Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

6LoWPAN	IPv6 Low-power Wireless Personal Area Networks
CDS.	Connected Dominating Sets
CL	Controleur Local
CLs.	Contrôleurs Locaux
Deg	Degrée de conectivité
DLC-CDS	Distributed Local Controler- Conected Dominating Set,
DSP-CDS	Distributed Single Phase_ Conected Dominating Set
FFD	Full Function Device
IdO	Internet des Objets
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
LPWAN	Low Personnel Area Network
MNLR_RPL	Mobile Node Local Repair_ RPL
NodId	Identificateur d'un noeud
OF	Objective Function
OF0	Objective Function Zero
PDR	Packet Delivery Ratio
PP	Preferred Parent
QdS	Qualité de Service
RCASFs	Reseaux de Capteurs et Actionneurs Sans Fil
RCSFs	Reseaux de Capteurs Sans Fil
RFD	Reduced Function Device
RFID	Radio frequency identification)
RPL	Routing protocol for Low power and Lossy Networks
SDN	Software Defined Network
SetID	Set Identificateur
WPAN	Wireless Personal Area Networks
WSN	Wireless Sensor Network

Introduction générale

1. Problématique et motivation
2. Objectifs
3. Principales contributions de la thèse
4. Organisation du manuscrit

Introduction générale

1. Problématique et motivation

Nous assistons récemment, à l'émergence d'un nouveau concept ayant bel et bien bouleversé le domaine des réseaux de télécommunication sans fil modernes, avec de nouvelles technologies tendant vers la miniaturisation et à l'intelligence des dispositifs, d'où le concept populaire de la recherche et de l'industrie appelé *Internet des Objet* (ou *IoT* pour *Internet of Things*) [1] [2]. L'idée de base de l'*IdO* est la présence ubiquitaire autour de nous d'une variété d'objets intelligents via Internet, et qui consiste en une large interconnexion des étiquettes *RFID*, des capteurs, des actionneurs, des machines, des véhicules, et même des personnes équipées de capacité de communication filaire ou sans fil. Ces objets intelligents, et grâce à des schémas d'adressage uniques, sont capables d'interagir les uns avec les autres et de coopérer avec leurs voisins pour atteindre des objectifs communs. Le domaine d'application de l'*IdO* est vaste à savoir : villes intelligentes, soins de santé, systèmes de transport intelligents, fabrication future, surveillance des frontières, etc. Cette popularité croissante de l'*IdO* a entraîné une production et un déploiement à grande échelle des réseaux *RCSFs* [3] ou "*Wireless Sensor Network*" (WSN).

Dans ce contexte, cette thèse porte sur le contrôle et la gestion des *RCSFs* basés sur la technologie des réseaux personnels sans fil *IPv6* (*6LoWPAN*) afin d'assurer le bon fonctionnement de ce type de réseau, où la défaillance des nœuds capteurs est inévitable. Un *RCSF* est constitué d'un certain nombre de capteurs électroniques embarqués qui détectent et collectent certaines informations relatives à des tâches spécifiques, afin d'être traitées par un utilisateur distant via une connectivité *IP*. Ces dispositifs sont caractérisés par leurs ressources limitées en énergie, en bande passante, en débit (*250Kbps*), en mémoire (traitement, stockage et traitement) et en capacité de communication (courte portée). Ces caractéristiques inhérentes ne cessent de susciter le développement de nouvelles et vastes perspectives applicatives dans de nombreux domaines (militaires, environnementaux, médicaux, etc.).

D'un autre côté, la norme *6LoWPAN* [4], qui est proposée par la communauté *IETF* (*Internet Engineering Task Force*), s'est avérée utile car elle est développée afin de permettre aux dispositifs sans fil de faible puissance et ayant des capacités de traitement limitées de participer aux *RCSF*. Contrairement aux autres technologies, *6LoWPAN* résout certaines faiblesses tout

en offrant des avantages tels que l'évolutivité, la flexibilité et la connectivité directe des dispositifs à Internet. Une telle exigence est satisfaite grâce à l'utilisation des *RCSFs* basés sur *6LoWPAN*, et sans aucun doute, les *RCSFs* jouent un rôle important en réalisant plusieurs tâches de routage. De plus, la technologie *6LoWPAN* dans sa couche adaptative avec la norme *IEEE 802.15.4* fournit un nouveau protocole proactif de routage nommé *RPL* [5]. Ce protocole est développé par le groupe travail *IETF* (*ROLL*) [6] et qui est proposé pour le routage dans les réseaux avec pertes et à faible puissance. Puisque cette thèse dans sa première partie de contribution, se concentre sur le contrôle et la gestion des nœuds défaillants dans les *RCSFs* tout en intégrant le concept de la mobilité pour remplacer ces nœuds en question, alors nous nous référons au protocole *RPL* car la réparation locale des *RCSFs* basés sur *6LoWPAN* n'est offerte qu'avec *RPL*. Parmi les nombreuses problématiques abordées sur le contrôle des *RCSFs* telles que la tolérance aux pannes ou la réparation locale d'un réseau en cas de nœuds défaillants, peut être considérée comme l'une des métriques les plus importantes vis-à-vis de la continuité de fonctionnement du réseau ainsi que sa *QdS*. Malgré les nombreux travaux de recherche consacrés à ce sujet, la problématique du contrôle et celle liée au *RCSF* s'avèrent être d'un grand intérêt vu l'ampleur des exigences de l'hétérogénéité et de la scalabilité de ces réseaux. Avec ces exigences, il pourrait y avoir dans la prochaine décennie des milliards de nœuds capteurs intelligents interconnectés par Internet et en parallèle la nécessité d'avoir un contrôle fiable, révolutionnaire avec le nouveau paradigme *SDN* (*Software-Defined Networking*) [7] qui pourrait fournir une plate-forme solide afin de gérer et contrôler un si grand nombre de dispositifs en réseau et également résoudre certains problèmes clés rencontrés par les *RCSFs* [3].

Le réseau défini par logiciel ou *SDN* a été développé pour faciliter la gestion programmable des réseaux et permettre un contrôle programmatique simple du chemin des données. *SDN* permet aux administrateurs réseau de gérer les services réseau par abstraction des fonctionnalités. L'intelligence du réseau est centralisée dans les contrôleurs *SDN* logiciels, qui maintiennent une vue globale du réseau. En exploitant l'intelligence centralisée du contrôleur *SDN*, il est possible de modifier le comportement du réseau et de déployer de nouvelles applications et de nouveaux services réseau en temps réel. Par ailleurs, cette stratégie de contrôle centralisée permet à d'autres problématiques de se lancer. Des problématiques liées à la surcharge du réseau par du trafic de contrôle et la probabilité de défaillance du seul point centrale (*SDN contrôleur*). C'est pourquoi, il est important de proposer une architecture hybride et programmable capable de prendre en compte les différentes caractéristiques de technologies de communication sans fil, et de réduire la charge de trafic de contrôle et les messages de

signalisation. Contrairement au *SDN* classique, nous proposons, dans une deuxième partie de contribution de cette thèse, une architecture semi-distribuée avec répartition du rôle des contrôleurs tout en introduisant trois niveaux de contrôle : les contrôleurs Principal, Secondaire et Local. Dans cette partie, nous aborderons l'hétérogénéité des technologies de réseau dans l'*IdO,* et nous proposerons une nouvelle architecture hybride et programmable basée sur le paradigme *SDN*. Par conséquent, la décentralisation du concept *SDN* ainsi que de remplacer l'intelligence centralisée du contrôleur *SDN* sa reste un challenge de recherche ouvert aux exigences et aux contraintes des *RCSFs* dans l'*IdO* basé sur la technologie *6LoWPAN*.

2. Objectifs

Dans cette thèse, nous nous sommes intéressés au contrôle et la gestion des *RCSFs* pour l'*IdO,* afin de surmonter et d'atténuer les problèmes rencontrés et avec l'objectif d'assurer la continuité du fonctionnement de ce type de réseaux. A cet effet, l'objectif principal de cette thèse se résume en deux catégories d'attente.

2.1 Premier Objectif

Notre premier objectif consiste à proposer une méthode nouvelle et efficace pour la gestion de la défaillance des nœuds avec le protocole *RPL*, tout en mettant en œuvre l'algorithme de réparation locale avec le concept de la mobilité pour remplacer ces nœuds en question. La mobilité est pertinente avec un protocole de routage, ainsi que la réparation locale dans les *RCSFs* basés sur la technologie *6LoWPAN* n'est offerte qu'avec *RPL*. Contrairement aux protocoles de routage existants, le protocole *RPL* qui est conçu pour les réseaux caractérisés par des ressources limitées de leurs dispositifs, est doté d'un mécanisme de réparation qui se déclenche lors d'une détection de nœuds ou liaisons défaillants dans le réseau. Et ce, malgré l'énorme coût de la technique de réparation de ce protocole en termes de consommation d'énergie, et le coût de trafic, de contrôle et de signalisations supplémentaires par l'effet de la reconstruction dans l'arbre *RPL* forcé par des défaillances de nœuds. D'un autre côté, *RPL* a été conçu à l'origine pour les réseaux statiques, sans support pour la mobilité. Cependant, gérer le mécanisme de réparation de la topologie réseau *RPL* avec des nœuds mobiles est un véritable défi. Simultanément, et pour remplacer la défaillance de nœuds par la mobilité de leur prédécesseur sans la reconstruction de l'arbre *RPL* avec l'algorithme proposé *MNLR_RPL* [9], est une solution proposée avec pour objectif d'estimer la continuité de fonctionnement du réseau.

2.2 Deuxième Objectif

Notre deuxième objectif dans cette thèse n'en est guère de moindre importance. A cet effet, nous nous sommes intéressés au problème de contrôle et gestion des *RCSFs* dans l'*IdO* avec une hétérogénéité de technologies. Le paradigme *IdO* a besoin d'un logiciel spécialisé avec des fonctionnalités de réseau et de support *IP*. En effet, de nombreuses technologies de communication et de réseau sont proposées pour le contexte *IdO* sans tenir compte de la coexistence et de l'interopérabilité. Pour garantir une coexistence technologique, le besoin d'un modèle de contrôle fiable avec le nouveau paradigme *SDN* peut fournir une base solide pour gérer et contrôler un si grand nombre de dispositifs technologique incluant les *RCSFs*. Contrairement aux contrôleurs *SDN* classique utilisés dans la majorité des travaux de recherche et qui sont munis que d'une seule direction, nous avons opté, dans notre travail, pour une architecture différente dans sa nature de celui du contrôle centralisé. Notre principal objectif consiste à proposer une nouvelle architecture proactive et semi distribuée avec trois niveaux de contrôle : les contrôleurs Principal, Secondaire et Local, et ce, afin d'améliorer la fonctionnalité du contrôle dans les *RCSFs,* tout en respectant les caractéristiques des ressources limitées de ces derniers.

Un autre objectif est de considérer l'hétérogénéité des technologies de réseau mais aussi de réduire les messages de signalisation et trafics de contrôle liés aux mécanismes de contrôle du réseau, et d'éviter la probabilité de défaillance du seul point de contrôle qu'est celui du contrôleur central du *SDN*. En effet, nous introduisons un nouveau mécanisme de Contrôleurs Locaux (*CLs*) basé sur les ensembles *CDS* (*Connected Dominating Sets*) [8] et des approches de la logique floue (*LF*). Ce mécanisme est basé sur le développement de notre méthode proposée *DLC-CDS* (*Distributed local Controler- Conected Dominating Set*) [10] pour la sélection des dominants qui vont jouer le rôle de contrôleurs locaux connectés dans l'architecture proposé.

3. Principales contributions de la thèse

La présente thèse est une contribution au problème de contrôle dans les *RCSF* basés sur la technologie *6LowPAN*. Notre contribution se résume en deux parties complémentaires mais différentes dans l'environnement d'implémentation et d'évaluation.

La première partie de nos contributions consiste en premier lieu à proposer une méthode de contrôle et de gestion des *RCSFs* basés sur la technologie *6LoWPAN*, capable de remplacer des nœuds défaillants par des nœuds mobiles [9]. Ce choix a été motivé par le fait que la technologie *6LoWPAN* nous offre le protocole de routage *RPL* mené par la solution de réparation (*locale ou*

globale) de la défaillance de nœuds ou de liens dans le réseau. Mais cette technique de réparation est étroitement contraignante en termes de consommation d'énergie, coût de trafic de contrôle supplémentaire par l'effet de reconstructions dans un arbre forcé par des défaillances de nœuds. Un autre facteur de motivation, est que la possibilité de support de mobilité pour ce protocole (*RPL*) fut un défi à relever, car *RPL* a été conçu à l'origine en tant que routage statique. Cependant, gérer en même temps le mécanisme de réparation de *RPL* avec des nœuds mobiles sans reconstruction de l'arbre *RPL*, est une solution proposée afin d'assurer la continuité de fonctionnement du réseau. Cette méthode a été réalisée à l'aide d'un algorithme de gestion et de réparation locale *MNLR_RPL* [9].

Cette méthode a fait l'objet d'une communication internationale avec une publication.

- D.Bendouda, L.Mokdadb, H.Haffaf, "Method For Fault Management With RPL Protocol In WSNs", The International Conference on Advanced Wireless, Information, and Communication Technologies, Tunisia, (AWICT 2015), ScienceDirect Procedia Computer Science Volume 73, 2015, Pages 395-402.
- D.Bendouda, L.Mokdadb, H.Haffaf, " Exploiting node mobility for fault management in RPL-based wireless sensor networks", International journal of High Performance Computing and Networking, inderscience, accepté en juin 2016, DOI: 10.1504/IJHPCN.2016.10004838, http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijhpcn

Dans la deuxième partie de nos contributions, nous avons pris en considération la présence d'un nouveau paradigme de contrôle du *SDN*, qui peut changer les limites des infrastructures des *RCSFs* dans l'*IdO* actuelle. En effet, ces derniers ont besoin d'architectures intelligentes, robustes, programmables et gérables pour satisfaire les exigences de ces futures applications. Nous étions donc motivées par le fait que le paradigme *SDN* offre de nouvelles perspectives de gestion et de contrôle des *RCSF*s dans l'*IdO*, ainsi qu'un contrôle centralisé que ce paradigme offre. Cependant, nous avons proposé une nouvelle architecture de contrôle hybride et programmable basée sur ce paradigme. Cette proposition est basée sur la répartition de la fonction du contrôle tout en proposant trois types de contrôleurs, dont des contrôleurs locaux (*CLs*), au niveau de l'infrastructure des *RCSF* dans l'*IdO*. Le mécanisme des *CLs* est basé sur la méthode de sélection des sous-ensembles connectés CDS (Connected Dominating Sets) et sur des approches de la logique floue. Cela fut réalisé à travers l'algorithme distribué à une seule phase *DLC-CDS*.

Enfin, l'implémentation et l'évaluation des performances de l'algorithme de la première proposition ont été réalisées à l'aide du simulateur *Cooja* sous *Contiki OS* et sous le simulateur *Matlab*, quant à la deuxième proposition.

Cette étape a été valorisé par une communication internationale et publication.

- D.Bendouda, A.Rachedi, H.Haffaf, "An hybrid and proactive architecture based on SDN for Internet of Things", 13th International Wireless Communications and Mobile Computing Conference (IWCMC) Wireless Communications and Mobile Computing Conference (IWCMC), pp951-956, Valencia, Spain, Jun 2017.
- D.Bendouda, A.Rachedi, H.Haffaf, "Programmable architecture based on Software Defined Network for Internet of Things: Connected Dominated Sets approach", An International Journal of *Future Generation Computer Systems*, Elsevier, 2018, volume 80, pp.188- 197.

4. Organisation de la thèse

Ce manuscrit s'articule autour de quatre chapitres (comme illustré par la figure 1) suivis d'une conclusion générale et des perspectives de recherches. Le positionnement bibliographique global, et qui fut le principal de nos travaux, sera présenté au premier chapitre. Etant donné qu'il y a deux contributions distinctes dans ce manuscrit, alors un positionnement sera présenté séparément dans chaque chapitre parmi les deux derniers. Dans ces mêmes chapitres, seront, en outre, détaillées nos différentes contributions.

Le premier chapitre de ce manuscrit présente un état de l'art sur l'*IdO*. Ensuite, nous présentons un aperçu sur les *RCSFs* avec leurs caractéristiques et contraintes, ainsi que leurs différents domaines d'application. Egalement seront détaillés dans ce chapitre, les technologies de communications dans des *RCSFs* et en particulier la technologie *6LowPAN* sur laquelle se base la méthode proposée, de gestion et de réparation de nœuds défaillants.

Le deuxième chapitre est une étude bibliographique sur le concept de la tolérance aux pannes et en particulier les nœuds défaillants ainsi que les différentes stratégies et solutions proposées pour la réparation locale dans les *RCSFs*. Ce chapitre permet de positionner notre première partie de contribution au problème de la défaillance de nœuds dans les *RCSFs* par rapport aux approches proposées dans la littérature.

Le troisième chapitre se concentre sur notre choix de solution de tolérance aux nœuds défaillants à travers le mécanisme de réparation locale du protocole *RPL* et l'intégration du concept mobile pour le remplacement des nœuds défaillants. Nous présenterons tout d'abord et en détail le protocole de routage *RPL*, ainsi qu'une étude bibliographique sur le support de la mobilité pour le protocole *RPL* et le mécanisme de réparation locale. La méthode proposée pour la gestion et la réparation de nœuds défaillants sera détaillée par la suite avec le concept

de mobilité des nœuds prédécesseurs. L'implémentation et l'évaluation des performances de cette méthode seront traitées dans la dernière partie de ce chapitre où seront illustrés l'environnement de simulation et les différents résultats de performances.

Le quatrième chapitre constitue notre deuxième contribution, il s'agit d'une amélioration de l'architecture centralisé du nouveau paradigme *SDN*. Dans ce chapitre, nous présentons d'abord un bref aperçu sur le *SDN* et son concept de contrôle dans les *RCSFs*.

Ensuite une étude bibliographique sur la décentralisation de la fonctionnalité du contrôle dans le réseau avec le paradigme *SDN*, sera présentée dans ce même chapitre. Nous présentons par la suite notre nouvelle architecture proposée puis décrivons l'architecture proposée avec illustration, ainsi que ses différents modules et niveaux de contrôle. Nous présentons par la suite la méthode de sélection des contrôleurs locaux à travers l'algorithme proposé *DLC-CDS* qui se base sur l'approche des *CDS*. Cette approche ainsi que la logique floue sont décrites dans ce même chapitre. Enfin, la dernière section sera consacrée à la partie implémentation et évaluation des performances de l'algorithme *DLC-CDS* avec différents scenarios de simulation. Par ailleurs, différentes comparaisons de performances avec l'algorithme de référence *DSP-CDS* existantes seront effectuées pour valider la méthode de sélection des *CLs* dans l'architecture proposée.

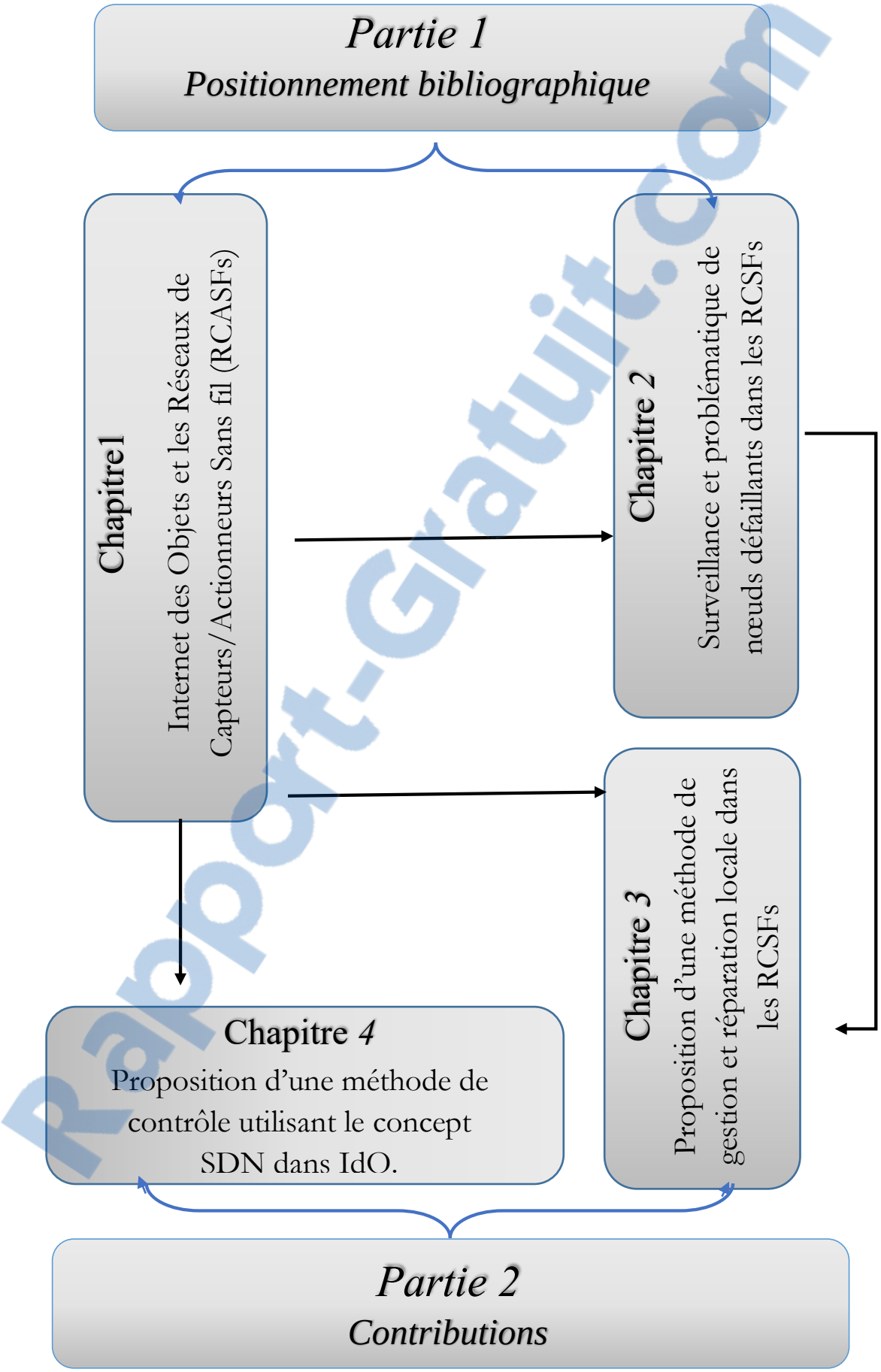


Figure 1 : Navigation à travers la thèse.

Partie 1

Positionnement bibliographique

Internet des Objets et les Réseaux de Capteurs/Actionneurs Sans fil (RCASFs)

Chapitre 1

1.1 Introduction

L'Internet des Objets (*IdO*), est un nouveau concept qui représente une révolution technologique, apparue comme une tendance de recherche prometteuse et ciblant un grand nombre d'applications dans le domaine des télécommunications sans fil modernes [1, 2]. L'*IdO* fait référence à une variété d'équipements et de systèmes d'informations de détection tels que les réseaux de capteurs sans fil. Cette révolution permet de changer radicalement la façon de concevoir les systèmes de surveillance à grandes échelles. L'interaction entre les différents équipements réseaux connectés à internet en *IPv6*, présente le défi de rendre plus faciles et intelligentes certaines tâches de notre vie quotidienne. À ces fins, L'application de ce paradigme nécessite l'utilisation de technologies différentes. Une telle exigence est satisfaite grâce à l'utilisation de réseaux de capteurs sans fil (*RCSFs*) [3] basés sur la technologie *6LoWPAN* [4], qui se caractérise par les ressources limitées de leurs équipements. Sans aucun doute, le *RCSF* joue un rôle important en accomplissant plusieurs tâches à la fois. Le concept des *RCSFs* consiste à intégrer des capteurs informatiques embarqués dans l'infrastructure Internet avec *6LowPAN*, qui est standardisé par l'*IETF* [6]. La norme *6LoWPAN* s'est révélée utile car elle facilite le déploiement de l'offre de l'*IdO*, notamment l'évolutivité, la flexibilité et la connectivité de bout en bout, entre autres.

Ce chapitre se concentre sur la mise en évidence des technologies et protocoles standards les plus importants liés aux réseaux sans fil à faible puissance et à perte de données. Puisque cette thèse se focalise sur le contrôle des *RCSFs*, dont la première contribution se base sur la gestion des nœuds capteurs défaillants, et s'avère pertinente avec un mécanisme de réparation local d'un protocole de routage, et dont l'orientation principale pour le protocole *RPL* est décrite dans ce chapitre.

Dans le reste de ce chapitre, nous présentons dans la section 1.2 une brève description de l'*IdO*, puis une description rapide sur les technologies de l'*IdO*. Dans la section 1.3, nous

présentons les *RCASFs* (*Réseaux de Capteurs et Actionneurs Sans Fil*) comme une classe des réseaux hétérogènes. Les principes de conception des *RCSFs* ainsi que leurs différentes caractéristiques et domaines de recherche sont présentés dans la section 1.4, les technologies de communication des *RCSFs* sont présentées dans la section 1.5. La section 1.6, présente une discussion sur les protocoles de routages conçus pour la réparation locale des nœuds en pannes dans un réseau. Enfin, dans la section 1.7, nous tirons nos conclusions et avancerons nos propositions concernant la première partie de contribution dans le reste de cette thèse.

1.2 Concept de l'Internet des Objets (IdO)

L'Internet des objets (*IdO*, ou en anglais « *Internet of Things* » *IoT*) [1], est un nouveau concept ayant prouvé rapidement du succès dans le domaine des télécommunications sans fil modernes. L'*IdO* correspond à un ensemble d'objets physiques et intelligents connectés qui communiquent via de multiples technologies avec diverses plateformes de traitement de données, en lien avec les vagues du *Cloud* et du *Big data* [11] [12]. Les objets intelligents peuvent être considérés comme des machines, appareils, capteurs, actionneurs, véhicules, voire des personnes équipées de capacités de communication filaires ou sans fil, de telle sorte qu'il y ait une fusion entre le monde réel (physique) et le monde numérique (virtuel).

L'objectif principal de son développement consiste à rendre certaines tâches faciles pour les utilisateurs et les environnements intelligents, en utilisant la fonctionnalité de contrôle à distance des choses. Cette technologie est d'abord appliquée aux capteurs et aux actionneurs, ensuite, elle est étendue à différents systèmes, à savoir : les réseaux intelligents, les maisons intelligentes, les transports intelligents et les villes intelligentes. De nouveaux modèles d'architecture permettent d'intégrer des capteurs et le réseau Internet (voir figure 1.1). Cette communication entre un capteur et le Cloud se fait par une couche virtuelle qui implémente le fonctionnement des capteurs réels. Une telle couche donne naissance à de nouvelles architectures appelées réseaux de capteurs (*Sensor Cloud*) ou *Cloud* des capteurs virtuels [13].

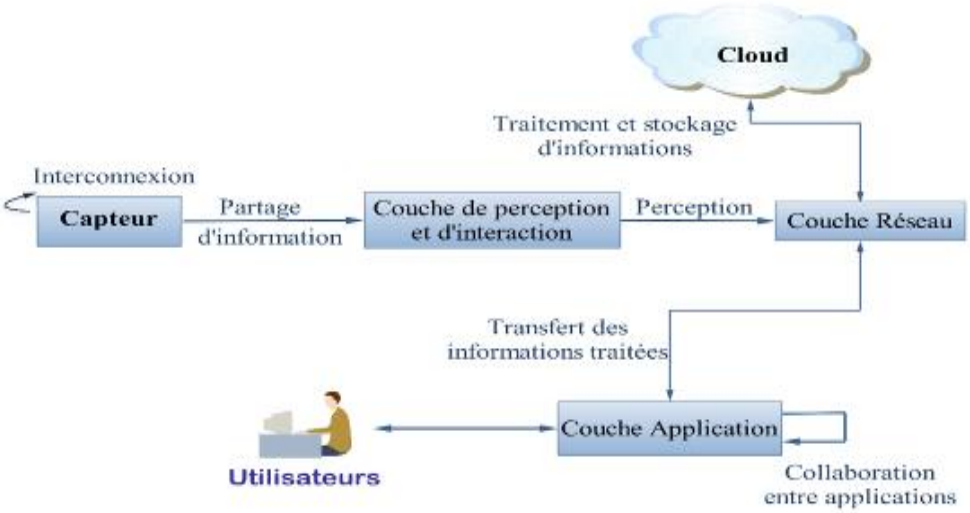


Figure 1.1 : Modèle d'architecture capteur cloud [13].

1.2.1 Historique d'évolution d'IdO

La première occurrence d'*IdO* est retrouvée en 1999 au sein des travaux du groupe Auto-*ID* du *MIT* travaillant sur l'identification de la fréquence radio (*RFID*) en réseau et sur les technologies de détection (*Radio Frequency Identification*, *RFID*). C'est lors d'une présentation pour *Procter & Gamble* que *Kevin Ashton* [14] évoquait pour la première fois l'idée selon laquelle la *RFID* pourrait avoir un rôle majeur au sein de la chaîne d'approvisionnement au travers d'objets connectés [14]. En 2000, la société *LG* annonce son premier réfrigérateur intelligent connecté à internet. De plus, la technologie *RFID* qui est l'une des technologies constitutionnelles de l'*IdO*, a commencé à être massivement déployée vers les années 2003 et 2004. D'autre part, une initiative très intéressante a été prise en 2008, un groupe de recherche appelé *IPSo Alliance* s'est consacré à promouvoir l'utilisation du protocole *IP* (*Internet Protocol*) pour les réseaux d'objets miniatures intelligents.

De nombreux travaux de recherches se sont succédé et se sont tous concentrés autour de la réalisation, dans les meilleures conditions, de la vision de l'Internet des objets et la mener à sa maturité en dépit de tous les défis soulevés. Et ce, avec la considération des progrès technologiques continus dans le marché des dispositifs intelligents et dans le domaine de technologies de télécommunication comme : *le Cloud Computing*, le concept du *SDN* [7] [15], etc.

1.2.2 Les objets connectés

L'apparition du concept de l'*IdO* est due à la variété des équipements et des objets utilisés dans notre vie quotidienne (voir figure 1.2) : ordinateurs, capteurs, actionneurs, smartphones,

véhicules connectés, smart homes, etc. Les objets ayant des identités et des personnalités virtuelles, opérant dans des espaces intelligents et utilisant des interfaces intelligentes pour se connecter et communiquer au sein de contextes d'usages variés [4].

Un objet connecté est un matériel électronique qui peut communiquer avec un Smartphone, une tablette tactile et/ou un ordinateur, ses caractéristiques pouvant évoluer au cours du temps (position, niveau de batterie, etc.). Il peut envoyer et recevoir des informations, via une liaison sans fil, *Bluetooth* ou *WiFi*, etc. D'autres définitions s'accordent à dire qu'un objet connecté possède des capacités de calcul, d'acquisition (capteur) et d'action (actionneur). Les objets identifiés par *RFID* sont exclus de cette définition, car une puce *RFID* classique ne peut pas être considérée comme un dispositif de calcul, celle-ci représente un identifiant stocké dans une mémoire.

L'intérêt principal d'un objet connecté est l'interactivité, la possibilité de récupérer des informations, ou d'envoyer des statistiques, de garder le contact, etc.

Figure 1.2. Quelques exemples d'objets connectés.

1.2.3 Domaine d'applications de l'*IdO*

L'Internet des objets n'est pas uniquement un immense ensemble d'objets intelligents interconnectés et connectés à Internet mais c'est également, mais plus considérablement, des applications qui sont en fait la raison d'être de cette nouvelle vague de connectivité sur Internet.

L'existence des objets intelligents avec de nouvelles possibilités de communications automatiques et intelligentes vont sensiblement améliorer le mode de vie des gens ainsi que la *QdS* (*qualité de services*) dans divers domaines à travers des degrés élevés d'autonomie et d'intelligence. Aujourd'hui de nombreux secteurs sont concernés par l'Internet des Objets. Parmi les principaux secteurs nous pourrions citer : l'énergie et le développement durable, la logistique, la domotique, les transports, la téléphonie mobile, la distribution, le service à la

personne et la sécurité, etc. Nous allons voir par la suite les principaux secteurs avec des exemples de projets. Le domaine d'application comme présenté dans la figure 1.3 est vaste, notamment : les maisons et villes intelligentes (appelées « smart *homes & cities* »), soins de santé, systèmes de transport intelligents, fabrication future, surveillance des frontières, etc. [4], [1], [2] [17].

1.2.3.1 La Domotique

C'est l'ensemble des techniques et technologies de l'électronique, de l'électricité, de la mécanique, des télécommunications et de l'informatique appliquées à la gestion automatisée des bâtiments individuels et collectifs. Elle pilote de façon intelligente l'ensemble des systèmes automatisés présents dans les habitations individuelles et collectives (on parle alors de maison intelligente). La domotique permet d'améliorer quatre secteurs dans l'habitat : le confort, la sécurité, les économies d'énergie, la dépendance [4].

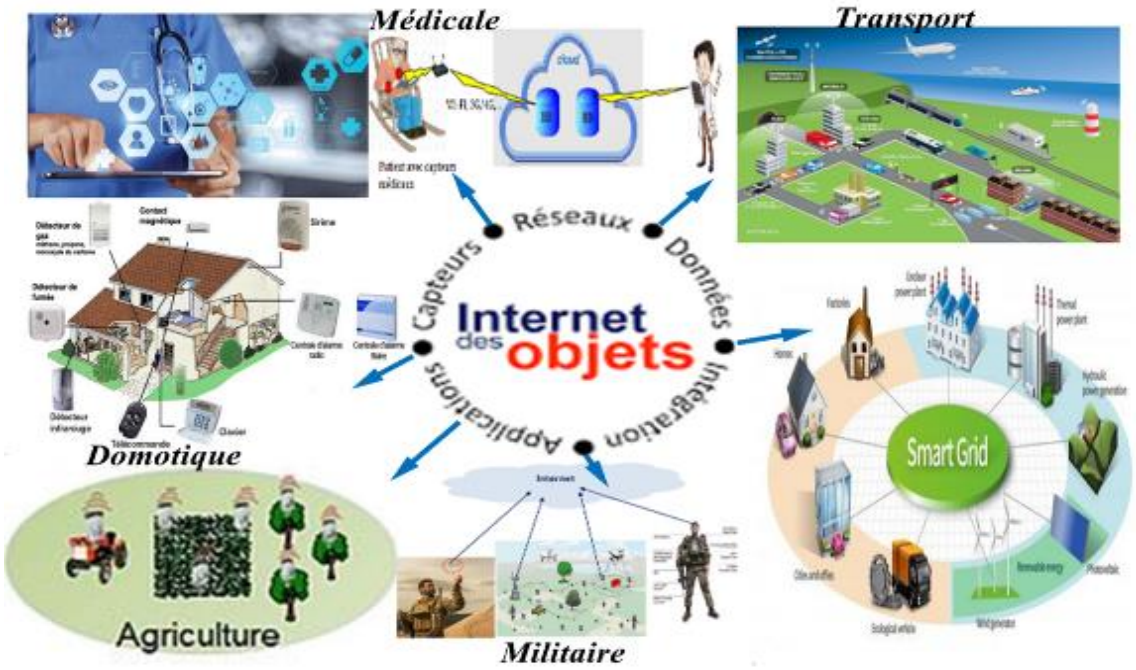


Figure 1.3 : Domaines d'applications de l'Internet des Objets.

- **Confort** : Volets roulants électriques et portail électrique, Gestion intelligente de l'éclairage (allumage/extinction automatique ...), Commande à distance par smartphone
- **Sécurité** : Alarme intrusion, alarme incendie, détection fuite d'eau, détection fuite de gaz, coupure automatique des circuits à risque en cas d'absence (plaque de cuisson, cafetières ...).

- **Economies d'énergies :** Détection fuites d'eau, gestion intelligente du chauffage (programmation horaire, températures confort/réduit ...) et la gestion intelligente de l'éclairage (détection présence, variation intensité lumineuse ...).

1.2.3.2 Les villes intelligentes (*Smart Cities*)

Une ville intelligente (*smart city*), consiste globalement en l'optimisation des coûts, de l'organisation, du bien-être des habitants. Elle désigne une ville utilisant les technologies de l'information et de la communication (*TIC*) pour améliorer la qualité des services urbains ou encore réduire ses coûts. Une ville intelligente serait capable de mettre en œuvre une gestion des infrastructures (d'eau, d'énergies, d'information et de télécommunications, de transports, de services d'urgence, d'équipements publics, de bâtiments, de gestion et tri des déchets, etc.) communicantes, adaptables, durables et plus efficaces, automatisées pour améliorer la qualité de vie des citoyens, dans le respect de l'environnement. Un des aspects d'une ville intelligente est les Systèmes de Transport Intelligents (*STI*) (en anglais Intelligent Transportation System (*ITS*)) qui sont les applications des nouvelles *TIC* au domaine des transports. On trouve les *STI* dans plusieurs champs d'activité : dans l'optimisation de l'utilisation des infrastructures de transport, dans l'amélioration de la sécurité (notamment de la sécurité routière) et de la sûreté ainsi que dans le développement des services [16]. Pour la création et la meilleure gestion de la ville intelligente, les réseaux de capteurs sans fil (*RCASFs*) [18] sont utilisés comme une technologie spécifique avec un but de créer un réseau réparti de noyaux de capteurs et actionneurs intelligents qui peuvent mesurer plusieurs paramètres intéressants. Toutes les données sont transmises en temps réel aux citoyens ou aux autorités concernés [16].

1.2.3.3 Les Réseaux Intelligents « *Smart Grids* »

Les Réseaux Intelligents (*RI*) sont des réseaux d'électricité qui, grâce à des technologies informatiques, ajustent les flux d'électricité entre fournisseurs et consommateurs. En collectant des informations sur l'état du réseau, les *Smart Grids* contribuent à une adéquation entre production, distribution et consommation. Il est nécessaire de différencier *Smart Grids* et compteur communicant ou « *Smart Meter* », qui renseigne le consommateur sur sa demande en électricité. « *Smart Grids* » est une appellation générale pour l'ensemble des technologies et des infrastructures « intelligentes » installées. Chez le particulier, le compteur communicant est une première étape dans la mise en place des *Smart Grids*. Les réseaux intelligents peuvent être définis selon quatre caractéristiques en matière de [16] :

- **Flexibilité** : ils permettent de gérer plus finement l'équilibre entre production et consommation.
- **Fiabilité** : ils améliorent l'efficacité et la sécurité des réseaux.
- **Accessibilité :** ils favorisent l'intégration des sources d'énergies renouvelables sur l'ensemble du réseau.
- **Economie :** ils apportent, grâce à une meilleure gestion du système, des économies d'énergie et une diminution des coûts (à la production comme à la consommation).

1.2.3.4 Les applications médicales

Grâce aux *RCASF*, l'*IdO* aura de nombreuses applications dans le secteur de la santé où l'objectif serait d'arriver à prévenir des situations graves et de suivre à distance des patients atteints des maladies chroniques tout en agissant rapidement si cela s'avère nécessaire. Des capteurs corporels implantés dans le corps du patient récoltent des informations relatives aux paramètres médicaux, tels que la température, la glycémie, le rythme des battements du cœur ou encore même la tension artérielle. Ces informations seront stockées et traitées sur internet (plus précisément sur un cloud) et mises à la disposition du médecin qui pourrait les consulter n'importe quand et depuis n'importe quel dispositif connecté à Internet (Smartphone ou tablette). Le médecin est alerté en temps réel (en lui envoyant un mail ou un *SMS*) de tout changement brusque concernant l'état de son patient. Suivant le degré de gravité de la situation, le médecin réagit soit en se déplaçant chez le patient ou juste en le contactant et lui indiquant ce qu'il faut faire pour revenir à l'état normal [17] [2].

1.2.3.5 Les applications militaires

L'IdO est un domaine fertile tant pour les applications civiles que pour les applications militaires. Dans le domaine de défense les capteurs et les nano-drones connectés à internet permettent d'envisager des applications sophistiquées pour l'exploration, la surveillance des champs de batailles et des frontières, ainsi que la poursuite et la localisation géographique des objets connectés. Les forces militaires ont la tendance d'utiliser des infrastructures propriétaires pour la connectivité et les communications. En transitant vers l'internet, il sera plutôt possible d'utiliser des infrastructures cloud, qui offrent une flexibilité opérationnelle très intéressante. Le soldat en mission peut lui-même être connecté à Internet à travers les capteurs connectés, intégrés dans sa tenue. Ces capteurs peuvent être par exemple des capteurs médicaux qui rapportent l'état de santé du soldat, ou des capteurs multimédias qui captent des images, une vidéo ou du son, depuis la zone où se trouverait ce soldat [17].

1.2.3.6 Sécurité et vie privée

Si l'*IdO* introduit plusieurs cas d'utilisation prometteurs, il pose aussi plusieurs problèmes quant à la sécurité et à la vie privée, comme résumé dans le tableau suivant [5][2] :

- **Vulnérabilité :** L'impossibilité d'utiliser les technologies modernes de sécurité (chiffrement, authentification, échange de clé, signature, etc.) rend les objets vulnérables aux attaques informatiques ce qui, en raison de leurs capacités à influer sur le monde physique, représente un danger pour les biens et les personnes.
- **Surveillance de masse :** En s'intégrant naturellement à l'environnement, les objets peuvent mesurer n'importe quelle information sans qu'il soit possible de savoir où, quand et par qui sont collectées ces données. En effet, même en l'absence d'intentions malveillantes de la part des propriétaires d'objets, l'utilisation massive des réseaux sans fil facilite les écoutes clandestines. Enfin, l'identification unique des objets permet de dresser des profils très détaillés de leurs propriétaires et de les suivre à grande échelle.

1.2.4 Visions de l'IdO

L'Internet des Objets émerge des travaux menés dans plusieurs domaines de recherche dont les différentes « visions » convergent vers un même but. Dans [18] une identification de trois visions majeures est présentée : une vision orientée objet, une vision orientée Internet et une vision orientée sémantique. Comme présenté sur la figure 1.4, chaque vision possède ses concepts et ses technologies clés.

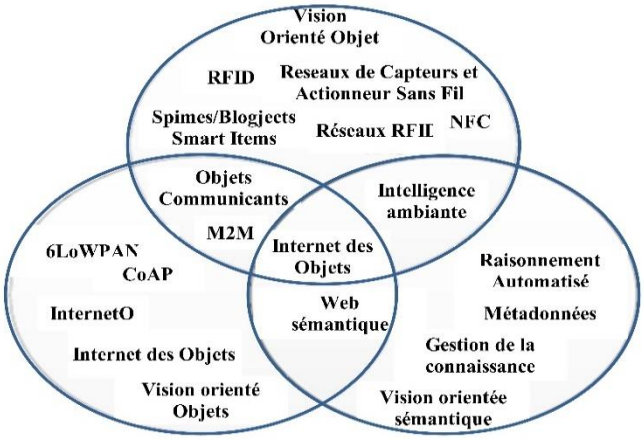


Figure 1.4 : Vision de l'Internet des Objets [18].

1.2.4.1 La vision orientée objet

Cette vision met l'accent sur les objets physiques et les systèmes qui y sont embarqués, avec notamment les problématiques suivantes [18] :

- Identifier les objets de manière unique et, de fait, les doter d'une identité propre.
- Permettre aux objets d'acquérir des informations sur leur environnement en utilisant des capteurs, pour ensuite échanger et traiter ces informations afin d'influer sur cet environnement au travers d'actionneurs.

Cette vision est principalement issue des travaux sur les réseaux d'objets identifiés par radiofréquence (*RFID*), la communication en champ proche (*Near Field Communication*, ou *NFC*) et les *RCASF*. L'Internet des objets est vu comme le concept émergeant de la mise en réseau des différents objets dans le but de faciliter les interactions entre les êtres humains et les objets, ainsi que les interactions entre objets eux-mêmes.

1.2.4.2 La vision orientée Internet

Elle suggère que cette interconnexion devrait se faire spécifiquement au travers du réseau Internet, considérant que celui-ci est déjà un « réseau de réseaux » qui connecte un très grand nombre de machines au moyen de protocoles standards conçus pour de nombreux usages.

Ainsi, une seule couche réseau, au sens *OSI* du terme, est utilisée pour connecter tous types d'objets, simplifiant de fait la fabrication et le déploiement de ces derniers, tout en assurant une certaine interopérabilité. Cette vision s'attache donc à étudier comment le protocole *IP* peut être adapté pour des systèmes embarqués caractérisés par de faibles ressources matérielles, notamment au travers de nouveaux standards. Par extension, si l'on connecte les objets physiques au réseau Internet, il devient possible de les intégrer aussi au Web pour bénéficier de ses avantages (*Web des objets*) [19].

1.2.4.3 La vision orientée sémantique

Elle se détache des problématiques techniques propres aux objets physiques et aux réseaux permettant de les faire communiquer, pour se concentrer sur la représentation, l'organisation et le stockage des données relatives à l'*IdO*, notamment [19] :

- Les identités de chaque objet physique (caractéristiques, états passés et présents, etc.) et les relations qui peuvent unir ces objets entre eux.
- Les flux d'informations acquis par ces objets sur leur environnement, par exemple au moyen de capteurs.

1.2.5 Les technologies de l'IdO

Avec les avancée récentes et l'augmentation des applications de l'*IdO*, plusieurs concepts technologiques sont apparus à l'objectif de répondre aux exigences de communications sans fil

de ces différentes applications. Dans cette section, nous allons citer quelques paradigmes de technologies existants selon leur portée radio (courte ou longue) et leur accès au canal avec ou sans licence. La figure 1. 5 illustre la classification de ces technologies.

1.2.5.1 Les technologies à longue portée

Avec les technologies à longue portée comme illustré par la figure 1.5, il y a deux classes de réseaux à savoir : les réseaux personnels *LPWAN* (*Low Personnel Area Network*) et les réseaux cellulaires. Avec les réseaux *LPWAN* on trouve la technologie *Sigfox*, *Neul* (par *Huawei*), et *LoRa*. D'un autre côté, les réseaux cellulaires (2*G*, *2.5G*, *3G*, *4G*, *5G*) sont eux même représentées par les technologies avec licence (comme *Lte-A*) et sans licence (comme *Lte-U*).

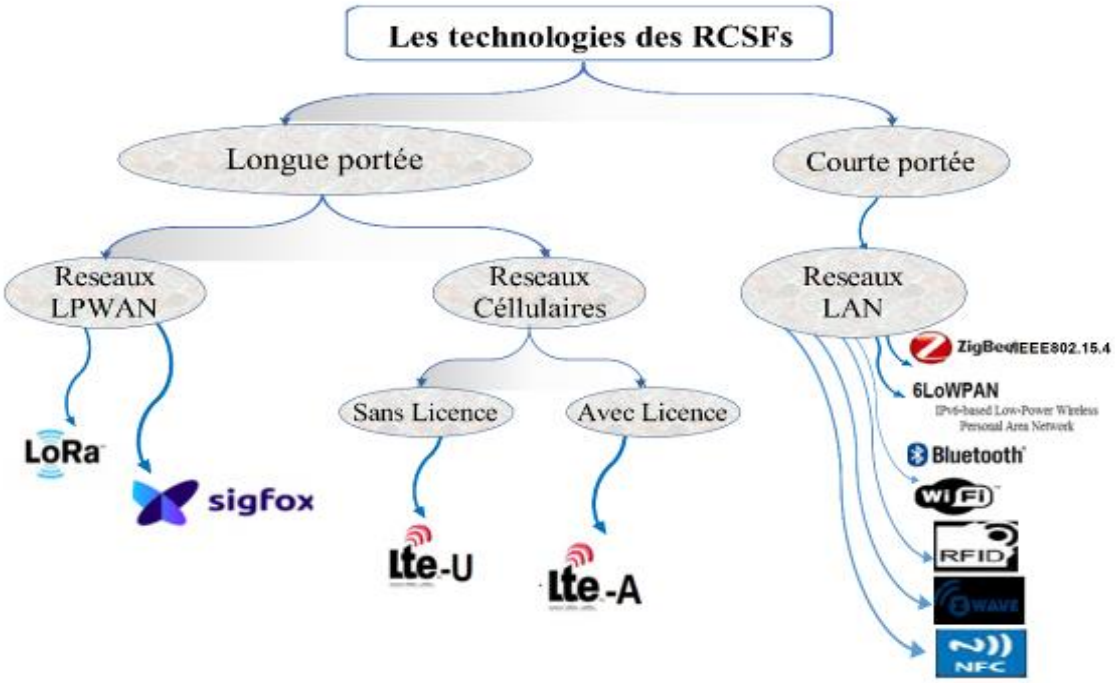


Figure 1.5 : Les Technologies des *RCSFs* [10].

- ***Sigfox*** [20] : Cette technologie propose une solution de connectivité *LPWA* (*Low Power Wide Area*) propriétaire. L'infrastructure se compose de stations de base déployées par des partenaires appelés *SNOs* (*SIGFOX Network Operators*). Les stations sont reliées à un ensemble de serveurs connectés à internet. Sigfox se positionne avant tout sur les capteurs industriels limitant sciemment la taille des messages à 12 bytes avec un maximum de 140 messages par jour, cette technologie autorise des messages plus importants (50 kbytes). Elle offre aussi davantage de bidirectionnalité, le débit descendant permettant des mises à jour logicielles ou de déclencher une action comme

couper un compteur suite à une alerte. Les fréquences radio utilisées par cette technologie sont ultra rapides et de longues portées, appelées bande son ultra-étroite ou *UNB* (*Ultra Narrow Band*). Les principaux avantages de la solution sont une utilisation très efficace du spectre de fréquences, des niveaux de bruits très bas et donc une meilleure sensibilité au niveau du récepteur, et une consommation très basse. En théorie, la distance séparant un objet d'une station de base peut atteindre 10 km en zone urbaine et 50 km en zone rurale. Parmi les cas d'applications actuelles de cette technologie, nous citons : Mesure de température Mesure de pression, Mesure de luminosité, Mesure de bruit, Comptage énergétique (eau, électricité, gaz), Détection de mouvements, Etat/Position (ouvert/fermé), Monitoring de batteries, suivi de places de parking, etc.

- ***LoRaWAN*** [21] : Cette technologie n'est pas nouvelle, elle est utilisée depuis plusieurs décennies dans le domaine militaire et spatial. *LoRaWAN* (*Long Range Radio Wide Area Network*), est une technologie de modulation des ondes radios sur laquelle sont basés les réseaux étendus à longue portée, comme elle fait aussi partie de la catégorie des réseaux à faibles consommation et à longue portée *LPWAN*. En effet, *LoRa* est un réseau ouvert (open source) naturellement bidirectionnel et crypté. Son principal intérêt réside dans l'augmentation de la portée et la résistance aux interférences. Cette technologie ne peut faire circuler que de petits paquets de données, émis par des capteurs de température ou d'humidité par exemple, fixés sur des objets connectés. Elle pourra faire transiter entre *0,3* et *50 kilobits* par seconde avec un débit de données variable entre *300 bps et 37,5 Kbps*. Un objet connecté en *LoRa* peut envoyer un message à une borne située à une distance d'environ *1 kilomètre* en zone urbaine et à *20 kilomètre*s dans une zone rurale.

- **Réseaux cellulaires** [22] **:** Les réseaux cellulaires sont optimisés pour le haut débit représenté par la technologie de *LTE*, où le mot < *LTE* > englobe l'évolution du réseau d'accès du système *UMTS* vers un réseau d'accès évolué appelé *E-EUTRAN*. L'accélération des réseaux cellulaires qui nourrissent nos smartphones se compte elle-aussi en « *G* » : *1G, 2G, 3G, 4G* et *5G*. Cette cinquième génération annoncée pour 2020 promet un débit de l'ordre de plusieurs gigabits par seconde, histoire de satisfaire notre appétence pour des médias de plus en plus volumineux, vidéos en tête. Dans les réseaux cellulaires se classent deux types de réseaux : les réseaux avec licence et sans licence. Le *LTE-U* (*Long Term Evolution*, « *U* » signifie « *unlicensed* ») est une technologie de réseaux sans licence. Par conséquent, cette technologie de réseaux mobiles s'appuie sur

l'utilisation d'un spectre de fréquence dit « sans licence ». *LTE-U* est utilisé d'une manière plutôt semblable à celle des 3G. La deuxième classe des réseaux cellulaires licenciés est représentée par la technologie *LTE-A* « *LTE-Advanced* » ou « réseau de téléphonie mobile de 4ème génération ». De tels réseaux se sont déployés mondialement.

1.2.5.2 Les technologies à courte portée :

Dans ce contexte, plusieurs technologies peuvent être utilisées (voir figure 1.5), à savoir *NFC* (*Near Field Communication*), *Bluetooth, Z-Wave, WiFi, ZigBee/IEEE 802.15.4* et *6LoWPAN*. Nous pouvons distinguer les trois technologies suivantes [23] :

- **Wi-Fi (IEEE 802.11a/b/g/n) [24]** : La technologie *Wi-Fi* (*IEEE 802.11*) permet la connexion d'un réseau local sans fil. Elle est disponible en plusieurs types : a, b, g et n. et plus récemment, les normes *AC* et *AD*. La différence entre ces types tourne essentiellement autour du débit maximal qu'un dispositif connecté puisse atteindre et la portée (la distance maximale possible entre un dispositif connecté et le point d'accès). *Wi-Fi* utilise la bande de fréquence *ISM* (*Industrial, Scientific and Medical*) *2.4 GHz* (à licence gratuite) ou la bande *5GHz*. Cette technologie est caractérisée par un débit théorique nettement élevé allant de 11Mb/s (pour *IEEE 802.11b*) jusqu'à *54Mb/s* (pour *IEEE 802.11a, g*). L'avantage du *WiFi* est qu'il est couramment utilisé et donc, les nœuds capteurs peuvent être facilement connectés aux réseaux *WLAN* (*Wireless Local Networks*) existants. A côté de ces avantages, cette technologie est inappropriée pour les réseaux de capteurs standards, en raison de la forte consommation d'énergie induite, et la complexité de sa pile protocolaire. Cependant, certains types de réseaux de capteurs, comme les réseaux de capteurs multimédias, exigent un débit effectif relativement important. Pour satisfaire à cette exigence, on fait appel à la technologie *WiFi*.

- ***Bluetooth* (*IEEE 802.15.1*)** [25] **:** La technologie Bluetooth a été initiée en 1994 et actuellement gérée par le groupe *SIG* (*Special Interest Group*), elle a été standardisée sous la norme *IEEE 802.15.1*. *Bluetooth* est conçu pour fonctionner sur des appareils à faible puissance et faible consommation d'énergie, il a comme but la mise en œuvre des réseaux à portée personnelle où le transfert de données se fait par un débit moyen.

- ***ZigBee/IEEE 802.15.4 et 6LoWPAN*** [26] : ce sont les deux technologies les plus utilisées actuellement dans l'internet des objets, beaucoup plus de détails concernant ces normes, seront présentés dans la section 1.5.

1.3 Contexte technologique et motivation

Pour la création et la meilleure gestion des réseaux intelligents avec l'*IdO*, les réseaux de capteurs sans fil (*RCSF*) sont utilisés comme une technologie de réseaux spécifiques répondant aux exigences de certaines applications dans l'*IdO*, avec l'objectif de créer un réseau réparti de noyaux de capteurs intelligents qui peuvent mesurer plusieurs paramètres intéressants. Toutes les données sont transmises en temps réel aux utilisateurs concernés.

Le nombre croissant des applications dans l'*IdO* nécessitent l'utilisation d'un autre type de réseaux tels que les Réseaux de Capteurs Actionneurs Sans Fil (*RCASFs*), qui représentent la classe des réseaux hétérogènes puisqu'ils sont constitués d'au moins deux deux types de nœuds : nœuds capteurs et actionneurs.

1.3.1 Des réseaux homogènes aux réseaux hétérogènes

Bien que les *RCSFs* soient utilisés dans de plusieurs applications, nous constatons l'apparition de nouvelles applications où le besoin d'un nouveau composant du réseau appelé actionneur se fait sentir. Cette extension des réseaux de capteurs inclut les nœuds actionneurs qui ne sont pas considérés seulement comme responsables pour agir sur l'environnement mais, d'un point de vue réseau, ils sont considérés comme des points de collecte locaux. Cette architecture est appelée réseau sans fil de capteurs et d'actionneurs (*RCASFs* ou *WSANs Wireless Sensors and Actuators Networks*) [27]. La différence majeure entre les *RCSFs* et *RCASFs* est que ces derniers sont des réseaux hétérogènes par nature.

Dans la section suivante nous commençons par introduire un bref aperçu sur les réseaux hétérogènes (*RCASF*), leurs caractéristiques et architectures. Par la suite, Nous décrivons en détail les réseaux homogènes (*RCSFs*). Enfin, nous nous concentrons sur les défis pour les *RCSFs* au niveau du contrôle du réseau et plus particulièrement les challenges de la tolérance aux pannes par de nœuds mobiles via de solutions de routage.

1.3.1.1 Les Réseaux de Capteurs Actionneurs Sans Fil (*RCASF*)

Un *RCASF* [38] est un réseau hétérogène par nature, il est constitué de nœuds capteurs et de nœuds actionneurs comme illustré dans la figure 1.6. Les nœuds capteurs collectent les données de l'environnement de déploiement, et les nœuds actionneurs sont responsables de deux fonctionnalités, 1) Agir sur l'environnement physique, 2) Recueillir les données capturées par les nœuds capteurs.

Nous nous intéressons dans cette thèse à cette dernière fonctionnalité, prise en charge par les actionneurs et représentant le rôle de points de collecte locaux, ainsi qu'aux communications sans fil entre les nœuds capteurs et les nœuds actionneurs.

1.3.1.2 Architectures de communication *RCASF*

Un *RCASF* est un réseau auto-organisé et autonome répondant aux exigences des applications visées par ce réseau. Les nœuds de capteurs et d'actionneurs d'un *RCASF* sont statiques dans certaines applications. Toutefois, il existe des applications spécifiques qui supposent la mobilité des nœuds actionneurs. L'architecture des *RCASFs* est schématisée en trois types d'architectures de communications illustrée dans la figure 1.6 (a), (b) et (c). Deux architectures de base [28] : 1) architecture appelée semi-automatique, 2) une architecture automatique, et 3) une architecture dite coopérative.

- **Architecture semi-automatique** : dans cette architecture, les capteurs envoient leurs données vers la Station de Base (*SB*), cette dernière choisit elle-même l'actionneur le plus approprié pour réceptionner ses données (figure 1.6 (a)).
- **Architecture automatique** : les capteurs dans cette architecture envoient les données directement vers les actionneurs qui traitent toutes les données entrantes, initient les actions appropriées et/ou vont faire suivre ces données vers la *SB* (figure 1.6 (b)).
- **Architecture coopérative :** Avec cette architecture, les capteurs transmettent les données vers les nœuds actionneurs en multi-sauts. Les actionneurs analysent les données et peuvent consulter la *SB* avant d'entreprendre toute action. Les actionneurs peuvent utiliser leur réseau point-à-point pour prendre des décisions et des mesures d'action, ou peuvent simplement informer la *SB* et attendre de nouvelles instructions de la part de cette dernière (Figure 1.6 (c)).

1.3.1.3 Caractéristiques des RCASFs

Les deux architectures de base et l'architecture automatique possèdent les caractéristiques principales suivantes [27] :

- **Faible temps de latence :** Les informations détectées sont transmises par des capteurs directement aux actionneurs vu qu'ils peuvent être proches géographiquement les uns des autres.
- **Durée de vie du réseau plus longue :** Pour l'architecture semi-automatique, les capteurs autour du puits sont susceptibles de consommer plus d'énergie que les autres

nœuds dans le réseau, car pouvant avoir une plus grande charge pour relayer les messages à destination de *SB*.

- **Richesse en ressources :** car ayant une grande capacité de calcul, de mémoire et une puissance de transmission importante. Toutefois, cette coexistence entre des nœuds capteurs à faibles ressources et les nœuds actionneurs riches en ressources introduit de nouveaux challenges liés à l'hétérogénéité.

1.3.1.4 Protocole de routage dans *RCASFs*

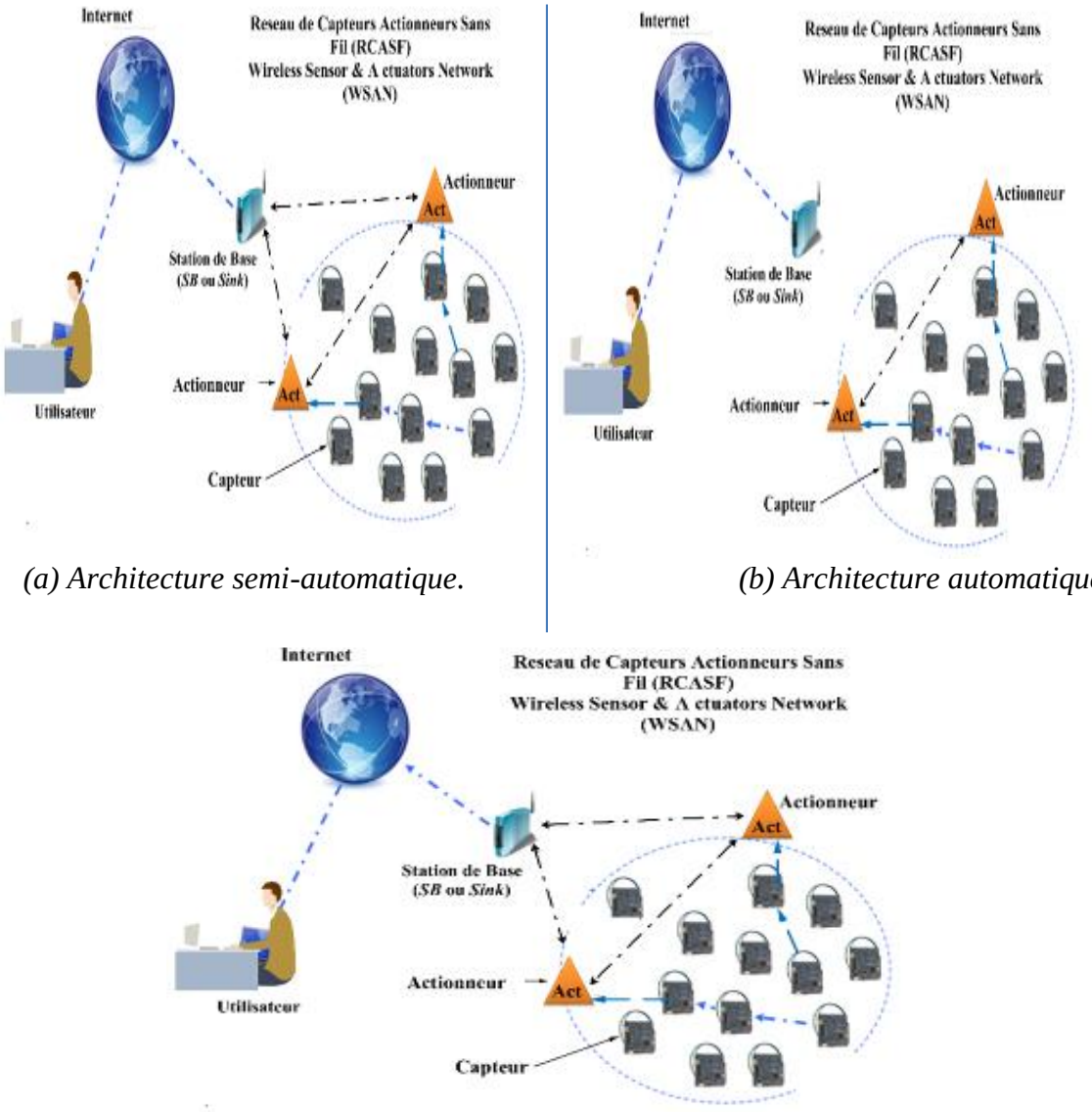
Un protocole de routage est une clé importante dans un réseau de communication. En effet, son rôle est d'assurer l'acheminement des messages entre une source et une destination. Les caractéristiques spécifiques entre *RCSFs* et les *RCASFs* sont différentes. De plus, l'absence de contrôle centralisé dans les *RCASFs* et les capacités limitées au niveau des nœuds capteurs compliquent la tâche du routage.

Il est à noter également que la coexistence entre les nœuds capteurs, à faibles ressources, et les nœuds actionneurs, riches en ressources, a un impact sur les performances des protocoles de routage. En effet, l'hétérogénéité au niveau des portées de transmission des nœuds implique l'existence des liens asymétriques. Ces liens peuvent dégrader les performances des protocoles de routage qui ne prennent pas en considération ce type de liens [27].

Cependant, Le problème de routage demeure un problème délicat. Le domaine de routage dans les *RCSFs* et *RCASFs* a bien attiré l'attention du monde scientifique pendant la dernière décennie jusqu'à atteindre un état de maturité permettant au groupe de travail *ROLL* [6] (*Routing Over Low power and Lossy networks*) de proposer un protocole en cours de standardisation.

A la suite de cette section, nous allons nous concentrer sur la description des *RCSFs* comme contexte principal de cette thèse, leurs caractéristiques et les protocoles de routages. En effet, le groupe de travail *ROLL* a été créé pour normaliser un protocole de routage pour ces types de réseaux appelé *RPL* (*Routing Protocol for LLNs*) [5].

La principale raison qui a été derrière la création du groupe *ROLL* est que les exigences des réseaux *LLNs* (*Low* Power and *Lossy Networks*) ont évolué à un point où des solutions standardisées pour les réseaux ad-hoc ne s'appliquent plus. Dans [27], les auteurs montrent qu'aucun protocole de routage standard au sein de *l'IETF* (à savoir : *OSPF*, *OLSRv2*, *AODV*, *DYMO* et *DSR*) ne peut répondre aux exigences des réseaux *LLNs*. Il est à noter que ces réseaux *LLNs* couvrent bien les *RCSFs* et les *RCASFs*.



(a) Architecture semi-automatique. *(b) Architecture automatique.*

(c) Architecture coopérative.

Figure 1.6 : Les architectures de communication dans les *RCASFs* (a), (b) et (c) [27].

1.4 Les Réseaux de Capteurs Sans Fil (*RCSF*s)

1.4.1 Définition

Les réseaux de capteurs sans fil (*RCSFs*) [3] représentent une révolution technologique qui change radicalement la façon de concevoir les systèmes de surveillance de très grandes échelles. En effet, les avancées récentes dans la micro-électronique et la communication sans fil ont permis le développement des capteurs de plus en plus petits, à faible coût, fonctionnant sur batterie et caractérisés par leurs ressources limitées en termes de bande passante, débit, énergie, zone de couverture, capacités de traitement et de stockage. Actuellement, les nœuds *RCSF* ont

tendance à intégrer le monde d'internet pour profiter de ses atouts dans le cadre des technologies, *IdO* et *Web des Objets* « *WoT* '*Web of Things*' » [29] [30] [31].

- **Un capteur :** est un dispositif qui transforme une grandeur physique observée (température, pression, humidité, etc.) en une grandeur utilisable (intensité électrique, position d'un flotteur) [3]. Pour cela, il possède au moins un transducteur dont le rôle est de convertir ce type de grandeurs.
- **Un capteur intelligent :** Le terme capteur intelligent (« smart » ou « intelligent sensor ») a été utilisé dans l'industrie des capteurs pour désigner des capteurs qui ne fournissent pas seulement des mesures, mais aussi une fonctionnalité aux mesures spécifiques à savoir : traitement de données collectées, calcul, communication numérique. En effet, un capteur intelligent intègre de nombreux éléments électroniques additionnels, ainsi que des unités programmables et des aspects logiciels nécessaires au traitement des données, aux calculs et à la communication numérique par rapport à un capteur classique [3].

1.4.2 Architecture d'un RCSF

L'architecture d'un réseau de capteurs sans fil est une architecture d'infrastructure sans fil à multi-sauts. La station de base (*SB*) appelée aussi nœud puits (ou *Sink*) peut être fixe ou mobile. Elle joue le rôle d'un relai (intermédiaire) entre un nœud capteur et l'utilisateur [32]. Comme le montre la figure 1.7, un réseau *RCSF* est défini comme une collection de nœuds de capteurs qui s'auto-organisent en un réseau sans fil multi-sauts de manière spontanée, qui collaborent entre eux pour détecter et/ou contrôler certaines données de paramètres physiques (telles que la température, l'humidité, les battements cardiaques, etc.) et qui sont traités et transmis de manière autonome à un point de collecte (*SB* ou *Sink*) avec une connectivité IP pour l'accès distant.

1.4.2.1 Anatomie d'un nœud capteur

La principale tâche d'un nœud capteur dans un *RCSF* est de collecter des données. En plus de cette tâche initiale, un capteur est doté d'une unité de traitement embarquée, à faible coût avec des capacités de calcul, de mémoire et de communication. La capacité d'un capteur est néanmoins limitée à cause, en partie, de la miniaturisation de ses composants. Un capteur se compose en général de quatre unités fondamentales (voir la figure 1.8) [33].

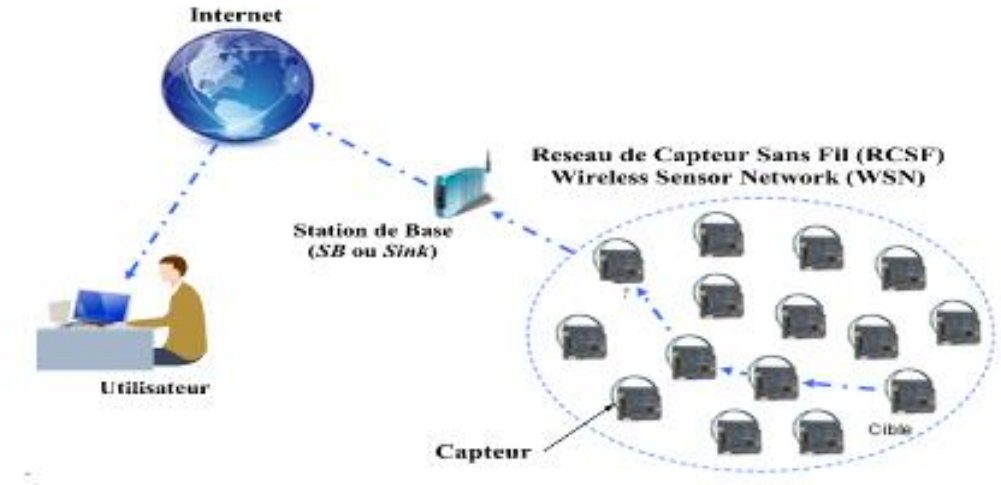


Figure 1.7 : Architecture des réseaux de capteurs sans fil.

- **L'unité d'acquisition** (ou de capture) : Elle est généralement composée de deux sous-unités, un capteur embarqué sur le nœud et un convertisseur analogique-numérique (*CAN*). Le capteur permet d'assurer la collecte des données de l'environnement. Les signaux provenant du capteur sont convertis à l'aide du convertisseur *CAN* en signaux compréhensibles par l'unité de traitement.
- **L'unité de traitement** : Elle est généralement composée d'une petite unité de stockage pour conserver les données collectées, et d'un processeur associé à un microcontrôleur pour exploiter et traiter les données. Il s'agit du module principal du capteur du moment qu'il contrôle le bon fonctionnement des autres unités.
- **Unité de transmission** (ou de transmission) : Elle est responsable de toutes les émissions et réceptions de données via un support de communication radio. Elle permet au capteur de communiquer en sein du réseau à l'aide d'un couple émetteur/récepteur qu'on appelle aussi transceiver. Elle utilise essentiellement des technologies sans fil tels que le *802.11* et le *802.15.4*. Ces trois unités sont alimentées par une batterie comme le montre la figure 8 ci-dessous.
- **Unité d'alimentation** : il s'agit de la source d'énergie qui alimente toutes les unités du capteur. Elle correspond généralement à une batterie ou une pile dont les ressources limitées représentent l'une des contraintes majeures des *RCSF*.

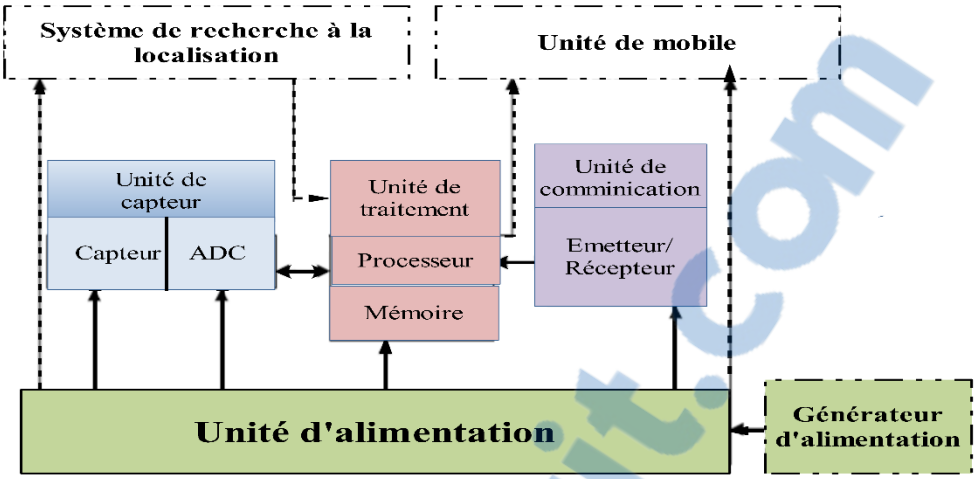


Figure 1.8 : Architecture d'un nœud capteur.

1.4.2.2 Les différents types de nœuds

Selon le type de réseaux de capteurs sans fil, ainsi que l'application et la structure choisie, un *RCSF* peut contenir différents types de nœuds de capteurs.

- **Nœud régulier :** est un nœud doté d'une unité de transmission et d'unité de traitement de données comme décrit par la figure 1.8. Selon le domaine d'application, un nœud peut être équipé d'unités supplémentaires ou optionnelles comme un système de localisation *GPS* (*Global Positionning Système*, etc.) pour déterminer sa position, ou bien un système générateur d'énergie (cellule photovoltaïque, etc.), ou encore un système mobile pour lui permettre de changer sa position ou sa configuration en cas de nécessité.
- **Nœud capteur ou nœud de source** : C'est un nœud de capteur équipé de deux unités fondamentales, décrit dans la figure 1.8. Il contient l'unité d'acquisition dotée d'un ou plusieurs capteurs et d'un convertisseur. Ce nœud peut être un nœud capteur classique ou un capteur intelligent.
- **Nœud actionneur :** Il peut être un robot car considéré comme un nœud régulier doté d'une unité lui permettant d'exécuter certaines tâches spécifiques comme des tâches mécaniques (se déplacer, combattre un incendie, piloter un automate, etc.).
- **Un nœud puit (Sink) :** est un nœud régulier doté d'un convertisseur série connecté à une seconde unité de communication (*GPRS- Global Packet Radio* Service), *Wifi*, *WiMax* (*Worldwide Interoperability for Microwar Access*, etc.). La seconde unité de

communication fournit une retransmission transparente des données provenant de nœuds capteurs à un utilisateur final ou à d'autres réseaux comme Internet.

- **Nœud passerelle (Ou *Gateway*) :** est un nœud régulier permettant de relayer le trafic dans le réseau sur le même canal de communication.

1.4.3 Systèmes d'exploitation des *RCSFs*

Afin de développer davantage le caractère intelligent des capteurs dans les applications récentes de nos jours, un système d'exploitation est embarqué au sein d'un capteur. Un système d'exploitation pour capteur en réseau (en anglais *Operating System ou OS*) est un ensemble de programmes responsables de la liaison entre les ressources matérielles d'un dispositif et les applications de l'utilisateur. Il existe plusieurs systèmes d'exploitation connus pour les réseaux de capteurs à savoir : *TinyOS [34], Contiki* [35], *SOS* [36], *FreeRTOS* [37], *Mantis OS* [38], *Nut/OS* [39].

Nous présentons dans ce qui suit les systèmes d'exploitation les plus utilisés dans les *RCSFs* et qui ont été proposés dans la littérature :

- ***TinyOS*** : *TinyOS* (*Tiny Operating System*) [34] , est un système d'exploitation "open source" pour les *RCSFs* développé par l'université Berkeley en Californie (*USA*), dans le but de réduire au maximum la taille d'allocation mémoire nécessaire à son installation et à son fonctionnement. La conception de *TinyOS* a été entièrement réalisée en *NesC*, langage orienté composant dérivé du C pour les systèmes embarqués [40]. Ce système d'exploitation est structuré en plusieurs composants avec interfaces bidirectionnelles pour permettre l'extension des capteurs par d'autres fonctionnalités. *TinyOS* repose sur : une architecture basée composant, un modèle de programmation basé sur des événements et un modèle de concurrence basé sur des événements et des tâches. La bibliothèque de composants de *TinyOS* est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données. Un programme exécuté sous *TinyOS* ne contient que des composants nécessaires à son exécution, cela permet de réduire la taille du programme à insérer dans l'unité de traitement du capteur. *TinyOS* opte pour une programmation, pour prolonger la durée de vie du capteur. Dans cette programmation (événementielle), l'exécution des instructions s'effectue en fonction des événements enregistrés par l'unité de traitement. L'avantage de cette technique est de laisser les capteurs en mode veille dans le cas où il n'y a aucun événement. Un certain nombre de plateformes sont directement

programmables comme par exemple les *tmote* ou les *MicaZ* (ces deux modèles sont compatibles avec *ZigBee*). *TOSSIM* est un simulateur de capteurs pour les programmes *TinyOS*, tout programme en *NesC* peut être compilé de manière à être exécuté dans *TOSSIM*, ce qui permet de simuler le comportement d'un ou plusieurs capteurs ainsi de les programmer. Les capteurs qui utilisent le système d'exploitation *TinyOS* sont essentiellement développés par la société *Xbow*, comme les capteurs *Mica2, MicaZ, TelosB, TelosA*, etc. [41]

- ***Contiki :*** *Contiki* [35] est un système d'exploitation open-source, léger, flexible, multitâche développé pour les systèmes embarqués et écrit en langage C. Il propose les principales caractéristiques et fonctionnalités d'un système d'exploitation tout en favorisant une consommation énergétique et une empreinte mémoire minimales. *Contiki* a introduit l'idée d'utiliser la communication *IP* dans des réseaux de capteurs basse consommation, il contient deux piles de communication : *uIP* et *Rime* :

– *UIP* est une petite pile de *TCP/IP RFC-CONFORME* qui permet à *Contiki* de communiquer sur Internet.
– Rime est une pile de communication légère conçue pour des radios de basse puissance.

Ses principaux avantages sont le support des protocoles *IPv6* et *6LoWPAN* et sa portabilité. *Contiki* est considéré comme le futur *OS* pour le déploiement d'un *IdO* fiable, robuste et efficace. Il fournit une vaste gamme de communications primitives.

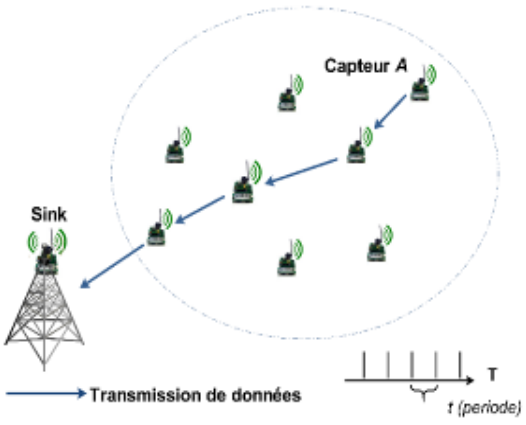
1.4.4 Classification des *RCSF* selon le modèle de surveillance

La collecte et la transmission des données entre nœuds capteurs dans un *RCSF* peuvent, par nature, être déclenchées dans plusieurs modèles de surveillance, classifiés de la manière suivante :

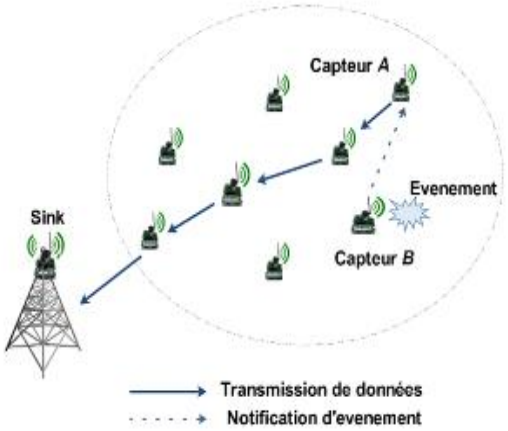
- **Surveillance périodique** : Tous les capteurs analysent leurs environnements à des intervalles réguliers et envoient périodiquement leurs mesures au nœud récepteur *SB* (voir figure 1.9 (a)). Le type d'application visé concerne les applications de surveillance dont le but principal est de récupérer des informations de manière régulière depuis la zone surveillée [42].

- **Surveillance événementiel :** Les capteurs analysent leurs environnements et les données sont détectées lorsqu'un événement se produit soudainement (voir figure 1.9 (b)), tel le dépassement d'un seuil par exemple. Un message d'alerte est alors envoyé sur le réseau [42]. Comme le montre la figure 1.9 (b), un capteur *B* détecte un événement

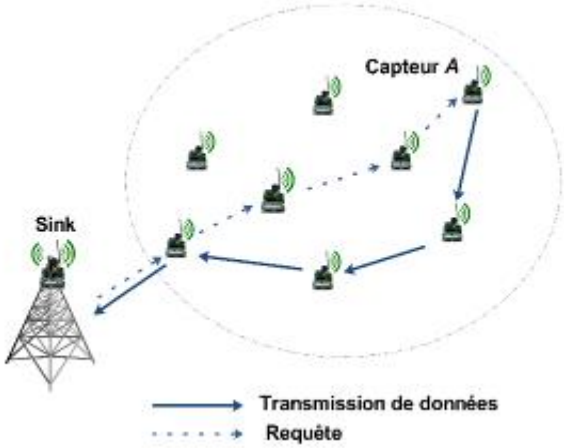
déclenché par un objet qui traverse sa zone de perception. Sachant que le capteur A couvre cette région, Le nœud B envoie alors un message de notification au nœud A. Finalement, le nœud A analyse et collecte les données et les transmet vers le nœud *Sink* (ou *SB*).



(a) Surveillance périodique.



(b) Surveillance évènementielle.



(c) Surveillance événementielle.

Figure 1.9: Les principaux modèles de surveillance (a), (b) et (c).

- **Surveillance par requête** : Les capteurs envoient les données collectées seulement après avoir reçu des requêtes explicites du *SB* (voir figure 1.9 (c)). C'est une collecte sur demande, donc une requête est envoyée au nœud concerné pour qu'il détecte et envoie les données requises. Ces requêtes peuvent être dirigées (sollicitation d'un capteur ou un groupe de capteurs) ou diffusées (requête de type conditionnel) [42].

Ces trois modalités selon la figure 1.9, ne sont bien évidemment pas exclusives, un même *RCSF* peut utiliser les trois types de fonctionnement selon des besoins d'un domaine

d'application, par exemple pour le e-santé dont le *RCSF* est destiné à monitorer les signes vitaux d'un patient sous surveillance (un malade cardiaque) [42].

- **Surveillance par application hybride** : Où l'on retrouve simultanément l'ensemble des différents modèles de surveillance cités auparavant.

1.4.5 Les caractéristiques et contraintes conceptuelles des *RCSFs*

Avec la tendance d'exploiter les *RCSF* et de profiter de ses atouts avec le concept d'*IdO*, dont les applications des *RCSF* devraient supporter la mise en œuvre et le fonctionnement de base d'un réseau de capteur, la *QdS* et la sécurité, deviennent désormais des exigences fondamentales pour la transmission dans un environnement contraint en ressources. En effet, il existe plusieurs caractéristiques et contraintes pour la conception des *RCSFs*, nous en citerons les plus importantes [42] :

- **Sources d'énergie :** L'un des principaux challenges qui font face à la conception de toute technologie liée aux RCSF et qui marquent ce type de réseaux est la pénurie des ressources d'énergie. En effet, Les capteurs sont limités par des ressources en termes de calcul, de stockage et d'autonomie d'énergie, car l'alimentation des capteurs sans fil est généralement assurée par de simples piles de capacités limitées. En pratique, le remplacement des piles n'est pas toujours une opération facile, ce remplacement s'avère même impossible dans certains cas de figure, comme le déploiement des nœuds *RCSF* dans des zones hostiles ou inaccessibles pour les êtres humains. Toutes les applications des *RCSF* : le routage [43], le maintien de la connectivité [45], etc. utilisent des techniques pour assurer une conservation et une bonne gestion d'énergie, car cette dernière influe directement sur la durée de vie des nœuds capteurs et donc sur le réseau global. La consommation d'énergie est une contrainte extrêmement forte dans les *RCSFs* [44], bien plus que dans n'importe quel autre type de réseau. Cette contrainte influe directement et lourdement sur la conception des *RCSF* et de leurs nœuds.

- **Bande passante :** Certains types de données nécessitent une bande passante très élevée pour leur transmission (par exemple le streaming de flux vidéo) par rapport à la capacité actuelle des capteurs. En effet, La bande passante découlant d'un nœud capteur est aussi très faible : 10 à 250 kb/s selon le standard *IEEE 802.15.4* [46], car les *RCSF* s'inscrivent dans la catégorie des réseaux personnels WPAN « *Wireless Personal Area Network* ». Pour le flux vidéo, par exemple, cela nécessite des débits supérieurs à l'ordre de 500 kbps. Des études ont montré que le débit réel utile est de l'ordre de 70 kbps [46]. Par

conséquent, des solutions matérielles et logicielles sont nécessaires pour une fourniture suffisante en bande passante afin de supporter la transmission de flux à contrainte temps réel.

- **Auto-configuration** : Ce mécanisme est employé dans différents aspects des *RCSF*. Le nœud capteur devrait être capable d'adapter ses paramètres de service pour tenir compte des défaillances des autres nœuds, des obstacles et de l'ajout de nœuds au réseau. Les nœuds capteurs dans un *RCSF* sont déployés aléatoirement, un domaine où aucune intervention humaine n'est envisageable, alors il est nécessaire de maintenir une structure qui répond de manière efficace aux besoins de l'application. Ceci peut être réalisé grâce à l'auto-configuration des capteurs, sans contrôle centralisé, et uniquement avec des interactions locales entre les capteurs. Pour répondre à ces différents problèmes, quatre notions liées à l'auto-configuration sont mises en avant dans [47] comme caractéristiques souhaitables pour les *RCSF*.

- *Auto-organisation (Self-organization)* : est la capacité d'adapter les paramètres de la configuration du réseau en fonction de son état et de son environnement,
- Auto-optimisation (*Self-optimization*) : est la capacité de monitorer et d'optimiser l'utilisation des ressources limitées du réseau.
- Auto-protection (*Self-protection*) : est la capacité de reconnaître les intrusions, les attaques et de s'en protéger.
- Auto-réparation (*Self-healing*) : est la capacité de découvrir, d'identifier la cause et de réagir aux pannes et défaillances du réseau.

- **Topologie dynamique :** les capteurs peuvent être attachés à des objets mobiles qui se déplacent de façon libre et arbitraire rendant ainsi la topologie du réseau fréquemment changeante à cause des nœuds capteurs mobiles. Par conséquent, il faudrait que ces derniers soient capables de communiquer et de collaborer entre eux en utilisant des protocoles d'auto-organisation afin de maintenir la topologie du réseau.

- **Latence :** la sensibilité aux délais est également une préoccupation majeure dans les applications multimédias temps réel. Les données émises doivent être correctement reçues au Sink dans les délais exigés par l'application, sinon elles deviendront obsolètes. Les applications de streaming vidéo temps réel par exemple, requièrent des garanties strictes de délai bout à bout, de bande passante et de gigue [3].

- **Scalabilité :** Les applications de surveillance dans des environnements hostiles peuvent nécessiter un déploiement très dense des capteurs. Lorsque le nombre de capteurs augmente, ceci ne doit en aucun cas influencer sur les performances du réseau. Les différents algorithmes et protocoles proposés pour les *RCSF* doivent prendre en considération le facteur de passage à l'échelle.

- **Sécurité** : Concernant les applications de surveillance qui nécessitent un niveau de sécurité plus élevé, telles que les applications militaires il faudrait que les données collectées par les capteurs soient sécurisées. Des mécanismes d'authentification, de confidentialité, et d'intégrité peuvent être utilisés dans ce type d'applications. Cependant, les algorithmes de cryptographie conçus pour les réseaux de capteurs doivent tenir compte des ressources restreintes des capteurs vidéo [48].

- **Les médias de transmission** : Dans un réseau de capteurs, les nœuds sont reliés par une architecture sans-fil. Pour permettre des opérations sur ces réseaux dans le monde entier, le média de transmission doit être normé. On utilise le plus souvent l'infrarouge (qui est license-free, robuste aux interférences, et peu onéreux), le *Bluetooth* et les communications radio *ZigBee*.

- **Couverture :** La couverture peut être considérée comme l'une des métriques de la *QdS* d'un réseau de capteurs. Une mauvaise répartition des capteurs vidéo peut rendre certains nœuds inutilisables s'ils ne sont plus en mesure de communiquer avec les autres capteurs pour atteindre à la fin la *SB*. La stratégie la plus adoptée dans la problématique de couverture est l'exploitation de la redondance issue du déploiement aléatoire des capteurs vidéo et la prise en charge de leur mise en veille en alternance. L'intérêt de cette technique adoptée dans cette thèse est de prolonger la durée de vie du réseau et d'assurer également une tolérance aux pannes dans le *RCSF*.

- **Les types de communication** : Il existe différents types de communication utilisés dans les *RCSF* [49] :

  - *Unicast* : Ce type de communication est utilisé pour échanger des informations entre deux nœuds sur le réseau.
  - Broadcast : La station de base transmet des informations vers tous les nœuds du réseau. Ces informations peuvent être des requêtes de données bien précises (ex : la température dans une région de surveillance), des mises à jour de programmes ou des paquets de contrôle.

- *Local Gossip* **:** Ce type de communication est utilisé par des nœuds situés dans une région bien déterminée et qui collaborent ensemble afin d'avoir une meilleure estimation de l'événement observé et d'éviter ainsi l'émission du même message vers le nœud « SB », chose qui contribue à consommer moins d'énergie.
- *Convergecast* : Il est utilisé dans les communications entre un groupe de nœuds et un nœud bien spécifique (qui peut être la « *SB* ou *Sink* »). L'avantage de ce type de communication est la diminution de contrôle d'en-têtes des paquets (« *Control Overhead* »), ce qui économise l'énergie au niveau du nœud récepteur.
- *Multicast* **:** Il permet une communication entre un nœud et un groupe de nœuds. Ce type de communication est utilisé dans les protocoles qui incluent le « *clustering* » dans lesquels, le « *Clusterhead* communique avec un groupe de nœuds.

- **La collaboration entre les nœuds :** Les contraintes strictes de consommation d'énergie mènent les nœuds capteurs à détecter et à traiter les données de manière coopérative afin d'éviter le traitement redondant d'une même donnée observée, source de perte d'énergie.

- **Qualité de Service (*QdS*) :** Dans un contexte d'applications critiques, la capacité à exploiter les informations visuelles d'une image à la réception est capitale. Pour ce faire, la vitesse de capture associée à chaque capteur doit être satisfaisante. De plus, et afin de réduire la quantité de données visuelles collectées, nous devons faire appel à la technique de compression d'images. Cette dernière peut s'effectuer de plusieurs manières : avec ou sans perte de données, centralisée ou distribuée. La compression avec perte de données demeure la plus pertinente pour des transmissions à bas débit et ce, afin de réduire les coûts en termes de mémoire et de transmission. Malgré le fait que la compression participe à l'amélioration des performances du réseau, un taux de compression maximal pourrait dégrader fortement la qualité de l'image restituée [50] [48].

- **Tolérance aux pannes :** La défaillance de certains capteurs peut être la conséquence de l'épuisement de leur batterie. Cela pourrait arriver suite à un endommagement physique du capteur dans l'environnement de surveillance (après un largage à partir d'un avion, par exemple). Ces problèmes n'affectent pas le reste du réseau, c'est le principe de la tolérance aux pannes. Cette tolérance aux pannes est la capacité de maintenir les fonctionnalités du réseau sans interruption lorsqu'un capteur ou plusieurs capteurs cesse(ent) de fonctionner. En effet, le *RCSF* doit pouvoir remédier aux

défaillances d'une partie de ses capteurs et continuer à fonctionner même avec des performances réduites. La tolérance est la capacité de soutenir les fonctionnalités d'un réseau de capteurs sans causer d'interruption lorsqu'un capteur cesse de fonctionner. Cette tolérance aux pannes dépend aussi de l'environnement de déploiement du *RCSF*, dans le cas où le réseau est déployé dans un habitat, la tolérance exigée peut être basse car le degré d'endommagement des capteurs est réduit.

Cependant, si le *RCSF* est déployé dans un environnement hostile de surveillance, la tolérance doit être plus importante à cause de la criticité de l'application [50]. Un mécanisme de gestion et de prise en charge de pannes doit être prévu et mis en place.

1.4.6 Les topologies des RCSFs

Le déploiement des nœuds capteurs, leur connectivité, et leur connectivité, et la station de base peuvent être principalement organisés en deux topologies réseau : topologie en étoile à l'instar d'un réseau cellulaire et topologie multi-sauts à l'instar des travaux menés par le groupe de travail *IETF ROLL* [6]. Actuellement, les nœuds *RCSFs* sont intégrés dans le monde d'Internet pour profiter de ses atouts dans le cadre des technologies, *IdO* et *WdO* [29] [31].

1.4.6.1 Topologie plate

Avec la topologie plate, un *RCSF* est un réseau homogène (voir figure 1.10), où tous les nœuds capteurs sont identiques en termes de batterie et de complexité du matériel, excepté le *Sink* (*SB*) qui joue le rôle de passerelle. En effet, le *Sink* (*SB*) est responsable d'en transmettre les données collectées à l'utilisateur final. Selon le service et le type de capteurs, une densité de capteurs élevée (plusieurs nœuds capteurs/m2) et une communication multi-saut peuvent être nécessaires pour un *RCSF* en topologie plate.

En présence d'un très grand nombre de nœuds capteurs, la scalabilité devient critique. Le routage et le contrôle d'accès au médium (*MAC*) doivent gérer et organiser les nœuds de manière très efficace en termes d'énergie. Alors que dans une topologie maillée, les nœuds capteurs ne se chargent pas que de la tâche de captage de données, se préoccupant tout aussi du routage.

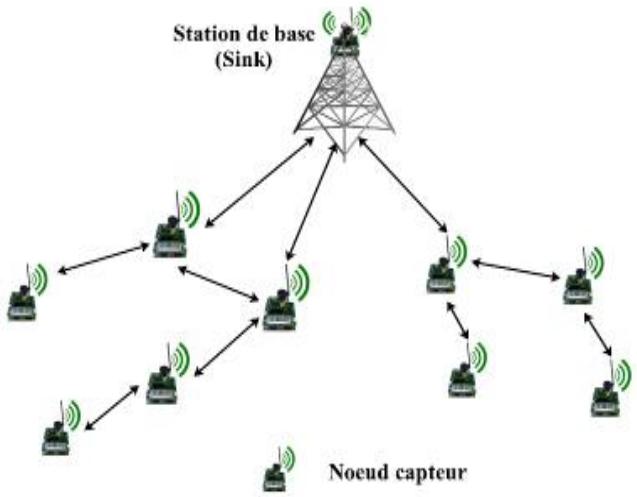


Figure 1.10 : *RCSF* avec une topologie plate.

1.4.6.2 Topologie hiérarchique

Le principe est de partitionner le réseau en plusieurs groupes (ou clusters) dont chacun est vu comme un sous réseau ayant une topologie en étoile, comme le montre la figure 1.11.

Chaque groupe possède un chef (appelé cluster-head) qui relie les membres de son groupe à la station de base. La communication entre les nœuds capteurs et le chef du cluster peut être directe ou indirecte (en multi-sauts) pour les nœuds distants. Ainsi, il peut y avoir plusieurs niveaux dans la hiérarchie, où les chefs des clusters forment entre eux des chaînes menant vers la station de base. Une architecture hiérarchique fut proposée pour réduire le coût et la complexité de la plupart des nœuds capteurs en introduisant un ensemble de nœuds capteurs plus onéreux et plus puissants. Cela devient possible par la création d'une infrastructure qui décharge la majorité des nœuds simples à faible coût de plusieurs fonctions du réseau.

L'architecture hiérarchique est composée de multiples couches : une couche de capteurs, une couche de transmission et une couche de point d'accès. Cette topologie est particulièrement avantageuse car elle est flexible et permet de garantir la durée de vie du réseau, du fait que les nœuds capteurs soient dans la majorité du temps endormis et ne deviennent actifs que s'ils veulent communiquer des informations au chef du cluster, ou pendant la mise à jour de la topologie. Un autre avantage marquant de la topologie hiérarchique réside dans le fait qu'elle réponde mieux au besoin d'extensibilité et d'évolutivité du réseau. La figure 1.11, présente un simple modèle de cette topologie.

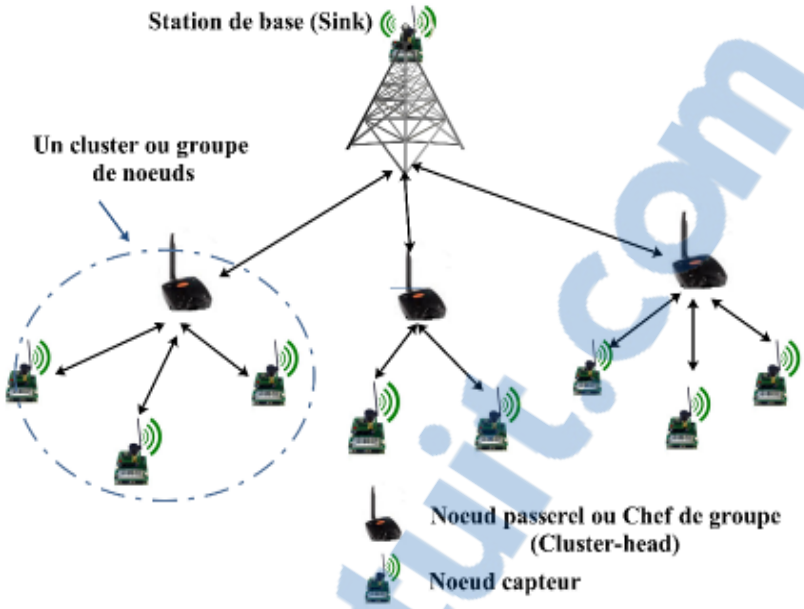


Figure 1.11 : *RCSF* Avec une Topologie hiérarchique (multi-étoiles).

1.4.7 Différents types d'architecture de *RCSF*

Comme le montre la figure 1.11, l'architecture des *RCSFs* comprend des nœuds de capteurs, des points d'agrégation (têtes de cluster), des stations de base (*Sink*, serveur central ou récepteur), un gestionnaire de réseau, un gestionnaire de sécurité et une interface utilisateur. Ces composants s'entraident mutuellement, ainsi, ils aident le *RCSF* à fonctionner, correctement [51, 52]. La figure 1.11 illustre les différents types de communication dans les *RCSFs* comme suit :

- **Architecture de communication directe :** Chaque nœud de capteur communique directement avec la station de base (*Sink*). Ainsi, cette architecture n'est pas appropriée pour les *RCSFs* larges, étant donné qu'elle n'est pas évolutive.
- **Architecture multi-saut et point-à-point :** Les nœuds de capteurs ont une capacité de routage, cette architecture souffre de scalabilité (n'est pas évolutive), parce que les nœuds de capteurs placés à proximité du nœud *Sink*, sont habituellement utilisés pour le routage de paquets entre les autres nœuds et le *Sink*. Par conséquent, si le *RCSF* est répandu, le trafic de ces nœuds augmentera et leur énergie sera gaspillée, consommée et épuisée, alors ils sortent rapidement du *RCSF*. Les nœuds de capteurs forment une structure de clustering. Le choix d'une tête de cluster pour n'importe quel cluster, permettra à la tête de communiquer directement avec le récepteur, ainsi, les nœuds de chaque cluster envoient les données collectées à la tête de cluster correspondante. La

faiblesse de cette architecture est la suivante : la plupart des opérations de communication se font par des têtes de cluster, ainsi, leur énergie sera consommée, diminuée et gaspillée, plus tôt que les autres nœuds (si les têtes de cluster ont des capacités faibles ou des *RCSFs* homogènes). Pour remédier à ce problème, il faudrait changer dynamiquement le rôle de tête de cluster vis-à-vis des les nœuds de cluster correspondants, ou utiliser des têtes de cluster fortes et hétérogènes.

- **Architecture multi-saut basé sur le clustering avec un cluster dynamique :** Cette architecture résout la faiblesse de l'architecture précédente en changeant dynamiquement le rôle de cluster-head vis-à-vis des nœuds de cluster correspondants [53, 54].

1.4.8 Les types des *RCSFs*

En général, les *RCSFs* peuvent être classifiés selon deux catégories, le modèle dynamique et le modèle statique. Le modèle dynamique est constitué d'un ensemble de capteurs mobiles évoluant dans un environnement statique, et dont le but est l'exploration de zones inaccessibles ou dangereuses. Avec ce type de réseau, les travaux de recherche sont orientés souvent robotique, où les nœuds jouent à la fois le rôle de capteur et d'actionneur. Concernant le modèle statique, le réseau est constitué de capteurs fixes servant uniquement à la surveillance d'occurrence d'événements sur une zone géographique, comme illustré précédemment dans la figure 1.7 dans la page 27.

Cependant, selon ces deux modèles de réseaux des *RCSFs* et selon que l'environnement de déploiement des nœuds capteurs, nous pouvons distinguer différents types de *RCSF* confrontés à différents défis et contraintes, tels que les réseaux de capteurs terrestres [55], sous-marins [56] [57], souterrains [58], multimédias [59] [60] et mobiles [61]. Une description détaillée de ces types de *RCSFs* est abordée dans [48].

1.4.9 Les Domaines d'application des RCSF

Les réseaux de capteurs sans fil sont adoptés dans divers domaines d'applications : les opérations civiles et militaires, la surveillance industrielle, la surveillance et la prévision environnementale en temps réel et les soins de santé [62] [63] [64] [65] [66] [67] [68] [69].

Actuellement, les *RCSF* sont exploités dans l'*IdO* pour les futures applications, concernant les maisons et villes intelligentes (appelées « smart homes & cities ») [16] [18], dont les capteurs sont embarqués dans la plupart des objets afin de les rendre intelligents, et sont ainsi connus sous le terme de « smart objects » (voir la section 1.2.3 page 14). Ces derniers pourront explorer

l'environnement, communiquer avec d'autres objets intelligents et interagir avec les humains [70]. Les *RCSF* traditionnels ont été développés en utilisant uniquement des nœuds statiques. Cependant, l'avancée rapide de la technologie des *RCSF* implique l'utilisation d'applications plus complexes nécessitant la mobilité des nœuds.

1.5 Les technologies de communication des RCSFs

La communication sans fil dans les réseaux de capteurs est extrêmement importante et critique. Les *RCSFs* peuvent en supporter plusieurs types, l'efficacité de ces communications et leur conformité aux particularités de l'application figée sont des critères clés pour le choix de telle ou telle technologie. Etant donné que les *RCSFs* sont des réseaux particuliers exploités dans l'*IdO*, alors le développement des applications de ces réseaux nécessite diverses conceptions technologiques de transmissions en fonction de la portée radio (*Range*) et de la contrainte de l'énergie consommée.

Dans la première section de ce chapitre, nous avons cité différentes conceptions technologiques et nous avons décrit la plupart des technologies à courte et à longue portée (voir la sous-section 1.2.5.1 pages 19-21), tel que : *Sigfox*, *Neul*, *LoRa*, les réseaux cellulaires (*GSM*, *2G*, 3G, *4G* et *5G*), ainsi que *Near Field Communication* (*NFC*), *WiFi*, *Bluetooth*, *Z-Wave* et *IEEE 802.15.4*.

Il est à noter que cette thèse porte sur les deux parties ayant trait à nos contributions. La première se rapporte à la technologie *IEEE 802.15.4*. Quant à la deuxième partie, elle prend en charge la globalité des différentes technologies de réseaux de capteurs utilisées dans l'*IdO*. En effet, il existe plusieurs technologies de communication utilisées dans les *RCSFs*, nous en citons les plus importantes et celles en relation avec notre travail :

1.5.1 La technologie IEEE 802.15.4

Dans sa première partie de contribution, cette thèse porte sur la technologie *IEEE 802.15.4* [72] [73], car elle est considérée comme l'une des principales technologies candidates aux réseaux de capteurs sans fil, *IdO*, *WdO*. En effet, cette technologie est essentiellement conçue pour les petits dispositifs embarqués du fait de leurs ressources limitées à savoir : faible consommation d'énergie, faible portée et faible débit des nœuds utilisant cette norme.

Cette technologie spécifie deux couches [74], la couche physique (*PHY*) et la sous-couche de contrôle d'accès au médium (*MAC*) pour les réseaux à très faible débit LR-*WPAN* (*Low Rate Wireless Personal Area Networks*), qui fonctionnent sur une bande de fréquence 2.4 GHz, avec un débit de 250kbps et une portée de 10 à 100 mètres. La sous-couche *MAC* de la norme utilise

l'accès multiple à détection de la porteuse avec évitement de collision, noté *CSMA/CA* (pour *Carrier Sense Multiple Access with Collision Avoidance*).

Comme illustré sur la figure 1.12, *IEEE 802.15.4* propose deux modes de fonctionnement pour l'accès au médium : un mode non coordonné (totalement *CSMA/CA*, sans *RTS/CTS*) [74], appelé non-beacon et un mode coordonné, ou mode balisé (*beacon-enabled mode* ou *slotted mode*). Ces couches (c.-à-d. *MAC et PHY*) sont très adaptées aux *RCSFs*, car ils offrent des fonctionnalités qui prennent en considération les contraintes des nœuds capteurs. L'objectif principal de l'*IEEE 802.15.4* est de réduire les coûts en termes de consommation d'énergie, basée sur le processus périodique sommeil/réveille [75].

En effet, Le protocole *MAC* [76] fonctionne selon deux modes pour l'accès au medium, 1) Le mode non-beacon, dans lequel les nœuds suivent le mécanisme classique *CSMA/CA*, qui est utilisé pour le type de capteur d'interrupteur. 2) Le mode beacon, utilise des trames beacons transmises pour économiser de l'énergie tout en omettant le mécanisme *CSMA/CA* pour prendre en charge le *duty cycle*. Pour garantir une synchronisation entre les nœuds capteurs, le mode *beacon* est basé sur une structure de *supertrame* comme indiqué sur la figure 1.13 [46].

Une *supertrame* peut comporter une période active et une période inactive. Durant la période inactive tous les nœuds (y compris le coordinateur) peuvent être en mode sommeil. La période active de chaque supertrame est divisée en seize intervalles de temps égaux et se compose de trois parties : la partie de la balise, la partie dans laquelle les nœuds accèdent au médium avec contention, notée *CAP* (pour *Contention Access Period*), et une autre partie avec un accès sans contention, notée *CFP* (pour *Contention Free Period*). Pendant la CAP, les nœuds accèdent au médium par compétition suivant l'algorithme *CSMA-CA* slotté [77].

Comme pour les couches supérieures de la pile de protocoles des *RCSFs*, d'autres technologies sont impliquées adoptant la norme *IEEE 802.15.4* et interagissant avec elle. Les technologies les plus connues sont : 1) *Zigbee*, qui est proposé par l'alliance *ZIGBEE* permettant le travail sur les couches Réseau et Application. 2) *6LoWPAN*, qui est proposé par *l'Internet Engineering Task Force (IETF)* permettant l'introduction de la mise en réseau sur l'*IPv6* [78], ces technologies vont être décrites par la sous-section suivante avec une discussion sur leur adaptabilité avec *IEEE*.

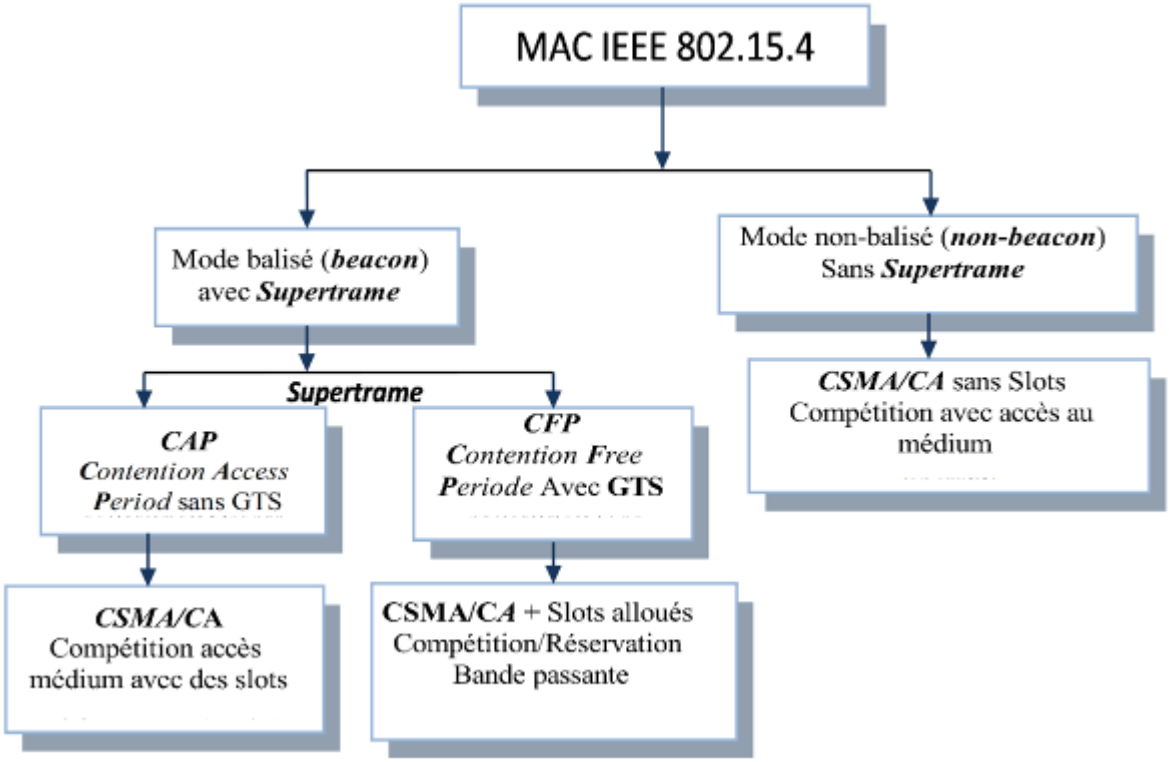


Figure 1.12 : Mode de fonctionnement du standard IEEE 802.15.4 [46].

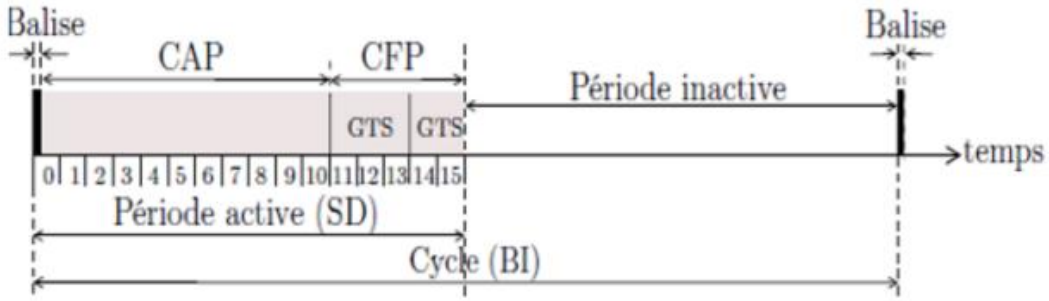


Figure 1.13 : La structure *Supertrame IEEE 802.15.4* du mode beacon.

1.5.2 La technologie *Zigbee*

Cette technologie permet de réduire les coûts de consommation des ressources. *ZigBee* [72] est structuré en couches comme tout réseau, il repose sur les couches inférieures définies par *l'IEEE 802.15.4* afin d'intégrer plus de protocoles dans les couches réseau « *NWK* » et « *APL* » Applications.

En ce qui concerne la couche réseau, *Zigbee* utilise les mêmes protocoles déjà utilisés pour le routage et l'adressage comme le protocole *AODV*. Le routage dans *ZigBee* représente une combinaison de trois types, celui du routage hiérarchique « *tree routing* », routage par voisinage

« *neighbour routing* », routage maillé « *mesh routing* ». Le routage par défaut dans *ZigBee* est hiérarchique, alors que le routage maillé est en fonction des tables de routage des nœuds du réseau *ZigBee*. Cela se fait généralement par le mécanisme de découverte de routes. Dans la couche application, certaines fonctionnalités sont appliquées en fonction de certains éléments de la couche réseau.

Cette technologie offre des avantages importants, notamment en termes de faible consommation d'énergie, de sécurité et d'efficacité, ce qui la rend plus adaptée pour les applications *RCSF*. Néanmoins, *Zigbee* a quelques faiblesses, principalement en termes d'interopérabilité avec le protocole *IP*. Cette lacune limite la connectivité directe entre les périphériques et les services *Web IP*. Ainsi, pour chaque communication externe, *Zigbee* nécessite un coordinateur *Zigbee* (*ZigBee Cordinator* « *ZC* ») ou une passerelle (*Gateway* « *GW* ») en tant que nœud intermédiaire qui centralise la communication [79]. En outre, *Zigbee* a des problèmes d'évolutivité (*scalability*).

1.5.3 La technologie *6LoWPAN*

La spécification de *6LoWPAN* est le nom d'un groupe de travail de l'*IETF* [31]. C'est aussi l'acronyme de l'*IPv6 Low Power Wireless Personnal Area Network* (réseaux sans fils à faible puissance). En effet, *6LowPAN* permet des communications *IPv6* sur des dispositifs sans fil de faible puissance.

L'objectif du groupe de travail *6LoWPAN* était de définir les mécanismes d'encapsulation et de compression d'en-têtes des paquets *IPv6* afin de leur permettre d'être envoyés ou reçus via le protocole de communication *IEEE 802.15.4* [80], qui est le protocole utilisé dans les réseaux à faible puissance comme les réseaux de capteurs sans fils, ce qui le rend idéal pour les applications *IdO*.

La norme *RFC 6LoWPAN* définit particulièrement la couche réseau de *WPAN* basée sur *IP*, en utilisant la norme *IEEE 802.15.4* dans les couches d'accès et physique. *6LoWPAN* bénéficie des avantages fournis par *l'IEEE 802.15.4* telles que la faible consommation d'énergie et la sécurité (*cryptage AES 128*) [81].

1.5.4 La technologie *LoWPAN*

Un *LoWPAN* ou *LR WPAN* (*Low Rate Wireless Personnal Area Network*) est constitué de dispositifs caractérisés par une faible consommation d'énergie, une faible portée, un faible débit et un faible coût de production. Les ressources des dispositifs formant ce type de réseaux sont généralement limitées, notamment en énergie, mémoire et puissance. L'exemple type de ces

réseaux est représenté les *RCSFs*. Les nœuds composant un *LoWPAN* ont pour rôle de collecter et de transmettre des informations vers un équipement central du réseau qui aura la charge de traiter ces informations et les router vers leur destination. Cet équipement est appelé station de base (*Sink*) ou *PAN* coordinateur. Dans un réseau *LoWPAN*, deux types de nœuds sont définis, les *FFD* pour *Full Function Device* et *RFD* pour *Reduced Function Device*. Les réseaux *LoWPAN* utilisent le protocole de communication *IEEE 802.15.4* que nous allons présenter dans la section suivante [82]. Ces nœuds peuvent avoir les fonctions suivantes [83] :

- ***Sink (station de base)* :** c'est le nœud central du réseau. Il a pour rôle d'initier le réseau, rassembler les informations et les données qu'il acheminera par la suite vers l'extérieur. Les *Sink* sont aussi appelés *PAN* coordinateur ou *Gateway* et sont de type *FFD*,
- ***Les coordinateurs* :** ce sont des nœuds de type *FFD* qui peuvent jouer le rôle de *Gateway* intermédiaire dans le cas où le réseau est composé de plusieurs zones ou clusters. Ils auront la charge d'acheminer les paquets vers le *Sink* ou vers les autres nœuds fils,
- ***Les nœuds terminaux :*** ce sont les derniers éléments de la chaîne. Ils ont la tâche de collecter les informations et de les transmettre vers leurs parents (nœuds coordinateurs), dans le cas d'un réseau multi-sauts, ou directement vers le *Sink* (*PAN*) quand il s'agit d'une architecture simple.

Nous pouvons dire qu'un *FFD* peut jouer le rôle de *Sink*, de coordinateur ou de terminal alors qu'un *RFD* ne peut assurer que le rôle de nœud terminal ou de capteur dans le cas d'un réseau de capteurs. Les *RFD* ne peuvent communiquer qu'avec des *FFD* alors que les *FFD* peuvent communiquer à la fois avec les *RFD* et les *FFD*. Dans les réseaux *LoWPAN*, on peut rencontrer trois types de topologie comme le montre la figure 1.14.

- *La topologie en étoile :* dans cette topologie les échanges se font directement avec le *PAN coordinateur*. Il n'existe plus de communication possible entre les autres nœuds. C'est à la station *Sink* (*PAN*) d'initier le réseau, d'établir la communication avec tous les dispositifs se trouvant dans sa portée radio (figure 1.14 (a)). Les échanges de transmission se font avec des trames beacons.
- *La topologie mesh :* Les nœuds de cette topologie peuvent communiquer entre eux. Il s'agit d'une topologie de réseau maillé où plusieurs liens sont établis et différents chemins sont possibles (figure 1.14 (b)).
- *La topologie cluster* : C'est une topologie mesh à la base. La différence est que le réseau est hiérarchisé, c'est-à-dire qu'il y a un *PAN* coordinateur et plusieurs

coordinateurs responsables chacun au niveau d'une zone contenant un ou plusieurs nœuds terminaux (figure 1.14 (c)).

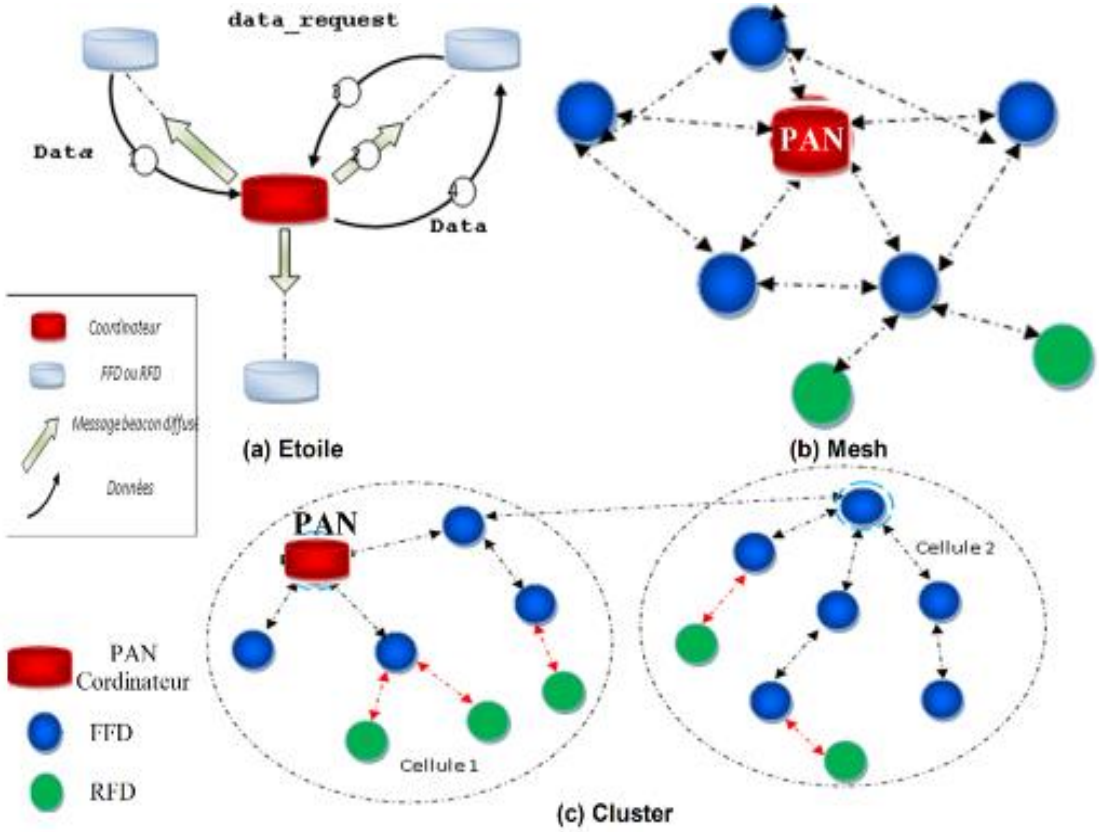


Figure 1.14 : Topologies dans les réseaux LoWPAN (a), (b) et (c).

1.5.4.1 Protocoles de routage dans *6LoWPAN*

Afin de pouvoir acheminer les trames et paquets dans un réseau *LoWPAN* adapté à *IPv6* et ses fonctionnalités, la technologie *6LoWPAN* permet la mise en place d'un routage effectué selon deux types de technique de routage [84] [85] :

- ***Mesh under*** *:* la décision de routage est au niveau de la couche adaptation *6LoWPAN* et uniquement avec les fragments du paquet *IPv6*. Cette méthode est plus rapide, sauf en cas d'erreur, auquel cas il faut rejeter tous les fragments et recommencer. Le protocole *LOADng* [86] (de l'anglais *Lightweight On-demand Ad hoc distance*-vector *routing protocol – next generation*), est utilisé dans cette technique de routage (voir figure 1.15).
- ***Route over :*** la figure 1.15 explique que ce type de routage est défini au niveau de la couche réseau du modèle *OSI*, les paquets IPv6 sont reconstitués au niveau de chaque équipement intermédiaire afin de déterminer un nouvel itinéraire de routage. Cette

technique permet une détection plus rapide en cas de perte de paquets et une demande de retransmission ne sera faite qu'au nœud émetteur. Le protocole RPL (*Routing Protocol for Low power and Lossy Network*) permet d'assurer ce type de routage. RPL a été mis au point par le groupe de travail *ROLL* [6] (*Routing Over Low power and Lossy Networks*) de l'*IETF* [87].

Actuellement et avec les avancées récentes des techniques de routage avec la technologie *6LoWPAN*, le protocole *RPL* est beaucoup plus utilisé pour de larges déploiements. Une description détaillée sur *RPL* est présentée dans le chapitre suivant. Les deux techniques de routage de *6LowPAN* sont présentées dans la figure 1.15.

- **Le routage dans les RCSF et tolérance aux pannes**

Le routage dans les *RCSFs* est une fonctionnalité cruciale basée sur le processus de communication intra-nœud et le processus de transfert de données à partir des nœuds individuels à un point de collecte central (*Sink*). Dans un *RCSF*, en plus le routage lié au fonctionnement distribué des protocoles, le réseau doit être capable d'assurer la continuité du service de surveillance et communication vis-à-vis des pannes des nœuds ou des liens.

De ce fait, les protocoles de routage multi-sauts existants et qui sont standardisés par l'*IETF*, tels que *DYMO*, *AODV*, *DSR* et *OLSR* [88] [89] [90], ont été largement évalués par le groupe de travail qui les a jugés peu satisfaisants, dans leur forme actuelle, quant à toutes les exigences de routage et contrôle spécifiques vis-à-vis l'existence des pannes dans le réseau. En effet, ces protocoles ne sont pas adaptés et donc peu capables d'établir une réparation locale ou globale lors d'une panne qui peut se produire dans un *RCSF*.

En effet, ils ne sont définis que par rapport au concept de routage entraînant ainsi une forte consommation énergétique des nœuds. Le contexte de cette thèse, porte sur le contrôle des *RCSFs* à l'objectif d'assurer la continuité de service de surveillance tout en minimisant l'énergie. Il existe aussi des protocoles de communication (*MAC* et routage) pouvant répondre à ces exigences tout en proposant des stratégies de contrôle spécifiques aux pannes des nœuds dans les *RCSFs*. Ceci intégrera, entre autres, la stratégie de réparation à temps réel pour garantir la stabilité et la fonctionnalité du réseau lors de la surveillance.

En effet, l'apparition du protocole de routage *RPL* est beaucoup plus sollicitée en termes de routage, de coût énergétique, permettant ainsi de gérer et de contrôler la topologie du réseau à travers un mécanisme de réparations locale et globale en cas de pannes de nœuds ou liens.

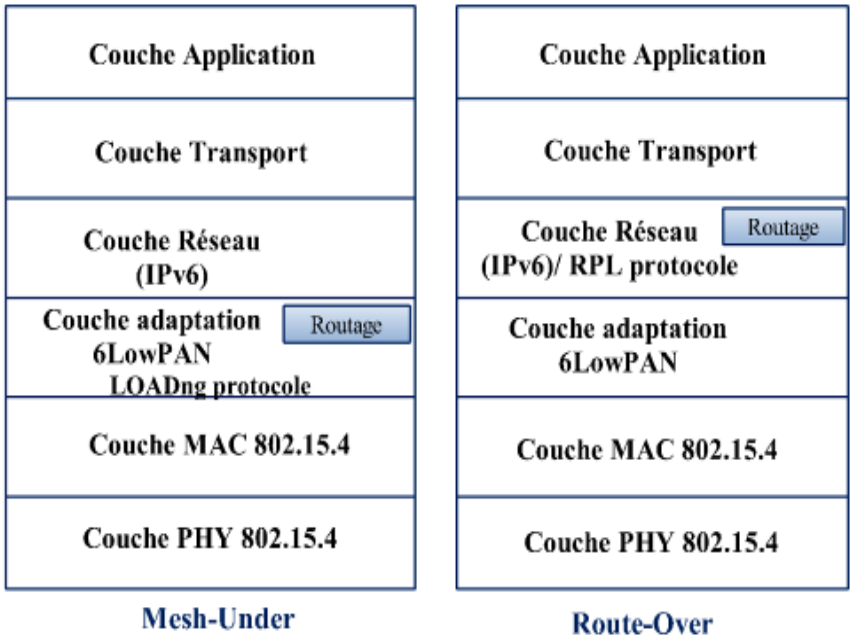


Figure 1.15 : Couche d'adaptation *6LowPAN* avec protocoles de routages [83].

1.6 Objectifs du routage dans les RCSFs

Dans le cadre des *RCSFs*, le souci majeur du routage est la conservation d'énergie qui peut étendre considérablement la durée de vie du réseau. Le routage doit, en outre, assurer la continuité de service de surveillance du réseau et optimiser la qualité de service (*QdS*) temps réel requise à tout instant par les applications. Cependant, une technique de routage efficace doit être censée appliquer les fonctions principales suivantes [91] :

- La détermination et la détection des changements de la topologie du réseau.
- Le maintien de la connectivité réseau,
- Le calcul et la détection des bons itinéraires,
- La réparation locale et globale de la topologie du réseau en cas de nœuds en panne,
- Garantir une latence faible,
- Assurer une consommation minimum d'énergie,
- Minimiser la charge du réseau.

1.7 Conclusion

Dans ce premier chapitre ayant trait au contexte technologique, nous avons donné dans un premier temps un bref aperçu sur l'*IdO*, en présentant ses domaines d'applications, ses technologies de communication et l'hétérogénéité de ses réseaux pour différentes applications. Une description rapide des *RCASFs* fut présentée comme exemple d'hétérogénéité des réseaux existant dans l'*IdO*. Nous avons présenté, en outre, un autre aperçu détaillé sur les *RCSFs* en abordant brièvement leurs différentes spécificités, le principe de leur fonctionnement avec caractéristiques et contraintes inhérentes.

Ce chapitre a également pris en charge les technologies adaptables avec les *RCSFs*, notamment à travers la description de *6LowPAN* comme contexte technologique de cette thèse avec le protocole de routage *RPL*. En effet, le nouveau protocole *RPL* apporte de nouvelles perspectives au niveau de la technique de routage, dont le but est d'en exploiter le mécanisme de réparation de nœud défaillants dans le réseau.

Le contexte technologique de notre travail étant bien explicité, à travers ce chapitre, nous allons nous focaliser dans le chapitre suivant sur une étude détaillée du protocole RPL et son mécanisme de réparation, et qui représente de ce fait une nouvelle solution de routage quant à la tolérance aux nœuds défaillants dans les *RCSFs*.

Surveillance et problématique des nœuds défaillants dans les RCSFs

Chapitre 2

2.1 Introduction

Afin de mettre en place un *RCSF* pour une application de surveillance intégrée dans l'*IdO*, nous devons assurer la fidélité de contrôle de ce type de réseau pour la supervision, c'est-à-dire assurer la continuité de service de surveillance et maintenir ses fonctionnalités sans interruption en cas de défaillance d'un de ses nœuds capteurs. En outre, nous devons aussi garantir la fidélité de routage, c'est-à-dire qu'il doit exister au moins un chemin entre les nœuds capteurs et la station de base.

Ces facteurs de routage nous ont permis de concevoir plusieurs protocoles de routage tolérant à la gestion et réparation locale des nœuds défaillants dans les *RCSFs*. Ces protocoles peuvent contribuer à la prévention, le contrôle et la gestion du bon fonctionnement du réseau.

Dans ce chapitre, nous abordons en premier lieu le concept de la supervision et le contrôle des *RCSFs*. Ensuite, nous décriront le concept de la tolérance aux pannes dans les *RCSF* et précisément le problème de nœuds défaillants, en second lieu, nous introduisons le concept de remplacement des nœuds défaillants et les méthodes proposées de réparations locales. En fin de ce chapitre, nous présentons les solutions proposées de réparation et remplacement des nœuds défaillants. Cette partie permet de positionner notre première partie de contributions au problème de gestion et réparation de la défaillance des nœuds dans les *RCSFs* par rapport aux travaux existants.

2.2 Concept de supervision

Le terme technique de supervision (ou en anglais monitoring) est la fonction de réagir ou même contrôler un cas particulier d'événement lors du service de surveillance ou d'observation des activités environnementales. Ces dernières années, les avancées technologiques et surtout dans le domaine informatique et électronique ont apporté de nouveaux champs d'application à la surveillance et surtout une nouvelle vision pour y procéder [92].

Un système de supervision comporte un réseau de dispositifs de surveillance équipés chacun de moyens de détection de type scalaire ou multimédia. L'auto-organisation de ces derniers est réalisée selon la configuration du réseau, qu'il soit filaire ou sans fil, sous un contrôle supervisé par un dispositif central en vue d'accomplir la tâche de gestion centrale, qui fait partie du réseau et qui se trouve en liaison radio permanente avec les dispositifs de surveillance.

Afin d'augmenter la fiabilité, la disponibilité et la sûreté de fonctionnement des processus, les systèmes de supervision sont mis en œuvre avec l'objectif d'être capable à tout instant, de fournir l'état de fonctionnement des différents équipements constitutifs d'un processus technologique. Et ce, tant au niveau de la détection et de l'isolation des fautes (*FDI*) qu'au niveau de la tolérance aux fautes (*FTC*). L'opérateur de supervision gère deux types d'information, le premier concerne la détection et l'isolation des défauts survenus, et le deuxième indique les possibilités de laisser fonctionner ou non le processus [93].

2.3 Les réseaux de supervision

Les différents réseaux de supervision peuvent être distingués selon le type de nœuds capteurs qui les compose. Nous allons à présent citer trois classes de réseaux de surveillance. Selon la classification présentée par les travaux de recherche dans [48], il existe des réseaux de supervision à base de capteurs scalaires, d'autres à base de capteurs multimédias ou encore une hybridation de ces deux types.

2.3.1 Réseaux de supervision à base de capteurs scalaires

Un réseau de supervision à base de capteurs scalaires réalise la détection grâce à des capteurs de mouvements, de température, de pression barométrique ou d'autres grandeurs physiques. Il s'agit alors de collecter des données de grandeurs scalaires. On trouve ce type de réseaux de surveillance dans plusieurs domaines d'application, telle que la détection de l'influence de l'activité sismique sur un pont où l'on place généralement des capteurs de distance pour mesurer les fissures et donc alerter en cas de risques majeurs [48].

2.3.2 Réseaux de supervision à base de capteurs multimédias

Un capteur sans fil multimédia est celui qui détecte et transmet des données multimédias telles que la vidéo, l'image et le son comme les caméras *WiFi,* les capteurs ultrasons et d'imagerie infrarouge. Ces moyens forment des réseaux de capteurs multimédias et sont surtout utilisés dans le domaine de la surveillance environnementale, sécuritaire et médicale. Un réseau de capteurs

multimédias se compose de nœuds caméra (type de nœud multimédia). Les réseaux de surveillance les plus répandus sont ceux qui utilisent des capteurs vidéo.

2.3.3 Réseaux de supervision à base de capteurs hybrides

Il s'agit d'une hybridation entre les deux types précédents. Un réseau de surveillance à base de capteurs hybrides se compose à la fois de capteurs à grandeurs scalaires et des capteurs multimédias pouvant communiquer entre eux via des transmissions *WiFi* ou *ZigBee*. Cette approche garantit à la fois la détection rapide fournie par les capteurs à grandeurs scalaires, puisque les données transférées sont simples (comme la position d'un intrus détecté par un capteur de mouvement), mais aussi une détection plus pertinente comme la vidéo fournie par des capteurs multimédias.

Cependant, la capture de données multimédias nécessite de nouveaux mécanismes de routage et une agrégation particulière des données [93].

2.4 Concept de contrôle des *RCSFs*

Un réseau de capteurs est conçu pour fonctionner le plus longtemps possible, durant des mois voire des années. A cet égard, les protocoles conçus pour les réseaux de capteurs doivent judicieusement suivre un plan énergétique efficace. En fait, la détection, la transmission et l'acheminement des données sont des tâches qui consomment beaucoup plus d'énergie. En effet, la limitation des ressources dans les capteurs sans fil constitue l'un des challenges rencontrés dans le maintien du fonctionnement des *RCSFs*. Le contrôle est une tâche de gestion et de détection des facteurs de disfonctionnement du réseau. Un nœud défaillant est l'un des facteurs participant au disfonctionnement et la non stabilité du réseau de capteurs. La défaillance d'un nœud capteur est survient lors d'une extinction d'un capteur suite à un épuisement de sa batterie, ou tout simplement à une destruction physique accidentelle ou intentionnelle.

D'un autre côté, le concept de contrôle dans un *RCSF* peut être réalisé selon des stratégies classiques via des solutions des protocoles de routages [143] ou par le nouveau concept de contrôle de supervision à travers des contrôleurs centralisés, à savoir le concept des réseaux défini par logiciel (*SDN*) [7]. Cette approche assure une supervision globale de l'état du réseau permettant de garantir une exactitude quant au diagnostic des problèmes complexes (par exemple, la défaillance des nœuds critiques, l'occurrence des fautes simultanées ou l'affection d'une zone complète des nœuds, etc.). Le SDN sera présenté dans le chapitre 4 comme deuxième contribution liée à cette thèse. De ce fait, il est donc nécessaire de considérer la tolérance aux pannes comme critères indispensable dans la conception des protocoles de routage pour les RCSF.

Dans la suite de cette section, nous présentons quelques notions sur la tolérance aux pannes et en particulier dans les *RCSF*.

2.5 Le concept de tolérance aux nœuds défaillants

Le dysfonctionnement d'un réseau de capteur peut se réaliser par le fait qu'un nœud capteur tombe en panne (manque d'énergie, une défaillance logique ou physique, interférences avec l'environnement d'observation…). Ce réseau doit continuer à fonctionner normalement sans interruption et assurer le service de surveillance. Ceci explique le fait qu'un *RCSF* doit adopter un concept de la tolérance aux nœuds défaillants et la possibilité de remplacer ces nœuds défaillants par d'autres nœuds plutôt fonctionnels et capables à maintenir le réseau.

La tolérance aux nœuds défaillants est donc la capacité de maintenir le fonctionnement du réseau de capteurs et d'assurer la continuité de service de surveillance après une panne d'un ou de plusieurs nœuds capteurs sans aucune interruption. Cependant, Un premier défi sera donc de détecter, d'identifier les nœuds de capteurs défaillants et de procéder rapidement à leur remplacement, ensuite de modéliser des méthodes et des techniques de tolérance aux nœuds défaillants à mettre en œuvre. La tolérance à la défaillance d'un nœud capteur est modélisée à l'aide de la distribution de poisson (voir formule 2.1) pour déterminer les probabilités de ne pas avoir de défaillance dans l'intervalle de temps (0, T) [43] :

$$R_k = e^{-\lambda t} * \lambda t \qquad (2.1)$$

Où λ_k **est** le taux de défaillance du nœud capteur k, et t est une période de temps.

On dit que la tolérance aux pannes est meilleure lorsque le nombre de nœuds en panne est grand et que le réseau maintien toujours son bon fonctionnement. En effet, l'objectif principale de la tolérance aux nœuds défaillants est d'éviter la défaillance globale du réseau même si elle se présente dans un sous ensemble de ses composants élémentaires.

En général, la méthode de tolérante aux défauts est classée selon [94] en deux approches distinctes : 1) Une approche passive (*PFTC*) et 2) une approche active (*AFTC*). L'idée principale dans la première approche est de faire en sorte que le système en boucle fermée (*BF*) soit robuste aux incertitudes et à quelques défauts spécifiques, par contre, *l'AFTC* est amenée à réagir aux diverses défaillances du système en reconfigurant les lois de commande tout en préservant la stabilité et les performances.

2.5.1 Notion de panne

Les concepts de panne ou faute, erreur et faille du système à base de *RCSF* est définie comme présenté par l'auteur [95] :

- **Une faille (ou panne) :** une faille du système à base de *RCSF* se produit lorsque son comportement devient inconsistant et ne fournit pas le résultat voulu. La panne est une conséquence d'une ou plusieurs erreurs.
- **Une erreur :** représente un état invalide du système dû à une faute (défaut).
- **La faute :** est donc la première cause de l'erreur, cette dernière provoque la faille du système. La figure 2.1 illustre la relation entre faute, erreur et faille. Selon cette figure, on peut aller de la faute à une faille (ou panne) comme suit : Quand une faute est activée, elle provoque une erreur. En se propageant, l'erreur engendre une faille [95].

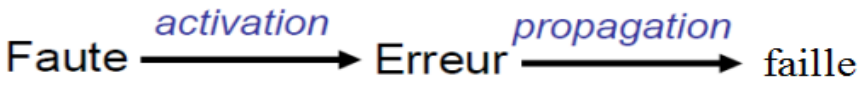


Figure 2.1 : La relation entre faute, erreur et faille [95].

2.5.2 Classification des pannes

Il est utile de classifier les pannes selon différents critères. Le schéma illustré par la figure 2.2 montre une classification générale selon la durée, la cause ou le comportement d'une panne [95] :

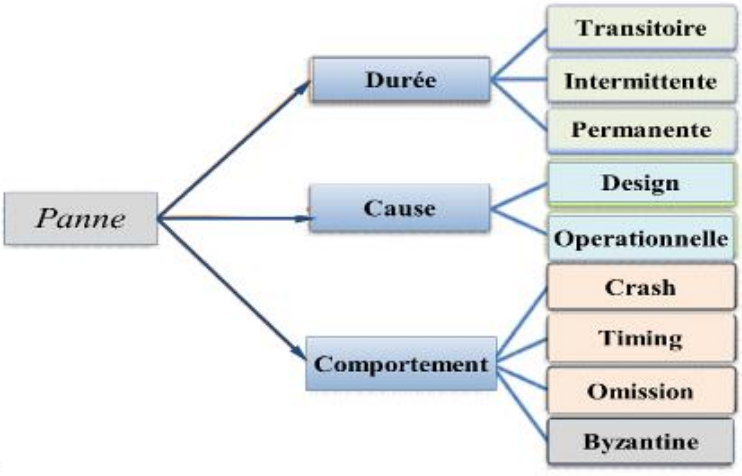


Figure 2.2 : Classification des pannes [95].

2.5.2.1 Pannes selon durée

Basée sur sa durée, la panne peut être classifiée en [95] :

- **Transitoire :** C'est une conséquence d'un impact environnemental temporaire, elle peut éventuellement disparaître sans aucune intervention. La radiation cosmique est un exemple de panne transitoire.
- **Intermittente** : Est une variante de la panne transitoire, elle se produit occasionnellement et de façon imprévisible. Elle est généralement due à l'instabilité de certaines caractéristiques matérielles ou à l'exécution du programme dans un espace particulier de l'environnement.
- **Permanente** : C'est une panne continue et stable dans le temps, la panne permanente persiste tant qu'il n'y a pas d'intervention externe pour l'éliminer. Un changement physique dans un composant provoque une panne matérielle permanente.

2.5.2.2 Pannes selon la cause

On distingue deux types de pannes selon leur cause :

- **Panne de design :** due à une mauvaise structuration du réseau ou du composant en particulier. En pratique, ce genre de panne ne devrait pas exister grâce aux tests et simulations exécutés avant la réalisation finale du réseau.
- **Panne opérationnelle :** qui se produisant durant le fonctionnement du système. Elle est généralement due aux causes physiques. En outre, on peut distinguer, spécialement pour les réseaux de capteurs, trois causes principales :
    - *Energie* **:** l'épuisement de la batterie cause l'arrêt du capteur. La consommation d'énergie est très importante pour déterminer la durée de vie d'un nœud capteur, et donc de tout le réseau.
    - *Sécurité* **:** la destruction physique accidentelle ou intentionnelle par un ennemi peut être une cause de panne. L'absence de sécurité dans les réseaux de capteurs augmente le risque des pannes de ce type.
    - *Transmission* **:** la nature vulnérable de transmission radio, la présence d'obstacles dans les environnements hostiles ainsi que les interférences électriques peuvent être la source d'une faute lors du transfert de données.

2.5.2.3 Pannes selon le comportement résultant

Après l'occurrence d'une panne, on distingue quatre différents comportements possibles du composant :

- **Panne accidentelle (Crash)** : le composant soit, s'arrête complètement de fonctionner ou bien continue mais sans retourner à un état stable (valide).

- **Panne d'omission :** le composant n'est plus capable d'améliorer son service (échec total).
- **Panne de synchronisation (Timing) :** le composant effectue son traitement mais fournit le résultat en retard.
- **Panne Byzantine :** Cette panne est de nature arbitraire, le comportement du composant est donc imprévisible. Dû à des attaques très malicieuses, ce type de pannes est considéré comme le plus difficile à gérer.

2.5.3 Procédure de tolérance aux pannes

La conception d'une procédure pour la tolérance aux pannes dépend de l'architecture et des fonctionnalités du système. Cependant, certaines étapes générales sont exécutées dans la plupart des systèmes, tel qu'il est démontré dans la figure 2.3 :

Figure 2.3: Procédure de tolérance aux pannes [95].

- **Détection de l'erreur :** C'est la première phase dans chaque schéma de tolérance aux pannes, dans laquelle on reconnaît qu'un événement inattendu s'est produit. Les techniques de détection de pannes sont généralement classifiées en deux catégories : en ligne et autonome (offline) :
  - *La détection offline :* est souvent réalisée à l'aide de programmes de diagnostic qui s'exécutent quand le système est inactif.
  - *La détection en ligne :* vise l'identification de pannes en temps réel et est effectuée simultanément avec l'activité du système.
- **Détection de la panne :** Cette phase établit des limites des effets de la panne sur une zone particulière afin d'empêcher la contamination des autres régions. En cas de détection d'intrusion, par exemple, l'isolation des composants compromis minimise le risque d'attaque des composants encore fonctionnels.
- **Recouvrement d'erreur :** C'est la phase dans laquelle on effectue des opérations d'élimination des effets de pannes. Les deux techniques les plus utilisées sont le masquage de la panne et la répétition :

- *Masquage de panne :* utilise l'information redondante correcte pour éliminer l'impact de l'information erronée.
- *Répétition :* après que la panne soit détectée, on effectue un nouvel essai pour exécuter une partie du programme, dans l'espoir que la panne soit transitoire.

2.5.4 Classification des approches de remplacement des nœuds défaillants dans les *RCSF*

Les approches de remplacement des nœuds défaillants dans un *RCSF* peuvent être proposées selon plusieurs angles bien différents. De ce fait, un ensemble de critères est défini afin de les classifier. Nous citons, entre autres, trois catégories distinctes de classifications [95] :

2.5.4.1 Classification selon la phase de traitement temporelle

Dans cette classification, l'ensemble des algorithmes développés se distinguent selon deux catégories principales à savoir les algorithmes préventifs, si le traitement est effectué avant la panne, sinon on parle donc sur les algorithmes curatifs [95].

- **Algorithme préventif :** c'est l'implémentation des techniques tolérantes aux pannes qui tentent de retarder ou d'éviter tout type d'erreur afin de garder le réseau fonctionnel le plus longtemps possible. La conservation d'énergie à titre d'exemple, permet de consommer moins d'énergie et évite donc une extinction prématurée de la batterie ce qui augmente la durée de vie des nœuds.
- **Algorithme curatif** : utilise une approche optimiste, où le mécanisme de tolérance aux pannes implémenté n'est exécuté qu'après la détection de pannes. Pour cela, plusieurs algorithmes de recouvrement après pannes sont proposés dans la littérature, par exemple : le recouvrement du chemin de routage, l'élection d'un nouvel agrégateur...etc.

2.5.4.2 Classification selon la phase de traitement architecturale

Cette classification traite les différents types de gestion des composants d'un *RCSF*, soit individuellement au niveau du capteur ou sur tout le réseau. Nous distinguons trois principales catégories :

- ***Gestion de batterie :*** Cette catégorie est considérée comme une approche préventive, où les protocoles définissent une distribution uniforme pour la dissipation d'énergie entre les différents nœuds capteurs, afin de mieux gérer la contrainte de consommation énergétique et augmenter la durée de vie du réseau. En outre, le mécanisme de mise en veille est une

technique efficace de gestion de batterie. En effet, les protocoles déterminent des délais de mise en veille des nœuds capteurs inactifs pour une meilleure conservation d'énergie.

- ***Gestion de flux :*** Cette catégorie regroupe les techniques qui définissent des protocoles de gestion de transfert des données (routage, sélection de canal de transmission, etc.). Nous pouvons trouver des approches préventives ou curatives sur les différentes couches (réseau, *MAC*, etc.) tels que :

  – *Routage multipath :* utilise un algorithme préventif pour déterminer plusieurs chemins depuis chaque capteur vers le nœud collecteur, c'est-à-dire le réseau devrait être de type *k-connected*. Ceci garantit la présence de plus d'un chemin fiable pour la transmission et offre une reprise rapide du transfert en cas de panne sur le chemin principal en choisissant un des chemins restant.

  – *Recouvrement de route :* après détection de panne, une technique curative permet de créer un nouveau chemin qui soit le plus fiable pour retransmettre les données.

  – *Allocation de canal* : cette solution est implémentée au niveau de la couche *MAC*. Elle permet d'effectuer une allocation du canal de transmission de manière à diminuer les interférences entre les nœuds voisins et éviter les collisions durant le transfert.

  – *Mobilité* : certains protocoles proposent comme solution tolérante aux nœuds défaillants, la sélection d'un ensemble de nœuds mobiles chargés de se déplacer entre les capteurs et collecter les données captées. Ceci réduira l'énergie consommée au niveau de chaque capteur en éliminant sa tâche de transmission. Un nœud mobile est généralement doté d'une batterie plus importante que celle d'un nœud capteur simple.

- ***Gestion des données :*** Les protocoles classés dans cette catégorie offrent une meilleure gestion et traitement de données. Deux principales sous-catégories sont déterminées :

– *Agrégation* : considérée comme approche préventive, l'opération d'agrégation effectue un traitement supplémentaire sur les données brutes captées depuis l'environnement. Un nœud agrégateur combine les données provenant de plusieurs nœuds en une information significative. Cela réduit considérablement la quantité de données transmises en consommant moins d'énergie pour leur dissémination. Ce qui permet donc d'augmenter la durée de vie du réseau.

– *Clustering* : une des approches importantes pour traiter la structure d'un réseau de capteurs est le *clustering*. Il permet la formation d'un *backbone* virtuel qui améliore l'utilisation des ressources rares telles que la bande passante et l'énergie. Par ailleurs, le *clustering* aide à réaliser du multiplexage entre différents clusters. En outre, il améliore les performances des

algorithmes de routage. Plusieurs protocoles utilisent cette approche préventive, qui est considérée parfois comme une approche curative.

2.6 Concept de remplacement de nœuds défaillants dans les *RCSF*

Plusieurs travaux se sont intéressés au concept de la réparation des nœuds défaillants ou, en général, au remplacement du nœud défaillant dans les *RCSFs* pour garantir la sûreté du fonctionnement du réseau en question. [96, 97, 98, 99, 100].

2.6.1 Solutions de remplacement de nœuds défaillants par des approches curatives

Dans [100], l'auteur étudie deux différentes approches présentées en deux classes visant à maintenir la connectivité des *RCSF* le plus longtemps et proposer des solutions pour prolonger la durée de vie du réseau dans le cas de dysfonctionnement des liens entre les nœuds de capteurs source et la station de base.

L'auteur dans [150] présente ces approches classées selon deux catégories de maintenance, qui sont la maintenance préventive et la maintenance curative, illustrées par la figure 2.4. Ainsi, la maintenance préventive définit des approches visant à prolonger la durée de vie du réseau, la maintenance curative intervient en cas de défaillance d'un nœud capteur pouvant causer le partitionnement du réseau ou laisser une partie du réseau sans couverture. L'objectif atteint par cet auteur fut de proposer comme résultat son approche de détection et remplacement de nœuds défaillants pour maintenir la connectivité du réseau en question.

- L'approche proposée dans [100] est *DRFN* (*Detection and Replacement of a Failing Node*) qui prend en considération la durée de vie du réseau et dont l'énergie totale consommée pour la restauration de la connectivité est partagée par plusieurs nœuds, de sorte que la consommation d'énergie individuelle soit minime, et ce, tout en respectant des caractéristiques des capteurs.

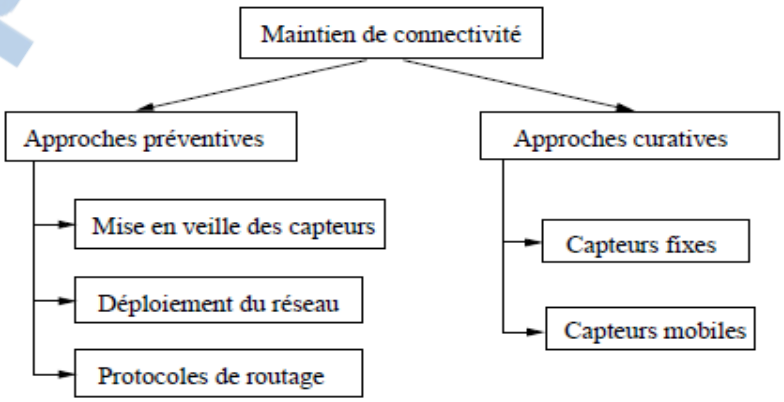


Figure 2.4 : Classification des approches de maintien de la connectivité [100].

- Les auteurs dans [96, 97, 98, 99] ont défini des approches de maintenance curative afin de remplacer les nœuds défaillants, à l'objectif de régler le problème de connectivité dans le cas du partitionnement du réseau, et/ou le problème de couverture de la zone de surveillance.
- L'approche *DARA* est proposée par les travaux menés dans [96]. Cette solution permet de remplacer le nœud défaillant par l'un de ces voisins directs à un seul saut. Pour effectuer un tel déplacement de position de l'un de ses voisins directs du nœud défaillant, les auteurs ont défini les critères suivants pour l'élection du meilleur nœud remplaçant :
    1. Le nœud ayant moins de voisins directs.
    2. Le nœud le plus proche du nœud défaillant.
    3. Le nœud qui a l'identifiant le plus élevé.

Un exemple illustré sur la figure 2.5 (a) (b) et (c) permet de présenter le principe de l'approche *DARA*. L'auteur de cette approche suppose que si le nœud n1 dans la figure 2.5 (a) tombe en panne alors les nœuds n2, n7, n8, n9 doivent initier la procédure de remplacement du nœud défaillant (*n1*). Vu que les nœuds *n9 n7* ont des voisins en multi-sauts, alors ils seront exclus, par contre les nœuds *n2* et *n8* sont en un seul saut du nœud défaillant *n1*, alors ils seront donc sollicités pour le remplacer.

Par supposition que *n2* et *n8* ont la même distance les séparant de *n1*, le nœud *n8* sera élu comme nœud remplaçant par exécution du critère 3. La figure 2.5 (b) montre le réseau après la restauration de la connectivité. Dans le cas où le nœud *n2* sera élu remplaçant (voir la figure 2.5 (c)), le nœud *n3* devra également se déplacer [96].

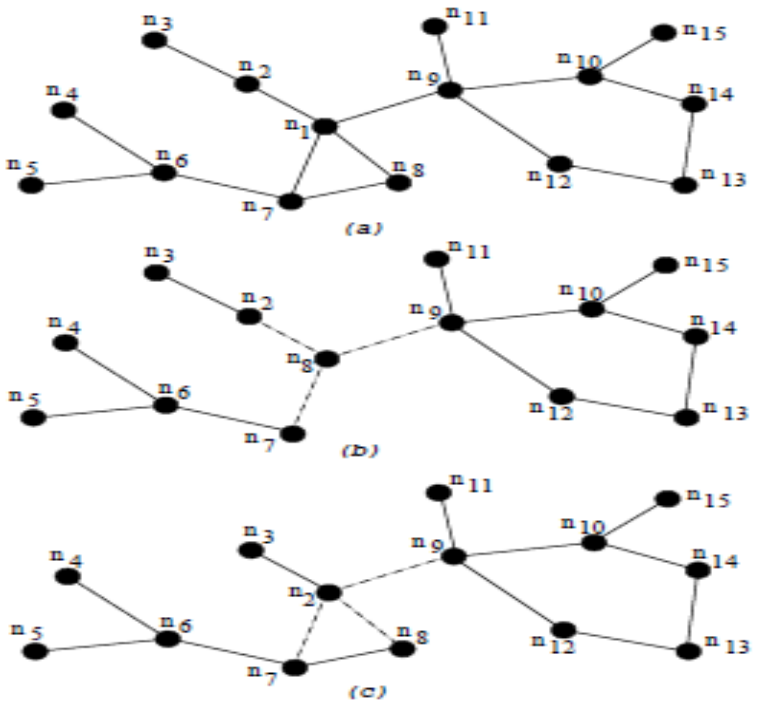


Figure 2.5 : Exemple de scénario avec l'approche *DARA* [96].

L'approche *DARA* ne prend pas en considération les caractéristiques des *RCSF*, en ce qui concerne la couverture du réseau et le taux d'énergie de chaque nœud, qui représentent des facteurs très importants pour la durée de vie d'un *RCSF*. L'énergie n'est pas tenue en compte lors de l'élection du nœud voisin destiné à remplacer le nœud défaillant.

DARA se focalise seulement sur le maintien de la connectivité sans se préoccuper de ces deux caractéristiques (énergie et couverture).

- Dans [97], l'auteur propose l'approche *C3R* qui permet de remplacer un nœud défaillant nommé *C3R* par ses voisins directs. Le maintien de la connectivité et la couverture du réseau sont réalisés grâce au principe de cette approche, ainsi, les nœuds destinés à remplacer le nœud défaillant se déplacent par un va-et-vient entre leur position et la position du nœud défaillant. La figure 2.6 démontre le fonctionnement de *C3R* dans le cas du nœud n9 tombant en panne. Après la détection du nœud *n9* en panne, ses nœuds voisins directs *n1* : *n2*, *n3*, *n10*, *n11* perdent la connectivité et se déplacent vers l'intérieur pour élaborer un plan de reprise après la défaillance du nœud en question. Etant donné que le nœud *n11* est le plus proche de *n9*, il l'atteint plus rapidement que les autres nœuds voisins et agit en tant que coordinateur comme le montre la figure 2.6 (b). Les nœuds récupèrent le calendrier de restauration du nœud *n11* et reviennent à leur position, à l'exception du nœud *n2*, qui est le premier nœud prévu pour la restauration. La figure 2.6 de (d) à (g) montre comment les différents voisins de *n9* se déplacent successivement, en va-et-vient, pour réparer le réseau de la défaillance du nœud *n9*.

Il est à noter que à cause des mouvements physiques des nœuds voisins participant à la réparation du réseau, ces derniers seront dans l'état d'épuisement d'énergie. Ceci pourrait faire tomber plusieurs nœuds en panne en un temps réduit, et de mettre en péril le réseau tout entier.

Cependant, prolonger la durée de vie du réseau représente une contrainte majeure à respecter par chaque approche proposée dans les *RSCFs*. Par conséquent, la fonction de remplacement du nœud défaillant doit être partagée par plusieurs voisins avec la technique du tour de rôle ou l'utilisation d'un remplacement à la chaîne.

- La méthode proposée dans [98], appelée *AOM*, et dont l'exécution est réalisée selon 5 étapes, à savoir :
    1. La mise en œuvre du réseau.
    2. La détection des voisins et la détection des nœuds défaillants.
    3. La restauration du réseau.
    4. Déterminer les nœuds critiques et l'ensemble des dominants de la connectivité *CDS* (*Connectivity Dominating Set*), et ce, après l'exécution de la première étape.
    5. Déterminer une politique de restauration pour chaque nœud critique [101].

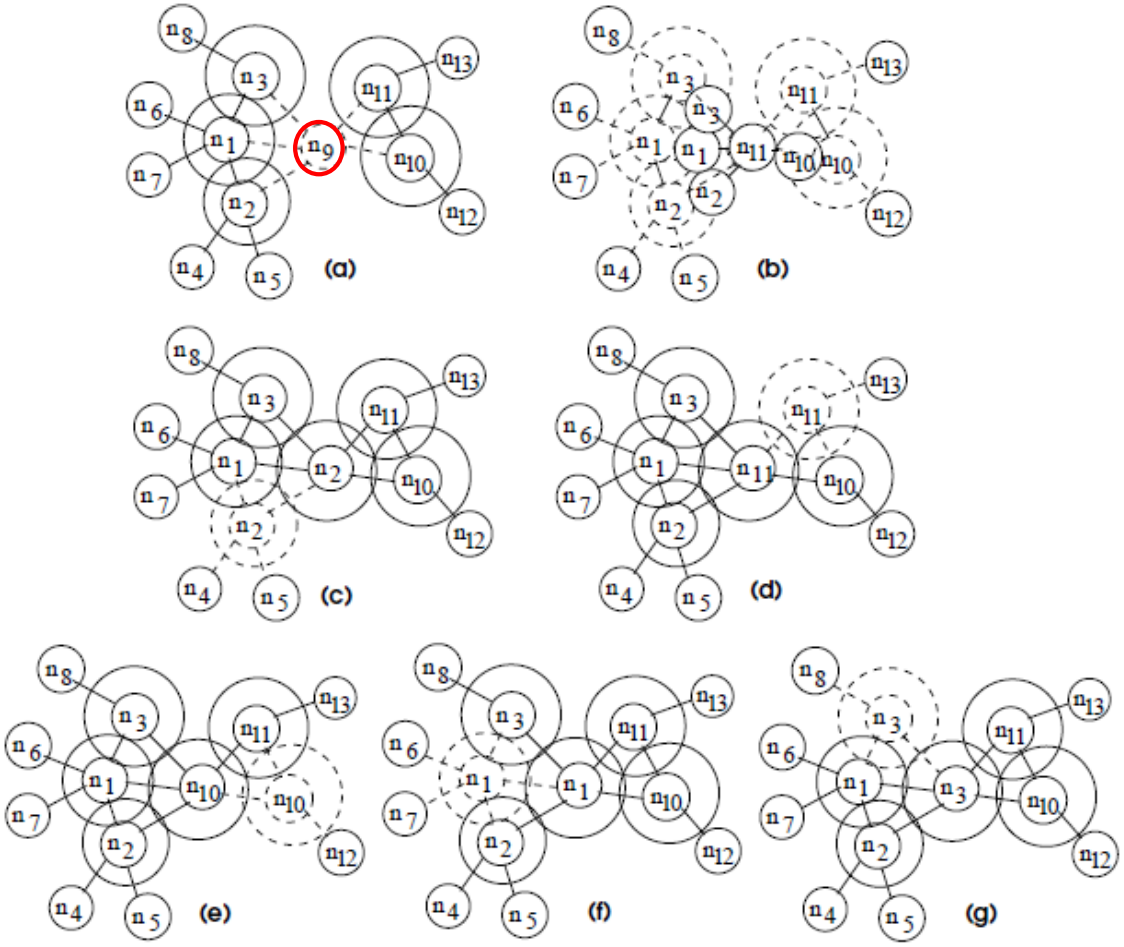


Figure 2.6 : Exemple de fonctionnement de l'approche C3R [97].

- Dans [101], l'auteur considère le nœud critique comme celui dont la défaillance partitionne le réseau et que la défaillance d'un nœud dominant est celle qui ne partitionne pas le réseau. Cependant, les auteurs de cette approche ont utilisé respectivement la méthode de localisation proposée dans [101] qui détecte rapidement le nœud critique, et la méthode des *CDS* proposée dans [102] pour la sélection des dominants.

A travers l'application de la méthode proposée *AOM*, ils peuvent déterminer les nœuds critiques qui peuvent remplacer le nœud défaillant pour ne pas partitionner le réseau, la méthode des *CDS* garantit, quant à elle, la non défaillance du nœud de remplacement de celui en situation de panne.

2.6.2 Solutions de tolérance nœud défaillants par protocoles de routage

Dans les *RCSF*, les capteurs peuvent ne pas fonctionner correctement à cause de l'occurrence des nœuds capteurs défaillants, due à leurs ressources limitées d'énergie, l'épuisement de leurs

batteries, ou la destruction physique/logique accidentelle ou intentionnelle, etc. Ceci résulte des difficultés qui encapent l'acheminement des données collectées à la station de base.

D'un autre côté, dans les *RCSFs,* le routage est une fonctionnalité cruciale s'appuyant sur le processus de transmission et de communication entre les nœuds capteurs et la SB. En effet, pour remédier à ces problématiques, des approches de remplacement de nœuds défaillants et qui sont orientées vers des solutions avec protocoles de routage ont été proposés dans plusieurs travaux.

La plupart de ces solutions existantes sont menées pour le maintien de la connectivité des *RCSFs* comme les travaux présentés dans [105], [104], dont nous discuterons par la suite les limites.

- **Le protocole de routage dynamique** [103] **:** la solution de routage avec ce protocole est proposée avec l'objectif de maintenir la connectivité du réseau, quand un nœud est sur le point d'épuiser son énergie et ce, pour assurer la livraison de données à la station de base tout en prolongeant la durée de vie du réseau [104]. Ceci est réalisé par le capteur en question avec l'effort de trouver un chemin alternatif pour établir une nouvelle connexion avec ses nœuds voisins. En effet, ce chemin augmente la fiabilité de transmission de données. L'exécution de ce protocole se fait en trois phases :
  1. Mise en œuvre et établissement de chemin.
  2. Transmission de données.
  3. Rétablissement de chemin.

Ce protocole [104] montre ses limites à travers la durée de temps écoulée par les phases de son exécution pour trouver un nouveau nœud voisin, affectant de la sorte la durée de livraison de données.

- **Protocole de routage tolérant aux pannes multi-niveaux (*FMS*) [105] :** En plus de l'objectif de base ayant trait au maintien de la connectivité du réseau, cette solution de routage avec le *FMS* permet d'assurer la fiabilité et la rapidité de livraison des données à la station de base, même si un nœud capteur en question risque de tomber en panne à cause de l'épuisement de son énergie. Ce protocole est conçu pour les applications orientées événement. Dans un *RCSF*, le déploiement des capteurs est aléatoire à travers une densité, qui implique une redondance dans la livraison de données. Par conséquent, ceci affecte la durée de vie du réseau de capteurs. Afin de remédier à cette limite, *FMS* permet un ordonnancement d'activité des capteurs par la mise en veille de certain nombre de capteurs sans affecter la fiabilité de livraison de données et tout en économisant l'énergie. Le protocole *FMS* effectue deux opérations de base :
  1. Détermination des niveaux des nœuds et établissement de chemin.
  2. Ordonnancement d'activité des capteurs et transmission de données.

Il est à noter que le protocole *FMS* présente les mêmes limites que le protocole de routage dynamique [104] avec une différentiation sur ses performances remarquables d'après les travaux de [105] mais se dégradant toutefois dans un environnement réel

- **AODV tolérant aux pannes (*ENFAT-AODV*) :** *ENFAT-AODV* [106] est un protocole de routage qui permet la tolérance aux pannes, l'auto-démarrage et le routage multi-sauts entre les nœuds. *ENFATAODV* établit les plus courts chemins entre les nœuds en nombre de sauts. En outre, il permet aussi aux nœuds d'établir un chemin de secours quand le chemin principal présente des ruptures. Toutefois, dans *ENFAT-AODV,* il y a des champs additionnels, si l'on compare ce protocole à la version originale du "*AODV*". Certains champs sont ajoutés dans les paquets de contrôle tels que "*BACKUP*" (dans *RREQ* et *RREP*), "*UPDATE*" dans *RREQ* et "Distance" dans *RREQ*. *ENFAT-AODV*. Il réduit également la complexité de mise en œuvre par suppression des paquets de contrôle inutiles dans le réseau et il s'exécute comme suit :
  3. Découverte du chemin principal.
  4. Entretien de la route.

La limitation de ce protocole concerne essentiellement la consommation de l'énergie à cause de l'inondation des messages de contrôle (*RREQ*, *RREP*), et ce, en plus des champs ajoutés sur le protocole de base *AODV* (*BACKUP*, *UPDATE*) [103].

2.6.3 Solutions tolérance aux nœuds défaillants basées sur le *clustering*

Dans un *RCSF* les protocoles du *clustering* divisent le réseau en un ensemble de clusters ayant chacun un *cluster Head* « chef de groupe de nœuds capteurs », donc les données collectées par les capteurs appartenant à ce cluster seront récupérées par le *cluster-head* pour les acheminer vers la station de base. Les traitements effectués sur les données, dans chaque cluster, avant de les propager vers la station de base, permettent à cette solution d'alléger la quantité d'informations qui circule dans le réseau. Les protocoles permettant ce type de solution pour la tolérance aux nœuds défaillants sont présentés dans les travaux [106, 107, 108, 109, 110]

- **Protocole CPEQ [108] :** *CPEQ* (*Cluster-based PEQ*) [108] est la variante du protocole *PEQ* [106], dont le mécanisme de tolérance aux nœuds défaillants est implémenté dans ce dernier protocole. L'objectif d'implémentation du protocole *CPEQ* à base du *PEQ* est d'ajouter d'autres fonctionnalités en plus de celle de la tolérance aux nœuds défaillants. Dans les RCSFs avec clustering, chaque capteur du réseau peut devenir agrégateur (*clusterhead*) pendant une certaine période de temps selon son niveau de batterie. Cependant, CPEQ permet d'offrir une meilleure gestion de routage, de distribuer d'une manière uniforme la dissipation d'énergie

entre les capteurs, et de réduire la latence et le trafic de données dans le réseau, ceci est réalisé grâce au mécanisme de clustering. A cet effet, le protocole *CPEQ* s'exécute en cinq étapes :

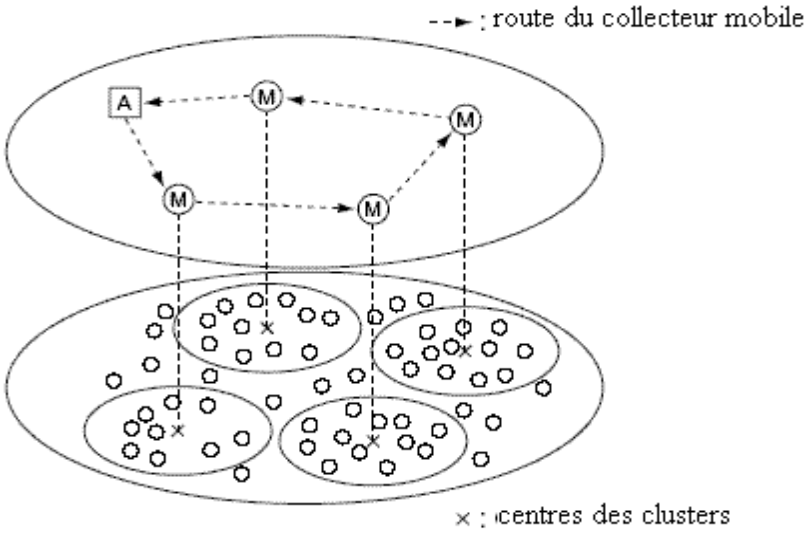
- *Configuration initiale* : dans cette phase, *CPEQ* introduit un champ additionnel à PEQ contenant le pourcentage des capteurs qui deviendront *clusterhead*.
- *Sélection d'agrégateur :* C'est la phase d'élection des *clusterheads*.
- *Configuration de clusters :* c'est la phase de formation des clusters
- *Transmission de données à l'agrégateur :* Chaque capteur utilise sa table de routage pour envoyer la donnée vers son agrégateur.
- *Transmission de données au collecteur :* Après la réception des données depuis les capteurs de son cluster, l'agrégateur doit acheminer en multi-sauts ces données au collecteur.

- **Algorithme K-CDS [109] :** Avec *K-CDS*, le réseau de capteurs est modélisé par le graphe $G = (V, E)$ où V est l'ensemble des capteurs et E est l'ensemble des liens sans fil entre ces capteurs.
  - *Définition 1 :* un réseau G est *k-connexe* si chaque deux capteurs du graphe sont connectés par au moins k chemins disjoints. Par conséquent, le réseau peut tolérer la défaillance d'un nombre de capteurs inférieur à k.
  - *Définition 2 :* un sous-ensemble V', V' est un *k-DS* (*k-dominating Set*) de G si chaque nœud de V qui n'appartient pas à V' a au moins k voisins dans V'. Le *k-DS* V' devient *k-CDS* si le sous-graphe $G = (V', E')$ est *k-connexe*.

L'approche utilisée par *K-CDS* est préventive et basée sur le clustering, proposant une construction d'un backbone virtuel avec l'ensemble k-connexe dominant *k-CDS*, et ce, pour offrir un routage efficace aussi bien qu'une bonne tolérance aux nœuds défaillants. Pour cela, quatre approches ont été introduites ; dont deux sont des algorithmes probabilistes, une déterministe, et la dernière une hybridation des approches déterministe et probabiliste.

- **KAT-Mobility [110] :** En plus du concept de clustering utilisé dans *KAT-mobiliy* (*K-means And TSP-based mobility*) [110], on trouve le concept de mobilité implémenté au niveau des *SB*. Ces deux mécanismes représentent une technique préventive hybride tolérante aux nœuds défaillants offrant, en résultat, une meilleure gestion d'énergie et augmentant ainsi la durée de vie du réseau. Le principe de *KAT-mobility* se résume en deux procédures : le clustering et l'optimisation du routage, vu que la méthode proposée utilise la technique de collecteurs mobiles. En effet, après une réorganisation du réseau en clusters, le collecteur

mobile se déplace vers les centres des clusters à travers un chemin optimal afin de récupérer les données depuis les capteurs des clusters visités. La figure 2.7 illustre le principe de fonctionnement de la solution *KAT-mobility*.



La figure 2.7 : Le principe de fonctionnement de *KAT-mobility* [110].

Les deux principales de procédures de fonctionnement de *KAT-mobility* se résument comme suit :

- *Algorithme de clustering* : Avec *c*ette procédure l'ensemble des *N* capteurs dans un *RCSF* est divisé en *k* clusters *C1, C2, ..., Ck*. Le coût du cluster est évalué par l'erreur approximative entre le collecteur et les nœuds capteurs.
- Soit $d\ (x, y_i)$ cette erreur, où x est un capteur et y_i est un collecteur ($i = 1, 2, ..., k$). $d\ (x, y_i)$ est définie par la distance euclidienne entre le capteur et le collecteur.
- Le but de cette phase est d'affecter chaque capteur au *cluster-Head* le plus proche.
- *Optimisation du routage :* Cette phase consiste à retrouver le chemin optimal pour le nœud mobile, cette technique ressemble à celle du voyageur de commerce (*TSP*), un collecteur représentant le voyageur, et les centres des clusters définissant les villes. L'optimisation de la route du collecteur mobile consiste à visiter tous les centres des clusters une et une seule fois. La méthode *KAT-mobility* [100] a prouvé sa performance de fournir une meilleure conservation d'énergie aussi bien qu'une bonne tolérance aux nœuds défaillants.

2.7 Conclusion

Afin de mettre en place voire intégrer un *RCSF* dans l'*IdO* pour de différentes applications de surveillance, nous devons apporter tout d'abord une solution au problème de défaillance des nœuds capteurs qui peuvent se présenter dans un *RCSF* et peuvent tout aussi être dans le domaine de l'*IdO*, qui reflète la continuité de service de surveillance même dans ce type de réseau.

Après avoir présenté une brève description sur la tolérance aux pannes en particulier pour les *RCSF* qui sont composés de capteurs trop sensibles aux pannes, les types de pannes existantes, les procédures de remplacement de nœuds en panne ou défaillants, nous avons exposé la classification des approches et les solutions de routage possibles et les plus efficaces sur le concept de la tolérance aux nœuds défaillants qui puissent exister pour traiter le problème de capteurs défaillants dans les *RCSFs*.

Dans le chapitre suivant sera présentée une nouvelle technique de réparation tolérante aux nœuds défaillants dans les *RCSFs* intégrés dans l'*IdO* à l'aide de la technique de capteurs mobiles et avec le nouveau protocole *RPL* doté du mécanisme de réparation locale et globale de la topologie de réseau.

Partie 2

Contributions

Proposition d'une méthode de gestion et réparation locale dans les RCSFs

Chapitre 3

3.1 Introduction

L'IdO a émergé comme une tendance de recherche prometteuse ciblant un grand nombre d'applications. D'autre part, l'intégration des *RCSFs* dans l'*IdO* permet à de nouvelles applications d'y apparaître tout en respectant les exigences et les caractéristiques conceptuelles des *RCSF*. Cependant, il est devenu nécessaire de suivre les technologies avancées de communication, ce qui rend les *RCSFs* omniprésents à l'égard de tous les utilisateurs et environnements. En outre, l'idée des *RCSFs* consiste à intégrer certains capteurs informatiques embarqués dans l'infrastructure Internet. Par ailleurs, l'*IdO* est constitué de certains périphériques qui sont intrinsèquement mobiles. D'un autre, la norme *6LoWPAN* s'est avérée utile car elle a facilité le déploiement de l'*IdO* en offrant des avantages telles que l'évolutivité, la flexibilité, et la connectivité de bout en bout. Une telle exigence est satisfaite grâce à l'utilisation de RCSF basés sur la technologie *6LoWPAN*, caractérisée par les ressources limitées de leurs appareils.

Cette thèse se concentre, dans sa première partie de contribution, développée dans ce chapitre-ci, sur le contrôle et la gestion de la défaillance des nœuds dans le réseau, tout en intégrant le concept de mobilité pour remplacer les nœuds en question. La mobilité des nœuds est pertinente avec un protocole de routage. Cette pertinence est valable aussi pour la réparation locale des *RCSF* basés sur la technologie *6LoWPAN* qui n'est offerte qu'avec le protocole *RPL*.

Dans ce chapitre, nous avons mis l'accent sur le protocole *RPL*, le reste de ce chapitre est organisé comme suit : dans la section 3.2, nous présentons une description approfondie du protocole de routage conçu pour les réseaux à consommation d'énergie et à perte de données (RPL), ainsi que celle de son support pour la mobilité et son mécanisme de réparation de défaillance.

Ensuite, à la section 3.3, nous élaborons notre synthèse et nos propositions pour cette thèse, puis nous exposons la première partie de notre contribution dans la section 3.4.

Enfin, la section 3.5 sera consacrée à la description de la méthode proposée pour la gestion et réparation des nœuds défaillants avec l'utilisation des nœuds mobiles, à la présentation de l'environnement de simulation sous le simulateur *Cooja* et à l'évaluation de cette méthode sur les différents résultats de performances obtenus par des scénarios de simulation.

3.2 RPL : Le nouveau protocole de routage pour les *RCSFs*

RPL (*Routing Protocol for Low-Power and Lossy Networks*) [111], est un protocole de routage à vecteur de distance (*VD*) destiné aux *RCSF* à faible consommation d'énergie et à faible perte de données dans l'Internet des Objets. *RPL* est conçu pour fonctionner sur les couches *PHY* et *MAC IEEE 802.15.4*, il se concentre sur la maintenance d'une table de routage indiquant la prochaine décision de saut pour le transfert de paquet.

3.2.1 Concept de routage

Pour les protocoles de routage il existe deux classes [85] : 1) le routage basé sur le vecteur de distance (*VD*), et 2) le routage basé sur l'état du lien.

- **Routage à vecteur de distances (*Distance Vector*)**

Ce routage détermine la direction (vecteur) et la distance (nombre de sauts) vers les nœuds destinataires. Avec cette classe de routage, le chemin de routage est donné par défaut, représentant le chemin le moins coûteux en termes de distance. Ces protocoles se basent sur un échange, entre nœuds voisins, et chaque nœud envoie à ses voisins des informations concernant la liste des destinations qui lui sont accessibles et le coût correspondant. Le nœud récepteur met à jour sa liste locale des destinations avec les coûts minimums. Le processus de calcul se répète, s'il y a un changement de la distance minimale séparant deux nœuds, et cela jusqu'à ce que le réseau atteigne un état stable. Les protocoles de routage basés sur le vecteur de distance les plus connus pour les réseaux ad hoc sont : *DSR, DSDV6, AODV* et le nouveau protocole *RPL*.

- **Routage à état de liens (*Link state*)**

Ce routage est connu sous le nom *shortest path first*, et dont les chemins de routage sont définis via la Carte topologique du réseau, il se base sur les informations rassemblées et disséminées périodiquement sur l'état des liens dans le réseau. Chaque routeur détermine le meilleur chemin, ce qui permet de détecter le chemin le plus court vers la destination. Un nœud

qui reçoit les informations concernant l'état des liens, met à jour sa vision de la topologie du réseau et applique un algorithme de calcul des chemins optimaux afin de choisir le nœud suivant pour une destination donnée. Les principaux protocoles de routage dans les réseaux ad hoc qui appartiennent à cette classe sont les suivants : *TORA, OLSR, …* etc.).

Pour les *RCSFs* basés sur la technologie *6LoWPAN*, le routage par vecteur de distance semble le plus approprié en raison du moindre calcul de complexité et du faible coût de signalisation des messages de contrôle, comparé au type de routage d'état de liens. De plus, un protocole de routage peut être proactif ou réactif. Le routage proactif consiste à construire les premiers chemins de routage vers toutes les destinations possibles. Ainsi, les nœuds trouvent immédiatement une route disponible quand elle s'est nécessaire, ce qui le rend approprié pour les applications en temps réel.

Cependant, le processus proactif est coûteux en termes de performances de coût de signalisation, alors que les protocoles de routage réactifs ne découvrent des routes que lorsque cela est nécessaire. Ainsi, il n'a pas besoin de plus de coûts de signalisation, mais il augmente la latence et peut provoquer une perte de données. Pour la technologie *6LoWPAN*, deux types de protocoles de routage existent, (comme déjà illustré dans la figure 1.15 et à voir dans la sous-section 1.5.4.1 page 47) : 1) le *Mesh Under*, dont le routage est défini dans la couche inférieure (la couche d'adaptation), 2) le Route Over, définit dans la couche supérieure (la couche réseau).

Avant le développement et l'apparition de la technologie *6LoWPAN*, le routage, dans les RCSFs, repose sur le routage hiérarchique à travers une topologie arborescence avec des clusters (*Cluster-Tree*), utilisant le routage d'état de liens qui ne peut pas prendre en charge la complexité des problèmes de maintenance [112]. Par la suite, après l'exploitation de la technologie *6LoWPAN*, de nombreuses recherches ont été proposées afin de trouver un protocole adapté aux réseaux avec pertes [113]. Dans ce contexte, plusieurs protocoles de routage sont proposés tels que : *Hydro* [114], *Hilow* [115], *LOAD* [118] et *Dymo-low* [117]. Cependant, certaines lacunes sont toujours présentes. Concernant les RCSFs avec leurs ressources limitées des capteurs, ainsi que les exigences critiques des applications en termes de latence et le taux de transmission des données [116], le groupe de travail *IETF* (*ROLL*) [120] [6] a proposé un protocole de routage pour les de réseaux avec pertes de données et à faible puissance. Nous nous référons à ce protocole (*RPL*), car il définit le mécanisme de réparation locale et globale de la topologie du réseau par l'effet des nœuds défaillants, ainsi que son support de la mobilité.

3.2.2 Protocole *RPL* : Description

RPL est un protocole de routage proactif à vecteur de distance utilisant *IPv6* [169], standardisé à l'*IETF* par le groupe de travail *ROLL* [119]. Il a été spécialement développé pour répondre aux besoins des réseaux à faible consommation d'énergie, pouvant aller jusqu'à plusieurs milliers de nœuds déployés dans des environnements tels que les réseaux urbains, les réseaux domotiques, les réseaux industriels. Ce protocole est optimisé pour les réseaux de collecte de données qui sont basés sur trois types de trafics : Point-à-Point, Point à-Multipoint et Multipoint-à-Point. RPL est basé sur un graphe hiérarchique et multi-sauts pour le transfert de données, à travers un graphe en arbre (tree-graph) créé à l'aide de certains messages de signalisation *ICMPv6*. En outre, *RPL* est un protocole *Route-Over*, qui permet d'effectuer le routage sur plusieurs types de couches de liaison, telles que *IEEE 802.15.4*, *IEEE 802.15.4g*, *WiFi* et Power Line Communication (*PLC*). Certaines caractéristiques de la *RPL* sont résumées dans le tableau 3.1. Dans cette section, nous présentons le fonctionnement de ce protocole et les mécanismes de réparation de pannes ou nœuds défaillants existant.

Caractéristique	**Valeur**
Topologie	Hiérarchique
Type	Proactive
Algorithme	Vecteur de distance
Messages de signalisation (ou trafics de contrôle)	*ICMPv6*
Réparation locale	utilisée
Nœuds	Statique/ Mobile
Scalability	Elevée
Mémoire utilisée	faible
Energie utilisée	faible
Trafics supportés	*MP2P - P2MP - P2P*
Support d'*IPv6*	oui
Cadres de disponibilité	*ContikiRPL - TinyRPL*

Tableau 3.1 : Caractéristiques du protocole *RPL* [121].

3.2.3 Topologie *RPL*

La topologie *RPL* est basée sur le concept du graphe acyclique dirigé (*DAG*) en tant que graphe sans cycles. C'est une topologie dynamique et optimisée avec l'évitement des boucles et la considération des paramètres de qualité de service (*QdS*) pour l'acheminement des datagrammes *IPv6* depuis et vers les nœuds capteurs.

Les nœuds RPL s'interconnectent en formant une topologie spécifique appelée *DODAG* (*Destination Oriented Directed Acyclic Graph*), c'est-à-dire un graphe acyclique orienté dirigé vers une destination qui est la racine du réseau. Un réseau *LoWPAN* contient au moins une instance RPL qui elle-même se compose d'un ou plusieurs DODAGs. Chaque instance RPL est associée à une fonction objectif (*Objectif Function (OF*)) qui permet d'optimiser la topologie en fonction d'un ensemble de contraintes et/ou de métriques comme la préservation de l'énergie.

Dans certains cas, le même réseau physique *6LoWPAN* devrait être optimisé pour supporter plusieurs applications ayant chacune son propre graphe (communément appelée instance *RPL*), construit selon une métrique de routage bien déterminée. La métrique peut être le niveau d'énergie résiduelle des nœuds capteurs dans le réseau *6LoWPAN*, le nombre de transmissions nécessaire pour atteindre la racine (*EXT*), le délai moyen des communications, le taux de pertes, etc.

Un nœud peut faire partie d'un seul *DODAG* par instance, comme il peut participer à plusieurs instances simultanément (voir figure 3.1). Chaque *DODAG* est identifié par un *ID DODAG* (128 bits). Chaque *DODAG* comprend :

- **Dispositif *RFD* (***Reduced Function Device***) :** Représentant les nœuds terminaux qui collectent les données et les transmettent vers le nœud récepteur.
- **Routeur *FFD* (***Full Function Device***) :** Ce sont des nœuds pouvant générer et acheminer des données vers le nœud récepteur.
- **Racine *DAG* (***destination Oriented DAG root (DODAG Root)***)** : Désigne le nœud récepteur et la passerelle pour Internet.

Chaque nœud dans le *DoDAG* est assigné par une valeur numérique appelée le rang (Rank), comme illustre la figure 3.2 et qui définit sa position par rapport au nœud racine. Le numéro de rang est également utilisé pour détecter et éviter les boucles de routage (*routing loops*). Ce paramètre est calculé par *OF* et il est augmenté dans le sens descendant de la racine *DoDAG* (ayant le moins de rang) aux nœuds feuille s(terminaux).

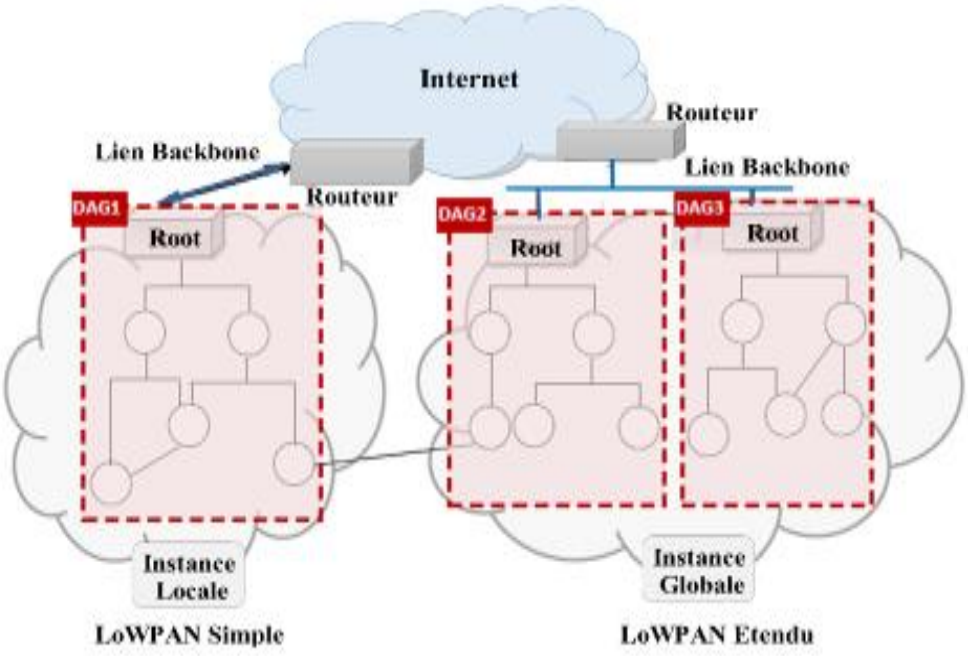


Figure 3.1: Topologie d'un réseau *RPL*.

Comme l'illustre la figure 3.1, toutes les racines *DODAG* sont interconnectées par un *backbone* commun (*Iink Backbone*). Le réseau *6LoWPAN* se compose d'une ou plusieurs instances *RPL* identifiées par un *RPL Instance-ID* (8bits). Deux types d'instance *RPL* peuvent être identifiées : 1) l'instance *RPL* globale qui se composent de plusieurs *DoDAG*, 2) l'instance *RPL* locale qui n'a qu'un seul *DODAG*. Un nœud peut appartenir à plus d'une Instance *RPL*, mais à un seul *DoDAG* pour chacune d'elles [122]. Par conséquent, il peut avoir différents rôles comme un nœud routeur dans une certaine instance de *RPL* et un nœud feuille dans d'autres.

De plus, un nœud racine peut appartenir à plus d'un *DODAG* dans différentes Instances *RPL*, cela est connu sous le nom de routage multi-topologique (*MTR*) (voir la figure 3.2). Ce concept permet de fournir des chemins basés sur différentes fonctions, métriques et contraintes. Par exemple, pour un même nœud racine de destination, un nœud peut avoir des chemins différents avec différents objectifs, à savoir : le chemin avec le moins de latence, ou le chemin suivant des nœuds avec la ressource énergétique la plus élevée, entre autres.

Dans la structure *DODAG* et contrairement à l'arbre typique, un nœud peut être associé à plusieurs nœuds parents et associé à des nœuds frères de même rang, et chaque nœud de routeur stocke une liste de parents et de frère qui aident dans le routage des paquets vers le nœud racine, mais il doit néanmoins sélectionner un parent préféré (*Prefered Parent* (*PP*)) pour le chemin par défaut.

La sélection des *PP* est effectuée à l'aide de la fonction objectif (*OF*), qui permet de sélectionner l'itinéraire optimisé avec le chemin le plus court suivant une métrique ou une contrainte définie. Selon cette topologie, *RPL* envisage de combiner à la fois des topologies maillées et hiérarchiques. Ainsi, les paquets peuvent être routés via la relation fils/parent ou par le biais d'un frère de même rang. Par conséquent, *RPL* offre une grande fiabilité et une grande flexibilité avec plusieurs possibilités de chemins. De plus, chaque nœud peut passer d'un *DODAG* à un autre, avec une modification sur l'ensemble des nœuds parents, mettre à jour son nouveau *Rang* et une nouvelle sélection de *PP*. *RPL* peut prendre en charge le changement de topologie dynamique causé par la mobilité ou la défaillance de certains nœuds en utilisant rarement les listes de parents et de frères de même rang.

- Les nœuds parents (ou routeurs), sont des nœuds parents qui possèdent au moins un fils (un nœud de *Rang* inférieur). Le nœud parent assure la circulation des informations entre le nœud *Root* et ses fils.
- Les nœuds feuilles, ce sont des nœuds qui ne possèdent aucun fils. Ils sont seulement connectés à leur nœud parent qui les transmettent les informations en provenance du *Root* du *DODAG*.

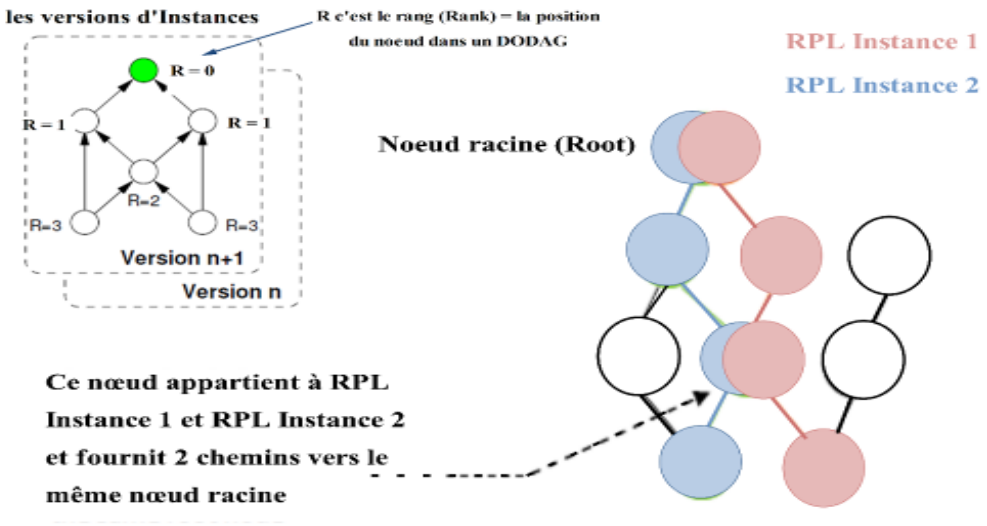


Figure 3.2 : Routage Multi-Topology (*MTR*).

3.2.4 Construction de la topologie *DODAG*

Lors de la construction d'un *DODAG*, les nœuds s'échangent des messages *ICMPv6* [122, 123], en effet, *RPL* utilise quatre types de ces messages à savoir : *DIO* (*DODAG Information Object*), *DIS* (*DODAG Information Solicitation*), *DAO* (*DODAG Destination Advertisement Object*) et *DAO-ACK* (*DAO Acknowledgement*).

- **Message *DIO*** : est utilisé pour construire et maintenir une topologie *DODAG*, à partir de laquelle il fournit le transfert ascendant. Ce message est initié par le nœud racine (*Root*), puis, diffusé d'une manière multicast à tous les nœuds du *DODAG*.
- **Message *DIS* :** est utilisé pour vérifier les nœuds voisins dans le voisinage, demandant la réception *DIO*.
- **Message *DAO* :** est utilisé pour fournir l'itinéraire inverse pour le transfert vers le bas par le biais d'un envoi unicast aux parents.
- **Message *DAO-ACK* :** est utilisé comme réponse sur les messages *DAO*, afin d'éviter la boucle *DAO*.

3.2.4.1 Concepts de construction de la topologie

Le concept de la construction du graphe *DODAG* formé par le protocole *RPL* est une topologie de routage logique construite à partir d'un réseau physique, utilisant un certain nombre de critères fournis par l'administrateur du système. Ci-dessous, nous présentons les principales étapes conceptuelles de construction d'un *DODAG*, à partir d'un réseau physique donné.

- ***1ère étape*** *:* La topologie *DODAG* est initié par le nœud racine (*Root*), dont il diffuse les messages *DIO* dans son voisinage à un saut, le message *DIO* incluant son instance *RPL*, son *DODAG-ID*, son numéro de version *DODAG-Ver*, son rang (*Rank*), le coût du chemin, la contrainte ou la métrique et la fonction objectif (*FO*). La figure 3.3 illustre cette première étape de construction.
- ***Deuxième étape :*** Comme illustre la figure 3.4, chaque nœud ayant reçu le *DIO* envoyé par le *Root* rejoint le *DoDAG*, dont il établit une mise à jour des paramètres (calcul de rang (*Rank*), la liste des nœuds parents mise à jour par l'expéditeur du message *DIO*, sélection du parent préféré (*PP*) sur la base de la *FO* et paramètres dans le *DIO*). Une fois le parent préféré choisi, le nœud envoi à ce parent un message *DAO* pour signaler son choix, que le message *DAO* sera envoyé au nœud racine via l'ensemble des parents pour l'itinéraire inverse. Ces messages *DAO* contiennent des informations permettant au *Root* de construire des chemins vers les nœuds feuilles du *DODAG*.
- ***Troisième étape :*** Les nœuds ayant choisi le *Root* comme parent vont à leur tour diffuser des messages *DIO* en multicast dans leur voisinage. Les nœuds ne faisant pas encore partie du *DODAG* qui vont recevoir ces messages *DIO*, vont donc faire un choix de nœuds parents

puis répondent par des messages *DAO* indiquant leur attachement au *DODAG*. La figure 3.5 montre le principe de cette étape.

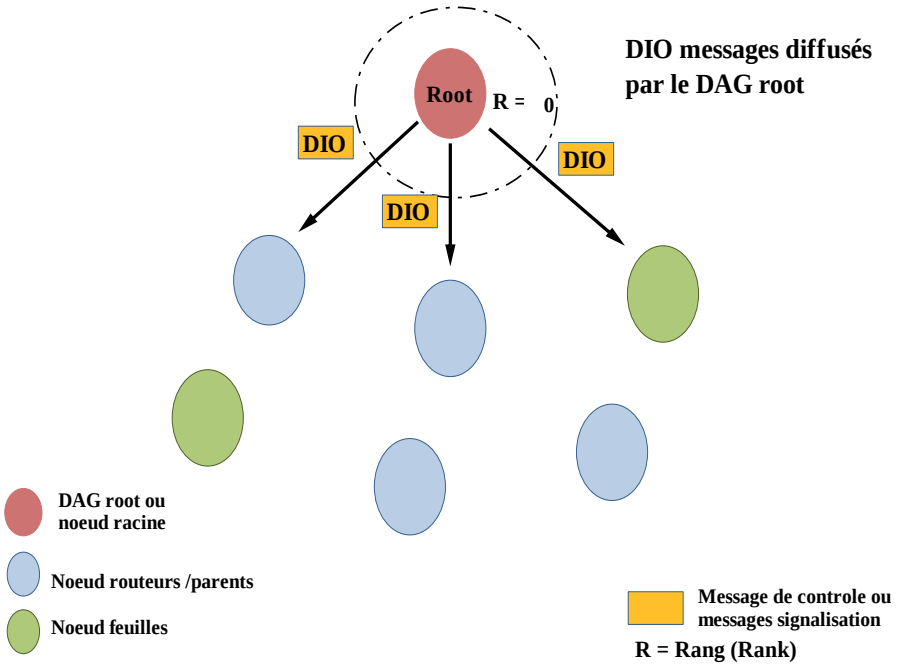


Figure 3.3 : Etape 1, diffusion des *DIO* par la racine (*DAG Root*).

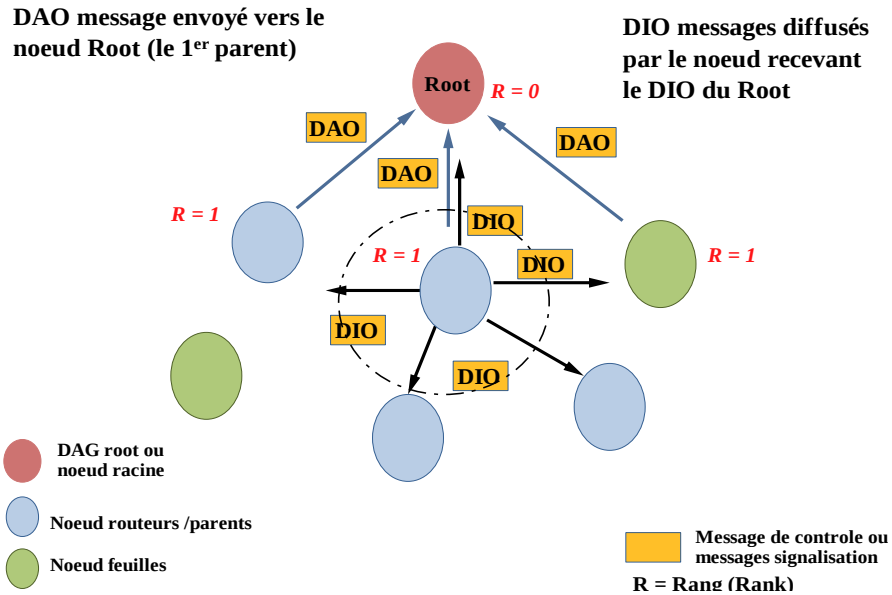


Figure 3.4 : Etape 2, Choix du Root comme nœuds parent.

Ensuite, s'il s'agit d'un routeur, il transmet ce message de façon multicast après la mise à jour des valeurs des champs. Un nœud rejette un message *DIO*, si le rang de l'expéditeur est supérieur à son rang afin d'éviter le routage en boucle (*Rooting Loop*) [124], ou s'il provient

d'un numéro de version différent. De plus, un nœud doit diffuser un message DIS pour confirmer la réception d'un message *DIO*, s'il ne reçoit pas de message *DIO* et n'appartient à aucun *DODAG*.

D'un autre côté, un nœud utilise un message *DAO* indiquant "No-Path" pour supprimer un chemin vers le bas, lorsqu'il perd sa disponibilité pour cette cible. Ce processus est illustré dans l'algorithme 1.

Enfin, dans le but de maintenir la topologie *DODAG*, *RPL* effectue un envoi périodique des messages *DIO* à l'aide de l'algorithme *trickle timer* [125], qui est ajusté suivant l'état du réseau afin d'optimiser les messages de signalisation de contrôle (*overhead)*.

RPL fonctionne sur la base d'un graphe orienté, aussi les routes qui vont vers la racine (*upward*) ne sont-elles pas nécessairement les mêmes que celles qui descendent vers les nœuds (*downward*) [124].

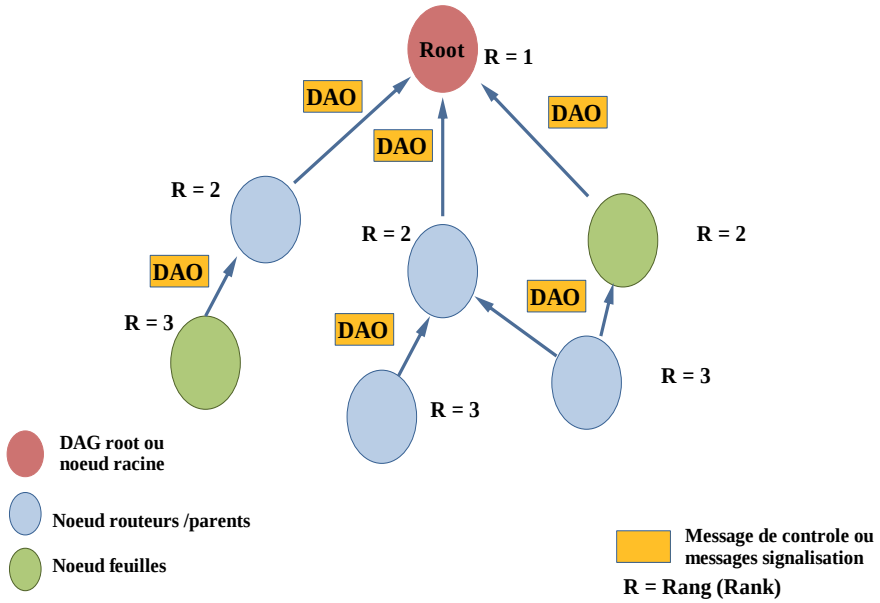


Figure 3.5 : Etape 3, Poursuite de la construction du *DODAG*.

3.2.4.2 Algorithme *Trickle timer*

Contrairement aux protocoles de routage traditionnels, qui utilisent un processus périodique pour la maintenance des routes et qui est coûteux en termes de messages de contrôle ou de signalisation, *RPL* utilise un mécanisme intelligent et adaptatif, appelé "*Trickle timer* " [125]. Ce mécanisme est un temporisateur ajusté d'une façon dynamique en fonction des besoins du réseau dans le but d'optimiser la transmission des messages *DIO*. En outre, les routeurs

configurent les intervalles d'émission des messages de contrôle en utilisant l'algorithme de *Trickle timer*.

Ce temporisateur permet aux nœuds du réseau *6LoWPAN* d'échanger les informations et les données d'une manière simple, évolutive, tout en économisant de l'énergie et assurant la bonne transmission. L'idée de base d'un temporisateur *Trickle* est d'envoyer des *DIOs* plus fréquemment quand il s'agit d'une détection d'incohérence dans un *DODADG*. Tant qu'un routeur reçoit des informations qui sont "*Consistentes*", il va augmenter d'une façon exponentielle son intervalle de transmission, jusqu'à atteindre une fréquence minimum prédéfinie. Pour *RPL*, "Consistent" signifie que le message *DIO* envoyé par un expéditeur avec un rang inférieur ne provoque aucun changement pour les paramètres suivants : l'ensemble des parents destinataires, le parent préféré, ou le rang.

Par contre un événement "*Inconsistent*" peut se produire quand il s'agit d'un événement qui change la structure d'un *DAG*, citons par exemple une détection d'une boucle de routage, une réception d'un message *DIO* indiquant une nouvelle version du *DODAG* ou nouvelle instance *RPL*, une sélection d'un nouveau parent ou bien un lien parental devenant inaccessible. Dans le cas d'un tel événement, le temporisateur *Trickle* sera réinitialisé et l'intervalle de transmission est mis à un minimum prédéfini [125].

3.2.4.3 Les identifiants de topologie RPL

Dans les *RCSFs*, les routes *RPL* sont optimisées pour la transmission du trafic sur une topologie définie par un *DAG* et répartie sur un ou plusieurs *DODAGs* de manière à avoir un *DODAG* par racine. Ainsi, pour identifier et maintenir une topologie, *RPL* utilise quatre identifiants comme la présente la figure 3.6 [123].

- ***RPLInstanceID* :** Il identifie un ensemble d'un ou plusieurs *DODAGs*. Un réseau peut posséder plusieurs *RPLInstanceIDs* dont chacune définit une organisation indépendante des *DODAGs* et qui peut être optimisée avec des fonctions objectif différentes. L'ensemble des *DODAGs* identifiés par une *RPLInstanceID* est appelé une instance *RPL*. Tous les *DODAGs* avec la même instance *RPL* utilise la même Fonction Objectif (*FO*).
- ***DODAGID*** : La combinaison *RPLInstanceID* et *DODAGID* identifie un *DODAG* unique dans un réseau. Une instance RPL peut avoir des *DODAGs* multiples et chaque *DODAG* a un *DODAGID* unique.

Algorithme 1 Traitement du nœud à la réception d'un message *DIO*

```
if First DIO then
    Adds the sender to the parent list
    Computes Rank
    identify the PP as the sender
    Broadcasts DIO
   Sends DAO to the root node through the parent set
else
  if DIO Rank < node Rank && same version number then
      Updates parent list
      Updates Rank
      Selects a new PP
      Broadcasts DIO
      Sends DAO to the root node through the parent set
      Sends DAO No-Path to the root node through the previous PP
    else
      discard the DIO message
  Endif
Endif
```

- ***DODAGVersionNumber* :** Cet identifiant peut avoir des valeurs différentes au sein d'un *DODAG*. Un *DODAG* est parfois construit à partir de la racine en incrémentant la valeur du *DODAGVersionNumber*. Le triplet (*RPLInstanceID, DODAGID, DODAGVersionNumber*) identifie d'une façon unique une version *DODAG*.

- ***Rang* (*Rank*) :** Pour une version d'un *DODAG,* il existe un autre identifiant *RPL*, c'est la valeur du rang d'un nœud. Cette valeur est une représentation scalaire qui définit la position d'un nœud par rapport à la racine d'un *DODAG*. Le calcul du rang est considéré parmi les fonctionnalités de la *FO*. Ce calcul dépend de plusieurs facteurs à savoir, l'ensemble des parents, les métriques des liens et des nœuds, la configuration des nœuds. Il est utilisé également pour éviter et détecter les boucles. Le rang présenté par un nœud doit avoir nécessairement une valeur supérieure à l'ensemble des valeurs des rangs présentés par ses parents. Dans ce cas, il n'y a pas de risque de création d'une boucle (*Routing Loop*). Un réseau possédant des nœuds avec des valeurs du rang similaire peut déclencher une boucle quand un nœud choisit un chemin vers un nœud du même rang. La valeur du rang diminue vers le sens montant au fur et à mesure qu'on s'approche du nœud racine, et augmente dans le sens inverse.

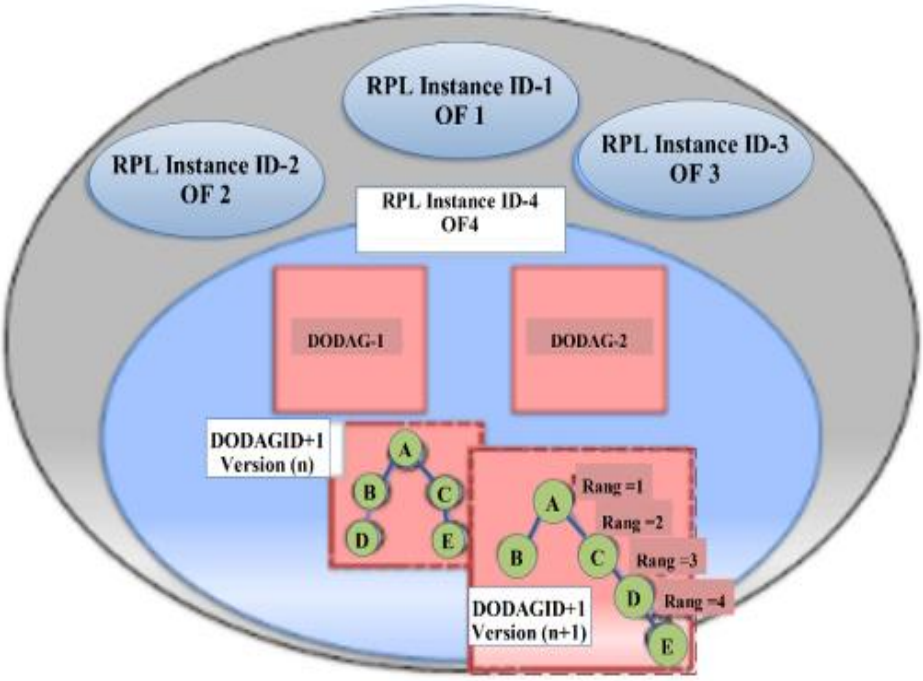


Figure 3.6 : Les identifiants de la topologie RPL.

3.2.4.4 Fonction Objectif (*FO*)

Le protocole de routage *RPL* dans des *RCSFs* forme des *DODAGs* avec des instances, ou chaque instance est associée à une fonction objectif (*FO*) désignée pour résoudre les problèmes abordés par l'instance *RPL*. La *FO* ou *Objectif Function* (*OF*) représente le concept le plus important de RPL sur lequel se base la construction de la topologie *RPL*. La fonction est identifiée par *Objective Code Point* (*OCP*) avec l'option de configuration des messages *DIO* et contribue à plusieurs opérations.

En effet, *FO* définit comment les nœuds RPL sélectionnent et optimisent les chemins avec une instance *RPL* pour joindre un *DODAG*. Le concept de la *FO* consiste à définir une seule *FO* pour une telle instance *RPL*. En effet cette fonction utilise quelques métriques et contraintes de routage statiques et dynamiques prédéfinies par le nœud racine dans les messages *DIO* [126], pour le calcul du Rang, le calcul du coût de chemin et la sélection par défaut des parents préférés (*PP*) selon la figure 3.7. Le *PP* est sélectionné à partir de la liste des parents, suite à l'optimisation d'un chemin basé sur un objectif prédéfini tel que le chemin évitant les nœuds critiques alimentés par batterie, le chemin avec le délai minimum de bout en bout. Il est à noter que les routes sont basées sur une métrique et une contrainte pouvant concerner des liens, ainsi que des nœuds.

Certaines fonctions objectifs sont proposées à savoir : *MRHOF* (*Minimum Rank with Hysteresis Objectif Function*) [128], *l'ETXOF* [129] et la *Fonction Objectif Zéro* (*OF0*) [130] [127]. Le *MRHOF* se concentre sur la sélection du chemin avec le coût de chemin minimum, en tenant compte du concept d'hystérésis, afin d'améliorer la stabilité.

Par conséquent, le nœud change son itinéraire par défaut uniquement si la différence entre le chemin minimal et les coûts du chemin actuel atteint un seuil prédéfini.

ETXOF [129] est similaire au *MRHOF* avec l'utilisation de la métrique *ETX* (*Expected Transmission Number*). *ETX* est le nombre attendu de la transmission requise pour transmettre avec succès un paquet sur le lien. *ETXOF* se concentre sur la sélection du chemin le plus économe en énergie, grâce à la réduction du nombre de retransmissions de paquets, ou *OF0* est la fonction objective de base utilisé par défaut dans *RPL*. *ETXOF* n'utilise aucune métrique dans son processus, en insistant sur le fait qu'il n'y a aucune garantie pour l'optimisation de l'itinéraire selon une métrique spécifique. Ainsi, il est basé sur le paramètre *Rang* (*Rank*) dans son processus de sélection.

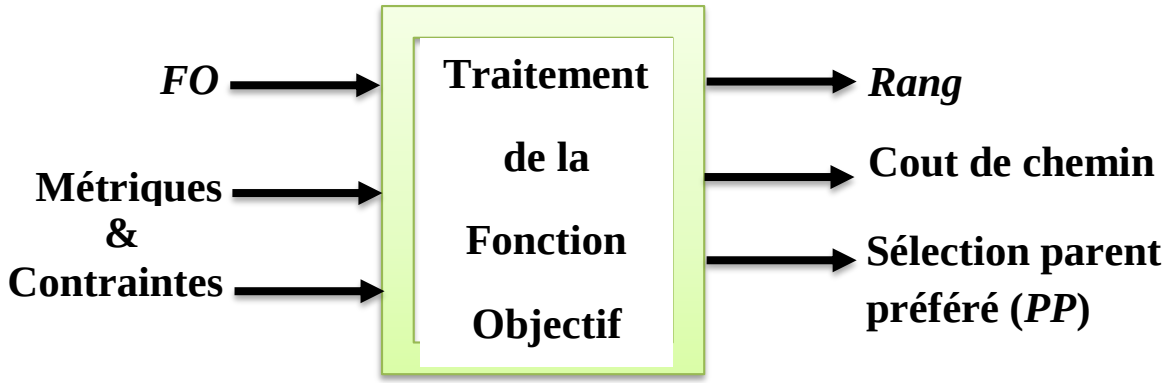


Figure 3.7 : Processus de la fonction objectif.

3.2.4.5 Les métriques et contraintes du routage RPL

La construction du graphe acyclique orienté (*DAG*) par *RPL* est basée sur des métriques et contraintes de routage. Une métrique de routage est une valeur quantitative qui est utilisée pour évaluer le coût du chemin avec des contraintes de qualité. Certaines caractéristiques de liaison ou nœud (fiabilité des liaisons, taux d'énergie d'un nœud) peuvent être utilisées par *RPL* soit comme des contraintes de routage ou métriques [126].

L'ensemble des métriques de routage et des contraintes utilisées par *RPL* est signalé le long du *DODAG* construit suivant des règles introduites dans la *FO*. *RPL* peut être utilisé pour construire des *DODAGs* avec des caractéristiques différentes, par exemple, il peut être

souhaitable de construire un *DAG* avec l'objectif de maximiser la fiabilité en utilisant la métrique de fiabilité des liaisons pour calculer le meilleur chemin. Un autre exemple est la caractéristique d'énergie d'un nœud utilisée comme une contrainte nœud lors de la construction d'un *DODAG* [126].

Nous pouvons ainsi spécifier quelques métriques et contraintes pour les nœuds :

- **Énergie :** Comme il est souhaitable d'éviter de sélectionner un nœud avec de faible énergie ainsi l'énergie est considérée comme une contrainte de base pour le routage. Dans de tels cas, RPL peut calculer un chemin plus long qui satisfait cette contrainte afin d'augmenter la durée de vie du réseau vu que la puissance et l'énergie sont considérées comme des ressources critiques dans la plupart des *RCSFs*.

- **Le nombre de sauts :** Le nombre de sauts (*Hops-Number*) est utilisé pour indiquer le nombre de nœuds traversés le long du chemin, il peut être utilisé comme une contrainte ou une métrique. Comme contrainte, la racine du *DODAG* indique le nombre maximal de sauts établis dans un chemin. Lorsque ce nombre est atteint, aucun autre nœud ne peut joindre ce chemin. Comme métrique, chaque nœud visiteur incrémente tout simplement ce champ. Nous pouvons également spécifier quelques métriques et contraintes pour les liens.

- **Le débit :** La valeur du débit permet de signaler le débit d'une liaison. Le débit peut être utilisé comme une métrique additive pour signaler un maximum ou un minimum ou comme une contrainte pour n'autoriser que les chemins qui fournissent un débit minimum.

- **La latence :** Cette valeur est utilisée pour signaler la latence dans un chemin. Comme le débit, la latence peut être utilisée comme une métrique ou une contrainte. Lorsqu'elle est utilisée comme une métrique, sa valeur exprime la latence totale (métrique additive), la latence minimale ou maximale le long du chemin. Utilisé comme une contrainte, le temps de latence servirait à exclure les liens qui fournissent une latence plus grande qu'une valeur prédéfinie.

- **La fiabilité :** Dans les *RCSF*, la fiabilité des liens pourrait être dégradée pour un certain nombre de raisons, citons par exemple l'atténuation du signal et les interférences. Un changement dans la qualité du lien peut affecter la connectivité réseau, ainsi, la qualité d'un lien peut être prise en compte comme une métrique de routage critique. Il existe trois indicateurs de fiabilité de liaison et qui se présentent comme suit [126] :

1) ***Le niveau de qualité de liaison (LQI) :*** *LQI* (*Link Quality Indicator*) est utilisé pour quantifier la fiabilité d'un lien en utilisant une valeur discrète qui varie dans l'intervalle [0,7], où le scalaire "0" indique un niveau de qualité inconnue et le scalaire "1" présente un lien de meilleure qualité.

2) ***Le nombre de Transmission Attendues (ETX) :*** *ETX* est le nombre de transmissions qu'un nœud s'attend à réaliser à une destination, afin de livrer avec succès un paquet.

3) ***Couleur du lien (LC) :*** *LC* est un objet de lien utilisé comme une contrainte ou métrique pour éviter ou suivre des liens spécifiques pour des types du trafic spécifique.

3.2.5 Types d'acheminement du trafic

Le concept de la topologie *RPL* permet de fournir différents types d'acheminement [126] :

A) La transmission multipoint à point pour la collecte de données

Ce type fait référence à l'envoi vers le haut (*upward*), il est établi au cours de la construction du *DODAG* en définissant la route par défaut vers le nœud racine via le parent préféré. Ces routes sont gérées par les messages *DIO*.

B) La transmission point à multipoint pour la diffusion des données

Ce type fait référence à l'expédition vers le bas (*downward*), il est établi grâce aux messages *DAO* envoyés d'une manière unicast, vers le nœud racine *DODAG* via l'ensemble de nœuds parents. Ces messages sont envoyés lorsqu'un nœud reçoit un message *DIO*, afin de mettre à jour les informations de routage suite au changement de topologie. En fonction de la capacité des nœuds en termes de mémoire et de la taille éventuelle du réseau, le protocole *RPL* offre deux modes pour l'expédition vers le bas, le mode en "*storing mode*" et celui en "*Non-Storing mode*" :

- **Le mode de stockage " *Storing mode*" :** La figure 3.8 illustre ce mode de stockage. Il consiste à sauvegarder des informations de routage dans la table de routage de chaque nœud routeur intermédiaire, y compris le saut suivant pour chaque retransmission de paquets. Les informations de la table de routage sont extraites du message *DAO*, lorsqu'il est envoyé au nœud racine. Chaque nœud transmet son message à son parent direct (parent à un saut) qui maintient une table de routage à son niveau.

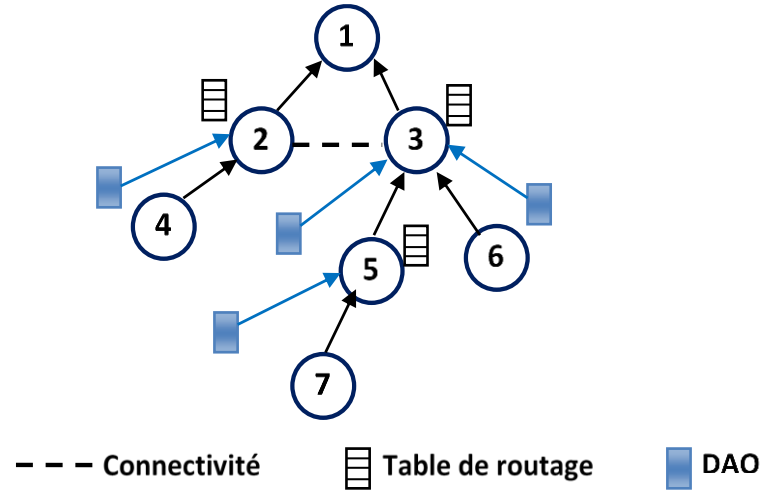


Figure 3.8 : Fonctionnement en "*Storing-mode*" du protocole *RPL*.

- **Le mode sans stockage "*Non-Storing mode*" :** Le mode sans stockage consiste à sauvegarder les informations de routage sur le nœud racine [126]. Ainsi, les nœuds de routeur intermédiaires qui reçoivent le message *DAO* ne stockent aucune information. Ils insèrent simplement leurs adresses dans le champ d'option Informations de transit du message *DAO* transféré. Ensuite, les informations de routage sont stockées sur le nœud racine à partir des informations collectées dans le message *DAO*. Pour le transfert de paquets, le nœud racine ajoute les informations de chemin dans l'en-tête de routage source (*SRH, Source Routing Header*) du paquet. Chaque nœud intermédiaire vérifie l'en-tête de l'information et l'envoie au bond suivant jusqu'à ce qu'il atteigne sa destination. Ce mode est utilisé pour éviter la grande table de routage dans les nœuds, en raison de leurs contraintes en mémoire. La figure 3.9 illustre ce mode de fonctionnement.

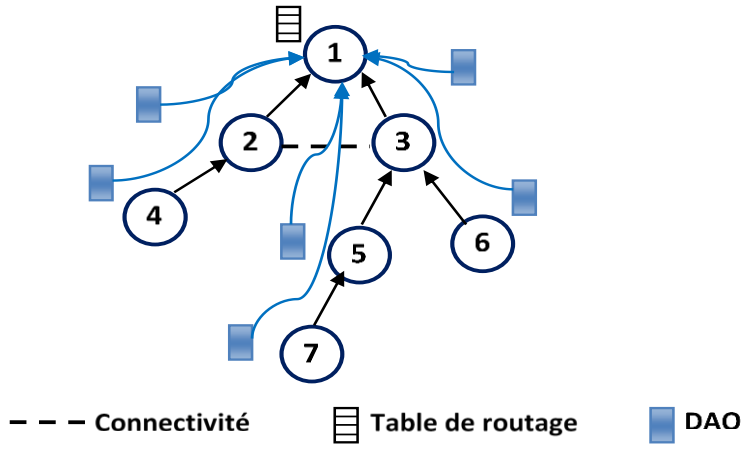


Figure 3.9 : Fonctionnement en "*Non-Storing mode* "du protocole *RPL*.

C) La transmission point à point pour la lecture des données

RPL permet l'acheminement point à point de différentes manières. En direct, lorsque le nœud expéditeur et le nœud de destination sont dans la même portée de transmission (*Range*), le paquet étant directement transmis en utilisant un seul saut. Sinon, pour le mode de stockage, les paquets du nœud expéditeur sont routés vers un parent commun au nœud expéditeur et destinataire en fonction des tables de routage dans les routeurs. Ensuite, les paquets sont acheminés jusqu'au parent commun, puis vers la destination. Cependant, pour le mode sans stockage, le transfert point à point nécessite l'implication du nœud racine, puisque les informations de routage sont enregistrées dans ce nœud. Par conséquent, les paquets atteignent le nœud racine, puis retournent à la destination.

Il convient de noter que l'acheminement point à point reste une recherche ouverte, car *RPL* ne définit pas de routes optimales pour ce type de fonctionnement.

3.2.6 Gestion du réseau *RPL*

RPL est un protocole proactif, et ce, pour faire face au changement de topologie, qui est causé par des incohérences soudaines telles que la défaillance du nœud ou la boucle de routage. Ses caractéristiques et son concept de topologie aident à fournir des stratégies pour gérer le changement de topologie et améliorer son impact sur les performances du réseau [126].

3.2.6.1 Auto-récupération

RPL peut gérer le changement de topologie grâce à l'auto-récupération, qui consiste à mettre à jour de manière dynamique les décisions de routage et à modifier la topologie. Ce processus peut être établi par l'envoi périodique des messages *DIO* basés sur le *Trikle timer*, le mécanisme de découverte de voisins utilisé pour vérifier la disponibilité de certains routeurs, ou par le concept de topologie qui permet de changer dynamiquement les chemins de routage avec un autre parent ou nœud frère de même rang.

3.2.6.2 Boucle de routage (*Routing Loop*)

RPL prend en charge l'évitement de l'occurrence de la boucle de routage (*Routing Loop*) dans le processus de construction de la topologie, grâce au paramètre rang (*Rank*). Ce paramètre est utilisé pour organiser la topologie hiérarchique, dont un nœud a un rang supérieur à tous ses rangs parents. Par conséquent, le rang est augmenté dans la direction aval, dans le but de permettre l'acheminement de paquets dans une direction, sans retour au nœud émetteur.

Cependant, cette lacune peut se présenter dans certains cas lorsque les messages de signalisation sont perdus, ou en cas de mouvement du nœud.

3.2.6.3 La technique de réparation avec protocole *RPL*

RPL dispose de deux mécanismes pour réparer la topologie de *DODAG*, qui sont complémentaires, la réparation globale et locale.

A) Réparation locale

RPL est le premier protocole qui effectue une phase de réparation locale afin de surmonter le problème et d'atténuer ses conséquences telles que la perte de données. Telle la détection d'un problème d'incohérence qui provoque la déconnexion du nœud du reste du graphe de la topologie. Par exemple, si un nœud détecte une défaillance de lien ou de nœud, le nœud détecteur tente de réparer l'itinéraire en passant par un frère ayant le même rang ou le nœud routeur parent. La réparation locale autorise le *DODAG* réparé dans la version *DODAG*, dans laquelle chaque nœud peut se détacher du *DODAG*, déplace son parent vers sa table de routage, localise le sous-*DODAG* en annonçant le rang par la valeur de l'infini, et à la fin, il se rattache à l'original ou à un nouveau *DODAG*. Cette technique présente le risque de créer une boucle de routage**.** Ce type de réparation se concentre sur un autre chemin, même s'il n'est pas optimal. Ensuite, si ce processus ne peut pas aboutir à la résolution du problème, une réparation globale est déclenchée par le nœud racine [131].

B) Réparation globale

La réparation globale n'est déclenchée qu'à partir de la racine, c'est un mécanisme de reconstruction graphique d'une nouvelle version du *DODAG* à partir de zéro. La racine *DAG* commence à incrémenter le numéro de version du *DODAG* lors de l'envoi de *DIO* pour une nouvelle version *DODAG* [131]. Lorsque les nœuds reçoivent *DIO*, ils n'acceptent que le *DIO* dont le numéro de version soit supérieur ou égal au numéro actuel. Ce numéro de version assure que les informations circulant dans le réseau sont à jour, dans la mesure où les anciens *DIO* sont écrasés (en *crush*) par les plus récents. Cependant, la réparation globale est une technique d'optimisation, mais elle a un coût de trafic de contrôle supplémentaire dans le réseau et le délai de réparation dépend du taux de rafraîchissement du numéro de séquence, ce qui provoque un impact négatif sur les performances du réseau.

3.2.7 La mobilité sous le protocole *RPL*

Depuis son apparition, le protocole *RPL* a largement satisfait des exigences des réseaux de capteurs caractérisés par leurs ressources limitées. Cependant certaines problématiques qui lui sont inhérentes demeurent en attente d'éventuelles améliorations. Notamment ce qui concerne la prise en charge de la méthode de mobilité pour différents besoins des *RCSFs* basés sur la technologie *6LoWPAN*. En effet, le protocole *RPL* offre de nombreux avantages, notamment sa fiabilité quand il s'agit de permettre différents types de retransmission, sa souplesse pour adapter le chemin de routage aux exigences, son souci de fournir les performances requises, sa capacité à gérer le réseau et la réparation des liens ainsi que les nœuds défaillants dans le réseau de graphe *DODAG*.

En outre, *RPL* a été conçu à l'origine pour les réseaux statiques, sans support pour la mobilité. En effet, il n'est pas bien adapté aux environnements dynamiques. Ainsi, la mobilité est devenue une exigence pour certains types d'applications dans l'internet des objets, ce qui entraîne des déconnexions fréquentes du réseau et un changement de topologie. La fourniture d'une assistance à la mobilité s'avère donc nécessaire.

D'un autre côté, un routage robuste et fiable dans un type de réseau hybride comme celui des RCSFs basés sur la technologie *6LoWPAN* est toujours l'un des défis les plus importants dans les réseaux de distribution de contenu à temps réel. Cependant, nous allons nous concentrer, dans la section suivante, sur la mise en évidence des problèmes du concept de la mobilité sous le protocole *RPL*, de ses conséquences et des directives esquissées afin de l'exploiter avec le mécanisme de réparation locale défini par *RPL*, en fonction de liens et nœuds défaillants, ressources limitées et de la bonne performance de fonctionnement d'un *RCSF* basé sur *6LoWPAN*.

3.2.8 Positionnement bibliographique

IETF ROLL [119] a adopté un nouveau protocole appelé protocole de routage *IPv6* pour le routage *RPL* (*Low Power and Lossy Networks*) [5]. Ce protocole définit deux techniques, qui sont complémentaires dans la nature et les actions en cas de défaillance de lien et de nœud connues sous le nom de réparation locale et globale. De plus, le support de mobilité pour le protocole *RPL* a toujours été un sujet de recherche difficile car il a été initialement conçu pour les réseaux statiques, sans support pour la mobilité. Par ailleurs, le but de ce chapitre est de proposer une nouvelle méthode de réparation locale efficace avec *RPL* dans les *RCSF* basés sur la technologie *LoWAPN* afin de remplacer les nœuds défaillants par des nœuds mobiles.

Cependant, gérer le mécanisme de réparation du *RPL* avec des nœuds mobiles dans les réseaux de capteurs est un véritable défi.

Pour ces raisons, plusieurs approches et techniques de support de mobilité pour le protocole *RPL*, ainsi que des améliorations effectuées sur ce protocole, ont été proposées dans la littérature.

Au cours de cette section, nous allons nous concentrer sur les travaux les plus importants menés sur le protocole *RPL* dans différents aspects de la recherche et précisément dans les aspects suivants :

1. L'évaluation des performances de *RPL* en mode statique
2. L'évaluation des performances de *RPL* en mode dynamique (mobilité)
3. Etude de réparation locale *RPL*.
4. Protocoles avec support mobilité dans les *RCSF* basés sur *6LoWPAN*

3.2.8.1 L'évaluation des performances de *RPL* en mode statique

Dans le premier aspect, une étude approfondie sur la mise en œuvre du *RPL* a été menée pour fournir des idées et des lignes directrices pour l'utilisation de cette norme [131], en utilisant le simulateur *COOJA* sous Contiki [132]. Les auteurs dans [132] ont réalisé une étude de simulation pour évaluer le processus de construction du *DAG* dans le protocole de routage *RPL*. Il n'y a pas beaucoup d'évaluation sur la réparation locale ou la gestion des défaillances avec RPL. L'accent étant mis uniquement sur la performance de ce protocole.

3.2.8.2 L'évaluation des performances de *RPL* en mode dynamique (mobilité)

Dans le second aspect, nous citons les travaux qui apportent leurs contributions dans le support de la mobilité par le protocole *RPL*. Parmi ces travaux, on trouve [135, 136, 137, 138,139], dans lesquels, les auteurs suggèrent la méthode de mobilité pour améliorer les performances *RPL* dans un réseau dynamique.

- Ces auteurs dans [135], ont évalué la performance de *RPL* dans trois scénarii différents : 1) évaluer les caractéristiques de *RPL* avec des nœuds fixes, (*SB* ou *Sink*) et d'autres qui seraient des expéditeurs), 2) ajouter de la mobilité, en effet, l'auteur compare les nœuds mobiles aux nœuds fixes afin de montrer comment la mobilité peut influencer sur les paramètres du protocole *RPL* et 3) étudier le comportement de *RPL* lorsque le réseau est dense afin d'évaluer les performances du protocole. Pour cet article [135], l'auteur montre clairement que le modèle de mobilité du point d'acheminement aléatoire (*Random Waypoint mobility model (RWP)*) donne de meilleures métriques que le modèle de mobilité aléatoire (*RWK*) en

termes de nombre de sauts et de valeur de transmission attendue (*ETX*). En outre, dans le modèle *RWP*, les nœuds consomment plus de 10,73% d'énergie que celle consommée avec *RWK*.

- Les auteurs dans [136], proposent un nouveau protocole inter-couches (*cros-layer*) fonctionnant dans les couches 2 et 3 (liaison et réseau) connu sous le nom mobility-triggered *RPL* (*MT-RPL*). Pour maintenir une connectivité efficace avec le réseau, le *MT-RPL* proposé permet d'évaluer parallèlement avec la détection d'inaccessibilité des voisins et la détection de transfert bidirectionnel grâce à une large simulation avec différents scenarii. Les résultats de ce nouveau protocole de couche *Cros-Layer* montrent que *MT-RPL* réduit considérablement le temps de déconnexion, ce qui augmente le taux de livraison de paquets et réduit la consommation d'énergie par paquet de données.
- Dans l'étude [137], les auteurs visent principalement à proposer des améliorations à la spécification standard afin de fournir des garanties de *QoS* pour les *RCSF* à base *6LoWPAN* statiques et mobiles. Les auteurs ont proposé un nouvelle fonction objectif (*OF-FL*), qui surmonte les limitations des *FOs* normalisées et qui ont été conçues pour RPL en considérant des métriques de lien et de nœud importantes, à savoir le délai de bout en bout, le nombre de sauts, le nombre de transmissions attendues et le niveau de qualité de liaison (*LQL*). Ces auteurs ont utilisé en même temps la conception de la *Co-RPL* expliquée et présentée dans [134], où il propose une extension de la RPL pour soutenir la mobilité (*Co-RPL*). A l'objectif d'améliorer les performances du réseau, l'extension proposée par l'auteur permet de conserver la trace des positions des nœuds mobiles lors de leurs déplacements. Pour permettre la localisation des nœuds RPL en mouvement, l'extension s'appuie sur le mécanisme Corona via une étude de simulation utilisant le simulateur *COOJA* sous Contiki.
- La plupart des travaux basés sur la mobilité des nœuds suggèrent d'améliorer la durée de vie du réseau ou la conservation de l'énergie [138], mais uniquement avec la mobilité du sink (Root).
- Dans [139], un nouveau protocole cross-layer (*MT-RPL*) est proposé, ce protocole *MT-RPL* diffère de celui proposé dans [139] car il bénéficie du protocole *MAC XMachiavel*, qui privilégie les nœuds mobiles souhaitant transmettre des données. L'auteur analyse également la distribution de paquets pour le trafic de la racine à un nœud mobile.
- De nouvelles recherches dans [134] proposent une *RPL* avec découverte de voisin améliorée qui optimise le routage d'un nœud mobile vers un *Sink* statique. Cette solution utilise le mécanisme *DIS*, c'est une fonction d'estimation de qualité de lien adaptée.

- Dans ce second aspect de recherche, les auteurs n'ont pas utilisé la mobilité pour les mécanismes de réparation *RPL* en cas de défaillance de lien et de nœud. En effet, ces travaux ont porté uniquement sur les avantages de performance du protocole *RPL* par l'intégration du concept de la mobilité dans le mécanisme de routage du *RPL*, que ce soit par proposition des nœuds routeurs d'acheminement des donnée mobiles ou le cas du sink (*Root*) mobile, et ce, en vue de garantir une *QoS* par la proposition de nouvelles extensions du *RPL*.

3.2.8.3 L'évaluation de la réparation locale dans RPL

Le troisième aspect de recherche concerne la méthode de réparation locale définie par *RPL*. La réparation dans un *RCSF* devient une clé pour soulager tout protocole de routage, faisant référence à la capacité de réparer la topologie de routage.

- [140] présente une analyse expérimentale des mécanismes de correction d'erreur *RPL* en utilisant l'implémentation de *Rally Contiki* avec un agent *SNMP* pour surveiller les performances de *RPL*. Dans [141], les mesures d'évaluation de la performance du RPL sont simulées dans une sous-station de réseau intelligent extérieure typique avec l'utilisation des scénarii réels de réparation locale. Ils n'ont pas utilisé la mobilité pour les mécanismes de réparation *RPL* en cas de défaillance de lien et de nœud.

L'auteur dans [142], propose un algorithme efficace pour les agents mobiles tolérants aux fautes.

- Dans [143], l'auteur aborde la composition de service tolérant aux pannes avec une attention particulière à la qualité de service, ou une stratégie tolérante aux fautes est ainsi proposée pour améliorer la performance de la composition du service. La stratégie est composée d'un mécanisme d'invocation, d'un mécanisme de synchronisation et d'un mécanisme d'exception. La gestion et l'application du concept de mobilité sont présentées et décrites dans [144] pour le *Big Data* mobile. Certaines applications mobiles dans des environnements d'hétérogénéité sont présentées avec le travail [145].

3.2.8.4 Protocoles de support mobilité pour les *RCSF* à base *6LoWPAN*

Dans les *RCSF* basés sur la technologie *6LoWPAN*, certains protocoles sont proposés pour résoudre le problème de la défaillance des liens et des nœuds capteurs dans un routage multi-sauts, pour lequel la consommation d'énergie est une caractéristique majeure. Il existe plusieurs travaux dont le contexte de recherche se base sur la proposition de différents protocoles de routage qui peuvent supporter la mobilité des nœuds et répondre aux exigences de certaines applications. Nous allons citer les plus récents et les plus importants dans cette sous-section.

- On trouve dans [146], que le problème de la défaillance des communications multi-sauts est traité par la combinaison du concept correct de *MIPv6* et du protocole de routage *OLSR* (*MIPv6 + OLSR*). Dans [147] [148], le problème est résolu par la combinaison des protocoles *NEMO* et *HWSN*. *NEMO-HWSN* est créé pour les applications de santé afin d'éviter le problème de goulot d'étranglement rencontré avec le protocole *NEMO* dans l'entité routeur mobile (*RM*)). Ainsi, son concept consiste à partager la fonctionnalité du *RM* avec d'autres types de dispositifs *FFD-MN* (*Full Function Device- Mobile Node*). D'autres protocoles sont proposés pour assurer la communication multi-sauts avec les nœuds statiques (*Statique Node* (*SN*)) déployés dans le *6LoWPAN*, où ils sont chargés de suivre le nœud mobile et les paquets de routage de/vers lui.
- Parmi ces protocoles, citons Inter-*PAN* [149] [150], *LoWMob* [151], *LoWMob* distribué *DLoWMob* [149], *Inter-Mobility* [152], *Mobile IP-Based* [153] et *RPL-Weight* [138] pour la mobilité des nœuds et *Scheme Based Based* [154] pour la mobilité des réseaux. Les critères utilisés pour ces protocoles sont représentés dans [121], l'impact de la mobilité sur les performances du réseau est présenté dans le tableau 3.2 selon les travaux de recherche dans [121].

3.3 Synthèse des discutions

Après notre étude et notre analyse des travaux existants, le principal constat à mentionner est qu'il n'y a pas de solution efficace avec le concept de mobilité pour répondre à toutes les exigences et à la contrainte relative à la tolérance aux nœuds défaillants ou aux pannes, en général, dans les *RCSF* basés sur la technologie *6LoWPAN*. Ainsi, certaines améliorations sont encore nécessaires pour répondre aux exigences de certaines applications dans ce type de réseau. Dans cette thèse et dans sa première partie de contribution, nous nous sommes concentrés sur l'amélioration du mécanisme de réparation locale du protocole *RPL* tout en utilisant et exploitant le concept de la mobilité des nœuds dans un *RCSF* basé sur la technologie *6LoWPAN*.

Par conséquent, l'intégration de la mobilité des nœuds dans le mécanisme de réparation locale du protocole *RPL*, représente un challenge soulevé dans ce chapitre et il est nécessaire de noter que la mobilité des nœuds ne peut être traitée sans tenir compte du protocole de routage et son support à ce concept. En effet, le groupe de travail *IETF ROLL* a proposé un protocole de routage pour les réseaux à faible puissance et à perte de données appelé "*RPL*". Ce protocole

est pris en charge dans cet écrit pour soutenir la réparation des nœuds défaillants avec le concept de mobilité des nœuds dans les *RCSFs* basés sur la technologie *6LoWPAN*.

Dans la section suivante, nous présentons en détail notre méthode et notre modèle de gestion de la défaillance avec le protocole *RPL*.

Table 2.4: Mobility Support Protocols for Networks with multi-hop consideration

	Address	Movement detection	Data buffered	Topology architecture	Mobility model	Deploy. of SNs
OLSR+MIPv6/ NEMO-HWSN6 [113]/[114][115]	IPv6	RS/RA	HA	Hybrid: Mesh-Star	Unspecified	Random
LoWMob [72]	Out: IPv6 In: 16-bit short	RSSI	PSN	Hybrid: Mesh-Bus	Random waypoint [117]	Grid
D-LoWMob [72]	Out: IPv6 In: 16-bit short	RSSI	PSN	Hybrid: Mesh-Star-Bus	/Fluid flow [118]	Random
Inter-PAN(1) [70]	Out: IPv6 In: 16-bit short	RSSI	GW	Hybrid: Mesh-Star-Bus	Fluid flow	Grid
Inter-PAN(2) [71]	Out: IPv6 In: 16-bit short	RSSI	NPSN	Hybrid: Mesh-Star-Bus	Fluid flow [118]	Grid
Inter-Mobility [87]	Out: IPv6 In: 16-bit short	RSSI / PAN-ID	Intra: NPA Inter: FA	Hybrid: Mesh-Star	Unspecified	Random
Mobile IP-Based[86]	Out: IPv6 In: 16-bit short	RSSI	unspecified	Hybrid: Tree-star	Random Waypoint	Random
Cluster-Based Scheme [80]	Hierarchical	RSSI	Previous AN	Hybrid: Cluster tree-Bus	Random walk [119]	Grid
RPL-Weight [116]	IEEE 802.15.4	Intended movement	Sink node	Hybrid: DoDAG-Mesh	To computed position place	Grid

Tableau 3.2 : Protocoles de support mobilité à base *6LoWPAN* pour les *RCSFs* [121].

3.4 Méthode de gestion des nœuds défaillants avec le protocole *RPL*

Dans une perspective de tolérance aux pannes ou aux nœuds défaillants et d'économie d'énergie dans les *RCSFs*, la gestion de la défaillance des communications multi-sauts est considérée comme un aspect très important. En outre, la réparation, la gestion du réseau et la vérification de l'efficacité du bon fonctionnement d'un protocole de routage, représentent des caractéristiques conceptuelles et nécessaires à respecter par le protocole sélectionné pour répondre à ces exigences.

Cependant, le protocole RPL offre de nombreux avantages, notamment sa capacité à gérer ainsi qu'à contrôler le réseau, sa fiabilité pour permettre différents types de retransmission, sa souplesse pour adapter le chemin de routage aux exigences, son souci de fournir les performances requises, et enfin sa technique de réparation globale ou locale des liens ainsi que les nœuds défaillants dans le réseau de graphe *DODAG*.

Dans ce contexte, nous proposons comme suite à cette section une stratégie de contrôle et de gestion de la défaillance d'un nœud, en utilisant des nœuds mobiles avec la réparation locale du protocole *RPL*. Cette méthode est représentée par la méthode *MN-LR_RPL* [9] (*mobile node local repair with RPL [9]*).

3.4.1 La méthode *MNLR_RPL*

Cette section décrit notre première partie de contribution dans cette thèse. La méthode proposé dans ce chapitre représente une nouvelle technique de réparation locale *MNLR_RPL* [9] (*Mobile Node Local repair_RPL protocol*), exploitant la mobilité des nœuds dans le réseau de capteurs sans fil avec l'extension du protocole *RPL*.

3.4.1.1 Les identifiants de *MNLR_RPL*

Comme fut présenté dans les sections précédentes, le protocole *RPL* définit la notion de nœud parent, nœud fils et le nœud feuille. En effet, dans la topologie en arbre du protocole *RPL*, chaque routeur de nœud identifie une liste stable de parents ainsi qu'un parent préféré (*PP*). Chaque routeur *RPL* est un prochain saut potentiel sur un chemin vers la « racine » du *DODAG* et chaque nœud terminal est un nœud feuille.

Dans notre méthode *MNLR_RPL*, le nœud ***S*** définit le nœud en panne ou nœud défaillant, le nœud mobile est représenté par ***M***. Le nœud prédécesseur (***Pred***) présente le nœud parent qui fait partie de la liste des nœuds parents définis par *RPL* et la liste des nœuds parents multiples selon l'unique fonction *OF* [124]. Les nœuds feuilles et les nœuds fils de rang supérieur à celui de leurs nœuds parents sont nommés nœuds successeurs (***Succ***) dans *MNLR_RPL*, dont les nœuds feuilles peuvent envoyer les données jusqu'à la racine (*Root*) de la topologie du réseau en transmettant simplement les données à leurs parents immédiats (***Preds***). La liste des nœuds successeurs et prédécesseurs est représentée par ***L***, où $\boldsymbol{L_{Suc}}$ définit la liste des nœuds successeurs et $\boldsymbol{L_{pred}}$ reflète la liste des prédécesseurs. Le tableau 3.3 résume les différents identifiants utilisés dans la méthode *MNLR_RPL*

Pour *MNLR_RPL* nous avons utilisé la liste des nœuds parents et nous avons choisi un nœud qui doit satisfaire l'hypothèse de la procédure 1, définie dans les étapes d'exécution de

l'algorithme, où les nœuds sont censés être mobiles pour remplacer les nœuds défaillants. Pour gérer les nœuds défaillants dans une topologie de réseau, *MNLR_RPL* traite trois cas de défaillance de nœuds tout en proposant une nouvelle technique de réparation locale avec le protocole RPL, utilisant la mobilité des nœuds parents par une procédure de choix basée sur :

1. Le cas de défaillance du nœud feuille dans la topologie du réseau
2. Le cas d'un nœud autre que la défaillance du nœud feuille tel que le nombre de prédécesseur et/ou de successeur de la défaillance du nœud est supérieur à zéro
3. Le cas d'un nœud défaillant, dans lequel le prédécesseur est la racine.

La réparation locale sera effectuée par l'exécution de l'algorithme *MNLR_RPL* de reconfiguration intégrant le contrôle des capteurs mobiles, tout en utilisant la mobilité des nœuds parents par une procédure de choix basée sur trois règles définies par la suite.

Nœuds	**Définition**
S	Nœud défaillant ou en panne
Pred (S)	Prédécesseur du nœud ***S***
Succ(S)	Successeur du nœud ***S***
L_{Pred}	Liste des prédécesseurs
L_{Succ}	Liste des Successeurs
M	Nœud Mobile

Tableau 3.3 : Les identifiants de MNLR_RPL.

3.4.1.2 Description des règles : Hypothèses.

L'algorithme *MNLR_RPL* intègre trois nouvelles règles qui seront appliquées sur une topologie du réseau. En premier temps pour détecter le nœud défaillant (***S***), pour les deux cas, dans le réseau, nœud feuille, nœud parent. Dans la deuxième étape, la solution est de remplacer le nœud (***S***) par la mobilité des nœuds de ***Pred (S)*** ou ***Succ(S)*** choisis à partir des listes L_{Succ} et L_{Pred}**.** Ceci est prédéterminé par les *règles 1, 2 et 3* définies ci-dessous :

Dans chaque règle du tableau 3.4, on fait appel à la *procédure 1* comme présentée dans le tableau 3.5. Cette procédure est un processus conditionnel et important pour la réussite de la sélection du nœud mobile parmi la liste qui contient les parents de l'arbre topologique du réseau *DODAG*, pour le remplacement du nœud défaillant. La procédure *1* nous permet de choisir et

sélectionner les nœuds qui peuvent vérifier les critères ou les paramètres de la fonction *f*, dont *f = p1, p2, p3, p4,... pi*. Les paramètres de cette fonction reflètent les caractéristiques conceptuelles des nœuds capteurs dans un *RCSF*, ou *p1* = énergie, *p2* = nombre successeurs, *p3* = nombre de prédécesseurs, *p4* = distance entre les nœuds voisins, etc. Ces paramètres seront choisis pour maximiser la fonction *f*, si par exemple un nœud, pour qu'il soit sélectionné parmi la listes des nœuds parents à être mobile pour remplacer un nœud défaillant et vérifier les paramètre *p1* et *p2 ;* alors la fonction *f (p1, p2)* doit être maximisé par *p1* et *p2*.

Règles	***Description d'étapes***
• ***Règle 1***	Cas d'un nœud ***feuille*** en panne.
	1. Choisir un nœud de la liste L_{Pred} des ***Pred(S),*** appel de la ***procédure** 1*, 2. Soit ***M*** le nœud prédécesseur choisi, 3. Supprimer ***S***.
Règle N° 2 :	Cas d'un nœud en panne autre que le nœud ***feuille***, tel que : ***Pred(S)* > 0** et/ou ***Succ(S)* > 0**.
	1. Soit ***M*** le nœud ***prédécesseur*** choisi de la liste $L_{Pred(S)}$, appel de la ***procédure 1***, 2. ***Succ(S)*** deviennent les ***successeurs*** de ***M,*** ***3.*** ***Pred(S)*** deviennent ***prédécesseurs*** de ***M,*** **4.** Supprimer ***S.***
Règle N° 3 :	Cas d'un nœud en pane dont son ***prédécesseur*** est le nœud racine "***Root***".
	1. Soit M le nœud successeur choisi de la liste ***L*** des successeurs ***Succ(S)*** appliquant la ***procédure 1***, ***2.*** ***Succ(S)*** deviennent ***successeurs*** de ***M***, 3. ***Successeur*** du "***Root***" devient ***M,*** ***4.*** Supprimer ***S***.

Tableau 3.4 : Description des étapes des règles de *MNLR_RPL*.

Procédure 1
Choisir ***M***, un nœud de la liste ***L*** tel que ce nœud maximise la fonction ***f***. ***f*** = ***p1***, ***p2***, ***p3***, ***p4***,..... ***pi***, des paramètres tel que : ***p1*** = énergie, ***p2*** = nombre successeurs, ***p3*** = nombre de prédécesseurs, ***p4 =*** distance entre les nœuds voisins....ect. ***f (p1, p2)*** est la fonction dont le nœud choisi nœud***(n)*** a l'énergie la plus élevée, et le minimum de nombre de successeurs.

Tableau 3.5 : la procédure 1 de *MNLR_RPL*.

3.5 Implémentation et évaluation des performances

Cette section décrit l'environnement de simulation utilisé pour la mise en œuvre de la méthode de réparation local avec nœuds mobiles proposé *(MNLR_RPL*). Nous présentons par la suite les paramètres de simulation ainsi qu'une analyse comparative des différents résultats obtenus.

3.5.1 La plateforme Contiki/Cooja

L'évaluation des performances de notre proposition *MNLR_RPL* est réalisée à l'aide du simulateur *Cooja* sous le système d'exploitation Contiki 2.6. Nous présentons brièvement dans les sous-sections suivantes le principe d'exécution d'un modèle de simulation sous *Cooja* [155] sou *Contiki OS* (voir annexe A).

3.5.1.1 Système d'exploitation Contiki

Contiki est un système d'exploitation léger et flexible développé pour les systèmes embarqués fortement contraints en mémoire de ciblage des *RCSF* basés sur la technologie *6LoWPAN*. Ses principaux atouts sont le support des protocoles *IPv6* et *6LoWPAN*, sa flexibilité et sa portabilité.

Contiki comprend un micro mise en œuvre relativement mature *IP* (*uIP*), ainsi qu'une mise en œuvre *6LoWPAN* (*SICSlowpan*). Contrairement à *TinyOS*, qui développe son propre langage, *Contiki* utilise le langage de programmation C. La pile réseau fournit un support pour la mise en réseau *non-IP* via la pile *Rime*, et de mise en réseau *IP* en utilisant soit *uIPv4* ou *uIPv6* sur *6LoWPAN*.

Les protocoles de la couche supérieure dans *Contiki* comprennent *UDP* et *CoAP*. L'empilement des couches basses contiennent de nombreuses implémentations *MAC* et *RDC*. Ils sont très peu les systèmes libres qui mettent en œuvre *6LoWPAN* technologie. Les plus importants sont *TinyOS* [34] et *Contiki* [35]. *Contiki* supporte le chargement dynamique de code.

- **La communication dans *Contiki 2.6***

Contiki est un système événementiel dans lequel les processus sont mis en œuvre en tant que gestionnaires d'événements qui vont jusqu'à l'achèvement. Le système d'exploitation *Contiki* fournit des modules pour différentes tâches. Il fournit les modules de routage dans un répertoire distinct "*contiki/core/net / RPL*" [146] et se compose d'un certain nombre de fichiers. Ces fichiers sont séparés logiquement sur la base des fonctionnalités qu'ils offrent, par exemple, *RPL-dag.c* contient la fonctionnalité de graphe acyclique (*DAG*) formation dirigée, *RPL-icmp6.c* fournit des fonctionnalités pour l'emballage de messages *ICMP*, etc.

Un système *Contiki* [156] [35] est divisé en deux parties : le noyau et les programmes chargés. Le noyau se compose du noyau *Contiki*, le programme chargeur, la langue d'exécution, et une pile de communication avec les pilotes de périphériques pour le matériel de communication comme illustré la figure 3.10.

Dans notre simulation, la plupart des scenarios sont liés à ces fichiers du système d'exploitation *Contiki*.

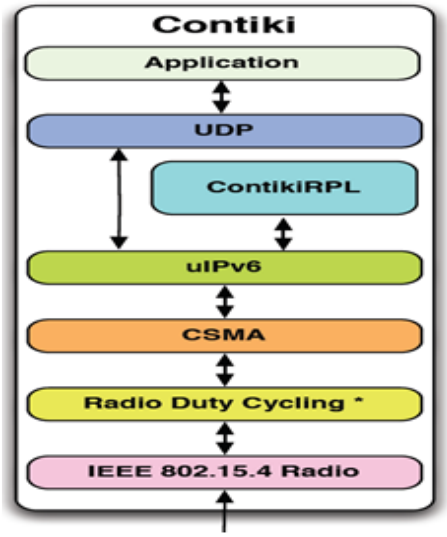


Figure 3.10 : Pile de Communication dans *Contiki* [156].

3.5.1.2 Le simulateur Cooja

Pour permettre le déroulement et l'évaluation sous plusieurs scénarii de notre méthode *MNLR_RPL*, nous avons utilisé simultanément le simulateur inclut dans le système *Contiki* appelé *Cooja* avec le modèle *UDGM*, avec les caractéristiques suivantes [155] :

- *COOJA* combine des simulations de capteur matériel de nœud et simulation du comportement de haut niveau en une seule simulation.
- *COOJA* est flexible et extensible, ainsi tous les niveaux du système peuvent être modifiés ou remplacés.
- *COOJA* est une application Java, toutes les interactions avec *Code Contiki* se fait à travers Java Native Interface (*JNI*).
- *COOJA* est un simulateur à base de Java extensible capable d'émuler *Tmote Sky* et d'autres nœuds.
- *COOJA* offre divers outils pour déboguer et d'analyser un réseau via les deux interfaces de ligne graphique et commande.

L'interface du simulateur *Cooja* comme illustré par la figure 3.11, est composée de plusieurs fenêtres (plugins), (une description brève est présentée dans l'annexe A) et fournies par *Contiki* avec le modèle *UDGM*. Cette approche fournit une méthode pour quantifier l'impact de l'attaque d'incohérence *DAG* dans des conditions où le canal *IEEE 802.15.4* avec perte de données n'entraîne pas de perte de paquets. Cela permet une évaluation dans des conditions idéales, sans caractéristiques externes, qui provoquent un biais dans les résultats.

- ***Mote output :*** présente ce que les capteurs génèrent comme sortie via leurs ports séries.
- ***Network*** : présente le réseau simulé et le flux de communication durant la simulation.
- ***Simulation control :*** cette fenêtre contient quatre boutons :
  1) *Start* : pour démarrer une simulation.
  2) *Pause* : pour arrêter la simulation.
  3) *Reload* : pour recharger une simulation.
  4) *Step* : pour régler la vitesse de la simulation.
- ***Power tracker :*** pour voir la consommation d'énergie par les capteurs durant la simulation.
- ***Mote output*** : présente ce que les capteurs génèrent comme sortie via leurs ports séries.
- ***Network*** : présente le réseau simulé et le flux de communication durant la simulation.

- ***Simulation control*** : cette fenêtre contient quatre boutons :
    5) Start : pour démarrer une simulation.
    6) Pause : pour arrêter la simulation.
    7) Reload : pour recharger une simulation.
    8) Step : pour régler la vitesse de la simulation.
- ***Power tracker*** : pour voir la consommation d'énergie par les capteurs durant la simulation.

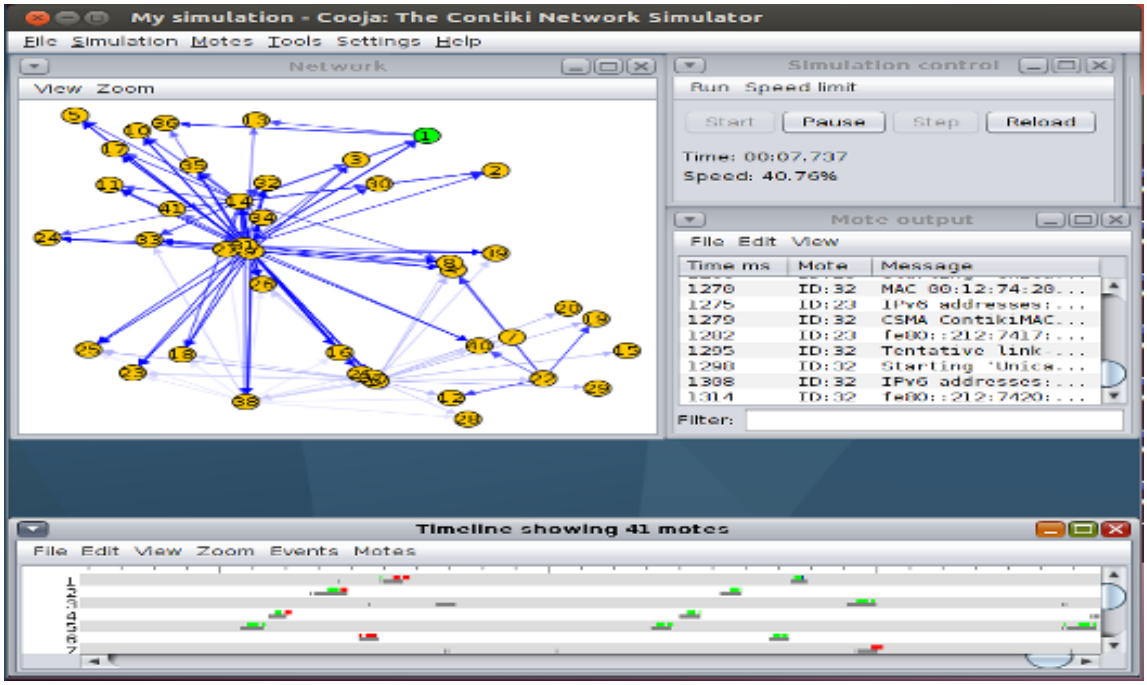


Figure 3. 11 : Interface du simulateur *Cooja* avec le protocole *RPL*.

3.5.2 Le modèle de simulation de *MNLR_RPL*

L'implémentation du modèle de contrôle et de gestion des nœuds défaillants dans un *RCSF* basé sur la technologie *6LoWPAN* avec le protocole *RPL*, proposée dans ce chapitre, est réalisée à l'aide des modules de *RPL* développés par l'équipe *ROLL* [6].

Nous présentons dans ce qui suit les principaux scripts et fichiers du code source du protocole *RPL* écrit en *C*, que nous avons traités afin d'obtenir des extensions nécessaires pour la réalisation du nouveau modèle de réparation locale des nœuds défaillants exécuté par la méthode *MNLR_RPL* [9]. Les autres fichiers utilisés sont présentés dans l'annexe A. Dans la partie 1, une description détaillée du le code source du protocole *RPL* sous *Cooja*, dans la partie 2, on présente la simulation du modèle de réparation locale avec l'algorithme *MNLR_RPL* et les différentes modifications effectuées sur les fichiers du code source du protocole *RPL*.

3.5.2.1 Le protocole *RPL* avec le simulateur *COOJA*

Récemment, plusieurs recherches ont été proposées afin de faire face à différents problèmes de RPL, à savoir la mobilité des nœuds, la boucle de routage, la sécurité, etc. Dans le but d'évaluer les performances de la *RPL*, ces recherches sont basées sur des technologies de simulateurs et de *frameworks* différents tels que le *framework RLP ContikRPL* sous Contiki Operating System utilisant l'outil discret Cooja [155], *tinyRPL* sous *TinyOS*, et *NanoQplus* multithread Système. Dans cette thèse, nous avons fait appel à *ContikiRPL* [156], car considéré comme l'implémentation la plus robuste et la plus efficace pour les *WSN* basés sur la technologie *6LoWPAN*.

ContikiRPL (selon la figure 3.10) implante le protocole de routage *RPL* destiné pour les *RCSF* avec deux fonctions objectif *OF0* et *MRHOF25*. *ContikiRPL* a été testé sur plusieurs plateformes. Il est ainsi exécutable avec le simulateur *COOJA* avec une liaison *IEEE 802.15.4* pour simuler les réseaux de capteurs à faible puissance. *RPL* gère l'ensemble des informations générales d'un *DODAG*, par exemple des informations sur les parents, la connectivité, l'ajout ou suppression des routes, etc. La figure 3.12 présente les différentes classes du *RPL* implémentées dans *Contiki* :

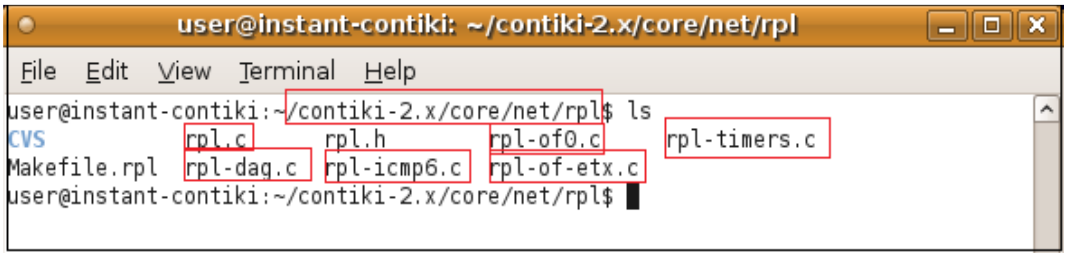


Figure 3.12 : Les différentes classes du module *RPL* implémentées dans Contiki.

- ***rpl-dag*** : détermine les différents événements et actions pour joindre un *DAG*, sélectionner un parent, faire des calculs du rang ou des réparations globales en cas de perte de connectivité.
- ***rpl-icmp6* :** permet de faire la mise en œuvre des messages d'entrée/sortie du contrôle RPL avec le format *ICMPv6*.
- ***rpl-Of0* :** détermine la fonction objectif par défaut de *RPL* qui permet uniquement d'optimiser le nombre de sauts par chemin .
- ***rpl-of-etx* :** permet de déterminer la métrique *ETX* utile pour le calcul du rang pour la fonction objectif *MRHOF* et pour la sélection du parent pour *OF0*.

- ***rpl-timers* :** présente les différentes fonctions et méthodes indispensables pour le traitement de l'algorithme *Trickle*.

Dans le script principal *rpl_script.csc* de la simulation, nous pouvons définir la configuration du réseau en topologie contenant les paramètres de construction tels que : le nombre de nœud, le type de transmission, la portée radio de transmission et de réception, …etc.

3.5.3 Simulation de scénarios et évaluation de performance

Dans cette section nous nous concentrons sur la simulation de scénarios de *MNLR_RPL* [9] lors de l'utilisation d'un nœud mobile pour remplacer le nœud défaillant, afin d'évaluer l'impact de la nouvelle technique de réparation locale sur le fonctionnement du protocole *RPL* destiné à la gestion de la topologie du réseau. On notera également le type de défaillance de nœud utilisé dans nos simulations et qui est proposé par l'utilisation du *UDGM* (*Unit Disk Graph Medium*) dans *Cooja*.

3.5.3.1 Configuration de la topologie du réseau

Nous avons utilisé un exemple de topologie de réseau dans le simulateur *Cooja* contenant 100 nœuds et 1 nœud agissant comme racine du *DODAG* (*DAG Root*). Le scénario de la topologie du réseau est représenté sur la figure 3.13. Le nœud *Root* utilise un exemple d'application *udp-server*.c alors que tous les autres nœuds utilisent *udp-client.c*. Nous utilisons un plugin *Cooja* appelé *Contiki* Test Editeur pour mesurer le temps de simulation et d'arrêter la simulation après l'heure indiquée. Ce plugin crée également un fichier journal (*COOJA.testlog*) pour toutes les sorties de la simulation que nous analyserons à l'aide d'un script *AWK.*

3.5.3.2 Définition des scenarios

Le comportement du standard *RPL* avec la méthode *MNLR_RPL* effectué par des nœuds mobiles a été validé par simulation extensive en considérant deux scenarios :

Le premier scenario prend en charge les simulations avec exécution simultanée des trois règles (voir tableau 3.4 et 3.5), tandis que le second est réalisé par le déclenchement de chaque règle séparément, chacune de ces règles correspond à chaque cas de nœuds défaillants dans le réseau.

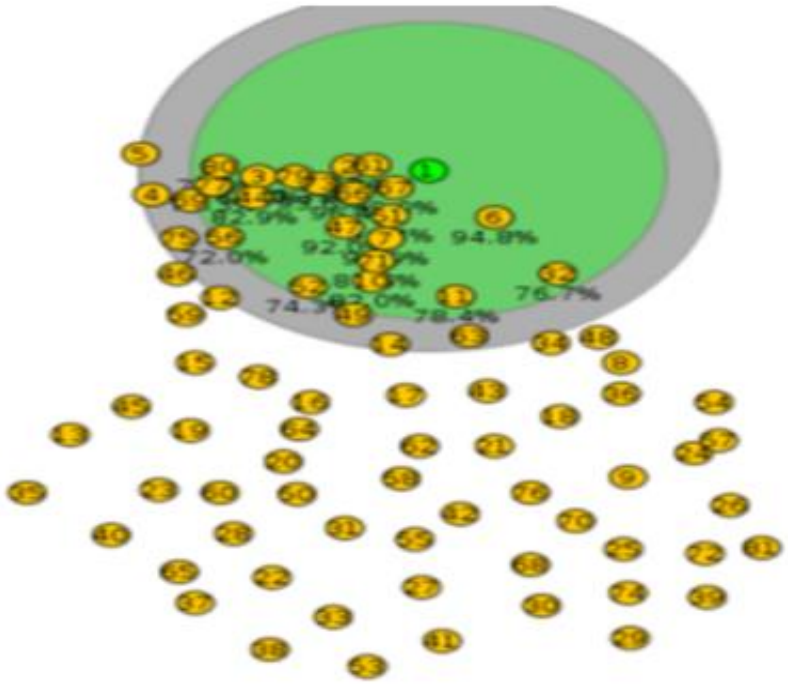


Figure. 3.13 Scénarios simulés.

3.5.3.3 Paramètres de simulation

Les paramètres de simulation sont donnés dans le tableau 3.6, en outre, une description détaillée sera présentée dans cette section.

Paramètres	**Valeurs**
Zone de surveillance	200 * 200 m²
Environnement de la Radio	*UDGM* (Unit Disk Graph Medium)
Temps de Simulation	60 Mns
Nœuds Clients	100 et 1 *Root*
Type de Mote	*Tmote Sky*
Nombre de nœuds	25 -100
Duty Cycle	*ContikiMAC*
Couches: *PYH* et *MAC*, réseau	*IEEE 802.15.4*, *uIPv6*
Objective Function	*ETX*
RX Ratio	30, 40, 50, 60, 70, 80, 90, 100
Max Paquets	32583
DIO Min	2,3,4,5,6,7,8,9,10,11,12,13,14, 15,16

Tableau 3.6 : Paramètres de simulation dans le cadre de l'évaluation de *MNLR_RPL*.

Le réseau que nous avons simulé contient 100 nœuds et un seul nœud récepteur, déployés dans un environnement de 200 m x 200 m.

La transmission des paquets commence après le délai de démarrage (65s) de sorte que le temps réel de simulation sera soustrait du délai de démarrage (Temps Simulation - Start Delay).

Nous avons choisi le mode de fonctionnement de *RPL* du sens *UP* (des nœuds feuilles vers la racine en passant par plusieurs nœuds intermédiaires), parce que nous avons jugé préférable l'utilisation du trafic multipoint à point (*MP* à *P*) pour nos évaluations et limiter ainsi le cadre de cette étude.

Le *DIO-Min* et le *DIO- Doublings* sont définis par *ContikiRPL* comme des valeurs par défaut. Ainsi que le rapport de réception (*RX*) représente la moyenne de perte de la radio et est fixé en pourcentage au cours des itérations successives de la simulation.

Le rapport de transmission (*TX*) est fixé à 100% parce que nous ne voulons pas introduire des pertes à l'émission, mais seulement à la réception. La portée de radio *TX* est réglée à 50m et la plage d'interférence à 55m. Nous mettons en œuvre la simulation pendant 1 heure (3600s), pour 25 et 100 nœuds dans la simulation.

Le paramètre *ETX* représentant la fonction objectif fut utilisé dans nos simulations. *ETX* peut très bien être remplacé par les autres paramètres tels que le nombre de sauts, de latence, débit, etc. Néanmoins, *ETX* définit une transmission réussie, couronnée par la réception d'un accusé de réception en réponse à une transmission unicast.

3.5.3.4 Métriques de performances

Pour l'évaluation de la méthode *MNLR_RPL*, nous avons utilisé les métriques de performance suivantes :

- **Latence de réseau : C**ette mesure de performance est définie comme la latence moyenne de tous les paquets utilisés par tous les nœuds dans le réseau, elle est calculée par la formule 3.2. La latence totale des paquets, est la quantité de temps prise par un paquet du nœud pour atteindre le *Root*, elle est calculée selon la formule 3.1.

$$Latence\ totale = \sum_{n=1}^{m}(Temps_{Reçu}(n) - Temps_{transmit}\ (n)) \qquad (3.1)$$

$$Latence\ Moyenne(Latence\ totale|Total\ des\ Paquets\ Reçus) \qquad (3.2)$$

- **Pourcentage moyen de délivrance de paquets** ($PDR\%$) : Il mesure le rapport entre le nombre total de paquets reçus et le nombre total de paquets envoyés dans la sous-couche *MAC*, le *PDR* (*Packet Delivery Ratio*), étant calculé par la formule 3.3.

$$PDR = (Total\ des\ paquets\ reçus | Total\ des\ paquets\ transmis) * 10 \quad (3.3)$$

- **Consommation d'énergie :** Il mesure l'énergie dissipée par les nœuds afin de transmettre un paquet à partir du nœud *SB* (Station de base ou *Sink*).
- **Trafics de contrôle *ICMPV6* :** Cela inclut les messages *ICMPV6 DIO*, *DIS* et les messages *DAO* générés par chaque nœud et il s'avère impératif de limiter le contrôle de la circulation en tenant compte des ressources limitées en réseau *LLN*.
- **Trafics de contrôle *ICMPV6* :** Cela inclut les messages *ICMPV6 DIO*, *DIS* et messages *DAO* générés par chaque nœud, par ailleurs, et il est impératif de limiter le contrôle de la circulation en tenant compte des ressources limitées en réseau *LLN*. Cette métrique est calculée selon la formule 3.4.

$$Message\ de\ controle = \sum_{n=1}^{k}(DIO + DIS + DAO) \quad (3.4)$$

- **CollectView :** cette métrique représente la moyenne des paquets reçus par le *DODAG Root* à partir du nœud transmetteur.

3.5.4 Evaluation de performance

Nous présentons dans ce qui suit les résultats de performances obtenus à l'aide du simulateur *COOJA*. Les résultats sont présentés en deux scenarios :

3.5.4.1 Résultats de simulation du premier scenario : analyse et interprétation

Nous présentons dans cette sous-section les résultats du premier scenario qui consiste à exécuter les trois règles de *MNLR_RPL* en même temps. Ces résultats sont présentés selon l'impact de deux facteurs de simulation : A) aspect de l'impact de la *DIO Intervalle Min* et B) l'impact du pourcentage de réception de paquets (*RX*).

A) L'impact de la DIO Intervalle Min avec MNLR_RPL dans un DODAG

Les premiers résultats d'évaluation de performance de *MNLR_RPL* sont obtenus lors des différentes itérations de simulation sous l'impact de la variation de (*IMDIO*) l'intervalle minimum de *DIO* (*RPL_DIO_INTERVAL_MIN*) par {2, 3, 4, 16}, tout en fixant *RX* à 70%. Cela est entrepris pour étudier et analyser les métriques : le nombre de trafics de contrôle, *PDR*, la latence du réseau et l'énergie consommée. Les résultats de cette expérimentation sont illustrés par les figures 3.14, 3.15, 3.16 et 3.17 selon trois états du réseau : 1) état normal (*Network-*

NormalState), 2) état de défaillance de nœuds (*Network-NodeFailure*) et 3) état avec méthode (*Network-MNLR_RPL*).

- **Message de contrôle *ICMPV6***

Comme nous pouvons le constater sur la figure 3.14, le fait d'augmenter *IMDIO* (l'intervalle Minimum de *DIO*) du réseau (plus de 9) permet d'augmenter les trafics de contrôle dans le réseau. En effet, la valeur du trafic de contrôle est très élevée pour *IMDIO,* allant de 2 à 8 et diminue rapidement en présence de 9 à 16, qui est l'ensemble optimal pour le contrôle du trafic dans le protocole *RPL* état normal. De plus, les trafics de contrôle pour l'état normal et le *MNLR_RPL* sont remarquables pour l'approximation de valeurs pendant la construction du *DAG* et la transmission de données.

Cela s'avère nettement différent avec des valeurs qui sont en augmentation dans l'état de panne des nœuds. Cela est aussi lié à la défaillance et à l'absence de nœuds dans le réseau, où les paquets de contrôle de retransmission *ICMPv6* (messages *DIO*) par chaque nœud sont augmentés, de sorte que le nœud répond, en vue rejoindre le *DAG,* en établissant de telles transmissions.

Ces résultats de comparaison montrent l'efficacité de notre méthode *MNLR_RPL* pour la réparation locale en *RCSF*.

- **Pourcentage moyen de délivrance de paquets**

Nous avons également examiné l'effet de la variation de l'*IMDIO* sur le taux de délivrance de paquets (*PDR*). A partir des résultats de simulation de la figure 3.15, nous pouvons constater que dans l'état de défaillance dans le réseau (*NetwotkNodeFailure*) par nœuds défaillants, la dégradation de la performance du réseau est remarquable pour les valeurs de l'*IMDIO* comprises entre 2 et 6 et entre 13 et 16.

En effet, le *PDR* est inférieur à 75%, ce qui paraît évident car ce pourcentage nous permettra de justifier la présence de plusieurs nœuds défaillants, simultanément dans les trois cas de défaillance définis dans la section 3.4.1, cela se justifie aussi de par les simulations parallèles des trois règles. D'autre part, le *MNLR_RPL* fournit un bon taux de livraison de paquets de plus de 95%, de par le remplacement des nœuds défaillants par les nœuds mobiles.

Cependant, les résultats de simulation montrent que les performances sont meilleures en utilisant *MNLR_RPL* pour assurer le contrôle de la continuité de fonctionnement du réseau.

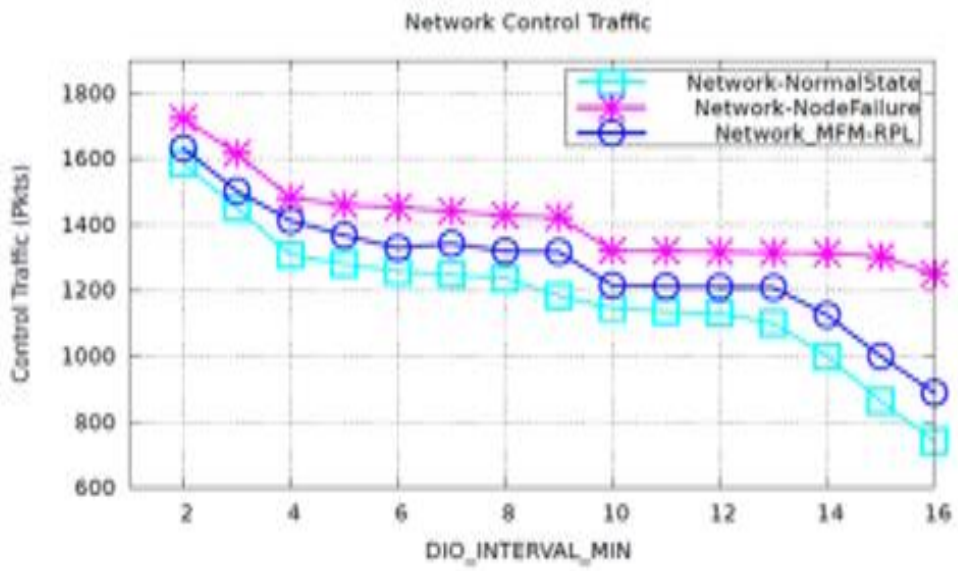


Figure 3.14: L'impact de *IMDIO* dans la topologie de réseau à états différents (Normal, défaillance et *MNLR_RPL*) sur le trafic de Contrôle.

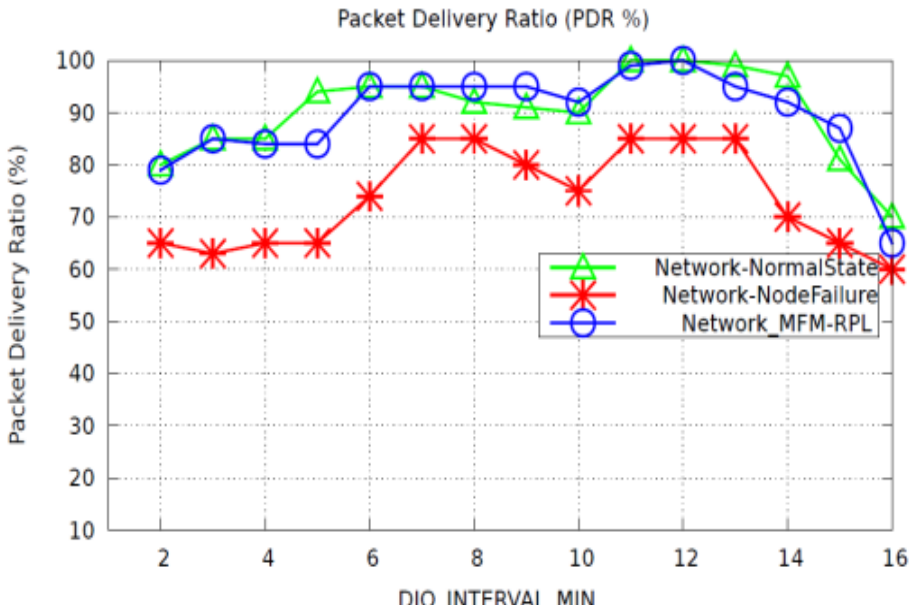


Figure 3.15: L'impact de *IMDIO* dans la topologie de réseau à états différents (Normal, défaillance et *MNLR_RPL*) sur *PDR*.

- **La latence du réseau**

A partir des résultats illustrés sur la figure 3.16, nous pouvons constater que la valeur de latence obtenue en utilisant *MNLR_RPL* est meilleure par rapport à celle obtenue dans l'état de défaillance (voire dans l'état normal). Ce qui est vérifié dans les figure précédente 3.14 et 3.15, par l'obtention du meilleur taux de délivrance qui est égal à 95%, à travers l'utilisation de la

méthode *MNLR_RPL* et le nombre minimum de trafics de contrôle obtenus par rapport à l'état de défaillance des nœuds dans le réseau.

Cela est provoqué par l'effet de la défaillance des nœuds et le trafic de contrôle augmenté dans le réseau. Cette méthode nous permet de garantir le fonctionnement du réseau après une défaillance de nœuds.

- **Consommation d'énergie**

Comme nous pouvons le constater sur la figure 3.17, le réseau consomme plus d'énergie du fait que le paramètre de performance *IMDIO* a des faibles valeurs, égales à 2 et 3. Etant donné que nous avons une meilleure communication entre les nœuds du réseau par la réparation des nœuds défaillants à travers l'application de l'algorithme *MNLR_RPL*; nous pouvons déduire que l'utilisation des nœuds mobiles, pouvant être déplacés vers le nœud défaillant pour le remplacer, a un impact non négligeable sur le niveau d'énergie des nœuds capteurs.

La performance du modèle proposé s'avère plus significative par l'utilisation de la méthode de réparation locale, car nous obtenons un pourcentage réduit de nœuds défaillants du moment que nous aurons généré plus les nœuds mobiles pour les remplacer. Ces résultats sont intéressants puisque nous aurons d'autres nœuds disponibles qui pourront assurer la couverture des capteurs défectueux.

MNLR_RPL permet de préserver les performances de routage tout en exploitant le fait de la redondance pour atteindre la gestion de la défaillance des nœuds dans le réseau, de par le fait que RPL crée une liste des parents dont l'un est le préféré, cela donne la possibilité d'exploiter les autres parents et de choisir le nœud qui devient mobile pour remplacer le nœud défaillant.

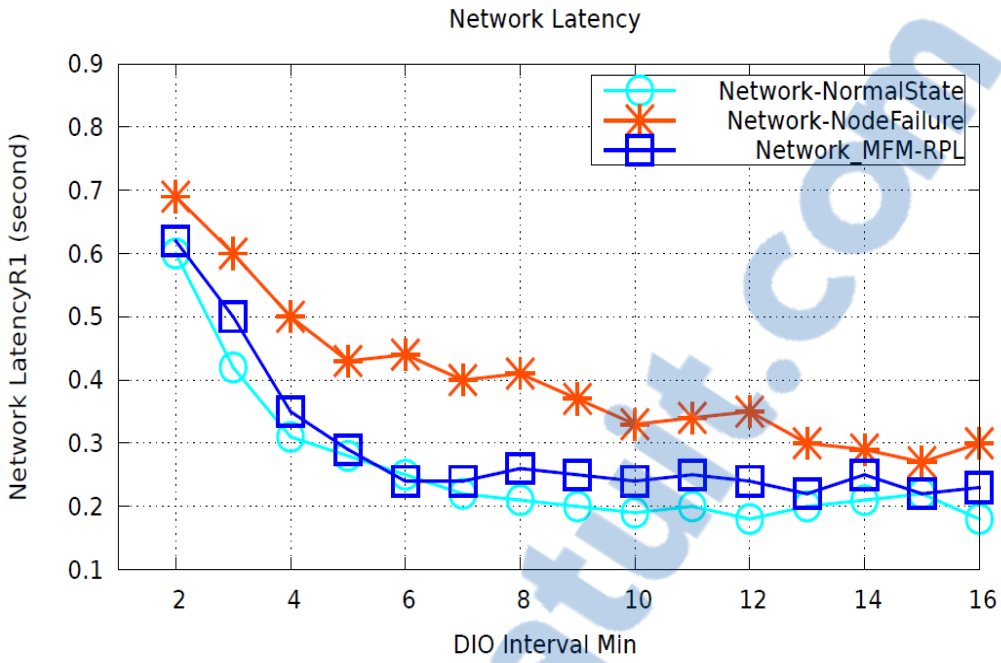


Figure 3.16 : L'impact de *IMDIO* dans la topologie de réseau à états différents (Normal, défaillance et *MNLR_RPL*) sur la latence du réseau (seconde).

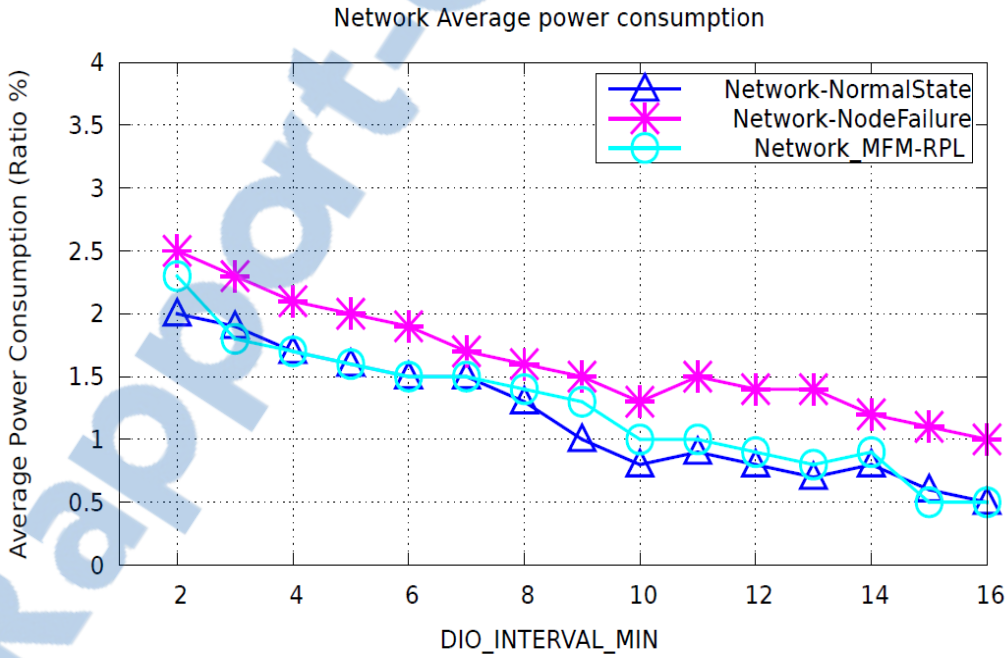


Figure 3.17 : L'impact de *IMDIO* dans la topologie de réseau à états différents (Normal, défaillance et *MNLR_RPL*) sur la Consommation d'énergie.

B) L'impact du pourcentage de réception de paquets (*RX*) avec *MNLR_RPL* dans un *DODAG*

Dans cette deuxième partie d'évaluation de performance de *MNLR_RPL* du scenario 1, l'environnement d'exécution des simulations est similaire à celui que nous avons présenté précédemment dans la sous-section 3.4.1. Cependant, dans cette deuxième catégorie de scénario de simulation, nous avons fixé la valeur de *IMDIO* à 12 comme valeur par défaut du protocole *RPL* et nous avons fait varier la valeur de *RX* (*Packet Reception Ratio*) de 30, 40, 50, 60, 80, 90 et 100 (%). Nous avons utilisé la fonction objectif *ETX* pour calculer les meilleurs chemins. Les résultats de simulation sont présentés dans les figures 3.18 et 3.19.

- **La latence du réseau**

A partir des résultats de simulation illustrés sur la figure 3.18, nous pouvons observer qu'avec l'augmentation de RX, nous pouvons remarquer que la latence du réseau est meilleure et même significative dans l'état d'exécution de la méthode *MNLR_RPL* par rapport à celle de l'état normal ou celle en défaillance.

En effet, le *RPL* offre une fonction objective avec *ETX* qui offre la possibilité de calculer les meilleurs chemins, ainsi *ETX* est calculé par saut et utilisé pour sélectionner le parent préféré. De plus, lorsque RX, qui représente le ratio de réception de paquets, a une valeur élevée dans le réseau, ceci donne la possibilité d'accuser une bonne réception des données dans un délai minimum avec *ETX*. A cet effet, la latence diminue, cela signifie que le *MNLR_RPL* nous offre une autre caractéristique positive, celle de trouver de meilleurs chemins avec la fonction objectif *ETX* dans des délais minimum tel que défini dans la norme.

- **Consommation d'énergie**

Dans les résultats de simulation de la figure 3.19 avec l'augmentation du rapport *RX* dans le réseau, la consommation d'énergie diminue lentement pour les différents états du réseau. En effet, les performances de *MNLR_RPL* prouvent l'efficacité de la méthode. Et ce, à cause du délai minimum de réacheminement des données jusqu'au nœud racine, ce qui est montré par la figure 3.18. Pour les valeurs assez importantes de la consommation d'énergie dans la figure 3.19, nous dirons que cela est produit par les retransmissions des paquets sans perte de données avec *MNLR_RPL*. Toutefois, nous pouvons remarquer que la méthode proposée *MNLR_RPL* permet de respecter la contrainte de l'énergie consommée avec les nœuds mobiles exploités pour la réparation locale de la défaillance dans le réseau, et réduire par voie de conséquence la latence du réseau.

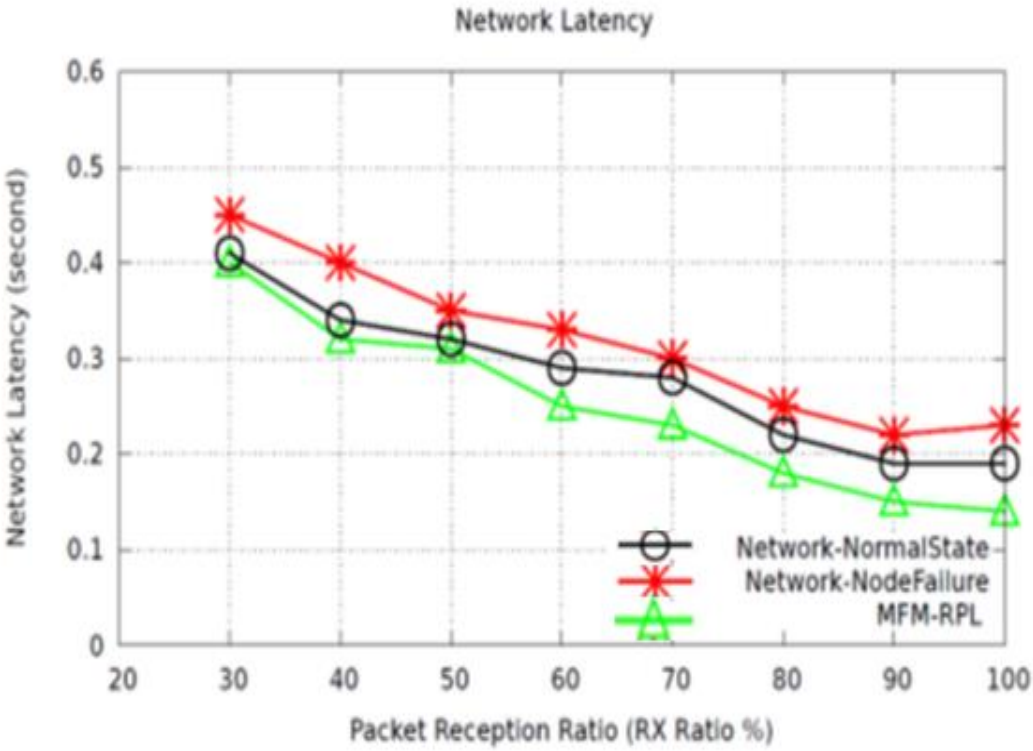


Figure 3.18 : L'impact de RX dans la topologie de réseau à états différents (Normal, défaillance et *MNLR_RPL*) sur la latence du réseau (seconde).

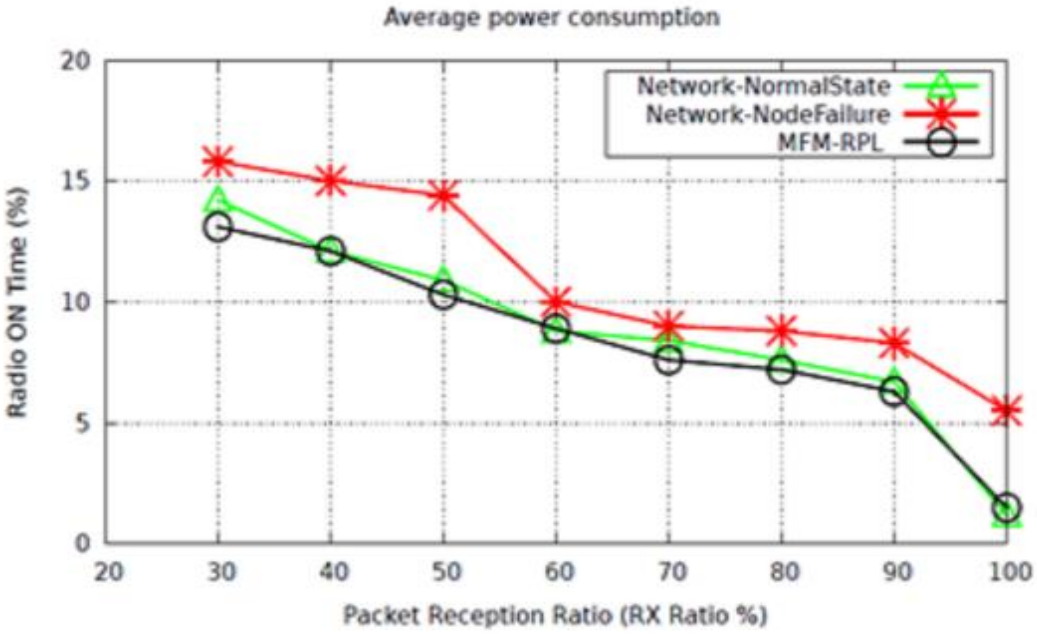


Figure 3.19 : L'impact de RX dans la topologie de réseau à états différents (Normal, défaillance et *MNLR_RPL*) sur la Consommation d'énergie.

3.5.4.2 Résultats de simulation du deuxième scenario : analyse et interprétation

Pour cette phase de simulation, la méthode *MNLR_RPL* est simulée avec des règles séparées (*regle1*, *regle2* et *règle3*). L'objectif de ce scénario est de présenter l'influence de la position du nœud défaillant dans le réseau selon les cas présentés dans la sous-section 3.4.1.2, page 93.

Pour ces raisons, nous augmentons le taux de réception de paquets RX de 30, 40, 50, 60, 70, 80, 90 et100. Ensuite, nous comparons les paquets de *collectview* moyens reçus par le *DAGroot* et la consommation moyenne d'énergie à chaque exécution de règle dans la topologie du réseau. Les résultats de la simulation sont représentés dans les figures 3.20, 3.21.

- **CollectView**

A partir des résultats de simulation illustrés dans la figure 3.20, nous pouvons également observer l'effet de la position du nœud défaillant dans le réseau. Pour l'état du réseau dans le cas 1 (nœud feuille défaillant), la *collectview* moyenne est de 20% comparé à 40% après exécution de la règle 1 pour la réparation et le remplacement du nœud feuille défaillant. En outre, en observant la règle 2 la valeur sera de l'ordre de 73% par rapport à l'état du réseau avec nœud défaillant pour le cas 2, qui signale une perte de données de l'ordre de 20%.

La même observation va pour la règle 3, lorsque la moyenne du *collectview* présente un bon résultat capable de recueillir 100% comparé à l'état du réseau dans le cas 3 qui est inférieur à 10% de perte de données.

On constate que lorsque les règles sont exécutées séparément dans le temps avec le cas de la position du nœud défaillant dans le réseau, la méthode présente des résultats satisfaisants pour la moyenne de *collectview* dans l'état de réparation locale du réseau. Cela montre bien l'influence de la localisation de la défaillance dans le réseau, si elle est dans la position des nœuds proches à la racine comme le cas 3, dans ce cas la réparation est très importante voire essentielle pour assurer le bon fonctionnement du réseau.

Cependant, la figure 3.21 montre bien l'efficacité de l'exécution des règles de *MNLR_RPL* séparément. En effet, les résultats de simulation de la *règle 3* concernent les nœuds les plus proches du *DODAGRoot* (*Sink*)). Pour la règle 2, la *collectview* moyenne est comprise entre 70% et 80%. Quant à règle 1, le point de vue de la collecte moyenne est plus de 40% et moins de 50%. On peut observer que la méthode proposé *MNLR_RPL* par les règles séparées effectue une récupération efficace et plus rapide.

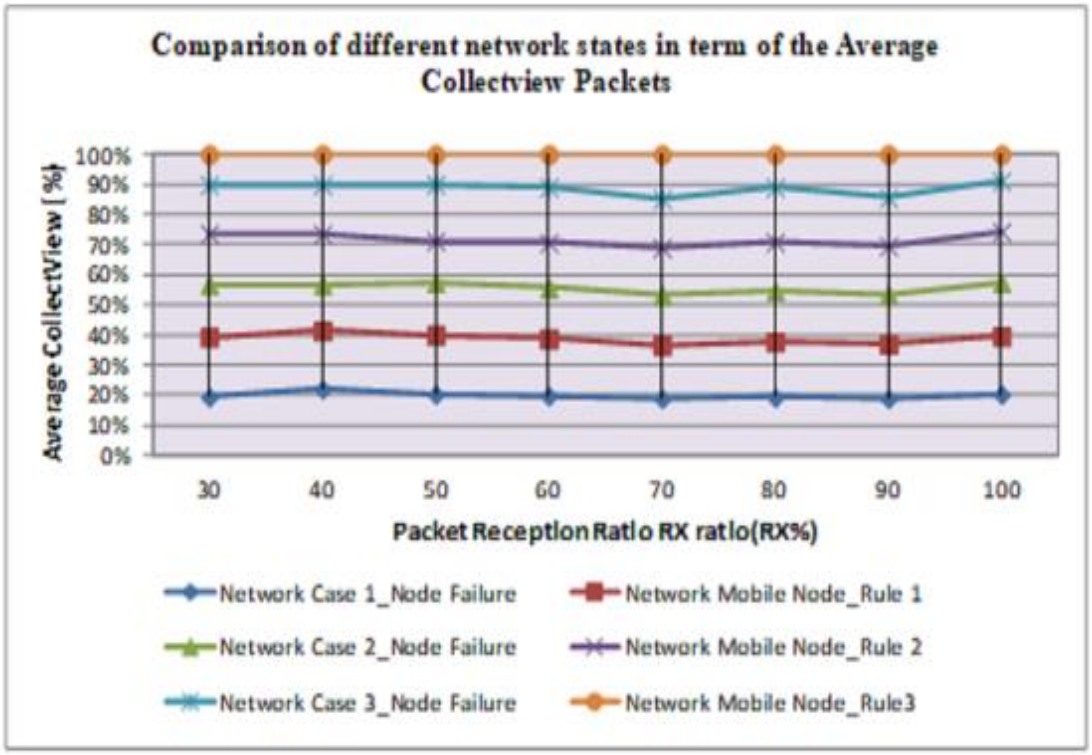


Figure 3.20: *CollectView* moyenne des paquets pour cas de nœuds défaillants avec les règles séparées.

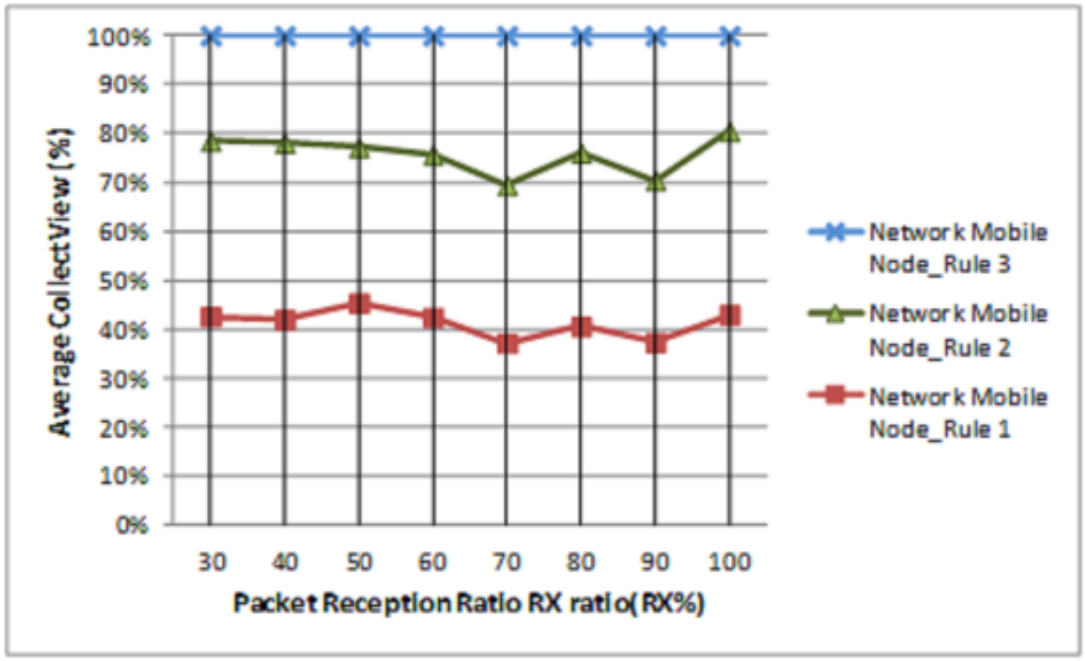


Figure 3.21 : *CollectView* moyenne des paquets pour la règle 1, regèle 2 et règle.

3.6 Conclusion

Dans ce chapitre nous avons proposé une solution au problème de nœuds défaillants dans un *RCSF* pour l'*IdO* avec le protocole *RPL*. Les défaillances des nœuds dans un réseau sont inévitables. Cependant, le réseau devrait pouvoir les détecter et réagir plus rapidement face à cette anomalie pour assurer la continuité du fonctionnement du réseau. D'un autre côté, avec RPL une défaillance de nœud de liaison ou de nœud voisin est détectée et le mécanisme de réparation est déclenché. La technique de réparation pour *RPL* est coûteuse en termes de consommation d'énergie, coût de trafic de contrôle supplémentaire de par la reconstruction dans un arbre forcé par des défaillances de nœuds car *RPL* est un protocole proactif, où les routes sont définies dans la phase de construction de topologie arborescente (*DODAG tree*). En outre, *RPL* a été conçu à l'origine pour les réseaux statiques, sans support pour la mobilité. Néanmoins, il s'agit d'un protocole réactif face aux cas d'incohérence de la mobilité, ce qui le rend inefficace et certains problèmes peuvent survenir en conséquence. Cependant, gérer le mécanisme de réparation de la *RPL* avec des nœuds mobiles fut un véritable défi.

Dans ce chapitre, nous avons fourni une étude approfondie et détaillée du protocole *RPL* et de son support pour la mobilité à travers les travaux effectués et réalisés récemment sur la norme *RPL* en mobilité. Ainsi, pour remplacer les défaillances de nœuds par la mobilité de leurs prédécesseurs sans reconstruction de l'arbre *RPL*, est une solution proposée pour estimer la continuité de service pour la collecte et la transmission de données dans l'environnement surveillé.

Dans cette première partie de contribution, nous avons proposé une nouvelle méthode qui s'est avérée efficace pour la gestion des défaillances avec le protocole *RPL* (*MNLR_RPL*) dans un *RCSF* basé sur la technologie *6LoWPAN*.

Enfin, l'efficacité de la méthode proposée a été validée en effectuant plusieurs scénarios de simulation exécutés à travers les métriques de réseau.

Proposition d'une méthode de contrôle utilisant le concept SDN dans IdO.

Chapitre 4

4.1 Introduction

Le nouveau concept de l'*IdO* est une concrétisation technique de l'informatique ubiquitaire, et qui nécessite un logiciel spécialisé avec fonctionnalité réseau et support *IP*, il combine les dimensions de l'ordre conceptuel et technique. D'un point de vue conceptuel, l'*IdO* caractérise les objets physiques connectés ayant leur propre identité numérique et capables de communiquer entre eux. Ce réseau crée une passerelle entre le monde physique et le monde virtuel. D'un point de vue technique, l'I*dO* consiste en une identification numérique directe et standardisée (adresse *IP*, protocole *SMTP*, *http*, etc.) d'un objet physique au moyen d'un système de communication sans fil qui peut être une puce, *RFID*, *Bluetooth*, ou *WiFi*. Ce concept est très prometteur, en effet, la technologie est intégrée naturellement aux objets du quotidien.

D'un autre coté, de nombreuses technologies de communication et de réseau sont proposées pour le contexte de l'*IdO*, sans prendre en charge la coexistence et l'interopérabilité, et dans lesquelles, on distingue les technologies sous licence et sans licence. Dans le cas des technologies sans licence et en fonction du contexte des applications, un nombre important de normes peuvent être utilisées, telles que : *IEEE802.11* (*WiFi*), *IEEE802.15.1* (*Bluetooth), IEEE803.15.3* (*UWB*), *IEEE802.15.4* (*Zigbee*, *6lowpan*), *IEEE802.15.6* (*WBAN*), etc. Dans le cas d'une communication omniprésente, les technologies sous licence sont bien adaptées, à savoir : LTE-A et la communication de type machine (*Machine-Type Communication MTC*). C'est pourquoi il est important de proposer une architecture hybride et programmable capable de prendre en compte les différentes caractéristiques technologiques, et d'en assurer la coexistence.

Dans cette deuxième partie de contribution de cette thèse, nous abordons l'hétérogénéité des technologies de réseau dans IdO, et proposons une nouvelle architecture hybride et programmable basée sur le paradigme des réseaux définis par logiciels (*Software Defined Network* (*SDN*)).

L'objectif principal de ce chapitre est non seulement de considérer l'hétérogénéité des technologies de réseau mais aussi de réduire la charge de trafic de contrôle liée aux mécanismes de contrôle du réseau et d'éviter la défaillance du seul point de contrôle au niveau du contrôleur *SDN* central.

Contrairement au *SDN* classique, nous proposons une approche semi-distribuée avec répartition du rôle des contrôleurs tout en introduisant trois niveaux de contrôle : Principaux, Secondaires et Locaux. En outre, nous introduisons un nouveau mécanisme de Contrôleurs Locaux (*CLs*) basé sur des ensembles connectés *CDS* (*Connected Dominating Sets*) et des approches de la logique *floue*.

A la fin de ce chapitre, nous proposons un mécanisme de sélection des *CL* représenté par un algorithme distribué de construction et de sélection d'un ensemble de dominant qui va jouer le rôle des contrôleurs locaux connectés dans l'architecture proposée. Cet algorithme est nommé *DLC-CDS* (*Distributed local Controler- Conected Dominating Set*). Enfin, des résultats d'expérimentation sont illustrés pour analyser le comportement de cette algorithme.

Le reste de ce chapitre est organisé de la manière suivante : nous présentons dans la section 4.3 un bref aperçu sur le paradigme *SDN*, ainsi qu'une description rapide de l'intégration de ce paradigme dans les *RCSFs*. Quelques notions sur les *CDS* ainsi que la logique floue dont nous avons besoin pour la sélection des *CLs* dans l'architecture proposée, sont également présentées. Ensuite nous parcourons les travaux existants dans ce contexte. Dans la section 4.4, une description détaillée sur l'architecture proposée basée sur *SDN* avec ces différents modules est présentée. Nous exposons par la suite le nouveau modèle de sélection des contrôleurs locaux avec la méthode de calcul du score. Enfin, nous développons l'algorithme *DLC-CDS* [10] proposé, tout en utilisant le simulateur *Cooja* sous *Contiki* 2.6. En outre, des résultats d'expérimentation sont illustrés dans cette section, pour analyser le comportement de l'algorithme *DLC-CDS* et le comparer avec l'algorithme *DSP-CDS* existant dans la littérature.

4.2 Contexte et motivation

Avec l'avancement de la miniaturisation et les technologies sans fil, qui, presque dans leur totalité, font partie du réseau mondial, comme les technologies qui nous entourent actuellement : l'*IdO*, *Machine-to-Machine* (*M2M*) et les réseaux de capteurs sans fil (*RCSFs*). Ces derniers types de réseau (*RCSFs*) sont constitués de dispositifs avec une durée de vie et une puissance de traitement limitées, et pour lesquels il est nécessaire d'utiliser aux mieux leurs ressources dans l'*IdO*. De plus, et en raison de leur évolutivité et de leur nature mobile, les *RCSF* avec l'*IdO* exigent une gestion très flexible, intelligente, robuste et efficace par l'implémentation des applications de routage intelligentes et qui prennent en considération les niveaux de batterie des nœuds de capteurs et des concepts architecturels.

C'est pourquoi, la recherche de solutions comme celle du paradigme *SDN* [7] pour les *RCSFs* dans l'*IdO*, est très importante car leurs dispositifs font de plus en plus, partie du réseau mondial. Ainsi le besoin se fait sentir pour l'élaboration d'une architecture intelligente, robuste, programmable et gérable pour satisfaire les exigences de ces futurs réseaux. En effet, *SDN* représente un paradigme

de réseautage émergent qui donne l'espoir de changer les limites les infrastructures des *RCSFs* dans l'*IdO* actuelle.

Dans ce chapitre, nous nous focalisons sur le paradigme *SDN* du point de vue architecturel, où il est question de proposer une nouvelle architecture de contrôle hybride et programmable basée sur ce paradigme. Cette proposition se base sur la répartition de la fonction de contrôle tout en proposons trois types de contrôleurs, y compris les contrôleurs locaux (*CLs*) évoluant au niveau de l'infrastructure des *RCSF* dans l'*IdO*. Le mécanisme des *CLs* est basé sur la méthode de sélection des sous-ensembles connectés *CDS* et des approches de la logique *floue*. Cependant, notre objectif est de réduire à travers cette proposition la charge du trafic de contrôle liée aux mécanismes de contrôle du réseau et de prévenir la défaillance du seul point de contrôle fourni par le contrôleur central du *SDN* classique. Plusieurs facteurs nous ont motivés à proposer et réaliser ce travail et dont nous citons :

- *SDN* est un nouveau paradigme révolutionnaire avec un succès dans le contrôle et la gestion des *RCSF*, ses caractéristiques conceptuelles et logiciels peuvent transformer les paradigmes de réseau existants. Nous voulions donc explorer et tirer profit de ces avantages de contrôle sur les *RCSF*.
- SDN est un paradigme prometteur qui est censé simplifier la gestion des réseaux et minimiser l'enfermement des fournisseurs en adoptant des normes et des spécifications ouvertes.
- SDN peut également être utilisé pour simplifier la configuration du réseau et la gestion des ressources dans les réseaux à ressources limitées.
- Le déploiement de cette technologie pour les réseaux sans fil à ressources limitées.
- Enfin, notre inclination pour la recherche appliquée et donc pratique, ce travail de thèse a à la fois une implémentation logicielle et une expérimentation, chose qui rend le travail de recherche fort intéressant.

4.3 Positionnement bibliographique

Tout au long de ce chapitre, nous présentons un bref aperçu du concept *SDN*, son architecture pour l'*IdO*, ainsi qu'une description de la méthode *CDS* utilisée dans la nouvelle architecture proposée pour *SDN*, par ailleurs, la logique floue sera également exploitée comme un outil d'optimisation pour le choix de la sélection des *CLs*.

4.3.1 Dispositifs à contrainte de ressources

Les dispositifs à ressources restreintes sont de petits périphériques avec des capacités de traitement limitées (processeur et fréquence d'horloge), de stockage (*RAM* et mémoire flash) et fonctionnant souvent avec des batteries à durée de vie limitée. Ces dispositifs peuvent se connecter

les uns aux autres et former des réseaux qui sont utilisés dans différents domaines d'application. Ils constituent les éléments constitutifs des technologies de réseau émergentes telles que les réseaux *RCSF*, *IdO* et *M2M*.

4.3.2 Réseaux définis par logiciels (SDN) : Description du paradigme

L'appellation réseau à définition logicielle, ou *SDN* (*Software-Defined Networking*) [7] [157], désigne un ensemble de technologies innovantes. Le *SDN* est également défini comme une approche de conception de réseau qui facilite la gestion du réseau en réduisant les écarts entre les applications, les services réseau et les dispositifs. Cela peut être réalisé en déployant un seul point de contrôle centralisé communément appelé contrôleur *SDN* [159]. Le contrôleur orchestre, gère et facilite la correspondance entre les applications et les dispositifs réseau. Il expose et résume les fonctions réseau et les opérations via des interfaces programmables aux administrateurs réseau, ce qui leur donne plus de contrôle sur les fonctionnalités du réseau. Les organisations de premier plan qui visent à promouvoir le *SDN* sont les sociétés ouvertes, telles que *Cisco*, *Microsoft*, *Google*, Telecom, etc. [158].

Dans un environnement réseau traditionnel, les plans de contrôle et de données résident sur le même dispositif. Le plan de contrôle, qui peut également être considéré comme le cerveau des dispositifs du réseau, prend toutes les décisions concernant les tables de routage. Le plan de données utilise ces tables pour transmettre les paquets de données. Un dispositif avec un plan de contrôle local devra être configuré manuellement et séparément. En outre, dans un scénario où des centaines de dispositifs pareils doivent être gérés, cela peut s'avérer une tâche fastidieuse. De plus, aucun dispositif n'a la visibilité de l'ensemble du réseau. En d'autres termes, chaque dispositif doit fonctionner seul et partager des informations avec ses voisins pour former une sorte de vue synoptique. Cependant, avec les réseaux traditionnels, les nouveaux protocoles de routage ne peuvent pas être implémentés facilement. Il est également difficile d'intégrer des dispositifs de marques différentes pour fonctionner sur le même réseau car ils gèrent des logiciels propriétaires.

Avec le nouveau modèle du SDN, au lieu d'avoir pour chaque dispositif son propre plan de contrôle, un plan de contrôle commun est implémenté sur un dispositif de contrôle à distance (*Remote Controller*) pour tous les dispositifs du réseau. Ceci introduit une gestion de politique de contrôle centralisée. Ainsi, les dispositifs doivent devenir de simples éléments d'acheminement de paquets tandis que toutes les prises de décision sont effectuées sur le dispositif de contrôle central à distance. Le contrôleur peut donc manipuler le flux de trafic à travers le réseau et cela permet aux dispositifs individuels de gérer eux-mêmes les protocoles et les stratégies de routage et de gérer le trafic réseau, ce qui évite l'encombrement du réseau. Avec *SDN*, un utilisateur peut exécuter plusieurs systèmes

d'exploitation sur des périphériques qui ne sont pas spécifiques à l'application. En fait, tous tendent à convenir que *SDN* simplifie la gestion du réseau [160].

4.3.2.1 Architecture des réseaux programmables par *SDN*

Une architecture *SDN* se présente en trois couches (voir Figure 4.1) à savoir [161] :

- **Couche d'application :** Elle prend en charge les applications qui communiquent avec le contrôleur et le dirige pour 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 réseau dans la prise de décision concernant la gestion du réseau. Ces applications peuvent également être utilisées pour effectuer une analyse de données.
- **Couche de contrôle** : Elle contient le contrôleur de réseau qui est l'entité principale interconnectant la couche d'application et d'infrastructure. Le contrôleur est responsable de la gestion de la communication entre les deux couches. Il transmet les instructions reçues des applications aux périphériques physiques ou virtualisés sous-jacents et collecte les données de ces périphériques pour les renvoyer aux applications.
- **Couche d'infrastructure :** qui est constituée de dispositifs réseau physiques qui exécutent le transfert réel de données. Cela inclut 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 sont appelées l'interface *Northbound*, tandis que la connexion entre le contrôleur et le matériel réseau physique est connue sous le nom d'interface *Southbound*. Le *SDN* repose essentiellement sur les quatre piliers suivants [161] : a) Les plans de contrôle et de données doivent être séparés les uns des autres.

b) Les décisions d'acheminement doivent être basées sur les flux plutôt que sur .la destination.

c) La logique de contrôle doit être déplacée vers un contrôleur *SDN* externe.

d) Le plan de commande du réseau doit être rendu directement programmable.

Avec SDN, existe aussi trois composants majeurs qui peuvent être listés comme suit :

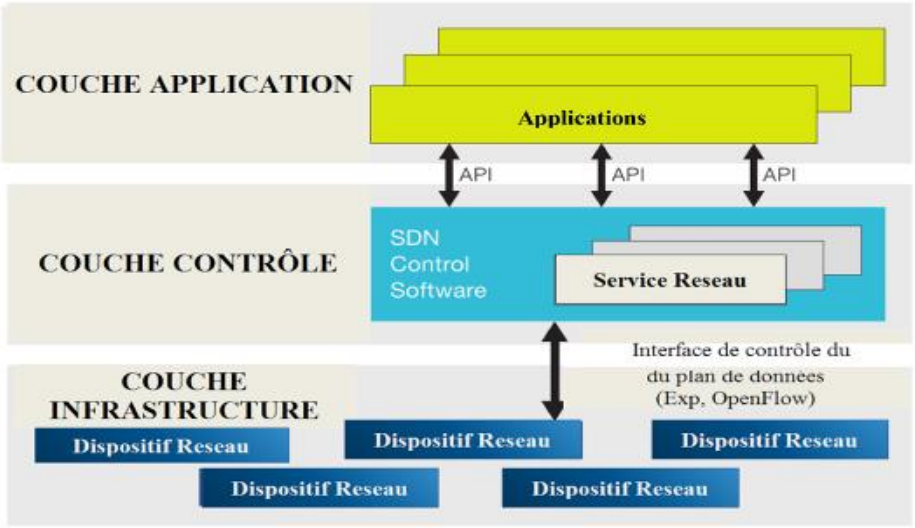


Figure 4.1 : L'Architecture du paradigme *SDN* [161].

A) Plan de contrôle

La tâche principale d'un plan de contrôle est de créer des tables de transfert de données pour le plan de données [157]. Le plan de contrôle prend ses décisions en fonction des informations fournies par la base d'informations de routage (*Routing Information Base* (*RIB*)), qui est l'entité qui stocke la topologie du réseau. Il recueille des informations par l'observation, la programmation manuelle ou l'intégration avec d'autres entités du plan de contrôle. Une fois les décisions prises, elles sont ensuite stockées dans la base d'informations de transfert (*Forwarding Information Base* (*FIB*)) qui est responsable de la transmission des paquets vers leur interface appropriée. Le plan de contrôle peut être de trois types :

- **Strictement centralisée :** Cette approche du modèle *SDN* est appelée « approche révolutionnaire » car elle propose une séparation complète du plan de contrôle du dispositif et de son infrastructure physique. Dans ce modèle, aucune fonction de plan de contrôle n'existe sur un appareil, cette fonction est en effet totalement prise en charge par le contrôleur centralisé situé à distance.
- **Semi-centralisé :** Un plan de contrôle semi-centralisé, est nommé « *approche évolutive* ». Il fournit de nouvelles fonctionnalités mais ne supprime pas complètement le plan de contrôle de l'appareil. Certaines fonctionnalités du plan de commande, tel que l'apprentissage des adresses MAC, sont toujours exécutées sur le périphérique, tandis que le contrôleur centralisé reçoit plus d'autorité sur les autres domaines de la fonctionnalité réseau. Ce modèle utilise les

meilleures caractéristiques des plans de contrôle strictement centralisés et entièrement distribués.

- **Entièrement distribué :** Dans ce modèle, chaque appareil exécute un plan de contrôle complet pour chaque plan de données. Tous les plans de contrôle sont interconnectés pour former un réseau cohérent. Cette approche n'offre rien de nouveau et a donc peu de signification.

B) Plan de données

Un plan de données en *SDN* est celui qui effectue le transfert réel de paquets de données. Ces paquets sont ensuite fonction des tables de flux qui leur sont affectées par le contrôleur. Un flux est un ensemble de valeurs de champ de paquet qui filtrent les paquets entrants. Si un paquet correspond aux critères définis dans un flux particulier, les actions correspondantes sont effectuées sur ce paquet en fonction des instructions fournies par le contrôleur. Tous les paquets appartenant à un flux particulier recevront un traitement identique.

Dans le cas où un paquet n'appartient pas aux entrées de la table de flux répertoriées, le périphérique demandera alors au contrôleur de fournir de nouvelles instructions pour traiter ce paquet. Les tables de flux peuvent être facilement mises à jour en cas de modification de politique. Plusieurs méthodes ont été proposées pour un transfert de paquets rapide et rentable [157]. La classification matérielle peut être utilisée pour augmenter le débit de traitement car l'utilisation de logiciels dans les commutateurs peut entraîner des performances inefficaces.

C) Plan de gestion

Le plan de gestion est responsable de l'exécution des tâches qui sont en dehors de la portée du contrôle et des plans de données. Il gère les allocations de ressources, les accords commerciaux client-fournisseur, la mise en place d'une infrastructure réseau physique et sa configuration. Chaque entreprise a ses propres entités administratives.

4.3.2.2 Exemple de contrôleur *SDN*

L'entité qu'un plan de contrôle utilise pour gérer les contrôles de flux dans un réseau s'appelle un contrôleur *SDN*. Citons dans ce sens l'exemple du contrôleur *SDN OpenDaylight* (*ODL*) [162], dont l'architecture *ODL* définit les éléments suivants :

- Une base de données qui stocke des informations concernant l'état du réseau, sa configuration et sa topologie.
- Un modèle de données de haut niveau qui établit des relations

Le projet *ODL* est une initiative de la fondation *Linux* avec l'objectif de souligner l'importance de *SDN*. L'*ODL* offre le plus grand contrôleur *SDN* open source utilisé dans diverses organisations et universités. L'architecture d'*ODL* selon [162] comporte une interface *Southbound* qui prend en charge l'environnement multifournisseur et une interface *Northbound* qui offre plusieurs fonctionnalités à diverses applications via différentes applications (*API*). De plus, il existe une couche *LAS* (*Service Abstraction Layer*) qui non seulement interconnecte les demandes de service aux plugins concernés, mais fournit également une plate-forme de base pour la construction de services de niveau supérieur. Les normes de protocole ouvert tels que *OpenFlow* [158] ou les protocoles standards peuvent être utilisés pour communiquer avec le physique ou le matériel virtualisé.

Certaines des fonctionnalités clés offertes par le contrôleur *ODL* sont les suivantes [162] :

- ***Services à la demande*** : il fournit des services facilement disponibles sur la planification de la bande passante.
- ***Cloud computing et virtualisation*** : il offre un service de qualité sur les infrastructures cloud tel *OpenStack* qui est le plus couramment utilisé.
- ***Optimisation des ressources*** : optimise dynamiquement les ressources réseau en fonction de l'équilibrage de charge.
- ***Modèle de réseau fiable*** : Il fournit des modèles de réseautage hautement actifs et automatisés pour les réseaux du gouvernement, des universités et du secteur privé.
- ***Visibilité et contrôle du réseau*** : Il offre une administration centralisée de l'ensemble du réseau à l'aide d'un ou de plusieurs contrôleurs.

4.3.3 Paradigme du *SDN* dans les *RCSFs*

Plusieurs *RCSFs* sont déployés pour plusieurs applications dans la même zone. De même, les fournisseurs ne parviennent pas à utiliser les fonctionnalités communes car ils développent un *RCSF* de manière isolée. En outre, la nature de déploiement à distance du *RCSF* nécessite des dispositifs hautement autonomes et configurables automatiquement chose qui n'est guère réalisable en raison des limitations de ressources de ces dispositifs. Certains des problèmes communs dans les *RCSF* sont l'économie d'énergie, la mobilité de nœud capteur, la gestion de réseau, la précision de localisation et un *RCSF* virtualisé [163]. Le *SDN* peut contribuer à améliorer ces performances.

Ce paradigme encourage le développement de protocoles rentables qui peuvent conduire à une augmentation considérable de la productivité du *RCSF*.

La séparation du plan d'acheminement de la logique de contrôle permet une gestion plus aisée du réseau et permet la virtualisation du réseau.

En outre, la popularité croissante de l'*IdO* a entraîné une production et un déploiement à grande échelle des réseaux *RCSFs* [164]. La prochaine décennie pourrait voir des milliards de nœuds de capteurs interconnectés reliés par Internet dans lesquels le *SDN* pourrait fournir une plate-forme solide pour gérer un si grand nombre de dispositifs en réseau et également résoudre certains des problèmes clés rencontrés par les *RCSFs*.

Les fonctionnalités les plus significatives qui peuvent être obtenues en utilisant des nœuds capteurs activés par *SDN* sont la gestion des nœuds et des ressources [163]. Un contrôleur peut prendre en compte l'énergie disponible pour différents nœuds tout en prenant les décisions de routage pour assurer la meilleure durée de vie du réseau.

- **Economie d'énergie avec *SDN***

Avec des ressources énergétiques limitées dans les *RCSFs*, les nœuds sont toutefois déployés dans des situations où ils ont un accès limité à toutes les ressources d'énergie renouvelable. Ceci restreint en conséquence le développement de protocoles efficaces en énergie, qui à leur tour affecte la performance globale du réseau. Avec le *SDN*, la puissance consommée par les nœuds peut être considérablement réduite [167]. Le contrôleur peut déterminer les meilleures politiques de routage et éviter ainsi aux nœuds de prendre leurs décisions eux-mêmes. Dans le cas où le nœud est sur le point de manquer de batterie, il enverra un avertissement au contrôleur afin qu'il puisse modifier les tables de routage à temps [168]. De plus, puisque le contrôleur prend en charge les fonctionnalités du plan de contrôle, la gestion du trafic, l'allocation des ressources et la qualité de service (*QdS*) peuvent être efficacement réalisées avec un surcoût énergétique moindre.

- **Mobilité du nœud de capteur avec SDN**

Dans le cas des capteurs mobiles, la topologie du réseau change fréquemment, ce qui entraîne un temps de convergence retardé pour les protocoles de réseau basés sur le vecteur. Cela affecte les performances globales du réseau étant donné qu'un *RCSF* a une topologie spécifique pour une application spécifique [169].

Lorsqu'une application change, la topologie réseau correspondante change également. Ce faisant, les nœuds du capteur perdent de l'énergie et leur durée de vie sera raccourcie. Avec *SDN*, un contrôleur centralisé peut soit injecter ou modifier les politiques de réseau à la volée. Cela entraînera une diminution du temps de convergence pour les protocoles. Le contrôleur affecte également un protocole de gestion de la mobilité qui ordonne aux nœuds d'informer en continu le contrôleur de

leurs informations de localisation. De cette façon, le contrôleur continuera à mettre à jour les tables de flux avec de nouvelles décisions de routage et garantira ainsi des performances réseau optimales.

- **Gestion de réseau avec *SDN***

La gestion de réseau est un processus complexe et difficile pour les administrateurs des *RCSFs*. La mise en réseau traditionnelle nécessite la gestion de logiciels propriétaires sur des dispositifs matériels propriétaires. Dans le cas de nœuds de capteurs, l'utilisation de composants réseau de différents fournisseurs rend le processus de gestion encore plus complexe [165]. Le coût de gestion d'un RCSF est relativement élevé et toute nouvelle implémentation de politique ou de protocole nécessiterait la modification du matériel des nœuds capteurs. Un tel processus nécessite un accès physique à tous les nœuds capteurs, ce qui n'est pas toujours possible. Par conséquent, *SDN* peut aider à transformer le problème d'administration réseau en un problème de programmation réseau. Ainsi, la complexité du *RCSF* est considérablement allégée avec *SDN*. Par ailleurs, de nouveaux protocoles de routage peuvent être facilement utilisés sur le réseau et facilitent ainsi la compilation de différentes versions des mêmes applications réseau pour différents types de nœuds de capteurs.

- **Précision de localisation avec *SDN***

Les données fournies par un nœud de capteur sans informations de localisation correctes pourraient être considérées comme inutiles. En raison de la nature à contrainte d'énergie des nœuds, la mise en réseau traditionnelle ne peut pas obtenir d'informations de localisation très précises car elle nécessite l'exécution d'algorithmes de localisation sophistiqués qui pourraient représenter une surcharge pour ces périphériques. Il est montré dans [165] qu'avec *SDN*, une information de localisation très précise peut être obtenue en utilisant un algorithme de routage centralisé. Les informations de localisation collectées peuvent être utilisées par un algorithme de découverte de topologie de réseau pour améliorer encore les décisions de routage prises par le contrôleur.

- **Les *RCSFs* virtualisés avec *SDN***

Il est suggéré dans [164] que l'application de *SDN* dans les *RCSFs* permettra à différentes organisations et applications de partager la même infrastructure physique sous-jacente au lieu de déployer des réseaux séparés. Cela se traduira par une réduction des coûts pour les clients, un coût de propriété réduit permettant au réseau de se développer économiquement. Bien que le *SDN* n'ait pas été conçu pour les *RCSFs* à contrainte de ressources, ses fonctionnalités peuvent être exploitées pour former un environnement virtualisé pour ce type de réseau.

4.3.3.1 Architecture du *SDN* dans les *RCSFs*

Ces dernières années, l'utilisation des réseaux définis par logiciel (*SDN*) dans les *RCSF* est devenue une branche de recherche émergente et importante attirant de plus en plus l'attention.

En effet, l'idée originale d'exploiter la technologie *OpenFlow* [158] pour résoudre les problèmes de fiabilité dans les *RCSFs* a été présentée par [170], tandis que la première proposition architecturale a été présentée par [171] sous la forme *SD-WSN* (*Software Defined Wireless Sensor Network*). Certaines des architectures notables proposées pour l'utilisation de *SDN* dans les *RCSFs* sont les suivantes :

a) Réseau de capteurs sans fil défini par logiciel (*SD-WSN*)

b) *TinySDN*

c) Mise en réseau centrée sur le service pour l'évaluation à l'échelle *URban Systèmes* (*SURF*)

d) Mise en réseau logicielle dans les capteurs sans fil (*SDN-WISE [171]*).

Dans cette sous-section, nous allons présenter une seule architecture qui concerne le *SDN-WISE* constitué d'un *SDN* dans un *RCSF*. Pour davantage de détails sur cette architecture et les autres architectures, nous proposons de consulter les travaux de recherche des auteurs de [162] et [163]. La figure 4.2 illustre l'architecture de *SDN-WISE [171]* avec les nœuds capteurs et le *Sink* dans un RCSF.

- Dans [162], les auteurs ont démontré que les architectures mentionnées dans leurs travaux ne sont pas mises en œuvre dans la pratique et aucune évaluation des performances des solutions proposées n'a donc été réalisée. Par contre, *SDN-WISE* conçu dans [171] est la première implémentation pratique d'une solution *OpenFlow* comme le *SDN* conçu spécifiquement pour les réseaux de stockage de masse. Contrairement aux autres architectures le *SDN-WISE* vise à limiter l'échange d'informations entre les nœuds et le contrôleur, mais aussi pour rendre les nœuds de capteurs directement programmables.

De plus, *SDN-WISE* [171] offre la facilité d'implémentation de la logique du contrôleur *SDN*. Cela représente un avantage majeur par rapport aux solutions proposées et étudiées dans [162] car il augmente la flexibilité et la simplicité de la programmation réseau. Le *SDN-WISE* offre également la possibilité de mise en pratique de son contrôleur dans un environnement simulé. En effet, des logiciels de simulation tels que *OMNET ++* et *COOJA* peuvent être utilisés pour tester ces fonctionnalités.

- D'après [162], le *SDN-WISE* s'efforce d'être compétent dans l'utilisation des ressources de capteur, indépendamment du fait qu'une telle productivité puisse entraîner un débit de données inférieur. Pour être économe en énergie, il encourage l'utilisation du cycle de service pour allumer et éteindre périodiquement le module radio, ce qui aiderait à économiser l'énergie.

De plus, comme les *RCSF* sont intrinsèquement pilotés par les informations, cela rend le système plus conscient du contenu des paquets. Les nœuds peuvent traiter des paquets en fonction des informations disponibles dans leur en-tête et leur charge utile. Des opérateurs relationnels plus complexes sont également introduits dans les tables de flux. Dans *OpenFlow*, les ressources système sont séparées par le *FlowVisor* en petites parties. Comme illustré dans la figure 4.2, chaque partie est associée à un seul contrôleur à la fois.

Le *SDN-WISE* [171] permet donc également à différents contrôleurs de spécifier des règles différentes pour un même paquet en fonction de leurs besoins. Comme le montre la figure 4.2, dans l'architecture de *SDN-WISE* on trouve les nœuds de capteurs *SDN-WISE*, sink, les serveurs qui représentent les contrôleurs centraux. *SDN-WISE* définissent le comportement des nœuds capteurs à être complètement codés dans trois structures de données : 1) le tableau des états *WISE*, 2) le tableau des identifiants acceptés et 3) le tableau de flux *WISE* [173]. Ces structures sont ensuite remplies par les instructions provenant des contrôleurs s'exécutant sur les serveurs distants.

Les contrôleurs définissent les dispositions d'administration des systèmes qui sont ensuite mises en œuvre par les nœuds de capteurs. A tout moment, chaque nœud est décrit par un état courant pour chaque contrôleur actif.

En termes plus précis, le tableau d'état *WISE* est la structure de données qui contient ces valeurs. La nature de diffusion du support sans fil permettra aux nœuds de recevoir tous les paquets de données dont certains ne leur sont peut-être pas destinés.

Un autre exemple d'utilisation du *SDN* pour les *RCSF* est réalisé par les auteurs de [172], qui ont proposé un nouveau cadre (*framework*) des réseaux de capteurs et actionneurs sans fil (*RCASF*) basés sur *SDN*, et dont l'architecture est illustrée dans la figure 4.3.

- Les auteurs de [172], ont étudié le cadre d'application et les méthodes pertinentes pour appliquer l'approche SDN dans un *WSAN*, dans le but d'améliorer l'efficacité et l'évolutivité du réseau. Les détails du cadre comprennent une structure à trois couches, 1) les entités système pertinentes, 2) la pile de protocole améliorée et 3) les types de messages programmables pour la communication coopérative et l'exécution de tâches parmi les nœuds *RCASF*. Sur la base de ce cadre, ce document explore les défis et mécanismes pertinents pour une gestion efficace du système à partir de nombreux aspects, y compris la mobilité, les économies d'énergie, la maintenance de la fiabilité et la construction de la topologie. Les auteurs ont proposé également une méthode d'optimisation pour l'ordonnancement de tâches décomposées sur des nœuds pertinents, avec un exemple implémenté par l'algorithme génétique. Ensuite, les auteurs y présentent les scénarii d'application typiques, y compris la surveillance des catastrophes militaires, industrielles, celles liées au transport et

environnementales. De plus, un scénario d'application intérieure et un scénario d'application extérieure sont présentés pour démontrer l'application du transfert de communication assisté par *SDN*. La figure 4.3 montre un exemple de réalisation d'architecture des *RCASF* avec le concept *SDN*.

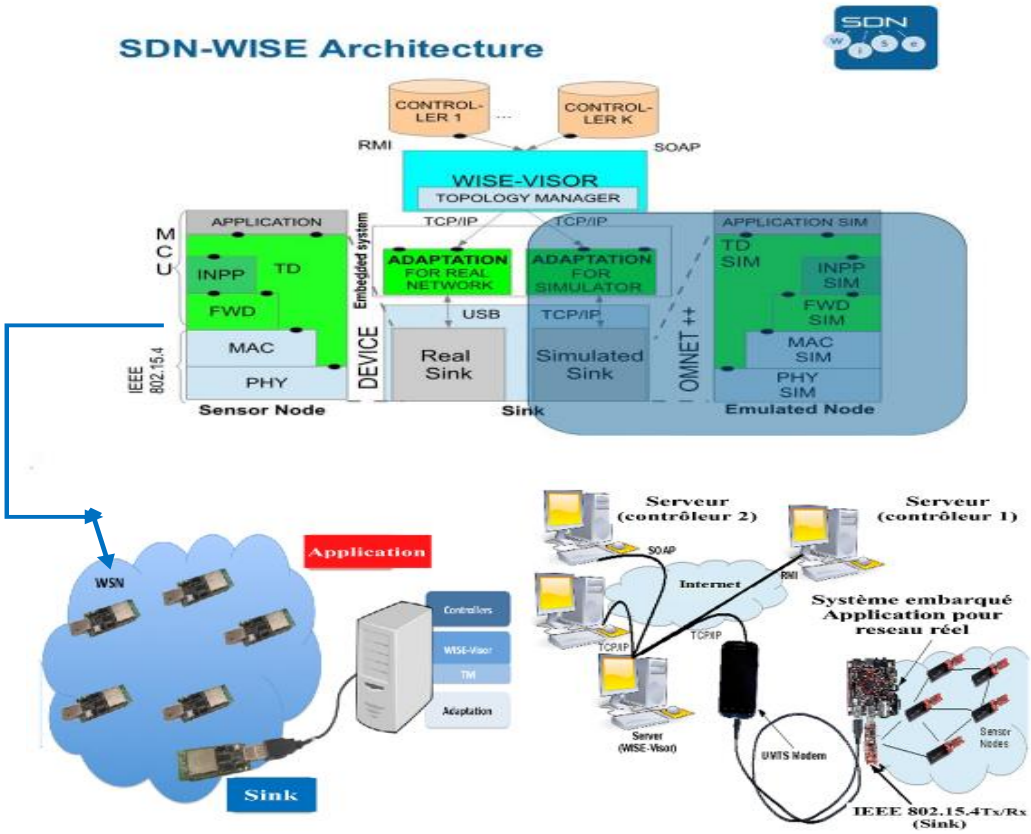


Figure 4.2 : Architecture de SDN-WISE [162].

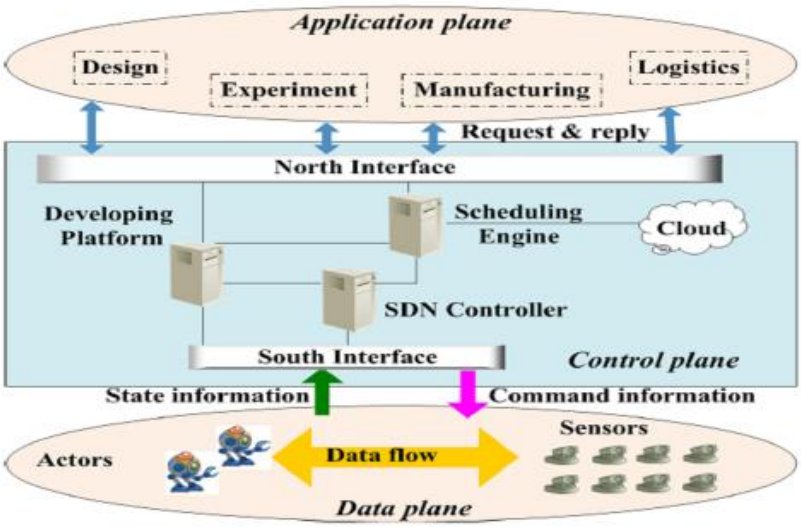


Figure 4.3 : Le cadre des RCASF basés sur *SDN* [172].

4.3.3.2 Classification des architectures SDN pour IdO

Par ailleurs, notre architecture, proposée dans ce chapitre et pour cette thèse, représente une architecture hybride et semi-distribuée basé sur le concept *SDN*, tout en proposant la rénovation de la fonction du contrôle via l'utilisation des contrôleurs locaux (*CL*). En effet, le choix et la sélection des *CLs* sont effectués par l'application de l'algorithme *CDS*.

Nous présentons dans cette section, un bref aperçu des architectures *SDN* existantes pour l'Internet des Objets et plus particulièrement dans les *RCSFs*, puis nous parcourons les travaux centrés sur les algorithmes *CDS* (*Connected Dominating Sets*) avec différentes approches.

En effet, il existe deux catégories de recherche pour l'exploitation de la position du contrôleur *SDN* (*SDN-C*ontroller) dans un RCSF, l'une étant la classe des SDN-Contrôleurs en position centrale (*Unique Central SDN-Controller*), la seconde classe correspond aux SDN-Contrôleurs en multi position (*Multi SDN-Controller*).

A) La classe des *SDN-Contrôleurs* en multi position

Le *SDN* a été proposé dans la littérature comme une solution prometteuse pour introduire une solution flexible pour ce *SDN*. Par exemple, les problèmes de normalisation et l'impact de *SDN* sur les RCSFs et *IdO* en termes de déploiements sont discutés dans [173] [174].

- Dans [173], l'auteur présente l'introduction de *SDN* dans les *RCSFs* et en particulier le protocole de plate-forme *Tiny* nommé *TinySDN*. Le protocole de communication entre les nœuds du contrôleur et du capteur est y largement décrit.
- Dans [174], *TinySDN* est proposé en tant qu'architecture basée sur *TinyOS* à plusieurs contrôleurs dans les RCSFs. Il définit deux composants principaux : 1) le nœud capteur SDN-activé, et 2) le nœud contrôleur SDN. Les résultats expérimentaux de *TinySDN* sont présentés et discutés dans ce travail. Cependant, la description de la communication entre les contrôleurs et le processus de sélection du contrôleur n'est pas présente. En outre, *TinySDN* est basé sur une approche centralisée sans aucun détail sur la valeur ajoutée du processus de routage comme le protocole RPL.

B) La classe des SDN-contrôleurs en proposition centrale

Un autre travail est proposé pour étendre le *SDN* aux *RCSF* celui du *SDN-WISE* [171] précédemment décrit avec son architecture en section 4.3.3.1, pages 123-124.

- Dans [171], les auteurs présentent une solution efficace de *SDN* pour les *RCSFs* dans le but de réduire les informations échangées entre les nœuds de capteurs et les contrôleurs *SDN*. Deux types de nœuds sont définis dans le réseau *SDN-WISE* : le nœud de capteur fonctionnant sur le

plan de données et le nœud de puits qui sont les passerelles entre les nœuds de capteurs et le contrôleur qui s'exécute sur le plan de contrôle. Le *WISE-Visor* est défini comme la couche d'adaptation pour le contrôle de gestion. L'amélioration du protocole *SDN-WISE* est proposée dans [175] pour réduire les surcharges liées au contrôle du réseau, et pour supporter le gros traitement de données, en utilisant l'opération *MapReduce*.

- Dans [176], le système d'exploitation Open Network (*ONOS*) est proposé comme une amélioration du *NOS* (*Network Operating System*) pour l'*IdO*. L'idée consiste à distribuer un *OS* de réseau pour gérer les opérations réseau en utilisant l'approche *SDN*. La solution *ONOS* a été étendue pour améliorer encore la plate-forme *SDN-WISE*, proposée récemment, afin de prendre en charge *SDN* en utilisant le standard *OpenFlow* dans les *RCSFs* [170].

Selon l'architecture proposée par *SDN-WISE* [171], il convient de noter que les nœuds de capteurs effectuent des opérations sur le plan de données, qui dépassent leur capacité de mémoire et de consommation d'énergie.

D'autre part, sachant que le nœud *Sink* est différent des nœuds de capteurs en termes de capacité de ressources, il ne représenterait qu'une simple passerelle entre le plan de données et le plan de contrôle. Le *SDN-WISE* ne respecte pas les caractéristiques des *RCSFs* en termes de ressources limitées (processus, occupation de la mémoire et consommation d'énergie). De plus, il y a un manque de coopération entre les contrôleurs au cas où ils fonctionneraient dans le même nœud hébergeant la couche TM ou dans des serveurs distants. Enfin, l'architecture principale proposée à base de *SDN* est considérée comme une approche centralisée avec un seul contrôleur situé sur l'infrastructure *Sink* ou réseau.

4.3.4 Concept des ensembles dominants connectés (CDS)

Avant de parcourir les travaux effectués sur cette approche, nous allons présenter quelques notions de base sur les sous-ensembles des dominants connectés (*Connected Dominating Set* (*CDS*)) [8], ainsi qu'un aperçu sur la classification des algorithmes des *CDS*.

4.3.4.1 Concept de base des CDS

L'un des concepts les plus connus et les mieux étudiés pour construire un réseau *backbone* virtuel dans des *RCSFs* est la *CDS*. Ce concept aide à surmonter le problème de tempête de diffusion et facilite le routage. Les *CDS* ou le *backbone* virtuel fait partie de la théorie des graphes. Dans ce qui suit, nous allons donner la définition de ces différentes structures. Pour ce faire, nous avons besoin d'introduire quelques notions nécessaires, tel que le réseau des nœuds modélisé par un graphe non orienté $G = (V, E)$, où V représente l'ensemble des nœuds et l'ensemble E correspond à l'ensemble

des arêtes représentant les liens existants entre les différents nœuds. La notation *N(v)* correspond au voisinage d'un nœud *v* et inclut les sommets adjacents à v tout en comptabilisant le nœud v. Les structures présentées se basent sur la notion de dominance. Ainsi, un nœud appartenant à ces structures est appelé dominant, les autres nœuds étant des dominés [8].

- **Un ensemble dominant (DS : Dominating Set) :** est un ensemble de nœuds dans lequel tout nœud du réseau est voisin d'au moins un nœud de cet ensemble. Soit V' un ensemble dominant du graphe *G (V, E)*, alors selon la définition mathématique 4.1 (voir figure 4.3 (a) :

$$\forall u \in V, \exists\, v \in V' \,/\, v \in N(u) \qquad (4.1)$$

- **Un ensemble dominant connecté (*CDS* : Connected Dominating Set) :** est un ensemble dominant où tous les nœuds dominants sont directement connectés, la figure 4.3(b) illustre cette définition. Soit V un ensemble dominant connecté du graphe *G (V, E)* alors :

$$\forall u \in V, \exists\, v \in V' \,/\, v \in N(u) \qquad (4.2)$$

$$\forall (u, v) \in V'^2, \exists\, c = chemin_{u \to v} \,/\, \forall\, w \in c, w \in S \qquad (4.3)$$

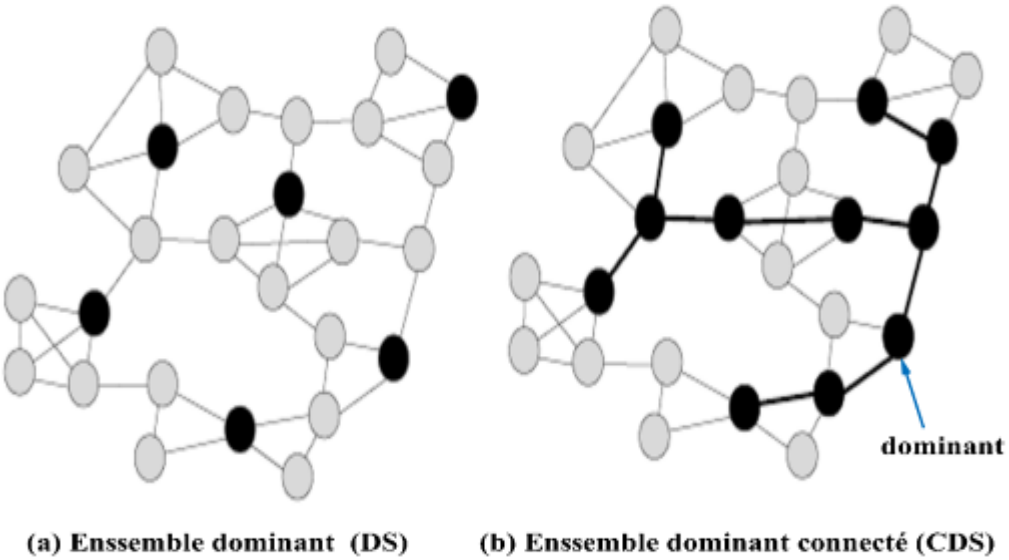


Figure 4.4 : Exemple de structures de l'ensemble des dominants.

4.3.4.2 Classification des algorithmes de construction de *CDS*

Les algorithmes de construction *CDS* peuvent être divisés en deux catégories [8] : 1) Algorithmes centralisés et 2) Algorithmes décentralisés. En littérature, nous distinguons non seulement ces deux algorithmes de construction de *CDS*, mais aussi ceux à plusieurs phases de ceux à une seule phase.

A) Les algorithmes centralisés

La connaissance des informations à l'échelle du réseau est essentielle, ces algorithmes donnent un *CDS* de plus petite taille par rapport aux algorithmes décentralisés. Ces algorithmes supposent que les informations complètes sur la topologie du réseau sont disponibles, ce qui n'est généralement pas pratique dans le cas des réseaux sans fil mobiles. De plus, il n'est pas toujours possible de contrôler les nœuds dans les réseaux sans fil à partir d'une autorité centralisée.

B) Les algorithmes décentralisés

Dans cette classe d'algorithmes, les informations sur le réseau local sont essentielles. Ces algorithmes définissent eux-mêmes 2 classes de catégorie [8] : 1) Algorithmes distribués et 2) Algorithmes localisés. Dans le cas d'algorithmes localisés, le processus de décision est distribué avec l'exigence supplémentaire d'un nombre constant de tours de communication [8]. Les algorithmes de construction distribués *CDS*, dans la littérature, ont généralement au moins un point de synchronisation réseau, que tous les nœuds du réseau doivent atteindre avant de poursuivre le processus de construction. Le point de synchronisation divise le processus de construction en phases séparées, ces phases s'assurent que l'ensemble dominant est connecté. De nombreux algorithmes distribués ont été discutés et analysés dans [8]. Un algorithme avec plusieurs phases séparées a toutefois deux inconvénients principaux. L'un concerne la faible évolutivité et l'autre a trait à la faible capacité à s'adapter à la topologie de réseau dynamique, chose qui s'avère typique dans les réseaux mobiles.

Quant à notre contribution, et dans sa deuxième partie de notre recherche nous nous sommes basés sur le principe des algorithmes distribués à une seule phase. En effet, Dans [8], l'auteur propose l'algorithme *DSP-CDS* (*Distributed Single-Phase-CDS*), un nouvel algorithme de construction *CDS* monophasé (single-Phase) distribué pour les réseaux ad hoc. L'algorithme converge rapidement en une seule phase et génère un *CDS* de petite taille compétitive. Dans *DSP-CDS*, chaque nœud utilise uniquement les informations de ses voisins à saut unique pour prendre une décision locale quant à l'opportunité de rejoindre l'ensemble dominant, les dominateurs, quant à eux, sont sélectionnés simultanément sur le réseau.

La *DSP-CDS* n'a pas besoin de phase de connexion séparée. Les clusters (appelés morceaux) dans *DSP-CDS* se développent jusqu'à ce qu'ils soient connectés les uns aux autres, et les plus gros morceaux fusionnés continuent de croître jusqu'à ce qu'un *CDS* soit formé dans le réseau. Chaque nœud base sa décision sur une variable clé, *Strength*, qui garantit que l'ensemble dominant est connecté lorsque l'algorithme converge. Les règles de calcul de la puissance peuvent être modifiées pour s'adapter à différents besoins d'application, par exemple pour inclure la capacité d'équilibrer la consommation d'énergie entre les nœuds voisins. *DSP-CDS* s'adapte bien aux topologies de réseau dynamiques provoquées par la mobilité de nœud, la panne de nœud ou le déploiement de nouveaux nœuds. Lors de changements de topologie, les nœuds ajustent leurs forces et déclenchent des mises à jour locales de la structure *CDS* actuelle.

4.3.4.3 Avantages de CDS

Certains des avantages des protocoles de routage dans les *RCSFs* basés sur *CDS* sont :

- *CDS* est utile dans le routage, la diffusion et l'évitement des collisions.
- Le routage basé sur *CDS* réduit le processus de recherche de chemin et de routage au sous-réseau induit par le *CDS*. Seuls les dominateurs doivent maintenir les informations de routage.
- L'efficacité du routage multicast peut être améliorée via *CDS*.
- Restreindre le routage au *CDS* réduit la surcharge de message associée aux mises à jour de routage. *CDS* a formé une architecture sous-jacente utilisée par les protocoles, notamment la coordination de l'accès aux médias, la monodiffusion, la multidiffusion / diffusion, le routage basé sur la localisation, la conservation de l'énergie et le contrôle topologique.
- La consommation d'énergie, une préoccupation essentielle dans les réseaux sans fil, peut être réduite en utilisant *CDS* comme nœuds d'acheminement.
- Le *backbone* virtuel formé par l'ensemble dominant peut propager des informations de qualité de liaison pour la sélection d'itinéraire pour le trafic multimédia et peut servir de serveurs de base de données [8].

Pour plus de détails sur la classification et la performance de l'algorithme de construction *CDS*, une comparaison des principaux travaux liés à la construction de *CDS* a également été fournie dans [8]. Dans le présent chapitre, nous nous intéressons à l'algorithme distribué de construction de *CDS* et en particulier l'algorithme distribué à une seule phase *DSP-CDS* (*Distributed Single-Phase CDS construction algorithm*) [8]. Cette approche est adaptée aux réseaux sans fil.

4.4 Modèle de contrôle fiable utilisant le paradigme du *SDN*

Dans cette section, nous abordons l'hétérogénéité des technologies de réseaux sans fil dans le contexte IdO tout en proposant une méthode fiable de contrôle utilisant le concept des réseaux définis par logiciels (*SDN*). Cette méthode consiste à proposer une nouvelle architecture hybride et programmable basée sur le paradigme *SDN* (*Software Defined Network*). Contrairement au *SDN* classique, nous proposons une approche semi-distribuée avec trois niveaux de contrôle : les contrôleurs : principal (*CP*), secondaire (*CS*) et local (*CL*).

4.4.1 Architecture proposée du *SDN* dans les *RCSFs*

Dans cette section, nous décrivons en détail l'architecture proposée, qui est hybride et proactive basée sur le paradigme *SDN* avec l'approche semi-décentralisée pour le processus de contrôle du réseau. Cette architecture est hybride car elle est capable de supporter une approche centralisée et semi-distribuée pour le processus de contrôle. Elle est décrite comme proactive en raison de sa capacité à prendre une décision dynamique sur la stratégie du processus de contrôle, tout en tenant compte des caractéristiques du réseau.

Dans l'*IdO*, de nombreuses technologies de communication et de réseau existent, mais avec des caractéristiques différentes et hétérogènes abordées dans le premier chapitre de cette thèse, dans la section : communication à courte et longue portée, topologie statique et dynamique, communication à haut et à faible débit, avec et sans contrainte énergétique, communication sous licence et sans licence, etc. Cependant, la coexistence de ces technologies et la gestion des ressources réseau demeure encore une tâche difficile dans *IdO* [177,178,179].

L'utilisation du paradigme *SDN* permet de bénéficier des principaux avantages, qui sont la séparation entre les plans de contrôle et de données, et le fait de faciliter la configuration et le déploiement du réseau. Le plan de contrôle a un rôle clé dans le réseau, notamment dans la gestion des ressources, le contrôle d'admission, le routage et l'acheminement processus, etc. Quant au plan de données, il se concentre uniquement sur le contenu des paquets et leurs caractéristiques en termes de besoins en ressources. C'est pourquoi, la flexibilité et la reconfigurabilité du réseau constituent une réelle valeur ajoutée dans des réseaux hétérogènes ayant des caractéristiques et des besoins différents.

Contrairement aux architectures existantes basées sur le paradigme *SDN* où le nœud de contrôle est centralisé pour avoir une vue d'ensemble du réseau [173,160,180], dans l'architecture proposée, nous introduisons différents niveaux du processus de contrôle : 1) *Contrôleur Principal* (*CP*), 2) *Contrôleur Secondaire* (*CS*), et 3*) Contrôleur Local* (*CL*).

L'idée principale consiste à sélectionner des nœuds particuliers pour agir en tant que contrôleur dans certaines parties du réseau. La figure 4.5 illustre un aperçu de l'architecture proposée divisée en

cinq modules principaux : (1) Module de contrôle, (2) Module de données, (3) Module Cloud et Fog Computing et, (4) Module de sécurité et de confidentialité, et (5) module utilisateurs finaux. L'interaction existante entre ces modules est illustrée dans la figure 4.6 [10].

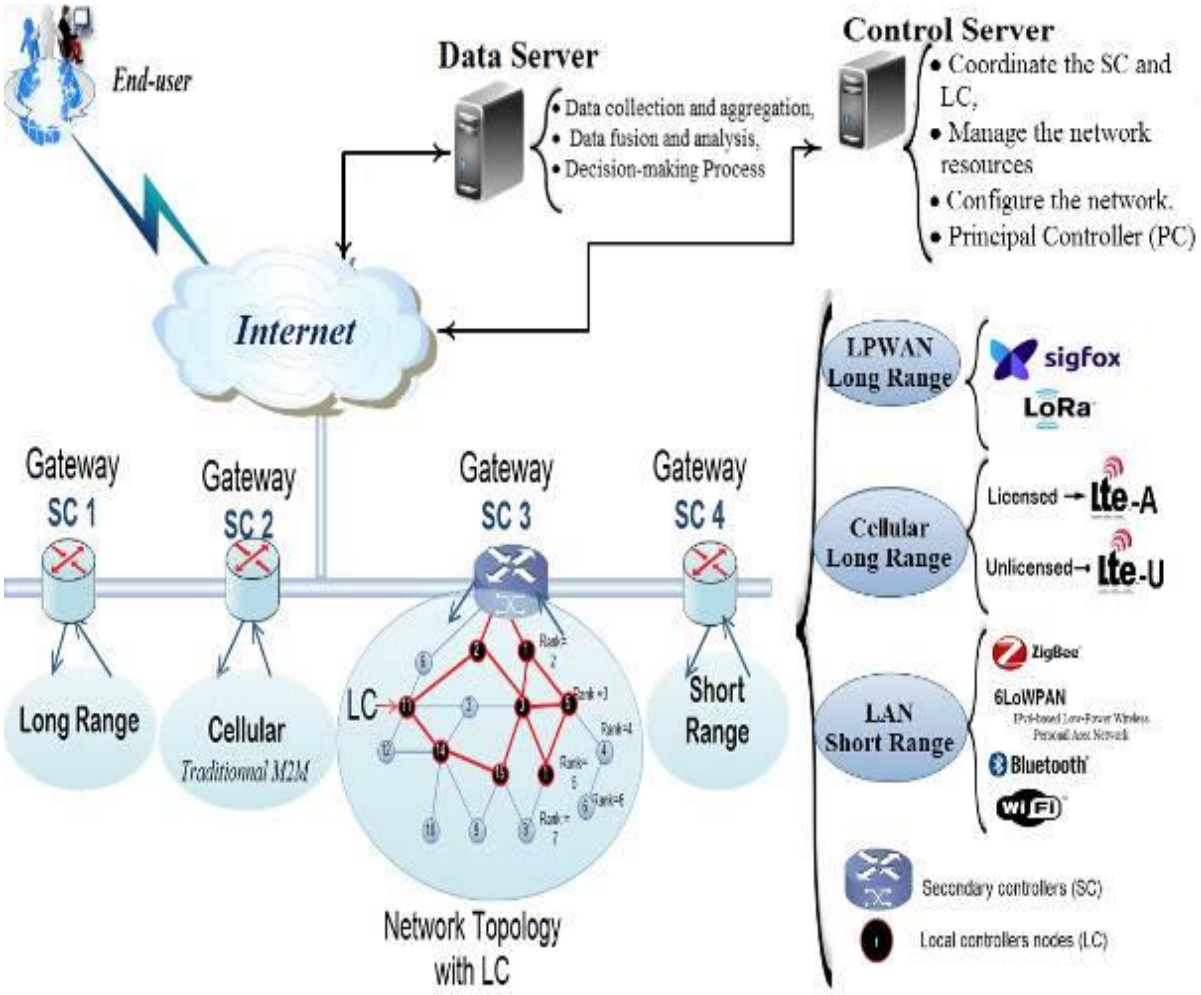


Figure 4.5 : L'architecture hybride et proactive proposée basée sur le paradigme *SDN* [10].

4.4.1.1 Module de contrôle

Dans le module de contrôle, on distingue trois types de contrôleurs : (1) le contrôleur principal (*PC*), (2) les contrôleurs secondaires (*SC*) et (3) les contrôleurs locaux (*LC*).

- **Le contrôleur principal (*PC*)**

Le *CP* est un contrôleur de réseau centralisé, qui représente le premier niveau de contrôle du réseau. Le *CP* est situé dans l'infrastructure du réseau et il a une vue globale sur l'ensemble du réseau en termes d'architecture/topologie et de différents paramètres de réseaux. Les principaux rôles du *CP* sont les suivants : coordonner les autres contrôleurs (*CS* et *CL*), gérer les ressources réseau tout en

tenant compte de l'hétérogénéité du réseau, et configurer le réseau. Sur la figure 4.5, le *CP* est présenté par le serveur de contrôle.

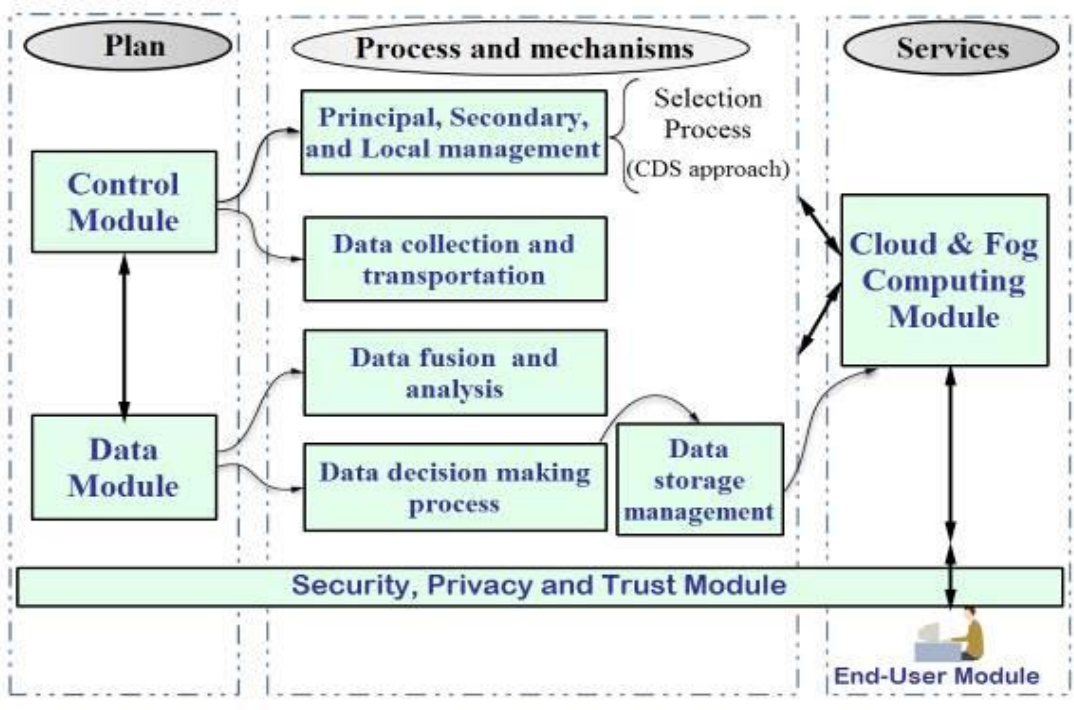


Figure 4.6 : L'interaction entre les principaux modules de l'architecture proposée [10].

- **Le Contrôleur Secondaire (*CS*)**

Le *CS* est le deuxième niveau du contrôle situé sur les *Routeur Edge (Edge-Router)* ou *passerelle (Gateway)*. Le *SC* agit comme des nœuds intermédiaires entre le *CP* et les Contrôleurs Locaux (*CL*), ce qui signifie que toutes les communications de contrôle entre *CP* et *CL* passent par CS. En raison de leur emplacement stratégique, les *SC* sont des points d'entrée pour répondre aux différentes contraintes liées aux réseaux et aux technologies sans fil. Le *CS* peut jouer un rôle clé dans le cas de réseaux hétérogènes. Par exemple, *CS* peut adapter les stratégies de partage et de gestion des ressources (par exemple, politiques d'ordonnancement, processus de routage) en fonction de différents paramètres telles que : communication courte et à longue portée, avec ou sans contraintes énergétiques, communication sous licence ou sans licence, etc. Etant donné que les contrôleurs secondaires sont proches de l'utilisateur final ou des nœuds de capteurs, cela en fait un bon candidat pour héberger le *fog-computing*.

- **Le Contrôleur Local (*CL*)**

Les nœuds *CL*s (*Local-Controllers (LC)*) sont des nœuds spécifiques d'accès aux réseaux et leurs rôles consistent à relayer et gérer les messages de contrôle. Par exemple, les messages de contrôle peuvent être envoyés par des nœuds agissant en tant que routeur avec ou sans mobilité [181]. L'utilisation de ces nœuds pour envoyer des paquets de contrôle et pour communiquer avec la passerelle (nœuds SC*)* contribue à réduire les trafics de contrôle (*overheads*) liés aux messages de contrôle et de signalisation. La sélection des nœuds *CL* comme illustré sur la figure 4.6, est basée sur plusieurs paramètres tels que : le degré de connectivité (*Deg*), la qualité de la liaison avec les nœuds voisins (*Link Quality Indicator (LQI*)) et leur position dans la topologie du réseau (*Rank*). Dans l'architecture proposée, nous utilisons l'algorithme des ensemble de dominants connectés (*CDS (Connected Dominating Set*)) pour sélectionner et choisir ces nœuds *CL*. L'algorithme de sélection est détaillé dans la section ci-dessous.

4.4.1.2 Module de données

Contrairement au module de contrôle, le module de données est responsable de la gestion du plan de données, selon le paradigme *SDN* et sans effectuer le contrôle ni la gestion du réseau. Cependant, le lien fort et la synergie entre les modules de données et de contrôle (en particulier le contrôleur principal) sont nécessaires pour gérer efficacement les services au niveau de la couche application. Les principaux rôles sont orientés vers les données telles que : l'agrégation de données, la collecte de données, l'analyse et la fusion de données, la prise de décision et les notifications. Tous ces rôles peuvent être assurés par le serveur de données comme illustré par la figure. 4.6.

- **Collecte et agrégation de données**

De nombreuses applications dans *IdO* requièrent la collecte et l'agrégation de données à partir de nœuds. Dans notre architecture, la stratégie de collecte et d'agrégation de données est basée sur les contraintes de réseau et de communication sans fil. Par exemple, il est important de prendre en compte la durée de vie du réseau en réduisant la consommation des ressources des nœuds. D'autres caractéristiques de réseau doivent être prises en compte : bande passante, temps système, latence et tolérance aux pannes.

- **Fusion de données et analyse**

L'analyse des données est un ensemble de techniques et de méthodes permettant d'extraire les informations pertinentes de différents nœuds source. Dans le cas de la fusion de données, les techniques utilisées consistent à combiner des données provenant de différents nœuds pour extraire

des inférences et obtenir plus d'informations que si elles provenaient d'un seul nœud (source). Dans cette architecture, le processus d'analyse des données sélectionne la technique adaptée aux contraintes d'application en termes de qualité de service (*QdS*) et de consommation d'énergie comme la sensibilité au retard pour une application en temps réel.

- **Processus de prise de décision**

Ce processus est un point clé de l'architecture proposée, il repose sur des mécanismes antérieurs (agrégation/collecte de données, et fusion/analyse de données) et des outils adaptés pour prendre une décision pertinente (voir figure 4.6). Parmi les outils, divers et prometteurs, qui peuvent être utilisés, nous citons des algorithmes d'apprentissage et des modèles tels que : processus de décision de *Markov MDP* (*Markov Decision Process*), *R-learning*, et *Q-learning* [182]. Dans notre architecture, le module contrôleur peut interagir directement ou indirectement avec ce processus (processus de décision) pour prendre des décisions pertinentes telles que reprogrammer le réseau. L'idée derrière la reprogrammation du réseau en utilisant le paradigme *SDN* est d'adapter les processus de contrôle aux contraintes applicatives.

4.4.1.3 Module Cloud et Fog-Computing

Ce module représente l'intégration des concepts de cloud et celui du *Fog-Computing* dans notre architecture. Le *Fog-Computing* est un ensemble de serveurs connectés situés sur un réseau de base capable d'offrir différents services allant de logiciel en tant que service (*Software as a Services* (*SaaS*)) à Infrastructure en tant que service *(Infrastructure as a Services* (*IaaS*)). Cependant, le brouillard informatique est considéré comme une extension du *Cloud-Computing* à partir du cœur du réseau jusqu'aux nœuds routeurs (*Edge-Router*). Le *Fog-Computing* est hébergé sur les nœuds de *Edge-Router* avec des Contrôleurs Secondaires (*CS*) afin de rendre les services proches des utilisateurs finaux. Comme décrit ci-dessus, l'architecture prend en compte différentes caractéristiques : l'hétérogénéité des technologies de réseau et de communication, l'approche non-centralisée, l'interopérabilité flexible et l'évolutivité. C'est pourquoi, le *Fog-Computing* est un bon candidat pour rendre des services plus fiables et plus efficaces.

4.4.1.4 Module de sécurité et de confidentialité

Ce module est responsable de la gestion dynamique des services de sécurité et de confidentialité. Contrairement aux architectures existantes, l'architecture proposée prend en considération les services de sécurité dynamiques (confidentialité, intégrité, authentification) avec différents niveaux de sécurité capables de reprogrammer automatiquement la politique de sécurité du réseau. C'est pourquoi

l'interaction avec le module contrôleur comme le montre la figure 4.6 est nécessaire pour rendre la détection des attaques plus efficace et la réaction aux attaques potentielles appropriée.

Par exemple, les pare-feu sur les nœuds périphériques (*contrôleurs CS*) peuvent être réglés dynamiquement en ajoutant ou en supprimant des règles grâce au paradigme *SDN*. De plus, le modèle de confiance est requis et il doit être dynamique pour suivre les comportements des nœuds (dispositifs) [182,183, 184]. Par exemple, (*introduisez le concept Block Chain*) dans le cas de la gestion de confiance distribuée [185].

4.4.1.5 Module utilisateur final

Les applications *IdO* ont plusieurs profils et les exigences hétérogènes des utilisateurs finaux doivent être prises en compte dans la conception de toute architecture logicielle. Par exemple, l'utilisateur final peut être une personne, une machine (matériel) ou un programme (logiciel) avec différentes exigences et capacités. Dans le cas de cette architecture basée sur le paradigme *SDN*, non seulement nous distinguons les différents profils d'utilisateurs finaux, mais nous prenons également en considération l'évolution de leurs exigences, et ce, pour rendre l'utilisateur final actif. Ce module présente différentes méthodes et outils permettant aux utilisateurs de comprendre, configurer, personnaliser et contrôler les applications.

4.4.2 Méthode de sélection des *CLs* avec l'algorithme de construction des *CDS*

Dans cette section, nous nous intéressons à la partie du réseau située derrière les passerelles (Contrôleurs Secondaires) et en particulier à la stratégie de sélection des Contrôleurs Locaux (*CL*). Comme décrit ci-dessus, les nœuds *CL* ont un rôle important à jouer pour réduire la charge du réseau lié aux messages ou aux trafics de contrôle et autres paquets de signalisation.

Nous décrivons l'algorithme de construction des *CDS* (*Connected Dominating Sets*) utilisé pour sélectionner un sous-ensemble de nœuds qui va agir comme *CL*. En outre, nous présentons le modèle de calcul de la fonction « *Score* » basé sur l'approche de l'ensemble flou pour rendre la sélection de la stratégie adaptable à des différents scénarios.

Nous utilisons l'approche ***DSP*** (***D****istributed* ***S****ingle-****P****hase*) pour l'algorithme de construction de *CDS*. Cet algorithme donne de meilleures performances en termes de délai et de taille des *CDS* [8]. L'algorithme commence en donnant une couleur blanche à tous les nœuds, puis en fonction de la stratégie utilisée, certains nœuds se transforment en noir. Seuls des nœuds noirs créent un *CDS* et agissent en tant que contrôleur local (*CL*).

La stratégie de sélection de notre solution proposée, appelée ***DLC-CDS*** (***D****ominated* ***L****ocal* ***C****ontroller* ***C****onnected* ***D****ominating* ***S****et*) [10], repose sur plusieurs paramètres tels que : le degré de

connectivité (***Deg***), la qualité moyenne de liaison entre les voisins ($\overline{LQI}$) et le *rang* (Rank) avec une distance (en termes de numéro de saut) par rapport au Contrôleur Secondaire (*CS*).

Ces paramètres sont regroupés dans un paramètre principal nommé " "*Score* " qui est discuté dans la section suivante. Cet algorithme suppose que chaque nœud a un identifiant unique (***NodId***) et qu'un sous-ensemble de nœuds connectés pendant la construction *CDS* a un *ID* unique (*SetID*). Trois couleurs sont utilisées pour définir l'état des nœuds : la couleur blanche (pour les nœuds non dominants), grise (étape intermédiaire) et noir (pour nœuds dominants). Par ailleurs, nous distinguons deux étapes principales :

- **L'étape initiale :** Cette étape consiste à identifier et initialiser l'état des nœuds. De plus, cette étape lance le processus de calcul de **Score** afin de donner un poids et une importance différents aux nœuds en fonction de plusieurs paramètres, nous allons détailler cette étape dans la sous-section suivante.
- **L'étape de la compétition** : Elle présente le processus de décision où les nœuds sont colorés en blanc ou en noir. L'organigramme de la figure 4.7 représente l'exécution du processus de notre algorithme *DLC-CDS* proposé. Cette représentation schématique illustre un modèle de solution de la stratégie de sélection des nœuds *CLs* dans le module de contrôle.

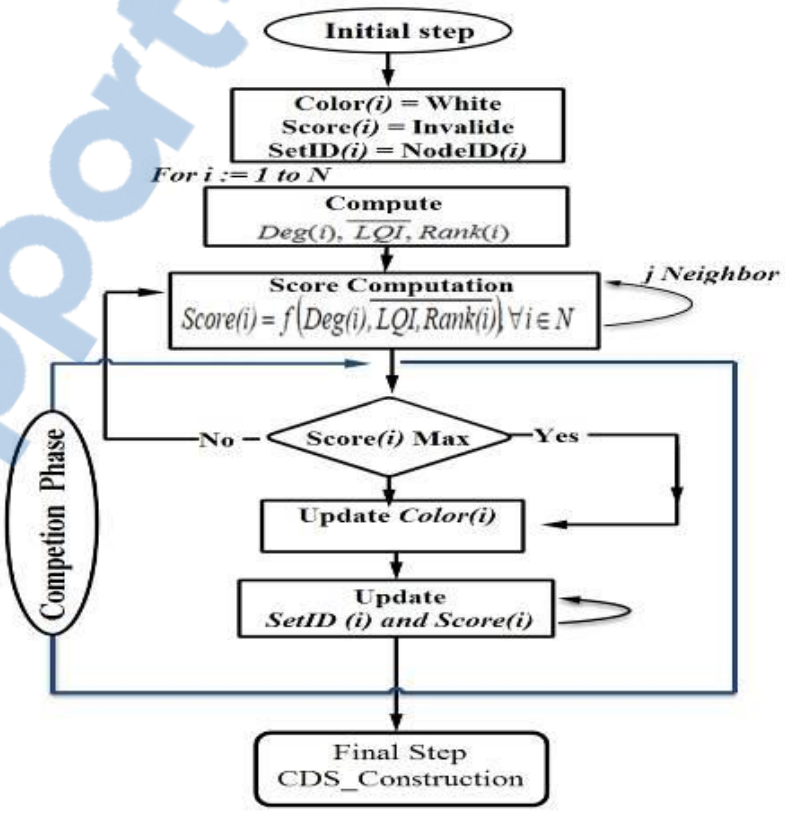


Figure 4.7 : L'organigramme de l'exécution du processus *DLC-CDS* [10].

4.4.2.1 Étape d'initialisation du *DLC-CDS*

Cette étape selon l'algorithme 1, considère que tous les nœuds de la topologie du réseau sont blancs et que leur valeur de ***Score*** est invalide (non disponible). De plus, à cette étape, le nombre de sous-ensembles réseau est égal au même nombre de nœuds dans le réseau. Par conséquent, l'identité de chaque sous-ensemble (*SetID*) est la même sur chaque nœud (*NodID*).

Algorithm 1: Initial step: All nodes are white

Input : Score,*i*, *N*
Output: NodID, SetID, Color

```
1 N is a set of nodes
2 foreach i ∈ N do
3 |   i.node ← NodID
4 |   Net ← SetID
5 |   if (i.Score ← InvalidScore) then
6 |   |   i.Color ← white
7 |   |   SetID ← NodID
8 |   end
9 end
```

4.4.2.2 Étape d'évaluation des paramètres réseau

Dans cette étape, chaque nœud calcule et évalue au moins trois paramètres réseau importants : (1) le degré de connectivité (*Deg*) qui représente le nombre de nœuds voisins directs, l'indice de qualité de liens moyen ($\overline{LQI}$) avec ces nœuds voisins et le R*ang* (*Rank*) qui représente la distance (en termes de nombre de sauts) d'un nœud para apport au nœud Edge routeur ou passerelle. Selon ces paramètres de réseau, le paramètre « *Score* » est calculé en utilisant le modèle de la Logique Flou (*LF*) [186] [187] proposé et qui sera détaillé dans la sous-section suivante.

4.4.2.3 Étape de la concurrence et de la prise de décision

Dans cette étape et selon l'organigramme d'exécution du processus *DLC-CDS* illustré par la figure 4.7, chaque nœud calcule son *Score* en fonction des paramètres du réseau calculés et évalués dans l'étape précédente. Afin de prendre des décisions sur l'état du nœud, nous distinguons deux algorithmes principaux : Algorithme 2 et Algorithme 3 qui s'exécutent respectivement par le nœud expéditeur et nœuds récepteurs. Dans le cas du nœud émetteur selon l'algorithme 2, après avoir calculé son *Score* et après l'avoir comparé avec celui des nœuds voisins non noirs, il prend une décision sur son statut de couleur. Si la couleur devient noire, elle transmet le message à ses nœuds

voisins avec la mise à jour de ces paramètres : *NodID*, *SetID*, *Couleur* et *Score*. Dans le cas d'un nœud récepteur noir ou gris, et s'il reçoit un paquet de nœuds noirs avec un plus grand *SetID*, alors le récepteur met à jour son *SetID* (qui appartient au même sous-ensemble avec le récepteur comme chef. Cependant, dans le cas du nœud récepteur blanc, il met à jour son *SetID* et change sa couleur en gris comme décrit dans l'algorithme 3.

Algorithm 2: Competition step at sender node *i*

Input : Score,NodID, SetID, i, neighbors(i)
Output: State decision of change Color nodes

```
1  foreach node i ∈ N do
2      Score(i) ← ComputeScore()
3      if (Score(i) ≠ 0) and (i.Color ≠ black) then
4          foreach k ∈ non-black-neighbor(i) do
5              S ← MaxScore(k)
6              if (Score(i) > S) then
7                  i.Color ← black
8                  SetID = NodID
9                  Broadcast(SetID, NodID, Score)
10             end
11         end
12     end
13 end
```

Algorithm 3: Competition step at receiver node *j*

Input : SetID, NodID, Color, j
Output: A connected dominated set with black node

```
1  foreach Node j receives packet from black node i do
2      if ((j.Color = black)OR(j.Color = gray))and(i.SetID >
       j.SetID) then
3          j.SetID = i.SetID
4      else
5          if (j.Color = white) then
6              j.SetID = i.SetID
7              j.Color ← gray
8          end
9      end
10 end
```

4.4.3 Méthode de calcul du Score utilisé par *DLC-CDS*

Cette sous-section présente le modèle de calcul du *Score* proposé et qui est basé sur l'approche de l'ensemble flou [187]. Le score est un paramètre important pour créer le *CDS* et pour sélectionner les nœuds *CL*. Cela dépend de trois paramètres clés : (1) *Deg (i)*, (2) $\overline{LQI}(i)$ et (3) le *Rang (i)*. Cependant, la fonction appropriée *f*, représentée par la formule 4.4 avec une capacité de combiner ces paramètres est nécessaire d'être introduite dans ce modèle.

$$Score(i) = f\left(Deg(i), \overline{LQI(\imath)}, Rank(i)\right);\ \forall\ i\ \in N \qquad (4.4).$$

En effet, nous introduisons le contrôleur de la logique flou afin de combiner ces paramètres et de calculer le *Score* selon la formule 4.4. A cet effet, deux étapes principales sont nécessaires pour développer le contrôleur de la LF : (1) l'étape qui définit les fonctions d'appartenance pour chaque paramètre d'entrée/sortie, (2) la conception des règles floues. Cependant, Nous concevons un nouveau système illustré par la figure.4.8, qui a trois fonctions d'appartenance d'entrée *Deg (i)*, $\overline{LQI}(i)$ et *Rank (i)*, et la fonction d'appartenance de sortie de *Score (i)*. Les fonctions d'appartenance d'entrée et de sortie prennent trois valeurs linguistiques : *Minimum*, *Moyenne* et *Maximum* comme présenté dans le tableau 4.1.

Afin d'obtenir la valeur maximum pour le *Score* d'un nœud, nous considérons la topologie du réseau avec le degré de connectivité (*Deg*) maximal, la performance de liaison avec l'indice de qualité de liaison ($\overline{LQI}$) maximum et la distance par rapport à la passerelle avec le *Rang* minimum. Ces fonctions d'appartenance (voir ci-dessous) définissent les ensembles flous à l'entrer avec des règles générales d'inférence. Ces fonctions sont considérées comme des facteurs de pondération pour déterminer leur influence sur les ensembles de sortie.

- **Les fonctions d'appartenance de la fuzzification**

Dans la solution proposée, le contrôleur de la logique floue [186] utilise les trois paramètres d'entrée : *Deg(i)*, $\overline{LQI}(i)$ et *Rank (i)* comme décrit et calculé dans la sous-section suivante. Nous avons utilisé la fonction d'appartenance *trapez trapmf* pour les paramètres d'entrée, définis comme suit :

1) ***Deg (i)* :** C'est l'entrée n° 1 de FL, qui est le nombre de nœuds voisins. Il doit être maximisé pour satisfaire le choix du nœud CL dans la topologie du réseau.
2) **$\overline{LQI}(i)$:** La fonction d'appartenance de l'indice de qualité de liens est une mesure de la qualité de lien de la trame reçue telle que définie par la norme IEEE 802.15.4 et le calcul relatif de la valeur $\overline{LQI}$ est le *RSSI* (*Received Signal Strength Indicator*) valeur.

3) **Le *Rang (i)* :** La fonction d'appartenance pour le paramètre Rang (*Rank*) est calculée via le rang du parent qui est obtenu dans le cadre des informations concernant les nœuds voisins.

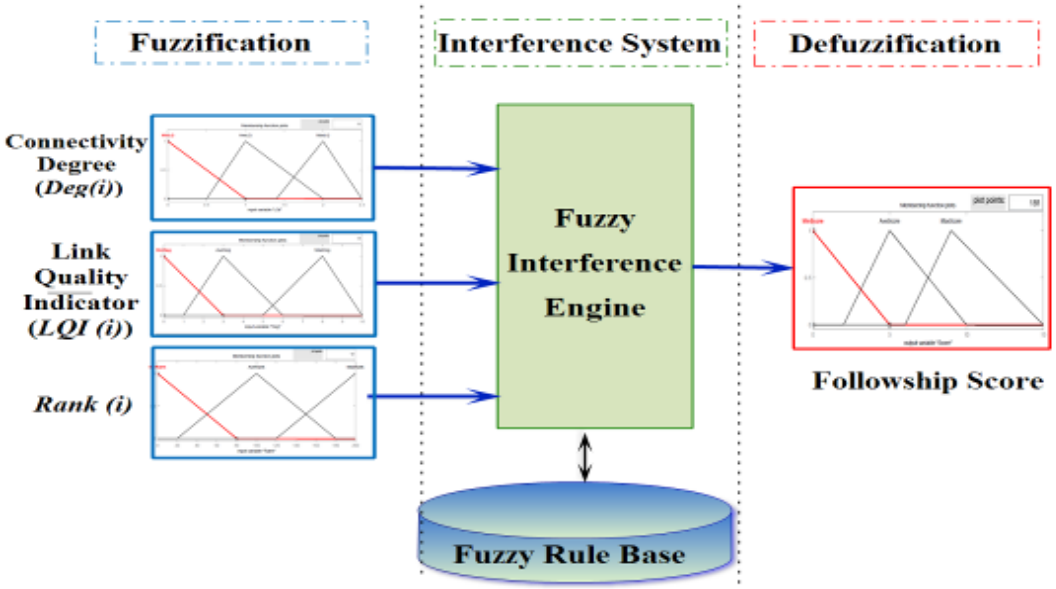


Figure 4.8 : L'approche de la logique floue pour le score de calcul : 3 entrées, 1sortie.

Entré et Sortie de la fonction d'appartenance Fuzzification	**Variable linguistiques**
Entré 1 : Degré de connectivité (*Deg*)	Maximum (*MaxDeg)* Average (*AvgDegr*) Minimum (*MinDegr*)
Entré 1 : Indicateur de qualité de lien moyen ($\overline{LQI}$)	Maximum (*MaxLQI*) Average (*AvgLQI*) Minimum (*MinLQI*)
Entré 1 : la position du nœud (*Rang*)	Maximum (*MaxRank*) Average (*AvgRank*) Minimum (*MinRank*)
Sortie : le *Score*	Maximum (*MaxScore*) Average (*AvgScore*) Minimum (MinScore)

Tableau 4.1 : Les valeurs d'appartenance d'entré et sorties de la logique flou.

- **Le système d'inférence**

Le moteur d'inférence floue permet d'évaluer les règles de contrôle stockées dans la base de règle floue. Nous avons défini vingt-sept (27) règles en utilisant la méthode *Centroide* [187] pour traiter la fonction d'appartenance en sortie du *Score (i)*. Le principe de la règle floue est d'exprimer la connaissance avec les instructions conditionnelles Si -Alors (-Sinon), par exemple :

Si ((*Deg (i)*) est de valeur maximum ***ET*** $\overline{LQI}$*(i)* est de valeur maximum ***ET*** (*Rank (i)* est de valeur minimum)) ***ALORS*** le score du nœud (*Score (i)* est maximum.

- **La sortie Flowship Score-Defuzzification**

Pour obtenir un nombre fini de la fonction *Score (i)*, nous devons passer par le processus de D*éfuzzification*, il y a plusieurs façons de le faire comme détaillé dans [186]. Le plus commun est la méthode gaussienne, utilisée dans notre cas d'évaluation du *Score*. Dans la dernière étape de la *FL*, la valeur de sortie de la fonction d'appartenance des résultats de *Défuzzification*, nous obtiendrons une valeur maximale du *Score (i)*, qui est utilisée dans l'algorithme de construction des CDS et qui est présentée dans la sous-section 4.3.4 pp 127. Sachant que la valeur maximum du *Score* indique le meilleur choix du nœud dominateur (nœud noir).

4.4.4 Implémentation et évaluation des performances

Dans cette section, nous effectuons des études numériques dans deux parties de simulations pour la stratégie de sélection des *CL* avec l'algorithme *DLC-CDS*. La première partie, permet d'évaluer la fonction du score avec l'approche de la logique floue et dans la deuxième partie, nous évaluons la performance de *DLC-CDS* sous différents paramètres.

On décrit aussi l'environnement de simulation utilisé pour la mise en œuvre du modèle de contrôle proposé. Nous présentons par la suite les paramètres de simulation ainsi qu'une analyse comparative des différents résultats obtenus.

4.4.4.1 Première partie de simulation : Evaluation de performance du *Score*

Dans cette première partie de simulation, nous présentons l'évaluation du calcul du *Score* avec l'approche de la logique floue. Afin d'évaluer l'impact des paramètres d'entrée sur la fonction du score, nous les sont présentons dans différents scénarii tels que décrits dans le tableau 4.2 et décrits dans la sous-section ci-dessous.

Règles Entrés/Sorties	***Deg***	$\overline{LQI}$	***Rang***	***Score***
Scenario 1	Moyen	Minimum	Minimum	Minimum
Scenario 2	Maximum	Minimum	Moyen	Moyen
Scenario 3	Maximum	Maximum	Minimum	Maximum

Tableau 4.2 : Scénarios de la *LF* pour le calcul du Score de réseau.

A) Paramètres de simulation de l'approche Logique Floue

Pour évaluer la fonction du *Score* comme présenté avec la formule 4.4, nous avons utilisé le simulateur *MATLAB*. Les paramètres de simulation de cette partie sont décrits comme suit :

Les nœuds sont déployés de manière aléatoire dans un réseau contenant 49 nœuds et un nœud de passerelle. Tous ces nœuds sont regroupés dans un réseau *100 × 100*. La plage radio *R = 10* est utilisée dans toutes les simulations et pour chaque nœud, avec une densité de nœuds *p = 0,04* pour la même configuration de la topologie du réseau.

Dans cette simulation, nous calculons en premier temps les trois paramètres de : *Deg (i)*, $\overline{LQI}(i)$ et *Rang (i)*. Ensuite, ces paramètres sont exploités dans la l'étape de la *LF* en tant que paramètres d'entrée dans le système flou qui va être présenté dans la sous-section suivante. Nous avons utilisé les paramètres obtenus de *Deg (i)*, $\overline{LQI}(i)$ et de *Rang (i)*, comme une fonction d'appartenance à trois entrées pour évaluer la fonction *Score (i)*.

Les variables d'entrée et de sortie prennent trois valeurs linguistiques : maximum, moyen et minimum. Nous avons utilisé trois scénarii, qui sont résumés dans le tableau 4.2, comme référence d'évaluation parmi les vingt-sept règles implémentées dans le contrôleur *FL* (voir figure 4.8). Pour chaque scenario, comme décrit dans le tableau 4.2, les paramètres *Deg (i)*, $\overline{\text{LQI}}$(i) et *Rang (i)* vont prendre les valeurs linguistiques *Maximum* et *Moyenne*. L'objectif principal de ces scénarios est d'évaluer le Score et d'obtenir une valeur finie de la fonction score (i).

B) Performance d'évaluation de la fonction Score

Les résultats de simulation obtenus sont représentés sur la figure 4.9 qui montre le comportement de la fonction du *Score* avec différents paramètres (*Deg (i)*, $\overline{LQI(\imath)}$ et le *Rang (i)*).

Nous remarquons que la valeur maximale du score est atteinte lorsque les deux paramètres du degré de connectivité (*Deg*) et l'indicateur de qualité de liens ($\overline{LQI}$) sont variés vers les

valeurs maximales, et avec la valeur du *Rang* fixé minimum, sinon la valeur minimum et la valeur moyenne dépendent de la valeur minimum du $\overline{LQI}$, de la valeur moyenne du *Deg* et minimum de celle du *Rang*. Bien que le *Rang* soit de valeur minimale, il convient de noter qu'il y a un impact du degré de connectivité (*Deg*) sur l'indicateur de la qualité de liens ($\overline{LQI}$).

Dans cette phase de simulation, le *Score* augmente pour prendre les valeurs maximales dans l'ensemble des valeurs {8, 9, 10} et le *MaxScore (i)* équivaut à *10*. Pour une valeur maximale du *Score*, il est très intéressant de choisir un nœud *CL* avec plus de nœuds voisins (*MaxDeg (i)*) pour le processus de contrôle, avec une meilleure qualité de lien et à proximité du nœud de passerelle (*CS*). Alors forcément, les résultats de ce scénario s'avèrent être de bons résultats en termes de sélection du contrôleur local avec les valeurs maximums du *Score* obtenues dans la plage des valeurs de : 8, 9, 10.

Dans la dernière étape, nous avons introduit la valeur maximale du score dans l'algorithme *DLC-CDS* illustré à travers l'organigramme de la figure 4.7, pp 137 et ce, afin de poursuivre l'exécution de ces étapes et effectuer une simulation étendue du réseau pour la création de nœud *CL*, qui est présentée dans la sous-section suivante.

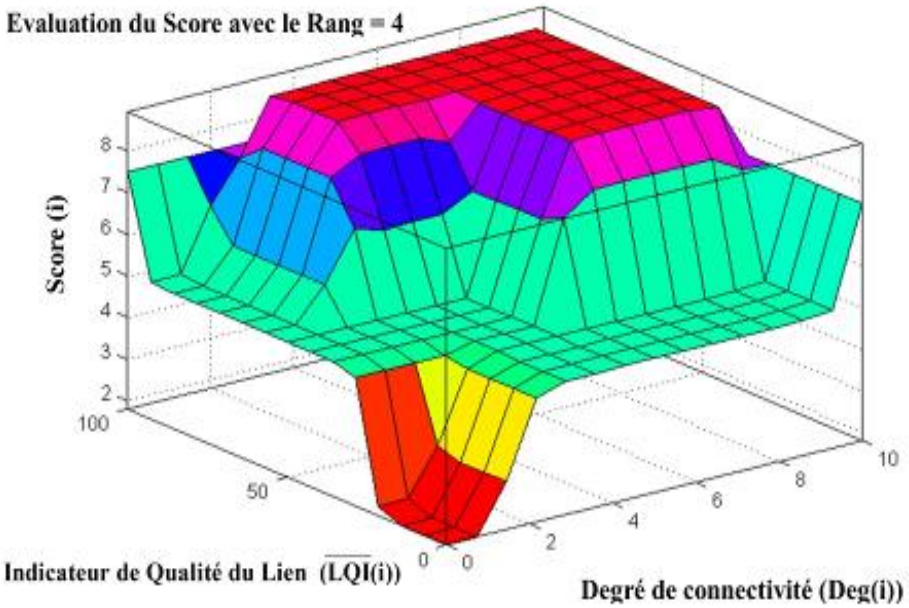


Figure 4.9 : Résultat de calcul du Score avec la logique floue.

4.4.4.2 Deuxième partie de simulation : Les scénarios et évaluation de performance

Dans cette partie, nous avons exécuté autant de simulations nécessaires pour évaluer les performances de l'algorithme distribué à une seul phase *DLC-CDS* sous différents paramètres. Une comparaison de la solution proposée de *DLC-CDS* avec l'algorithme *DSP-CDS* [8] est également effectuée dans cette sous-section en utilisant le simulateur Matlab.

A) Description des Scenarios de simulation

La simulation de cette phase modélise un réseau avec *N* nœuds déployés aléatoirement dans une zone carrée de longueur *L* variant de *40 m* à *120 m* et la portée radio *R* supposée égale à *R = 10* pour chaque nœud et qui sera variée dans les différents scénarii de simulation.

Pour l'obtention d'un certain nombre de nœuds dans le réseau, défini par *N*, ou $N = L \times L \times \rho$, *et* ρ, la densité de nœuds telle qu'elle permet de définir le nombre moyen de nœuds dans une unité de surface. Les scénarii de simulation ainsi que les paramètres sont présentés dans le tableau 4.3. Nous avons pris en considération quatre scénarii différents (Tableau 4.3) avec des paramètres de simulation qui vont être décrits par la suite de cette section afin d'évaluer l'algorithme *DLC-CDS*.

La comparaison entre l'algorithme *DSP-CDS* [8] et notre solution proposée de *DLC-CDS* [10] est présentée dans cette sous-section.

- **Scénario 1 :** Dans ce scénario, nous avons effectué une comparaison entre les méthodes *DLC-CDS* et *DSP-CDS* afin d'évaluer la métrique de performance *CDS-Size*. Pour cette simulation, nous avons pris en considération le réseau avec la variation des nœuds de 48 à 576 et utilisé la portée radio fixe à *R = 10*. Nous avons fait varier la densité de nœuds (ρ) avec des valeurs de 0.03 et 0.04.

- **Scénario 2 :** Dans ce scénario, nous avons utilisé la même configuration du réseau que celle introduite dans le scénario 1. Nous avons fixé la densité des nœuds $\rho = 0{,}04$ et varié la portée radio de *R = 10* et *R = 18*. Une autre évaluation des performances, à travers la comparaison de la métrique CDS-Size (La taille de la CDS) est étudiée dans ce scénario entre le *DLC-CDS* et le *DSP-CDS*.

- **Scénario 3 :** Dans ce scénario, nous évaluons notre solution proposée de l'algorithme *DLC-CDS,* tout en variant la densité de nœuds ρ de 0,02 à 0,09 et en fixant la portée radio *R = 10*.

- **Scénario 4 :** Dans ce cas de simulation, la configuration du réseau est fixée à un nombre de nœuds *N = 864* nœuds et la portée radio de 10 à 18. Dans ce scénario, nous évaluons les

performances de *DLC-CDS* en termes de métrique de performance *CDS-Size* sous l'impact de la densité ρ variée de 0,02 à 0,06.

Scénarios Du réseau	***Nombre de nœuds (N)***	***Densité du nœud (ρ)***	***Longueur du réseau (L)***	***Portée de la Radio (R)***
Scenario 1	48 à 576	0.03,0.04	40 à 120	10
Scenario 2	64 à 576	0.04	40 à 120	10, 18
Scenario 3	32 à 1296	0.02 à 0.09	40 à 120	10
Scenario 4	864	0.02 à 0.06	120	10 à 20

Tableau 4.3 : Paramètres de simulation de DLC-CDS.

B) Paramètres de simulation de DLC-CDS

Nous avons utilisé deux paramètres importants de la simulation du *DLC-CDS*, la densité du nœud (ρ) et la portée radio (*R*) comme décrit ci-dessous.

- **Densité de nœud (ρ) :** C'est un paramètre important influençant la connectivité du réseau, il détermine le nombre de nœuds voisins dans la portée de transmission du nœud. Afin d'augmenter la densité des nœuds dans le réseau, nous ciblons la densité du réseau. Chaque nœud de la compétition peut décider de devenir un dominateur (*CL*), s'il a un nombre maximum de voisins avec la densité de nœuds la plus élevée dans la zone du réseau. D'un autre côté, le nœud ayant la valeur maximum a plus de chance de devenir un dominateur pour couvrir plus de nœuds voisins et partager le même *SetID*. Chaque nouveau dominateur essaiera de couvrir une nouvelle zone du réseau avec une portée radio fixe *R*. Pour cette raison, la taille du *CDS* (*CDS-Size*) doit être principalement déterminée par la taille du réseau et dépend de la densité du nœud.
- **Porté radio (R) :** Dans notre algorithme proposé (*DLC-CDS*), nous introduisons la portée radio comme un deuxième paramètre important. Ce paramètre a un effet sur la densité du nœud. En effet, si la portée radio pour chaque nœud augmente, le nœud peut avoir plus de voisins capable d'augmenter leur degré de connectivité. Dans notre simulation, nous changeons la portée radio pour évaluer la performance de l'algorithme *DLC-CDS*. Partie d'expérimentation, nous proposons de changer la portée radio dans les scénarios de simulation 2 et 4, afin de cibler le degré de connectivité des nœuds. La densité relative du nœud (*DR*n) est définie par la formule 4.5, lorsque la portée radio est fixe, par exemple si elle est à 18 et si la densité de nœuds ρ = 0,01, la densité des nœuds relative est égale à 10,17. De cette manière,

la portée radio et la densité des nœuds ont une correspondance un à un et ont un impact sur la taille du CDS dans la zone du réseau. En augmentant la portée radio, nous avons affiné l'évaluation des performances *DLC-CDS* par rapport à l'algorithme *DSP-CDS* en termes de taille CDS.

$$DRn = \pi * R^2 * \rho \qquad (4.5)$$

C) Les métriques de performance

Dans cette sous-section, nous présentons l'évaluation de performance de la méthode proposée du *DLC-CDS* avec les métriques de performances. Les scénarii de comparaison avec les méthodes existantes [9] sont effectués en fonction de la métrique de la taille des *CDS* (*CDS-Size*) selon les différents paramètres présentés dans le tableau 4.3.

Toutes les simulations réalisées dans cette étude (implémentation et évaluation de la performance) montrent l'effet des paramètres de la portée radio (R) et de la densité du nœud (ρ) pour la construction des *CDS* dans le réseau.

Une seule métrique de performance a été utilisée, celle de *CDS-Size* (Taille des *CDS*) afin d'évaluer la performance du *DLC-CDS* et d'établir une comparaison avec le DSP-CDS et les autres algorithmes distribués. Les scénarii de simulation se déroulent en plusieurs itérations (rounds), et à chaque itération le nœud décide de changer de couleur en fonction de la valeur de la partition et de ses nœuds voisins.

- **La métrique de performance, *CDS-Size* :** En ce qui concerne l'ensemble des simulations, nous avons utilisé la métrique d'évaluation de la taille *CDS* pour l'évaluation de la performance de la méthode *DLC-CDS*. La taille des *CDS* est le nombre de nœuds dominants connectés, qui peuvent être construits dans le *CDS* du réseau.

4.4.5 Résultats de simulation et interprétation

Dans cette section, nous évaluons l'algorithme proposé *DLC-CDS* à travers les différents résultats de simulation ainsi que de comparer sa performance en termes de *CDS-Size* para port aux deux algorithmes : *DLC-CDS* et *DSP-CDS* [9].

Les résultats illustrés dans les figures 4.10 (a) et (b) présentent par ordre les résultats du scénario 1 et scénario 2, à l'objectif de comparer les performances des algorithmes *DLC-CDS* et *DSP-CDS* en termes de CDS-Size. Les figures 4.11 et 4.12 présentent en ordre respectif les résultats du scénario 3 et scénario 4, qui montrent l'impact de la densité des nœuds ρ et de la portée radio R sur la performance de *DLC-CDS* en termes de *CDS-Size*.

- **L'impact de la densité des nœuds (ρ)**

La figure 4.10 (a) illustre la comparaison entre les algorithmes *DLC-CDS* et *DSP-CDS*. Ces résultats montrent que le *DLC-CDS* est plus performante que l'algorithme *DSP-CDS* avec une différence de 64% en termes de taille de *CDS*. Cette différence entre les deux algorithmes est causée par la stratégie de sélection des nœuds noirs (Dominants) spécifiés par chaque algorithme. La stratégie de sélection de l'algorithme *DSP-CDS* est basée uniquement sur le paramètre degré de connectivité, qui offre la possibilité à tous les nœuds ayant un degré supérieur de devenir dominant (nœuds noirs). Pour l'algorithme *DLC-CDS*, la technique additionnelle de la logique floue est intégrée : les nœuds avec une performance $\overline{LQI}$ médiocre sont éliminés.

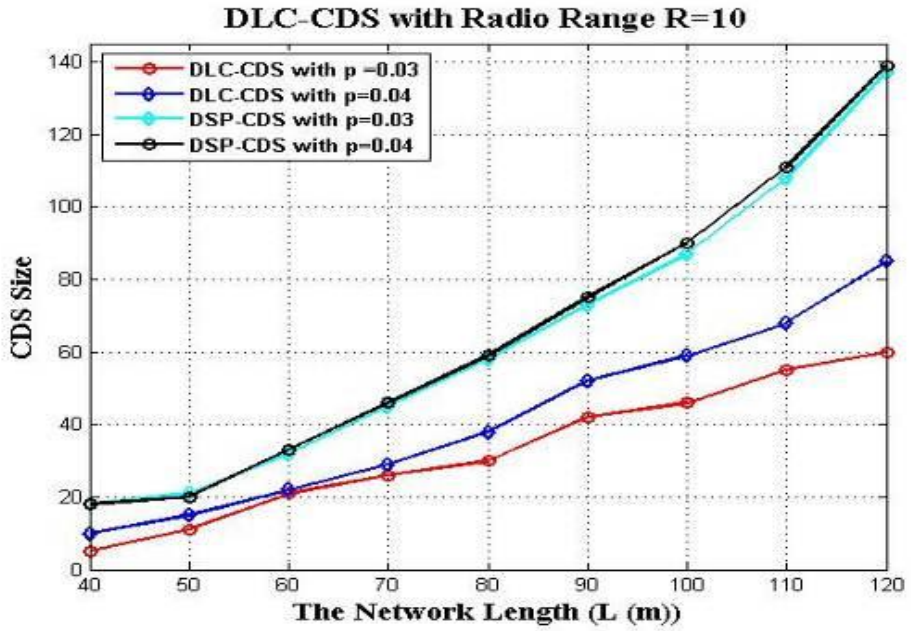
En effet, le $\overline{LQI}$ a un impact sur la stratégie de sélection du dominant dans la zone du réseau. Pour l'algorithme *DLC-CDS*, on remarque que pour la performance de *DLC-CDS*, il y a une différence de 12% de la taille des *CDS* générée dans le réseau avec la variation de la densité du nœud qui équivaut à 0,03 et 0,04). Cela est dû à l'impact de la densité du nœud sur la contrainte $\overline{LQI}$, lorsque la densité du nœud augmente, la probabilité d'une collision augmente aussi. Ceci peut avoir un impact négatif sur la qualité du lien $\overline{LQI}$.

La figure 4.11 illustre de manière nette l'impact de la densité du nœud sur la taille des *CDS*, ainsi, lorsque la densité du nœud augmente la taille des *CDS* augmente. Le résultat de ce scénario présente la densité du réseau lorsque les nœuds voisins augmentent du fait de l'augmentation de la densité de nœuds.

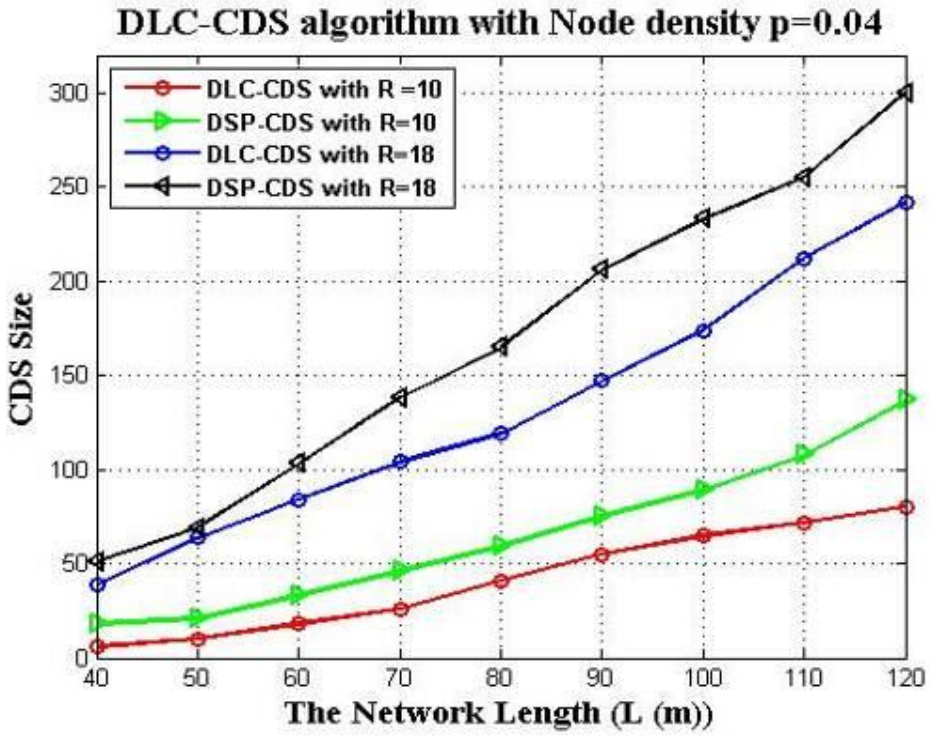
- **L'impact de la portée radio (R)**

Dans les simulations de portée radio, la figure 4.10 (b) et la figure 4.12 montrent que la taille des CDS générée par le *DLC-CDS* est inférieure à celle générée par le *DSP-CDS* avec une différence de 63%. Lorsque la portée radio augmente, il y aura des nœuds avec un degré de connectivité plus élevé (*Deg*), en raison de la densité du voisinage. Pour cette raison, le $\overline{LQI}$ présente le paramètre important qui diminue la taille des CDS avec l'algorithme *DLC-CDS*, lorsque certains nœuds avec $\overline{LQI}$ médiocre sont éliminés.

Lorsque la portée radio est augmentée, il y a plus de voisins, ce qui implique que le $\overline{LQI}$ diminue et par conséquent, on obtient un CDS minimum. La figure 4.10 (b) confirme que, si le $\overline{LQI}$ diminue en raison de l'augmentation de la portée radio et si les ressources de communication peuvent être partagées avec tous les nœuds voisins, il y a plus de collisions qui entraîneront une diminution de performance.



(a) ***Scénario 1 :*** L'impact de la densité des nœuds p.



(b) Scénario 2 : *L'impact de la portée radio (R).*

Figure 4.10 : La taille du CDS en fonction de la longueur du réseau L (m).

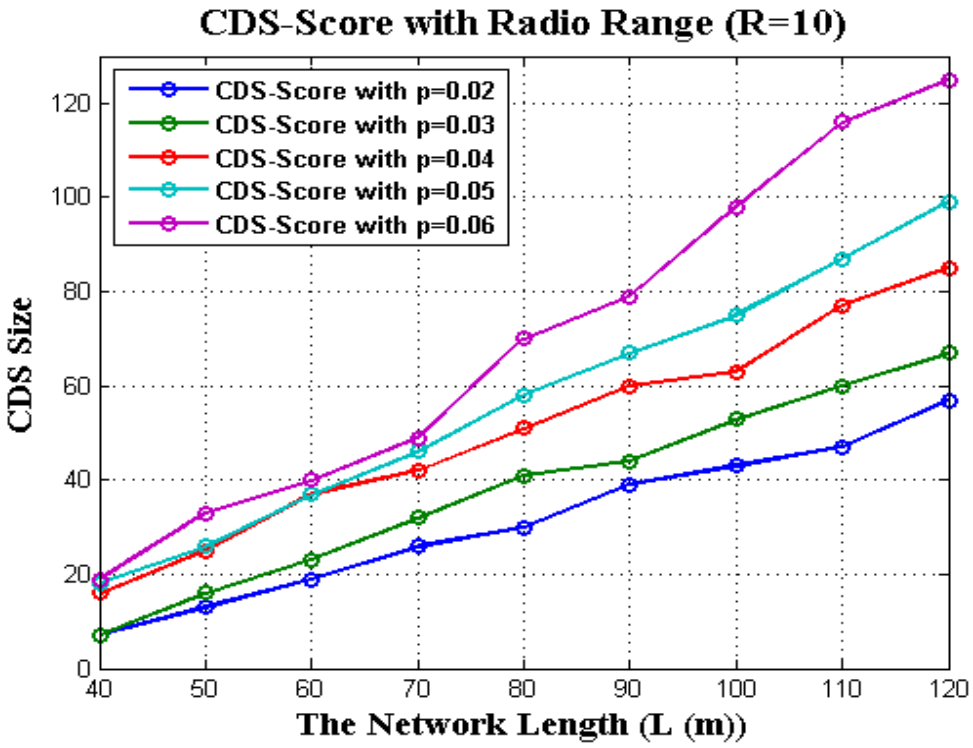


Figure 4.11 : Scenario 3, L'impact de la densité de nœuds ρ sur la taille des CDS générée par l'algorithme DLC-CDS.

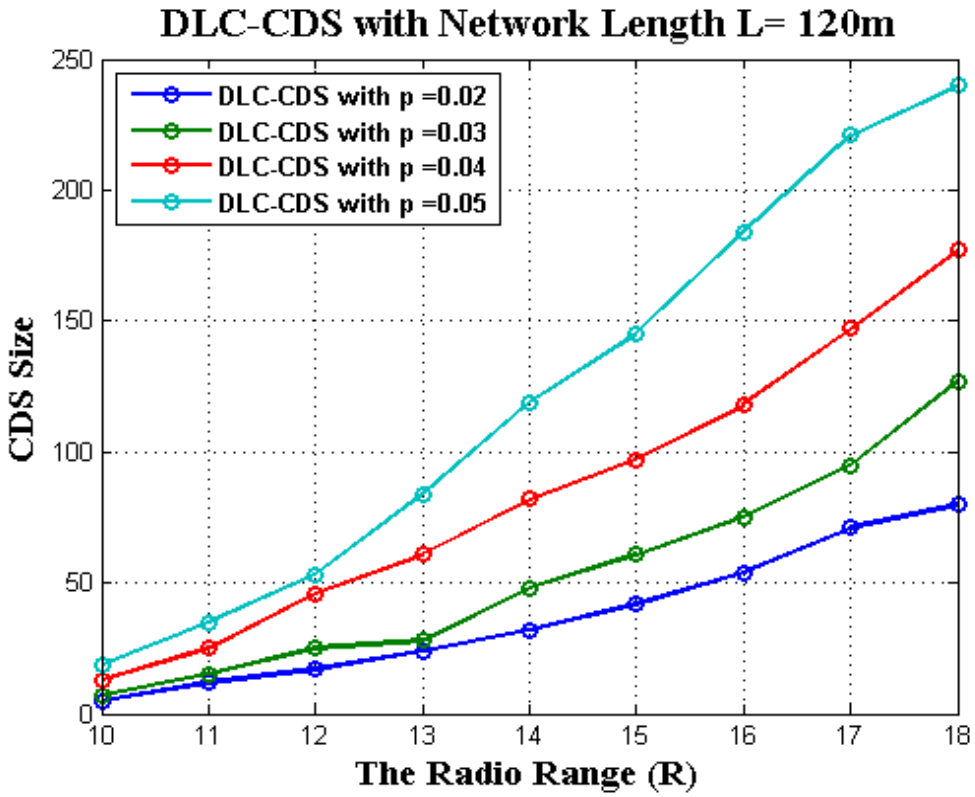


Figure 4.12 : Scenario 4, L'impact de la densité des nœuds ρ sur la taille des CDS générée par l'algorithme DLC-CDS.

4.5 Conclusion

Dans ce dernier chapitre, nous avons proposé une nouvelle architecture hybride et programmable basée sur *Software Defined Network* (*SDN*). Les différents modules et composants logiciels de cette architecture sont décrits et détaillés. Afin de rendre l'architecture proposée adaptable à différentes technologies de réseau, nous avons introduit trois niveaux de contrôles : les contrôleurs principaux, secondaires et locaux. En outre, nous nous sommes concentré sur la sélection des contrôleurs locaux (*LCs*) en utilisant deux approches : *CDS* et la logique flou.

L'algorithme de construction des *CDS* est de classe distribuée avec une seule phase, c'est la *DLC-CDS* qui est utilisée avec une fonction importante nommée *Score* pour sélectionner les nœuds *CLs*. La fonction *Score* dépend de plusieurs paramètres tels que : le degré de connectivité, la qualité de liaison moyenne et le rang qui est la distance de la passerelle (contrôleurs secondaires).

Afin d'agréger ces paramètres et de calculer la fonction Score, nous avons utilisé une approche de logique floue. L'évaluation de la performance de la fonction Score a été proposée tout au long de la simulation du réseau. L'algorithme *DLC-CDS* est implémenté dans *Matlab* tout en utilisant la fonction *Score* pour construire les dominants connectés agissant en tant que Contrôleur Local (*LC*). *Le DLC-CDS* a prouvé sa performance par apport à l'algorithme *DSP-CDS* tout en réduisant la taille du *CDS* d'environ 40%, dans le cas de la densité élevée des nœuds. Dans le cas d'une portée radio différente, les résultats de la simulation montrent que l'algorithme proposé s'améliore de 36% de la taille des *CDS* par rapport aux *DSP-CDS*.

En effet, l'efficacité de la méthode de contrôle proposée a été validée en effectuant des comparaisons avec les algorithmes existants dans [176]. Les performances ont été évaluées à travers plusieurs scénarii de simulation exécutés sous différents paramètres et métriques.

Enfin, les résultats obtenus montrent l'importance du modèle proposé et sa sensibilité aux différents réseaux.

Conclusion générale

Dans la globalité de notre recherche, nous nous sommes intéressés au contrôle et la gestion efficace des *RCSFs* pour la supervision. Nous avons dans un premier temps pensé à réaliser le concept de contrôle et de gestion d'un *RCSF* par le fait de garantir sa continuité de service de surveillance en cas d'anomalie ou dans le cas de nœuds en panne ou défaillants. En effet, nous avons développé une solution algorithmique apte à satisfaire les besoins des *RCSF* basés sur la technologie *6LoWPAN* voire les utilisateurs de ce type de réseaux en termes de performance et de robustesse. Et ce, tout en proposant une méthode de gestion et de réparation locale avec le protocole de routage *RPL* défini par la norme de 6LoWPAN. Cette méthode a été réalisée à travers l'implémentation et l'exécution de l'algorithme *MNLR_RPL* (*Mobile Node Local Repair-RPL*) avec le simulateur *Cooja* sous *Contiki OS*. Le principe de l'algorithme *MNLR_RPL* est de réagir rapidement face à la défaillance des nœuds pour les remplacer par la mobilité de leurs prédécesseurs, en utilisant le protocole *RPL*. *MNLR_RPL* et en prenant en considération la position du nœud défaillant dans la topologie du réseau. Cela fut réalisé grâce à l'exécution des règles définies par la méthode proposée. En effet, dans cette méthode nous avons défini trois règles traitant le remplacement des nœuds défaillants selon leurs positions par rapport à leurs prédécesseurs, afin de faciliter la mobilité de ces derniers pour les remplacer. Cette méthode, à travers l'algorithme *MNLR_RPL*, a apporté plusieurs avantages pour le protocole *RPL* dans les réseaux à faible ressources dans l'*IdO*, en termes de nouvelle technique, en vue de rénover celle liée à la réparation locale définie par *RPL*.

Notre objectif, dans la première partie de nos propositions de solution, ayant trait à notre recherche, est d'évaluer la performance de la nouvelle technique proposée *MNLR_RPL* pour le protocole de réparation locale *RPL* dans les *RCSFs*. Il s'agit également d'ajouter une nouvelle technique de réparation locale au protocole *RPL*. L'efficacité de cette méthode a été validée dans un environnement de simulation, en présence de plusieurs cas de position du nœud défaillant dans le réseau, et ce, à travers plusieurs scénarios de simulation. Les résultats d'expérimentation obtenus dans cette partie ont prouvé l'efficacité de la méthode de gestion des défaillances pour assurer la continuité de la fonctionnalité du réseau.

Dans la deuxième partie de ce travail, une autre proposition de solution, bien distincte, pour le contrôle des *RCSF* dans l'*IdO* a été présentée. Cette nouvelle solution de contrôle est représentée par la proposition d'une méthode de contrôle fiable à travers une architecture programmable proposée basée sur le paradigme de contrôle programmable *SDN* (*Softwar Defined Network*). En effet, nous avons proposé une architecture hybride et programmable basée sur les réseaux définis par logiciels. Les différents modules et composants logiciels de cette architecture sont décrits et détaillés dans le chapitre quatre. Afin de rendre l'architecture proposée adaptable à différentes technologies de réseau, nous avons introduit trois niveaux de contrôles : les contrôleurs principaux, secondaires et locaux. Nous nous sommes concentrés sur la sélection des contrôleurs locaux (*LCs*) tout en utilisant deux approches : l'approche de sélection et construction des dominants connectés CDS (*Dominating Set*) et l'ensemble flou.

L'algorithme *CDS* distribué avec une seule phase *DLC-CDS* (*Distributed Local Controler-CDS*) est utilisé avec une fonction importante nommée Score en vue de sélectionner les nœuds LCs. Le calcul et l'évaluation de la fonction Score dépend de plusieurs paramètres tels que : le degré de connectivité, la qualité de liaison moyenne et le rang qui représente la position d'un nœud par apport à la passerelle (contrôleurs secondaires). Afin d'agréger ces paramètres et de calculer la fonction Score, nous avons utilisé une approche de la logique floue. L'évaluation de la performance de la fonction Score est proposée tout au long de la simulation du réseau.

En effet, l'algorithme *DLC-CDS* a été implémenté tout en utilisant la fonction score pour construire les dominateurs connectés agissant en tant que contrôleur local (*LC*). L'algorithme *DLC-CDS* a été implémenté sous Matlab et comparé par l'algorithme *DSP-CDS*, qui a été étudié dans les travaux de recherche présentés dans le chapitre quatre. L'efficacité de cette approche a été validée dans un environnement de simulation en présence de plusieurs paramètres et métriques d'évaluation de performance de *DLC-CDS*, à travers plusieurs scénarii de simulation.

Les résultats obtenus ont montré l'amélioration de taille des *CDS* par rapport aux valeurs obtenues par l'algorithme *DSP-CDS*, en outre ces résultats ont montré l'importance du modèle proposé et sa sensibilité aux différents réseaux.

Perspectives

Plusieurs perspectives de recherche futures peuvent être envisagées sur la base des travaux présentés dans cette thèse.

Les *RCSFs* dans l'*IdO* et dont les dispositifs connectés à l'Internet dépasseraient largement le nombre d'individus présent sur Terre, sont des réseaux omniprésents dans notre environnement. Ces réseaux sont appelés aussi, réseaux à faible puissance et à perte de données (*LLN*). C'est une classe de réseau dans laquelle les dispositifs fonctionnent avec des contraintes fortes sur la puissance de traitement, la capacité mémoire et l'énergie disponible. Leurs interconnexions sont caractérisées par des taux relativement faibles de livraison de paquets et une grande instabilité de routage. Le protocole RPL a été dédié à ces réseaux, c'est un protocole de routage proactif à vecteur de distance qui construit des topologies en *DODAG* (*Destination Oriented Directed Acyclic Graph*). Les évaluations de performances de la méthode proposée dans la première partie de solution de notre thèse ont montré son efficacité et sa performance vis-à-vis de la réparation des nœuds défaillants sur une topologie où le nombre de nœuds est peu important, allant de 25 nœuds à 101 nœuds. De plus, le protocole RPL propose les métriques de la fonction objectif pour la sélection de meilleurs chemins pour le routage des données via les nœuds parents préférés (pp), ceci permet d'ouvrir les portes de recherche avec ce protocole pour les différentes métriques de la qualité de service (*QdS*) et la sécurité dans l'*IdO*.

D'un autre coté nous avons exposé, le support de la mobilité pour le protocole RPL, puisqu'enfin réalisé, sachant que ce concept fut absent dans le développement de ce protocole et fut considéré comme un défi dans les recherches scientifiques. Le protocole *RPL* permet au *RCSF* de s'intégrer et d'effectuer la technique du routage dans le réseau d'*IdO*, ce qui permet à plusieurs applications d'y apparaître et de répondre aux exigences des utilisateurs dans ce réseau mondial. Avec les avantages que la méthode proposée (*MNLC-RPL*) a pu apporter au protocole *RPL* en termes de réparation locale et de support de mobilité, nous proposons pour un futur travail d'appliquer cette méthode sur un réseau plus étendu avec un nombre de nœuds augmenté afin d'en étudier l'efficacité avec un réseau à grand échelle. En outre, nous nous proposons d'utiliser d'autres métriques de performance pour atteindre d'autres objectifs.

Enfin, l'efficacité de la méthode proposée (*MNLC-RPL*) a été validée par simulation sous *Cooja*. A présent et avec la disponibilité des capteurs, il serait judicieux de l'évaluer par la suite sur une plateforme réelle, composée d'un nombre fini de capteurs.

On ce qui concerne la deuxième partie de cette thèse, nous avons proposé une nouvelle architecture hybride et semi distribuée basée sur le concept *SDN*. Ce concept a connu une nouvelle révolution dans la gestion, le contrôle et la programmabilité des réseaux. Dans une architecture de *SDN*, le "plan de contrôle" (les logiciels qui pilotent) est ainsi découplé du "plan de données" (partie pilotée, en charge de transférer les données), nous avons respecté dans notre architecture ce concept de séparation entre les deux plans. Cela ouvre des perspectives pour étudier davantage le plan de contrôle et proposer d'autres règles de pilotage pour la gestion des *RCSF* dans l'*IdO*.

Dans l'architecture proposée, nous avons présenté les différents modules à savoir : module de contrôle, module de données, module *Cloud* et *Fog-Computing*, module de sécurité et de confidentialité et module utilisateur final.

Toutefois, nous avons vu dans cette partie de solutions le module de contrôle et en particulier le niveau du contrôleur local, par la proposition de sélection de contrôleurs locaux, à travers la méthode de sélection des dominants connecté *CDS*. Comme futur travail, nous prévoyons de développer d'autres modules liés à l'architecture proposée et de faire une vaste simulation de réseau avec d'autres paramètres de sécurité et de qualité de service (*QdS*).

Enfin, l'efficacité du modèle proposé a été validée par simulation sous *Matlab* pour une nécessité de calcul de paramètres de simulation. Maintenant, avec la disponibilité simulatrice de réseau *SDN*, nous sommes en mesure d'évaluer notre méthode avec un environnement de simulation dynamique.

Enfin et avec la disponibilité des capteurs et des serveurs *SDN*, il serait judicieux d'évaluer notre architecture avec la méthode de sélection des *CL* sur une plateforme réelle.

Bibliographie

[1] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions”, Future Generation Computer Systems journal, pp.1645-1660, 2013.

[2] I. Lee, K. Lee, “The Internet of Things (IoT): Applications, investments, and challenges for enterprises”, Bus. Horizons, pp.431–440, 2015).

[3] J. Yick, B. Mukherjee, D. Ghosal, “Wireless sensor network survey”, International Journal of Computer and Telecommunications Networking, pp.2292-2330, 2008.

[4] A.I. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, “Internet of Things: A survey on enabling technologies, protocols, and applications”, IEEE Commun. Surv. Tutorials 17 (4), pp.2347–2376, 2015.

[5] T. Winter, P. Thubert, A. Brandt, and all, “Rpl: Ipv6 routing protocol for low-power and lossy networks”, IETF ROLL working group, March, 2012, http://datatracker.ietf.org/doc/rfc6550/, (accé on Decembre, 2014).

[6] Routing Over Low power and Lossy networks, Internet Engineering Task Force (IETF), http://www.ietf.org/dyn/wg/charter/roll-charter.html.

[7] N. Bizanis, F. Kuipers, “SDN and virtualization solutions for the Internet of Things: A survey”, IEEE Access, PP. 5591-5606, 2016.

[8] B. Yin, H. Shi, Y. Shang, “An efficient algorithm for constructing a connected dominating set in mobile ad hoc networks”, J. Parallel Distrib. Comput, pp 27–3971, 2011.

[9] D. Bendouda, L. Mokdadb, H. Haffaf, “Method For Fault Management With RPL Protocol In WSNs”, Procedia Computer Science, volume 73, pp 395 – 402, 2015.

[10] D. Bendouda, A. Rachedi, H. Haffaf, “Programmable architecture based on Software Defined Network for Internet of Things: Connected Dominated Sets approach”, Journal of Future Generation Computer Systems, pp.188-197, Mars 2018.

[11] A. Divyakant, D. Sudipto, E. A. Amr, “ Big data and cloud computing: current state and future opportunities “, 14th International Conference on Extending Database Technology, ACM, pp.530-533, 2011.

[12] M. Chen, S. Mao, Y. Liu, “Big data: A survey”, Journal of Mobile Networks and Applications, 2014.

[13] M.T. Lazarescu, “Design of a WSN platform for long-term environmental monitoring for IoT applications”, IEEE J. Emerg. Sel. Top. Circuits Syst, pp 45–54, 2013.

[14] K.Ashton, "That 'Internet of Things' Thing", RFID Journal, 22 June 2009.

[15] A. Hakin, A. Gokhale, P. Berthou, D. C. Schmidt, T. Gayraud, “Software-Defined Networking: Challenges and research opportunities for Future Internet”, Computer Networks, pp 453-471, 2014.

[16] Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: “Internet of things for smart cities”, Internet of Things J. IEEE 1(1), pp 22–32 (2014).

[17] D. Miorandi, S. Sicari, F.De Pellegrini, I. Chlamtac, “Internet of things: vision, applications and research challenges”, Ad Hoc Netw, pp 1497-1516, 2012.

[18] I. Ishaq, D. Carels, G. K. Teklemariam and all, “IETF standardization in the field of the internet of things (IoT): a survey”, Journal of Sensor and Actuator Networks, vol. 2, no. 2, 2013.

[19] B. Benjamin, “Système de gestion de flux pour l’Internet des objets intelligents”, Data Stream Management System for the Future Internet of Things”, l’université de Versailles Saint-Quentin-En-Yvelines, 19 Mars 2015.
[20] http://www.sigfox.com/. Accessed 10 August 2017
[21] LoRaWAN™” Specification v1.0, LoRa Alliance, Inc. 2400 Camino Ramon, Suite 375 San Ramon, CA 94583 (2015).
[22] M.R. Bhalla, A.V. Bhalla , “Generations of Mobile Wireless Technology: A Survey”, International Journal of Computer Applications, pp.0975-8887, 2010.
[23] O. Hersent, D. Boswarthick, O. Elloumi : “L’internet des objets Les principaux protocoles M2M et leur évolution vers IP avec BACnet, LonWorks, ModBus, KNX, Z-Wave, 6LoWPAN, ZigBee SE 2.0, ETS “, Dunod 2014.
[24] Perez, U.A. () Low Power Wi-Fi: A Study on Power Consumption for Internet of Things. Thèse de magistère, Faculté international de Barcelona (FIB), Universitat Politècnica de Catalunya (UPC), BarcelonaTech, 2015.
[25] I. Gifford, C. Bisdikian, B. Heile, “New Standard Fosters Wireless PANs for Notebook Computer”, PDAs, Cell Phones and Other Portable, Handheld Devices, IEEE, 2002
[26] E. Toscano, L. Lo Bello, “Comparative Assessments of IEEE 802.15.4/ZigBee and 6LoWPAN for Low-Power Industrial WSNs in Realistic Scenarios”, 9th IEEE International Workshop on Factory Communication Systems, pp 115-124, 2012.
[27] B. Romdhani, “Exploitation de l'hétérogénéité des réseaux de capteurs et d'actionneurs dans la conception des protocoles d'auto-organisation et de routage”, Thèse de doctorat INSA de Lyon, France, 2012.
[28] I. Akyildiz, I. H. Kasimoglu. “Wireless sensor and actor networks: research challenges” Ad Hoc Networks, pp 351–367, 2004.
[29] L. Mainetti, L. Patrono, et A. Vilei, “Evolution of wireless sensor networks towards the Internet of Things: A survey “, 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), p. 1-6, 2011.
[30] J. Stankovic, “Research Directions for the Internet of Things “, IEEE Internet of Things Journal, vol. 1, no 1, p. 3-9, févr. 2014.
[31] Z. Shelby et C. Bormann, “6LoWPAN: The Wireless Embedded Internet”, John Wiley & Sons, 2011.
[32] V. Handziski, A. Kopke, H. Karl, and A. Wolisz, ”A common wireless sensor network architecture”, Technische University Berlin, pp.10-17, July 2003.
[33] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, “Wireless sensor networks: a survey”, Computer Networks, Vol. 38, No. 4, pp. 393–422, 2002.
[34] TinyOS, accessible via le lien : http://www.tinyos.net/ (consulté le 15/07/2014)
[35] Contiki, accessible via le lien : http://contiki.sourceforge.net/docs/2.6/ (consulté le 10/11/2014).
[36] C.-C. Han, R. Kumar, R. Shea, E. Kohler, M. Srivastava. “A dynamic operating system for sensor nodes”, 3rd international conference on Mobile systems”, applications and services (MobiSys’05), New York, NY, USA, pp. 163-176, 2005.
[37] FreeRTOS, accessible via le lien : http://www.freertos.org/ (consulté le 20/12/2014).
[38] Mantis, accessible via le lien : http://www.cs.colorado.edu/~rhan/sensornets.html (consulté le 20/12/2014).
[39] Nut/OS, accessible via le lien : http://www.ethernut.de/en/software/index.html (consulté le 14/09/2014).
[40] J.P. Fassino, “vers une architecture de systèmes flexibles”, Laboratoire Architecture des Systèmes Répartis de la Direction des Techniques Logicielles de France Télécom Recherche et Développement 2001.

[41] A. Chefi, " Conception d'un micro-capteur d'image CMOS à faible consommation d'énergie pour les réseaux de capteurs sans fil" , Thèse en cotutelle, Université de Grenoble, Université de Monastir, Janvier 2014.

[42] K. Roussel, "Evaluation et amélioration des plates-formes logicielles pour réseaux de capteurs sans-fil, pour optimiser la qualité de service et l'énergie ", Thèse de doctorat, Université de Loraine, France, Juin 2016.

[43] R.V. Biradar, S. R. Sawant, R. R. Mudholkar, and V.C .Patil, "Multihop Routing In Self-Organizing Wireless Sensor Networks", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 1, pp. 155-164, 2011.

[44] Y. Liu, Z. Wang, "Maximizing energy utilization routing scheme in wireless sensor networks based on minimum hops algorithm", Computers and Electrical Engineering, Vol. 38, Issue 3, PP. 703-721, 2012.

[45] A. Boudries, M. Aliouat, "Connectivity maintenance in the wireless sensors networks: Detection and replacement of a failing node", 2nd International Congress on Telecommunication and Application, ICTA'2012. April 11-12, 2012. Bejaia, Algeria.

[46] D. Bendouda, B.Kechar, H.Haffaf, "Protocole de routage réactif multi-chemin pour le transport d'un trafic multimédia dans les réseaux de capteurs Ad-Hoc ZigBee/IEEE 802.15.4 ", Mémoire de magister, Université de Mascara, Octobre 2011.

[47] D. Poellabauer, W. Dargie, and C. Poellabauer, "Fundamentals of Wireless Sensor Networks: Theory and Practice", Wireless Communications and Mobile Computing. John Wiley & Sons, 2010.

[48] N. Bendimerad, "Système de surveillance d'infrastructures publiques à l'aide des réseaux de capteurs vidéo sans fil ", thèse de doctorat, Université d'Oran, Algérie, 2015.

[49] K. E. Gholami, " La gestion de la qualité de service temps-réel dans les réseaux de capteurs sans fil ", Thèse de doctorat, Université Blaise Pascal - Clermont-Ferrand II, France, 2014.

[50] I. Akyildiz, W.Su, Y.Sankarsubramaniam, E.Cayirci, "A Survey on Sensor Networks», Georgia Institute of Technology", IEEE Communications Magazine, Aout 2002.

[51] S. Soro, W. Heinzelman, "Cluster head election techniques for coverage preservation in wireless sensor networks", Ad Hoc Networks, vol. 7, No. 5, pp. 955–972, 2009.

[52] H. Jadidoleslamy, "A Distributed and Hierarchical Intrusion Detection Architecture for Wireless Sensor Networks", International Journal of Network Security and its Applications (IJNSA), Vol. 3, No.5, pp. 131-154, September, 2011.

[53] A. Abbasi, M. Younis; "A survey on clustering algorithms for wireless sensor networks", Computer Communications, vol. 30, pp. 2826–2841, 2007.

[54] O. Younis, M. Krunz, and S. Ramasubramanian; Node Clustering in Wireless Sensor Networks: Recent Developments and Deployment Challenges", IEEE Network, 2006.

[55] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a urvey", Computer networks, Vol. 40, No. 8, pp. 393--422, 2002.

[56] I. Akyildiz, D. Pompili, T. Melodia, "Challenges for efficient communication in underwater acoustic sensor networks", ACM SIGBED Review - Special issue on embedded sensor networks and wireless computing, Vol. 1, Issue 2, pp. 3-8, 2004.

[57] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li, "Research challenges and applications for underwater sensor networking", IEEE Wireless Communications and Networking Conference (WCNC'06), pp. 228–235, Las Vegas, Nev, USA, Avril 2006.

[58] I. Akyildiz, E. Stuntebeck, "Wireless underground sensor networks: research challenges", Ad Hoc Networks, Vol. 4, No. 6, pp. 669–686, 2006.

[59] I. F. Akyildiz, T. Melodia and K. R. Chowdhury. "A survey on wireless multimedia sensor networks", Computer networks 51(4), pp. 921–960, 2007.

[60] A. Mariam, S. Farag, S.Khaled, “A survey of wireless multimedia sensor networks challenges and solutions ”, International Conference on Innovations in Information Technology (IIT), 2011, Vol. 51, No.4, pp. 191-196, 2011.

[61] BJ. Yick, B. Mukherjee, D. Ghosal, “Wireless sensor network survey”, Computer Networks, Vol. 52, No. 12, pp. 2292–2330, 2008.

[62] M. Welsh, D. Malan, B. Duncan, T. Fulford-Jones, S. Moulton, "Wireless Sensor Networks for Emergency Medical Care", Harvard and Boston Universities, A talk presented at GE Global Research, Mar 2004.

[63] S.B. Stankovic, “Medical Applications Based on Wireless Sensor Networks”, Transaction J. on Int. Research, V5, N2, pp.19-23, Jul 2009.

[64] S.Prasanna, S. Rao, “An Overview of Wireless Sensor Networks Applications and Security”, IJSCE, ISSN: 2231-2307, Vol-2, Iss-2, May 2012.

[65] J.Stankovic, A.D. Wood, T. He, “Realistic Applications for Wireless Sensor Networks”, Springer Theoretical Comp Sci, pp 835-863, 2011.

[66] A. Bagula, “Application of Wireless Sensor Networks”, A talk presented at WSN applications, UCT, 2012.

[67] S.Kalpana,G. MK, “Wireless sensor networks: An overview on its security threats”, IJCA, Special Issue on “Mobile Ad-hoc Networks” MANETs, pp.40-45, 2010.

[68] A. Th, L.John, M.Stamatis, “A survey of applications of wireless sensors and wireless sensor networks”, IEEE International Symposium on, Mediterrean Conference on Control and Automation, pp 719-724, 2005.

[69] M.Alan,C.David,P.Joseph and all, “Wireless sensor networks for habitat monitoring”, 1st ACM international workshop on Wireless sensor networks and applications, pp.88-97,2002.

[70] P. Rawat, K. D. Singh, H. Chaouchi, J. M. Bonnin, “Wireless sensor networks: a survey on recent developments and potential synergies”, The Journal of Supercomputing, Vol. 68, Issue 1, pp. 1-48, Avril 2014.

[71] L.D. Nardis, M.G.D. Benedetto, “Overview of the IEEE 802.15.4/4a standards for low data rate Wireless Personal Data Networks”, WPNC'2007, Hannover, Germany, 2007.

[72] ZigBee Alliance, “ZigBee Speci_cation”, October 2007.

[73] N. Anchal, K. Rajesh, “Routing prospective of IEEE 802.15. 4 ZigBee network: A Survey”, International Journal for Science, Management and Technology, vol.13, 2017.

[74] D. Culler, D. Estrin and M. Srivastava, “Overview of Sensor Networks”, IEEE journal of Computer Society, vol. 37, no. 8, pp 41-49, august 2004.

[75] N. Abdeddaim, “Analyse des performances d'un réseau de capteurs exploitant le standard IEEE 802.15.4”, Thèse doctorat, Grenoble, 2006.

[76] IEEE Computer Society. Part 15.4: “wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs)”, IEEE Standard 802.15.4,2007.

[77] Y. Huang, A.Y.Pang : “A comprehensive analysis of low-power operation for beacon-enabled IEEE 802.15.4 wireless networks”, IEEE Transactions Wireless Communications, vol. 8(11), pp. 5601–5611, 2009.

[78] M. Hossen, A.F.M.S. Kabir, R.H.Khan, A. Azfar, “Interconnection between 802.15.4 Devices and IPv6: Implications and Existing Approaches”, IJCSI, Vol.7, 2010.

[79] P.Baronti, P.Pillai, V.W.C.Chook, S.Chessa, A.Gotta , Y.Fun Hu,”WSNs: A survey on the state of the art and the 802.15.4 and ZigBee standards”, ELSEVIER J. ComCom, V30, Iss7, pp1655-1695, 2007.

[80] J.W.Hui, D.E.Culler, “Extending IP to low-power, wireless personal area networks”, IEEE JIC, vol:12, iss:4, pp 37-45, 2008.

[81] J. Hui, D.Culler, S. Chakrabarti, "6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture", (IPSO) Alliance, 2009.

[82] K. Korte, I. Tumar, J. Schonwalder, "Evaluation of IPv6 over Low-Power Wireless Personal Area Networks Implementations", 4th IEEE SenseApp'09, Zurich, pp.20-23, 2009

[83] M. B. Rezgui. " Evaluation de protocoles pour la qualité de service dans les réseaux de capteurs sans fils. Réseaux et télécommunications " [cs.NI]. 2013. <dumas-01270767>

[84] G.K. Ee, C.K. Ng, N.K. Noordin, B. Mohd, "A Review of 6LoWPAN Routing Protocols", Asia-Pacific Advanced Network 30th Meeting, Hanol, Vietnam, 2010.

[85] P.Levis, A.Tavakoli, S.Dawson-Haggerty, "Overview of Existing Routing Protocols for Low Power and Lossy Networks". Internet-Draft draft-ietf-roll-protocols-survey-07, Internet Engineering Task Force, Apr. 2009.

[86] K. Kim, S. Daniel Park, G. Montenegro, S. Yoo, N. Kushalnagar, "6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD)", IETF Internet Draft: draft-daniel- 6lowpan-load-adhoc-routing-03, 2007.

[87] M. Vucinic, B. Tourancheau, A. Duda,: "Performance Comparison of the RPL and LOADng Routing Protocols in a Home Automation Scenario", IEEE Wireless Communications and Networking Conference, Shanghai, China, 2013.

[88] P. Nand, D. Sharma, "Routing Load Analysis of Broadcast based Reactive Routing Protocols AODV, DSR and DYMO for MANET". International journal of grid and distributed, pp 81–92, 2011.

[89] M. Bouallegue, k. R. R. Bouallegue, "Design and Implementation of New Routing Strategy for Enhanced Energy efficient in WSN ", The International Journal of Computer Networks and Communications (IJCNC), Vol. 7, No. 4, pp 127-137, 2015.

[90] M.Gupta, S. Kaushik, "Performance Comparison Study of AODV, OLSR and TORA Routing Protocols for MANETS", International Journal of Computational Engineering Research, May-June 2012, Vol. 2, Issue No.3 704-711

[91] M. Bouallegue, "Protocoles de communication et optimisation de l'énergie dans les réseaux de capteurs sans fil Réseaux et télécommunications [cs.NI] ", Université du Maine, 2016. France, NNT : 2016 LEMA1https://tel.archives-ouvertes.fr/tel-01400679011.

[92] M. Benazzouz, "Surveillance de tout point d'une zone d'intérêt à l'aide d'un réseau de capteur multimédia sans fil ", Rapport de recherche. Ecole nationale supérieure d'informatique, Oued-Smar, Alger, Algérie, 2013.

[93] N. BAHNES, "Gestion et supervision d'un réseau de capteurs sans fil à l'aide d'un protocole à économie d'énergie WSN-SNMP ", Université d'oran1, memoir de magistère 2010.

[94] K. RIAD, " Détection et isolation de défauts combinant des méthodes à base de données appliquées aux systèmes électro-énergétiques", Thèse de doctorat en science, UNIVERSITE FERHAT ABBES - SETIF 1 Algérie, Mars 2015.

[95] Y. Challal, "Réseaux de Capteurs Sans Fils : Systèmes intelligents pour le Transport ", manuscrit, Université de Technologie de Compiègne, France, 2008.

[96] A.A. Abbasi, M. Younis, K. Akkaya, "Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks", In IEEE Transactions on parallel and distributed systems, 20(9), pp. 1366-1379, 2009.

[97] N. Tamboli, and M. Younis, "Coverage-aware connectivity restoration in mobile sensor networks", Journal of Network and Computer Applications, pp. 363-374, 2010.

[98] A. Zamanifar, O. Kashe_ and M. Sharif, AOM : "An efficient approach to restore actor-actor connectivity in wireless sensor and actor networks, In International Journal of Computer Networks & Communications, pp. 61-72, 2009.

[99] G. Wang, G. Cao, T. La Porta, and W. Zhang, “Sensor Relocation in Mobile Sensor Networks”, in the Proceedings of the 24th Annual Joint Conference of IEEE Computer Communications Societies (INFOCOM'05), Miami, FL, March 2005.

[100] A. BOUDRIES, “Maintien de la Connectivité dans les Réseaux Ad hoc sans fil”, thèse de doctorat, Université Ferhat Abbas de Sétif 1, 2014.

[101] M. Jorgic, I. Stojmenovic, M. Hauspie, and all, “Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks”, of the 3rd Annual IFIP Mediterranean Ad Hoc Networking Workshop, Turkey, 2004, pp. 360-371.

[102] F. Dai, and J. Wu, “An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks”, IEEE Transaction on Parallel and Distributed Systems, 15(10), pp. 908-920, 2004.

[103] K.Ines, M.Pascale, L. Anis, M.Saoucene, “ Survey of deployment algorithms in wireless sensor networks: coverage and connectivity issues and challenges ”, International Journal of Autonomous and Adaptive Communications Systems
, pp.341—390, 2017.

[104] N. Ajay, S. Tarasia, S. Dash, S.Ray, ARSwain, “Une erreur dynamique tolérant protocole de routage pour prolonger la durée de vie des réseaux de capteurs sans fil”, Journal (IJCSIT), Vol. 2 (2), pp727-734, 2011.

[105] N. Ajay, S. Tarasia, S. Dash, S.Ray, ARSwain, “Protocole de routage tolérant aux fautes multi-niveaux avec des horaires du sommeil (FMS) pour les réseaux de capteurs sans fil“, European Journal of Scientific Research,, Vol. 55(1) : pp97-108, 2011.

[106] A. Boukerche et al., “A Fast and Reliable Protocol for Wireless Sensor Networks in Critical Conditions Monitoring Applications”, International Conference (MSWiM’04), Canada, 2004.

[107] A. Boukerche et al. “A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range”, Journal of computer communications, Vol. 29 no. 4, pp477-489, 2006.

[108] A. Boukerche et al., “Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments”, Journal of Parallel and Distributed Computing, Vol. 66 no. 4, pp586-599, 2006.

[109] F. Dai, J.Wu, “On constructing k-connected k-dominating set in wireless ad hoc and sensor networks”, Journal of Parallel and Distributed Computing, Vol. 66no. 7, pp947-958, July 2006.

[110] NAKAYAMA Hidehisa et al., “Fault-resilient sensing in wireless sensor networks”, Journal of Computer communications, vol. 30, no 11-12, pp. 2375-2384, 2007.

[111] A. Brandt, E. Baccelli, R. Cragie, P. van der Stok, “Applicability Statement: The Use of the Routing Protocol for Low-Power and Lossy Networks (RPL) Protocol Suite in Home Automation and Building Control”, Journal of Consultant, ISSN: 2070-1721, 2016.

[112] Olfa Gaddour, Anis Koubaa, “RPL in a nutshell: A survey”, ELSEVIER Journal of Computer Networks, pp 3163-3178, 2012.

[113] Gee Keng Ee, Chee Kyun Ng, Nor Kamariah Noordin and Borhanuddin Mohd. Ali, “A Review of 6LoWPAN Routing Protocols”, Asia-Paci_c Advanced Network 30th Meeting, Hanol, Vietnam, August 2010.

[114] A. Tavakoli, HYDRO: “A Hybrid Routing Protocol for Lossy and Low Power Networks”, IETF Internet Draft: draft-tavakoli-hydro-01, September 2009.

[115] K. Kim, S. Yoo, J. Park, S.D. Park, J. Lee, “Hierarchical Routing over 6LoWPAN (HiLow)”, IETF: Internet Draft: draft-deniel-6lowpan-hilow-hierarchical-routing-00.txt, vol. 38, December 2005.

[116] M. Dohler, T. Watteyne, T. Winter, D. Barthel, “Routing Requirements for Urban Low-Power and Lossy Networks”, RFC 5548, may 2009.

[117] K. Kim, S. Park, I. Chakeres, C. Perkins, “Dynamic MANET On-demand for 6LoWPAN (DYMO-low) Routing, Internet Draft”: draft-montenegro-6lowpan-dymo-low-routing-03, June 2007.
[118] JM Chang, HY Yang, HC Chao, “Multipath design for 6LoWPAN ad hoc on-demand distance vector routing”, International Journal of Information Technology, Communications and Convergence, pp 24-40, 2010.
[119] The Internet Engineering Task Force (IETF). http://www.ietf.org/.
[120] IETF ROLL Working Group. http://tools.ietf.org/wg/roll.
[121] M. BOUAZIZ, “Mobility Support and Routing in 6loWPAN based Wireless Sensor Networks”, Thèse de doctorat, ENSI, UNIVERSITY OF MANOUBA, Tunisia, 2017.
[122] A. Conta, S. Deering, E.M. Gupta, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specication”, RFC: 4443, March 2006.
[123] J. Hui, JP. Vasseur, “The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams”, IETF Request for Comments 6553, March 2012.
[124] JP Vasseur, Navneet Agarwal, Jonathan Hui, Zach Shelby, Paul Bertrand, Cedric Chauvenet, “RPL: The IP routing protocol designed for low power and lossy networks”, (IPSO) alliance, april 2011.
[125] P.Levis, T.Clausen, J.Hui, O.Gnawali, J.Ko, "The Trickle Algorithm", IETF Request for Comments 6206, March 2011.
[126] JP. Vasseur, M. Kim, K. Pister, N. Dejean, D. Barthel, “Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks”, IETF Request for Comments 6551, March 2012.
[127] P. Thubert, “Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)”, IETF Request for Comments 6552, March 2012.
[128] O. Gnawali, P. Levis, “The Minimum Rank with Hysterisis Objective Function”, draftietf-roll-minrank-hysterisis-of-11", 2012.
[129] O. Gnawali, P. Levis, “The ETX Objective Function for RPL”, IETF Internet Draft: draft-gnawali-roll-etxof-01, 2010.
[130] P. Thubert, “Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)”, IETF Request for Comments 6552, March 2012.
[131] J.P.Vasseur, N. Agarwal, J.Hui, , Z. Shelby, P. Bertrand, C.Chauvenet, “ RPL: The IP Routing Protocol Designed for Low Power and Lossy Networks”, Internet Protocol for Smart Objects (IPSO) Alliance, April. 2011.
[132] T.Zhang, L.Xianfeng, “Evaluating and Analyzing the Performance of RPL in Contiki”, the 1st international workshop on Mobile sensing, computing and communication (MSCC'14) Pages: 19-24, August 11, 2014, Philadelphia, PA, USA.
[133] O.Gaddour, A.Koubaa, “Co-RPL: RPL Routing for Mobile Low Power Wireless Sensor Networks using Corona Mechanism”, Industrial Embedded Systems (SIES), 9th IEEE International Symposium P. 200 – 209, 2014.
[134] S.Nabil,S. Shigenobu,A. Mohammed, A.Sabah, “A comprehensive survey on hierarchical-based routing protocols for mobile wireless sensor networks: Review, taxonomy, and future directions”, Wireless Communications and Mobile Computing, volume 2017,2017.
[135] C. Cobarzan, J. Montavont, and Noel, T. “Integrating mobility in RPL”, Wireless Sensor Networks, Lecture Notes in Computer Science, Springer International Publishing, Vol. 8965, pp.135–150, 2015.
[136] O.Gaddour, , A.Koubâa, M. Abid, “Quality-of-service aware routing for static and mobile IPv6-based low-power and lossy sensor networks using RPL”, Ad Hoc Networks, Sci. Direct J, October, Vol. 33, No. C, pp.233–256, 2015.

[137] L. Ben Saad, B. Tourancheau, “Sinks mobility strategy in IPv6-based WSNs for network lifetime improvement”, in International Conference on New Technologies, Mobility and Security (NTMS), IFIP, Paris, http://hal.inria.fr/inria-00563724, 2011.

[138] D.Carels, E. De Poorter, I. Moerman, P. Demeester, “RPL mobility support for point-to-point traffic flows towards mobile nodes”, International Journal of Distributed Sensor Networks, Article ID 470349, pp. 13, http://dx.doi.org/10.1155/2015/470349.

[139] K.D. Korte, A. Sehgal, J. Schönwälder, “A study of the RPL repair process using ContikiRPL, dependable networks and services”, 6th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, Lecture Notes in Computer Science, Springer International Publishing, pp.50–61, 2015.

[140] J.Tripathi, , J.C. de Oliveira, J.P. Vasseur, “Applicability study of RPL with local repair in smart grid substation networks”, First IEEE International Conference on Smart Grid Communications, pp 262–267.

[141] J.Yang, , J.Cao, , Wu, W. and Xu, C.Z. (2009) “Efficient algorithms for fault tolerant mobile agent execution”, Int. J. of High Performance Computing and Networking, Vol. 6, No. 2, pp.106–118.

[142] L.Chen, G. Fan, Y. Liu, “A formal method to model and analyse QoS-aware fault tolerant service composition”, Int. J. of Computational Science and Engineering, Vol. 12, Nos. 2/3, pp.133–145.

[143] Z. Liao, Q.Yin, , Y. Huang, L. Sheng, (2015) “Management and application of mobile big data”, Int. J. of Embedded Systems, Vol. 7, Nos. 3/4, pp.318–323.

[144] D. Chen, X. Zhu, W. Dai, R.Zhang, “Socially aware mobile application integrations in heterogeneous environments”, J. of High Performance Computing and Networking, Vol. 8, No. 1, pp.6 –70.

[145] V.Koster, D.Dorn, A.Lewandowski and C.Wietfeld, “A novel approach for combining Micro and Macro Mobility in 6LoWPAN enabled Networks”, IEEE VTC, pp 1-5, San Francisco, CA, Sept 2011.

[146] M.S.Shahamabadi, B.B.M.Ali, N.K.Noordin, M.b.A.Rasid, P.Varahram, and A.J.Jara, “A Network Mobility Solution Based on 6LoWPAN Hospital WSN (NEMOHWSN)”, IEEE IMIS,pp 433-438, Taichung, Jul 2013.

[147] M.S.Shahamabadi, B.B.M.Ali, P.Varahram, A.J.Jara, “A NEMO-HWSN Solution to Support 6LoWPAN Network Mobility in Hospital Wireless Sensor Network”, Computer Science and Information Systems.

[148] G.Bag, H.Mukhtar, S.M.S.Shams, K.H.Kim, S.Yoo, “Inter-PAN mobility support for 6LoWPAN”, Int. Conf on CHIT, Busan, Korea, Nov 2008.

[149] G.Bag, S.M.S.Shams, A.H.Akhbar, M.T.Raza, K.H.Kim and S.W.Yoo, “Network Assisted Mobility Support for 6LoWPAN”, 6th IEEE CCN, Las Vegas, USA, January 2009.

[150] G.Bag, M.T.Raza, K.H.Kim and S.W.Yoo,”LoWMob:Intra-PAN Mobility Support Schemes for 6LoWPAN”,Sensors'09,V9,I7,p5844-5877, Jul 09.

[151] Z.Zinonos and V.Vassiliou, “Inter-Mobility Support in Controlled 6LoWPAN Networks”, Globecom 2010, pp1718-1723, Miami, FL, Dec 2010.

[152] A.J.Jara, R.M.Silva, J.S.Silva, M.A.Zamora, A.F.G.Skarmeta, “Mobile IP-Based Protocol for Wireless Personal Area Networks in Critical Environments", Springer journal of WPC, V61, I4, pp 711-737, Dec 2011.

[153] W.Xiaonan, Z.Shan, Z.Rong,”A mobility support scheme for 6LoWPAN”, ELSEVIER journal of ComCom, V35, Iss3, pp392-404, Feb 2012.

[154] T. Voigt, “Contiki COOJA Crashcourse”, RoboSense School, December 2012.

[155] The Contiki Operating System. http://www.sics.se/contiki

[156] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T.urletti, “A survey of software-defined networking: Past, present, and future of programmable networks”, IEEE Commun. Surveys Tuts., vol. 16, no. 3, pp. 1617_1634, Sep. 2014.

[157] L. Bertaux et al., “Software defined networking and virtualization for broadband satellite networks”, IEEE Commun. Mag., vol. 53, no. 3, pp. 54_60, Mar. 2015.

[158] J. Tourrilhes, P. Sharma, S. Banerjee and J. Pettit, “SDN and OpenFlow Evolution: A Standards Perspective”, in Computer, vol. 47, no. 11, pp. 22-29, Nov. 2014.

[159] S. Costanzo, L. Galluccio, G. Morabito and S. Palazzo, “Software Defined Wireless Networks: Unbridling SDNs”, European Workshop on Software Defined Networking, pp.1-6, 2012.

[160] W.Timothy,R.KK,H.Jinho and all,“Toward a software-based network: integrating software defined networking and network function virtualization”, IEEE Network, pp. 36–41, 2015.

[161] D. Suh, S. Jang, S. Han, S. Pack, M.-S. Kim, T. Kim and C.-G.Lim, “Toward Highly Available and Scalable Software Defined Networks for Service Providers”, in IEEE Communications Magazine, vol. 55, no. 4, pp. 100-107, April 2017.

[162] M.A. Hassan, Q.T. Vien, M. Aiash. “Software Defined Networking for Wireless Sensor Networks: A Survey”, Advances in Wireless Communications and Networks. Vol. 3, No. 2, pp. 10-22, 2017.

[163] 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), Kingston, pp. 71-75, 2014.

[164] B. Trevizan de Oliveira, L. Batista Gabriel and C. Borges Margi, "TinySDN: Enabling Multiple Controllers for Software-Defined Wireless Sensor Networks," in IEEE Latin America Transactions, vol. 13, no. 11, pp. 3690-3696, Nov. 2015.

[165] K. Slavov, D. Migault and M. Pourzandi, “Identifying and addressing the vulnerabilities and security issues of SDN,” in Ericsson Technology Review, Vol. 92, No. 7, 2015.

[166] Y. Choi, Y. Choi and Y.-G. Hong, "Study on coupling of software-defined networking and wireless sensor networks," 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), Vienna, 2016, pp. 900-902.

[167] D. O’Shea, V. Cionca and D. Pesch, “The Presidium of Wireless Sensor Networks - A Software Defined Wireless Sensor Network Architecture,” in: Mobile Networks and Management, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 158. Springer, 2015.

[168] A. Boonsongsrikul, S. Kocijancic and S. Suppharangsan, "Effective energy consumption on wireless sensor networks: Survey and challenges," 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, pp. 469-473,2013.

[169] A. Mahmud and R. Rahmani, "Exploitation of OpenFlow in wireless sensor networks," Proceedings of 2011 International Conference on Computer Science and Network Technology, Harbin, pp. 594-600, 2011.

[170] T. Luo, H. P. Tan and T. Q. S. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," in IEEE Communications Letters, vol. 16, no. 11, pp. 1896-1899, Nov. 2012.

[171] L. Galluccio, S. Milardo, G. Morabito and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," 2015 IEEE Conference on Computer Communications (INFOCOM), Kowloon, 2015, pp. 513-521.

[172] J. Zhou et al, (Member, IEEE), LIHUA WU, CHUNSHENG ZHU, “SDN-Based Application Framework for Wireless Sensor and Actor Networks “, IEEE Access}, volume 4, pages = 1583-1594, 2016.

[173] B.T. de Oliveira, R.C.A. Alves, C.B. Margi, “Software-defined wireless sensor networks and Internet of Things standardization synergism”, IEEE Conference on Standards for Communications and Networking, CSCN 2015, Tokyo, Japan, October 28-30, 2015, pp. 60–65.

[174] B.T. de Oliveira, C.B. Margi, L.B. Gabriel, “TinySDN: Enabling multiple controllers for software-defined wireless sensor networks”, IEEE Latin-America Conference on Communications, LATINCOM 2014, Cartagena de Indias, Colombia, November 5-7, pp. 1-6, 2014.

[175] A.-C.G. Anadiotis, G. Morabito, S. Palazzo, “An SDN-assisted framework for optimal deployment of mapreduce functions in WSNs”, IEEE Trans. Mob. Comput, pp. 2165–2178, 2016.

[176] A. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, “Towards a software-defined network operating system for the iot,” in WF-IoT, pp. 579–584, 2015.

[177] A.A. Khan, M.H. Rehmani, A. Rachedi, “When cognitive radio meets the internet of things”, International Wireless Communications and Mobile Computing Conference, IWCMC, Paphos, Cyprus, September 5-9, pp. 469– 474, 2016.

[178] S. Hamdoun, A. Rachedi, H. Tembine, Y. Ghamri-Doudane, “Efficient transmission strategy selection algorithm for M2M communications: An evolutionary game approach”, 15th IEEE International Symposium on Network Computing and Applications, NCA 2016, Cambridge, Boston, MA, USA, October 31 - November 2, pp. 286–293, 2016.

[179] A.A. Khan, M.H. Rehmani, A. Rachedi, “Cognitive-radio-based Internet of Things: applications, architectures, Spectrum related functionalities, and future research dIrections”, IEEE Wirel. Commun, 2017.

[180] S. Tomovic, K. Yoshigoe, I. Maljevic, I. Radusinovic, “Software-Defined fog network architecture for IoT”, Wirel. Pers. Commun. pp. 181–196, 2017.

[181] M. Bouaziz, A. Rachedi, “A survey on mobility management protocols in wireless sensor networks based on 6LoWPAN technology”, Comput. Commun, pp.3-15, 2016.

[182] A. Schwartz, “A reinforcement learning method for maximizing undiscounted rewards”, The 10th international conference on machine learning, Amherst, pp. 298–305,1993.

[183] T. Gazdar, A. Rachedi, A. Benslimane, A. Belghith, ”A distributed advanced analytical trust model for VANETs”, Global Communications Conference, GLOBECOM, pp.01-206, 2012.

[184] G. Glissa, A. Rachedi, A. Meddeb, ”A secure routing protocol based on RPL for Internet of Things”, Global Communications Conference, GLOBECOM 2016, IEEE, pp.1–7, 2016.

[185] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C.P.A. Ogah, Z. Sun, ”Blockchainbased dynamic key management for heterogeneous intelligent transportation systems”, IEEE Internet Things, 2017.

[186] M. Works, Fuzzy logic toolbox users guide, Jan 1998, http://www.mathworks.com.

[187] B. Meunier, Bernadette, Marsala, Christophe, Logique floue : ”principes, aide à la décision ” livre, Version 1, 2003,URL = https://hal.archives-ouvertes.fr/hal-01533303.

Annexe A : Installation de Contiki et Le simulateur Cooja

A.1 Introduction

Contiki est un système d'exploitation pour les réseaux de capteurs sans fil. Cooja est un émulateur réseau basé sur Contiki qui permet d'exécuter des programmes sur Contiki sans avoir besoin du matériel. Pour éviter d'installer tout l'environnement de développement, nous allons utiliser une machine virtuelle (VM) nommée «Instant Contiki». Etapes à suivre pour mettre en place la VM :

1. Télécharger la machine virtuelle nommée : « Instant Contiki 2.7 ou 2.6 ». Le site : http://sourceforge.net/projects/contiki/files/Instant%20Contiki/ ou http://www.contiki-os.org/ Il s'agit d'un fichier de grande taille, un peu plus de 1 Gigaoctet. Une fois télécharger, décompressez le fichier et placez le répertoire décompressé sur le bureau de l'ordinateur.
2. Pour faire tourner cette machine virtuelle, il faut télécharger un lecteur des machines virtuelles comme *VirtualBox* ou *VMPlayer*. Si vous utilisez *VirtualBox*, il faut vérifier que l'option « *PAE/NX* » est activée (Settings → System → Processor), comme illustré dans la figure A.1.

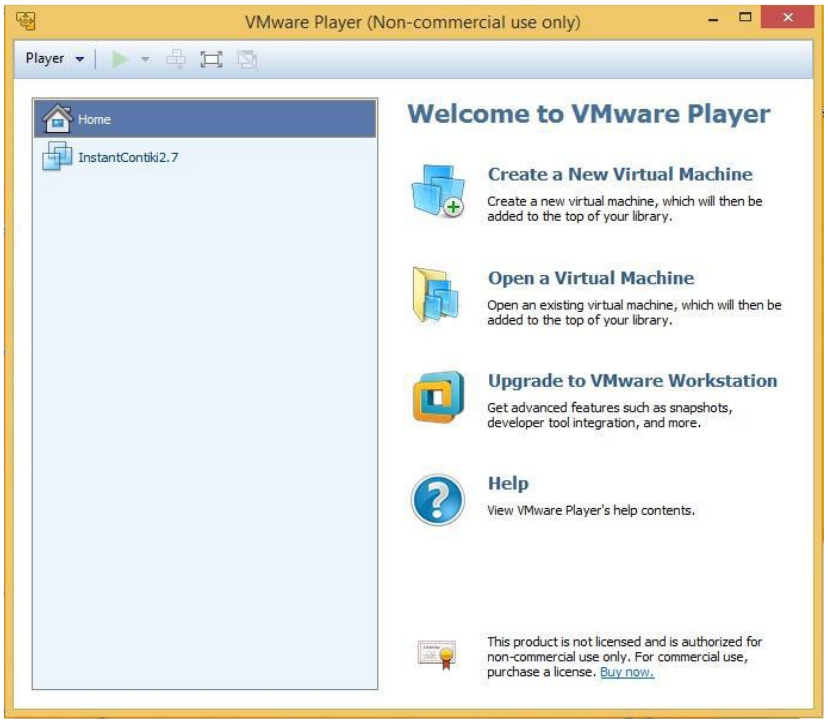


Figure A.1 : Interface de la machine virtuelle (***VMPlayer)***.

3. Une fois est installé le lecteur des machines virtuelles avec la machine Instant Contiki, on peut utiliser le login et mot de passe (voire figure A.2).

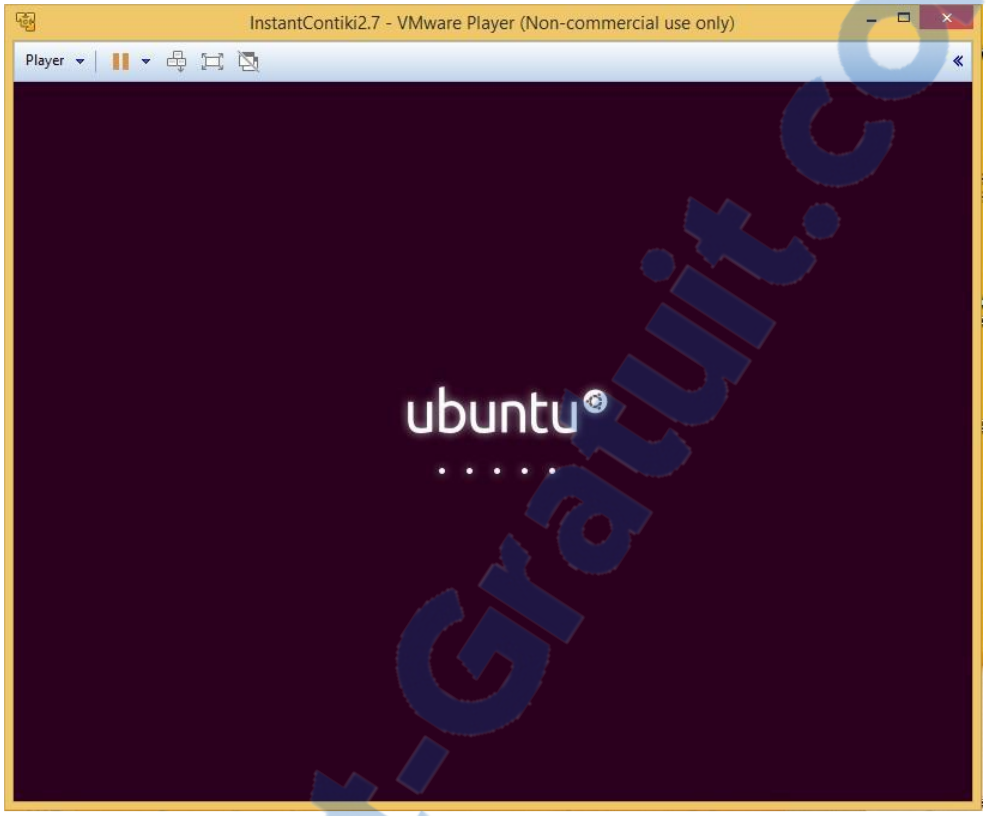


Figure A.2 : Interface de du lancement du système Contiki.

4. Dans cette étape on utilise le mot de passe : ***user*** (voire figure A.3).

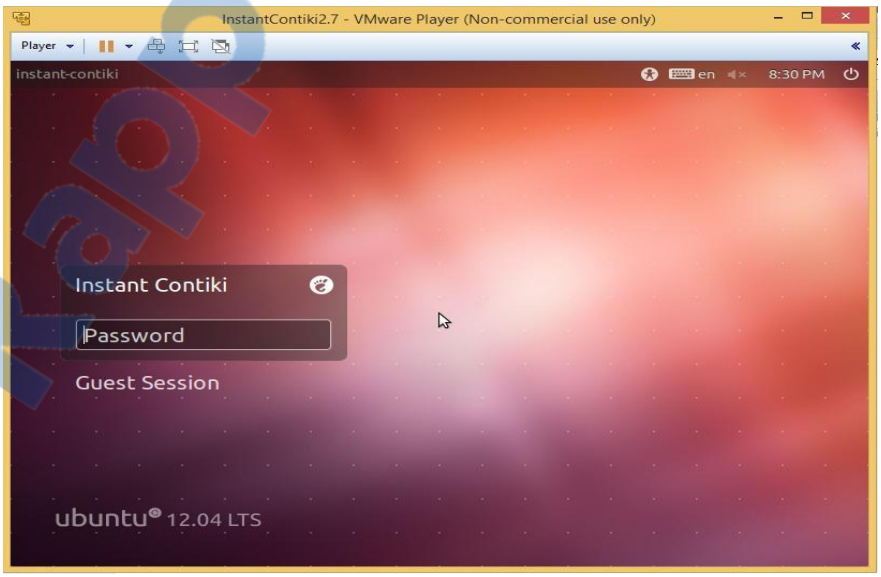


Figure A.3 : Interface de la cession du système Contiki.

5. Lancer l'émulateur Cooja : Pour lancer l'émulateur "Cooja", il suffit d'aller cliquer sur l'icône Cooja qui se situe sur le bureau, il va prendre un peu de temps pour se compilé puis afficher une fenêtre bleue.
6. Dés on aura cette fenêtre (voir figure A.4), on peut commencer à simuler nos topologies.

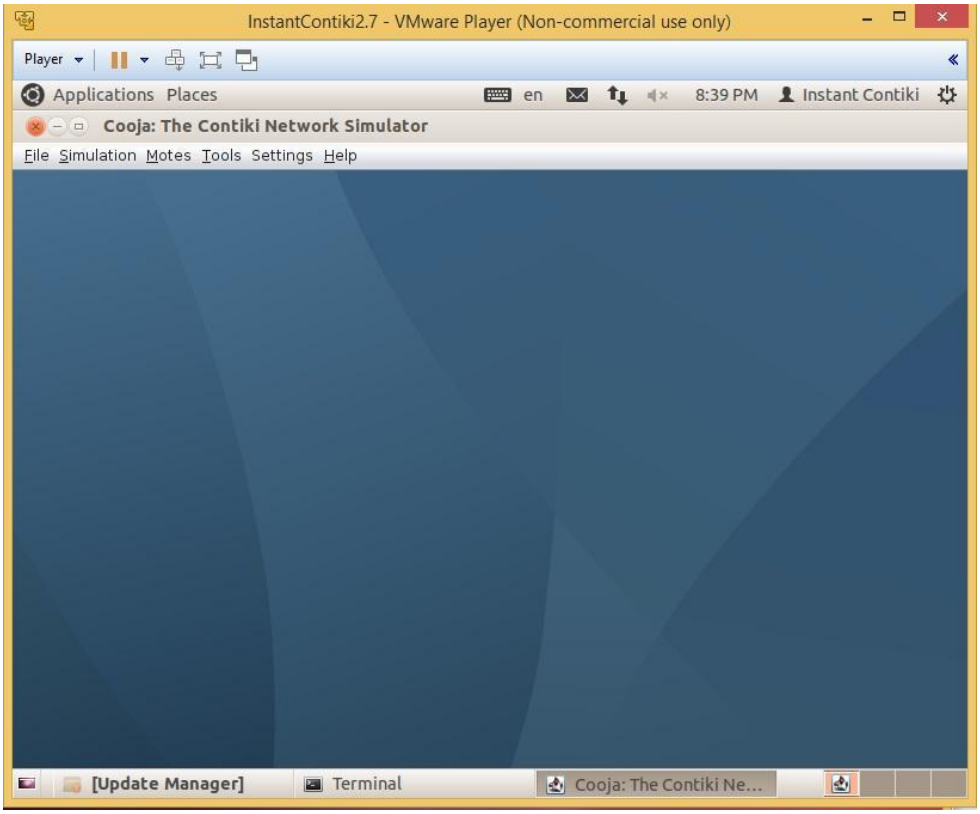


Figure A.4 : Interface du simulateur Cooja.

A.2 Un simulateur réseau pour Contiki : *Cooja*

COOJA [28] est l'acronyme de Contiki OS Java Simulator. Pour développer les programmes au sein de Contiki, le système met à disposition un simulateur réseau appelé *Cooja*. Le logiciel permet d'émuler des nœuds et de charger un programme compilé. Ceci est particulièrement utile pour tester les programmes avant de les mettre dans la mémoire flash des nœuds réels, puisque le logiciel simule les conditions d'exécution et de mémoire de la plateforme *TI MSP430*. Les données collectées provenant du sink via sa sortie standard peuvent être enregistrées dans des fichiers ou lus par des logiciels qui peuvent par la suite traiter et présenter les données à l'utilisateur.

On peut par exemple citer le logiciel *collect-view* intégré dans *Contiki* qui permet de visualiser les valeurs des capteurs et des données de supervision du réseau.

A.3 Fenêtre de simulation

Dans une simulation nous avons plusieurs fenêtres selon la figure A.5 :

- **La fenêtre *Timeline* :** en bas de l'écran, nous affiche tous les événements de communication dans la simulation dans le temps, très pratique pour comprendre ce qui se passe dans le réseau.
- **La fenêtre Network :** en haut à gauche de l'écran, nous montre tous les noeuds dans le réseau simulé.
- ***La fenêtre Mote Output***, sur le côté droit de l'écran, nous montre toutes les impressions port série de tous les noeuds.
- ***La fenêtre Notes*** en haut à droite est l'endroit où nous pouvons mettre des notes pour notre simulation.
- ***La fenêtre Simulation control*** : est où nous pouvons lancer, mettre en pause et charger de notre simulation.

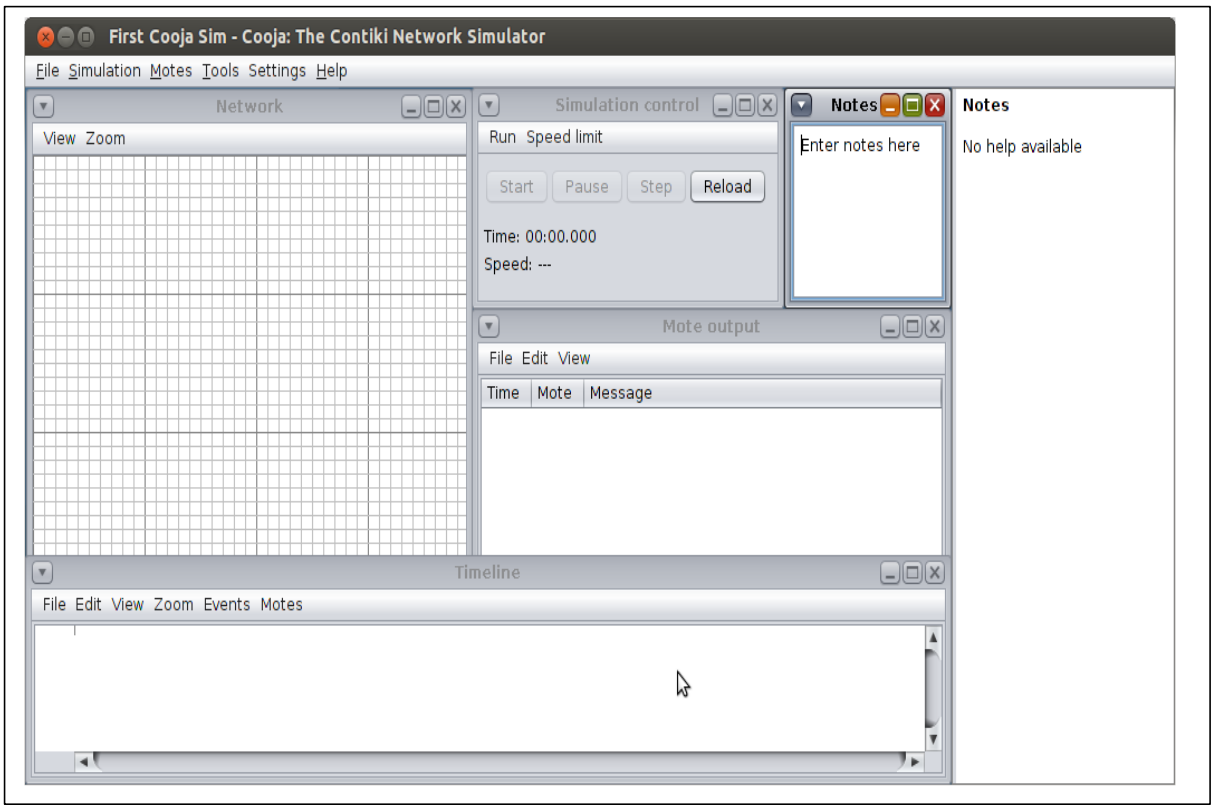


Figure A. 5 : Fenêtres de simulation Cooja.

Exploiting node mobility for fault management in RPL-based wireless sensor networks

Djamila Bendouda*

Department of Computer Science,
Faculty of Sciences,
Research Laboratory in Industrial Computing and Networks (LRIIR),
Oran 1 Ahmed Benbella University,
P.O. Box 1524, El M'Naouar, Oran, Algeria
Email: bendjamila39@gmail.com
*Corresponding author

Lynda Mokdad

LACL Laboratory,
Paris-Est University,
Creteil, France
Email: lynda.mokdad@u-pec.fr

Hafid Haffaf

Department of Computer Science,
Faculty of Sciences,
Research Laboratory in Industrial Computing and Networks (LRIIR),
Oran 1 Ahmed Benbella University,
P.O. Box 1524, El M'Naouar, Oran, Algeria
Email: haffaf.hafid@univ-oran1.dz

Abstract: This paper aims to propose an effective new local repair technique with RPL protocol in WSNs. To replace node failure by the mobility of their predecessor without rebuild the RPL tree, is a new proposed solution with MN-LR_RPL method. In addition, RPL was originally designed for static networks, with no support for mobility. However, handling the local repair mechanism for the RPL with mobile nodes is a real challenge. Our work is the first effort implemented for fault management using MN-LR_RPL that is presented by an algorithm which is simulated using COOJA with Contiki OS. The performance evaluation results show that MN-LR_RPL allow a great improvement when compared to the standard specification in case local repair, mainly in terms of packet loss ratio and average network latency. Based on our new solution, we offer a few suggestions to using wireless software defined network (WSDN) to improve RPL protocol in IoT.

Keywords: wireless sensor network; WSN; vehicular ad hoc networks; VANETs; fault management; local repair; mobile node; RPL protocol; Contiki; Cooja.

Reference to this paper should be made as follows: Bendouda, D., Mokdad, L. and Haffaf, H. (xxxx) 'Exploiting node mobility for fault management in RPL-based wireless sensor networks', *Int. J. High Performance Computing and Networking*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Djamila Bendouda

Comment [t1]: Author: Please provide the biographical details of D. Bendouda of no more than 100 words.

Lynda Mokdad received her PhD in Computer Science from the University of Versailles, France in 1997. She was Associate Professor at University Paris-Dauphine from 1998 to 2009. She is currently Full Professor at University Paris-Est, Créteil since 2009. Her main research interests are about performance evaluation techniques and applications in wired, mobile and wireless networks, ad hoc networks and in software technologies as web services. She is recipient of the best paper awards of IEEE International Conference on Communications and Information Technology (ICCIT 2011) and IEEE International Conference on Communications (ICC 2011). She has served as technical committee for more than 50 international IEEE/ACM conferences and workshops including ICC, GlobeCom, MSWIM, LCN, etc. She is a member of IEEE and

ACM. She served as a publicity chair of 12th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM'10) and as awards chair of the 8th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-10), She serves as Editor of *IJCS* and *WCMC*. She was Secretary and Vice Chair of IEEE Communication Software (CommSoft) and she is currently serving as a Chair. She is active member of IEEE Ad hoc and Sensor Networks.

Comment [t2]: Author: Please reduce the biographical details of L. Mokdad to no more than 100 words.

Hafid Haffaf obtained his Doctor degree in Computer Science in 2000. He is Professor at the University of Oran 1 (Algeria). He actually heads the R.I.I.R. Laboratory at Computer Science Department – Oran University. His researches concern different domains as industrial diagnosis, optimisation and reconfiguration systems. Using hypergraph and matroid theory, system of systems approaches, we can model many applications in monitoring domain. He has many collaborations projects with European laboratory: LAGIS in Polytech lille where he worked in intelligent transport systems infrastructures projects (Intrade, Weastflows) and where he had been Visiting Professor several times; and LIAUPA in PAU (France) in the domain of WSNs.

This paper is a revised and expanded version of a paper entitled 'Method for fault management with RPL protocol in WSNs (MFM-RPL)' presented at International Conference on Advanced Wireless, Information, and Communication Technologies (AWICT 2015), Tunisia, 5–7 October 2015.

1 Introduction

Recently, one of the main control methods for supervision regarding wireless sensor networks (WSNs) also called low power and lossy networks (LLNs) (IETF, http://www.ietf.org/dyn/wg/charter/roll-charter.html), is the fault and failure management in case of link and node failures. The faulty sensor node is one of the most important sources of faults in LLNs (Zhou et al., 2014). In the same times, the network should be able to detect faults and reacts more quickly to this anomaly and ensure the continuity of service monitoring. To ensure continuity of service for capture, collect and process data, it is necessary to implement a control strategy that meets their requirements while minimising energy consumption. Therefore, it is necessary that network failures are detected in advance, locate the faulty sensor nodes and appropriate measures are taken to continue network operations. In this context, the management and repair methods are a key feature for any routing protocol and refer to the ability to repair the routing topology when failures occur. To address the needs and problems of the failure in the LLNs, recently, the IETF ROLL working group (IETF, 2010) adopted a new protocol named IPv6 routing protocol for low power and lossy networks (RPL) routing protocol (Winter et al., 2012; Clausen et al., 2014). The RPL protocol specifies two techniques, which are complimentary in nature and actions in case of link and node failure (known as local and global repair, detailed in Section 3 (Vasseur et al., 2011). With the RPL a link or neighbouring node failure is detected and the repair mechanism is triggered. This integration provides a technique to repair the failure in the network, but in Vasseur et al. (2011) has shown that the RPL repair mechanisms introduce an additional complexity to the routing process and a cost of additional control traffic in the network by the effect of rebuilds in an RPL tree forced by node failures. In further, mobility for the RPL protocol has always been a challenging research topic because it was originally designed for static networks, with no support for mobility. An IPv6 distance vector routing protocol (RPL) designed for static WSN, can be adapted to our need. While a number of studies on RPL (Lamaazi et al., 2015) analyses the performance of the mobility method with different type of mobility to improve the RPL performance in dynamic network. The results of the performance assessment of the RPL protocol with mobile nodes in Lamaazi et al. (2015) show and prove that RPL can support the mobility method. However, handling the repair mechanism for the RPL with mobile nodes in sensor networks is a real challenge. In this way, our contribution consists in the design and implementation of a new method mobile node local repair with RPL (MN-LR_RPL) protocol to manage nodes failure in the network by proposing a new local repair technique with RPL protocol, using the mobility of parent nodes by a procedure of choice based on three cases:

1 case of the leaf node failure in the network topology

2 case of a node other than the leaf node failure such as the number of predecessor and/or successor of node failure is greater than zero

3 the case of a node such as its predecessor is the 'root'.

Extensive simulations with Cooja simulator (An Introduction to Cooja, https://github.com/Contiki-os/Contiki/wiki/An-Introduction-to-Cooja) under Contiki OS (http://www.Contiki-os.org), have been carried out to have a better idea on the behaviour of the proposed new repair method with RPL protocol by exploiting node mobility in the context of the fault and failure management in WSNs. Besides energy consumption and network latency, these experiments allow estimating other important metrics such as the packet delivery ratio and controlling traffic in the network. Behaviours of such a network in presence of our new method MN-LR_RPL is pointed out. This study shows MN-LR_RPL has the desired advantages of bounding control overhead, paket deliver and low delay. It also

provides quick repair of local outage in the network and presents path quality fairly close to an optimised shortest path compared to network normal state and the network with failure node. The remaining of the paper is structured as follow: we start by Section 2, to present a review of related works for mobility support method RPL based WSNs and the RPL local repair technique. Then we briefly describe the RPL protocol and the repair method in Section 3. In Section 4, we describe the repair technique with RPL protocol and the mobility method under RPL is explained in Section 5.

In Section 6, we address the detailed explanation of the proposed method MN-LR_RPL for local repair considering replacement mode by mobile nodes. In Section 7, the performance evaluation of the MN-LR_RPL method is presented. We give the results obtained from our experiments and evaluation study will be discussed in Section 8. Finally, conclusion and some perspectives are drawn in Section 9.

2 Related works

Since the appearance of IPv6 routing protocol for low power and lossy networks (RPL), several recent research assessments have studied the performance of this protocol in various scenarios based on simulations. Simulations and experimental models have been proposed by different papers.

In this section we will focus on the most important of these works in three research aspect for:

1 the performance evaluation of RPL in static mode

2 the performance evaluation of RPL in dynamic mode (mobility)

3 study RPL local repair.

For the first aspect, an in-depth study on the implementation of RPL was conducted to provide ideas and guidelines for the use of this standard (Vasseur et al., 2011; Zhang and Xianfeng, 2014), just to understand well its behaviour, and investigate its relevance using COOJA simulator under Contiki. In Gaddour and Koubaa (2014), the authors have performed a simulation study to evaluate the DAG construction process in RPL routing protocol. There is not much evaluation on local repair or fault management with RPL. The focus was only on the performance of this protocol.

In the second aspect, the RPL mobility support, the works (Lamaazi et al., 2015; Cobarzan et al., 2015; Gaddour et al., 2015; Gaddour and Koubaa, 2014; Ben Saad and Tourancheau, 2011; Carels et al., 2015) suggest the mobility method to improve the RPL performance in dynamic network. The author in Lamaazi et al. (2015) evaluate the performance of RPL in three different scenarios:

1 evaluate the characteristics of RPL with fixed nodes, (one sink and others are senders)

2 add mobility, it compares mobile nodes to fixed nodes in order to show how mobility can influence protocol parameters

3 study the behaviour of RPL when the network is dense in order to assess the protocol performances.

For this paper (Lamaazi et al., 2015), show clearly that the random waypoint (RWP) mobility model gives better metrics than random walk (RWK) mobility model in terms of number of hops and expected transmission (ETX) value. Also, in RWP model, nodes consume 10, 73% of power than RWK. The author in Cobarzan et al. (2015), propose a new cross-layer protocol operating at layers 2 and 3 known as mobility-triggered RPL (MT-RPL). To maintain efficient connectivity with the network, MT-RPL is evaluated together with neighbour unreachability detection and bidirectional forwarding detection through an extensive simulation campaign. The results of this new cross layer protocol show that MT-RPL significantly reduces the disconnection time, which increases the packet delivery ratio and reduces energy consumption per data packet. The study research in Gaddour et al. (2015), they focus mainly to propose some enhancements to the standard specification in order to provide QoS guarantees for static as well as mobile LLNs. With a new objective function (OF-FL), that overcomes the limitations of the standardised objective functions that were designed for RPL by considering important link and node metrics, namely end-to-end delay, number of hops, expected transmission count (ETX) and link quality level (LQL). It used in the same time the design of co-RPL explained and presented in Gaddour and Koubaa (2014), it proposes an extension to RPL to support mobility (co-RPL). To improve the network performance the extension keeps track of the mobile nodes positions while moving. To allow localisation of RPL nodes in motion the extension relies on the Corona mechanism via a simulation study using Contiki/COOJA simulator.

The most of works based on nodes mobility suggest improving the network lifetime or energy conservation (Ben Saad and Tourancheau, 2011), but with the sink mobility. In Carels et al. (2015), a new cross-layer protocol (MT-RPL) is proposed, this MT-RPL protocol benefits from the X-Machiavel MAC protocol that favours mobile nodes which want to transmit data. The new protocol achieves a packet for traffic from a mobile node towards a sink node. The paper also analyses the packet delivery for traffic from root to a mobile node. New research in Gaddour and Koubaa (2014) proposes RPL with enhanced neighbour discovery optimises the routing from a mobile node towards a static sink. This solution uses the DIS mechanism, an adapted link quality estimation function. In this second research aspect, they did not use the mobility for the RPL repair mechanisms in case of link and node failures.

The authors (Lamaazi et al., 2015; Cobarzan et al., 2015; Gaddour et al., 2015; Gaddour and Koubaa, 2014; Ben Saad and Tourancheau, 2011; Carels et al., 2015) did not use the mobility mechanisms for RPL repair in case of link and node failures, the objective of their research using

the mobility of nodes is different from one author to another, other than that used in our paper for the replacement of a node failure through our method MN-LR_RPL. Another important work (Islam et al., 2013) is focused to Link failures and packet drops due to congestion are two frequently occurring problems in mobile ad hoc networks. The author of Islam et al. (2013) propose a link failure and congestion-aware reliable data delivery (LCRDD) mechanism that jointly exploits local packet buffering and multilevel congestion detection and control approaches for increasing the data delivery performance.

For the third research aspect for the RPL local repair method, the repair becomes a key to relief for any routing protocol and refers to the ability to repair the routing topology. Korte et al. (2012) presents an experimental analysis of RPL error correction mechanisms by using the Contiki RPL implementation along with an SNMP agent to monitor the performance of RPL. In Tripathi et al. (2010), the performance evaluation metrics of RPL are simulated in a typical outdoor smart grid substation using real-life scenarios local repair. They did not use the mobility for the RPL repair mechanisms in case of link and node failure. The author in Yang et al. (2009) proposes an efficient algorithm for fault tolerant mobile agent and in Chen et al. (2016), the author address the fault tolerant service composition with particular attention to QoS. A fault tolerant strategy for improving the performance of service composition is proposed. The strategy is composed of invocation mechanism, synchronisation mechanism and exception mechanism. The management and application of mobility concept is presented and described in Liao et al. (2015) for mobile big data. Some mobile applications in heterogeneity environments are presented with the work (Chen et al., 2015).

Difference to previous works, this paper is aimed to focus on a totally autonomous and new method to exploiting the node mobility for the control and fault management in WSNs by MN-LR_RPL method using RPL as protocol.

This method provides the reconfiguration algorithm to replace the failure node by the mobile node in predecessor position. Despite the fact that several studies and implementations have been conducted to evaluate the performance of RPL, to our knowledge, there has been no evaluation of the local repair with RPL in the case of replacement nodes failure with mobile nodes.

3 Design overview of the RPL protocol

In this part, we present a brief overview of RPL protocol and explore the principal characteristics of RPL.

3.1 RPL protocol overview

RPL is a distance vector IPv6 routing protocol for LLNs that specifies how to build a destination oriented directed acyclic graph (DODAG). RPL is the routing protocol for applications over low-power and lossy networks (Winter et al., 2012). It is a key component of the layered IPv6-based architecture, is a source routing protocol that is designed to operate on top of IEEE 802.15.4 PHY and MAC layers (Figure 1). The RPL protocol organises nodes in a LLN as a directed acyclic graph (DAG) and partitions the DAG into one or more DODAGs, that is to say, oriented in the sense that each node sends packets to the sink (DAG root) and acyclic in that RPL guarantees the absence of loops in the graph. RPL used the objective function (OF) and a set of metrics and constraints (Clausen et al., 2014), the OF operates on a combination of metrics and constraints to compute the best path. In the DODAG, each node is assigned by a rank which determines its relative position in the graph and that represents its distance from the sink. This position is increases if nodes move away from the root and decreases when nodes move in the other direction, respectively. The rank is calculated as a function of the routing metric. The multiple DODAGs form the DAG INSTANCE, which a node can be a member of multiple DAG INSTANCEs but can belong to at most one DODAG per DAG INSTANCE as depicted in Figure 2.

Figure 1 Integration of the RPL protocol in the OSI architecture (see online version for colours)

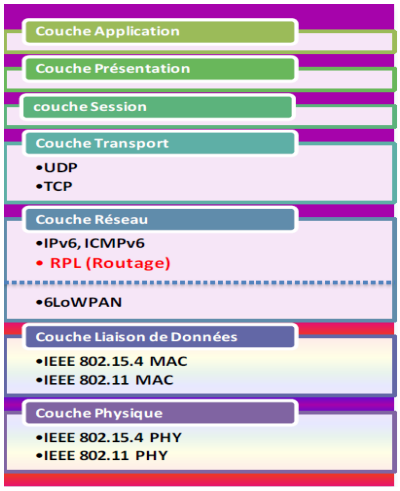
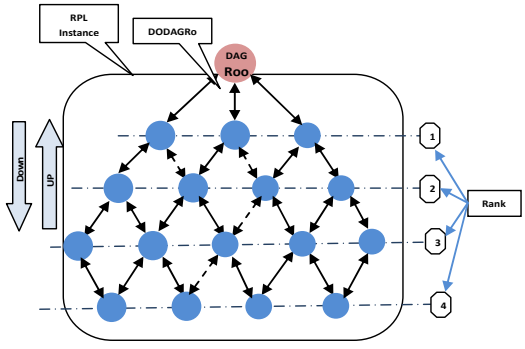


Figure 2 Design of the RPL protocol (see online version for colours)



The RPL protocol supports three traffic patterns: MP2P, point-to-multipoint (P2MP) and point-to-point (P2P) (Vasseur et al., 2011). The RPL protocol targets large WSN and supports a variety of applications, e.g., industrial, urban, home and buildings automation or smart grid. RPL used the OF and a set of metrics and constraints (Clausen et al., 2014).

3.2 Structure DODAG building process

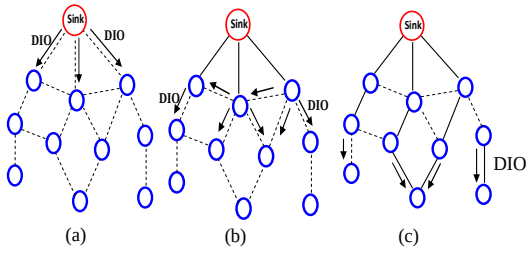
The graph building process starts at the root (sink), the DODAG construction is based on the neighbour discovery (ND) process, which consists in two main operations:

1 transmission of DODAG information object (DIO) control messages issued by the DODAG root to build routes in the downward direction from the root down to nodes,

2 broadcast of DAO control messages issued by nodes and sent up to the DODAG root to build routes in the upward direction.

In order to construct a new DODAG, two control packets are used, called DIO and DODAG information solicitation (DIS) to convey the DODAG information. The sink starts sending all its neighbours (nodes in range) DIO message, to announce its DODAGID, its rank and the OF. The nodes in the listening vicinity of the root will receive, process and make a decision based on the OF, whether to join the graph or not. When a node receives DIO, he chooses the best parent (preferred) from the list of candidates parents. This choice can be made using different metrics (minimum number of hops, Qos, and remaining battery). Each node has chosen its parent 'preferred'. The neighbouring peers will repeat this process and do parent selection. The node computes the 'rank' of itself within the graph, which indicates the 'coordinates' of the node in the graph hierarchy as show in Figure 2. In this formation each node of the graph has a routing entry towards its parent (or multiple parents depending on the objective function) in a hop-by-hop fashion and the leaf nodes can send a data packet all the way to root of the graph by just forwarding the packet to its immediate parent (Vasseur et al., 2011), The steps of the graph building are represented in Figure 3.

Figure 3 DODAG construction (a, b and c) (see online version for colours)



The implementation of RPL protocol reposes on expected transmission (ETX) as a routing metric. ETX defines a successful transmission as receiving an acknowledgement in response to a unicast transmission, which the minimum ETX value specifies the best path that is selected by RPL. ETX also serves to indicate bi-directional link quality. The use of the ETX metric in RPL networks will help the RPL nodes to use more reliable paths to reach the root (Vasseur et al., 2011; Zhang and Xianfeng, 2014).

The transmission of DIO is regulated by the algorithm 'trickle' (Contiki Operating System, http://www.Contiki-os.org).

3.3 The control messages in RPL

The RPL routing protocol specifies a set of new ICMPv6 control messages for the connectivity of nodes with exchange graph related information. These messages are called DIS (Vasseur et al., 2011).

- DIS: a DIS transmission indicates a nodes intention to discover a new route
- DIO: a DIO transmission indicates a nodes route information change or a solicitation from a neighbouring node for discovering new route
- DAO: a DAO transmission indicates a nodes route information change or route information change of another node in the nodes sub-DODAG.

4 The repair technique with RPL protocol

In this section, we describe the repair techniques defined by the RPL protocol, after we explain our contribution for the new local repair method exploiting mobile nodes in WSNs.

4.1 The global and local repair for RPL

RPL has two mechanisms to repair the topology of DODAG, which are complementary, the global and local repair. The global repair is triggered only from the root, it is a graphic reconstruction mechanism from scratch. The DAG root begins incrementing the version number of the DODAG when sending DIO for a new DODAG Version (Vasseur et al., 2011). When nodes receive DIO, they accept only the DIO whose version number is greater than or equal to the current number. This version number ensures that information circulating in the network is up to date, to the extent that the former DIO are 'crush' with the most recent. The global repair is an optimisation technique, but it has a cost of additional control traffic in the network and delay to repair depends on sequence number refresh rate. The local repair is complementary to the global, subsequently the global repair. Which allow the DODAG repaired within the DODAG Version. In which Each node can detach from the DODAG, moves its parent to its routing table, local poison the sub-DODAG by advertising the rank of infinity, and the end, it re-attach to the original or a new-branch DODAG. This technique presents the risk of creating a loop and count

infinity, it used DAG hop timer to wait for poisoning to occur (Tripathi et al., 2010). The local repair is not implemented in the RPL module with Contiki RPL (Contiki Operating System, http://www.Contiki-os.org).

5 The mobility under RPL protocol

Since its appearance, the RPL protocol is greatly satisfied the requirements of low power and lossy sensor networks, several issues remain open for improvement and specification, in particular with respect to support for mobility method for different needs of WSNS. In addition, RPL was originally designed for static networks, with no support for mobility. However, handling the repair mechanism for the RPL with mobile nodes is a real challenge. Serious works was recently published (Lamaazi et al., 2015; Cobarzan et al., 2015) which prove stability and mobility support by the RPL protocol. This encouraged us to propose the method of mobility to be exploited for efficient local repair without rebuilding graph topology. However, to replace nodes failures by the mobility of their predecessor without rebuild of the RPL tree, is a proposed solution to estimate the ensure continuity of service for collect and transmit data in the monitored environment.

For instance, the fact that nodes are supposed to be mobile, redundant node is deployed within the network by the list of the parent nodes which RPL protocol defines in the routing process. Through the MN-LR_RPL method for fault management in LLNS, we propose some enhancements to the standard specification in order to provide fault management in WSNs presented in static topology as well as propose the mobility for repair this anomaly. The following section describes the proposed MN-RKL RPL method.

6 The MN-LR_RPL

This section describes our main contribution by offering a new local repair method, exploiting the nodes mobility in wireless sensors network with RPL protocol (MN-LR_RPL). RPL protocol defines the notion of the parent node as presented in our method by the predecessor node. In the tree RPL topology, each node router identifies a stable list of parents as well as a preferred parent. Each RPL router is a potential next-hop on a path towards the ‘root’ of the DODAG. In our MN-LR_RPL method, the predecessor node presents the one of the list of parent nodes defined with RPL protocol and the list of multiple parent nodes depending on the single OF (Vasseur et al., 2011). The leaf nodes can send a data packet all the way to root of the graph by just forwarding the packet to its immediate parent. One node of the list of the parent nodes is consider as preferred parent for traffic transitions all the way to the root (sink). For our MN-LR_RPL method we used the list of nodes parent and choose one which satisfied the hypothesis of the procedure 1, defined in our algorithm as presented below. The nodes are supposed to be mobile to replace the failure nodes, redundant nodes are deployed within the network in the list of parent nodes.

To manage failure nodes in the network, MN-LR_RPL processes three cases for nodes failure in a WSN by proposing a new local repair technique with RPL protocol, using mobility of parent nodes by a procedure of choice based on:

1 case of the leaf node failure in the network topology

2 case of a node other than the leaf node failure such as the number of predecessor and/or successor of node failure is greater than zero

3 the case of a node such as its predecessor is the ‘root’.

The algorithm proposed in this paper allows at the first time, with the RPL detecting mechanism of the node failure in the network. In the second step, the solution is focused to replace the failure node by mobile node in the three cases of failure which are already mentioned above in Section 6.

The MN-LR_RPL algorithm incorporates three new rules, in first time to detect the node failure (defined by S as show in Table 1) for the two cases in the network (leaf node and parent node). In the second step the solution is to replace nodes failures by the mobility of their predecessor nodes [defined by Pred(S) as show in Table 1] for collection ability and data routing. This is predetermined by the rules 1, 2 and 3 further defined below.

Table 1 Definition of rules

Nodes	*Definition*
S	Failure node
Pred(S)	Predecessors of S
Succ(S)	Successors of S
L_{Pred}	List of predecessors
L_{Succ}	List of successors
M	Mobile node

6.1 Definition of rules

The MN-LR_RPL algorithm use the different nodes defined in Table 1.

6.2 Assumption of rules

- *Rule 1*: Case of the leaf node failure (Figure 4).
  1 choose one node of the list L_{Pred} of Pred(S) using procedure 1 (Table 2)
  2 let M be the chosen predecessor
  3 delete S,
- *Rule 2*: The case of a node other than the leaf node failure such as Pred(S) > 0 and/or Succ(S) > 0 (Figure 5).
  1 let M the chosen predecessor of list L of Pred(S) using procedure 1
  2 Succ(S) become the successors of M

3 Pred(S) become predecessors of M

4 delete S,

- *Rule 3*: The case of a node such as its predecessor is the 'root' (Figure 6).

1 let M the chosen successor of list L of Succ(S) using procedure 1 as chow in Table 2

2 Succ(S) become successors of M

3 successor of 'root' become M

4 delete S.

Figure 4 Topologies in case of leaf node failure (Rule 1) (see online version for colours)

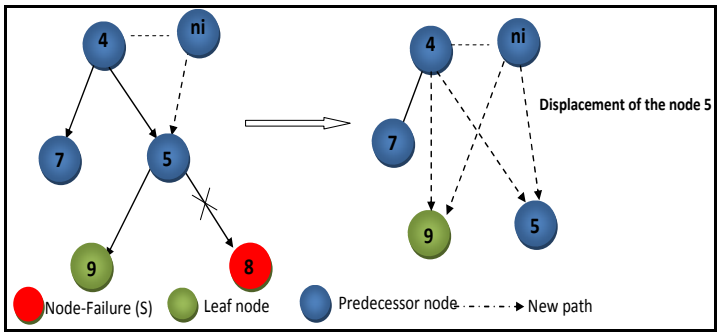


Figure 5 Topologies in case of a node other than the leaf node (Rule 2) (see online version for colours)

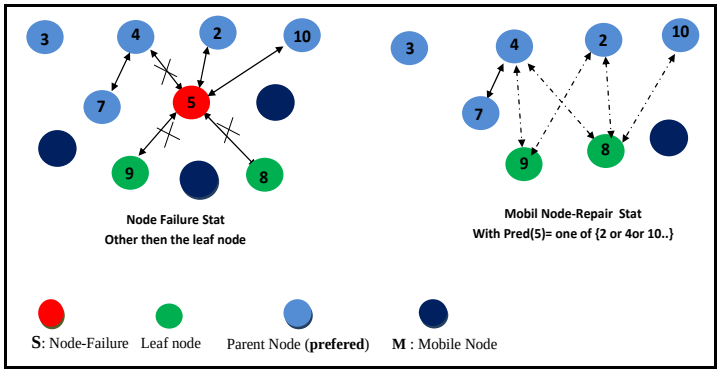


Figure 6 Topologies in case of a node such as its predecessor is the 'root' (Rule 3) (see online version for colours)

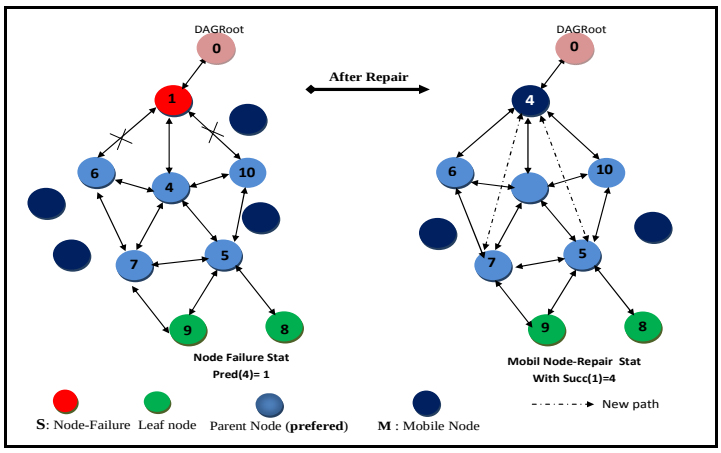


Table 2 Procedure 1

Procedure 1
Choose m one of list L such as this node maximises a function f.
f = p1, p2, p3, p4,..... pi parameters, p1 = energy, p2 = number successor, p3 = number of predecessor, ...etc.
f (p1, p2) can be the function which choose the node(n) which have the highest energy and the smallest number of successors.

6.3 Mathematical formulation for MN-LR_RPL

In this section we explain the mathematical function which is using with the weighted sum for the procedure of MN-LR_RPL, we suppose to define:

The Procedure 1 in Table 2, defines the function f, which the p1, p2, p3, p4,..... pi, are the parameters used to select the nodes (successor or predecessor)

- m: Actions {A1, A2, …. A_m], that represent the Rule 1, Rule 2 and Rule 3
- n: decision criteria {C1, C2….C_n}, that represent the pi parameters as presented with procedure 1 in Table 2, p1 = energy, p2 = number successor, p3 = number of predecessor, ...etc.
- w_j denotes the relative weight of importance of the criterion C_j
- a_{ij} is the performance value of alternative A_i when it is evaluated in terms of criterion C_j.

Then, the total (i.e., when all the criteria are considered simultaneously) importance of alternative A_i, denoted as $R(a_i)$, is defined as follows [equations (1) to (3)]:

$$R(a_i) = \sum_{j=1}^{n} w_j a_{ij} \quad \forall i \in [1, m],\ w_j > 0 \tag{1}$$

In case of our MN-LR_RPL algorithm the values of m and n are:

$$m = 3,\ \{\text{Rule 1},\ \text{Rule 2},\ \text{Rule 3}\}$$

n = {p1, p2, p3}, p1 = energy, p2 = number of successor, p3 = distance between nodes neighbouring.

Then:

$$a_{ij} = U_j(A_i) \Rightarrow a_{ij} = U_j(R_i) \tag{2}$$

The weighted sum model is:

$$f(a_{ij}) = \sum_{j=1}^{n} (w_j a_{ij}) \quad \forall i \in [1, m] \tag{3}$$

7 Performance evaluation

We simulate MN-LR_RPL under Cooja (Levis et al., 2010), is a Java-based simulator designed for simulating sensor networks running the Contiki sensor network operating system (Tripathi et al., 2010).

The evaluation of the rules of the MN-LR_RPL algorithm is performed separately, because we considered that the failure may occur depending on the three cases already explained above.

The objective of these assessment scenarios according to the three rules is to study the possible cases of the existence of the fault depending on the position in the network, either leaf node (Rule 1), in other than the leaf node (Rule 2), with node such as its predecessor is the 'root' (Rule 3).

The simulator allows sensor node software to be written in C. The energy model used by the Contiki operating system takes into account the energy consumption of listening conditions, transmission and rest of the radio component. The network we have simulated contains 100 nodes and a single sink node, deployed in an environment of 200 m × 200 m. For our evaluation, we use ETX as the objective function; ETX may very well be replaced with the other metrics such as hop count, latency, throughput, etc. Because ETX defines a successful transmission as receiving an acknowledgement in response to a unicast transmission (Vasseur et al., 2011).

7.1 Performance metrics

For the evaluation of our MN-LR_RPL algorithm in this paper, we used the performance metrics in the following:

- CollectView: it represents the average packets received from the node (mote) to CollectView (DAGroot).
- Network latency: this performance metric is defined as the average of the latencies [equation (3)] of all the packets in the network from all the nodes, the latency of packets [equation (4)], is the amount of time taken by a packet from node to reach the sink.

$$\text{Total latency} = (\text{RecvTime}(n) - \text{Sent Time}(n)) \tag{4}$$

$$\text{Average latency} = (\text{Total latency} / \text{Pakcets received}) \tag{5}$$

- Packet delivery ratio (PDR %): it measures the ratio of total packets received against total packets sent in the MAC sublayer, the PDR is computed by equation (5).

$$\text{Average PDR} = (\text{Packets received} / \text{Packets sent}) * 100 \tag{6}$$

- Energy consumption: presents the measurement of the energy which is dissipated by the sensors node for transmission data to the sink. We use the percent radio on time of the radio which dominates the power usage in sensor nodes.
- Control traffic overhead: this includes DIO, DIS and DAO messages generated by each node and it is imperative to confine the Control Traffic keeping in mind the scarce resources in LLN.

$$\text{Control traffic overhead} = \sum_{n=1}^{k} (\text{DIO} + \text{DIS} + \text{DAO}) \quad (7)$$

8 Network simulation parameters

The general parameters used in the simulation scenarios are summarised in Table 3. We simulate the MN-LR_RPL algorithm in two scenarios:

1 simulation of rules separated

2 simulation the global algorithm (Rule 1, Rule 2, Rule 3).

Finally, we discuss the different results obtained at the time of this simulation.

Table 3 General simulation parameters

Parameters	*Values*
Network simulator	COOJA under Contiki OS (2.6.1)
Radio environment	Unit disk graph medium (UDGM)
Simulation time	1 hour
Client nodes	100 and 1 sink
Mote type	Tmote Sky
Duty cycle	ContikiMAC
PYH and MAC layer, network layer	IEEE 802.15.4, uIPv6
Objective function	ETX
RX ratio	30, 40, 50, 60, 70, 80, 90, 100
Max packets	32,583
DIO min	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

9 Analysis of the simulation results

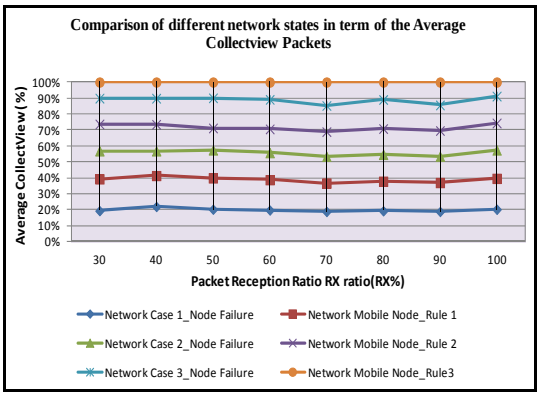
In the following paragraphs, we evaluate the performance of our proposed algorithm and analyse its cost by measuring the five parameters: Average CollectView, network latency, packet delivery ratio, energy consumption, control traffic overhead. We run each simulation for a sufficient time until obtaining a stable network.

9.1 Simulation of MN-LR_RPL algorithm in separated rules (Rule 1, Rule 2, Rule 3)

For this experiment, we simulate the MN-LR_RPL algorithm with separated rules (Rule 1, Rule 2, Rule 3). In this case, we increase the packet reception ratio to 30, 40, 50, 60, 70, 80, 90, 100) and we compare the average CollectView packets received by the DAGroot and the average power consumption to each rules (Rule 1, Rule 2, Rule 3) executed in the network topology. The simulation results are depicted in Figures 7 to 11.

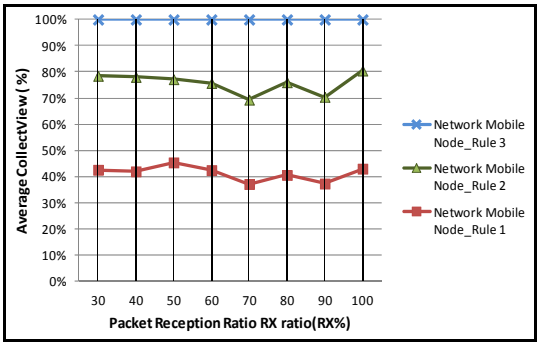
We can see in Figure 7 that for each failure network state as the case presented in Section 6, correspond the execution of the rules (Rules 1, 2 and 3) to the local repair by mobile node. For the network state in case 1 (leaf node failure), the average CollectView is 20% comparing to 40% for the Rule 1.

Figure 7 Average CollectView comparison of the different states network topology (failures and separated rules) (see online version for colours)



In addition, by observing the Rule 2 is 73% compared to the failure network state in the case 2 which report a loss of 20%. The same observation for the Rule 3, when the average CollectView presents good results according to collect 100% compared for the failure network state in case 3 is under than 10% loss. When the rules are executed separately in time with failures case in the network, the MN-LR_RPL algorithm presents the good results for the average CollectView in the all network states with local repair. Figure 8 depicts the average CollectView comparison for the three rules; we can be observed that the result of Rule 3 is performed better than the other rules with 100%. Which is the case of this repair is regarding the repair of the nearest node of the sink.

Figure 8 Average CollectView comparison Rule 1, Rule 2, Rule 3 in network (see online version for colours)



For the Rule 2 the average CollectView is varied between 70% and 80%. Moreover, the average collect view for the Rule 1 is more 40% and less of 50%. It can be observed that our proposed algorithm by the separated rules is performing a quicker recovery. Figures 9 to 11, present the average power consumption for each rules.

Figure 9 The average power consumption in MN-LR_RPL network with Rule 1 (see online version for colours)

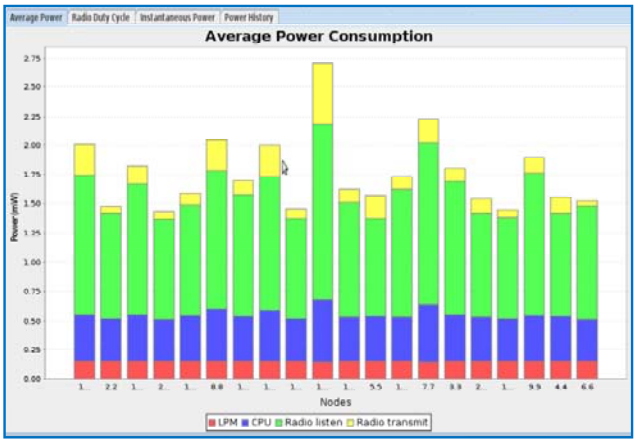


Figure 10 The average power consumption in MN-LR_RPL network with Rule 2 (see online version for colours)

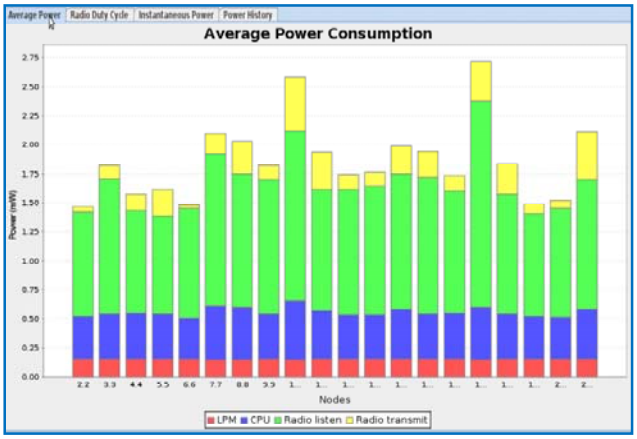
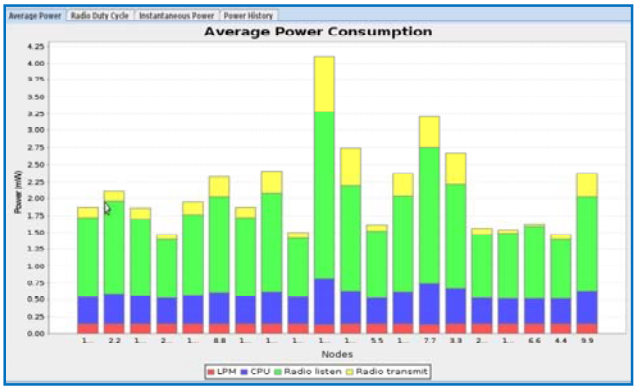


Figure 11 The average power consumption in MN-LR_RPL network with (see online version for colours)



9.2 Simulation the global MN-LR_RPL algorithm (Rule 1, Rule 2, Rule 3)

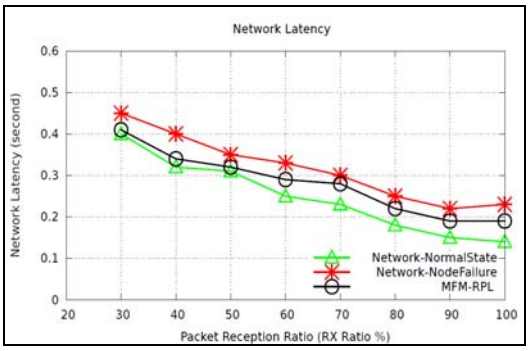
In the first scenario, we simulate the global MN-LR_RPL protocol algorithm by two experiments such as:

- the impact of the packet reception ratio with MN-LR_RPL
- the impact of the DIO interval min with MN-LR_RPL.

9.2.1 Impact of the packet reception ratio (RX%) with MN-LR_RPL

In the second experiment, we fixed the value of the DIO interval minimum to 12 as default value in RPL protocol and we varied the RX by 30, 40, 50, 60, 80, 90 and 100 (%). We used the ETX objective function to computing the best paths. The simulation results are presented in Figure 12, as shown the latency metric result the MN-LR_RPL performs better than the network-Normal state, knowing that RPL provides objective function for ETX to computing the best paths (Vasseur et al., 2011). When the RX increases (known as the loss packet), the latency decreases, it means that the MN-LR_RPL offers us another positive characteristic that of finding in minimum delays the best path with ETX function objective as defined in the standard.

Figure 12 The impact of the packet reception ratio (RX%) in different states network topology (normal, failure and MFM-RPL) on network latency (see online version for colours)



From Figure 13, with the increasing the RX ratio in the network, the consumption of the energy decreases as slowly for the different state network, the performance of MN-LR_RPL prove the effectiveness of fault management method and ensure service continuity control in LLN. The causes of the more consumption energy in the start simulation are for the retransmissions of the packets lossless.

9.2.2 Impact of the DIO interval min with MN-LR_RPL

For the first experiment, we used the variation of DIO minimum interval as evaluation performance metrics the MN-LR_RPL algorithm which is our objective. The simulation parameters used in this simulation are shown in Table 3. We varied the DIO values to (2, 3, 4–16) in iterations of the simulation and put RX ratio to 70%. The simulation results are depicted in Figures 14, 15, 16 and 17 for the performance metrics to the studied metric.

Figure 13 The impact of the packet reception ratio (RX%) in different states network topology (normal, failure and MN-LR_RPL) on energy consumption (see online version for colours)

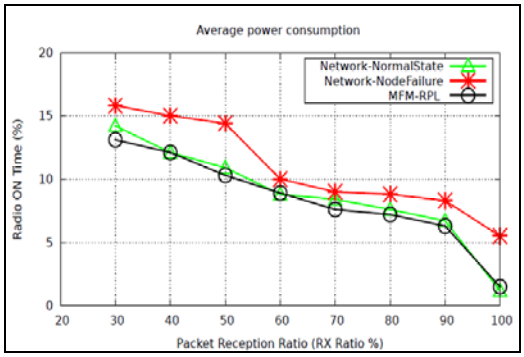
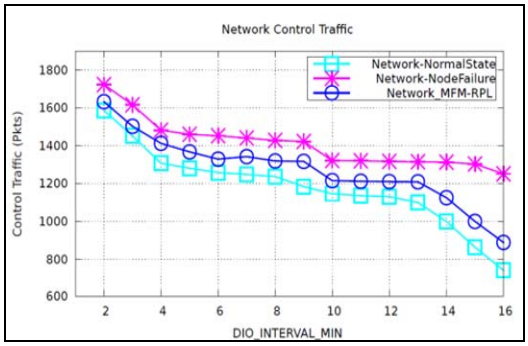


Figure 14 The impact of DIO interval minimum in different states network topology (Normal, failure and MN-LR_RPL) on: control traffic overhead (see online version for colours)



The results in Figure 14, shows the comparison between the three states of the network:

1 network normal state

2 network with the node failure state

3 network with the MN-LR_RPL state with the Rule 1, Rule 2 and Rule 3 as presented in Section 6.

So we note that from the tree states of the network, the control traffic overhead is very high for lower DIO Min (2 to 8) and decreases rapidly in the presence of 9 to 16 DIO Min, which is the optimum set for control traffic overhead in RPL protocol normal state.

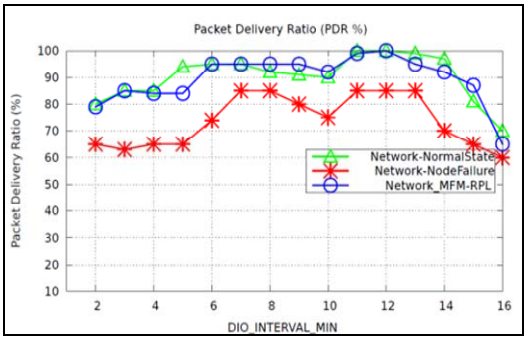
In addition, the control overhead for the normal state and MN-LR_RPL are remarkable for approximating values

during construction of the DAG and data transmission. This is different with the values that are in increase in nodes failure state.

This is related to the failure and the absence of the node in the networks, which retransmission ICMPv6 control packets (DIO messages) by each node is increased, so that the node responds to join the DAG and establishes such transmissions. These comparison results show the efficiency of our proposition MN-LR_RPL method for the local repair in WSN.

From Figure 15, which represents the PDR(%) shows the degradation of Network performance for DIO Mini between 2 and 6 and 13–16 for the node failure state in network, PDR below 75%, that means due to the failure and absence node in different cases (presented in Section 5) and the higher control overhead chewed in Figure 14.

Figure 15 The impact of DIO interval minimum in different states network topology (normal, failure and MN-LR_RPL) on: PDR (%) (see online version for colours)



On the other hand, the MN-LR_RPL provides a good Packet Delivery ratio of more than 95%, by the effect of the replacement of nodes failure by the mobile node. This result proved the effectiveness of fault management method to ensure service continuity control in the network.

We see in Figure 16, the low DIO Min (2 to 6) decreases slowly with minimum value for the latency performance metric, and its approximately equal for the network normal and MN-LR_RPL states. This can be explained by the increasing of the DIO min and the good Packet Delivery ratio (more 95%) and the lowest of the control traffic overhead.

We can also note the big difference of comportment for the Network-node failure state, when the latency value is decreasing but is greater than the values of the network normal stat and MN-LR_RPL. This is caused by the effect of the nodes failure and the control traffic augmented in the network. Figure 17 shows that the consumption of total energy is important when low of DIO interval min (2, 3), the network consumes a lot of energy about 2.3% ratio on time for the MFM-RPL, 2% for RPL network in normal state and 2.55 for node failure state.

Figure 16 The impact of DIO interval minimum in different states network topology (normal, failure and MFM-RPL) on: network latency (see online version for colours)

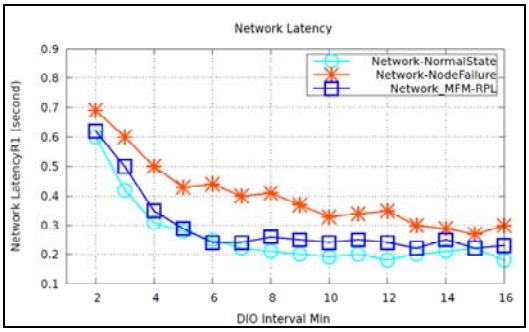
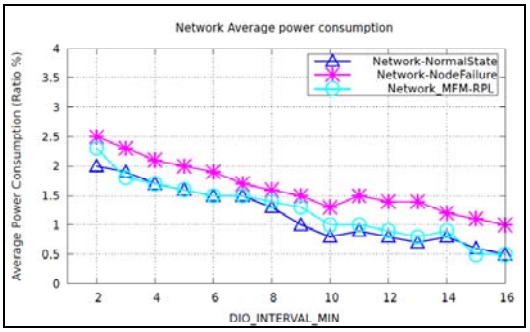


Figure 17 The impact of DIO interval minimum in different states network topology (normal, failure and MFM-RPL) on: energy consumption (see online version for colours)



10 Conclusions and future work

In this paper, we have proposed a new technique for fault management in RPL-based WSNs, with a new local repair mechanism while exploiting mobile nodes, as known MN-LR_RPL method. The performance evaluation of our experimentations was conducted on the Cooja simulator. The network latency, control traffic, the consumption energy and the packet delivery ratio were analysed as performance metrics. The experimentations results obtained in this paper have proved the effectiveness of MN-LR_RPL method to ensure the service continuity guaranteed in the network. The main conclusion from this study proposition is that MN LR_RPL has several benefits for RPL in LLN networks in term of new proposition technique for the local repair in RPL. Compared to the current local repair which offers the RPL protocol. This technique reacts quickly to replace the node failure by their node predecessor with mobility. Based on our new method, some opportunities are offered for MN-LR_RPL implemented in vehicular ad hoc networks (VANETs) applications with differentiating traffic. We can suggests to compare this technique with

others techniques for the local repair with RPL protocol if exist recently and with others techniques for fault and failure management in WSN. In addition and in another work, we propose a new control strategy using the software defined network (SDN) concept, to improve the performance of RPL protocol. Finally we hope in future work for my research work to achieve real testbed experimentation for MN-LR_RPL method.

References

An Introduction to Cooja [online] https://github.com/Contiki-os/Contiki/wiki/An-Introduction-to-Cooja (accessed December 2014).

Ben Saad, L. and Tourancheau, B. (2011) 'Sinks mobility strategy in IPv6-based WSNs for network lifetime improvement', in *International Conference on New Technologies, Mobility and Security (NTMS)*, IFIP, Paris, France [online] http://hal.inria.fr/inria-00563724.

Carels, D., De Poorter, E., Moerman, I. and Demeester, P. (2015) 'RPL mobility support for point-to-point traffic flows towards mobile nodes', *International Journal of Distributed Sensor Networks*, Article ID 470349, 13pp. [online] http://dx.doi.org/10.1155/2015/470349.

Chen, D., Zhu, X., Dai, W. and Zhang, R. (2015) 'Socially aware mobile application integrations in heterogeneous environments', *J. of High Performance Computing and Networking*, Vol. 8, No. 1, pp.6 –70.

Chen, L., Fan, G. and Liu, Y. (2016) 'A formal method to model and analyse QoS-aware fault tolerant service composition', *Int. J. of Computational Science and Engineering*, Vol. 12, Nos. 2/3, pp.133–145.

Clausen, T., Yi, J., Herberg, U. and Igarashi, Y. (2014) *Observations of RPL: Ipv6 Routing Protocol for Low Power and Lossy Networks*, Network Working Group, May [online] http://tools.ietf.org/html/draft-clausen-lln-rpl-experiences-08 (accessed September 2014).

Cobarzan, C., Montavont, J. and Noel, T. (2015) 'Integrating mobility in RPL', *Wireless Sensor Networks, Lecture Notes in Computer Science*, Springer International Publishing, Vol. 8965, pp.135–150.

Contiki Operating System, 'The open source OS for the internet of things' [online] http://www.Contiki-os.org (accessed December 2014).

Gaddour, O. and Koubaa, A. (2014) 'Co-RPL: RPL routing for mobile low power wireless sensor networks using corona mechanism', I*ndustrial Embedded Systems (SIES), 9th IEEE International Symposium*, pp.200–209.

Gaddour, O., Koubâa, A. and Abid, M. (2015) 'Quality-of-service aware routing for static and mobile IPv6-based low-power and lossy sensor networks using RPL', *Ad Hoc Networks, Sci. Direct J*, October, Vol. 33, No. C, pp.233–256.

Internet Engineering Task Force (IETF) *Routing over Low Power and Lossy Networks* [online] http://www.ietf.org/dyn/wg/charter/roll-charter.html.

Islam, M.M., Razzaque, M.A., Bosunia, M.M.R., Alamri, A. and Hassan, M.M. (2013) 'Link failure and congestion-aware reliable data delivery mechanism for mobile ad hoc networks'; *Annals of Telecommunications*, October, Vol. 68, No. 9, pp.539–551.

Korte, K.D., Sehgal, A. and Schönwälder, J. (2012) 'A study of the RPL repair process using ContikiRPL, dependable networks and services', *6th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, Lecture Notes in Computer Science*, Springer International Publishing, Vol. 7279, pp.50–61.

Lamaazi, H., Benamar, N., Imaduddin, M. and Jara, A. (2015) 'Performance assessment of the routing protocol for low power and lossy networks', *The International Conference on Wireless Networks and Mobile Communications (WINCOM)*, Morocco, October.

Levis, P., Clausen, T., Hui, J., Gnawali, O. and Ko, J. (2010) *The Trickle Algorithm*, IETF, draft-ietf-roll-trickle-03, August.

Liao, Z., Yin, Q., Huang, Y. and Sheng, L. (2015) 'Management and application of mobile big data', *Int. J. of Embedded Systems*, Vol. 7, Nos. 3/4, pp.318–323.

The Internet Engineering TaskForce (IETF) (2010) [online] http://www.ietf.org/.

Tripathi, J., de Oliveira, J.C. and Vasseur, J.P. (2010) 'Applicability study of RPL with local repair in smart grid substation networks', *First IEEE International Conference on Smart Grid Communications*, pp 262–267.

Vasseur, J.P., Agarwal, N., Hui, J., Shelby, Z., Bertrand, P. and Chauvenet, C. (2011) *RPL: The IP Routing Protocol Designed for Low Power and Lossy Networks*, Internet Protocol for Smart Objects (IPSO) Alliance, April.

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, S.R., Vasseur, J. and Alexander, R. (2012) *RPL: Ipv6 Routing Protocol for Low-power and Lossy Networks*, IETF ROLL Working Group, March [online] http://datatracker.ietf.org/doc/rfc6550/ (accessed October 2014).

Yang, J., Cao, J., Wu, W. and Xu, C.Z. (2009) 'Efficient algorithms for fault tolerant mobile agent execution', *Int. J. of High Performance Computing and Networking*, Vol. 6, No. 2, pp.106–118.

Zhang, T. and Xianfeng, L. (2014) 'Evaluating and analyzing the performance of RPL in Contiki', *The 1st international workshop on Mobile sensing, computing and communication (MSCC)*, Philadelphia, PA, USA, August 11, pp.19–24.

Zhou, Y., Zhou, S. and Luo, J. (2014) 'Explore city dynamics from the communication network between base stations', *Int. J. of Embedded Systems*, Vol. 6, Nos. 2/3, pp.140–147, DOI: http://dx.doi.org/10.1504/IJES.2014.063811.

INDERSCIENCE
PUBLISHERS
www.inderscience.com

IEL Editorial Office • PO Box 735 • Olney • Bucks • MK46 5WB • UK
Fax: (UK) +44 1234-240515 • E-mail: info@inderscience.com

International Journal of
High Performance Computing and Networking (IJHPCN)
Editor-in-Chief M A Dorgham BSc MSc PhD CEng PEng MIMechE Mem ASME MSAE

Djamila Bendouda
Oran I Ahmed Benbella University
Faculty of Science
Department of Computer Science
El M'Naouar
Oran, Algeria

18 September 2017

Dear Djamila Bendouda

Exploiting Node Mobility for Fault Management in RPL-based Wireless Sensor Networks

I am pleased to inform you that the above paper, by Djamila Bendouda, Lynda Mokdad and Hafid Haffaf, has been accepted for publication in the International Journal of High Performance Computing and Networking.

The paper will appear in a forthcoming issue.

Yours sincerely,

Richard Sharp
Journal manager, IJHPCN
Inderscience Publishers

2017 IWCMC June 26th - 30th, 2017	The 13th –IEEE International Wireless Communications and Mobile Computing Conference http://iwcmc.org/2017

Holiday Inn Express Hotel, Valencia, Spain

Official Acceptance Letter

Date: April 10, 2017

Author's Names: *Djamila Bendouda, Abderrezak Rachedi & Haffaf Hafid*

This letter is to certify that the mentioned author(s) above have an **ACCEPTED** paper(s) in the **13th — IEEE-IWCMC 2017** Conference **entitled**: *"An hybrid and proactive architecture based on SDN for Internet of Things"*

The ***I**nternational **W**ireless **C**ommunications and **M**obile **C**omputing* Conference 2017 (**13th — IEEE-IWCMC 2017**) will be held on: **June 26th – 30th 2017**, in ***Holiday Inn Express Hotel, Valencia, Spain***.

IEEE requires that each accepted paper **MUST** be presented **orally** in order to be included in the conference proceedings and the IEEE Xplore database.

Please feel free to contact me if you need any further information.
Sincerely,

Prof. Jaime Lloret Mauri
13th — IEEE-IWCMC 2017 General Chair
Universitat Politècnica de Valencia, Valencia, Spain
Tel. (+34) 60 954 9043 — **Email**: jlloret@dcom.upv.es // info@iwcmc.org

Résumé

L'*Internet des Objets* (*IdO*) est un nouveau concept révolutionnaire qui connaît un succès rapide dans le domaine des télécommunications sans fil modernes. Ce concept donne naissance à des réseaux numériques omniprésents au sein de l'espace physique, c'est la force vitale en réseau de plusieurs domaines de la société numérique pour améliorer les procédures de déploiement, d'exploitation et d'industrialisation. Parmi ces domaines, la ville intelligente, le transport intelligent, le smart grid, l'industrie, etc. L'idée de base de l'*IdO* est la présence omniprésente, autour de nous, d'une variété d'objets intelligents ou d'objets via Internet, qui peuvent être considérés comme des capteurs, des actionneurs, des étiquettes RFID, des téléphones portables, des machines, des appareils, des véhicules et les personnes équipées de capacité de communication filaire ou sans fil. Ces objets intelligents, grâce à des schémas d'adressage uniques, sont capables d'interagir entre eux et de coopérer avec leurs voisins pour atteindre des objectifs communs. L'*IdO* est très largement structuré par les infrastructures disponibles tels que les RCSF et des protocoles Internet. L'un des principaux défis de recherche concernant les réseaux de capteurs sans fil (*RCSFs*) dans l'*IdO* est le contrôle et la gestion du réseau.Cette thèse se focalise sur deux aspects de recherches, l'un se base sur la gestion des *RCSFs* basés sur la technologie *6LowPAN* pour garantir sa fonctionnalité tandis que le deuxième aspect traite le concept de contrôle via une architecture programmable. L'objectif de ce travail consiste en premier temps à proposer une nouvelle méthode efficace pour la gestion de la défaillance des noeuds avec le protocole *RPL*, tout en mettant en oeuvre un algorithme de réparation locale avec le concept de la mobilité pour remplacer ces noeuds en question. La mobilité est pertinente avec un protocole de routage, ainsi que la réparation locale des *RCSFs* basés sur la technologie *6LoWPAN* qui n'est offerte qu'avec le protocole *RPL*. Cette technique de réparation locale est réalisée par l'algorithme proposé *MNLR_RPL*, dont les performances ont été validées à l'aide du simulateur *Cooja* sous *Contiki 6.2* à travers plusieurs évaluations effectuées. Le deuxième aspect de cette thèse, permet de proposer une nouvelle architecture programmable de contrôle basée sur le paradigme du *SDN* (*réseaux définis par logiciels*) proactive et semi distribuée avec trois niveaux de contrôle : les contrôleurs Principal, Secondaire et Local. Avec comme objectifs d'améliorer la fonctionnalité du contrôle dans les *RCSF* tout en respectant l'hétérogénéité des technologies de réseau et les caractéristiques des ressources limitées des *RCSFs* dans l'*IdO*. Notre objectif, à travers cet aspect, est de réduire les messages de signalisation, les trafics de contrôle liés aux mécanismes de contrôle du réseau, et d'éviter la probabilité de défaillance du seul point de contrôle : celui du contrôleur central du *SDN*. En effet, nous introduisons un nouveau mécanisme de Contrôleurs Locaux (*CLs*) basé sur des ensembles *CDS* (*Connected Dominating Sets*) et des approches de la logique floue. Ce mécanisme est basé sur le développement de notre méthode proposée *DLC-CDS* (*Distributed local Controler- Conected Dominating Set*) pour la sélection des dominants qui vont jouer le rôle de contrôleurs locaux connectés dans l'architecture proposée. Les performances de la méthode de contrôle proposée ont été validées à l'aide du simulateur Matlab à travers plusieurs évaluations effectuées en comparaison avec des méthodes existantes.

Mots clés :

Internet des Objet (*IdO*); *RCSF*; Protocole *RPL*; Réseaux définis par logiciels (*SDN*); *CDS*; Logique flou; Mobilité; 6LoWPAN; Cooja; Contiki.