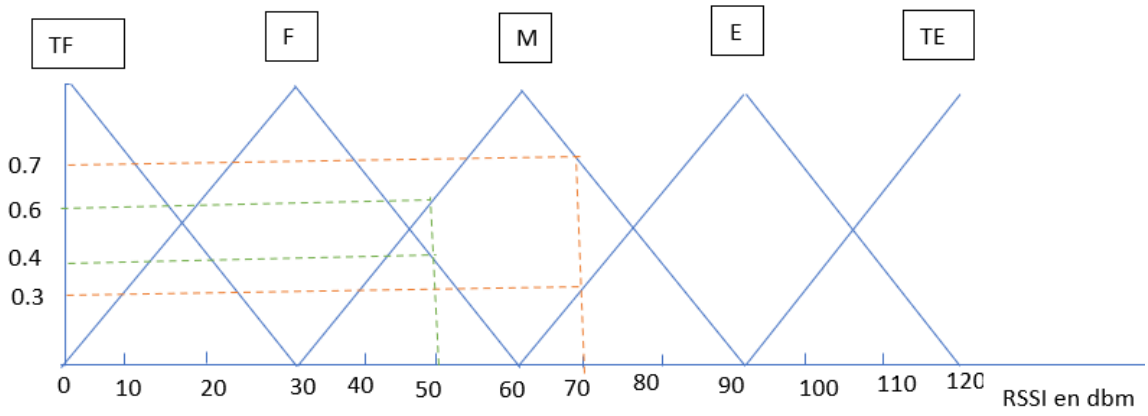

FIGURE 2.5 – Fuzzification du RSSI

Après la fuzzification, le $clusterI$ à comme valeurs floues (NN (0.66 élevé ; 0.34 moyen) RSSI (0.3Eleve ; 0.7 moyen)). Le $clusterJ$ à comme valeurs floues (NN (1 faible) RSSI (0.4 faible ; 0.6 moyen)).

Système d'inférence

Une fois les différents paramètres traduits en « langage flou », l'inférence a pour objectif de construire des règles de décision et de trouver pour chacune d'entre elles la règle d'appartenance de la sortie. La construction de ces règles, principalement basée sur des "ET" se traduit mathématiquement dans le Tableau 2.1.

Tableau 2.1 – Règles de décision

Règles i	Descriptif (Ri)	Règle de sortie Niveau de choix du cluster
1	NN (très faible) ET RSSI (très faible)	Moyen
2	NN (très faible) ET RSSI (faible)	Moyen
3	NN (très faible) ET RSSI (moyen)	Élevé
4	NN (très faible) ET RSSI (élevé)	Très élevé
5	NN (très faible) ET RSSI (très élevé)	Très élevé
6	NN (faible) ET RSSI (très faible)	Moyen
7	NN (faible) ET RSSI (faible)	Moyen
8	NN (faible) ET RSSI (moyen)	Élevé
9	NN (faible) ET RSSI (élevé)	Très élevé
10	NN (faible) ET RSSI (très élevé)	Très élevé
11	NN (moyen) ET RSSI (très faible)	Faible
12	NN (moyen) ET RSSI (faible)	Faible
13	NN (moyen) ET RSSI (moyen)	Moyen
14	NN (moyen) ET RSSI (élevé)	Moyen
15	NN (moyen) ET RSSI (très élevé)	Moyen
16	NN (élevé) ET RSSI (très faible)	Très faible
17	NN (élevé) ET RSSI (faible)	Faible
18	NN (élevé) ET RSSI (moyen)	Faible
19	NN (élevé) ET RSSI (élevé)	Moyen
20	NN (élevé) ET RSSI (très élevé)	Moyen
21	NN (très élevé) ET RSSI (très faible)	Très faible
22	NN (très élevé) ET RSSI (faible)	Très faible
23	NN (très élevé) ET RSSI (moyen)	Faible
24	NN (très élevé) ET RSSI (élevé)	Faible
25	NN (très élevé) ET RSSI (très élevé)	Faible

Après l'établissement de la base des règles, nous allons utiliser les valeurs de vérité associées aux clusters pour activer les règles en utilisant les opérateurs de Zadeh [73]. Les Tableau 2.2 2.3 représentent l'application des règles du système

d'inférence aux clusters I et J.

Tableau 2.2 – Application des règles au cluster I

Cluster I		
Règles	Valeurs	Résultats
NN (élevé) ET RSSI (moyen))	Min (0.66 ; 07)	0.66 faible
NN (élevé) ET RSSI (élevé))	Min (0.66 ; 0.3)	0.3 moyen
NN (moyen) ET RSSI (moyen))	Min(0.34 ; 0.6)	0.34 moyen
NN (moyen) ET RSSI (élevé))	Min(0.34 ; 0.4)	0.34 moyen

Tableau 2.3 – Application des règles au cluster J

Cluster J		
Règles	Valeurs	Résultats
NN (faible) ET RSSI (faible))	Min (1 ; 0.4)	0.4 Moyen
NN (faible) ET RSSI (moyen))	Min (1 ; 0.6)	0.6 Élevé

Agrégation des résultats

Cette étape de l'inférence consiste à regrouper toutes les règles. Ce regroupement est donc effectué à base de «Et» logique, ce qui se traduit par des minima «Min».

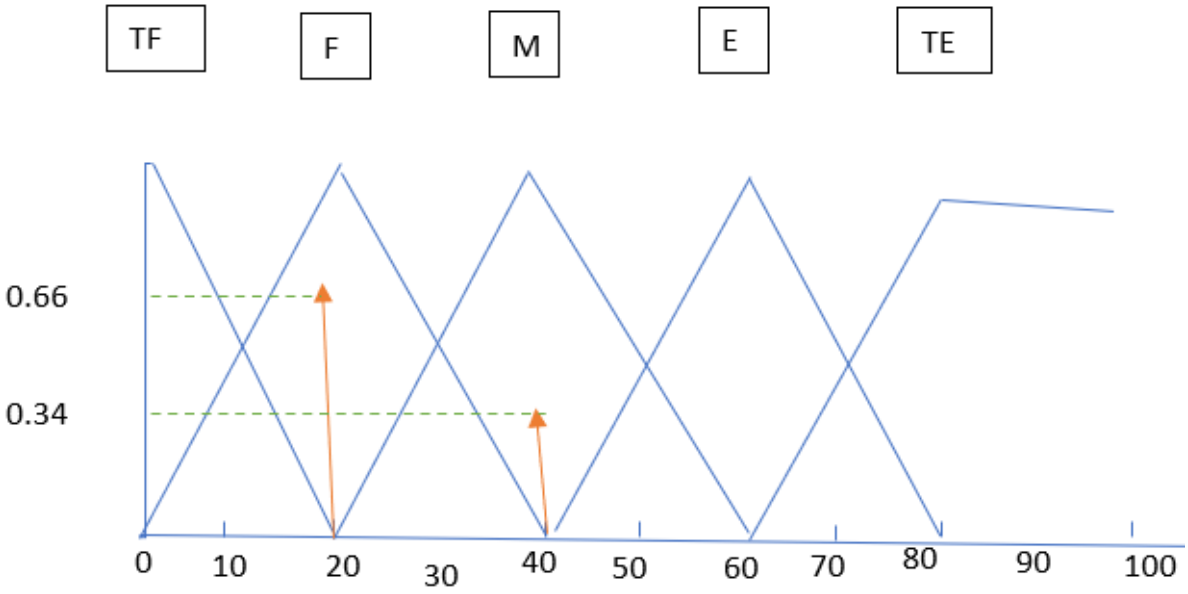

FIGURE 2.6 – Agrégation des valeurs de CHI du $ClusterI$

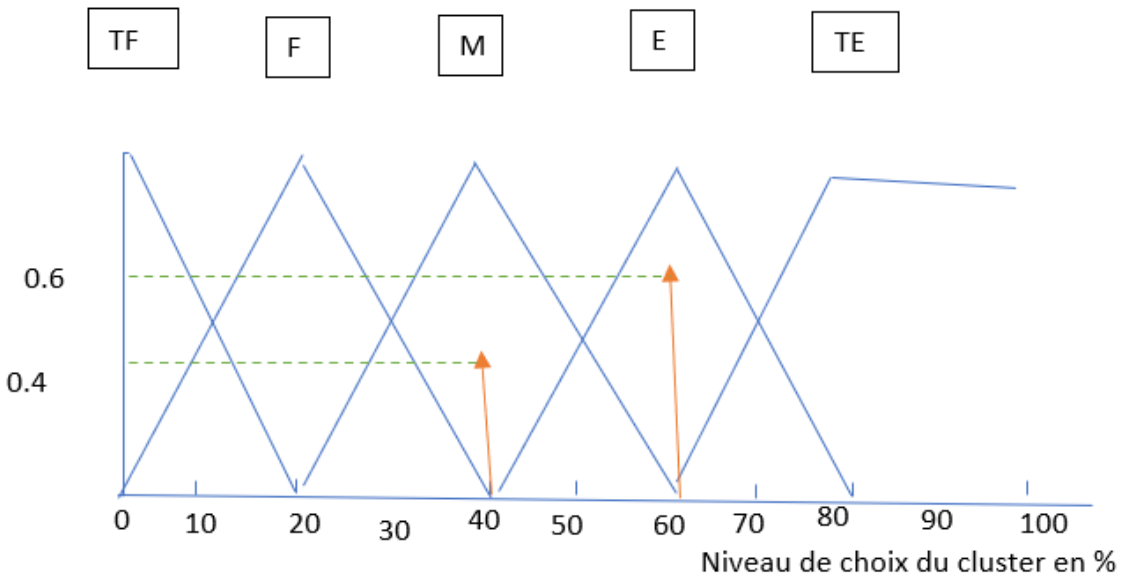

FIGURE 2.7 – Agrégation des valeurs de CHJ du $ClusterJ$

Défuzzification

La défuzzification consiste à transformer le sous-ensemble flou de sortie en une valeur non floue appelée coût associé à la tête de cluster. Un nœud voulant intégrer dans un cluster choisira le cluster dont le CH offre un meilleur coût. Pour le calcul du coût, nous allons utiliser la méthode de la moyenne pondérée qui consiste en la moyenne des maxima des valeurs de la sortie.

$$GC_{Sugeno} = \frac{\sum_{i=0}^{n} \mu_A.k_i}{\sum_{i=0}^{n} \mu_A}[74] \tag{2.3}$$

Le coût associé au cluster I

$$Cout(CHi) = \frac{30 * 0.66 + 60 * 0.34}{0.66 + 0.34}$$

$$Cout(CHi) = 40.2$$

Le coût associé au cluster J

$$Cout(CHj) = \frac{40 * 0.4 + 60 * 0.6}{0.6 + 0.4}$$

$$Cout(CHj) = 52$$

Nous constatons que le cluster I offre une meilleure qualité de liens (RSSI). Cependant, le nœud adhère au cluster J, car c'est le CH J qui offre un meilleur coût en faisant un compromis entre la qualité de liens et le nombre de nœuds du clusters en appliquant notre système flou.

2.4.3 Processus de changement de la tête de cluster

La fonction de tête de cluster se joue à tour de rôle. Contrairement à LEACH-S qui calcule la moyenne des énergies du cluster et compare avec son énergie résiduelle, dans notre nouvelle approche, pour résoudre le problème d'instabilité dans les clusters, nous définissons un quantum d'énergie que le nœud CH devra épuiser avant de céder sa place à un autre nœud. Pour ce faire, le nœud CH sortant informe la nouvelle CH de la liste des nœuds qui ont déjà été élus et aussi celle des nœuds non encore élus. Ainsi, la tête de cluster choisira pour successeur un nœud dans la liste des nœuds non encore élus et ajoutera son identité dans la liste des nœuds déjà élus avant de céder son rôle.

Lorsque tous les nœuds passent à tour de rôle comme tête de cluster, autrement dit lorsqu'une CH n'a plus de nœuds dans la liste des nœuds non encore élus, alors la liste des nœuds déjà élus est automatiquement copiée dans la liste des nœuds non élus, ensuite la liste des nœuds élus est vidée et le processus recommence. Ainsi, les nœuds membres n'auront plus besoin d'informer le CH de leur état d'énergie évitant de cette façon d'alourdir le paquet de contrôle. L'approche contribue également à éviter les opérations de calcul de la moyenne des énergies résiduelles des différents nœuds du cluster et de sa comparaison avec celle du CH.

2.5 Évaluation analytique de la solution

Pour mettre en évidence la pertinence de la solution proposée, nous allons mener une étude analytique. Cette étude consiste à décrire et comparer les processus de fonctionnement de LEACH-S et de BMN-LEACH-S. Pour ce faire, tout d'abord nous décrivons l'environnement de notre étude. Ensuite, nous présentons les résultats de l'expérimentation analytique. Pour finir, nous analysons et interprétons les résultats issus de l'étude.

2.5.1 Contexte d'estimation

Nous considérons deux scénarios d'un réseau de 17 nœuds, l'un fonctionnant avec le protocole LEACH-S et l'autre avec le BMN-LEACH-S. Nous observons le fonctionnement de ces deux scénarios du réseau à différents tours de communication ($T0T1T2 \ldots T16$). Nous supposons que durant un tour de communication, un nœud membre épuise 0.5 unité d'énergie (UE) pour communiquer avec sa tête de cluster et ce dernier aussi épuise 0.5 UE pour le traitement de chaque message. Donc, pour le traitement des messages de N nœuds, la tête de cluster épuisera $N * 0.5$ UE. Une tête de cluster choisit un remplaçant lorsqu'il épuise une quantité d'énergie égale $C = 4UE$. Dans la phase d'expérimentation, les 17 nœuds ont formé 3 clusters nommés cluster I, cluster J et cluster K. Chaque nœud à 20 UE comme énergie

initiale.

Tableau 2.4 – Description des scénarios de réseau

Scénarios de réseau	
Nombre de nœuds	17
Protocole utilisé - Scénario 1	LEACH-S
Protocole utilisé - Scénario 2	BMN-LEACH-S
Consommation d'énergie	
Énergie consommée par nœud pour communiquer avec la tête de cluster	0.5 UE
Énergie consommée par la tête de cluster pour traiter les messages de N nœuds	$N * 0.5$ UE
Remplacement de la tête de cluster	$C = 4$ UE
Phase expérimentale	
Nombre de clusters formés	3
Noms des clusters formés	I, J, K
Énergie initiale de chaque nœud	20 UE
Fonctionnement des scénarios du réseau à différents tours de communication	$(T0T1T2 \ldots T16)$

2.5.2 Résultats de l'expérimentation analytique

Dans cette section, nous menons des expérimentations et nous présentons les résultats de ces expérimentations en fonction des deux protocoles LEACH-S et BMN-LEACH-S.

2.5.2.1 Expérimentation avec LEACH-S

Dans LEACH-S, après la phase initiale, le réseau a formé 3 clusters I, J et K. Le cluster I contient 9 nœuds, le cluster J contient 5 nœuds et le cluster k contient 3 nœuds. NI1, NJ1 et NK1 sont respectivement les têtes de cluster des clusters I, J et K.

Au tour T1, nous avons la topologie présentée à la Figure 2.8. La tête de cluster est en bleu et les autres nœuds en vert. Les liens symbolisent le sens de la communication.

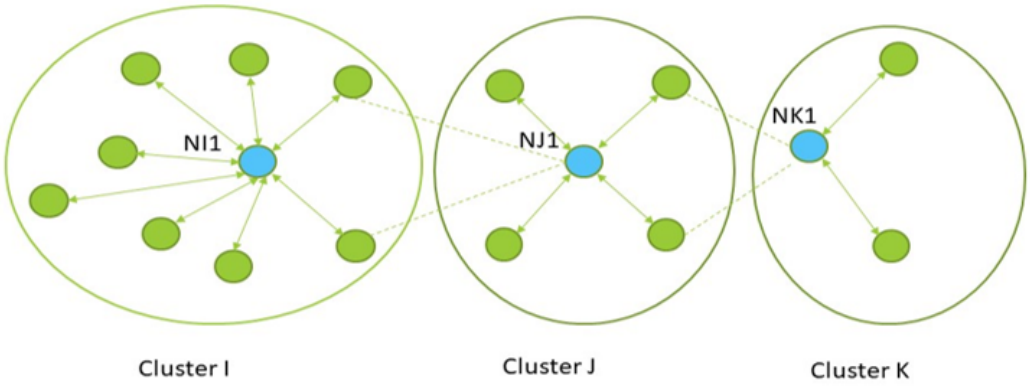

FIGURE 2.8 – Topologie des nœuds avec LEACH-S à la phase initiale

Tableau 2.5 – État des nœuds avec LEACH-S après le tour 1

	Cluster I									Cluster J					Cluster K		
Nœud	NI1	NI2	NI3	NI4	NI5	NI6	NI7	NI8	NI9	NJ1	NJ2	NJ3	NJ4	NJ5	NK1	NK2	NK3
E_r	16	19.5	19.5	19.5	19.5	19.5	19.5	19.5	19.5	18	19.5	19.5	19.5	19.5	19	19.5	19.5
État	INSTABILITE									STABLE					STABLE		

Les nœuds envoient périodiquement des données de leurs environnements au CH. Un tour correspond à l'évènement d'envoi de données au CH, soit une période. Nous répétons l'expérience jusqu'au tour T16 et nous récoltons les informations sur l'énergie résiduelle des nœuds afin de compter le nombre d'instabilités sur le réseau. Nous appelons instabilité le changement de CH. Ce changement engendre une diffusion spécifique de messages de contrôle pour annoncer la nouvelle CH. Nous supposons que l'énergie consentie pour la réception de paquet d'un nœud voisin par un nœud ordinaire (non CH) est négligeable. Un nœud non CH détruit ce paquet au niveau de la couche accès réseau. Le Tableau 2.6 récapitule les informations sur l'évolution du niveau d'énergie et le nombre de changements de CH des nœuds du tour T1 au tour T16.

Tableau 2.6 – Recapitulatif des énergies des nœuds avec LEACH-S de T_0 à T_{16}

LEACH-S																	
Cluster	I									J					K		
Nœud	NI1	NI2	NI3	NI4	NI5	NI6	NI7	NI8	NI9	NJ1	NJ2	NJ3	NJ4	NJ5	NK1	NK2	NK3
T0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
T1	16	19.5	19.5	19.5	19.5	19.5	19.5	19.5	19.5	18	19.5	19.5	19.5	19.5	19	19.5	19.5
État	INSTABILITÉ									STABLE					STABLE		
T2	15.5	15.5	19	19	19	19	19	19	19	16	19	19	19	19	18	19	19
État	INSTABILITÉ									INSTABILITÉ					STABLE		
T3	15	15	15	18.5	18.5	18.5	18.5	18.5	18.5	15.5	17	18.5	18.5	18.5	17	18.5	18.5
État	INSTABILITÉ									STABLE					STABLE		
T4	14.5	14.5	14.5	14.5	18	18	18	18	18	15	15	18	18	18	16	18	18
État	INSTABILITÉ									INSTABILITÉ					INSTABILITÉ		
T5	14	14	14	14	14	17.5	17.5	17.5	17.5	14.5	14.5	16	17.5	17.5	15.5	17	17.5
État	INSTABILITÉ									STABLE					STABLE		
T6	13.5	13.5	13.5	13.5	13.5	13.5	17	17	17	14	14	14	17	17	15	16	17
État	INSTABILITÉ									INSTABILITÉ					STABLE		
T7	13	13	13	13	13	13	13	16.5	16.5	13.5	13.5	13.5	15	16.5	14.5	15	16.5
État	INSTABILITÉ									STABLE					STABLE		
T8	12.5	12.5	12.5	12.5	12.5	12.5	12.5	12.5	16	13	13	13	13	16	14	14	16
État	INSTABILITÉ									INSTABILITÉ					INSTABILITÉ		
T9	12	12	12	12	12	12	12	12	12	12.5	12.5	12.5	12.5	14	13.5	13.5	15
État	INSTABILITÉ									STABLE					STABLE		
T10	11.5	11.5	11.5	11.5	11.5	11.5	11.5	11.5	08	12	12	12	12	12	13	13	14
État	INSTABILITÉ									INSTABILITÉ					STABLE		
T11	11	11	11	11	11	11	11	07.5	07.5	11.5	11.5	11.5	11.5	10	12.5	12.5	13
État	INSTABILITÉ									STABLE					STABLE		
T12	10.5	10.5	10.5	10.5	10.5	10.5	07	07	07	11	11	11	11	08	12	12	12
État	INSTABILITÉ									INSTABILITÉ					INSTABILITÉ		
T13	10	10	10	10	10	06.5	06.5	06.5	06.5	10.5	10.5	10.5	09	07.5	11.5	11.5	11
État	INSTABILITÉ									STABLE					STABLE		
T14	09.5	09.5	09.5	09.5	06	06	06	06	06	10	10	10	07	07	11	11	10
État	INSTABILITÉ									INSTABILITÉ					STABLE		
T15	09	09	09	05.5	05.5	05.5	05.5	05.5	05.5	09.5	09.5	08	06.5	06.5	10.5	10.5	09
État	INSTABILITÉ									STABLE					STABLE		
T16	08.5	08.5	05	05	05	05	05	05	05	09	09	06	06	06	10	10	08
État	INSTABILITÉ									INSTABILITÉ					INSTABILITÉ		

2.5.2.2 Expérimentation avec BMN-LEACH-S

Dans BMN-LEACH-S, grâce à notre processus d'équilibrage des nœuds membres, on se retrouve cette fois-ci avec 7 nœuds dans le cluster I, 5 dans le cluster J et 5 dans le cluster k (voir Figure 2.9). Les CH pour ces clusters I, J et K sont respectivement NI1, NJ1 et NK1. En rappel, dans BMN-LEACH-S l'adhésion à un cluster est fonction du coût que le nœud non CH a avec le CH. Chaque nœud choisit le CH avec lequel il a le plus grand coût. Ce coût est calculé avec notre système à logique floue présenté dans la sous section 2.4.2.2.

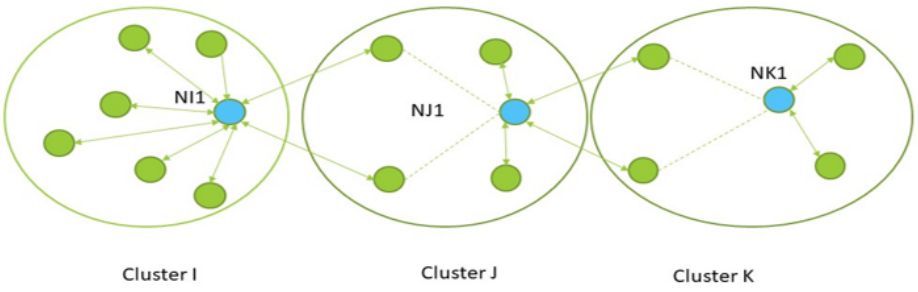

FIGURE 2.9 – Topologie des nœuds avec BMN-LEACH-S à la phase initiale

Au tour T_1, nous avons les résultats suivants :

Tableau 2.7 – Recapitulatif des énergies des nœuds avec BMN-LEACH-S au tour T_1

État	STABLE							STABLE					STABLE				
Nœud	NI1	NI2	NI3	NI4	NI5	NI6	NI7	NJ1	NJ2	NJ3	NJ4	NJ5	NK1	NK2	NK3	NK4	NK5
E_r	17	19.5	19.5	19.5	19.5	19.5	19.5	18	19.5	19.5	19.5	19.5	18	19.5	19.5	19.5	19.5

Nous répétons l'expérience jusqu'au tour T_{16} et nous récoltons les informations afin de compter le nombre d'instabilités sur le réseau. Les résultats sont récapitulés dans la Tableau 2.8.

Tableau 2.8 – Recapitulatif des énergies des nœuds avec BMN-LEACH-S de T_0 à T_{16}

BMN-LEACH-S																	
Cluster	I							J					K				
Nœud	NI1	NI2	NI3	NI4	NI5	NI6	NI7	NJ1	NJ2	NJ3	NJ4	NJ5	NK1	NK2	NK3	NK4	NK5
T0	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
T1	17	19.5	19.5	19.5	19.5	19.5	19.5	18	19.5	19.5	19.5	19.5	18	19.5	19.5	19.5	19.5
État	STABLE							STABLE					STABLE				
T2	14	19	19	19	19	19	19	16	19	19	19	19	16	19	19	19	19
État	INSTABILITÉ							INSTABILITÉ					INSTABILITÉ				
T3	13.5	16	18.5	18.5	18.5	18.5	18.5	15.5	17	18.5	18.5	18.5	15.5	17	18.5	18.5	18.5
État	STABLE							STABLE					STABLE				
T4	13	13	18	18	18	18	18	15	15	18	18	18	15	15	18	18	18
État	INSTABILITÉ							INSTABILITÉ					INSTABILITÉ				
T5	12.5	12.5	15	17.5	17.5	17.5	17.5	14.5	14.5	16	17.5	17.5	14.5	14.5	16	17.5	17.5
État	STABLE							STABLE					STABLE				
T6	12	12	12	17	17	17	17	14	14	14	17	17	14	14	14	17	17
État	INSTABILITÉ							INSTABILITÉ					INSTABILITÉ				
T7	11.5	11.5	11.5	14	16.5	16.5	16.5	13.5	13.5	13.5	15	16.5	13.5	13.5	13.5	15	16.5
État	STABLE							STABLE					STABLE				
T8	11	11	11	11	16	16	16	13	13	13	13	16	13	13	13	13	16
État	INSTABILITÉ							INSTABILITÉ					INSTABILITÉ				
T9	10.5	10.5	10.5	10.5	13	15.5	15.5	12.5	12.5	12.5	12.5	14	12.5	12.5	12.5	12.5	14
État	STABLE							STABLE					STABLE				
T10	10	10	10	10	10	15	15	12	12	12	12	12	12	12	12	12	12
État	INSTABILITÉ							INSTABILITÉ					INSTABILITÉ				
T11	09.5	09.5	09.5	09.5	09.5	12	14.5	11.5	11.5	11.5	11.5	10	11.5	11.5	11.5	11.5	10
État	STABLE							STABLE					STABLE				
T12	09	09	09	09	09	09	14	11	11	11	11	08	11	11	11	11	08
État	INSTABILITÉ							INSTABILITÉ					INSTABILITÉ				
T13	08.5	08.5	08.5	08.5	08.5	08.5	11	10.5	10.5	10.5	09	07.5	10.5	10.5	10.5	09	07.5
État	STABLE							STABLE					STABLE				
T14	08	08	08	08	08	08	08	10	10	10	07	07	10	10	10	07	07
État	INSTABILITÉ							INSTABILITÉ					INSTABILITÉ				
T15	07.5	07.5	07.5	07.5	07.5	05	07.5	09.5	09.5	08	06.5	06.5	09.5	09.5	08	06.5	06.5
État	STABLE							STABLE					STABLE				
T16	07	07	07	07	07	02	07	09	09	06	06	06	09	09	06	06	06
État	INSTABILITÉ							INSTABILITÉ					INSTABILITÉ				

D'après l'étude analytique que nous avons menée, au cours des 16 tours, nous

remarquons que le nombre total de cas d'instabilités dans LEACH-S pour tout le réseau est de 28 contre 24 dans BMN-LEACH-S(voir Tableau 2.6 et Tableau 2.8).

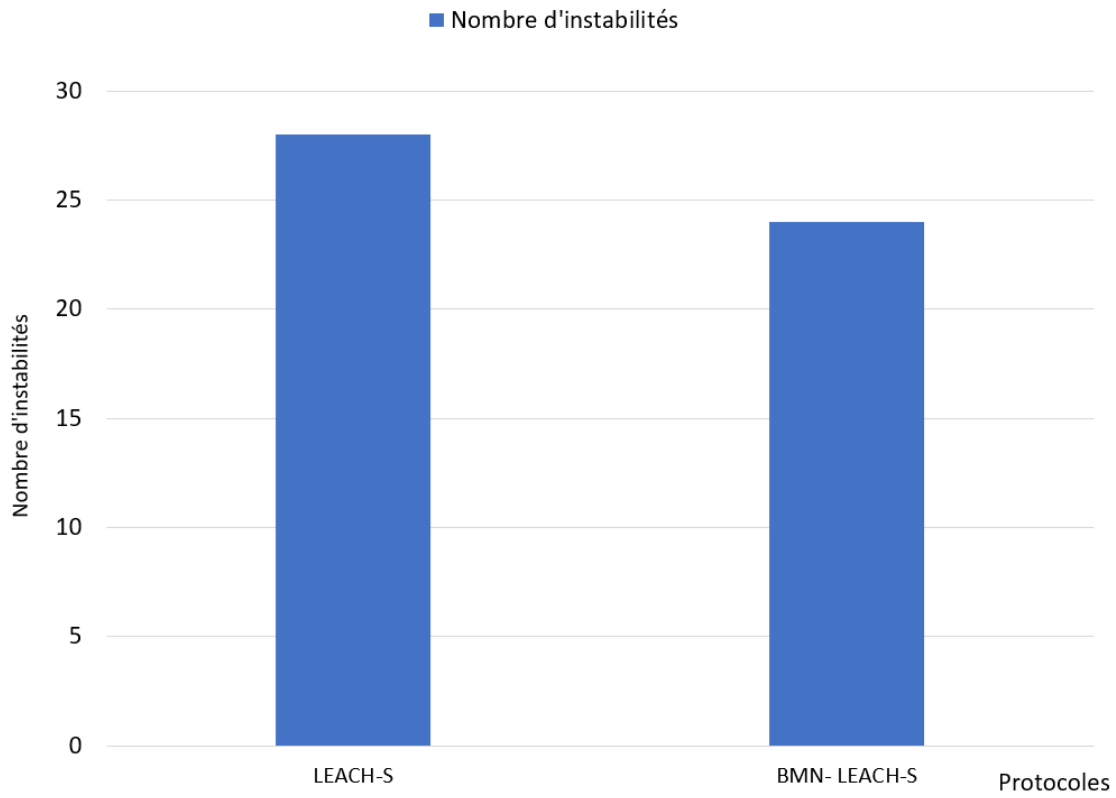

FIGURE 2.10 – Nombre d'instabilités dans le réseau

En observant en détail le nombre d'instabilités par cluster, nous remarquons que dans LEACH-S, pour les clusters I, J et K, nous avons respectivement 16, 8 et 4 cas d'instabilités. Dans le réseau fonctionnant avec BMN-LEACH-S, avec cette expérimentation, nous avons le même nombre (8) d'instabilités pour chacun des trois clusters I, J, K (voir Tableau 2.6 et Tableau 2.8).

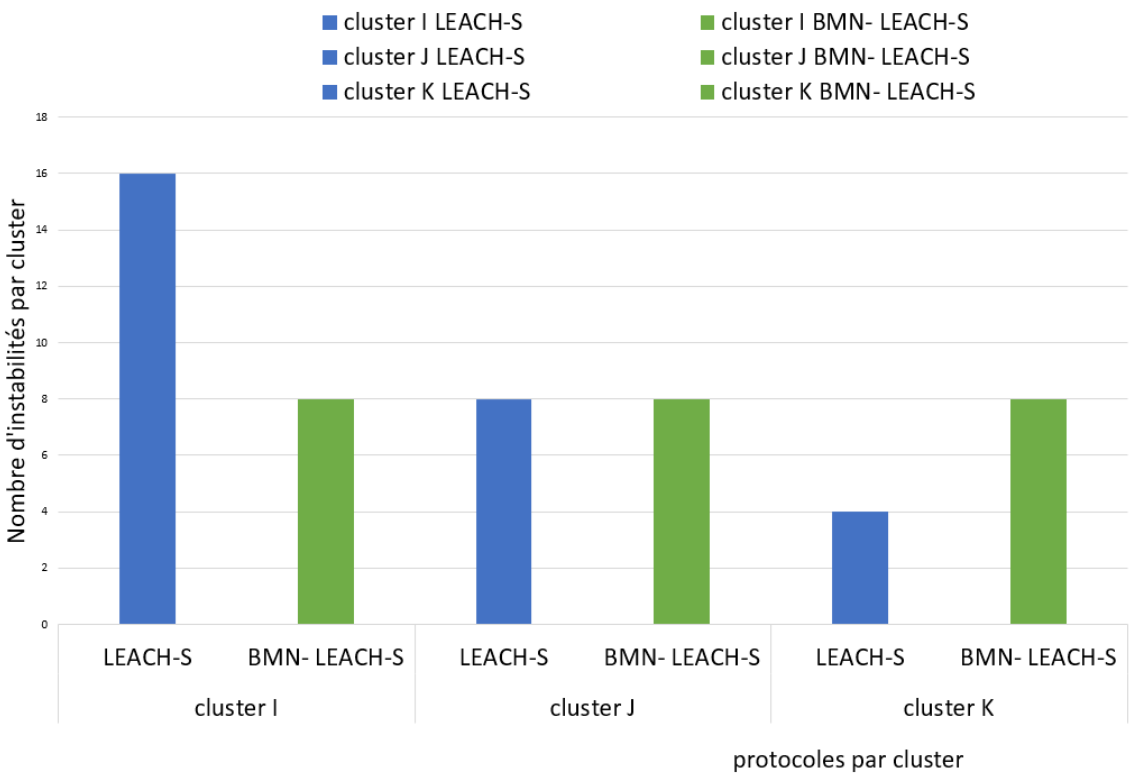

FIGURE 2.11 – Nombre d'instabilités par cluster et par protocole

2.5.3 Analyse et interprétation

Il ressort des résultats des expérimentations que dans tout le réseau, avec le protocole LEACH-S nous observons 28 instabilités contre 24 pour BMN-LEACH-S (voir Figure 2.10). L'utilisation de LEACH-S produit plus d'instabilités que l'utilisation de BMN-LEACH-S. Si nous analysons les résultats par cluster, le déséquilibre est plus visible.

Au Figure 2.11, nous constatons que le changement de CH est équilibré entre les clusters dans BMN-LEACH-S par rapport à LEACH-S. Sachons que chaque changement de tête de cluster engendre des diffusions spéciales d'annonces de nouvelles CH. Nous pouvons déduire que BMN-LEACH-S équilibre la charge de routage entre les différents clusters comparé à LEACH-S. Cela conduit à la réduction de la consommation globale d'énergie du réseau. BMN-LEACH-S améliore aussi le taux de pertes de paquets dues aux changements de topologie, à la surcharge du réseau et au délai moyen de transmission des paquets. Avec ces expérimentations, la CH a été changé au maximum 8 fois pour les clusters pilotés par BMN-LEACH-S contre 16 fois pour LEACH-S. Nous pouvons dire que BMN-LEACH-S équilibre la charge du trafic entre les différentes CH. Ce qui augmente la durée de vie du réseau.

2.6 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle solution de routage par clustering appelée BMN-LEACH-S. Cette méthode est une amélioration du protocole LEACH-S afin de diminuer l'instabilité de certains clusters due au changement fréquent de CH. Notre approche résout le problème de répartition inégale des nœuds membres entre les clusters.

Pour ce faire, BMN-LEACH-S alloue un quantum d'énergie à chaque tête de cluster. C'est lorsque le CH épuise le quantum qui lui est attribué qu'il est autorisé à désigner un remplaçant parmi les nœuds non encore élu. Aussi, il implémente une fonction qui se base sur un système à logique floue utilisant comme paramètres le nombre de nœuds du cluster et la puissance du signal reçu pour estimer le coût d'un CH. Chaque nœud se base sur ce coût pour effectuer son choix de CH. Cela permet une répartition plus équitable des nœuds dans les différents clusters. Une évaluation analytique de notre solution montre qu'elle permet de réduire les instabilités du réseau par rapport à LEACH-S, ce qui améliore la durée de vie du réseau. Ces performances devraient être confirmées par des simulations et des tests en réel.

Dans le prochain chapitre, nous exposons notre contribution sur la gestion de la consommation d'énergie dans les réseaux de capteurs sans fil défini par logiciel.

Chapitre 3

Proposition de mécanismes de gestion du routage et de réduction des messages de contrôle pour améliorer la durée de vie des WSN

3.1 Introduction

Les solutions dans ce chapitre sont basées sur le SDN-WISE. Nous présentons d'abord l'architecture SDN-WISE et son mode de fonctionnement. Ensuite, nous présentons les nouveaux mécanismes de routage que nous proposons. Enfin, nous exposons les résultats de l'étude des performances de nos solutions comparées aux approches standards.

3.2 Présentation du SDN-WISE

Dans cette section, un aperçu de la solution SDN-WISE est fourni. Nous décrivoons brièvement les exigences à satisfaire dans la conception du SDN-WISE. Nous donnons ensuite un aperçu de l'approche technique du SDN-WISE. Enfin, nous présentons l'architecture du protocole SDN-WISE.

3.2.1 Exigences du SDN-WISE

Des exigences ont été analysées pour étendre le paradigme SDN aux WSN. Ces exigences sont liées aux caractéristiques contraignantes des WSN par rapport au réseau filaire traditionnel. En effet, les WSN sont caractérisés par de faibles capacités en termes de traitement, de mémoire et d'énergie. Aussi, les applications WSN sont moins exigeantes en termes de débit de données. De ce fait, le SDN-WISE doit être efficace dans l'utilisation des ressources des nœuds capteurs, même si cette efficacité se traduit par une baisse du débit de données. Pour être économe en énergie, le SDN-WISE prend en charge le cycle de service, c'est-à-dire la possi-

bilité d'éteindre périodiquement l'interface radio d'un nœud capteur et d'effectuer l'agrégation des données. Ces mesures ne sont pas prises en compte dans les réseaux filaires d'OpenFlow. Par ailleurs, les interactions entre les nœuds capteurs et les contrôleurs doivent être réduites autant que possible pour atteindre l'efficacité du système. Dans ce contexte, un certain niveau de logique de contrôle programmable dans les nœuds capteurs peut leur permettre de prendre des décisions sans interagir avec le contrôleur lorsque seules des informations locales sont nécessaires. Cela nécessite toutefois l'introduction d'un état alors que l'instance OpenFlow standard de SDN est sans état.

En plus, les WSN étant intrinsèquement centrés sur les données, plusieurs solutions ont été proposées pour que les protocoles de réseau soient conscients du contenu des paquets. Par conséquent, les nœuds SDN-WISE peuvent traiter les paquets sur la base du contenu stocké dans leur en-tête et leur charge utile. De même, dans OpenFlow, les paquets sont classés sur la base de l'égalité entre un certain champ de l'en-tête du paquet et une chaîne d'octets donnée.

Contrairement à cela, dans le SDN-WISE cette classification peut être effectuée sur la base d'autres opérateurs relationnels plus complexes, par exemple, supérieur à, inférieur à, différent de, etc. Enfin, la nature centrée sur les données des WSN implique une autre différence significative entre le comportement attendu de SDN-WISE et d'OpenFlow. En effet, dans OpenFlow, les ressources du réseau sont divisées par le FlowVisor en tranches, chacune assignée à un contrôleur et un paquet ne peut appartenir qu'à une seule tranche.

Dans les WSN, au contraire, le même élément de données peut intéresser plusieurs applications utilisant des contrôleurs différents. Par conséquent, dans SDN-WISE, un paquet n'est pas nécessairement lié à un contrôleur, c'est-à-dire que différents contrôleurs peuvent spécifier différentes règles pour le même paquet.

3.2.2 Mode opératoire du SDN-WISE

Le comportement des nœuds de détection SDN-WISE [10] est entièrement codé dans trois structures de données, à savoir : le tableau des états WISE, le tableau des identifiants acceptés et le tableau des flux WISE. Comme dans la plupart des approches SDN, ces structures sont remplies avec les informations provenant des contrôleurs, exécutés dans les serveurs appropriés. De cette façon, les contrôleurs définissent les politiques de mise en réseau qui seront mises en œuvre par les nœuds capteurs.

A tout moment, les nœuds SDN-WISE sont caractérisés par un état actuel pour chaque contrôleur actif. Un état est une chaîne de bits. Le tableau des états WISE est la structure de données contenant les valeurs des états actuels. Étant donné la nature de diffusion du support sans fil, les nœuds capteurs recevront également des paquets qui ne leur sont pas destinés (même pas pour être transmis). Le tableau des

identifiants acceptés permet à chaque nœud capteurs de sélectionner uniquement les paquets qu'il doit traiter. En fait, l'en-tête des paquets contient un champ dans lequel un ID accepté est spécifié.

3.2.2.1 Processus de traitement d'un paquet par un nœud

Un nœud, à la réception d'un paquet, contrôle si l'ID contenu dans ce champ est répertorié dans son tableau d'IDs acceptés [10]. Si c'est le cas, le nœud poursuivra le traitement du paquet, sinon il l'abandonnera. Dans le cas où le paquet doit être traité, le nœud capteur parcourt les entrées de sa table de flux WISE. Chaque entrée de la table de flux WISE contient une section *Matching Rules* qui spécifie les conditions dans lesquelles l'entrée s'applique.

Dans SDN-WISE, les règles de correspondance peuvent prendre en compte n'importe quelle partie du paquet actuel ainsi que n'importe quel bit de l'état actuel. Si les règles de correspondance sont satisfaites, le nœud de détection effectue une action spécifiée de l'entrée de la table de flux WISE. Il convient de noter que cette action peut porter sur la manière de traiter le paquet ainsi que sur la manière de modifier l'état actuel. Si aucune entrée ne figure dans le tableau de flux WISE dont les règles de correspondance s'appliquent au paquet/état actuel, une demande est envoyée aux contrôleurs. Pour contacter les contrôleurs, un nœud doit avoir une entrée dans la table de flux WISE indiquant son meilleur prochain saut vers l'un des puits. Cette entrée est différente des autres, car elle n'est pas définie par un contrôleur, mais est découverte par chaque nœud de manière distribuée. A cette fin, un protocole approprié est exécuté par la couche de découverte de la topologie (TD).

3.2.2.2 Couche de découverte de topologie

La découverte de la topologie est basée sur l'échange et le traitement de paquet approprié appelé paquet TD [10]. Un paquet contient des informations sur le niveau de la batterie et la distance du puits le plus proche en termes de nombre de sauts. Chaque fois qu'un nœud reçoit un de ces paquets, il compare le meilleur saut suivant actuel avec les informations qu'il vient d'acquérir, puis il choisit le meilleur saut suivant en donnant la priorité au nombre de sauts, puis à la valeur RSSI reçue avec le message et enfin au niveau de l'énergie résiduelle. Ces informations sont également utilisées pour alimenter une liste de voisins WISE. Cette liste contient les adresses des nœuds voisins, leurs RSSI et leurs niveaux de batterie. Cette table est envoyée périodiquement à la couche de gestion de la topologie (TM) afin de construire une représentation graphique du réseau. Ensuite, la table est totalement effacée et reconstruite avec les paquets TD entrants afin d'avoir toujours une vue actualisée de la topologie locale.

L'un des contrôleurs agit comme un proxy entre le réseau physique et les autres contrôleurs. Il est appelé WISE-Visor et est l'analogue du FlowVisor dans les réseaux

OpenFlow traditionnels. Les contrôleurs spécifient les politiques de gestion du réseau qui doivent être mises en œuvre par le WSN et qui peuvent dépendre de l'application. En conséquence, les contrôleurs peuvent interagir avec l'application. Les nœuds capteurs ont des capacités limitées en termes de mémoire, par conséquent, le choix de la taille des différentes structures de données est très important. Le choix optimal de cette taille dépend de plusieurs caractéristiques spécifiques au déploiement définies par le WISE-Visor lors de la phase d'initialisation.

3.2.3 Architecture du protocole SDN-WISE

Dans les réseaux SDN-WISE, on distingue les nœuds capteurs et un ou plusieurs puits. Les puits sont les passerelles entre les nœuds capteurs qui exécutent le plan de données et les éléments qui mettent en œuvre le plan de contrôle. La pile de protocoles du plan des données, principalement exécutée par les nœuds capteurs, est présentée dans la partie gauche de la Figure 3.1. La pile de protocoles du récepteur et les autres éléments mettant en œuvre le plan de contrôle sont décrits dans la partie droite de la Figure 3.1. Les nœuds capteurs comprennent un émetteur-récepteur IEEE 802.15.4 et une unité de microcontrôleur (MCU). Au-dessus de la pile de protocoles IEEE 802.15.4, la couche d'acheminement s'exécute dans la MCU qui traite les paquets entrants comme spécifiés dans une table de flux WISE. Cette table est continuellement mise à jour par la couche d'acheminement en fonction des commandes de configuration envoyées par les contrôleurs.

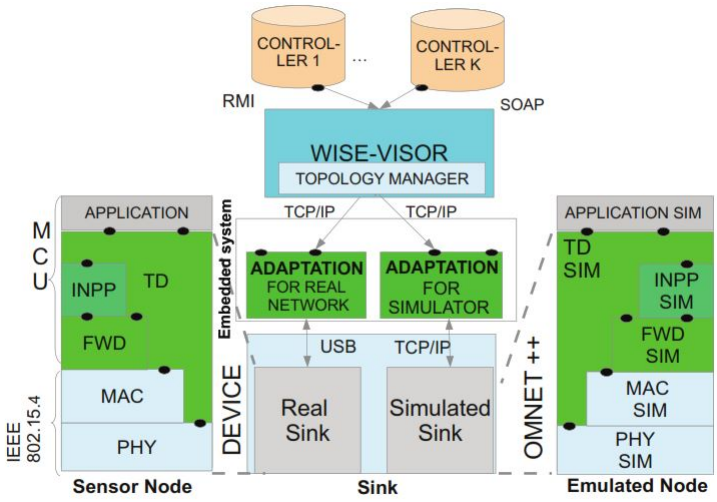

FIGURE 3.1 – Architecture protocolaire de SDN-WISE [10]

La couche INPP (*In-Network Packet Processing*) s'exécute au-dessus de la couche de transfert. Elle est responsable d'opérations telles que l'agrégation de données ou d'autres traitements au sein du réseau. Dans la mise en œuvre actuelle de SDN-WISE, la couche INPP concatène les petits paquets qui doivent être envoyés par des

chemins similaires permettent ainsi de réduire la surcharge du réseau.

La couche de découverte de topologie, en revanche, peut accéder à toutes les couches de la pile de protocoles au moyen d'API appropriées. Ainsi, elle peut recueillir des informations locales auprès des nœuds et contrôler leur comportement à toutes les couches, en fonction des indications fournies par les contrôleurs. La couche TD fournit également une API à la couche application, qui étend les API de l'IEEE 802. Cela garantit la compatibilité avec les applications existantes.

Dans le plan de contrôle, les logiques de gestion du réseau sont dictées par un ou plusieurs contrôleurs, dont le WISE-Visor. Le WISE-Visor comprend une couche de gestion de la topologie (TM) qui rend abstraites les ressources du réseau de sorte que différents réseaux logiques, avec différentes politiques de gestion définies par différents contrôleurs, peuvent fonctionner sur le même ensemble de dispositifs physiques. La couche TM a accès aux API offertes par toutes les couches de protocole. Ces API permettent de contrôler le comportement de toutes les piles de protocoles et donc de mettre en œuvre des opérations intercouches. L'utilisation de la couche TM est motivée par deux exigences :

- collecter des informations locales à partir des nœuds et les envoyer aux contrôleurs sous la forme d'un graphe du réseau (informations relatives à la topologie, au niveau d'énergie résiduelle, au SNR sur les liens, etc) ;
- contrôler toutes les couches de la pile de protocoles comme spécifiés par le(s) contrôleur(s).

À cette fin, entre le dispositif récepteur (caractérisé par la même pile de protocoles que les nœuds capteurs) et le WISE-Visor, il y a la couche d'adaptation qui est responsable du formatage des messages reçus du dispositif récepteur de manière à ce qu'ils puissent être traités par le WISE-Visor et vice versa. Les contrôleurs peuvent s'exécuter soit dans le même nœud qui héberge la couche TM ou dans des serveurs distants. Par conséquent, les interactions entre les contrôleurs et la couche TM peuvent se produire de plusieurs manières, comme le montre la partie centrale de la Figure 3.1. En fait, dans le cas où les contrôleurs sont exécutés dans le même nœud hébergeant la couche TM, les interactions se feront par le biais des méthodes Java offertes par la couche TM.

Les interactions peuvent également se faire par le biais de l'invocation de méthodes à distance (RMI) de Java ou du protocole SOAP (*Simple Object Access Protocol*). De cette façon, les programmeurs peuvent mettre en œuvre les contrôleurs soit en Java, soit dans certains langages de programmation Web. Enfin, notons que la pile de protocoles SDN-WISE comprend également une couche d'adaptation spécifique qui peut interagir avec un puits simulé (et non un puits réel). De cette façon, le plan de contrôle peut définir les politiques de mise en réseau d'un réseau simulé.

3.2.4 Problématiques dans le processus de routage dans SDN-WISE

Nous travaillons principalement sur deux problématiques dans SDN-WISE.

La prémière est decrite comme suit : Pour prolonger la durée de vie du réseau, le trafic doit être acheminé de manière à ce qu'il soit distribué aussi régulièrement que possible sur le réseau. En utilisant SDN-Wise, les paquets sont envoyés sur le chemin le plus court. Cette approche utilise l'algorithme de Dijkstra pour le calcul de route. Les nœuds sur les plus courts chemins en termes de nombre de sauts se trouvent être plus sollicités et par conséquent se déchargent plus rapidement de leurs énergies que les nœuds en périphérie du réseau. Cela réduit la durée de vie du réseau WSN. La Figure 3.3 présente le processus de fonctionnement de l'algorithme de Dijkstra. Dans ce travail, nous proposons l'approche DEARP (Dynamique Energie Aware Routing Protocol) qui va prendre en compte l'énergie résiduelle des nœuds dans les décisions de routage.

Pour la deuxième problématique, nous constatons que tout comme avec OpenFlow, dans SDN-WISE lorsqu'il n'y a pas de correspondance dans la table de flux WISE d'un nœud capteur, une requête est immédiatement envoyée au contrôleur afin que ce dernier donne la directive à suivre. Cependant, cette façon d'envoyer des requêtes après constatation pourrait entraîner l'envoi de doublons de requêtes conduisant à une surcharge du réseau lorsqu'elle se produit à grande échelle et simultanément. En effet, ce processus se répète à chaque réception de paquet même si ces paquets visent la même destination jusqu'à ce que le nœud reçoive la directive de continuer. Nous proposons un nouveau mécanisme appelé *Congestion Management Flow Request Control Message Quenching* (CMFR-CMQ) qui va permettre d'éviter la production des doublons des messages de types "*Request*" et aussi éviter que les nœuds soient congestionnés lors du processus de routage.

3.3 Notre approche DEARP

Pour résoudre la problématique 1, nous avons proposé la solution DEARP. Dans cette section, nous présentons dans un premier temps le mode de fonctionnement de DEARP. Ensuite, nous décrivons les paramètres et l'algorithme de DEARP. Enfin, nous menons une étude analytique des performances de la solution. En dernier lieu, nous faisons une implémentation par simulation afin d'évaluer et comparer les performances de la solution DEARP par rapport au processus de routage de SDN-WISE standard.

3.3.1 Mode de fonctionnement de la solution DEARP

DEARP est un protocole qui fonctionne sur l'architecture SDN-WISE. Il permet ainsi la séparation du plan de contrôle du plan de données. DEARP gère le plan de contrôle tout comme dans SDN-WISE. Cependant, à la différence de SDN-WISE qui utilise l'algorithme de Dijkstra pour le routage des paquets de données, le chemin utilisé pour l'envoi des paquets de données est établi par l'application DEARP qui calcule le meilleur chemin en termes de nombre de sauts et en basculant vers des chemins même plus longs lorsque le plus court chemin s'affaiblit en énergie. Ce mécanisme de basculement vers d'autres chemins est contrôlé par une variable seuil d'énergie nommée K. Cela permet aux nœuds du réseau de se décharger au même rythme. Ce qui augmente la durée de vie du réseau.

3.3.2 Description de l'algorithme utilisé par le DEARP

Dans cette section, nous présentons les principaux paramètres utilisés dans l'algorithme.

durée de vie d'un réseau TTL : est définie comme l'instant de déploiement du réseau jusqu'au moment où le premier nœud du réseau se décharge complèmentement de son énergie.

n_i : désigne un nœud i.

P_i : ensemble de nœuds appartenant à un chemin i

Q_i : ensemble de nœuds de tous les chemins possibles entre un nœud n_i et le puits.

$N(Path)$: est le nombre de chemins possibles pour envoyer des données d'un nœud n_i au nœud puits.

$E(n_i)$: énergie résiduelle d'un nœud.

$E(P) = minE(n_i)$ avec $n_i \in P_i$: correspond à l'énergie du nœud de la route qui a la plus petite énergie résiduelle.

$E(Q) = minE(n_i) n_i \in Q$: l'énergie de l'ensemble Q correspond à l'énergie du nœud qui a la plus petite énergie résiduelle dans l'ensemble Q.

K_i : le seuil d'énergie minimale pour le basculement pour un nœud n_i

C : la constante pour la décrémentation du seuil.

DEARP fixe dans une variable K, le seuil d'énergie minimale pour le basculement vers le chemin le plus court ayant une quantité d'énergie supérieure à K. S'il existe plusieurs plus courts chemins, alors le choix se portera sur le plus court chemin ayant une quantité d'énergie plus élevée. Lorsque l'énergie résiduelle d'un des nœuds du chemin choisi est en dessous du seuil fixé par le contrôleur, celui-ci recherche à nouveau le plus court chemin parmi les chemins possibles restants dont les nœuds

ont une énergie résiduelle supérieure au seuil fixé par le contrôleur. S'il n'y a plus de chemin avec une énergie supérieure au seuil, alors le contrôleur décrémente la valeur du seuil K de sorte à tenir compte à nouveau de tous les chemins possibles. Pour ce faire, le contrôleur affecte à la valeur seuil d'énergie K, la valeur minimale de l'énergie de l'ensemble des chemins possibles ($E(Q) = minE(n_i)n_i \in Q$) décrémenté de C qui est une constante. Avec cette approche, un chemin abandonné précédemment parce qu'ayant franchi le seuil prédéfini peut se retrouver être compétitif à nouveau après l'exploitation des chemins alternatifs et la décrémentation du seuil. Cela conduit à un meilleur équilibrage de la consommation générale d'énergie du réseau tout en profitant au mieux du plus court chemin pour la transmission des données.

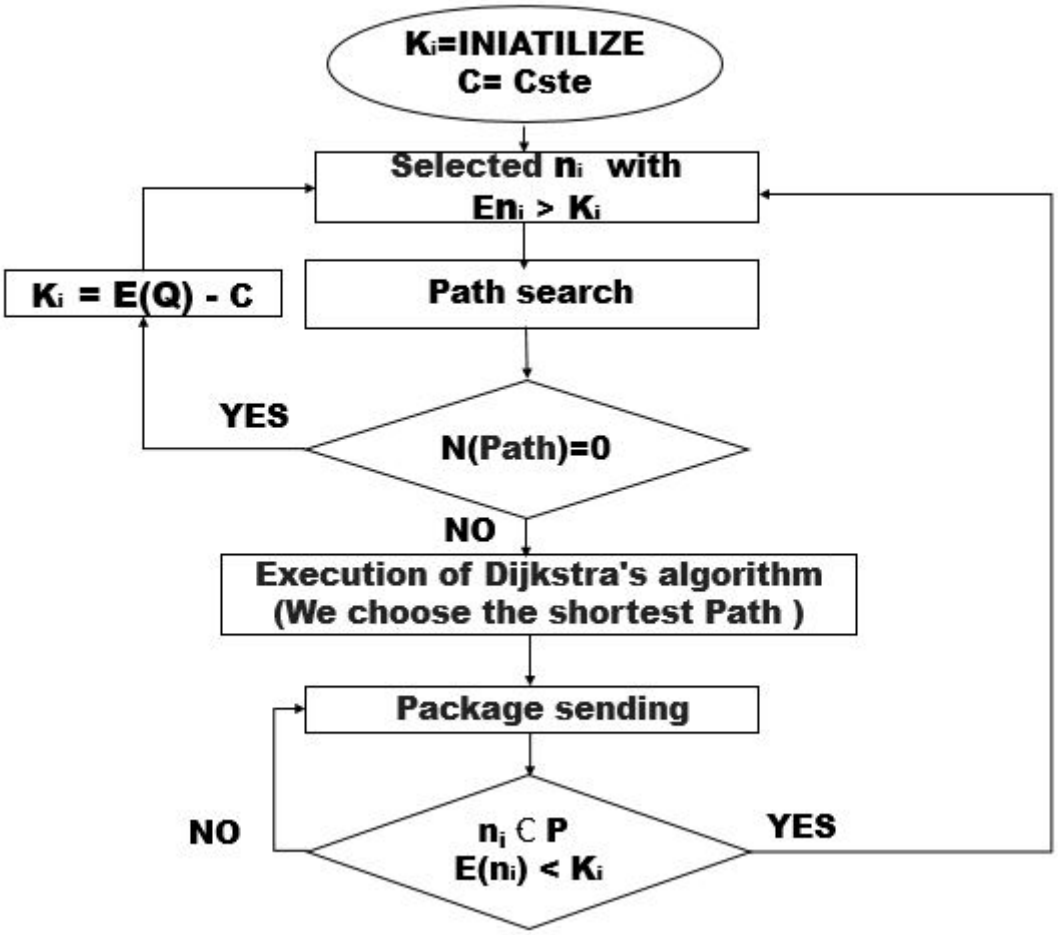

FIGURE 3.2 – Mécanisme de fonctionnement de SDN-WISE

3.3.3 Étude analytique des performances de la solution

Dans cette section, nous menons une étude analytique des performances de la solution proposée. Pour ce faire, nous faisons d'abord une mise en place de l'expérience. Ensuite, nous interprétons les résultats de cette étude analytique.

3.3.3.1 Mise en place de l'expérience

Dans cette partie, nous menons une étude analytique comparant le DEARP et le mode de fonctionnement standard de SDN-WISE. Pour ce faire, nous déployons deux architectures réseau SDN-WISE et mettons en scène les deux modes de routage afin d'évaluer la durée de vie du réseau. Les nœuds de ces deux réseaux tests seront déployés dans les mêmes conditions, mais avec des mécanismes de routage différents. Cela nous permettra d'affirmer lequel des deux mécanismes augmentera la durée de vie du réseau. Notons que dans les Figures 3.3 à 3.8, les nœuds en vert représentent les nœuds sources, les nœuds en bleu représentent les nœuds dont l'énergie résiduelle est supérieure au seuil, les nœuds en rouge sont des nœuds dont l'énergie est inférieure au seuil.

- **Cas du SDN-WISE routage basé sur Dijkstra**

Dans ce premier cas, le routage utilisé est l'algorithme de Dijkstra. Le seul chemin utilisé par le nœud 1 est $1- > 2- > 3- > 4- > sink$. Jusqu'a ce qu'un des nœuds du chemin épuise complètement son énergie.

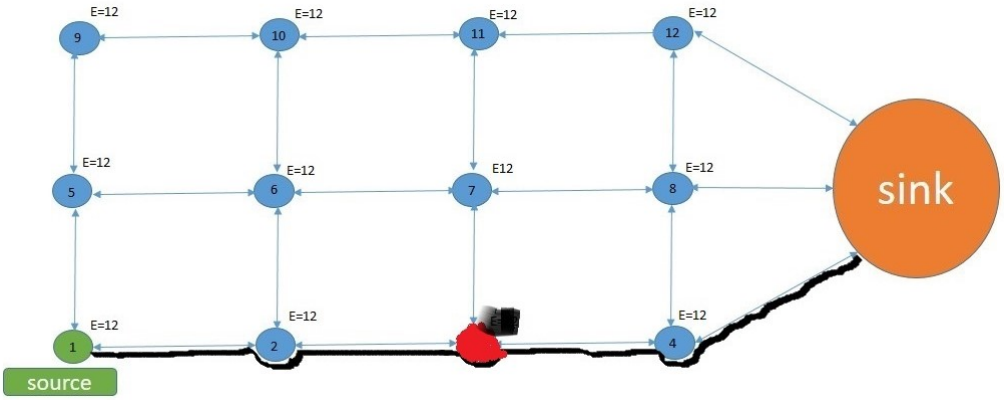

FIGURE 3.3 – Routage basé sur Dijkstra dans SDN-WISE

- **Cas du SDN-WISE routage basé DEARP**

Nous fixons $K_1 = 8$ au début de l'expérience, donc si un chemin choisis à un moment donné a une énergie inférieure à 8, alors on recherche un autre plus court chemin avec une énergie supérieure à 8. A l'étape 1 : $K_1 = 8$ $E(Route) = 12$ $E(Reseau) = 12$ $C = 2$. On choisit la Route $1- > 2- > 3- > 4- > sink$.

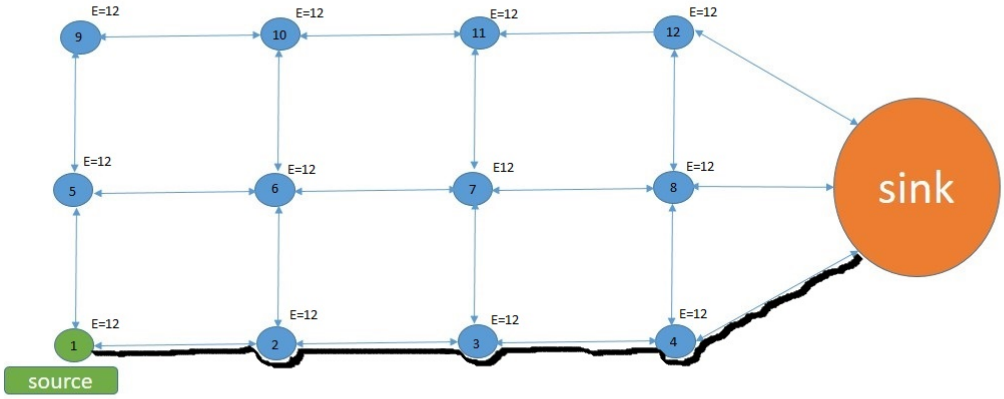

FIGURE 3.4 – Déroulement de DEARP à l'étape 1

Après un certain temps de fonctionnement (transmission de paquets), la valeur de l'énergie du nœud devient inférieure à 8. Le processus de recherche de route reprend. A l'étape 2 : $K_1 = 8$ $E(Route) = 12$ $E(Reseau) = 7$ $C = 2$. On choisit la Route $1- > 5- > 6- > 7- > 8- > sink$.

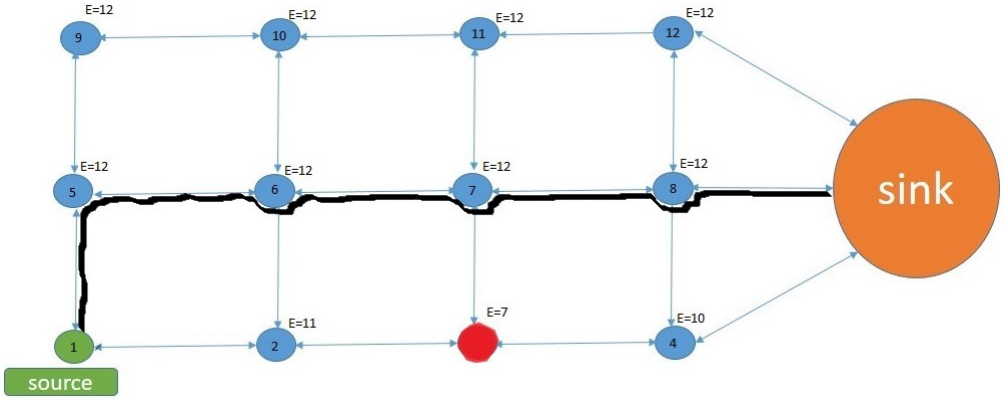

FIGURE 3.5 – Déroulement de DEARP à l'étape 2

A l'étape 3 : $K_1 = 8$ $E(Route) = 11$ $E(Reseau) = 7$ $C = 2$. On choisit la Route $1- > 5- > 9- > 10- > 11- > 12- > sink$.

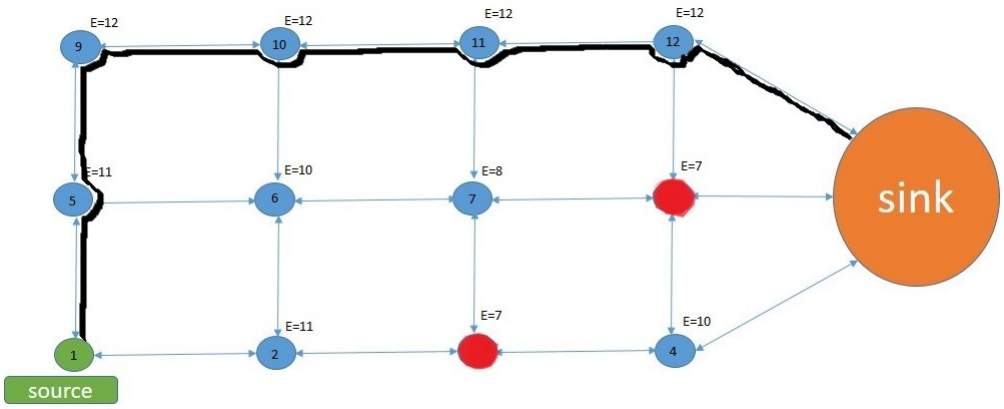

FIGURE 3.6 – Déroulement de DEARP à l'étape 3

A l'étape 4 : $K_1 = 8$ $E(Route) = 7$ $E(Reseau) = 7$ $C = 2$.

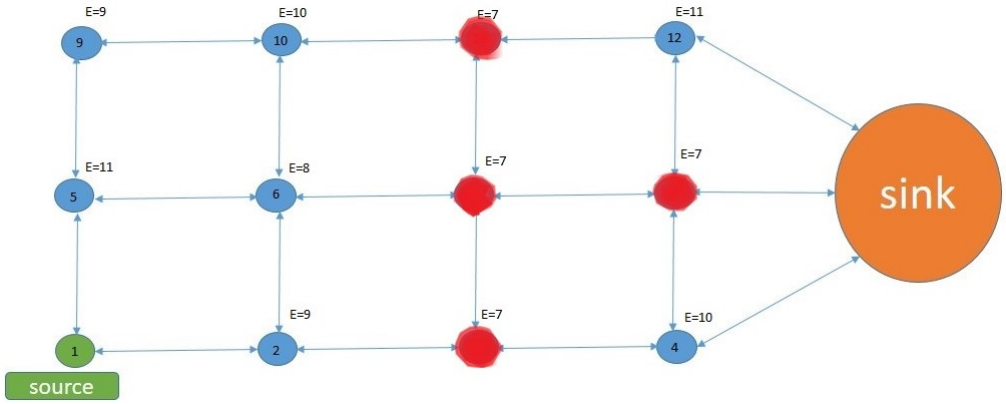

FIGURE 3.7 – Déroulement de DEARP à l'étape 4

Il y'a plus de chemin avec une énergie supérieure au seuil $K = 8$. Donc le contrôleur va décrémenter le seuil d'énergie minimale en lui affectant la valeur de l'énergie du réseau moins une constante C, ainsi on pourra réutiliser tous les chemins de nouveau. $K_1 = E(Reseau)–C$ or $E(Reseau) = 7$ et $C = 2$ et le seuil d'énergie minimale passe de $K_1 = 8$ à $K_2 = 5$ et $E(Route) = 7$.

A l'étape 5 : $K_2 = 5$ $E(Route) = 7$ $E(Reseau) = 4$ $C = 2$.

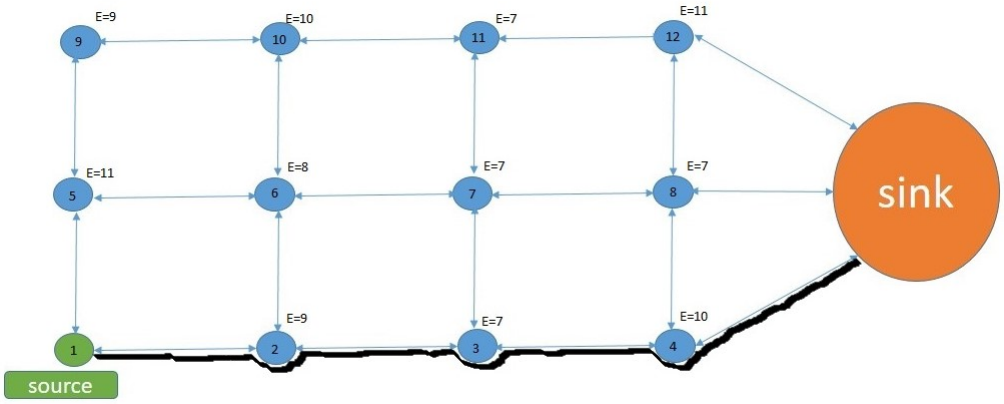

FIGURE 3.8 – Déroulement de DEARP à l'étape 5

Nous voyons que le chemin $1- > 2- > 3- > 4- > sink$ utilisé à l'étape 1 est à nouveau choisi à l'étape 5. Ce processus continue jusqu'à ce que l'un des nœuds du réseau épuise complètement son énergie. Le réseau est alors considéré comme obsolète.

3.3.3.2 Interprétation des résultats de l'expérience

Dans SDN-WISE standard, nous remarquons que lorsqu'on utilise un chemin optimal (plus court chemin) pour l'envoi des données, ce chemin est utilisé jusqu'à ce que le premier nœud du chemin se décharge de toute son énergie, avant de basculer l'envoi des données vers un autre chemin. Dans cette approche, il y a des nœuds qui se déchargent plus rapidement parce qu'ils appartiennent à un plus court chemin alors que d'autres avec une quantité d'énergie élevée ne sont pas utilisés parce qu'ils appartiennent à des chemins plus longs. D'après notre définition de la durée de vie d'un réseau, cette méthode de routage réduit la durée de vie d'un réseau. En effet, si l'un des nœuds du chemin optimal tombe, nous supposons que le système n'est plus efficace et nous concluons à la fin de la mission du réseau.

Dans le routage DEARP appliqué à l'architecture SDN-WISE, les nœuds utilisent aussi le plus court chemin pour l'envoi des données. Cependant lorsqu'un nœud du chemin utilisé pour la transmission commence à s'épuiser de son énergie au-delà d'un seuil, alors un autre chemin est utilisé même si celui-ci est moins optimal en termes de nombre de sauts par rapport au précédent.

3.3.4 Evaluation par simulation des performances de la solution

Dans cette partie, nous évaluons par simulation les performances de notre solution de routage et d'équilibrage de charge DEARP en le comparant au routage est

basé sur l'algorithme de Dijkstra. Pour ce faire, nous allons d'abord décrire l'environnement de simulation, ensuite nous présenterons les résultats obtenus à l'issue de l'analyse de données de la simulation de SDN-WISE et DEARP. Nous discuterons aussi les résultats basés sur les métriques tels que le nombre de sauts, le taux de pertes de paquets et la capacité de chaque protocole à équilibrer la charge du trafic réseau entre les différents nœuds. Nous montrons enfin lequel de ces deux protocoles améliore le plus la durée de vie du réseau.

3.3.4.1 Environnement et paramètres de simulation

Afin d'évaluer les performances de notre proposition, nous avons effectué des simulations. La simulation offre une facilité d'observation des performances de différentes solutions tout en modifiant le fonctionnement. Dans notre cas, nous agissons plus précisément sur le processus de routage.

Pour nos travaux, nous utilisons l'environnement de simulation Contiki Cooja [75]. Contiki Cooja est un simulateur des réseaux de capteurs sans fil. Les protocoles DEARP et SDN-WISE y ont été mis en œuvre et simulés pour évaluer leurs performances. Les résultats sont comparés au protocole de routage original de SDN-WISE basé sur l'algorithme de Dijkstra. Nous avons appliqué la norme 802.15.4 comme protocole de la couche MAC. Quant à l'analyse de la durée de vie du réseau suivant l'approche DEARP, nous avons construit un réseau de 50 nœuds. Nous avons récolté les résultats de simulation à des périodes T=10 jours et ce, durant 50 jours en tout. Le Tableau 3.1 synthétise les différents paramètres de simulation.

Tableau 3.1 – Paramètres de simulation

Paramètre	***Valeur***
Protocole de transport	UDP
Protocole de la couche MAC	802.15.4
Taille du réseau	100m*100m
Taille de chaque paquet de données	100 bytes
Stratégie de déploiement	aléatoire
Débit	448 Kbps
File d'attente des nœuds	50
Nombre de nœuds	50

3.3.4.2 Métriques d'évaluation

Les protocoles examinés sont différents dans leurs manières d'acheminer les paquets de données vers les destinataires. On s'attend ici donc à ce que les paquets prennent des chemins différents pour chacun des protocoles. Nous utilisons comme métriques le nombre moyen de sauts, le taux de pertes de paquets, l'écartype des

énergies résiduelles des nœuds et la durée de vie du réseau pour comparer les deux approches de routage. L'équation (6.1) présente la métrique nombre moyen de sauts.

$$Nombre_{sautsmoyen} = \frac{Somme(hops\_at\_dst(p))}{\#P} \tag{3.1}$$

P est l'ensemble des paquets envoyés ; **hops_at_dst(P) est le nombre de sauts** qu'ont effectués les paquets P avant d'atteindre la destination ; $\#P$ est le nombre de paquets envoyés.

La métrique taux de perte de paquets mesure la fraction des paquets de données qui ont été envoyés sans avoir atteint la destination.

$$TPP = 1 - \frac{nombredepaquetsrecus}{nombredepaquetsenvoyes} \tag{3.2}$$

L'écart-type des énergies résiduelles des nœuds en fonction du temps : cette métrique nous permet d'étudier l'étalement des énergies résiduelles autour de la moyenne. Si l'écart-type est plus petit, alors l'étalement des énergies résiduelles autour de la moyenne est moins important. Dans le cas contraire, l'étalement est plus important.

$$Ecartype(E_i) = \sqrt{\frac{\sum_{i=0}^{n}(E_i - E_{moy})^2}{N}} \tag{3.3}$$

où E_i est l'énergie résiduelle d'un nœud i.

E_{moy} est l'énergie moyenne des nœuds du réseau.

N est le nombre total de nœuds.

La durée de vie du réseau : est l'intervalle de temps qui debute du déploiement du réseau jusqu'au moment où le premier nœud épuise toute son énergie.

3.3.4.3 Présentation des résultats

Une comparaison entre DEARP et SDN-WISE en fonction de la métrique nombre de sauts moyen est représentée sur la Figure 3.9. Sur l'axe des abscisses, nous avons les différents protocoles et sur l'axe des ordonnées nous avons le nombre de sauts moyen. Nous constatons que SDN-WISE a un nombre de sauts moyen égal à 4,301 tandis que DEARP est à 6,947. Nous rappelons que le nombre de sauts maximum est de 12. Nous remarquons que DEARP à un nombre de sauts moyen plus élevé que SDN-WISE.

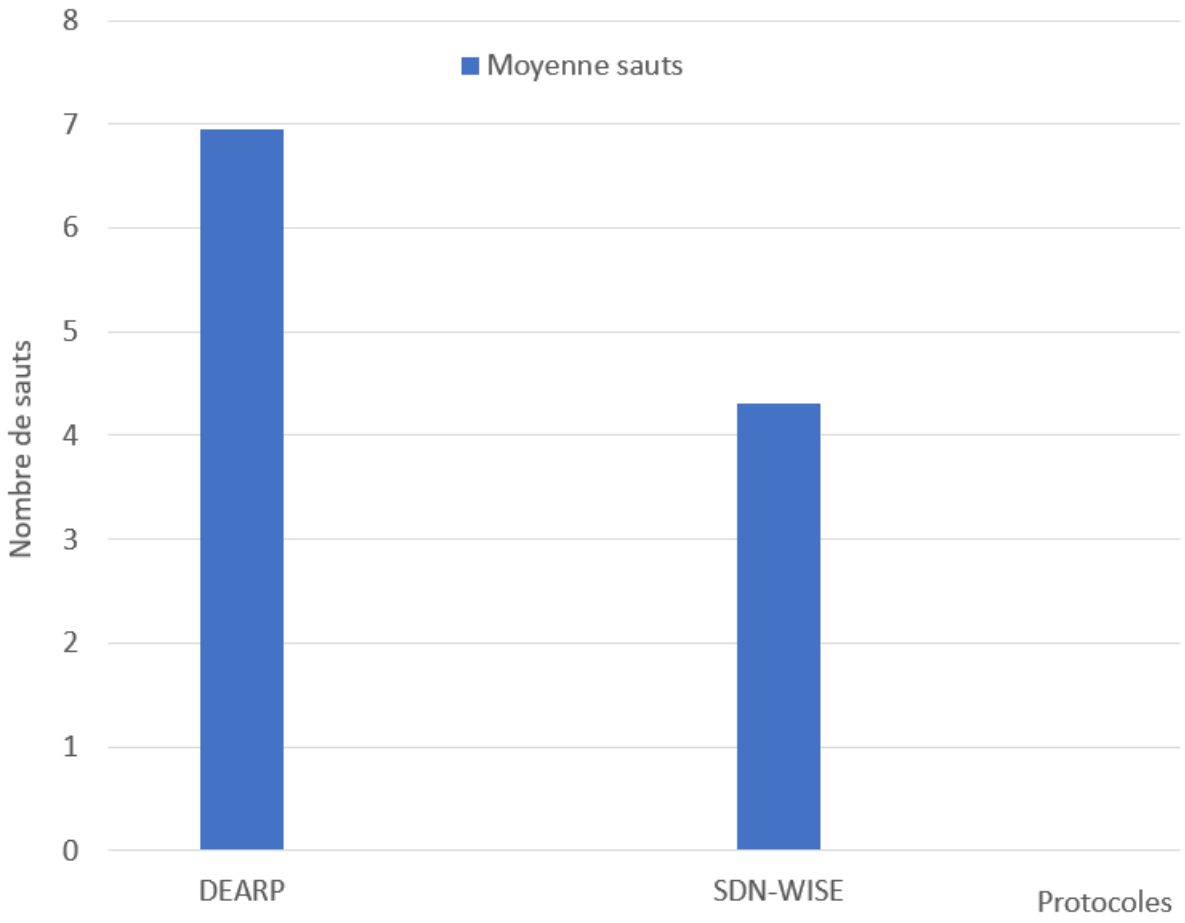

FIGURE 3.9 – Nombre de sauts moyen

La Figure 3.10 nous présente les résultats de mesures du taux de perte de paquets réalisée sur le réseau selon les deux protocoles. DEARP a un taux de perte de paquets légèrement inférieur à celui de SDN-WISE. DEARP est à un taux de 15% tandis que SDN-WISE est à 18%.

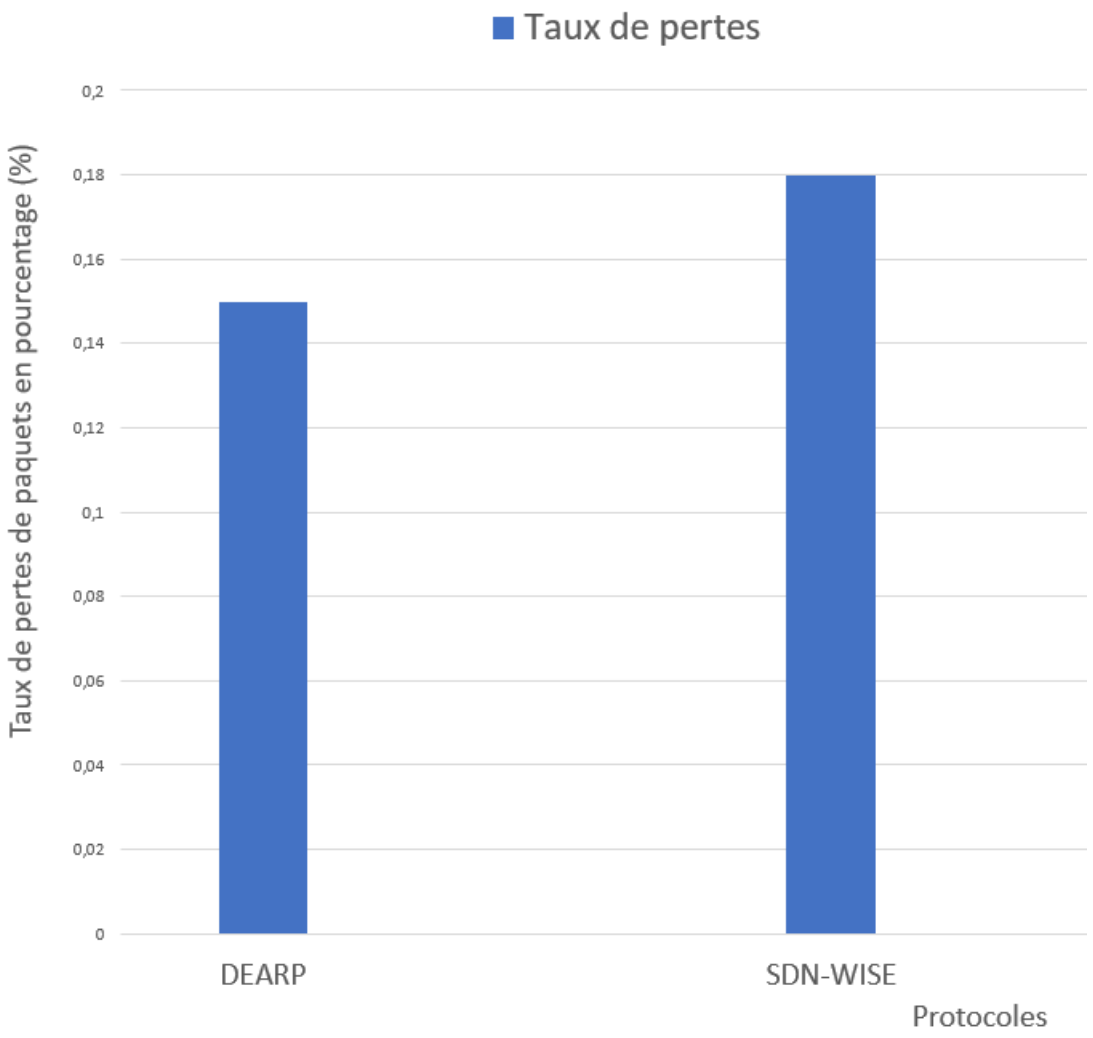

FIGURE 3.10 – Taux de perte de paquets

Afin d'étudier la capacité de chaque protocole à répartir les charges entre les nœuds, nous avons procédé à la récolte périodique de 10 jours de l'énergie résiduelle des nœuds. A la fin de chaque période, nous avons calculé la moyenne des énergies résiduelles et l'écart-type des énergies résiduelles par rapport à la moyenne. A partir de la Figure 3.11, nous constatons que l'écart-type des énergies résiduelles est beaucoup plus petit avec DEARP qu'avec le SDN-WISE et cela durant toutes les étapes du processus de simulation.

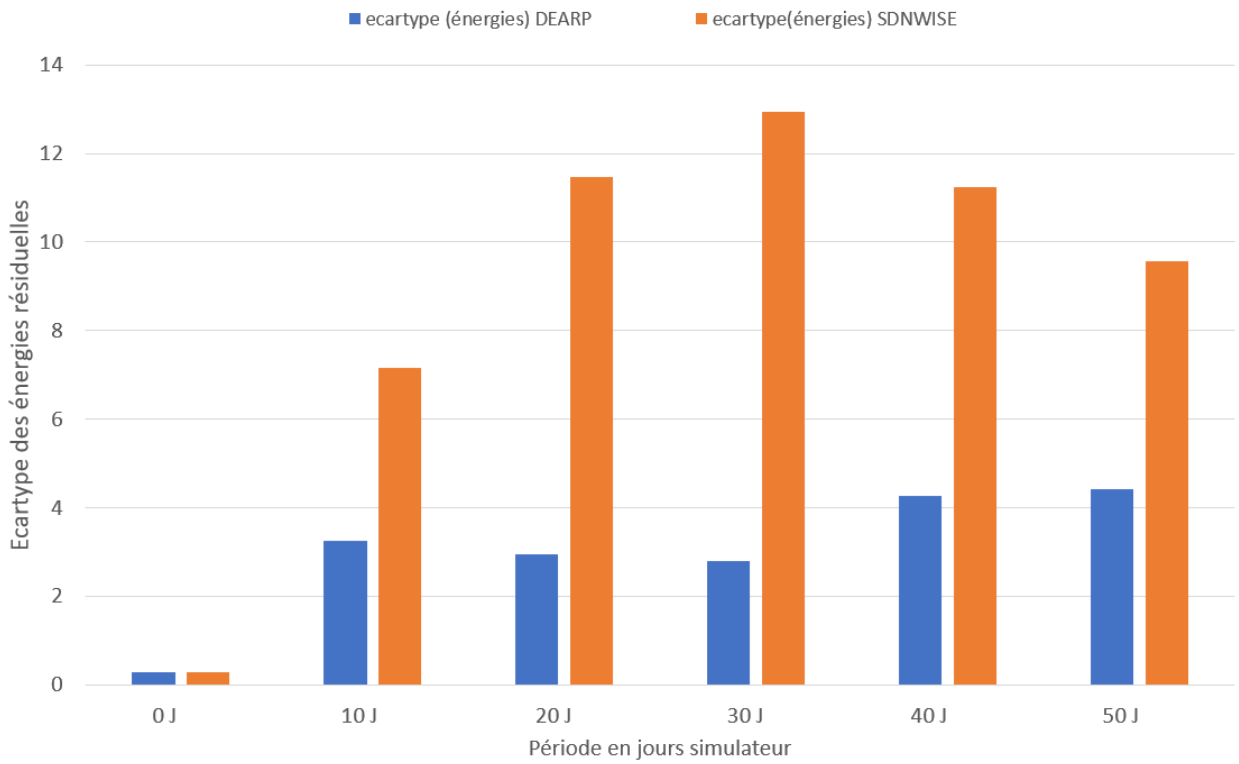

FIGURE 3.11 – Écart-type des énergies résiduelles DEARP et SDN-WISE

Pour évaluer la durée de vie du réseau, nous avons observé l'évolution de l'énergie du réseau durant des périodes de 10 jours successives. Nous rappelons que l'énergie du réseau à un instant T correspond à l'énergie du nœud ayant la plus petite énergie résiduelle. La Figure 3.12 nous présente l'évolution de l'énergie du réseau sur des périodes de 10 jours. Nous remarquons qu'avec SDN-WISE et DEARP, l'énergie du réseau est la même au début des simulations. A partir du 10^e jour nous constatons que la courbe de SDN-WISE décroît plus rapidement que celle de DEARP. Le 30^e jour, l'énergie du réseau SDN-WISE a atteint 0 tandis que celle de DEARP atteint 0 le 50^e jour.

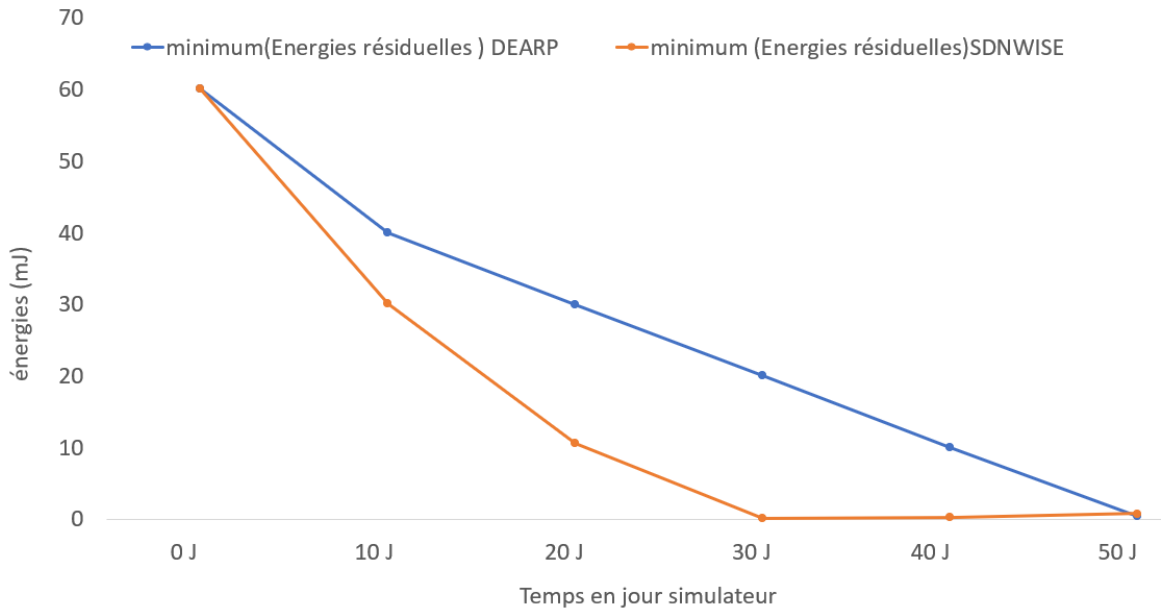

FIGURE 3.12 – Durée de vie des réseaux DEARP et SDN-WISE

3.3.4.4 Analyse et interprétations des résultats

Dans la Figure 3.9, les résultats des simulations montrent une différence nette entre les deux protocoles. Le nombre de sauts moyen du routage SDN-WISE est plus court que celui de DEARP, qui a un nombre de sauts moyen élevé. Cela s'explique par le fait que DEARP évite d'envoyer des paquets sur les nœuds dont l'énergie résiduelle est inférieure à un seuil, ainsi que les nœuds qui traitent beaucoup de trafic. Par conséquent, les chemins choisis pour envoyer les paquets peuvent être plus longs que le plus court chemin.

La Figure 3.10 montre que SDN-WISE (RSSI) et DEARP présentent un taux de perte de paquets presque identique. Cependant, DEARP réduit légèrement le taux de perte de paquets. Cela est dû au fait que DEARP évite de trop surcharger les nœuds en empruntant un chemin plus long, mais avec des nœuds moins surchargés.

La Figure 3.11 présente l'écart-type des énergies résiduelles par rapport à la moyenne. Nous remarquons que l'écart-type des énergies résiduelles de DEARP par rapport à la moyenne est beaucoup plus petit par rapport à SDN-WISE. Ce qui veut dire que l'étalage des énergies résiduelles des nœuds par rapport à la moyenne est moins important avec DEARP. Donc il y a une meilleure répartition de la consommation d'énergie entre les différents nœuds. Cela s'explique par le fait qu'avec DEARP, tous les nœuds participent à l'envoi des données. De ce fait, ils se déchargent presque au même rythme. C'est ce qui entraîne un écart-type des énergies résiduelles faible dans DEARP. Alors que dans SDN-WISE, on observe une hausse de l'écart-type des énergies résiduelles. Ce qui explique un étalage assez important des énergies résiduelles au tour de la moyenne. Il y a des nœuds qui sont beaucoup plus sollicités

pour l'envoi et le routage des données, notamment les nœuds centraux ou les nœuds se trouvant sur le plus court chemin. De ce fait, ils se déchargent rapidement de leurs énergies pendant que d'autres sont sous exploités et ont toujours de grandes quantités d'énergie. Nous déduisons que DEARP permet une meilleure répartition de charge entre les différents nœuds par rapport à SDN-WISE.

La Figure 3.12, nous montre l'évolution de la durée de vie du réseau. Au déploiement du réseau, nous avons constaté que DEARP et SDN-WISE avaient presque le même niveau d'énergie. Cela s'explique par le fait qu'au déploiement DEARP tout comme SDN-WISE utilisent le plus court chemin pour atteindre la destination. Il n'y a pas une grande différence d'énergie réseau entre les deux protocoles. Après 10 à 20 jours de simulations, nous constatons une baisse rapide de l'énergie réseau avec SDN-WISE comparativement à DEARP. Finalement, au 30^e jour, nous constatons que l'énergie du réseau SDN-WISE atteint zéro, c'est à dire épuise sa durée de vie. Tandis qu'avec DEARP, la durée de vie du réseau est épuisée au 50^e jour. De ce fait, nous déduisons que DEARP augmente la durée de vie du réseau.

3.4 Notre approche CMFR-CMQ

Nous proposons une nouvelle approche de gestion de l'énergie appelée *Congestion Management Flow Request Control Message Quenching* (CMFR-CMQ). CMFR-CMQ cherche à résoudre le problème 2 aborder dans la section **??**. Nous rappelons que cette problématique est due à la production des doublons des messages de types "*Request*" dans SDN-WISE lorsqu'un nœud n'a pas de règles pour une suite de donnée ayant la même destination et aussi lorsque la mémoire du nœud est saturée dans SDN-WISE. Il n'y'a pas de processus pour éviter que le nœud continue de recevoir des paquets à retransmettre. CMFR-CMQ est l'approche proposée pour résoudre cette problématique. Dans les sous-sections suivantes, nous définissons d'abord les variables pour faciliter la compréhension de cette solution, puis nous décrivons l'algorithme et enfin nous dressons quelques résultats analytiques de performance de cette approche proposée.

3.4.1 Mode de fonctionnement de CMFR-CMQ

Avant d'aborder plus en détail notre solution, il est nécessaire de définir les différentes variables requises :

- Capacité C_i : c'est la taille en octets de la mémoire tampon du nœud i.
- Quantité de paquets de données QP_i : elle désigne la quantité de paquets de données en attente de traitement dans la mémoire tampon d'un nœud i.
- Taux de congestion T_i : il est exprimé en pourcentage et correspond au quotient de la quantité de paquets de données en attente dans la mémoire tampon

d'un nœud i, sur sa capacité de la mémoire le tout multiplié par cent (100).

$$T_i = \frac{QP_i}{C_i} * 100 \tag{3.4}$$

- État de congestion (E_i) : c'est une valeur booléenne qui contrôle le niveau de saturation. Cette information permet à un nœud de signaler au contrôleur s'il est apte à recevoir d'autres paquets ou non. E_i est défini par défaut à 0 et passe à 1 lorsque le taux de congestion (T) est supérieur ou égal à 80 % et reprend la valeur 0 si T devient inférieur ou égale 20 %.
- Couple Source/Destination SD_i : il correspond au couple adresse source et destination du paquet en cours de traitement.
- Liste Sources/Destinations (LSD) : il s'agit de la liste de couples d'adresses sources et destinations stockées dans la mémoire dynamique du nœud à des fins de vérification.

L'algorithme que nous proposons consiste non seulement à réduire le nombre de doublons des paquets du type « Request », mais aussi à éviter la congestion des nœuds due aux paquets mis en attente de traitement. Pour ce faire, les différentes étapes sont d'abord conditionnées par la vérification de l'état (E) de congestion actuelle du nœud, c'est-à-dire que le contrôleur détermine la valeur E du nœud en calculant son taux de congestion (T) grâce aux informations (QP, C_i) obtenues par le biais du message « Report ».

a) Un nœud en état de congestion ne peut recevoir que des règles de flux venant du contrôleur. Ces règles de flux sont insérées dans la table WISE du nœud. Cette mise à jour de sa table WISE permet d'enclencher le traitement des paquets de données concernés mis en attente dans la mémoire tampon.

b) Un nœud non congestionné éffectue plusieurs mécanismes de vérification. A la réception d'un paquet, on vérifie d'abord le type de paquet afin de déterminer la démarche à suivre :

   b-1) dans le cas où le paquet correspond à une règle de flux du contrôleur, il met automatiquement à jour la table WISE du nœud à des fins de traitement.

   b-2) s'il s'agit d'un paquet de données, alors le nœud procède à la vérification de correspondance dans sa table WISE. Si une correspondance est trouvée, l'action adéquate est exécutée. Sinon, le nœud vérifie l'adresse source/destination du paquet en cours de traitement :

   - si l'adresse source/destinations appartient déjà à la liste de couples d'adresses sources/destination, alors le paquet est mis dans une file d'attente de traitement dans la mémoire tampon.
   - dans le cas contraire, la nouvelle adresse source/destination est insérée dans la liste de couples d'adresses sources/destinations, puis un mes-

sage du type « Request » est envoyé au contrôleur pour une demande de règles de flux afin de traiter le paquet concerné.

Les différentes étapes de notre algorithme peuvent être résumées dans la Figure 3.13.

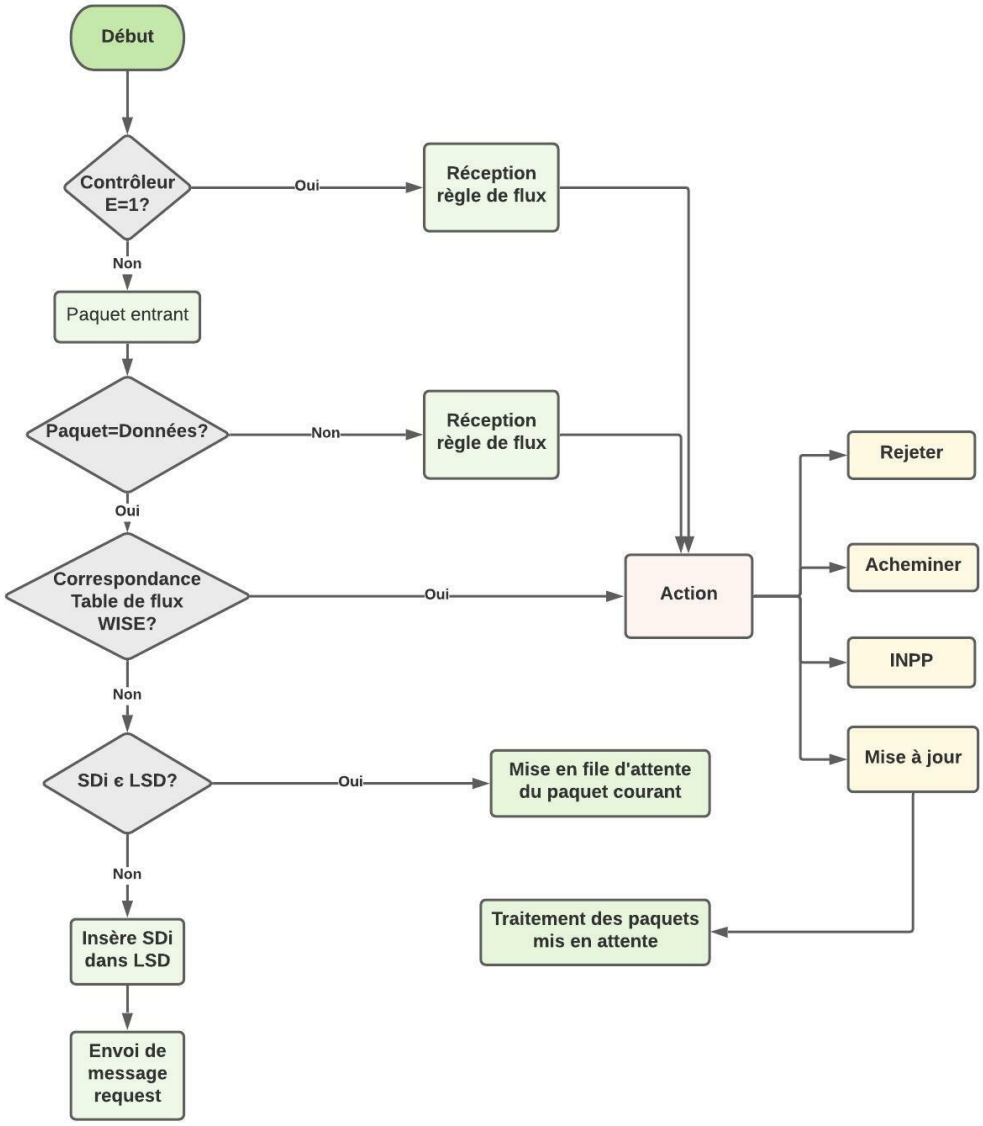

FIGURE 3.13 – Algorigramme du CMFR-CMQ

3.4.2 Étude analytique des performances de CMFR-CMQ

Notre étude analytique consiste dans un premier temps à définir le cadre d'expérimentation. Nous présentons ensuite les résultats de simulation. Nous menons enfin une interprétation des résultats obtenus.

3.4.2.1 Dispositif expérimental

Pour cette étude expérimentale, nous considérons trois nœuds capteurs A, B et C. Nous supposons que la mémoire tampon d'un nœud ne peut contenir que la taille de cinq paquets au maximum. Aussi, cette expérience est faite dans un contexte où un nœud envoie un message du type Request au contrôleur et n'ayant pas encore reçu de réponse. Les paquets du nœud A ont la possibilité de passer par le nœud B ou ou le nœud C pour atteindre le nœud puits. Nous décrivons ce scénario de communication en fonction de trois mécanismes de gestion des paquets : SDN-WISE, FR-CMQ et CMFR-CMQ. Nous supposons que le contrôleur ordonne au nœud A l'envoi de vingt paquets de données en passant par le nœud B.

Tableau 3.2 – Résumé des scénarios de communication

Mécanisme	Nœuds impliqués	Tampon (max)	Scénario	Paquets envoyés
SDN-WISE	A, B, C	5	A envoie 20 paquets via B au puits	20
FR-CMQ	A, B, C	5	A envoie 20 paquets via B au puits	20
CMFR-CMQ	A, B, C	5	A envoie 20 paquets via B au puits	20

Mode opératoire du SDN-WISE

Après la réception d'un paquet provenant du nœud A, le nœud B formule un message du type Request à l'endroit du contrôleur et place le paquet dans sa mémoire tampon en attendant des instructions. Ce processus est répété jusqu'à ce que le nombre de paquets dans la mémoire tampon atteigne cinq, d'où sa saturation. Le nœud A envoie vingt paquets au nœud B, cinq messages Request seraient produits, cinq paquets seraient stockés dans sa mémoire et quinze paquets seraient perdus si toute fois le contrôleur tarde à répondre pour que le nœud B puisse faire la redirection des messages et libérer sa mémoire.

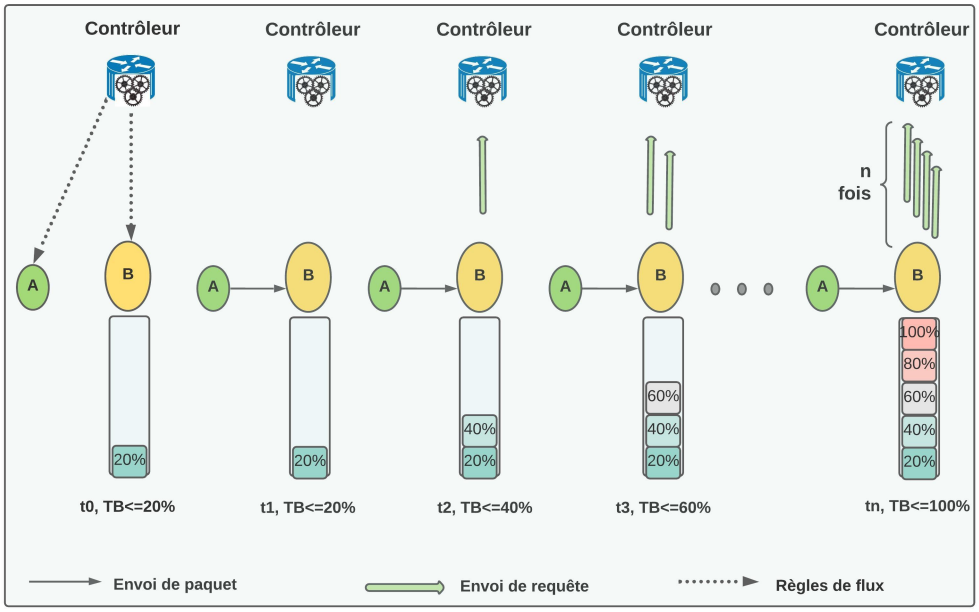

FIGURE 3.14 – Traitement de paquets dans SDN-WISE

Mode opératoire du FR-CMQ

Contrairement à SDN-WISE, dans FR-CMQ lorsque le nœud B reçoit des paquets de données ayant les mêmes sources/destinations, il produit un seul message du type Request. Ensuite, il stocke les paquets dans sa mémoire tampon jusqu'à ce qu'elle soit saturée. Par conséquent, si le nœud A envoie vingt paquets au nœud B, nous obtenons un seul message du type Request, cinq paquets seraient stockés en mémoire et quinze paquets seraient rejetés.

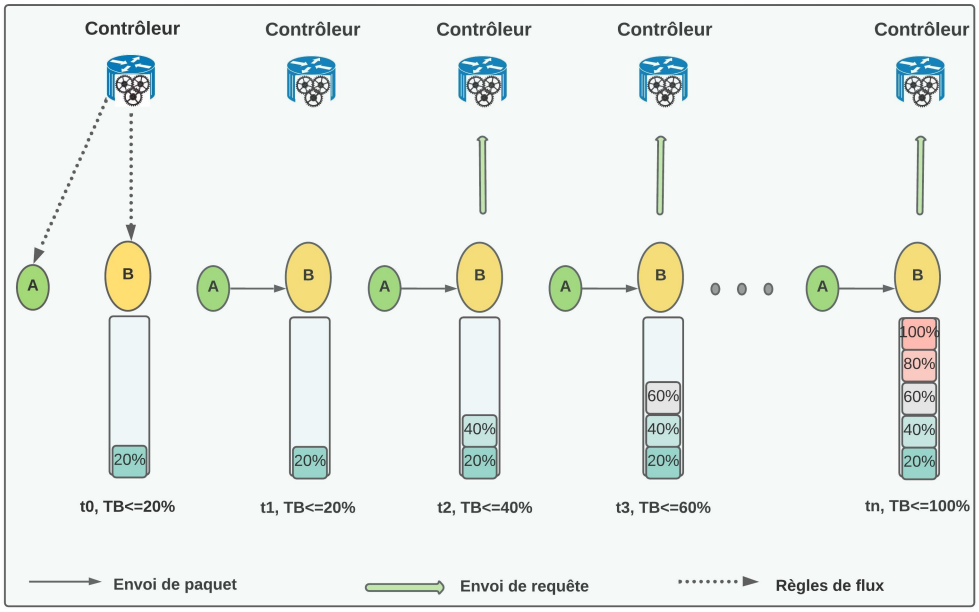

FIGURE 3.15 – Traitement de paquets dans FR-CMQ

Mode opératoire du CMFR-CMQ

Comme dans FR-CMQ, lorsque le nœud B reçoit plusieurs paquets de données provenant du nœud A, il envoie un seul message du type Request. Cependant, CMFR-CMQ prend en compte le niveau de congestion des nœuds. Ainsi, si le nœud A envoie vingt paquets au nœud B, un seul message du type Request est produit, quatre paquets seraient stockés en mémoire et seize paquets seraient acheminés vers un autre nœud non congestionné (C dans notre cas). Cela est possible, car le contrôleur a une vue globale du réseau et est conscient de l'état de la mémoire tampon de tous les nœuds. Ces états lui sont communiqués périodiquement grâce aux messages "report" qu'il reçoit fréquemment des nœuds. C'est ainsi qu'il pourra instruire le nœud A d'arrêter d'envoyer les messages au nœud B au profit du nœud C.

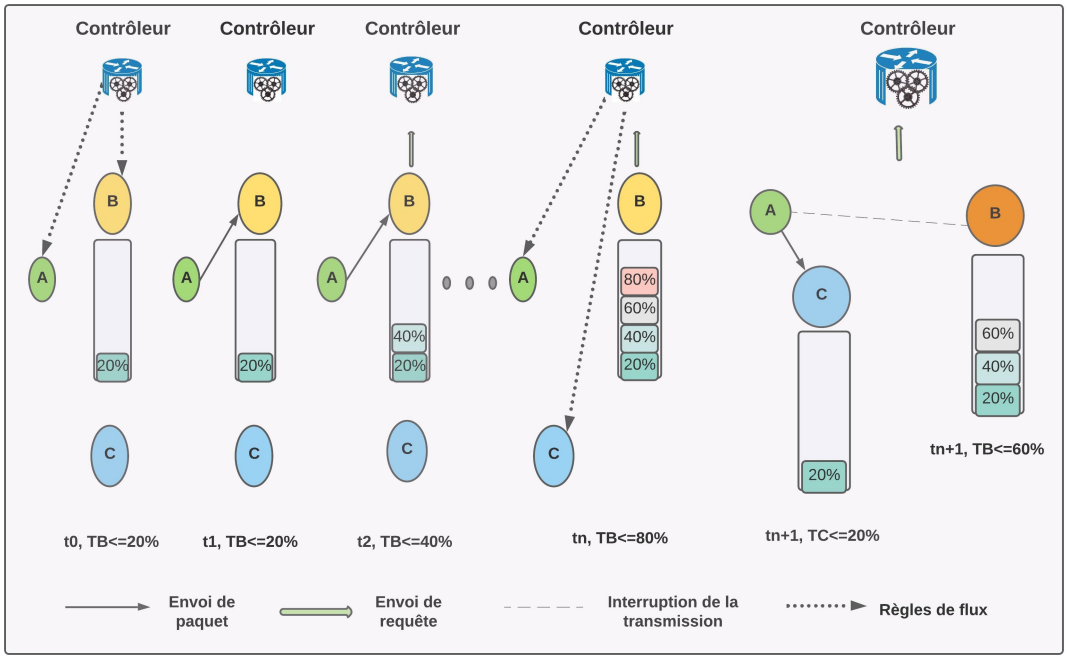

FIGURE 3.16 – Traitement de paquets dans CMFR-CMQ

3.4.2.2 Résultats

Après cette étude analytique, nous avons recueilli des résultats présentés sous forme de graphique. Ces résultats sont relatifs au nombre de paquets reçus, au nombre de paquets perdus et au taux de perte suite à l'envoi de 20 paquets de données du nœud A vers le nœud B. Nous tenons à préciser que ces résultats sont observés au niveau du nœud B.

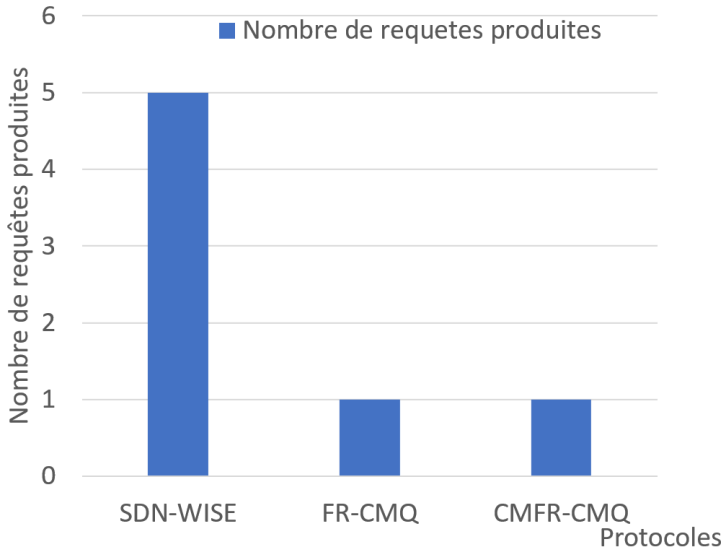

FIGURE 3.17 – Nombre de messages RReq produits

A la Figure 3.17, nous avons en abscisse des protocoles (SDN-WISE, FR-CMQ et CMFR-CMQ) et en ordonnée le nombre de messages de demandes de routes envoyés par le nœud B au contrôleur. Nous remarquons que FR-CMQ et CMFR-CMQ produisent moins de messages de demandes de routes par rapport à SDN-WISE.

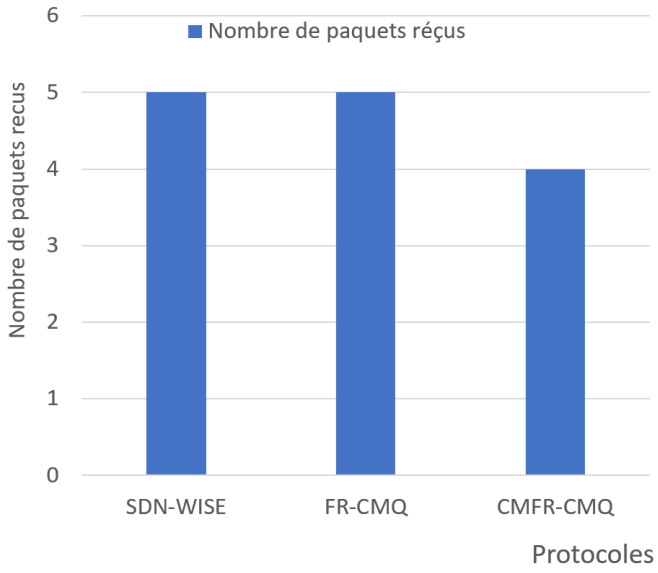

FIGURE 3.18 – Nombre de paquets reçus

A la Figure 3.18, nous avons en abscisse des protocoles(SDN-WISE, FR-CMQ et CMFR-CMQ) et en ordonnée le nombre de paquets reçus. Nous constatons que le nombre de paquets reçus par le nœud B est le même avec SDN-WISE et FR-CMQ. Cependant, le nombre de paquets reçus est légèrement inférieur avec CMFR-CMQ.

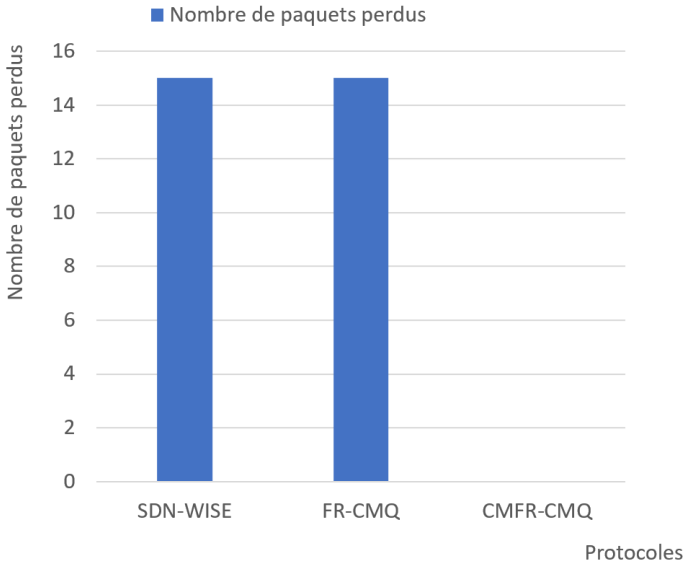

FIGURE 3.19 – Nombre de paquets perdus

A la Figure 3.19, nous avons en abscisse les protocoles (SDN-WISE, FR-CMQ et CMFR-CMQ) et en ordonnée le nombre de paquets perdus. Nous constatons que le nombre de paquets perdus par le nœud B est le même avec SDN-WISE et FR-CMQ. Cependant, le nombre de paquets perdus est quasiment nul avec CMFR-CMQ.

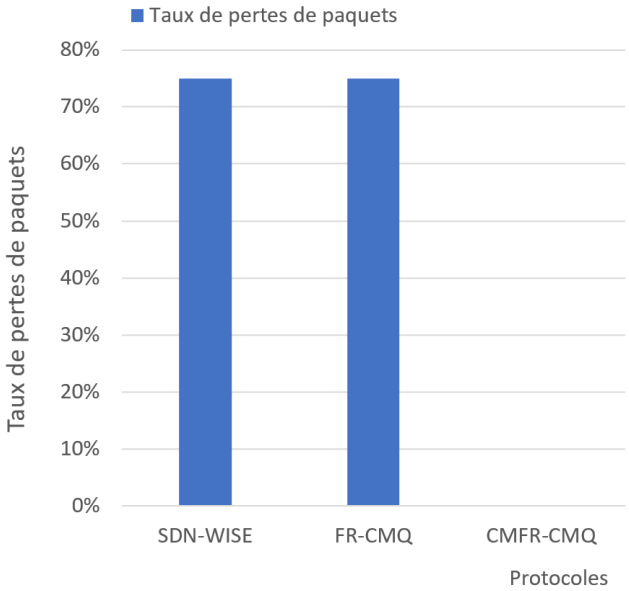

FIGURE 3.20 – Taux de perte de paquets

A la Figure 3.20, nous avons en abscisse les protocoles(SDN-WISE, FR-CMQ et CMFR-CMQ) et en ordonnée le taux de pertes de paquets. Nous constatons que le taux de pertes par le nœud B est le même avec SDN-WISE et FR-CMQ. Cependant, le taux de pertes est quasiment nul avec CMFR-CMQ.

3.4.2.3 Discussion

Au vu des résultats de la Figure 3.17, nous constatons dans que SDN-WISE produit plus de messages de type "*Request*" par rapport à FR-CMQ et à CMFR-CMQ. Ces messages "*Request*" étant sources de consommation supplémentaire d'énergie dans le réseau, nous concluons que FR-CMQ et CMFR-CMQ pourraient réduire la consommation d'énergie en évitant la production des messages "*Request*" en doublon. Dans les résultats présentés à la Figure 3.18, CMFR-CMQ présente un nombre de paquets reçus légèrement inférieur à celui de FR-CMQ et de SDN-WISE. Cela s'explique par le fait que dans CMFR-CMQ, il y a un mécanisme de contrôle de congestion. Quand le nœud B a occupé 80 % de sa mémoire tampon, le contrôleur a ordonné au nœud A de passer par un autre nœud non saturé, c'est-à-dire un nœud dont l'occupation de la mémoire tampon n'a pas atteint les 80 %. Ainsi, CMFR-CMQ évite la congestion des nœuds due à la saturation de leur mémoire tampon et réduit ainsi leur consommation en énergie.

Aussi, nous remarquons que cette saturation de la mémoire tampon des nœuds observé avec FR-CMQ et SDN-WISE pourrait engendrer des pertes de paquets. Nous pouvons observer cela à la Figure 3.20. En somme, CMFR-CMQ évite la création des messages "*Request*" en doublon et évite la saturation de la mémoire tampon des nœuds. Cela réduit le taux de pertes de paquets et pourrait améliorer la durée de vie du réseau. Notons que la perte de paquets estimée dans notre étude est liée uniquement à la congestion des nœuds et nous ne faisons pas cas de celle résultant de la qualité des liens.

3.5 Conclusion

Dans ce chapitre, nous avons présenté nos différentes propositions à savoir DEARP et CMFR-CMQ. A travers des résultats de simulation, nous avons pu montrer que la solution DEARP permettait une amélioration de la répartition de charge et la durée de vie du réseau comparativement à SDN-WISE standard. Aussi, nous avons décrit de façon analytique en nous basant sur le fonctionnement de l'algorithme que CMFR-CMQ est une solution qui supprime les doublons des messages "*Request*" dans SDN-WISE, Aussi elle libère les nœuds capteurs des saturations de leurs mémoires tampons.

Conclusion générale

Synthèse

Le réseau de capteurs a connu un essor considérable avec la cinquième génération de l'informatique. C'est une technologie de réseau qui s'applique à de nombreux domaines [76] [77]. Il permet une interaction entre le monde physique et le monde virtuel. Ce qui permet d'automatiser de nombreuses tâches et facilite la vie à ses usagers. Les nœuds constituants les réseaux de capteurs fonctionnent de façons autonomes et indépendantes du réseau électrique. Cependant, un des majeurs problématiques des réseaux de capteurs est qu'ils ont une source d'alimentation limitée. Ces nœuds capteurs fonctionnent généralement sur des batteries et deviennent inutilisables lorsqu'ils se déchargent complètement de leurs énergies. Ce qui constitue un frein au deploiement et pour une utilisation à long terme des WSN.

Afin de contribuer à la resolution de cette problématique énergetique, nous avons dans nos travaux, étudié de façon générale les réseaux de capteurs sans fil. Ensuite, nous avons introduit le concept SDN et son intégration dans les WSN. Aussi, nous avons étudié les approches trouvées dans la littérature portant sur l'optimisation de la consommation d'énergie dans les WSN et SDWSN. Dans un premier temps, nous abordons les problèmes d'optimisation de l'énergie dans les WSN à regroupement par "cluster". Dans un second temps, nous nous sommes intéréssés aux problématiques liées à la gestion d'énergie dans les réseaux de capteurs définis par logiciel. Nous avons proposé des solutions optimisant la consommation d'énergie dans les WSN. Ces solutions sont implémentées au niveau du mécanisme de routage ou lors de la construction de la topologie du réseau.

La première contribution proposée dans BMN-LEACH-S [78] est basée sur le protocole LEACH. BMN-LEACH-S a pour but de réduire la consommation d'énergie dans les réseaux à regroupement par cluster. Pour y parvenir, elle va permettre dans un premier temps d'équilibrer le nombre de nœuds constituant les clusters. Elle se base sur un système à logique floue qui prend en paramètres le nombre de nœuds déjà intégrés dans le cluster et la puissance du signal entre le nœud et la tête de cluster potentiel pour calculer un coût. Le nœud intégrera le cluster dont le coût avec la tête de cluster est le plus élevé. Aussi, BMN-LEACH-S permet une élection effi-

cace des têtes de cluster afin d'équilibrer la consommation d'énergie dans le réseau. Cette élection de tête de cluster se fait à tour de rôle. Cela permet d'équilibrer la consommation d'énergie entre les nœuds du cluster. Une étude analytique permet de montrer que notre approche diminue le nombre d'instabilités dues au changement fréquent des têtes de cluster.

La deuxième contribution appelée DEARP [17] est une solution de routage offrant une équité dans la répartition de charge entre les nœuds du réseau en se basant sur l'algorithme de Dijkstra de SDN-WISE. Dans cette solution, nous avons défini un seuil de contrôle des énergies restantes. Si l'énergie résiduelle d'un nœud du chemin de routage se trouve en dessous du seuil, alors l'algorithme de Dijkstra est lancé pour rechercher un nouveau chemin dont les nœuds auront une quantité d'énergie supérieure au seuil. Dans le cas où il n'y a plus de chemin avec une quantité d'énergie supérieure au seuil, alors la valeur du seuil est dimuniée de sorte à tenir compte de tous les chemins du réseau à nouveau. Ainsi, les nœuds se déchargent plus ou moins harmoniéusement et de façon progressive. Nous avons ainsi réussi à améliorer la durée de vie du réseau.

Dans la troisième contribution, nous avons proposé CMFR-CMQ [18] qui est une solution au problème de production en doubles des messages "*Request*" dans le SDN-WISE. Dans l'algorithme de CMFR-CMQ, une fois que plusieurs paquets arrivent au niveau d'un nœud pour la même destination, un seul message *Resquet* est envoyé au contrôleur pour tous les autres. Ainsi, cela évite la production de messages "Request" en plusieurs exemplaires à l'endroit du contrôleur pour la même demande. De plus, lorsque la mémoire tampon d'un nœud est saturée, le contrôleur informe les autres nœuds, d'arrêter de passer par ce dernier pour la retransmission pour un temps afin qu'il puisse évacuer les instances en cours et libérer sa mémoire. On évite ainsi des congestions et des pertes de paquets sur le réseau. Cela pourrait améliorer la durée de vie du réseau, un résultat à confirmer ses résultats par des tests en réel.

Perspectives

Pour l'évaluation de la contribution BMN-LEACH-S, nous avons mené une étude analytique. Nous sommes conscients que cette étude analytique uniquement n'est pas totalement suffisante pour confirmer la performance de notre solution. C'est pour cette raison dans nos travaux futurs nous prévoyons conduire par des tests en réel.

Dans la contribution DEARP, nous avons évalué notre solution selon les métriques "nombre de sauts", "taux de perte de paquets", "capacité à repartir la charge" et "durée de vie du réseau". Cependant, nous n'avons pas évalué les protocoles selon le temps de livraison des paquets. Dans la suite de nos travaux, nous évaluerons cette métrique afin d'adapter l'utilisation de notre protocole en fonction de son contexte selon que le temps de livraison importe ou pas.

Dans la solution CMFR-CMQ, tout comme dans la solution BMN-LEACH-S, l'évaluation des performances a été faite de façon analytique. Il nous faut poursuivre avec des tests en réel pour confirmer ces résultats.

Aussi, nous aborderons les problèmes de sécurité qui pourraient se poser lors de la conception de ces mécanismes. En effet, dans l'architecture SDN-WISE, les nœuds envoient des messages "*report*" au contrôleur pour l'informer de leur état (niveau d'énergie, nombre de voisins, etc.) afin que celui-ci ait permanemment une vue globale de l'état du réseau pour permettre aux applications de mieux interagir avec ces nœuds sous-jacents. Un attaquant pourrait modifier les informations contenues dans le message "*report*" pour perturber le bon fonctionnement du réseau. Dans nos travaux futurs, nous verrons comment sécuriser ces messages "*report*" afin d'assurer un transfert fiable des informations sur l'état des nœuds du réseau au contrôleur.

Liste des publications personnelles

1. Boulou, M., Yélémou, T., Ouattara, S.I., Sanou, B.H.M. (2022). LEACH-S Enhancement to Ensure WSN Stability. In 13th EAI International Conference on Ad Hoc Networks. ADHOCNETS TridentCom 2021. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 428. pp 99–113. Springer, Cham.
   DOI : http ://dx.doi.org/10.1007/978-3-030-98005-4_8
2. Mahamadi Boulou, Tiguiane Yélémou, Doda Afoussatou Rollande Sanou, Hamadoun Tall, "DEARP : Dynamic Energy Aware Routing Protocol for Wireless Sensor Network", 2020 IEEE 2nd International Conference on Smart Cities and Communities (SCCIC) DOI : 10.1109/SCCIC51516.2020.9377331, December 2020, Electronic ISBN :978-1-7281-9684-8, ISBN :978-1-7281-9685-5. https ://ieeexplore.ieee.org/document/9377331, indexé dans scopus, IEExplorer, indexé dans scopus : https ://www.scopus.com/authid/detail.uri ?authorId=43761783400
3. Go, A., Boulou, M., Yélémou, T., Tall, H. (2022). CMFR-CMQ : Congestion Management and Control Message Quenching Based on Flow Setup Requests in SDN-WISE. In : Sheikh, Y.H., Rai, I.A., Bakar, A.D. (eds) e-Infrastructure and e-Services for Developing Countries. AFRICOMM 2021. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 443. Springer, Cham. https ://doi.org/10.1007/978-3-031-06374-9_23
4. Mahamadi Boulou, Tiguiane Yélémou, Pegd-Winde Justin Kouraogo, Hatem Ben Sta, "A SDN based approach to improve energy consumption in wireless ad hoc network", 5th IEEE International Smart Cities Conference, ISC2 2019, 2019, pp. 50–53, 9071739, DOI : 10.1109/ISC246665.2019.9071739, Electronic ISSN : 2687-8860, ISSN : 2687-8852, IEExplorer https ://ieeexplore.ieee.org /document/9071739, scopus :https ://www.scopus.com/record/display.uri ?eid=2-2.0-85084683043&origin=inward&txGid=e0ef9e0744706a1952c4f1ab01c7ae37
5. Boulou, Mahamadi & Yélémou, Tiguiane & Go, Achille & Tall, Hamadoun. (2022). Energy management techniques in Software-Define Wireless Sensor Network. In Proceedings of the 4th edition of the Computer Science Re-

search Days, JRI 2021, 11-13 November 2021, Bobo-Dioulasso, Burkina Faso. http ://dx.doi.org/10.4108/eai.11-11-2021.2317974

6. Doda Afoussatou Rollande, Tiguiane Yélémou, Hamadoun Tall, Mahamadi Boulou, "Taking into Account Children Accurate Weights during Parent Selection Process in RPL to Extend WSN Lifetime", The Seventeenth International Conference on Wireless and Mobile Communications ICWMC 2021), August 2021, Nice, France. Proceedings IARIA, 2021, ISSN : 2308-4219, ISBN : 978-1-61208-878-5 page 60-64, ThinkMind(TM) Digital Library : https ://www.thinkmind.org/index.php ?
view=article&articleid=icwmc_2021_2_70_20032

Bibliographie

[1] J. Durand, "Le SDN pour les nuls," *JRES 2015-Montpellier*, pp. 1–12, 2015.

[2] E. Haleplidis, J. Hadi Salim, J. M. Halpern, S. Hares, K. Pentikousis, K. Ogawa, W. Weiming, S. Denazis, and O. Koufopavlou, "Network Programmability with ForCES," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 3, pp. 1423–1440, 2015.

[3] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A Survey on Software-Defined Wireless Sensor Networks : Challenges and Design Requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[4] Y. Duan, Y. Luo, W. Li, P. Pace, and G. Fortino, "Software Defined Wireless Sensor Networks : A Review," *Proceedings of the 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design, CSCWD 2018*, pp. 25–30, 2018.

[5] T. Luo, H. P. Tan, and T. Q. Quek, "Sensor openflow : Enabling software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[6] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks : Unbridling SDNs," in *Proceedings - European Workshop on Software Defined Networks, EWSDN 2012*, 2012, pp. 1–6.

[7] B. T. De Oliveira, L. B. Gabriel, and C. B. Margi, "Tinysdn : Enabling multiple controllers for software-defined wireless sensor networks," *IEEE Latin America Transactions*, vol. 13, no. 11, pp. 3690–3696, 2015.

[8] M. Jacobsson and C. Orfanidis, "Using Software-defined Networking Principles for Wireless Sensor Networks," *In : Proc. 11th Swedish National Computer Networking Workshop (SNCNW 2015) Karlstad, May 28-29, 2015*, no. Sncnw, pp. 1–5, 2015.

[9] R. C. A. Alves, D. A. G. Oliveira, N. S. Gustavo, and C. B. Margi, "IT-SDN : Improved architecture for SDWSN," *XXXV Brazilian Symposium on Computer Networks and Distributed Systems.*, 2017.

[10] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE : Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *Proceedings - IEEE INFOCOM*, vol. 26, 2015, pp. 513–521.

[11] B. Dhanalaxmi and G. A. Naidu, "A survey on design and analysis of robust IoT architecture," *IEEE International Conference on Innovative Mechanisms for Industry Applications, ICIMIA 2017 - Proceedings*, no. Icimia, pp. 375–378, 2017.

[12] S. G. H. Soumyalatha, "Study of iot : understanding iot architecture, applications, issues and challenges," in *1st International Conference on Innovations in Computing & Net-working (ICICN16), CSE, RRCE. International Journal of Advanced Networking & Applications*, vol. 478, 2016.

[13] J. Schaerer, Z. Zhao, and T. Braun, "DTARp : A dynamic traffic aware routing protocol for wireless sensor networks," in *RealWSN 2018 - Proceedings of the 7th International Workshop on Real-World Embedded Wireless Systems and Networks, Part of SenSys 2018*, 2018, pp. 49–54.

[14] J. Kipongo, T. O. Olwal, and A. M. Abu-Mahfouz, "Topology Discovery Protocol for Software Defined Wireless Sensor Network : Solutions and Open Issues," *IEEE International Symposium on Industrial Electronics*, vol. 2018-June, pp. 1282–1287, 2018.

[15] A. Bendjeddou, H. Laoufi, and S. Boudiit, "LEACH-S : Low Energy Adaptive Clustering Hierarchy for Sensor Network," in *2018 International Symposium on Networks, Computers and Communications, ISNCC 2018*, 2018.

[16] H. Wendi, Rabiner, C. Anantha, and B. Hari, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proceedings of the 33rd Hawaii International Conference on System Sciences - 2000*, 2000.

[17] M. Boulou, T. Yelemou, D. A. Rollande, and H. Tall, "DEARP : Dynamic Energy Aware Routing Protocol for Wireless Sensor Network," *2nd International Conference on Smart Cities and Communities : Smart Communities and Cities for Sahel's Sustainable Development : Towards Practical Solutions, SCCIC 2020*, 2020.

[18] A. Go, M. Boulou, T. Yélémou, and H. Tall, "Cmfr-cmq : Congestion management and control message quenching based on flow setup requests in sdn-wise," in *International Conference on e-Infrastructure and e-Services for Developing Countries*. Springer, 2022, pp. 355–365.

[19] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[20] ——, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–105, 2002.

[21] N. Bulusu, D. Estrin, and L. Girod, "Scalable coordination for wireless sensor networks : self-configuring localization systems," *Proc. of the 6th International*

Symposium on Communication Theory and Applications(ISCT A '01), Ambleside, UK, no. July, pp. 1–6, 2001. [Online]. Available : http://gicl.cs.drexel.edu/people/regli/Classes/CS680/Papers/Sensor{%}5CnNets/iscta-2001.pdf

[22] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS : An operating system for sensor networks," in *Ambient Intelligence*, 2005, pp. 115–148.

[23] D. Adam and Swedish Institute of Computer Science, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors," *IEEE computer society*, no. January, pp. 1–8, 2010.

[24] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han, "MANTIS OS : An embedded multithreaded operating system for wireless micro sensor platforms," in *Mobile Networks and Applications*, vol. 10, no. 4, 2005, pp. 563–579.

[25] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He, "The LiteOS operating system : Towards Unix-like abstractions for wireless sensor networks," *Proceedings - 2008 International Conference on Information Processing in Sensor Networks, IPSN 2008*, pp. 233–244, 2008.

[26] C. Li, M. Ye, G. Chen, and J. Wu, "An energy-efficient unequal clustering mechanism for wireless sensor networks," in *2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems, MASS 2005*, vol. 2005, 2005, pp. 597–604.

[27] K. Kaur and S. Deepika, "Improvement In LEACH Protocol By Electing Master Cluster Heads To Enhance The Network Lifetime In WSN," *International Journal of Science and Engineering Applications Volume*, vol. 2, no. 5, pp. 110–114, 2013.

[28] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Computer communications*, vol. 30, no. 14-15, pp. 2826–2841, 2007.

[29] D. Kumar, T. C. Aseri, and R. B. Patel, "EEHC : Energy efficient heterogeneous clustered scheme for wireless sensor networks," *Computer Communications*, vol. 32, no. 4, pp. 662–667, 2009. [Online]. Available : http://dx.doi.org/10.1016/j.comcom.2008.11.025

[30] Y. Ossama and F. Sonia, ""HEED : A Hybrid, Energy-Efficient, Distributed clustering approach for ad hoc sensor networks"," 2004.

[31] J. H. Lee, "Energy-efficient clustering scheme in wireless sensor network," *International Journal of Grid and Distributed Computing*, vol. 11, no. 10, pp. 103–112, 2018.

[32] S. Yi, J. Heo, Y. Cho, and J. Hong, "PEACH : Power-efficient and adaptive clustering hierarchy protocol for wireless sensor networks," *Computer Communications*, vol. 30, no. 14-15, pp. 2842–2852, 2007.

[33] B. Gong, L. Li, S. Wang, and X. Zhou, "Multihop routing protocol with unequal clustering for wireless sensor networks," in *Proceedings - ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2008*, vol. 2, 2008, pp. 552–556.

[34] T. Mu and M. Tang, "LEACH-B : An improved LEACH protocol for wireless sensor network," *2010 6th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2010*, pp. 2–5, 2010.

[35] G. S. Brar, S. Rani, V. Chopra, R. Malhotra, H. Song, and S. H. Ahmed, "Energy efficient direction-based PDORP routing protocol for WSN," *IEEE Access*, vol. 4, no. c, pp. 3182–3194, 2016.

[36] P. Marappan and P. Rodrigues, "An energy efficient routing protocol for correlated data using CL-LEACH in WSN," *Wireless Networks*, vol. 22, no. 4, pp. 1415–1423, 2016.

[37] A. L. Gupta and N. Shekokar, "A novel approach to improve network lifetime in WSN by energy efficient packet optimization," in *Proceedings of 2nd IEEE International Conference on Engineering and Technology, ICETECH 2016*, no. March, 2016, pp. 117–122.

[38] M. Ali Hassan, "Software Defined Networking for Wireless Sensor Networks : A Survey," *Advances in Wireless Communications and Networks*, vol. 3, no. 2, p. 10, 2017.

[39] J. Tourrilhes, P. Sharma, S. Banerjee, and J. Pettit, "SDN and OpenFlow evolution : A standards perspective," *Computer*, vol. 47, no. 11, pp. 22–29, 2014.

[40] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A Survey on Software-Defined Networking," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[41] S. D. Network and O. N. Foundation, "Le Protocole OpenFlow dans l ' Architecture SDN (Software Defined Network)," pp. 1–11, 2016.

[42] M. M. Mitchiner, "Software-Defined Networking and Network Programmability : Use Cases for Defense and Intelligence Communities," pp. 1–22, 2014.

[43] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," *2014 27th Biennial Symposium on Communications, QBSC 2014*, pp. 71–75, 2014.

[44] K. Slavov, D. Migault, and M. Pourzandi, "Identifying and addressing the vulnerabilitiesand security issues of SDN," *Ericsson Review (English Edition)*, vol. 93, no. 1, pp. 70–79, 2016.

[45] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking : A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[46] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks : A survey," *ACM Computing Surveys*, vol. 47, no. 2, 2014.

[47] Y. Zhu, F. Yan, Y. Zhang, R. Zhang, and L. Shen, "SDN-Based Anchor Scheduling Scheme for Localization in Heterogeneous WSNs," *IEEE Communications Letters*, vol. 21, no. 5, pp. 1127–1130, 2017.

[48] B. T. De Oliveira, C. B. Margi, and L. B. Gabriel, "TinySDN : Enabling multiple controllers for software-defined wireless sensor networks," *2014 IEEE Latin-America Conference on Communications, IEEE LATINCOM 2014*, 2014.

[49] P. Jayashree and F. Infant Princy, "Leveraging SDN to conserve energy in WSN-An analysis," *2015 3rd International Conference on Signal Processing, Communication and Networking, ICSCN 2015*, 2015.

[50] Y. Wang, H. Chen, X. Wu, and L. Shu, "An energy-efficient SDN based sleep scheduling algorithm for WSNs," *Journal of Network and Computer Applications*, vol. 59, pp. 39–45, 2016. [Online]. Available : http://dx.doi.org/10.1016/j.jnca.2015.05.002

[51] C. Gonzalez, S. M. Charfadine, O. Flauzac, and F. Nolot, "SDN-based security framework for the IoT in distributed grid," in *2016 International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*. IEEE, jul 2016, pp. 1–5. [Online]. Available : http://ieeexplore.ieee.org/document/7555946/

[52] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks : Unbridling SDNs," *Proceedings - European Workshop on Software Defined Networks, EWSDN 2012*, pp. 1–6, 2012.

[53] N. Q. HHieu, N. Q., Huu Thanh, N., Huong, T. T., Quynh Thu, N., & Quang, H. Van. (2018). Integrating trickle timing in software defined WSNs for energy efficiency. 2018 IEEE 7th International Conference on Communications and Electronics, ICCE 2018, 75–80. https, N. Huu Thanh, T. T. Huong, N. Quynh Thu, and H. V. Quang, "Integrating trickle timing in software defined WSNs for energy efficiency," *2018 IEEE 7th International Conference on Communications and Electronics, ICCE 2018*, pp. 75–80, 2018.

[54] N. Abdolmaleki, M. Ahmadi, H. T. Malazi, and S. Milardo, "Fuzzy topology discovery protocol for SDN-based wireless sensor networks," *Simulation Modelling Practice and Theory*, vol. 79, no. December, pp. 54–68, 2017.

[55] X. Tan, H. Zhao, G. Han, W. Zhang, and T. Zhu, "QSDN-WISE : A New QoS-Based Routing Protocol for Software-Defined Wireless Sensor Networks," *IEEE Access*, vol. 7, pp. 61 070–61 082, 2019.

[56] O. Flauzac, C. Gonzalez, and F. Nolot, "SDN Based Architecture for Clustered WSN," in *Proceedings - 2015 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2015*, 2015, pp. 342–347.

[57] Q. Xu and J. Zhao, "A wsn architecture based on sdn," in *4th International Conference on Information Systems and Computing Technology*. Atlantis Press, 2016, pp. 159–163.

[58] A. Ouhab, T. Abreu, H. Slimani, and A. Mellouk, "Energy-efficient clustering and routing algorithm for large-scale SDN-based IoT monitoring," *IEEE International Conference on Communications*, vol. 2020-June, 2020.

[59] Z. Haosen, W. Muqing, and L. Boyang, "Energy-Balanced Clustering Algorithm for Software-Defined Wireless Sensor Networks," pp. 147–152, 2019.

[60] J. Shen, M. Wu, and Z. Zhao, "Clustering Algorithm Based on Extending Dynamic Subnetwork Scheme for Software-Defined Wireless Sensor Networks," *2018 IEEE/CIC International Conference on Communications in China, ICCC Workshops 2018*, pp. 190–195, 2019.

[61] L. L. Bello, A. Lombardo, S. Milardo, G. Patti, and M. Reno, "Software-Defined Networking for Dynamic Control of Mobile Industrial Wireless Sensor Networks," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2018-Septe, pp. 290–296, 2018.

[62] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-defined fog network architecture for iot," *Wireless Personal Communications*, vol. 92, no. 1, pp. 181–196, 2017.

[63] M. Al-Hubaishi, C. Çeken, and A. Al-Shaikhli, "A novel energy-aware routing mechanism for SDN-enabled WSAN," *International Journal of Communication Systems*, vol. 32, no. 17, pp. 1–17, 2019.

[64] A. Banerjee and D. M. Akbar Hussain, "SD-EAR : Energy aware routing in software defined wireless sensor networks," *Applied Sciences (Switzerland)*, vol. 8, no. 7, 2018.

[65] M. Ndiaye, A. M. Abu-Mahfouz, G. P. Hancke, and B. Silva, "Exploring control-message quenching in sdn-based management of 6LoWPANs," in *IEEE International Conference on Industrial Informatics (INDIN)*, vol. 2019-July, 2019, pp. 890–893.

[66] K. Nagarathna, "Energy-aware strategy for data forwarding in IoT ecosystem," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 5, pp. 4863–4871, 2020.

[67] S. Buzura, B. Iancu, V. Dadarlat, A. Peculea, and E. Cebuc, "Optimizations for energy efficiency in software-defined wireless sensor networks," *Sensors (Switzerland)*, vol. 20, no. 17, pp. 1–23, 2020.

[68] Y. Zhang, Y. Zhu, F. Yan, W. Xia, and L. Shen, "Energy-efficient radio resource allocation in software-defined wireless sensor networks," *IET Communications*, vol. 12, no. 3, pp. 349–358, 2018.

[69] Y. Zhang, Y. Zhu, F. Yan, Z. Li, and L. Shen, "Semidefinite programming based resource allocation for energy consumption minimization in software defined wireless sensor networks," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, no. 61471164, 2016.

[70] R. Pradeepa and M. Pushpalatha, "SDN enabled SPIN routing protocol for wireless sensor networks," in *Proceedings of the 2016 IEEE International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2016*, 2016, pp. 639–643.

[71] Z. J. Han and W. Ren, "A novel wireless sensor networks structure based on the SDN," *International Journal of Distributed Sensor Networks*, vol. 2014, no. 1, pp. 1–7, 2014.

[72] H. Fotouhi, M. Vahabi, A. Ray, and M. Björkman, "SDN-TAP : An SDN-based traffic aware protocol for wireless sensor networks," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services, Healthcom 2016*, 2016, p. n.

[73] H. Ying, "A general technique for deriving analytical structure of fuzzy controllers using arbitrary trapezoidal input fuzzy sets and Zadeh AND operator," *Automatica*, vol. 39, no. 7, pp. 1171–1184, 2003.

[74] P. P. Angelov and D. P. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 484–498, 2004.

[75] A. Velinov and A. Mileva, "Running and testing applications for contiki os using cooja simulator," 2016.

[76] M. Yassine and A. Ezzati, "Etude et développement d'un protocole symétrique pour sécuriser les communications des RCSF Yassine Maleh , Abdellah Ezzati HAL Id : hal-01423684," no. September 2017, 2019.

[77] V. Bapat, P. Kale, V. Shinde, N. Deshpande, and A. Shaligram, "WSN application for crop protection to divert animal intrusions in the agricultural land," *Computers and Electronics in Agriculture*, vol. 133, pp. 88–96, 2017. [Online]. Available : http://dx.doi.org/10.1016/j.compag.2016.12.007

[78] M. Boulou, T. Yélémou, S. I. Ouattara, and B. H. M. Sanou, "Leach-s enhancement to ensure wsn stability," in *Ad Hoc Networks and Tools for IT*. Springer, 2021, pp. 99–113.