République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Centre Universitaire Belhadj Bouchaib D'Ain Témouchent

Institut des Sciences
Département des Mathématiques et de l'Informatique

Mémoire

Pour l'obtention du Diplome de Master en Informatique

Option : Réseaux et Ingénierie des Données (RID)

Présenté par :

M. Issa DEMBELE

M. Abdel Razak HAMADOU ADAMOU

L'OPTIMISATION DU DÉPLOIEMENT D'UN RCSF UTILISANT UN MODÈLE RADIO RÉALISTE, DÉDIÉ À LA SURVEILLANCE D'UN SITE SENSIBLE CLOS

Encadrant :

M. Ali BENZERBADJ

Maître de Conférences Classe "B" C.U.B.B.A.T.

Soutenu le 29/06/2020

Devant le jury composé de :

Président :	Zoheir BOUAFIA (M.A.A)	C.U.B.B.A.T
Examinateur :	Fethi Imad BENARIBI (M.A.A)	C.U.B.B.A.T
Encadrant :	Ali BENZERBADJ (M.C.B)	C.U.B.B.A.T.

Remerciements

Nous remercions Dieu le tout puissant qui nous a donné la force et la volonté pour réaliser ce projet.

Nous tenons à remercier en premier lieu M. Ali BENZERBADJ d'avoir accepté de diriger notre travail, pour son encadrement efficace, pour la confiance qu'il a bien voulu nous accorder, ses précieux conseils, son soutien et ses encouragements permanents, même dans les moments de doute.

Nous sommes très reconnaissant envers les membres du jury d'avoir accepté de juger notre modeste travail.

Nous tenons à remercier également M. Slimane Charafeddine BENGHELIMA Doctorant en première Année Informatique à l'université de Saâd DAHLAB de Blida.

Nos remerciements vont également à nos parents qui nous ont encouragés, qui nous ont appris à travailler honnêtement et qui nous ont toujours supporté moralement et financièrement pendant toutes nos longues années d'études et durant la réalisation de ce projet.

En fin, nous remercions vivement toutes les personnes qui nous ont soutenu et nous ont aidé, de près ou de loin, durant la préparation de ce projet.

Résumé

Dans ce mémoire, nous avons abordé le problème de déploiement des réseaux de capteurs sans fil (RCSFs) avec modèle de propagation radio réaliste, pour des applications de surveillance de sites sensibles clos. Pour optimiser ce déploiement, nous avons opté pour une optimisation multi-objectifs pour le placement déterministe de deux types de nœuds, à savoir les nœuds sentinelles et les nœuds relais. Cette topologie du réseau est dite topologie 2-tiers. Les nœuds sentinelles sont déployés à la frontière du site pour détecter les éventuelles intrusions, alors que les nœuds relais sont déployés à l'intérieur du site afin de relayer les alertes, générées par les nœuds sentinelles, jusqu'au nœud puits (sink).

Pour assurer une couverture maximale et redondante de la frontière, nous avons modélisé le site en grille à deux dimensions contenant des mailles qui ont été elles même discretisées en sous mailles (04 sous mailles). Cette discrétisation nous a permis d'avoir quatre (04) positions potentielles au niveau de chacune des mailles pour y placer les nœuds, ce qui nous a permis d'assurer une k-couverture au niveau de la frontière du site (k=3). Pour garantir une m-connectivité à moindre coût (m=2), en terme de nombre de sauts, depuis un nœud sentinelle jusqu'au sink, un premier BILP (Binary Integer Linear Program), ayant pour objectif la minimisation du

diamètre du réseau, est utlisé. Le résultat de ce BILP est utilisé comme contrainte par un second BILP ayant pour objectif la minimisation du nombre de relais, et ce afin de minimiser le coût du déploiement du RCSF. Ces minimisation ont un impact positif sur les performance du réseau, d'après les simulations que nous avons réalisées en utilisant le simulateur Contiki/Cooja.

Par ailleurs, nous avons procédé à l'amélioration du protocole RPL (Ipv6 Routing Protocol for Low Power and Lossy networks), en y ajoutant un mécanisme de prise en charge de l'équilibrage de charge. La nouvelle version du protocole RPL a été denommée (RPL with Load Balancing (RPL_LB).

L'évaluation des performances de notre réseau de surveillance a été faite en utilisant le simulateur Cooja/Contiki avec incorporation du protocole RPL amélioré (RPL_LB). Les résultats en termes de nombre de sauts, latence, energie et nombre de paquets reçus par le sink sont très encourageants.

Mot clés : Réseaux de Capteurs Sans Fil, Déploiement déterministe, Couverture, Connectivité, modèle de propagation radio réaliste, Optimisation combinatoire Multi-objectif, Programmation linéaire, Protocole multi-chemins, RPL.

Abstract

In this thesis, we adress the problem of deployment of wireless sensor networks (WSNs) with realistic radio propagation model, for fenced sensitive site surveillance applications. To optimize this deployment, we use a multi-objective optimization for the deterministic placement of two types of nodes, namely the sentinel nodes and the relay nodes. This network topology is called a 2-tier topology. Sentinel nodes are deployed at the site border to detect possible intrusions, while relay nodes are deployed inside the site in order to relay the alerts generated by the sentinel nodes, to the sink node.

To ensure maximum and redundant coverage of the border, we model the site in a two-dimensional grid containing meshes which have themselves been discretized in sub meshes (04 under meshes). This discretization allows us to have four (04) potential positions at the level of each mesh to place the nodes there, which allow us to ensure a k-coverage at the level of the border of the site (k = 3). To guarantee low-cost m-connectivity (m = 2), in terms of number of hops, from a sentinel node to the sink, a first BILP (Binary Integer Linear Program), with the aim of minimization of the network diameter, is used. The result of this BILP is used as a constraint by a second BILP aimed at minimizing the number of

relays, in order to minimize the cost of deploying the RCSF. These minimizations have a positive impact on network performance, according to the simulations we performed using the Contiki / Cooja simulator.

In addition, we have improved the RPL protocol (Ipv6 Routing Protocol for Low Power and Lossy networks), by adding a mechanism to support load balancing. The new version of the RPL protocol has been named (RPL with Load Balancing (RPL_LB).

The performance evaluation of our surveillance network was made using the Cooja / Contiki simulator with the incorporation of the improved RPL protocol (RPL_LB). The results in terms of number of hops, latency, energy and number of packets received by the sink are very encouraging.

Keywords : Wireless Sensor Networks, Deterministic deployment, Coverage, Connectivity, realistic radio propagation model, Multi-objective combinatorial optimization, Linear programming, Multi-path protocol, RPL.

Table des matières

Table des figures

Liste des tableaux

Abréviations

6LowPAN	IPv6 Low power Wireless Personal Area Networks
AODV	Ad hoc On Demand Distance Vector
AOMDV	Ad hoc On Demand Multiple Path Distance
AODVM	Ad hoc On Demand Distance Vector Multipath
API	Application Programming Interface
BBO	Biogeography Based Optimization
BILP	Binary Integer Linear Program
Contiki-NG	Contiki-Nex Generation
CPU	Central Processing Unit
Contiki OS	Contiki Operating System
COAP	Constrained Application Protocol
d	distance inter-nœuds
DAO	Destination Advertisement Object
DAO_ACK	DAO-Acknowledgment
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination-Oriented Directed Acyclic Graph
DSR	Dynamic Source Routing
DSDV	Dynamic Destination Sequenced Distance Vector
ELT	Expected Lifetime Transmission
ETT	Expected Transmission Time
ETX	Expected Transmission Count

GA	Genetic Algorithm
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPV6	Internet Protocol Version 6
LOADng	Lightweight On-demand Ad hoc Distance vector routing protocol
LLN	Low power and Lossy Networks
LPM	Low Power Mode
MAC	Media Access Control
MC	Muti-chemins
MP-OLSR	MultiPath Optimized Link State Routing Protocol
MDSDV	Multipath Dynamic Destination-Sequenced Distance- Vector
MPR	Multi Point Relay
MP-DSR	Multi-Path Dynamic Source Routing
MQTT	Message Queuing Telemetry Transport
MRHOF	Minimum Rank Hysterisis Objective Function
OF0	Objective Function 0
mw	Mili Watt
OLSR	Optimized Link State Routing Protocol
PDR	Packet Delivery Ratio
PMO	Problème Multi-Objectif
PLNE	Prgramme Linéaire en Nombre Entier
PO	Pareto optimal
QdS	Qualité de Service
RCSF	Reseau de Capteurs Sans Fil
Rs	Rayon de capture
Rc	Rayon de Communication
RREQ	Route REQuest

RREP	Route REPlay
RREP_ACK	Route REPlay Acknowledgment
RERR	Route ERRor
RPL	Ipv6 Routing Protocol for Low power and Lossy networks
RRCM	Route Confirmation Message
TC	Topology Control
TCP	Transmission Control Protocol

Introduction générale

Les Réseaux de Capteurs Sans Fil (RCSF) constituent une catégorie des Réseaux Ad hoc comportant un grand nombre de nœuds capable de recueillir et de transmettre des données de manière autonome.

C'est une technologie très courtisée de nos jours du fait des avantages qu'elle offre et aussi parce qu'elle est considérée comme une technologie verte d'avenir. Ainsi les RCSFs trouvent leurs applications dans plusieurs domaines de recherche.

Parmi ses domaines de recherche on retrouve la surveillance de sites sensibles clos (site pétrolier, frontières internationales, édifice gouvernementale, etc...) qui profite de l'autonomie, de l'aspect miniature et du faible coût des nœuds RCSFs. C'est dans cette optique que s'inscrit notre étude qui porte sur la surveillance de sites sensibles clos en utilisant les RCSFs.

Pour ce faire nous disposons de trois types de nœuds à savoir les nœuds sentinelles chargés de la surveillance qui génèrent une alerte en cas d'intrusion, les nœuds relais dont le rôle consiste uniquement à relayer les alertes générées par les sentinelles vers un point de contrôle et un nœud puit (sink) servant de centre de

contrôle.

Notre premier objectif est de déployer les nœuds capteurs tel que le diamètre de notre RCSF soit minimal. Ceci va garantir que les alertes générées par les nœuds sentinelles, lors de la detection d'une intrusion, soient tranmises au nœud puits à travers des routes constituées d'un nombre de sauts minimal. Ainsi, nous répondons aux exigences des applications critiques de surveillance, en matière de latence, fiabilité dans le routage des alertes et l'économie d'énergie.

Le déploiement optimal visé doit satisfaire certaines contraintes, notamment la k-couverture et la m-connectivité. Le but de la première contrainte est d'assurer une couverture multiple de chaque point de la zone à surveiller. Cette redondance dans la couverture est très importante pour une application critique de surveillance, dans la mesure où elle garantit la continuité de la détection des intrusions même quand un nœud sentinelle tombe en panne. Quant à la m-connectivité, elle vise à assurer que chaque nœud sentinelle dispose d'au moins m chemins vers le sink afin de pallier à toute rupture de lien en raison de tout type de défaillance d'un nœud capteur. C'est une exigence des applications critiques de surveillance.

D'autre part et dans un souci d'économie d'argent, on s'est imposé un deuxième objectif, à savoir un déploiement optimal à moindre coût. Pour cela nous avons décider de minimiser le nombre de relais déployés, sachant que le coût de ces derniers est plus important que les nœuds sentinelles.

La solution que nous proposons consiste tout d'abord en une discrétisation de notre zone à surveiller en mailles carrées constituées de quatre sous-mailles. Puis, nous avons formalisé notre problème d'optimisation du déploiement sous forme d'un problème d'optimisation multi-objectifs que nous avons modélisé en utilisant deux BILPs (Binary Integer Linear Program). La résolution de ces deux BILPs a été faite avec le solveur CPLEX, et en em-

ployant la méthode lexicographique.

La fonction objectif du premier BILP est la minimisation du diamètre du réseau alors que celle du deuxième BILP est la minimisation du nombre de nœuds relais déployés.

Aussi nous proposons dans ce mémoire une amélioration du protocole RPL (Ipv6 Routing Protocol for Low power and Lossy networks) que nous avons utilisé lors de nos simulations sous Contiki/Cooja. Cette amélioration consiste à rendre le routage multi-chemins actif et pemettre l'équilibrage de charge.

Ce document est structuré en trois chapitres, à savoir :

— Le premier Chapitre présente un état de l'art sur le déploiement dans les es Réseaux de Capteurs Sans Fil (RCSFs). Il détaille les méthodes de discrétisation de la zone d'intérêt puis décrit les concepts de couverture et de connectivité dans ce type de réseaux. En fin, il décrit les méthodes de resolution des problèmes d'optimisation combinatoire multi-objectifs

— Le deuxième Chapitre fait un état de l'art sur les protocoles de routage multi-chemins.

— Le troisième chapitre est consacré à l'implémentation, la simulation et à l'evaluation des performances de notre RCSF. On n'y détaille les deux BILPs proposés. Ensuite nous présentons et commentons les résultats de la résolution des deux BILPs. En fin, nous détaillons les résultats de la simulation sous Contiki/Cooja relatives à la performance de notre RCSF, et ce en termes de nombre de sauts, latence, PDR (Paquet Delivery Ratio) et consommation d'énergie. Les résultats sont jugés encourageants.

CHAPITRE 1

Optimisation du déploiement des RCSFs : Etat de l'art

1.1 Introduction

L'optimisation dans les RCSFs a été abordée dans de nombreux travaux traitant de différentes problématiques. L'une de ces problématiques est le déploiement des nœuds capteurs dans une zone d'intêret, ce qui constitue la prémière étape de la mise en place d'un RCSF. L'optimisation du déploiement a un impact direct sur la couverture, la connectivité et la consommation d'energie.

Dans ce Chapitre, nous allons passer en revue les méthodes de déploiements dans les RCSFs et l'optimisation sous les contraintes de la couverture et de la connectivité, notamment.

1.2 Méthodes de déploiement

Le déploiement des nœuds capteurs peut être soit déterministe, si nous connaissons la zone à surveiller, soit aléatoire dans le cas où nous avons peu de connaissance sur le zone, ou si celle-ci est vaste ou inaccessible (voir Figure 1.1).

De ce fait, nous pouvons affirmer que le choix de la méthode

de déploiement est influencé par le type d'application et de l'environnement d'étude.

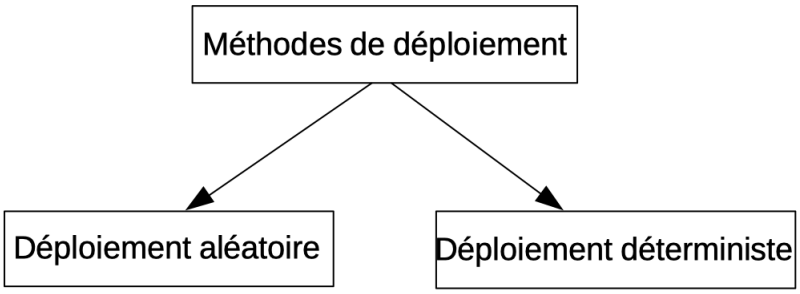

FIGURE 1.1 – Méthodes de déploiement

1.2.1 Déploiement aléatoire

Dans ce type de déploiement, la position des capteurs n'est pas connue à l'avance. Le déploiement des nœuds se fait généralement par avion, hélicoptère ou drône, à cause de l'hostilité ou carrément l'inaccessibilité de la zone d'intérêt.

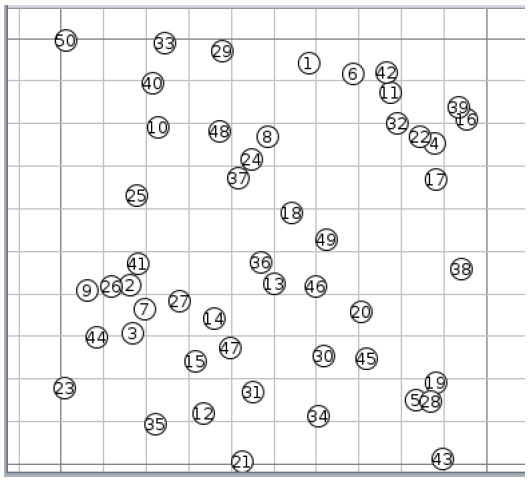

FIGURE 1.2 – Exemple de déploiement aléatoire

1.2.2 Déploiement déterministe

Dans le déploiement déterministe, appelé aussi planifié, les capteurs sont placés un par un, soit par un être humain soit par un robot dans des endroits déterminés au préalable. Il est utilisé dans les zones accessibles.

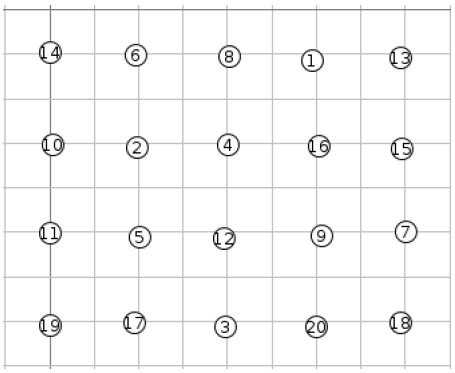

FIGURE 1.3 – Exemple de déploiement déterministe

1.3 Méthodes de modélisation des zones de déploiement

La phase de déploiement est une étape fondametale dans la mise en place d'un réseau de capteurs sans fil pour résoudre la problématique du déploiement. Pour cela, trois stratégies sont utilisées [2] :

— stratégie basée sur la force
— stratégie basée sur une grille
— stratégie basée sur la géométrie de calcul

1.3.1 Méthode basée sur la force

La stratégie de déploiement basée sur la force [18, 2] repose sur la mobilité des capteurs, en utilisant des forces répulsives et

attractives virtuelles, les capteurs sont obligés de s'éloigner ou de se rapprocher les uns des autres pour obtenir une couverture complète. Les capteurs continuent de se déplacer jusqu'à ce que l'état d'équilibre soit atteint ; là où les forces répulsives et attractives sont égales, elles finissent par s'annuler.

1.3.2 Méthode basée sur une grille

Il existe trois types de grilles couramment utilisées dans les réseaux de capteurs [40] :

— **la grille triangulaire** : Elle est la meilleure parmi les trois types de grilles car elle a la plus petite zone de chevauchement, donc elle nécessite le moins de capteurs à déployer. Comme le montre la Figure 1.4, on peut avoir jusqu'à six (06) connectivités. La couverture totale est atteinte si la distance inter-nœuds est inferieure ou égale à $\sqrt{3}R_s$ et $R_c \geq d$ (R_s est le rayon de capture, R_c est le rayon de communication et d est la distance inter-nœuds).

Le nombre minimal de nœuds dans ce réseau est donné par :

$$2\sqrt{3}/9 \text{ x } 1/R_s^2 \text{ [37].}$$

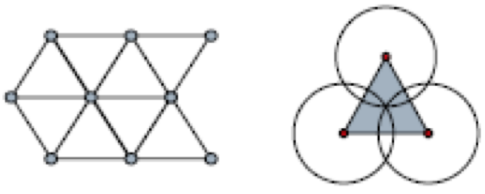

FIGURE 1.4 – Exemple de grille en triangle [2]

— **la grille carrée** : La zone est divisée en cellules carrées, et les nœuds peuvent être placés soit sur les sommets de la cellule, soit au centre de la cellule. Nous tenons à rappeler

que c'est la stratégie que nous avons utilisé dans la solution proposée dans ce mémoire.
Elle offre une couverture totale si la distance entre deux nœuds est inférieure ou égale à $4\sqrt{2}R_s$ et $R_c \geq d$ [37].

Le nombre minimal de nœuds capteurs est donné par :

$$1/2 \text{ x } 1/R_s^2 \text{ [37].}$$

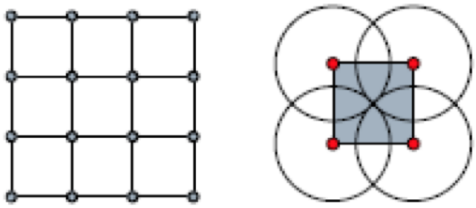

FIGURE 1.5 – Exemple de grille en carré [2]

— **la grille hexagonale** : Elle est considéré comme le modèle le plus couteux par rapport à la grille triangulaire et la grille carrée car elle a la plus grande zone de chevauchement.

Le nombre minimal de nœuds est obtenu par :

$$4\sqrt{3}/9 \text{ x } 1/R_s^2 \text{ [37].}$$

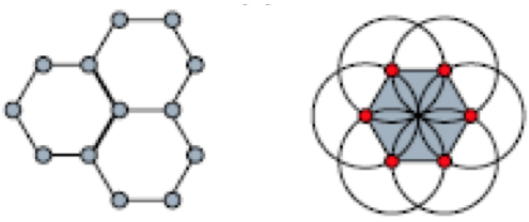

FIGURE 1.6 – Exemple de grille en hexagonale [2]

En plus du type de la grille, la taille de celle-ci joue également un rôle important. Elle doit être choisie en fonction de la densité du réseau. Pour un réseau très dense, les grilles de petite taille

aident à réduire les trous de couverture, offrant ainsi un meilleur résultat.

Par contre dans un réseau clairsemé, une grande taille de la grille est meilleure car elle évite le chevauchement de la plage de détection des capteurs, garantissant ainsi la pleine utilisation de leurs capacités de détection [5].

1.3.3 Méthode basée sur la géométrie de calcul

Dans cette stratégie, les deux approches les plus utilisées sont :

— **le diagramme de Voronoï** : c'est une repartition de la zone de telle manière que chaque nœud n'occupe qu'un polygone et est plus proche de tout point de ce polygone que de tout autre nœud des polygones voisins [13].

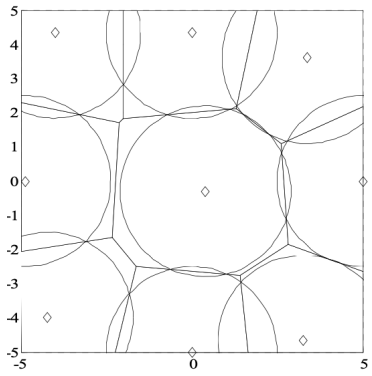

FIGURE 1.7 – Diagramme de Voronoi [2]

— **la Triangulation de Delaunay** : La triangulation de Delaunay est le dual du diagramme de Voronoï. Un triangle de Delaunay est formé de trois sites fournis si et seulement si le cercle du site ne contient pas d'autres sites [13].

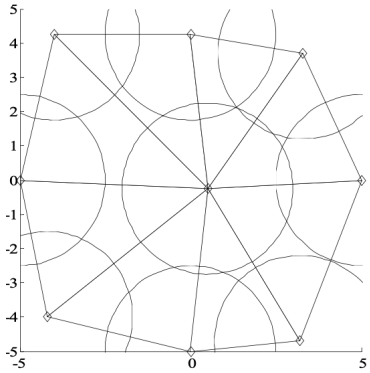

FIGURE 1.8 – Triangulation de Delaunay [2]

1.4 Couverture et connectivité dans les RCSFs

1.4.1 Couverture

La couverture est considérée comme étant une mesure de qualité de service (QdS) très importante dans les RCSFs. Elle reflète la façon dont une zone donnée est couverte par un nœud capteur.

La couverture d'une zone donnée varie en fonction des besoins de l'application.

1.4.1.1 Rayons de couverture et de communication d'un nœud capteur

Un nœud capteur possède deux rayons : un rayon de capture ou Sensing Range (R_s) et un rayon de communication ou Communication Range (R_C) [12], voir la Figure 1.9.

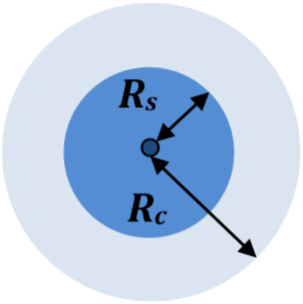

FIGURE 1.9 – Rayons de couverture et de communication d'un nœud capteur

1.4.1.2 Les types de couvertures

Dans les RCSFs on peut avoir trois (3) types de couverture [12] :

— **La couverture de zone** :
Dans ce type de couverture, chauqe point de la zone d'intérêt est couvert par un sous-ensemble de nœuds capteurs.

— **La couverture de barrière** :

Il s'agit de la couverture d'une partie de la zone d'intérêt. Les nœuds capteurs dédiés à ce types de couverture ne peuvent pas couvrir les évènement à l'intérieur de la zone. Ils couvrent généralement la bordure (ou la frontière) d'un site.

— **La couverture de cibles** :

Ce type de couverture sert à couvrir uniquement certains points spécifiques (cibles) dans un champ de capture, dont la position géographique est connue et chaque point spécifique doit être couvert par au moins un nœud capteur.

1.4.2 Connectivité

Dans les RCSFs, la connectivité est la communication entre deux nœuds de façon directe (connectivité à un saut), ou indirecte (connectivité à sauts multiple).

De ce fait, nous disons qu'un RCSF est connecté s'il y'a au moins une route entre la station de base (nœud puits) et chaque nœud du réseau pour pouvoir assurer l'achéminement des informations entre eux.

1.5 k-couverture et m-connectivité

Étant donné un ensemble de points cibles et un ensemble de positions potentielles préfixées, nous devons sélectionner un nombre minimal (k) de positions potentielles pour placer les nœuds capteurs afin qu'il couvre les zones cibles et assure une (m) connectivité des nœuds capteurs.
Nous disons qu'une cible est couverte si elle se trouve dans le rayon de sensation (Rs) d'un nœud capteur et que le nœud capteur peut communiquer avec la station de base directement ou via d'autres nœuds capteurs [23].

1.5.1 1-couverture

Une couverture est dite simple lorsque tout point p de la zone cible est couvert par un et un seul capteur.

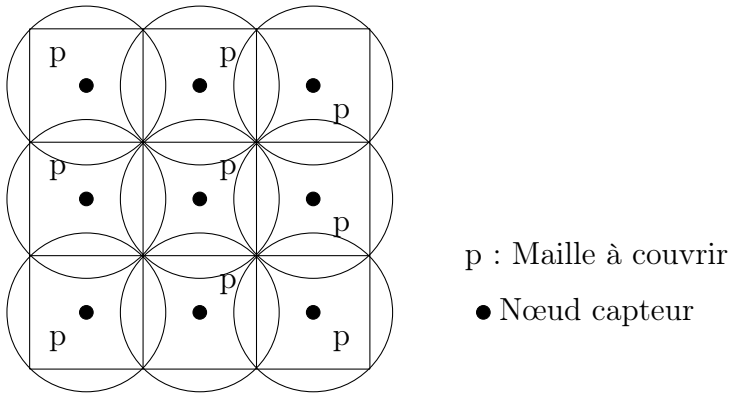

FIGURE 1.10 – 1-couverture

1.5.2 k-couverture

On parle de la k-couverture (k $>$ 1) lorsque tout point p de la zone cible est couvert par au moins k capteurs.

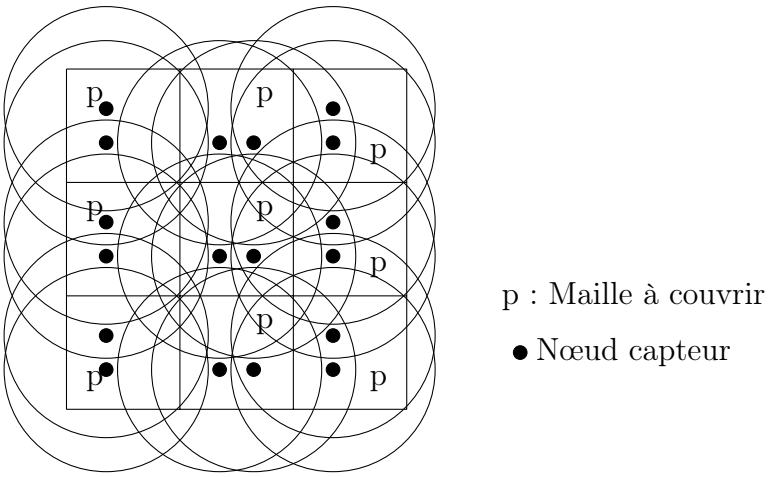

FIGURE 1.11 – k-couverture (k=2)

Il existe trois types de connectivité dans les RCSFs qui sont la connectivité complète simple (1-connectivité), la connectivité complète multiple (m-connectivité, $m > 1$), la connectivité intermittente (voir Figure 1.12).

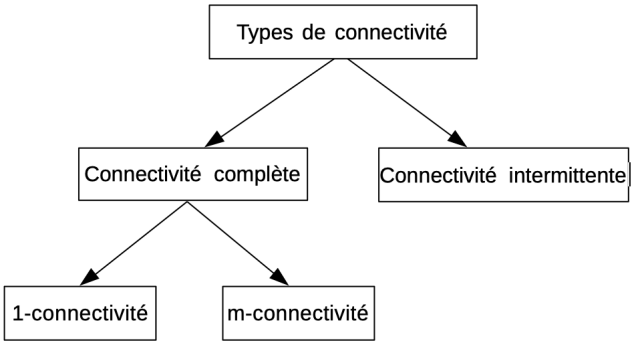

FIGURE 1.12 – Les types de connectivité

1.5.3 La connectivié complète

:

— Une connectivité complète est dite simple s'il existe un seul chemin entre chaque nœud et la station de base (1-connectivité).
— Une connectivité complète est dite multiple s'il existe plusieurs chemins distincts entre chaque nœud et la station de base (k-connectivité).

1.5.4 La connectivié intermittente

Une connectivité est dite intermittente si la communication se fait en intermittence c'est à dire de temps en temps.

1.6 Objectifs et contraintes du déploiement

Le processus de déploiement des nœuds capteurs a un impact sur la performance du réseau. Il permet de définir la topologie du réseau, donc définir le nombre et la position des nœuds capteurs. Le nombre de sauts et nombre de paquets reçu (PDR) du réseau, la latence, et la durée de vie sont aussi directement affectées par la topologie de réseau. La topologie à son tour est affectées par la

couverture et la connectivité.

- **Le nombre de sauts** : il est possible d'estimer la performance d'un reseau en fontion du nombre de sauts constituant les chemins entre les paires de nœuds. .
- **Le nombre de paquets reçu (PDR)** : le PDR (Packet Delivery Ratio) est le rapport du nombre de paquets reçu sur le nombre de paquets envoyé dans un réseau.
- **La latence** : la latence est le délai de transmission de données dans un réseau.
- **La durée de vie** : la durée de vie est définie comme étant l'intervalle de temps entre le déploiement et l'instant où le réseau est considéré comme non fonctionnel. Cela peut être, par exemple, l'instant où le premier capteur meurt ou un pourcentage de capteurs meurent.

Lors du déploiement des RCSFs la couverture et la connectivité peuvent, en fonction du contexte, être considérées soient comme contraintes soient comme des objectifs.

Les objectifs du déploiement peuvent ainsi être la maximisation de la couverture et la connectivité comme ils peuvent être la minimisation du diamètre du réseau ou le nombre de relais, avec la couverture et la connectivité comme étant des contraintes. C'est d'ailleurs la cas des BILPs (Binary Integer Linear Program) que nous proposons pour résoudre le problème de déploiement de notre RCSF de surveillance.

1.7 Problèmes d'optimisation combinatoire

Les problèmes d'optimisations combinatoire peuvent être classés en deux grandes catégories [14], à savoir les problèmes mono-objectis et multi-objectifs.

Les problèmes d'optimisation combinatoires rencontrés en pratique sont rarement mono-objectifs, ce qui nous pousse à nous

consacrer uniquement au PMO dans la suite de ce memoire.

1.7.1 Problèmes d'optimisation mono-objectifs

ILs ont pour but d'optimiser un seul objectif pour trouver la solution optimale qui est déterminée suivant un seul critère.

1.7.2 Problèmes d'optimisations multi-objectifs (PMO)

Ils visent à optimiser simultanement plusieurs objectifs. La solution d'un PMO (Problème Multi-Objectifs) n'est pas unique, mais un ensemble de solutions appelé Pareto optimales, noté PO [14].

1.7.3 Etapes de résolution d'un problème d'optimisation multi-objectifs

Comme nous l'avons souligné plus haut, le but d'un PMO est d'optimiser (minimiser ou maximiser) plusieurs objectifs. En effet, il n'existe pas qu'une seule mais plusieurs solution pour un PMO. Le premier objectif dans la résolution d'un PMO est d'obtenir l'ensemble des solutions appelé PO (Pareto optimales). Ensuite, en deuxième lieu il faut choisir une des solutions de l'ensemble PO suivant des préférences et des critères liés à la nature du problème [14].

La première étape de la résolution est laissée au compte d'un solveur et la deuxième est généralement délégué à un décideur expert du domaine. Vous conviendrez avec nous alors que la relation solveur décideur est un des principes fondamentaux de la résolution de PMO. Cette relation peut prendre trois formes :

— A priori : Dans cette approche, le problème est transformé en un problème mono-objectif. Ce Procédé consiste à combiner les différentes fonctions objectifs en une seule fonction. Dans ce cas, le compromis que l'on veut faire entre les objectifs a été défini avant l'exécution de la méthode ; ***Preferences***

-> Research.

— A posteriori : Dans cette approche, on cherche à générer un ensemble de bonnes solutions au décideur en lançant l'algorithme d'optimisation en premier, puis il choisit parmi celles-ci la solution qui lui convient le plus ; ***Research -> Preferences***.

— Interactive : Cette approche prend en compte les préférences formulées à partir des connaissances a priori du décideur et celles acquises pendant la résolution du problème, c'est-à-dire une coopération progressive entre le solveur et le décideur est appliquée, comme le montre la Figure 1.13.

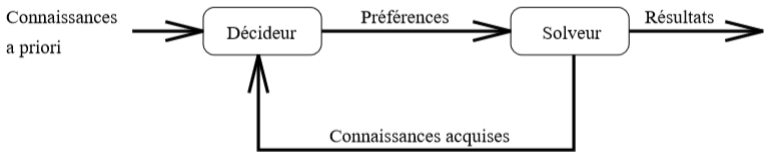

FIGURE 1.13 – Cooperation entre décideur et solveur

Le choix d'une méthode de résolution pour un problème multi-objectifs dépend de la complexité du problème ainsi que de sa taille. Un simple problème d'optimisation de petite taille peut être résolu en utilisant une méthode exacte dont le temps d'exécution est raisonnable. Mais, pour les problèmes difficiles, complexes ou de grande taille les méthodes exactes ne sont plus valables, et les méta-heuristiques deviennent les plus appropriées. Malgré que la solution n'est pas l'optimum mais une solution approchée, le temps d'exécution mis par les méta-heuristiques est raisonnable et favorise son choix pour les problèmes difficiles et complexes [4].

1.8 Méthodes de résolution des PMO

Les méthodes de resolution peuvent être classées en deux grandes catégories. La Figure 1.14 montre la classification des différentes methodes de résolution d'un PMO.

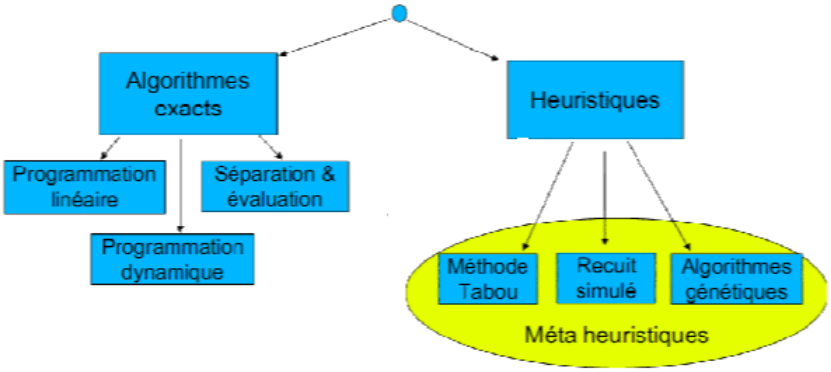

FIGURE 1.14 – Les méthodes métaheuristiques [16]

1.8.1 Méthodes exactes

Ce sont des méthodes qui fournissent des solutions optimales mais pour des problèmes de petites tailles, par contre pour les problèmes de grandes tailles le temps de calcul devient irraisonnable. Parmi ces méthodes, on trouve la méthode par séparation et évaluation (Branch and Bound) qui est la plus utilisée.

1.8.1.1 Branch & Bound

Branch and Bound est un algorithme de résolution de problème d'optimisation combinatoire. Il consiste à diviser un problème initial F en un ensemble de sous-problèmes F_i pour les résoudre (voir Figure 1.15) .

A chaque instant, l'algorithme maintient une liste de sous-problèmes actifs, un coût U de la meilleure solution obtenue jusque-là et la valeur initiale de U.

Les différentes étapes de l'algorithme sont :

1. sélectionner le coût initial U (depuis une solution initiale praticable),
2. diviser F en plusieurs sous-problèmes,
3. sélectionner un sous-problème actif F_i,
4. si F_i est non admissible, le supprimer, sinon, calculer b(F_i),
5. si b(F_i) est supérieur à U, supprimer Fi,
6. sinon soit résoudre F_i directement soit recommencer depuis (2) [26].

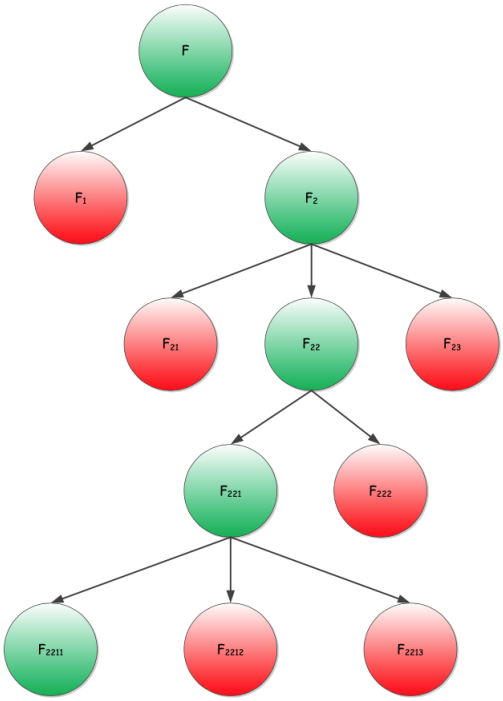

FIGURE 1.15 – Division du problème F en sous-problèmes [26]

1.8.1.2 Branch & Cut

L'algorithme de Branch & Cut est un Branch & Bound classique auquel on rajoute une génération dynamique de contraintes, appelées coupes. La méthode de coupe est appliquée localement sur chaque nœud et de manière répétitive jusqu'à l'obtention d'une solution entière optimale. Cette méthode s'est avérée très efficace pour la résolution de problèmes d'optimisation combinatoire difficiles.
Malgré les progrès réalisés dans le domaine des méthodes d'optimisation exactes, elles restent des méthodes appliquées à des problèmes de petites tailles (leurs temps de calcul risque d'augmenter exponentiellement avec la taille du problème) [1].

1.8.2 Méta-heuristiques

Les méta-heuristiques sont utilisées pour résoudre des problèmes d'optimisation combinatoire de grande taille. Ce sont des stratégies permettant de guider la recherche afin d'obtenir une solution proche de la solution optimale. Elles sont non déterministes puisque l'arrivée à la solution optimale n'est pas garantie [24, 19].

1.8.2.1 Recherche Tabou

Elle a été proposée par Fred Glover en 1986 [15], son idée principale consiste à garder en mémoire l'historique des solutions déjà explorées. L'algorithme commence par créer et initialiser une liste vide L de taille fixe. Cette liste contiendra toutes les solutions explorées et qui deviennent interdites ou "Tabou". Ceci permet à l'algorithme de ne pas tomber dans des cycles répétitifs. Le choix de la solution va s'effectuer de manière à ce que l'on choisisse la meilleure solution dans le voisinage, en évitant de choisir ceux contenus dans la liste Tabou.

1.8.2.2 Recuit Simulé

Cette méthode tire son nom de la physique des matériaux et plus précisément de la métallurgie. Le "Recuit" est une opération qui consiste à laisser refroidir lentement un metal pour l'amener à un état de quasi équilibre thermodynamique. Le principe de l'algorithme consiste à imiter (simuler) la méthode du recuit pour pouvoir trouver une solution quasi-optimale [20].

1.8.2.3 Les algorithmes génétique (GA)

Les algorithmes génétiques sont la combinaison de deux domaines qui sont la biologie et l'informatique. Ils utilisent des termes de la génétique humaine comme métaheuristiques pour résoudre un problème d'optimisation combinatoire, ces termes sont [39] :

- **Genèse** : c'est la première phase de l'algorithme, il s'agit d'une population initiale de taille N.
- **Chromosome** : c'est une chaîne représentant les caractéristiques de l'individu.
- **Phénotype** : c'est un ensemble de paramètres ou une structure décodée.
- **Evaluation** : c'est la phase de calcul de la fonction de fitness (fonction objectif).
- **Sélection** : c'est le choix des individus qui vont se reproduire.
- **Croisement** : c'est la phase de production des descendants.
- **Mutation** : c'est la modification d'un chromosome dans le but d'améliorer les caractéristiques de l'individu.

L'algorithme se compose de cinq étapes :

1. Initialisation

2. Sélection pour la reproduction Croisement des individus sélectionnés
3. Mutation des individus sélectionnés
4. Sélection pour le remplacement
5. Si la condition d'arrêt est vérifiée STOP, S = les meilleurs individus
6. Sinon retour à l'étape (2).

1.8.2.4 L'algorithme d'optimisation basé sur la biogéographie (BBO)

L'optimisation basée sur la biogéographie (BBO) est une technique d'optimisation globale basée sur la population qui a été conçue par Simon en 2008 [3]. Le problème est modélisé en utilisant le concept d'immigration naturelle et d'émigration des espèces.

Il s'inspire de la théorie de la biogéographie insulaire. Ces deux principaux opérateurs de fonctionnements sont la migration et la mutation.

L'algorithme commence par générer un nombre fini d'individus choisi généralement par tirage aléatoire uniforme dans l'espace de recherche formant la population initiale.

Après évaluation de la population initiale, certains individus sont choisis pour participer à l'opération de migration qui permet de créer un nouvel ensemble d'individus. Les descendants vont être à leur tour mutés. Le taux de mutation fixe la proportion de la population qui sera renouvelée à chaque génération. Enfin, une phase de remplacement consiste à remplacer les parents par les nouveaux descendants, afin de former une nouvelle population, de la même taille qu'au début de l'itération. L'algorithme se termine après un certain nombre de générations [3].

1.9 Conclusion

Dans ce chapitre, nous avons réalisé un état l'art sur le déploiement dans les réseaux de capteurs sans fil. Nous avons d'abord présenté les différentes méthodes de médélisation de la zone d'intérêt, puis nous avons décrit les concepts de couverture et de connectivité dans les RCSFs. Ensuite, nous avons présenté les méthodes de resolution des problèmes d'optimisation combinatoire multi-objectifs.

Le prochain Chapitre sera consacré à un état de l'art sur les protocoles de routage multi-chemins.

CHAPITRE 2

Protocoles de routage multi-chemins dans les RCSFs : Etat de l'art

2.1 Introduction

Le routage mono-chemin dans les RCSFs s'est avéré défaillant sur certains points d'où l'avènement du routage multi-chemins qui permet d'envoyer les données sur un ensemble de chemins menant d'une source à une destination. Le routage multi-chemins semble être une solution efficace dans les RCSFs qui sont des réseaux avec des liens instables et aussi des réseaux à forte mobilité et/ou à forte charge. Cette solution permet de se prémunir contre le problème de rupture de route et de congestion, ce qui permet d'améliorer les performances des communications et d'agréger les ressources disponibles dans le réseau.

Dans ce Chapitre, nous présentons un état de l'art sur les principaux protocoles de routage multi-chemins dans les RCSFs.

2.2 Métrique de routage

Une métrique est un algorithme qui traite le coût en termes de distance (nombre de sauts), de qualité de lien, ou d'énergie résiduelle associée à une route, afin de mesurer l'efficacité des protocoles de routage. Ainsi grâce au protocole, chaque nœud compare les métriques calculées pour déterminer les routes optimales à emprunter [9].

2.2.1 Métrique basée sur le nombre de sauts

Elle est utilisée dans les protocoles pour minimiser le nombre de sauts d'une route. Cette minimisation se fait en calculant le nombre de nœuds qui seront traversés (nœuds intermédiaires) lors de l'acheminement d'un paquet de données d'un nœud source vers un nœud destination.

2.2.2 Métrique basée sur la qualité de lien

Le calcul de la qualité de lien peut se faire de plusieurs manières différentes, à savoir :

- **Expected Transmission Count (ETX)** : C'est le nombre de transmissions et retransmissions nécessaire pour assurer une bonne livraison des données [10].
- **Expected LifeTime (ELT)** : Estime la durée de vie attendue, c'est-à-dire le temps avant la mort du premier nœud du reseau [8].
- **Expected Transmission Time (ETT)** : C'est le temps nécessaire pour une bonne reussite de la transmission d'un paquet de données [32].

2.2.3 Métrique basée sur l'énergie résiduelle

Elle est utilisée dans les protocoles de routages pour évaluer l'énergie restante dans chaque nœuds du réseau. Cette évaluation se fait en soustrayant l'energie maximale du nœud avec son énergie en un instant donné.

2.3 Classification des protocoles de routage

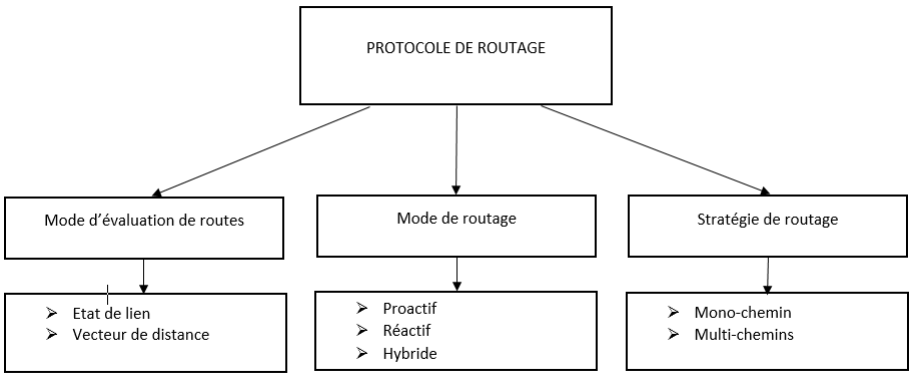

FIGURE 2.1 – Classification des protocoles de routage

Les protocoles de routage peuvent être classés selon trois principaux critères [28], à savoir :

— Mode d'établissement des routes
— Mode de routage
— Stratégie de routage.

2.3.1 Classification selon le mode d'établissement des routes

Dans les réseaux de capteurs sans fil, il existe deux grandes familles de protocoles :

— Les protocoles basés sur l'état de lien
— Les protocoles basés sur le vecteur de distance

Les deux méthodes exigent une mise à jour périodique des données de routage qui doivent être diffusées par les différents nœuds du réseau. Les algorithmes de routage basés sur ces deux méthodes, utilisent la même technique qui est la technique des plus courts chemins, et permettent à un hôte donné, de trouver le prochain hôte pour atteindre la destination en utilisant le trajet le plus court existant dans le réseau [28].

2.3.1.1 Protocoles basés sur l'état de lien

Dans les protocoles basés sur l'état du lien, chaque nœud du réseau maintient une vue globale de la topologie du réseau qui lui permet de calculer les chemins vers toutes les destinations. Les nœuds diffusent périodiquement (par inondation) l'état des liens avec leurs voisins. La diffusion peut aussi se produire lorsqu'un des nœuds détecte un lien corrompu vers un de ses voisins ou lorsqu'une mise à jour de la topologie a lieu.

Le problème de cette approche est que les messages de contrôle surchargent le réseau et surtout quand il s'agit d'un réseau avec un grand nombre de nœuds [28].

2.3.1.2 Protocoles basés sur le vecteur de distance

Dans les protocoles basés sur le vecteur de distance, chaque nœud transmet à ses voisins la distance en termes de nombre de sauts qui le sépare de chaque destination dans le réseau et le nœud voisin utilisé pour atteindre cette destination. Donc, un nœud établit les routes vers les autres destinations en se basant sur les informations reçues de tous ses voisins. Ce dernier calcule le chemin le plus court vers n'importe quelle destination dans le réseau. Si la distance séparant deux nœuds change, on répète le processus de calcul.

L'approche basée sur le vecteur de distance évite l'inondation, mais elle est moins précise que l'approche basée sur l'état de lien puisqu'il est aussi difficile de trouver des routes alternatives en cas de rupture de routes [28].

2.3.2 Classification basée sur le mode de routage

2.3.2.1 Protocoles proactifs

Ils établissent les routes à l'avance et sont particulièrement utilisés dans les réseaux dont la topologie est plutôt stable (réseau peu mobile). Ils maintiennent en permanence une vision globale de l'état du réseau Ad Hoc grâce à une gestion périodique des tables de routage et l'échange de trames périodiques de contrôle.

L'avantage d'un protocole de routage proactif est le gain de temps lorsqu'une route est demandée. Le problème c'est que dans le réseau Ad Hoc, les changements de routes peuvent être plus fréquents que la demande de route. D'ailleurs, la plupart des informations ne sont jamais utilisées, ce qui gaspille les ressources du réseau sans fil. Les protocoles proactifs tiennent continuellement à jour une vue de la topologie afin d'avoir à tout instant une route disponible pour toute paire de nœuds [27].

2.3.2.2 Protocoles réactifs

Un protocole réactif détermine la route uniquement sur demande et est particulièrement adaptée aux réseaux plus mobiles. L'avantage du protocole réactif est de ne pas gaspiller les ressources du réseau. Par contre, ses retards dépassent bien souvent les délais moyens admis par les applications, aboutissant à une impossibilité de se connecter alors que le destinataire est bien là [11].

2.3.2.3 Protocoles hybrides

Le routage hybride combine les aspects des routages proactif et réactif [27].

2.3.3 Classification basée sur la stratégie de routage

Dans les protocoles de routage, les données peuvent être router de deux manières différentes qui sont :

- **Routage mono-chemin** : Le routage momo-chemin permet d'acheminer les données via un chemin unique, entre un nœud source et un nœud destination.
- **Routage multi-chemins** : Le routage multi-chemins permet d'acheminer les données via un ensemble de chemins menant d'un nœud source à un nœud destination.

2.4 Protocoles de routage multi-chemins

Le routage multi-chemins a pour but d'améliorer les performance du réseau et de combler les lacunes du routage mono-chemin.

2.4.1 Objectifs du routage multi-chemins

Le routage multi-chemins a la particularité de considérer plusieurs routes au lieu d'une seule, entre une source et une destination. Cette particularité lui permet de répondre à trois objectifs [34] :

- **Tolérance aux pannes** : Elle peut se faire principalement de deux manières différentes, soit par l'envoi de paquets redondants sur plusieurs routes ou par l'établissement des routes de secours pour prévenir les pannes.
- **Equilibrage de charge** : Pour éviter la congestion dans le réseau, il est plus que nécessaire de répartir les données sur

plusieurs routes différentes au lieu d'une seule pour alléger le traffic sur certaines routes du réseau.

— **L'augmentation de la bande passante** : Il est possible d'agréger la bande passante avec les différentes routes utilisées en parallèles, qui sera supérieure à celle d'une seule route.

2.4.2 Mécanismes de fonctionnement du routage multi-chemins

Il se base sur trois (3) mécanismes [11] :

— **La découverte des chemins** : C'est le processus de recherche des routes.

— **La maintenance des chemins** : Consite à répérer et réparer les routes en panne ou réchercher des routes de secours.

— **L'allocation du trafic** : C'est la répartition des données à transmettre sur les routes disponibles.

2.4.3 Types de chemins dans le routage multi-chemins

Dans un réseau de capteur sans fil multi-sauts, la communication directe (un seul saut) entre deux nœuds est assurée sur un lien. En revanche, la communication multi-saut entre deux nœuds distants est assurée sur un chemin.

Il existe principalement trois types de chemins, à savoir les chemins à nœuds disjoints, les chemins à liens disjoints, les chemins non disjoints [6].

2.4.3.1 Chemins à nœuds disjoints

Il n'y a que la source et la destination comme nœuds en commun entre les différents chemins voir Figure 2.2. Ils assurent une bonne fiabilité par rapport aux chemins non disjoints et à liens disjoints. En effet, avec des chemins à nœuds disjoints, quand un

nœud tombe en panne ou un lien est rompu, c'est un seul chemin qui sera défaillant, ce qui n'est pas le cas pour les deux autres types [29].

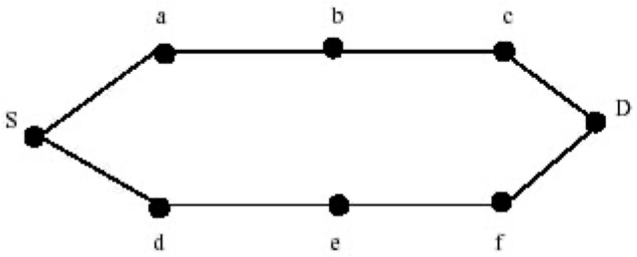

FIGURE 2.2 – Chemins à nœuds disjoint [29]

Comme nous pouvons le constater, SabcD et SdefD sont deux chemins a nœuds disjoints.

2.4.3.2 Chemins à liens disjoints

Ce sont des chemins qui n'ont aucun lien en commun mais peuvent avoir des nœuds en commun [29], comme le montre la Figure 2.3.

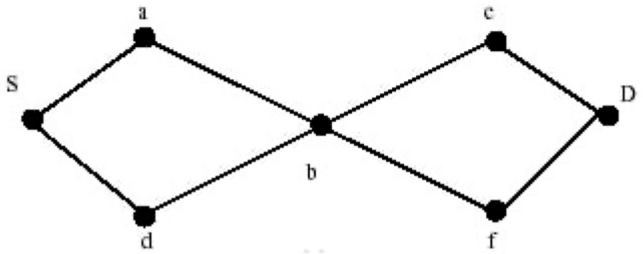

FIGURE 2.3 – Chemins à liens disjoints [29]

2.4.3.3 Chemins non disjoints

Ce sont des chemins qui peuvent avoir des nœuds ou des liens en commun [29]. Leur principal avantage est qu'ils sont faciles à découvrir parce qu'ils n'ont aucune contrainte sur la recherche des chemins, ils peuvent être à nœuds disjoints ou à liens disjoints, comme le montre la Figure 2.4.

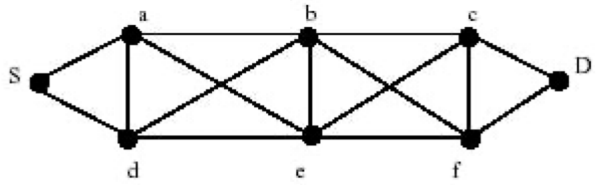

FIGURE 2.4 – Chemins non disjoints [29]

2.4.4 Exemples de protocoles de routage multi-chemins

Dans les réseaux de capteurs sans fil, il existe une large gamme de protocoles de routage qui sont de base mono-chemin comme AODV (Ad Hoc On demand Distance Vector), LOAD-ng (Light-weight On-demand Ad hoc Distance vector, Next Generation), et tant d'autres. Dans ce qui suit, nous allons décrire certains d'entre eux.

2.4.5 Protocole AODV-Multipath (AODVM)

L'AODV-Multipath (AODVM) est un protocole réactif. C'est une extension du protocole AODV dans laquelle les chemins sont à nœuds ou lien disjoints [41]. L'établissement des routes se fait par le biais des messages de contrôle suivants :

— **RREQ (Route REQuest)** : Le message RREQ est toujours utilisé pour demander la création d'une route vers une

destination lorsqu'un nœud doit envoyer un message de données et que le chemin vers la destination est inconnu .

— **RREP (Route REPlay)** : Le message RREP est utilisé par la destination qui reçoit le RREQ comme réponse à la demande de création d'itinéraire .

— **RREP_ACK** : Le message RREP_ACK (accusé de réception de réponse d'itinéraire) est utilisé pour répondre à un RREP reçu .

— **RRCM (Route Confirmation Message)** : Le message RRCM est envoyé par la source et utilisé pour confirmer une route .

— **RERR (Route ERRor)** : Le messages RERR est utilisé pour informer l'expéditeur du message de données qu'un problème est survenu. Il peut également être utilisé lorsque la destination du message de données est inconnue par le nœud intermédiaire.

Nous notons qu'il existe d'autres variantes du protocole AODV, à savoir AOMDV (AdHoc On Demand Multiple Path Distance Vector) [30] et MP-AODV (Multipath Routing Protocol Based on AODV) [30].

2.4.6 Protocole Multipath Dynamic Destination-Sequenced Distance-Vector (MDSDV)

Le MDSDV [31] est un protocole proactif et une extension multi-chemins du protocole Dynamic Destination Sequenced Distance Vector (DSDV) [35]. Il construit des chemins à nœuds disjoints. Deux nouveaux champs, dénommés "second hop" et "link-id", sont utilisés pour obtenir ces chemins disjoints. Le champ "link-id" est généré par un nouveau nœud arrivant dans le réseau pour distinguer les différents liens vers chacun des voisins à un saut.

2.4.7 Protocole MultiPath Optimized Link State Routing (MP-OLSR)

Le protocole MP-OLSR est un protocole de routage multi-chemins hybride basé sur OLSR (Optimized Link State Routing) [22].

Le protocole MP-OLSR utilise deux types de message de contrôle, à savoir :

- **HELLO** : Les messages HELLO sont utilisés pour la détection des voisins et le calcul des relais multipoints (Multi Point Relay -MPR-). Ils sont transmis périodiquement à tous les voisins à 1 saut.
- **Topology Control (TC)** : Les messages TC sont utilisés pour construire la table de routage. Ce sont des messages d'état de liaison, diffusés périodiquement dans le réseau.

MP-OLSR est similaires à OLSR dans les premières phases dans la mesure où il utilise l'envoi périodique et proactif des messages HELLO afin de mettre à jour les voisinages et les relais multipoints, ainsi que les messages de topologies (TC) afin de transmettre l'information de voisinage au reste du réseau. Ces derniers messages permettent à chaque nœud de maintenir une vision globale sur les informations de la topologie dans le réseau à chaque instant.

Pour le calcul des chemin, à la différence d'OLSR, MP-OLSR ne maintient pas les tables de routage en permanence. MP-OLSR s'apparente à algorithme de recherche réactif : l'information de topologie est bien stockée en permanence mais elle ne sera exploitée qu'à la demande. MP-OLSR affecte des poids à chaque lien, ces poids sont ensuite envoyés au reste du réseau via les messages TC. Ainsi, la table de topologie de chaque nœud dispose d'attributs supplémentaires relatifs au poids des liens.

2.4.8 LOADng (Lightweight On-demand Ad hoc Distance-vector Routing Protocol–Next Generation)

LOADng est un protocole de routage à vecteur de distance et réactif [25] dérivant d'AODV. Ainsi, LOADng établit un itinéraire vers une destination donnée uniquement sur demande lorsqu'il y a des données à envoyer.

Il garde le même principe de fonctionnement de base d'AODV, sauf que certains aspects sont simplifiés dans le but de réduire la complexité du protocole et la quantité de ressources de calcul nécessaires à son exécution. Parmi les simplifications, un nœud intermédiaire peut ne pas répondre à un message qui ne lui est pas destiné, ce qui induit une baisse de l'utilisation des messages de contrôle [36].

Nous rappelons que le processus de découverte des route est effectué à l'aide de messages de contrôle inspirés d'AODV [21].

Fonctionnment

Lorsqu'un nœud souhaite envoyer un message de données et que l'itinéraire vers la destination est inconnu, il doit commencer un nouveau processus de découverte d'itinéraire.

Le nœud diffuse un message de demande d'itinéraire (RREQ) pour rechercher un itinéraire vers la destination souhaitée. Chaque nœud qui reçoit un RREQ doit effectuer le traitement des messages et considérer le message à transmettre.

Ce processus se poursuit jusqu'à ce que le RREQ atteigne la destination recherchée. La destination doit alors générer un message de réponse d'itinéraire (RREP) pour répondre au RREQ reçu. Le RREP est transmis en unicast à l'expéditeur RREQ, construisant une route entre les deux nœuds intéressés par l'échange de messages. Enfin, le RREP est reçu par l'expéditeur RREQ, qui devrait commencer à envoyer des messages de données en utilisant le chemin créé par le processus de découverte d'itinéraire [25].

2.4.9 Ipv6 Routing Protocol for Low power and Lossy networks (RPL)

Le protocole RPL [38] est un protocole de routage IPv6 développé par l'IETF (Internet Engineering Task Force) pour répondre aux besoins des réseaux à perte et à faible puissance. Il a été optimisé pour prendre en charge le trafic multipoint à point (des périphériques à l'intérieur du Low Power and Lossy network (LLN) vers un centre de contrôle), le trafic point à multipoint (du point de contrôle central aux périphériques à l'intérieur du LLN), et le trafic point à point (du point de contrôle central à un peripherique specifique à l'intérieur du LLN).

2.4.9.1 Construction de la topologie

Étant donné un ensemble de nœuds déployés de façon planifiée ou aléatoire, le RPL organise sa topologie en un graphe acyclique orienté vers une destination (la racine ou root) appelé DODAG (Destination-Oriented Directed Acyclic Graph).

Initialement, seule la racine fait partie intégrante de la topologie. Elle envoie périodiquement dans son voisinage des informations de configuration dans des messages de contrôle RPL spécifiques appelés DIO (DODAG Information Objet). Les nœuds qui reçoivent ces informations peuvent rejoindre le réseau en choisissant leur prochain saut (parent RPL) vers la racine. Dès qu'il fait partie intégrante de la topologie RPL, le nœud commence par envoyer ses propres DIO. Lorsqu'un nœud reçoit plusieurs DIO consistants émanant de voisins différents, un choix doit être opéré pour sélectionner parmi ceux-ci, celui offrant le meilleur coût au regard de l'objectif recherché et des contraintes imposées par l'application [33].

2.4.9.2 Les messages de contrôle RPL

Pour mettre en place et maintenir sa topologie de routage, RPL utilise quatre messages de controle [17, 25] :

- **DIO (DODAG Information Object)** : Ce message est utilisé pour la création des routes ascendantes. IL contient aussi les paramètres de configuration d'une instance RPL permettant ainsi à un nœd de calculer son rang et de choisir son parent. Il est envoyé d'un nœud parent a son fils [25].
- **DIS (DODAG Information Sollicitation)** : Il est utilisé par un nœud pour rejoindre la topologie ou réclamant des informations de configuration plus récentes [25].
- **DAO Destination Advertisement Object)** : il est responsable de la construction des routes descendantes. Il est envoyé d'un nœud fils à son parent [25].
- **DAO-Ack (DAO-Acknowledgment)** : Il est utilisé par la destination en réponse à un message DAO réçu [25].

Le DIO est prévu pour transporter de nombreuses options, toutefois certaines informations sont obligatoires. Parmi celles-ci on a :

- **RPLinstanceID** : C'est un identifiant unique du réseau, voir Figure 2.5.
- **DODAGID** : C'est l'identifiant d'un DODAG, il est égal à l'adresse IPv6 de la racine. Le tuple (RPLInstanceID, DODAGID) identifie de manière unique un DODAG dans le réseau.
- **DODAGVersionNumber** : C'est un compteur séquentiel qui est incrémenté par la racine lors de la formation une nouvelle version du DODAG, voir Figure 2.6.

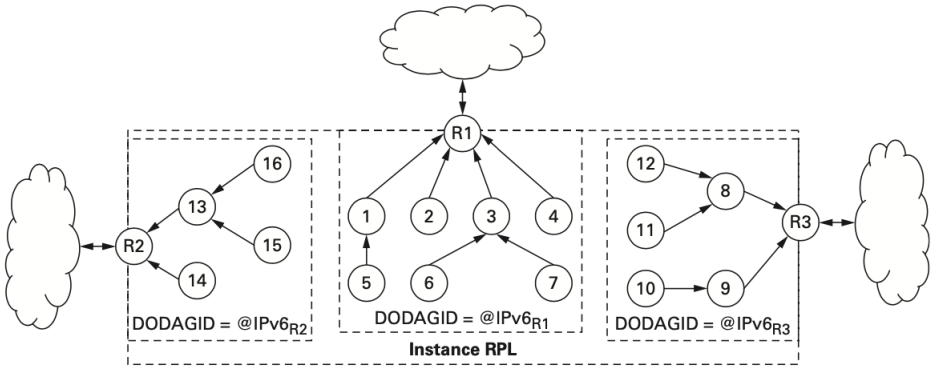

FIGURE 2.5 – Plusieurs DODAG dans une seule InstantRPL [33]

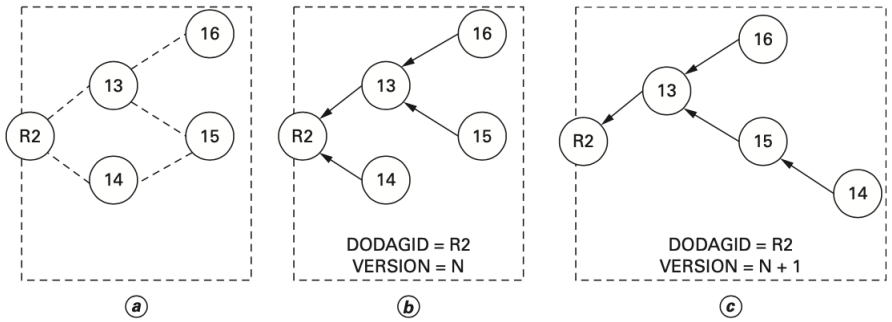

FIGURE 2.6 – Reconfiguration du DODAG suite à un changement de version [33]

2.4.9.3 Rang d'un nœud

A tout moment les nœuds utilisant RPL maintiennent un nombre qui determine leur position par rapport à la racine, c'est le rang du nœud. En effet un nœud calcule son rang lorsqu'il rejoint un DODAG. Le calcul du rang est pris en charge par la fonction objectif et est toujours lié à la metrique utilisé.

Dans un DODAG le rang augmente en fonction qu'on s'eloigne de la racine. Ainsi en plus de garantir une hiérarchie le rang permet d'eviter la formation des boucles de routage dans le reseau.

2.4.9.4 Mécanismes de réparation des routes

Dans RPL, il en existe deux qui sont :

— La réparation locale

— La réparation globale

La réparation globale

La réparation globale est un processus déclenché par la RPL racine. Pour cela, elle incrémente son numéro de version DODAG-VersionNumber ce qui entraîne la construction d'une nouvelle topologie en autorisant tous les nœuds du réseau à pouvoir choisir une nouvelle position dans le DODAG sans être contraint par leur rang dans l'ancienne version du DODAG [33].

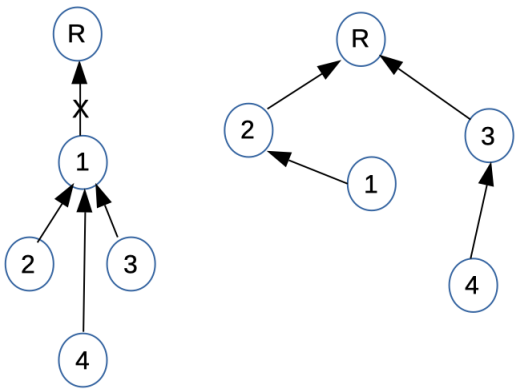

FIGURE 2.7 – Exemple de réparation gloale

La réparation locale

RPL utilise se processus lorsqu'un nœud n'a plus de parents potentiels disponibles. Elle permet au nœud de choisir un parent préféré sans prendre en compte son propre rang. La réparation locale s'accompagne d'un empoisonnement de routes c'est à dire le nœud envoie un DIO avec un rang infini , pour avertir les fils du nœud qu'il n'est plus en mesure d'offrir une connectivité vers la racine. Les fils à la réception du DIO avec rang infini ont alors

connaissance du fait que leur parent actuel ne peut plus être considéré comme un parent et déclenchent alors la sélection d'un nouveau parent [33].

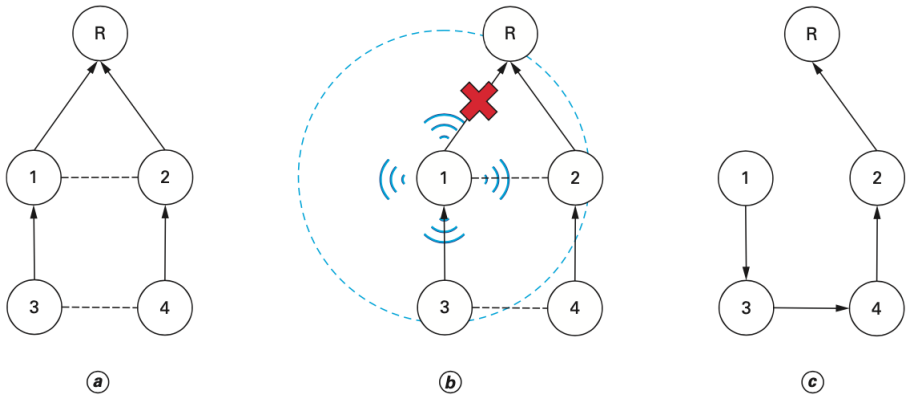

FIGURE 2.8 – Exemple de réparation locale [33]

2.4.9.5 Fonctions objectif dans RPL

La fonction objectif permet de spécifier comment les métriques de routage sont transformées en rang. Elle est également responsable de la définition de la manière dont le nœud sélectionne le meilleur parent de la liste de ses parents potentiels. L'IETF n'a défini que deux fonctions d'objectif pour RPL [38, 17] :

OF0 (Objective Function 0 (OF0)

Elle utilise le nombre de sauts comme métrique pour la selection du meilleur parent et le calcul du rang [33].

Methode de selection du parent dans OF0

Dans OF0, la selection du parent préféré obéit aux règles suivantes :

— le rang du parent préféré doit être inférieur au rang du nœud ;

— le nœud doit posséder une connectivité avec le parent préféré ;
— le parent offrant la meilleure connectivité avec la racine doit être selectionné ;
— lors de la comparaison de deux parents appartement au même DODAG, le parent ayant une connectivité avec la version la plus récente du DODAG est selectionné ;
— le parent ayant le plus petit rang par rapport au nœud est selectionné.

```
best_parent(rpl_parent_t *p1, rpl_parent_t *p2)
{
  rpl_rank_t r1, r2;
  rpl_dag_t *dag;

  PRINTF("RPL: Comparing parent ");
  PRINT6ADDR(rpl_get_parent_ipaddr(p1));
  PRINTF(" (confidence %d, rank %d) with parent ",
        p1->link_metric, p1->rank);
  PRINT6ADDR(rpl_get_parent_ipaddr(p2));
  PRINTF(" (confidence %d, rank %d)\n",
        p2->link_metric, p2->rank);


  r1 = DAG_RANK(p1->rank, p1->dag->instance) * RPL_MIN_HOPRANKINC  +
         p1->link_metric;
  r2 = DAG_RANK(p2->rank, p1->dag->instance) * RPL_MIN_HOPRANKINC  +
         p2->link_metric;
  /* Compare two parents by looking both and their rank and at the ETX
     for that parent. We choose the parent that has the most
     favourable combination. */

  dag = (rpl_dag_t *)p1->dag; /* Both parents must be in the same DAG. */
  if(r1 < r2 + MIN_DIFFERENCE &&
     r1 > r2 - MIN_DIFFERENCE) {
    return dag->preferred_parent;
  } else if(r1 < r2) {
    return p1;
  } else {
    return p2;
```

FIGURE 2.9 – seclection du meilleur parent dans OF0

MRHOF (Minimum Rank Hysterisis Objective Function)

Elle utilise comme métrique principale l'ETX, pour le calcul du rang et la selection du parent préféré. D'autres métriques additives

sont aussi implémentées, notamment l'energie, le lien métrique et le nombre de sauts.

Methode de selection du parent dans MRHOF

Afin de sélectionner un parent, MRHOF introduit une fonction hystérésis qui dont le pseudo-code est listé ci-dessous [33] :

— *Soit P1 et P2 les coûts de chemins respectifs d'un parent 1 et d'un parent 2.*
— *P1 étant le parent préféré et P2 un parent candidat.*
— *Si P1CoûtChemin – P2CoûtChemin > SeuilParent*
— *Le parent P2 devient le parent préféré.*
— *Sinon*
— *P1 reste le parent préféré.*
— *FinSi*

SeuilParent représente la fonction d'hystérésis, c'est-à-dire la différence minimale entre le coût du chemin du parent préféré et le coût du chemin du parent candidat qui déclenche la sélection d'un nouveau parent préféré.

L'ensemble des parents est construit de manière à respecter les règles suivantes :

— les nœuds appartenant à l'ensemble des parents pour un nœud donné doivent avoir une connectivité avec celui-ci ;
— la racine du DODAG doit avoir un ensemble de parents de taille 0 (c.-à-d., la racine ne doit pas avoir de parents)
— le parent préféré doit être membre de l'ensemble des parents ;
— le rang d'un nœud doit être supérieur à celui des nœuds constituants l'ensemble des parents.

2.4.9.6 Description du contenu du répertoire RPL dans contiki Os

Dans contiki, le repertoire RPL se trouve dans **contiki/core/net/rpl**. Il contient plusieurs types de fichiers, à savoir le makefile, des fichier .c et .h :

```
static rpl_parent_t *
best_parent(rpl_parent_t *p1, rpl_parent_t *p2)
{
  rpl_dag_t *dag;
  rpl_path_metric_t min_diff;
  rpl_path_metric_t p1_metric;
  rpl_path_metric_t p2_metric;

  dag = p1->dag; /* Both parents are in the same DAG. */

  min_diff = RPL_DAG_MC_ETX_DIVISOR /
             PARENT_SWITCH_THRESHOLD_DIV;

  p1_metric = calculate_path_metric(p1);
  p2_metric = calculate_path_metric(p2);

  /* Maintain stability of the preferred parent in case of similar ranks. */
  if(p1 == dag->preferred_parent || p2 == dag->preferred_parent) {
    if(p1_metric < p2_metric + min_diff &&
       p1_metric > p2_metric - min_diff) {
      PRINTF("RPL: MRHOF hysteresis: %u <= %u <= %u\n",
             p2_metric - min_diff,
             p1_metric,
             p2_metric + min_diff);
      return dag->preferred_parent;
    }
  }

  return p1_metric < p2_metric ? p1 : p2;
}
```

FIGURE 2.10 – seclection du meilleur parent dans MRHOF

— rpl.c est le programme principal qui asssure l'initiation et le contrôle du protocole. Il surcharge plusieurs fonctions d'autre progamme comme uip-ds6.c qui fait appel les fonctions de maintien de route. Il a pour entête :
  — rpl.h qui contient les structures, les prototypes des fonctons et des variables.
— rpl-dag.c contient toutes les fonctions relatives à la création et au maintien du DODAG.
— rpl-icmp6.c se charge principalement des messages de contrôle de RPL. On retrouve particulierement les formats des en têtes et les champs des messages DIO, DAO, DIS, DAO-ACK.
— rpl-ext-header.c est le programme qui est chargé de la gestion des extentions d'entête créées par la surcharge des fonc-

tions dans rpl.c.
— rpl-timers.c assure la gestion de tous les timers relatifs à la regulation des messages de contrôle du RPL.
— rpl-mrhof.c est le programme qui assure le calcul du rang selon la fonction objectif MRHOF.
— rpl-of0.c est le programme qui assure le calcul du rang selon la fonction objectif OF0.
— rpl-conf.h, rpl-private.h contiennent tous les paramétres de configurations de RPL.

2.4.9.7 RPL et routage multi-chemins

Pour garantir un routage multi-chemins, RPL offre la possibilité de sélectionner d'autres parents autres que le parent préféré depuis l'ensemble des parents.

L'ensemble des parents est construit de manière à respecter les règles suivantes [33] :
— les nœuds appartenant à l'ensemble des parents pour un nœud donné doivent avoir une connectivité avec celui-ci ;
— la racine du DODAG doit avoir un ensemble des parents de taille 0 ;
— un nœud qui n'est pas la racine du DODAG essaye de maintenir une taille de l'ensemble des parents supérieure ou égale à 1 ;
— le parent préféré doit être membre de l'ensemble des parents ;
— le rang d'un nœud doit être supérieur à celui des nœuds constituants l'ensemble des parents.

Un nœud étblit, en plus de la route par défaut vers son parent préféré, des routes vers les nœuds appartenant à son ensemble de parents.

Par défaut cette fonctionnalité n'est pas activée sur RPL. Mais par contre à chaque fois qu'un nœud choisit son parent préféré et établit sa route par défaut, il choisit un voisin comme potentiel

parent préféré pour s'en servir en cas de défaillance de la route par défaut .

Donc nous pouvons dire que RPL est un protcole de routage mono-chemin avec la possibilité de basculer en mode routage multi-chemins en cas de rupture de liens.

2.5 Tableau comparatif des protocoles de routage multi-chemins

Caractéristiques / Protocoles	Pas de boucle de routage	Chemins à nœuds ou lien disjoints	Chemin complètement connu par la source	Métrique	Chemins utilisés simultanément	Implémentation de base (protocole mono-chemin)
RPL-LB	Oui	Possible	Non	Nombre de sauts	Non	RPL
µLOADng	Oui	Possible	Non	Nombre de sauts	Information non disponible	LOADng
AODVM	Oui	Oui	Non	Nombre de sauts	Oui	AODV
AOMDV	Oui	possible	Non	Nombre de sauts	Possible	AODV
MP-DSR	Oui	Oui	Oui	Fiabilité de bout-en-bout	Oui	DSR
MP-OLSR	Oui	Oui	Possible	Nombre de sauts	Oui	OLSR
MSR	Oui	Oui	Oui	Délai aller/retour	Oui	DSR
SMR	Oui	Possible	Oui	Nombre de sauts	Oui	DSR
MDSDV	Oui	Oui	Oui	Nombre de sauts	Possible	DSDV

FIGURE 2.11 – Tableau comparatif des protocoles MC [27]

2.6 Conclusion

Dans ce chapitre, nous avons décrit les principales solutions de routage multi-chemins pour les RCSFs selon les études qui ont été ménées jusqu'à ce jour. Nous avons donné une description générale des différents protocoles de routage selon leur principe de fonctionnement et de découverte de chemins.

La majorité des protocoles de routage existants se base sur le critère du nombre de sauts ou sur un critère de qualité de service lié à un besoin applicatif pour la sélection des chemins. Nous avons décrit les types de chemins multiples et les métriques qui se basent sur des techniques d'évaluation de la qualité des liens comme ETX et de la durée de vie du réseau ELT.

La technique de répartition des données à router sur les différents chemins multiples est à spécifier selon le cas d'usage tout en tenant compte de la complexité de l'approche et du gain en performance réseau.

Dans le prochain Chapitre, nous allons décrire l'implémentation de notre solution et l'évaluation des performances de notre réseau de surveillance à l'aide du simulateur Cooja/Contiki, et ce en termes de nombre de sauts, nombre de paquets reçu par le sink, latence et consommation d'énergie.

CHAPITRE 3

Implémentation et évaluation des performances

3.1 Introduction

Dans ce chapitre, nous allons tout d'abord décrire la solution proposée pour optimiser le déploiement déterministe des nœuds capteurs de notre RCSF de surveillance, ainsi que l'amélioration apportée au protocole de routage RPL (Ipv6 Routing Protocol for Low power and Lossy Networks) pour lui permettre de faire de l'équilibrage de charge. Ensuite, nous présentons les résultats de la résolution des deux BILPs (Binary Integer Linear Problem). En fin, nous analysons et discutons les résultats de la simulation que nous avons conduite pour évaluer les performances de notre RCSF.

3.2 Optimisation du déploiement

3.2.1 Discrétisation de la zone à surveiller

Nous tenons à rappeler que, dans notre cas, le déploiement des nœuds du RCSF dédié à la surveillance de la zone sensible,

se fait d'une manière déterministe. Nous avons considéré deux types de nœuds : les nœuds capteurs sentinelles qui sont déployés à la frontière de la zone à surveiller et les nœuds relais qui sont palcés à l'intérieur de la zone afin d'acheminer les alertes, en cas d'intrusion, vers la station de base (le sink). Cette topologie est dite toplogie 2-tiers [7]

Nous avons discrétisé la zone à surveiller en une grille carrée à deux dimensions constituant ainsi un ensemble de mailles. Chacune de ses mailles est discrétisée (divisée) à son tour en un ensemble de quatre (4) sous-mailles, dont leurs centres respectifs sont des positions candidates avec un indice ij tel que j est compris entre (0,1,2,3) et i de (0 à (n-1)) avec "n" le nombre de mailles principales dans la grille .

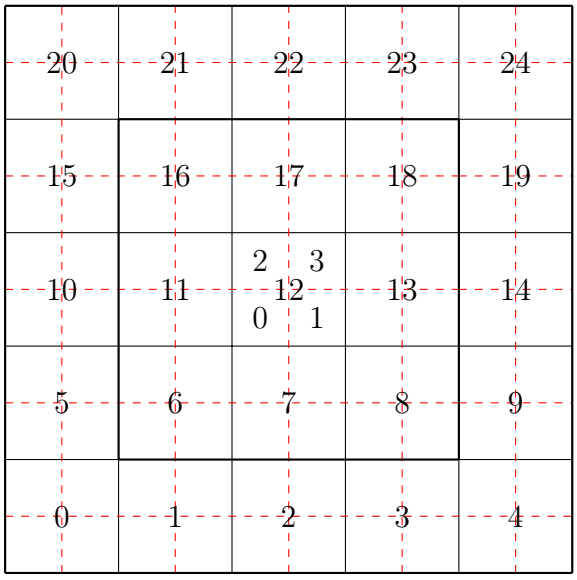

FIGURE 3.1 – Discrétisation de la zone en grille 5x5

Les coordonnées cartésiennes des positions ij des sous-mailles

sont données par les formules suivantes :

$$\begin{cases} x_{i0} = (i\ mod\ Grid\ ) * pas + \frac{pas}{2} - \frac{pas}{4} \\ y_{i0} = (i\ div\ Grid\ ) * pas + \frac{pas}{2} - \frac{pas}{4} \\ x_{i1} = (i\ mod\ Grid\ ) * pas + \frac{pas}{2} + \frac{pas}{4} \\ y_{i1} = (i\ div\ Grid\ ) * pas + \frac{pas}{2} - \frac{pas}{4} \\ x_{i2} = (i\ mod\ Grid\ ) * pas + \frac{pas}{2} - \frac{pas}{4} \\ y_{i2} = (i\ div\ Grid\ ) * pas + \frac{pas}{2} + \frac{pas}{4} \\ x_{i3} = (i\ mod\ Grid\ ) * pas + \frac{pas}{2} + \frac{pas}{4} \\ y_{i3} = (i\ div\ Grid\ ) * pas + \frac{pas}{2} + \frac{pas}{4} \end{cases}$$

Où :

- Grid = $\sqrt{n}$
- pas est la distance entre les centres des deux (2) mailles adjacentes.
- mod est le reste de la division .
- div est le quotient de la division .

3.2.2 Formalisation du problème d'optimisation

Notre problème d'optimisation consiste, en premier lieu, à minimiser le nombre de sauts depuis le nœud sentinelle le plus éloigné jusqu'à la station de base (le sink), c-à-d., la minimisation du diamètre du réseau, et de prendre ce nombre de sauts minimal K comme valeur seuil du coût des chemins (qui doivent être inférieur à K) entre les autres nœuds et la station de base (le sink). Pour ce faire, nous avons utilisé la méthode d'optimisation combinatoire multi-objectifs pour proposer un BILP (Binary Integer Linear Program).

En deuxième lieu, nous souhaitons minimiser le nombre de nœuds relais. Pour cela nous proposons un deuxième BILP2, ayant pour fonction objectif la minimisation du nombre de relais et qui a pour contrainte supplémentaire par rapport au premier BILP, à savoir la valeur du K obtenue en résolvant le premier BILP.

La méthode de résolution de notre PMO est dite méthode lexicographique [19]

3.2.2.1 Minimisation du diamètre du réseau : BILP 1

L'objectif de ce programme linéaire est de minimiser le nombre de sauts depuis le nœud sentinelle (SN) le plus éloigné dans le RCSF jusqu'au Sink.

Description des variables

- S : Ensemble des positions possibles pour les nœuds sentinelles
- R : Ensemble des positions possibles pour les nœuds relais
- P : Ensemble des points (mailles) à couvrir
- S^p : Ensemble des positions possibles $\in S$, pouvant couvrir le point $p \in P$
- V_{ij}^1 : Ensemble des positions candidates $\in R$,situées à un saut d'un nœud $\in (S \cup R)$ qui est déployé à la position ij
- V_{sink}^1 : Ensemble des positions possibles $\in (S \cup R)$, situées à un saut du sink
- Cov : Nombre minimum de nœuds capteurs sentinelles qui doivent couvrir le point $p \in P$ (assure la Cov-Couverture)
- m : Nombre minimum de connexions dont doit disposer un nœud sentinelle (assure la m-connectivité).
- $|S|$: Nombre de nœuds sentinelles
- K : Nombre maximum de sauts autorisé

Variables de décision

- x_{ij}^k ($\in \{0,1\}$, $k = 1 \ldots K+1$) : Nœud capteur sentinelle déployé à la position ij à k sauts du sink
- r_{ij}^k ($\in \{0,1\}, k = 1 \ldots K$) : Nœud relais déployé à la position

ij à k sauts du sink
— y_k ($\in \{0,1\}$, $k = 1 \dots K$) : Il existe un nœud relais déployé à k sauts du sink
— z_{ij} ($\in \{0,1\}$) : Nœud capteur sentinelle déployé à la position ij
— t_{ij} ($\in \{0,1\}$) : Nœud relais déployé à la position ij

Fonction objectif

$$\min \quad \sum_{k=1}^{K} y_k$$

Les contraintes

$$\sum_{k=1}^{K+1} x_{ij}^k \leq 1 \qquad ij \in S \tag{3.1}$$

$$\sum_{k=1}^{K} r_{ij}^k \leq 1 \qquad ij \in R \tag{3.2}$$

$$\sum_{k=1}^{K+1} \sum_{j=0}^{3} x_{ij}^k \geq Cov \qquad i \in P \tag{3.3}$$

$$mx_{ij}^k \leq \sum_{pq \in V_{ij}^1} r_{pq}^{k-1} \qquad ij \in S \backslash V_{sink}^1, \quad k = 2 \ldots K+1 \tag{3.4}$$

$$mr_{ij}^k \leq \sum_{pq \in V_{ij}^1} r_{pq}^{k-1} \qquad ij \in R \backslash V_{sink}^1, \quad k = 2 \ldots K \tag{3.5}$$

$$mx_{ij}^1 \leq 1 \qquad ij \in V_{sink}^1 \backslash R \tag{3.6}$$

$$mx_{ij}^k \leq \sum_{pq \in V_{ij}^1} r_{pq}^{k-1} \qquad ij \in V_{sink}^1 \backslash R, \quad k = 2 \ldots K+1 \tag{3.7}$$

$$x_{ij}^1 = 0 \qquad ij \in S \backslash V_{sink}^1 \tag{3.8}$$

$$r_{ij}^1 = 0 \qquad ij \in R \backslash V_{sink}^1 \tag{3.9}$$

$$y_k \geq r_{ij}^k \qquad ij \in R, \quad k = 1 \ldots K \tag{3.10}$$

$$y_k \leq \sum_{ij \in R} r_{ij}^k \qquad k = 1 \ldots K \tag{3.11}$$

$$y_k \leq y_{k-1} \qquad k = 2 \ldots K \tag{3.12}$$

$$y_k \leq \sum_{l=k+1}^{K+1} \sum_{ij \in S} x_{ij}^l \qquad k = 1 \ldots K \tag{3.13}$$

$$z_{ij} \geq x_{ij}^k \qquad k = 1 \ldots K+1 \tag{3.14}$$

$$z_{ij} \leq \sum_{k=1}^{K+1} x_{ij}^k \qquad ij \in S \tag{3.15}$$

$$t_{ij} \geq r_{ij}^k \qquad k = 1 \ldots K \tag{3.16}$$

$$t_{ij} \leq \sum_{k=1}^{K} r_{ij}^{k} \qquad k = 1 \ldots K \tag{3.17}$$

$$\sum_{ij \in S} z_{ij} \leq |S| \tag{3.18}$$

Description des contraintes

— La contrainte (3.1) garantit qu'un seul nœud sentinelle est déployé par sous maille.
— La contrainte (3.2) garantit qu'un seul nœud relais est déployé par sous maille.
— La contrainte (3.3) assure que chaque point $p \in P$ est couvert par au moins Cov nœuds capteurs sentinelles qui sont situés à k sauts du Sink.
— La contrainte (3.4) garantit que chaque nœud sentinelle, situé à k sauts du Sink ($k > 1$), a au moins m nœuds voisins de type relais situés à $k - 1$ sauts du Sink.
— La contrainte (3.5) garantit que chaque nœud relais, situé à k sauts du Sink ($k > 1$), a au moins m nœuds voisins de type relais situés à $k - 1$ sauts du Sink.
— La contrainte (3.6) sert à imposer qu'un nœud capteur voisin direct du sink ne soit pas considéré comme un nœud à 1 saut du sink et par conséquent lui trouver m chemins à k sauts ($k \geq 2$) (voir contrainte 3.7.
— La contrainte (3.7) assure qu'au moins m chemins à k sauts ($k \geq 2$) soient disponibles pour les nœuds capteurs voisins directs du sink.
— La contrainte (3.8) évite l'incohérence du modèle, c.-à-d., aucun nœud de type sentinelle n'est voisin du Sink que s'il est situé à 1 saut du Sink.
— La contrainte (3.9) évite l'incohérence du modèle, c.-à-d., aucun nœud de type relais n'est voisin du Sink que s'il est situé à 1 saut du Sink.

— La contrainte (3.10) détermine la longeur des chemins constitués de nœuds relais vers le Sink, s'il y a un nœud relais dans la ij^{eme} position situé à k sauts du Sink alors il existe un chemin de longueur k depuis un nœud relais vers le Sink.
— La contrainte (3.11) assure que s'il y a un chemin de longueur k sauts, cela signifie qu'il y a au moins un nœud relais dans la ij^{eme} position situé à k sauts du Sink et cela pour éviter l'incohérence du modèle.
— La contrainte (3.12) garantie que l'existence d'un chemin de longueur k sauts induit la présence d'un chemin de longueur $k-1$ sauts.
— La contrainte (3.13) garantit que si un nœud relais est déployé à k sauts du sink, un nœud capteur sentinelle est forcément déployé à l sauts du sink ($l = (k+1) \vee l = (k+2) \vee \ldots \vee l = (K+1)$)
— La contrainte (3.14) assure que s'il existe un nœud sentinelle dans la ij^{eme} position situé à k sauts du Sink, il sera réellement deployé dans la ij^{eme} position.
— La contrainte (3.15) assure que si un nœud sentinelle est déployé à la ij^{eme} position alors il y a forcément un nœud sentinelle à la ij^{eme} position situé à k sauts du Sink pour éviter l'incohérence du modèle.
— La contrainte (3.16) assure que s'il y a un nœud relais dans la ij^{eme} position situé à k sauts du Sink, alors il sera réellement déployé dans la ij^{eme} position.
— La contrainte (3.17) assure que si un nœud relais est déployé à la ij^{eme} position, alors il y a forcément un nœud relais à la ij^{eme} position situé à k sauts du Sink pour éviter l'incohérence du modèle.
— La contrainte (3.18) impose que le nombre de nœuds capteurs deployés soit inférieur au nombre de nœuds capteurs disponibles.

Solution obtenue après exécution du BILP 1

Les paramètres fournis en entrée sont :

Paramètre	Valeur
Grille	7x7
Nombre mailles principales	49
Nombre mailles de bordure	24
Nombre mailles internes	25
k-Couverture	k=3
m-Connectivité	m=2
Rayon de couverture	25 mètres
Rayon de communication	30 mètres
Nombre de positions possibles pour les sentinelles	96 $(24 * 4)$
Nombre de positions possibles pour les realis	100 $(25 * 4)$
Nombre de nœuds total	196

TABLE 3.1 – Paramètres du modèle BILP 1

Le déploiement résultant est illustré dans La Figure 3.2.

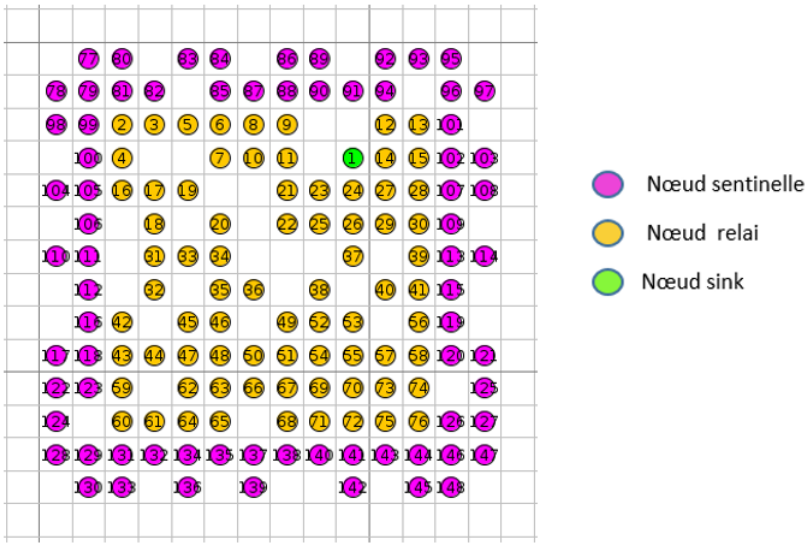

FIGURE 3.2 – Grille 7x7 résultat de la résoltion du BILP 1

3.2.2.2 Minimisation du nombre total de nœuds relais : BILP 2

L'objectif de ce modèle et de minimiser le nombre de nœuds relais total déployés afin d'impacter le coût de déploiement du RCSF et éventuellement réduire encore la longueur moyenne des chemins entre les nœuds sentinelles et le nœud puits (Sink). Nous tenons à rappeler que ce BILP utilise la valeur du nombre de sauts du plus long chemin (k) entre un nœud sentinelle et le nœud puits, obtenu après la résolution du précédent BILP.

Description des variables

Le BILP 2 utilise les mêmes variables que le BILP 1 avec une variable en plus.

— D : Longueur du diamètre obtenue en résolvant le premier BILP (BILP 1).

Variables de décision

Le BILP 2 utilise les mêmes variables de decision que le BILP 1

Fonction objectif

$$\min \quad \sum_{ij \in R} t_{ij}$$

Les contraintes

Toutes les contraintes du premier BILP 1 restent valables pour le deuxieme, avec en plus la contrainte suivante :

$$\sum_{k=1}^{K} y_k \leq D \tag{3.19}$$

Description des contraintes

— La contrainte (3.19) assure que la valeur du diamètre du réseau obtenu en résolvant le premier BILP soit prise en

compte.

Solution obtenue après exécution du BILP 2

Les paramètres fournis en entrée sont :

Paramètre	Valeur
Grille	7x7
Nombre mailles principales	49
Nombre mailles de bordure	24
Nombre mailles internes	25
k-Couverture	k=3
m-Connectivité	m=2
Rayon de couverture	25 mètres
Rayon de communication	30 mètres
Nombre de positions possibles pour les sentinelles	96 $(24 * 4)$
Nombre de positions possibles pour les realis	100 $(25 * 4)$
Nombre de nœuds total	196
D (Objectif de BILP 1)	**6**

TABLE 3.2 – Paramètres du modèle BILP 2

Le déploiement résultant est illustré dans la Figure 3.3.

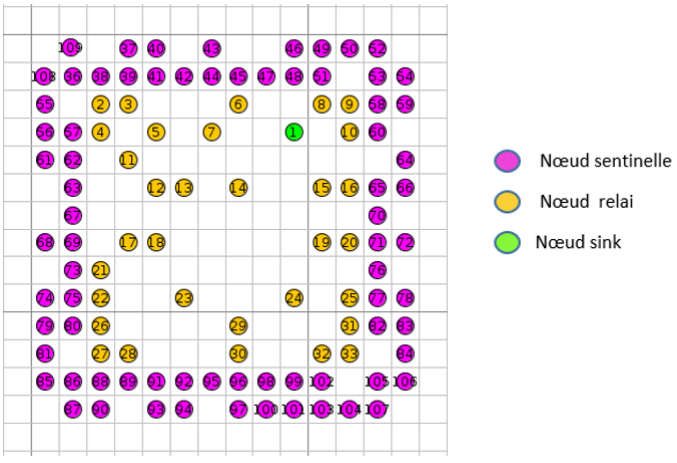

FIGURE 3.3 – Grille 7x7 résultat de la résolution du BILP 2

3.2.3 Tableau récapitulatif des résultats des deux BILP

Le Tableau de la Figure 3.4 récapitule les résultats obtenus après la résolution successive des deux BILP.

Notons que ces résultats vont être utilisés pour la simulation sous Contiki/Cooja.

Grille	BILP1: Minimisation du diamètre réseau					BILP2 : Minimisation du nombre de relais				
	Relais	Sentinelles	Total	**Diamètre**	Run Time (seconde)	Relais	Sentinelles	Total	**Diamètre**	RunTime (seconde)
5x5	21	48	69	**3**	0.28	18	48	66	**3**	0.60
7x7	75	72	147	**6**	0.69	32	74	106	**6**	129.7
10x10	139	110	249	**8**	5.3	-	-	-	-	+ 6 (heures) outofmemory

FIGURE 3.4 – Récapulatif des résultat des BILPs

3.3 Amélioration du RPL

Dans cette Section, nous décrivons les différentes modifications apportées à RPL pour le rendre multi-chemins actif, et de mettre en place un système d'équilibrage de charge.

3.3.1 RPL et multi-chemins actif

Les modifications apportées ont fait de sorte que les potentiels parents soient sélectionnés et ajoutés à la liste des parents. Ainsi tous les voisins sont considérés comme prochain saut vers une destination.

```
p = rpl_find_parent(dag, from);
if(p == NULL) {
  previous_dag = find_parent_dag(instance, from);
  if(previous_dag == NULL) {
    /* Add the DIO sender as a candidate parent. */
    p = rpl_add_parent(dag, dio, from);
    if(p == NULL) {
      PRINTF("RPL: Failed to add a new parent (");
      PRINT6ADDR(from);
      PRINTF(")\n");
      return;
    }
    PRINTF("RPL: New candidate parent with rank %u: ", (unsigned)p->rank);
    PRINT6ADDR(from);
    PRINTF("\n");
    instance->current_dag->preferred_parent =p;
```

FIGURE 3.5 – code permettant d'agrandir la liste des parents

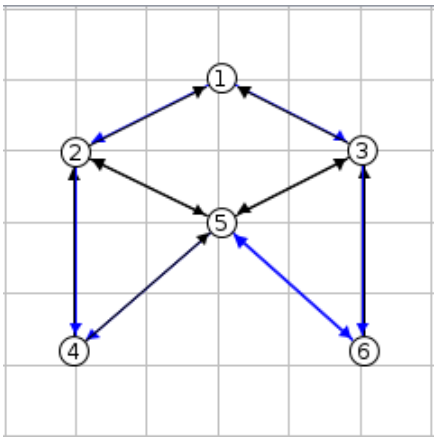

FIGURE 3.6 – RPL mutli-chemins

3.3.2 RPL et équilibrage de charge

Deux modifications principales ont été faites, l'une dans rpl-dag.c et l'autre dans uip-ds6-route.c.

Le processus de sélection du parent préféré dans RPL se fait de telle manière qu'une fois le meilleur voisin sélectionné (parent préféré), tous les autres voisins sont considérés comme étant des parents potentiels.

3.3.3 Mise en place du tourniquet

Une fois un voisin sélectionné comme meilleur parent, une route par défaut est automatiquement ajoutée, depuis un nœud jusqu'au sink, dont le prochain saut à partir du nœud est le parent préféré.

Dans la fonction **choose_route()** de **uip-ds6-route.c**, un tourniquet a été mis en place permettant à chaque nœud de choisir le parent en tête de la liste des parents et d'ajouter une route par défaut via ce parent et ensuite effacer ce parent de tête pour le mettre en fin de liste, et ainsi de suite .

```
for(d = list_head(defaultrouterlist);
    d != NULL;
    d = list_item_next(d)) {
  PRINTF("Defrt, IP address ");
  PRINT6ADDR(&d->ipaddr);
  PRINTF("\n");
  bestnbr = uip_ds6_nbr_lookup(&d->ipaddr);
  if(bestnbr != NULL && bestnbr->state != NBR_INCOMPLETE) {
    PRINTF("Defrt found, IP address ");
    PRINT6ADDR(&d->ipaddr);
    PRINTF("\n");
    list_remove(defaultrouterlist, d);
    list_add(defaultrouterlist, d);
    return &d->ipaddr;
  } else {
    addr = &d->ipaddr;
    PRINTF("Defrt INCOMPLETE found, IP address ");
    PRINT6ADDR(&d->ipaddr);
    PRINTF("\n");
  }
}
```

FIGURE 3.7 – Code du tourniquet

3.4 Evaluation des performances

Pour évaluer les performances de notre RCSF déployé d'une façon optimale, nous avons utilisé les paramètres décrits dans le Tableau 3.3.

Nous allons évalué les performances de notre modèle en termes **de nombre de sauts, latence, PDR et consommation d'énergie**, en

Par la suite nous déboucherons sur l'évaluation des améliorations portant sur le protocole RPL toujours en utilisant le model minimisant le nombre de relais.

Paramètre	Valeur
Surface du site	10000 m^2 , 194600 m^2 Grille 5*5, Grille 7*7
Taille d'une maille	20 mètres
Rayon de couverture	25 mètres
Rayon de communication	30 mètres
Temps de simulations	210 secondes
Temps debut de collecte	90 secondes
Temps fin de collecte	210 secondes
Duty cycle	100%
Nombre de simulation	2,2
Nombre de nœuds total	66,106
Nombre de nœuds relais	18,32
Nombre de nœuds sentinelle	48,74
Nombre de nœuds puits	1,1
Nombre de paquets envoyé par interval de temps	1 paquet chaque 2 secondes
Protocole mac	nullmac
Protocole de routage	RPL,RPL_LB
Fonction Objectif	OF0

TABLE 3.3 – Paramètres de simulation

3.4.1 Analyse des résultats de la simulation avec RPL sans amélioration

3.4.1.1 En termes de nombre de sauts

Dans la Figure 3.8, nous pouvons constater la variation moyenne (moyenne des deux simulations réalisées pour chaque grille) du nombre de sauts pour les deux grilles, en l'occurrence 5*5 et 7*7.

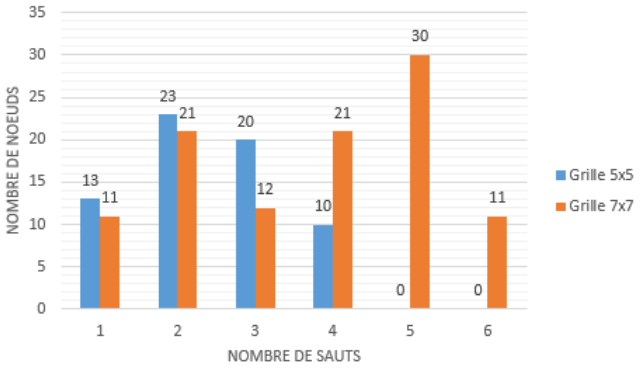

FIGURE 3.8 – Distribution des nœuds en fonction du nombre de sauts

Ainsi nous avons un nombre de saut moyen de 2.4 pour les deux simulations de la grille 5x5 et de 3.6 pour pour les deux simulations de la grille 7x7.

Nous tenons à faire remarquer qu'il y a un décalage entre le résultat du BILP 1 et le résultat de la simulation. En effet, partant d'un déploiement prévoyant un nombre maximal de 3 sauts pour la première grille on se retrouve avec 4 sauts. Ce décalage est du au protocole RPL, plus précisément au niveau de la fonction objectif. En effet c'est de la sélection du parent préféré que né ce décalage, car en plus de la distance par rapport à la racine du DODAG, l'ordre de diffusion et l'hystérésis interviennent aussi comme facteur dans la sélection. Contrairement au modèle qui ne factorise que la distance .

3.4.1.2 En termes de Packet Delivery Ratio (PDR)

La Figure 3.9 illustre le taux de délivrance des paquets dans les deux grilles.

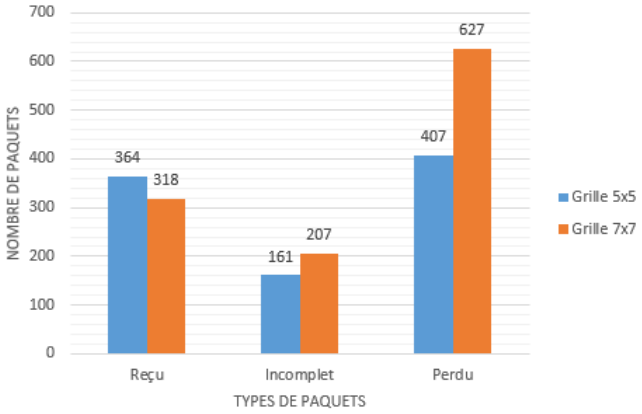

FIGURE 3.9 – Taux de délivrance des paquets

Nous avons un PDR de 56% et 45% respectivement pour les deux grilles. Nous avons près de 50% des paquets qui sont perdu. Cela est principalement du au nombre élevé de sentinelles pris en charge par les relais, en plus aucun protocole Mac n'est utiliser dans nos simulations on se retrouve alors avec beaucoup de collisions.

3.4.1.3 En termes de latence

La Figure 3.10 montre la variation de la latence.

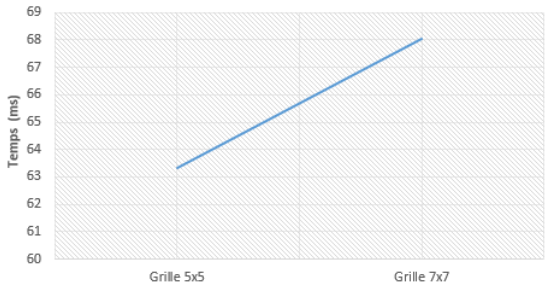

FIGURE 3.10 – La variation de la latence

La latence est de 63.33 ms pour la première grille et 68.02 ms pour la deuxième grille. Nous constatons que la latence augmente pour la grille 7*7, ce qui tout à fait normale vu le nombre supérieur d'envoi d'alertes.

3.4.1.4 En termes de consommation d'énergie

Dans la Figure 3.11, nous avons une répartition de la consommation d'énergie moyenne des nœuds en fonction des différents modes des nœuds, à savoir :

— CPU (mode traitement)
— listen (mode écoute du canal)
— Transmit (mode transmission de donnée)
— LPM (Low Power mode économie d'énergie)

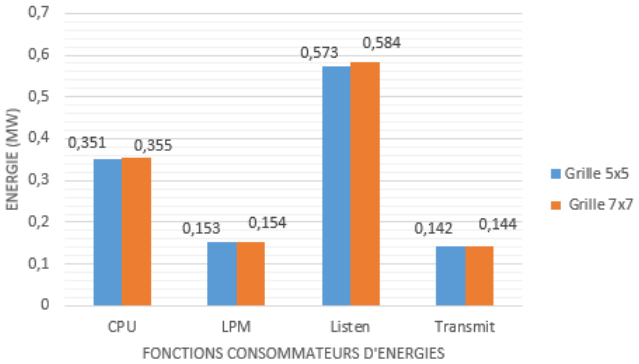

FIGURE 3.11 – Consommaton d'énergie moyenne par nœud (pour les différents modes enumérés ci-dessus)

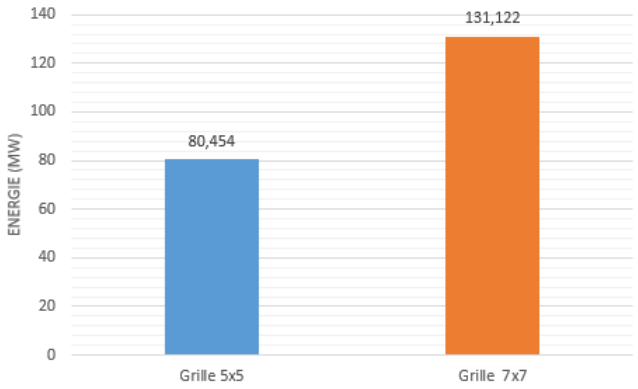

FIGURE 3.12 – Variation de la consommation d'énergie totale

Il en ressort que le mode CPU et listen sont les modes qui consomment le plus d'énergie. Ce qui s'explique par le fait qu'aucun des nœuds de notre RCSF ne s'endort (duty Cycle=1).

Dans la Figure 3.12, nous avons une courbe croissante qui montre la consommation d'energie totale pour les deux grilles. Ce qui est tout à fait normal.

3.4.2 Analyse des résultats des simulations avec RPL amélioré

3.4.2.1 En termes de nombre de sauts

Le nombre de sauts est obtenue dans cooja/contiki en analysant le DODAG. Cependant avec le RPl_LB, il est impossible d'obtenir le nombre de sauts exacte du reséau à cause du tourniquet qui empéche au DODAG de se stabiliser.

3.4.2.2 En termes de Packet Delivery Ratio (PDR)

Nous avons un PDR de 50% et 45% respectivement pour les deux grilles. En termes de PDR nous n'avons pratiquement pas d'amélioration étant donné que les deux configurations donnent presque les mêmes résultats. En effet, il est impossible d'obser-

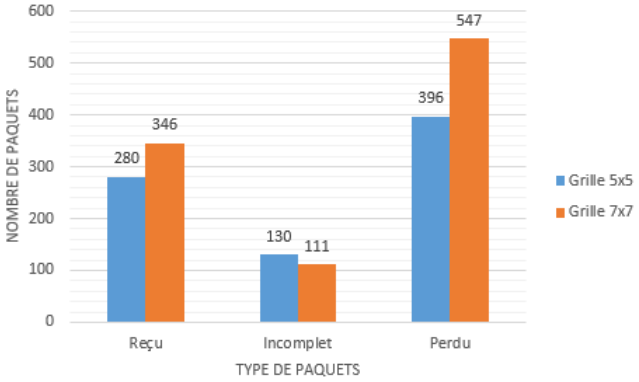

FIGURE 3.13 – Taux de délivrance des paquets

ver l'avantage du RPL amelioré avec la topologie proposée par le deuxiéme BILP car les collisions se produisent au niveau des relais à un saut des sentinelles. Or le multi-chemin commence généralement après les premiers relais à l'interieur de la grille avec l'utilisation de plusieurs routes pour atteindre le sink.

3.4.2.3 En termes de latence

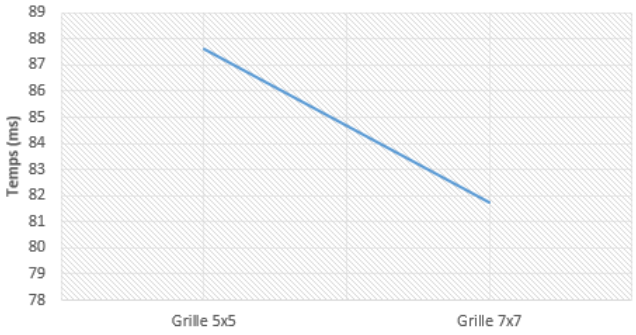

FIGURE 3.14 – Variation de la latence

La latence du réseau avec l'utilisation du RPL amélioré est de 87.63 ms pour la première simulation et de 81.72 ms pour la deuxième simulation. Une latence légèrement supérieure à la

latence du réseau utilisant le RPL normal.
Cette legère augmentation peut se traduire par le fait que les paquets n'utilisent pas tous des chemins optimals.

3.4.2.4 En termes de consommation d'énergie

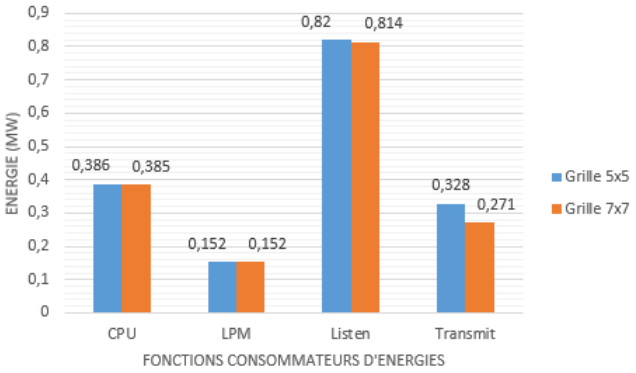

FIGURE 3.15 – Consommaton d'énergie moyenne par nœud (pour les différents modes)

Tout comme la Figure 3.11, la Figure 3.15 represente la repartion de la consommation d'energie moyenne des nœuds en utilisant le protocol RPL amelioré. Nous remarquons une net augmantation de la consommation d'energie par rapport au RPL nomal. Cela s'explique par le fait que notre RPL amolioré exige plus de traitement et de transmission.

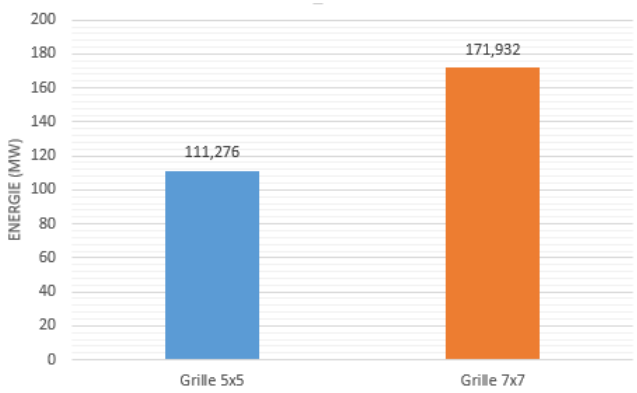

FIGURE 3.16 – Variation de la consommation d'énergie totale

La Figure 3.16 représente la consommation totale d'énergie du réseau et traduit encore l'augmentation de la consommation d'énergie.

En résumé nous pouvons nous satisfaire des resultats de simulation avec le RPL amelioré car en plus d'offrir le multi-chemins il ne s'eloigne pas des resultats avec le RPL normal.

3.4.3 Discussion des résultats

Suite aux analyses des résultats obtenus après nos simulations, dans cette section nous essayerons de discuter la concordance de notre model avec les simulations effectués. Nous discutons également un peu plus loin la validation des résultats face aux exigences de l'application.

— Pour ce qui est du respect des contraintes de notre modèle, nos simulations cadrent parfaitement car le fichier position de simulation est généré en sortie par le BILP. Et nous nous sommes assurées de la concordance des rayons de communication entre modèle et simulations.

— Concernant le nombre de saut et la latence on peut dire que les résultats sont acceptables parce que pour une surface de 10000 m^2 à 20000 m^2 avec un rayon de communication max

de 30 mètre n'importe quelle sentinelle peut atteindre le sink avec une moyenne de 3 sauts et une très bonne latence d'à peu près 85 milisecondes.
— La consommation d'énergie est également acceptable étant donné que c'est une application de surveillance et qu'aucune politique de gestion du canal n'est adoptée.
— Seul le PDR est jugé très faible avec près de 50% des paquets qui sont perdues ce qui n'est pas acceptable pour une application de surveillance qui ne peut pas tolérer 50% des intrusions.
— Les améliorations portées sur RPL sont également acceptables car elle nous permettent de tirer profit de la m-connectivité du réseau, en cas où il y aurait un trafic intense (éviter la congestion) et en cas de rupture de lien.

3.5 Conclusion

Dans ce chapitre, nous avons tout d'abord proposé deux BILP qui ont été utilisés pour optimiser le déploiement du RCSF de surveillance. Le premier BILP permet de minimiser le diamètre de notre réseau et un deuxième BILP minimise le nombre de relais du réseaux en ayant comme contrainte la valeur du diamètre donnée par le premier BILP. Cette méthode de résoltion d'un problème multi-objectifs s'appelle méthode lexicographique.

Nous avons ensuite présenté et commenté les résultats de la résolution des deux BILP. En fin, nous avons analysé les résultats de la simulation concernant la performance de notre réseau, en termes de nombre desauts, latence, PDR et consommation d'énergie. Nous estimons que ces résultats sont encourageants.

Conclusion générale et perspectives

Dans ce mémoire, nous nous sommes intéressés au problème d'optimisation multi-objectifs du déploiement d'un RCSF dédié à la surveillance d'un site sensible clos, et ce sous deux principalse contraintes à savoir, la k-couverture et la m-connectivité.

Notre solution a été construite comme suit.

Nous avons discrétisé la zone à surveiller en mailles carrées qui représentent les points à couvrir. Ensuite, nous avons divisé ces mailles principales en quatre (04) sous mailles dont le centre de chacune d'elle représente une position potentielle d'un nœud capteur (sentinelle ou relais).

Nous avons proposé un premier BILP (Binary Integer Linear Problem) qui minimise le nombre de sauts entre les nœud sentinelle le plus éloigné et le nœud sink. Cette minimisation va impacter la latence, le PDR (Packet Delivery Ratio) et la consommation de l'énergie.

Ensuite, nous avons utilisé le résultat du premier BILP comme contrainte dans un deuxième BILP minimise le nombre de nœuds relais dans le réseau. Cette minimisation permet de réduire le coût du déploiement.

D'autre, nous avons proposé une amélioration du protocole RPL le rendant multi-chemins avec equilibrage de charge, pour

pouvoir exploiter la m-connectivité imposée comme contrainte.

En fin, nous avons également procédé à des simulations en utilisant le simulateur Contiki/Cooja. Les résultats des simulations nous sembles satisfaisants.

Nous proposons comme perspectives :

— Prendre en compte le nombre de nœuds sentinelles pris en charge par un nœud relai lors de la minimisation du deuxieme BILP dans le but d'avoir un multi chemin à nœuds disjoint et d'éviter des collisions et favoriser un bon PDR.

— Penser à un protocole MAC (Media Access Control) adapté pour notre problématique afin d'avoir une politique de gestions du canal ce qui favorisera une fois de plus le PDR et même la consommation d'énergie.

— Développer un système de timer qui permettra à l'ensemble des sentinelles surveillant un même point de se synchroniser et de surveiller à tour de rôle. C'est-à-dire qu'à un instant donné un seul nøeud sentinelle sera actif et les autres se mettent en veille.

— Considéré l'énergie résiduel des nœuds pour le choix des routes dans RPL au lieu de l'emploi du tourniquet. Donc au lieu d'utiliser les routes à tour de rôle on utilisera la route dont les nœuds auront le plus d'énergie résiduelle.

— Prévoir une amélioration du programme collect-view du simulateur Contiki/Cooja qui dessine le DODAG parce que la lecture sur le graphe devient fastidieuse lorsqu'on dispose d'un nombre important de nœuds.

— Penser à l'utilisation d'une methode de resolution approchée avec les métaheuristiques afin d'avoir un temps d'execution raisonnable pour les données de grandes tailles.

ANNEXE A

Logiciels et plateformes utilisés

A.1 Architecture de l'application

Dans cette section nous decrivons les interactions de notre application :

- En premier lieu, l'utilisateur donne les paramétres (longueur de la surface à couvrir, taille des mailles, nombre des nœuds sentinelles, k, m, le rayon de couverture et le rayon de communiction) au programme MatLab qui à son tour génère un fichier.dat ;
- Ensuite le fichier.dat servira d'entre à notre programme.mod pour que après excution CPLEX nous génére un fichier position.dat ;
- Puis le fichier position.dat sera integré aux module de mobilité du Simulateur Cooja/Contiki ;
- Enfin on lance la simulation sur le simulateur Cooja/Contiki.

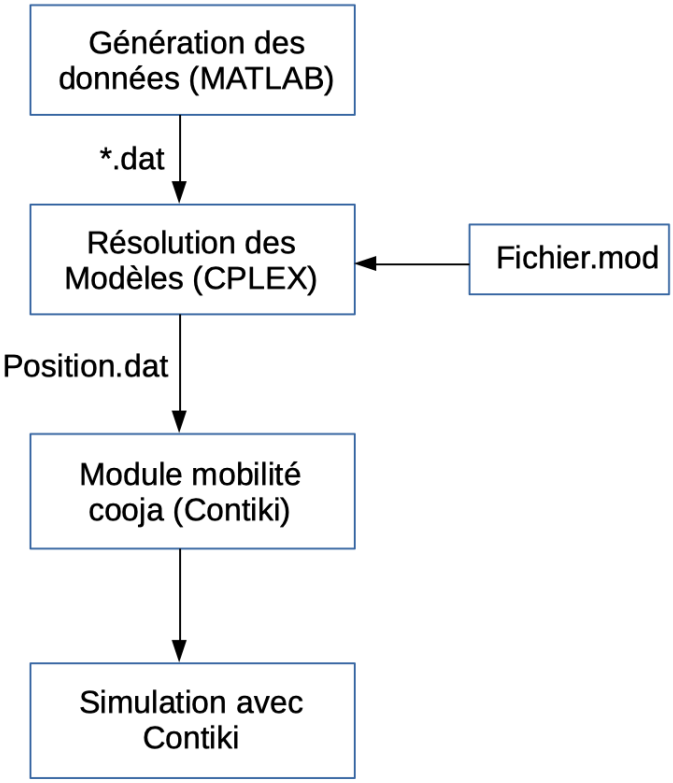

FIGURE A.1 – Architecture de l'application

A.2 Matlab

Matlab (MATrix LABoratory) est une application scientifique interactive orientée calcul vectoriel et matriciel, avec une puissante librairie de visualisation. C'est un logiciel payant proposant une interface graphique vers un éditeur de code et un outil de debugging pour exécuter des programmes en modes pas à pas.

Matlab propose différents frameworks (toolbox), avec des fonctionnalités très avancées permettant de réaliser facilement des tâches complexes (p. ex., toolbax équations différentielles, toolbox aérospatial).

A.2.1 Interface Matlab

L'interface Matlab se compose principalement de 6 (six) zones comme indiqué dans la Figure ci-dessous. Chaque zone possède

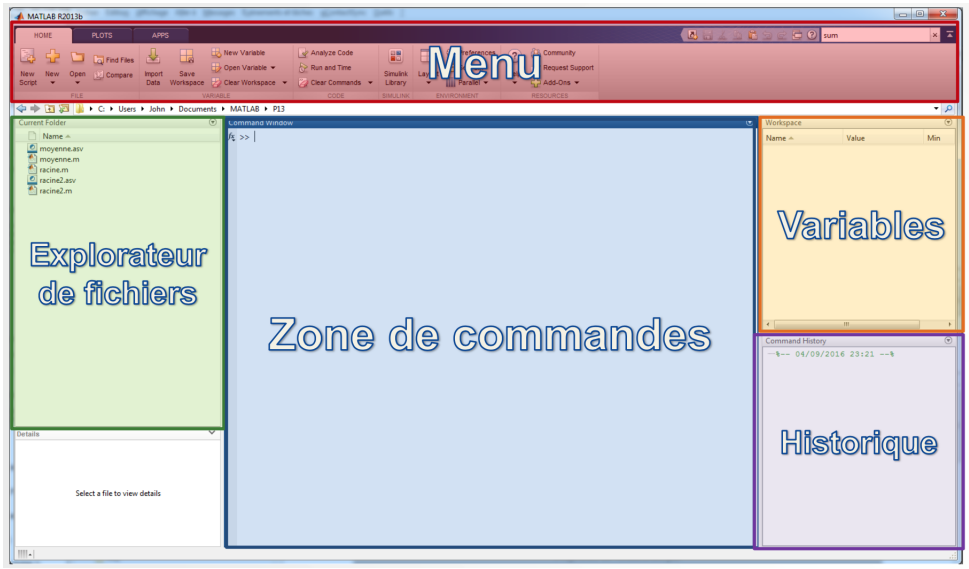

FIGURE A.2 – Interface principale de Matlab

un objectif bien précis :

- **Le menu** : Regroupe des commandes de base de Matlab, comme enregistrer, afficher, les préférences, etc...
- **L'explorateur de fichiers** : Permet de visualiser ses fichiers scripts et de les ouvrir pour les éditer.
- **La zone de commande** : Permet d'écrire des commandes et de visualiser leur résultat.
- **La zone des variables** : Permet de visualiser toutes les variables en mémoire à l'instant présent avec leur nom ainsi que leur contenu.
- **L'historique** : permet de visualiser l'historique des commandes précédemment exécutées.

On peut écrire des commandes simples dans Matlab, cependant quand on veut écrire un programme complexe et complet, on uti-

lise l'éditeur de script Matlab.

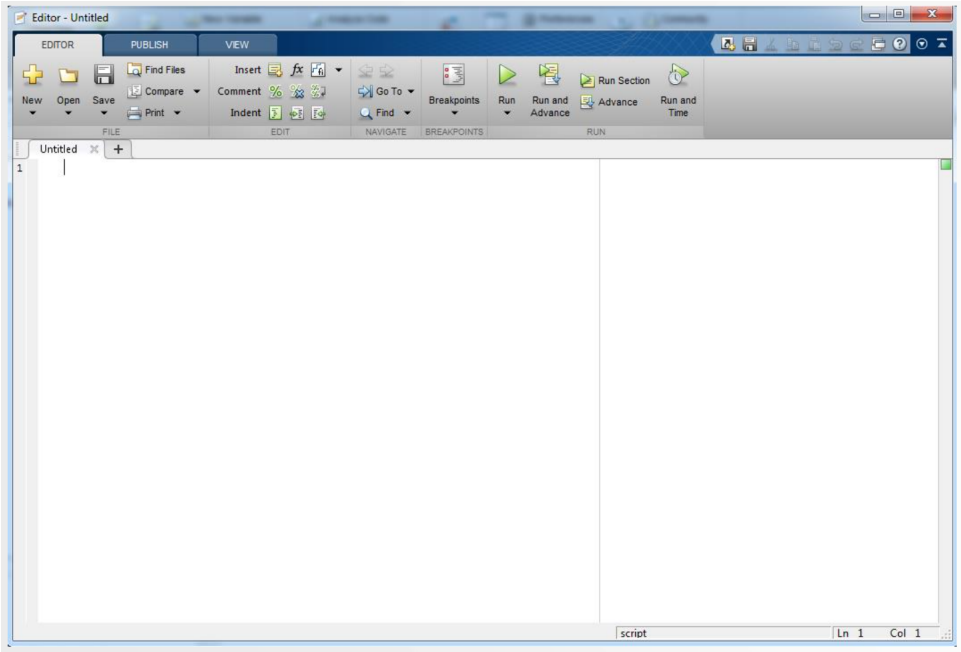

FIGURE A.3 – L'éditeur de script Matlab

A.2.2 Matrices dans Matlab

Le calcul matriciel est l'une des domaines où Matlab excelle en termes de performances.
On distingue plusieurs types de matrices dans Matlab :

— **Les Matrice classiques** : Pour déclarer une matrice dans une variable A, on énumère entre crochets ses éléments (séparés par des espaces), et on utilise un point virgule pour passer d'une ligne à l'autre.
» A = [1 2 3 ; 4 5 6 ;7 8.5 9 ; 10 11.5 12]
A est une matrice à 4 (quatre) lignes et 3 (trois) colonnes, que l'on peut vérifier dans la zone des variables. Si on double cliques sur la variable dans cette zone son contenu s'afficher.

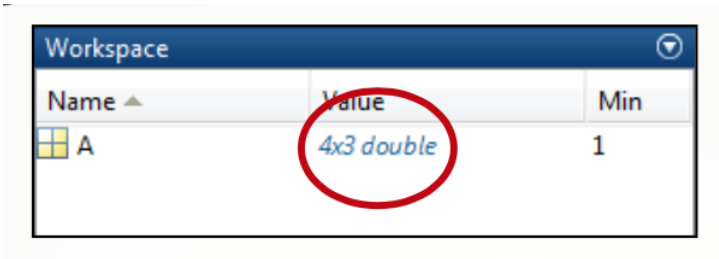

FIGURE A.4 – Editeur de script Matlab

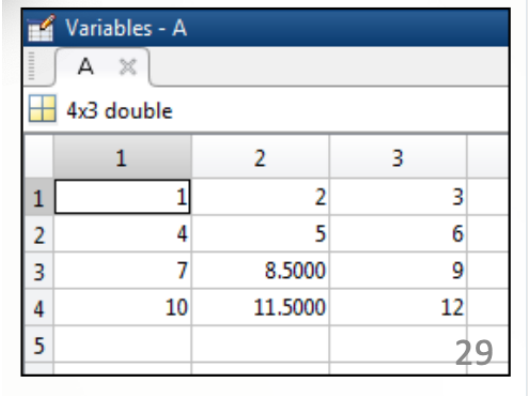

FIGURE A.5 – Contenu de la matrice A

— **Les vecteurs** : Sont des matrices avec une seule ligne ou une seule colonne. Il y'a des vecteurs lignes et dess vecteurs colonnes.

A.3 CPLEX

CPLEX (édité par la société ILOG) est un solveur de programmation linéaire, c'est-à-dire un logiciel permettant de résoudre des problèmes d'optimisation linéaire, entre autres. Plusieurs types d'algorithmes de résolution sont disponibles, en particulier l'algorithme du simplexe et un algorithme de points intérieurs.

Initialement, CPLEX est un solveur de programmes linéaires. A ce titre, il repose donc sur une implémentation performante du simplexe primal. Il dispose également du simplexe dual et du sim-

plexe de réseau. Il peut aussi résoudre des programmes linéaires mixtes, en combinant le simplexe, le branch and bound et la génération de coupes. Depuis peu, il intègre également une technique à base de points intérieurs et peu traiter des problèmes quadratiques. Actuellement, CPLEX est un des solveurs les plus performants disponibles, sinon le plus performant. Il peut ainsi traiter des problèmes contenant plusieurs dizaines de milliers de variables et plusieurs centaines de milliers de contraintes. Pour les problèmes mixtes, la limite est sensiblement plus basse, mais elle dépend grandement du type de problèmes et du modèle appliqué.

Il existe deux manières d'utiliser CPLEX. La première consiste à travailler de manière interactive en invoquant un interpréteur de commande dédié. La seconde consiste à appeler directement les fonctionnalités du moteur depuis son propre code, que ce soit du C, du C++ ou du Java.

Les problèmes traités par la suite d'optimisation ILOG sont : les programmes linéaires et linéaires mixtes, les programmes quadratiques et quadratiques mixtes, les programmes avec contraintes quadratiques et avec contraintes quadratiques mixtes.

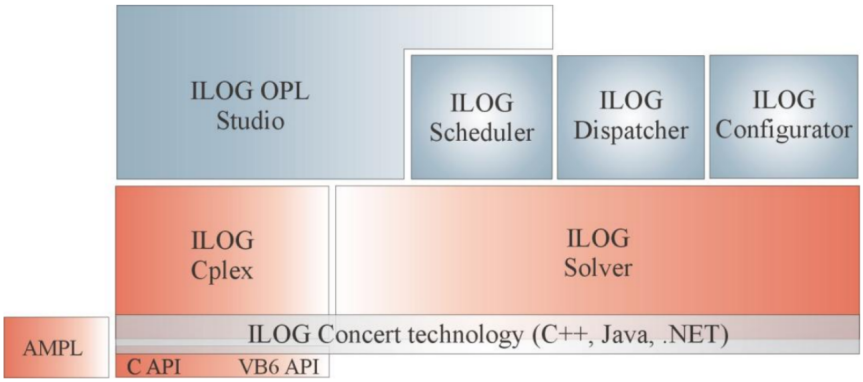

FIGURE A.6 – Suite d'optimisation ILOG

— **ILOG CPLEX** : Le cœur du système résout des problèmes de programmation mathématique.

— **ILOG Solver** : La partie principale du système, résout des applications en utilisant la programmation par contraintes.
— **ILOG concert technology** : Les bibliothèques contienant les fonctionnalités du système. Elles sont disponibles pour les langages C++, Java et .NET.
— **ILOG Scheduler** : Fournit des extensions pour résoudre des problèmes de planification.
— **ILOG Dispatcher** : Fournit des extensions pour la résolution de problèmes de tournées de véhicules.
— **ILOG Configurator** : Ce module contient des utilitaires pour l'optimisation des ventes en ligne (problèmes de E-Commerce).
— **ILOG OPL Studio** : OPL est un langage pour la modélisation des problèmes d'optimisation. Il interagit directement avec les modules ILOG Cplex, ILOG Solver et ILOG Dispacher.
— **AMPL** : C'est un autre langage pour la modélisation, qui interagit avec le module ILOG CPLEX. AMPL a été développé par les laboratoires Bell.
— **C et VB6 APIs** : Ce sont des bibliothèques pour des utilisateurs du langage C et de l'environnement VB6. Elles permettent de les interfacer avec le module ILOG CPLEX.

A.3.1 Optimization Programming Language (OPL)

Il s'agit d'un langage permettant de spécifier des problèmes d'optimisation, qui peuvent être ensuite passés à un solveur pour la résolution. Le fait qu'OPL soit un langage de modélisation implique qu'il n'a pas vocation à être exécuté comme du C, du Java ou du Python. En revanche, il permet d'écrire des problèmes d'optimisation dans une syntaxe abstraite que les solveurs peuvent exploiter.
L'interface utilisée pour écrire des problèmes d'optimisation en

OPL est l'IBM Ilog CPLEX Optimization Studio.

A.3.1.1 Structure d'un projet OPL

Un projet OPL est constitué de 4 (quatre) types de fichiers :

- **Un fichier projet (.projet)** : Ce fichier n'a pas réellement vacation à être ouvert dans un éditeur de texte. Il contient l'information permettant à CPLEX Studio d'afficher les différents autres fichiers contenus dans le project et les liens entre eux. C'est donc le fichier .project qui permet d'afficher la racine d'un projet dans CPLEX Studio.
- **Un ou plusieurs fichiers de modèles (.mod)** : Les fichiers modèles sont le cœur d'OPL. C'est dans un fichiers modèle qu'on indique les constantes, les variables de décision, les contraintes et la fonction objectif d'un problème d'optimisation.
- **Un ou plusieurs fichiers facultatifs de données (.dat)** : Lorsqu'on définit un problème d'optimisation dans un fichier modèle, il est souvent préférable de déclarer les constantes du problème mais de ne pas en préciser la valeur en utilisant le symbole "..." . Ce symbole indique que la valeur de ces constantes sera fournie dans un fichier de données.
- **Un ou plusieurs fichiers facultatifs de paramètres (.ops)** : Ces fichiers permettent de paramétrer le solveur d'optimisation (ou de contraintes). Ils permettent notamment de gérer la stratégie de parcours de la recherche arborescente en PLNE (Prgramme Linéaire en Nombre Entier), les heuristiques utilisées, les méthodes de construction de coupes, etc. ...On y accède directement via l'interface graphique de CPLEX.
- **Un ou plusieurs fichiers de configuration d'exécution (.oplproject)** : Ces fichiers n'ont pas comme vocation d'être ouverts en tant que fichier textes. On y accède directe-

ment via l'interface graphique CPLEX Studio. Ils indiquent précisément ce qu'il faut faire dans une exécution donnée d'un solveur. Il précise donc quel modèle utiliser, avec quel fichier de données et quel de paramètre.

A.3.1.2 Exemple d'éléments de symtaxe OPL

`;`	symbole de fin de déclaration
`//`	symbole de commentaire
`int n=5;`	`n` est un entier qui vaut 5
`float cost[1..n][1..n] = ...;`	`cost` est une matrice de taille `n×n` dont les éléments seront spécifiés dans un fichier .dat
`{string} names = ...;`	`names` est un ensemble discret de noms dont les éléments seront spécifiés dans un fichier .dat
`range I=1..n;`	`I` est un intervalle d'entiers de `1` à `n`
`dvar float+ weight;`	`weight` est une variable de décision continue positive
`dvar int ship[1..n];`	`ship` est un vecteur de `n` variables de décision entières
`maximize sum(i,j in I) w[i][j]*ship[i];`	spécifie une fonction objectif
`subject to {ship[1]<=8;}`	spécifie une liste de contraintes entre accolades
`forall(i in 2..n) ship[i]>=3;`	spécifie un ensemble de contraintes indicées par `i`

FIGURE A.7 – Quelques éléments de syntaxe

A.3.1.3 Exemple de création de projet

Tout d'abord, il faut créer un projet OPL dans lequel on pourra définir notre modèle. Pour cela dans Cplex Studio il faut cliquer sur Fichier / Nouveau / Projet OPL. Une fenêtre s'ouvre : entrez un nom de projet, choisissez son emplacement (dossier parent) et cochez "Création d'un modèle" ainsi que "ajouter une configuration d'exécution par défaut". Dans le fichier modèle (.mod), on saisit le modèle présenté dans la Figure ci-dessous.

A.3.1.4 Comment resoudre un modèle d'optimisation

Pour lancer la résolution, il faut faire un clic droit sur "Configuration d'exécution" dans l'onglet Projets OPL situé à gauche de la fenêtre principale puis "exécuter / configuration d'exécution

Minimiser $3x + 2y$
Sous
$x - y \geq 5$
$3x + 2y \geq 10$
	`dvar float x;`
	`dvar float y;`
	`minimize`
	`3*x + 2*y;`
	`subject to {`
	`x - y >= 5;`
	`3*x + 2*y >= 10;`
	`}`

formulation mathématique	*formulation dans le langage OPL*

FIGURE A.8 – Exemple d'un fichier modèle (.mod)

par défaut". Le bouton exécuter dans la barre d'outils permet de lancer une nouvelle fois la dernière configuration exécutée.

Une fois le modèle résolu, plusieurs informations s'affichent dans les onglets situés en bas de la fenêtre principale (sous le fichier modèle).

— L'onglet "solution" donne des informations sur la solution (optimalité, coût de la fonction objectif, valeur des variables à l'optimalité).
— L'onglet "journal du moteur" affiche la sortie de CPLEX,
— L'onglet "statistique" montre différentes mesures liées à la résolution (nombre d'itérations du simplexe, nombres de nœuds de branchement...).
— Tout en bas à droite de la fenêtre, s'affiche le chronomètre qui mesure le temps mis par CPLEX pour résoudre le modèle.

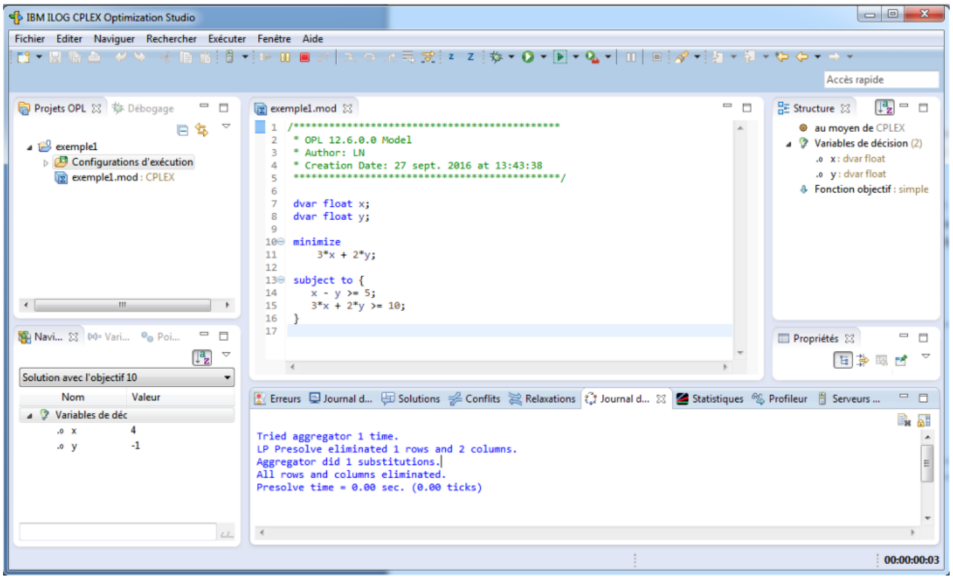

FIGURE A.9 – Interface de résolution

A.4 Coontiki OS

Contiki est un système d'exploitation multi-plateformes qui cible spécifiquement les petits appareils avec une mémoire, une puissance, une bande passante et une puissance de traitement limitées. Il utilise un design minimaliste tout en conservant les outils courants des systèmes d'exploitation modernes. Il fournit des fonctionnalités pour la gestion des programmes, processus, ressources, mémoire et communication.

A.4.1 Architcture de Contiki OS

L'architecture de contiki est basée sur le système modulaire, son noyau suit le modèle événementiel mais il fournit des fonctionnalités de threading optionnelles aux processus individuels.
Le noyau Contiki comprend un planificateur d'événements léger qui distribue les événements aux processus en cours d'exécution.

L'exécution du processus est déclenchée par les événements envoyés par le noyau aux processus ou par un mécanisme d'interrogation. Il existe Deux types d'événements qui sont : les événements asynchrones et les événements synchrones. La différence entre les deux est que les événements synchrones sont envoyés immédiatement au processus cible qui provoque sa planification. D'autre part, les événements asynchrones ressemblent plus à des appels de procédure différés qui sont mis en file d'attente et distribués plus tard au processus cible.
C'est grâce à ceux-ci que contiki peut faire une meilleure gestion des ressource physique telles que le processeur, la mémoire, les périphériques (d'entrée et sortie).

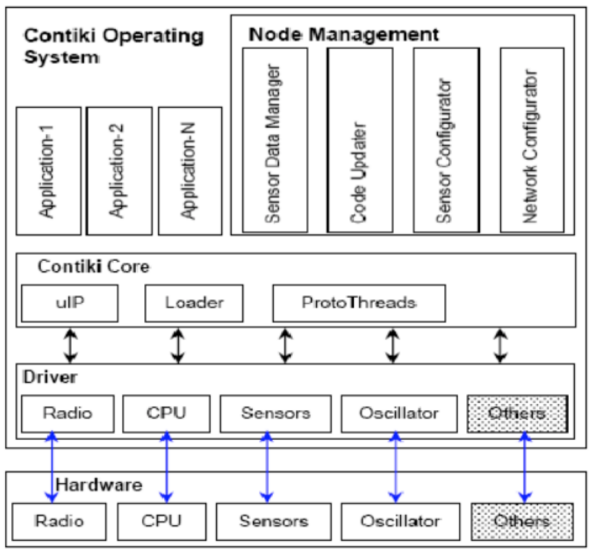

FIGURE A.10 – Architecture de contiki OS

A.4.2 Piles protocolaires dans conitki OS

Contiki offre deux types de connectivité :

— **La couche Rime** : Permet un dialogue avec les capteurs voisins pour établir des schémas de routage. Cette pile alternative fournit une solution lorsque IPv4 ou IPv6 s'avèrent prohibitifs. Il propose un ensemble de primitives pour les systèmes basse consommation.
— **La couche uIP** : Orientée Internet, offre les services essentiels du protocole IP mais nécessite plus de ressources que Rime.
— **uIPv6 (pour IPv6)** : Il s'agit d'une extension IPv6 entièrement conforme à uIP.
— **6LowPAN** : Cela signifie IPv6 sur les réseaux personnels sans fil à faible puissance. Il fournit une technologie de compression pour prendre en charge le faible débit sans fil requis par les appareils aux ressources limitées.
— **MQTT (Message Queuing Telemetry Transport)** : est un protocole de messagerie publish-subscribe basé sur le protocole TCP/IP. Il a été initialement développé par Andy Stanford-Clark (IBM) et Arlen Nipper (EuroTech). Il est conçu pour les connexions avec des sites distants où la bande passante du réseau est limitée.

A.4.3 Exemple de protocoles dans contiki OS

Contiki prend en charge les protocoles standard et les protocoles d'activation récents pour l'IoT,LLN :

— **RPL** : Ce protocole IPv6 proactif à vecteur de distance pour les LLNs (réseaux à faible puissance et à pertes) permet de trouver le meilleur chemin possible dans un réseau complexe de périphériques aux capacités variées.
— **MQTT (Message Queuing Telemetry Transport)** : est un protocole de messagerie publish-subscribe basé sur le protocole TCP/IP. Il a été initialement développé par Andy Stanford-Clark (IBM) et Arlen Nipper (EuroTech). Il

est conçu pour les connexions avec des sites distants où la bande passante du réseau est limitée.

— **CoAP** : Ce protocole prend en charge la communication pour les appareils simples, généralement les appareils nécessitant une surveillance à distance intensive.

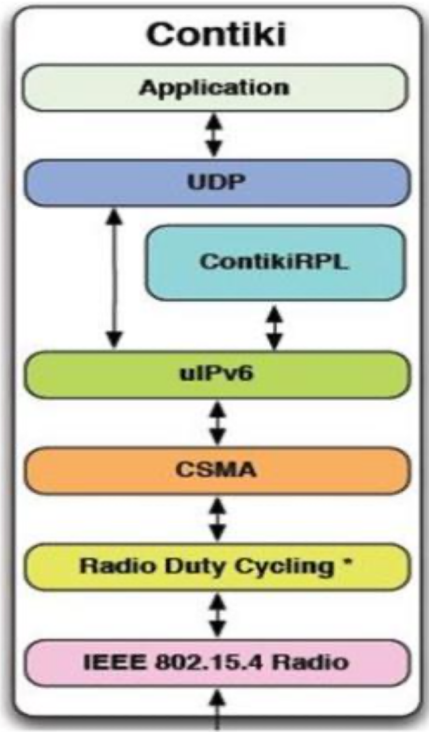

FIGURE A.11 – Pile protocolaire du Contiki OS

A.4.4 Principaux répertoires du Contiki

— **apps** Applications Contiki (utilisables dans d'autres applications) -> APPS += <application> dans Makefile

— **cpu** : Fichiers spécifiques par architecture (arm, avr, 6502, cc26xx-cc13cc, etc.)

— **dev** : Pilotes périphériques spéciaux (MMC, radio, capteurs divers)

— **examples** : Exemples Contiki (+++)
— **platform** : Fichiers spécifiques et pilotes (srf06-c26xx)
— **core** : Principales fonctions Contiki (IPv6/IPv4, processus, threads, timers, etc.)
— **tools** Outils (dont émulateur Cooja)

A.4.5 Le simulateur Cooja

Contiki propose un simulateur de réseau appelé Cooja. Ce simulateur permet l'émulation de différents capteurs sur lesquels seront chargés un système d'exploitation et des applications. Cooja permet ensuite de simuler les connexions réseaux et d'interagir avec les capteurs. Cet outil permet aux développeurs de tester les applications à moindre coût. Les capteurs supportés à ce jour par Cooja sont : exp5438, z1, wismote, micaz, sky, jcreate, sentilla-usb, esb.

A.4.5.1 Prise en main de cooja

Le cooja est un simulateur facile à manipuler, pour accéder à son interface certaines etapes sont nécessaires comme l'illustre les figures ci dessous. Tout d'abord lancer le contiki et ouvrir son terminal et suivre les étapes suivantes :

1. Dans le terminal taper la commande : **: '$cd contiki/tools/cooja**

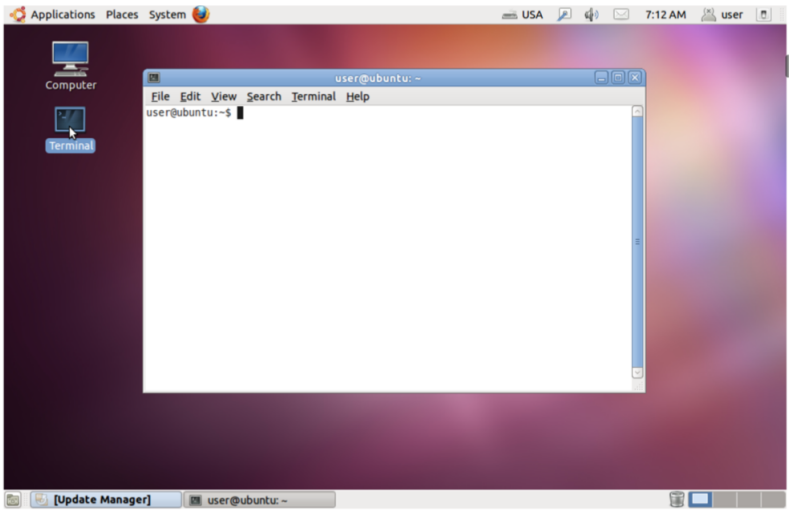

FIGURE A.12 – Interface contiki avec le termianl lancé

2. Une fois dans le repertoire cooja **:/contiki/tools/cooja$ ant run**

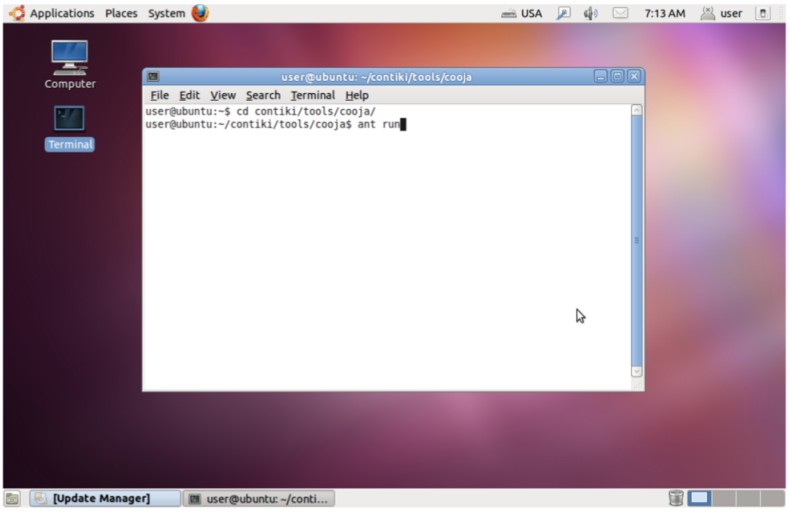

FIGURE A.13 – Lancement du cooja

3. **Interface du cooja**

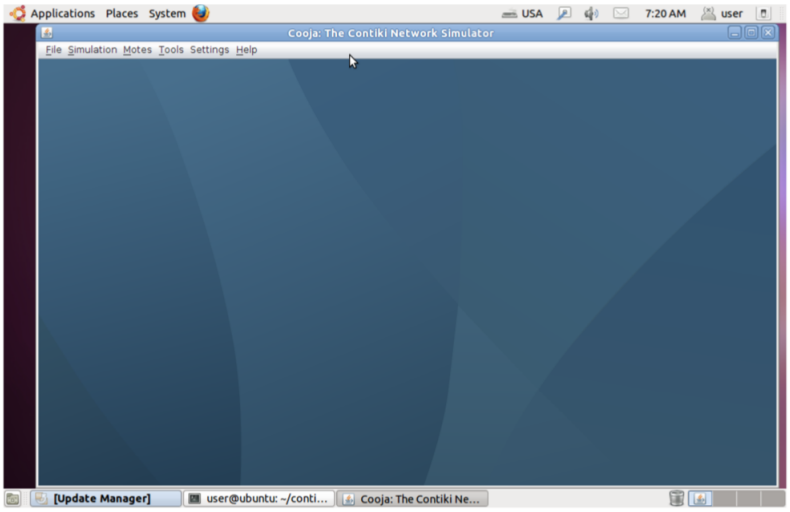

FIGURE A.14 – Interface du cooja

A.4.5.2 Nouvelle simulation

Pour lancer une nouvelle simulation **cliquer sur file -> new simulation**. Une nouvelle fénètre apparait (**create new simulation**) sur cette dernière nommé votre nouvelle simulation et cliquer sur le boutun **create**

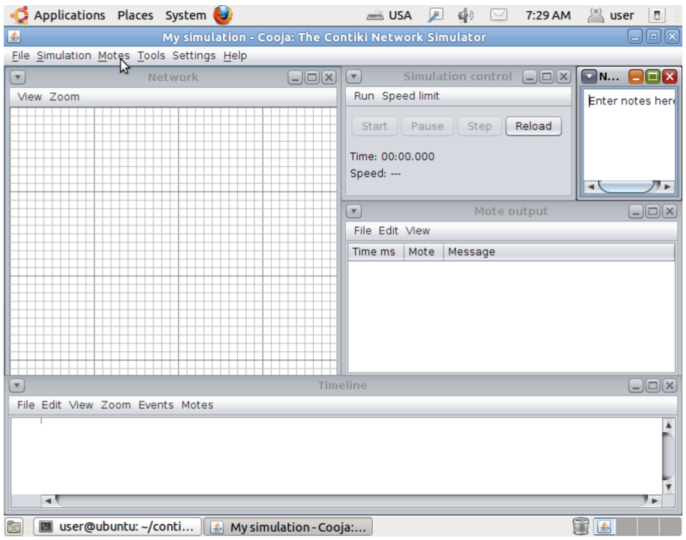

FIGURE A.15 – Interface de simulation

A.5 Configuration RPL et cooja pour notre projet

Procedure Contiki pour le respect des contraintes du modèle :

— Dans le fichier UDGM.java du répertoire **contiki/tools/cooja/se/sics/cooja/radiomediums** mettre la variable **TRANSMITTING_RANGE = R_COM.** Ainsi nous assurons le respect du rayon de communication défini dans le modèle.

— Utiliser le ficher de sortie du modèle comme fichier position.dat afin de permettre le placement des nœuds dans la bonne position par le plugin mobilité de cooja

— Démarrer cooja avec sudo ant *runbigmem* afin de permettre l'allocation d'une mémoire suffisante pour la simulation

— Dans le fichier Rpl-conf.h du répertoire **..contiki/core/net/rpl** mettre la variable **RPL_LEAF_ONLY à 1** pour interdire l'utilisation des nœuds sentinelles dans le processus de routage.

A.6 Contiki-NG (Contiki-New Genration)

Nous tenons à notifier la disponibilité de la derniére version de contiki appelé Contiki-NG qui a été spécialement conçu pour IoT (avec un API spécial cloud computing).

Bibliographie

[1] Tobias Achterberg. Scip : solving constraint integer programs. mathematical programming computation. *Dunod Paris*, 2009.

[2] Kamarulzaman Ab. Aziz Azlina Ab. Aziz and Wan Zakiah Wan Ismail. Coverage strategies for wireless sensor network. *World Academy of Science, Engineering and Technology 50*, 2009.

[3] Vincent Barichard. Approches hybrides pour les problèmes multi objectifs, thèse doctorat, École doctorale dangers. 2003.

[4] Mohamed Amin Benatia. *Optimisation multi-objectives d'une infrastructure réseau dédiée aux bâtiments intelligents*. PhD thesis, Université de Normandie.

[5] E.S. Biagioni and G. Sasaki. Wireless sensor placement for reliable and efficient data collection. *Proceeding of the 36 Hawaii International Conference on System Sciences*, 2003.

[6] Mehdi BOUALLEGUE. *Protocoles de communication et optimisation de l'énergie dans les réseaux de capteurs sans fil*. PhD thesis, Université Bretagne Loire, 2006.

[7] Meng Zheng Chaofan Ma, Wei Liang. Delay constrained relay node placement in two-tiered wireless sensor networks : A set-

covering-based algorithm optimal placement of relay nodes over limited positions in wireless sensor network. *Journal of Network and Computer Applications*.

[8] Yunxia Chen. On the lifetime of wireless sensor networks. *IEEE*, Novembre 2005.

[9] Heinzelman W. B. Chen L. A survey of routing protocols that support qos in mobile ad hoc networks. *IEEE Network*, 2007.

[10] J. Bicket et R. Morris D. De Couto, D. Aguayo. High-throughput path metric for multi-hop wireless routing. *In ACM MOBICOM*, 2003.

[11] Inès Doghri. *Stratégies de routage multi-chemin dans les réseaux sans fil multi-sauts*. PhD thesis, Université de Lion1.

[12] Ines Khoufi et al. Survey of deployment algorithms in wireless sensor networks : Coverage and connectivity issues and challenges. *In : Internatio- nal Journal of Autonomous and Adaptive Communications Systems (IJAACS)*, 2017.

[13] D. S. Deif et Y. Gadallah. Classification of wireless sensor networks deployment techniques. *IEEE Communications Surveys Tutorials 16.2 (2014)*, 2013.

[14] El ghazali Talbi. Métaheuristiques pour l'optimisation combinatioire multi-objectif : Etat de l'art.

[15] Fred Glover. Future paths for integer programming and links to artificial intelligence. 1986.

[16] Omessaad HaJJI. *Contribution au développement de méthodes d'optimisation stochastiques. Application à la conception des dispositifs électrotechniques*. PhD thesis, Ecole Centrale de Lille, 2003.

[17] Tall HAMADOUN. *Réseau maillé sans fil à basse consommation énergétique*. PhD thesis, Université de Lion1.

[18] Mataric M.J Howard, A. and Sukhatme. Mobile sensor network deployment using potential fields : A distributed, scalable solution to the area coverage problem. *Proceeding of the 6th International Symposium on Distributed Autonomous Robotics Syatems Fukuoka*, 2002.

[19] G. Laporte I.H. Osman. Metaheuristics : A bibliography, annals of operations research, 63 : 511-623, 1996.

[20] Philippe Galinier Jin-Kao Hao and Michel Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence artificielle*, 1999.

[21] Kashif Saleem José V. V. Sobral, Joel J. P. C. Rodrigues Ricardo A. L. Rabêlo and † Vasco Furtado. An enhanced routing protocol for internet of things applications over low power networks. 2019.

[22] S.Hamma B.Parrein J.Yi, E.Cizeron. Simulation and performance analysis of mp-olsr for mobile ad-hoc networks. *IEEE Wireless Communications and Networking Conference (WCNC 2008)*, 2008.

[23] Mathieu Liedloff. *Algorithmes exacts et exponentiels pour les problèmes NP-difficiles : domination, variantes et généralisations.* PhD thesis, Université Paul Verlaine – Metz.

[24] A. Hertz et D. Costa M. Widmer. Les métaheuritiques apparu dans « ordonnancement de la production. *Edité par Pierre lopez et François Roubellat.*

[25] Bernard Tourancheau Malisa Vucinic and Andrzej Duda. Performance comparison of the rpl and loadng routing protocols in a home automation scenario. *University of Grenoble, CNRS Grenoble Informatics Laboratory UMR 5217 LIG, Grenoble, France*, 2014.

[26] Michel Minoux. Programmation mathématique : théeorie et algorithmes. *Dunod Paris*, 1983.

[27] Paul MÜHLETHALER. Routage dans les réseaux ad hoc. 2004.

[28] BENDIMERAD Nawel. *DYMO Multi-chemins à nœuds disjoints sans interférences pour les réseaux ZigBee/Standard IEEE 802.15.4*. PhD thesis, Université d'Oran.

[29] Diery NGOM. *Optimisation de la durée de vie dans les réseaux de capteurs sans fil sous contraintes de couverture et de connectivité*. PhD thesis, Université de Haute Alsace (France) et de Université Cheikh Anta Diop de Dakar (Sénégal), 2016.

[30] Sang Yoon Park Nguyen Xuan Tien, Jong Myung Rhee. A novel effective multipath routing technique providing high availability in wireless networks. *Department of Information and Communications Engineering, Myongji University, 116 Myongji-ro, Yongin, Gyeonggi 17058, Korea*, 22 March 2018.

[31] A. Etorban P.J.B. King and I.S. Ibrahim. A dsdv-based multipath routing protocol for mobile ad-hoc networks. *In in The 8th Annual PostGraduate Sym- posium on The Convergence of Telecommunications, Networking and Broadcasting*, 2007.

[32] J. Padhye R. Draves and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. *in proceedings of the ACM MobiCom*, Septembre 2004.

[33] Tanguy ROPITAULT. Protocole de routage rpl. *Technologies de l'information | Réseaux Télécommunications*, 2016.

[34] P. Tsang S Mueller and D Ghosal. Multipath routing in mobile ad hoc networks : Issues and challenges. *Performance Tools and Applications to Networked Systems*, pages 209–234, 2004.

[35] BENINE Safa. *le routage multi-chemin dans les réseaux de capteurs sans fil.* PhD thesis, Université Echahide Hamma Lakhdar El-oued.

[36] Kumar N. Zhu C. Ahmad R.W Sobral J.V., Rodrigues J.J. Performance evaluation of routing metrics in the loadng routing protocol. *University of Grenoble, CNRS Grenoble Informatics Laboratory UMR 5217 LIG, Grenoble, France*, 2017.

[37] FELLAH Soumaya. *Optimisation Multi-objectif appliquée au déploiement et à la performance des réseaux de capteurs sans fil.* PhD thesis, Université d'Oran1-Ahmed Ben Bella.

[38] WINTER (T.). Rpl : Ipv6 routing protocol for low-power and lossy networks. *rfc6550*, 2016.

[39] M. Yildizoglu Vallée. Présentation des algorithmes génétiques et de leurs applications. 4.2, 2003.

[40] Takahara G. Xu, K. and H. Hassanein. On the robustness of grid- based deployment in wireless sensor networks. *2nd IEEE international Conference on Broadband Networks, Boston, Octobre 2005.*, IWCMC 06(:1183- 1188), 2006.

[41] S. K. Tripathi Z. Ye, S. V. Krishnamurthy. A framework for reliable routing in mobile ad hoc networks. *IEEE INFOCOM*, 2003.