Université du Québec
Institut National de la Recherche Scientifique
Centre Energie Matériaux Télécommunications

Conception et Implémentation Matérielle et Évaluation en Conditions Réelles de Techniques Avancées de Localisation pour les Réseaux de Capteurs Sans Fil

Par

Hussein Olayan

Mémoire présenté pour l'obtention du grade de
Maître es Sciences, M.Sc.
en Télécommunications

Jury d'évaluation

Président de jury et évaluateur interne	Prof. Serge Tatu INRS
Évaluateur externe	Prof. Paul Fortier Université Laval
Directeur	Prof. Sofiène Affes INRS
Co-directeur	Prof. Nahi Kandil INRS
Co-directeur	Dr. Ahmad El-Assaf IBM

Remerciements

Ce travail a été développé et réalisé au sein du Wireless Lab du Centre Energie, Matériaux et Télécommunications (EMT) de l'Institut National de la Recherche Scientifique (INRS).

Tout d'abord, je tiens à remercier les membres du jury qui ont accepté d'évaluer ce mémoire. Je leur suis extrêmement reconnaissant pour le temps et les efforts qu'ils ont consacrés à cette tâche.

Je voudrais également exprimer ma profonde gratitude à mon superviseur, le professeur Sofiène Affes, et à mes co-superviseurs, le professeur Nahi Kandil et DR. Ahmad El Assaf, pour leur aide, leur mentorat et leur motivation à m'aider dans ce projet. Merci également à tous les membres du l'équipe du Wireless Lab pour leur soutien et à toutes les personnes qui m'ont aidé tout au long de ma maîtrise.

Résumé

Les réseaux de capteurs sans fil (WSNs) ont toujours été un domaine d'intérêt pour la communauté de recherche. L'utilité principale des WSNs est leur capacité à collecter des données physiques à partir d'environnements spécifiques tels que la température, l'humidité et la lumière, entre autres. Par conséquent, les WSNs ont toujours gagné leur place dans la pratique dans de nombreuses applications impliquant des processus de surveillance, la domotique et l'automatisation des villes.

Les WSNs sont particulièrement conçus pour les communications embarquées de faible puissance. Ces capteurs ont une capacité de calcul très limitée, sont alimentés par de petites batteries ou un composant de récupération d'énergie, et communiquent par des liaisons sans fil. L'un des inconvénients populaires des WSNs est qu'ils ne sont pas capables de reconnaître leur propre position contrairement aux capteurs de positionnement global (GPS) qui sont capables de reconnaître leur position en communiquant par satellite.

Dans ce contexte, la localisation dans les WSNs est un sujet critique et en constante évolution. Dans ce mémoire, nous implémentons d'abord les meilleurs algorithmes de localisation connus dans la littérature (ex : DV-Hop et LEAP) sur la plate-forme Contiki Cooja. De plus, nous proposons une nouvelle solution selon laquelle le nœud régulier (c.-à-d. les positions inconnues) estime directement sa distance à toutes les ancres (c.-à-d. les positions connues - p. ex., équipées d'un GPS) du réseau en se fiant uniquement aux renseignements disponibles localement.

Les résultats de simulation montrent que la technique proposée permet d'obtenir de meilleures performances et moins de coût de communication que les meilleures solutions de localisation représentatives actuellement disponibles dans la littérature. Des essais expérimentaux avec des nœuds de capteurs MICAz confirment sa nette supériorité dans des conditions de fonctionnement réelles.

Mots-clés Réseaux de capteurs sans fil (WSNs), localisation, nombre de sauts, MICAz mote, système d'exploitation Contiki.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Le réseau de capteurs sans fil WSN se définit comme un réseau auto-configuré pour surveiller l'environnement physique tel que la température, le son, les vibrations, la pression, le mouvement. Le WSN peut transmettre ses données via le réseau à l'emplacement principal comme une station de base où les données peuvent être observées et analysées. Une station de base sert d'interface entre le réseau et les utilisateurs. Sachant que chaque réseau de capteurs sans fil contient des centaines de milliers de nœuds de capteurs. Les nœuds capteurs communiquent entre eux à l'aide de signaux radio. Alors que chaque nœud de capteur a sa vitesse de traitement, sa capacité de stockage et sa bande passante de communication.

Dans le contexte, nous présentons les WSN en général et leurs principales fonctionnalités et applications, le WSN en utilise plusieurs comme la surveillance de l'environnement, les services militaires, la domotique et d'autres utilisations larges et efficaces. Notre contribution au premier chapitre représente en expliquant l'importance d'utiliser WSN au lieu d'utiliser le système de positionnement global «GPS». Cela nous motive à travailler sur le sujet de thèse.

Pour éviter l'utilisation d'un GPS qui présente des tas d'inconvénients tels que l'ajout de par-dessus au réseau, le plus difficile de remplacer un GPS au cas où le GPS serait situé dans une zone intérieure comme une forêt. Actuellement, le WSN est le substitut numéro un au lieu du GPS, le WSN offre de multiples avantages en considérant son faible coût et en l'utilisant dans de nombreux types d'environnements. Alors que l'inconvénient du WSN apparaît en manquant la capacité de communiquer avec les satellites, cela ouvre la porte aux chercheurs pour qu'ils adoptent un algorithme de localisation à utiliser par le WSN. Le chapitre deux montre les types d'algorithmes de localisation (ex: sans plage et basés sur la plage) qui ont été mis en œuvre sur le WSN. Nous avons exposé chaque algorithme et nous sommes apparus tous les inconvénients et avantages de chaque

algorithme et les exigences. Toutes ces nécessités nous ont inspiré pour créer notre algorithme de localisation proposé pour WSN.

Pour rendre notre algorithme de localisation proposé et les autres algorithmes de localisation capables de fonctionner dans le monde réel, cela nous obligeait à travailler dans le programme Contiki Cooja tandis que tous les algorithmes sont implémentés par le langage Contiki Cooja après avoir implémenté tous les algorithmes sur MATLAB. Notez que le langage du programme Contiki Cooja est C et $C++$. Le chapitre trois représente le travail de Contiki Cooja qui consiste à implémenter notre algorithme proposé et d'autres algorithmes de localisation par le langage de programmation Contiki Cooja. tandis que notre algorithme proposé fera un bon résultat en donnant une estimation de position précise au nœud en utilisant le programme Contiki Cooja.

Après avoir implémenté l'algorithme de localisation sur le programme Contiki Cooja, nous implémentons ces algorithmes sur des appareils réels (ex: MICAz). Le chapitre quatre montre une explication claire des périphériques matériels utilisés et un moyen d'exécuter un code Contiki Cooja sur les périphériques matériels. Et notre projet Java initial donne au chapitre 4 qui sera l'interface GUI Java qui utilise une implémentation sur le monde réel pour collecter les données du réseau et les enregistrer dans un fichier de base de données. Notre façon de déboguer et d'observer dans les algorithmes du monde réel donne déjà dans le chapitre 4 en utilisant la technique smart easy.

Le résultat de l'implémentation matérielle donne des performances élevées à notre algorithme proposé par rapport aux autres algorithmes de localisation, sur la base à la fois de la précision de détection de l'emplacement du nœud et du temps le plus rapide pour chaque nœud pour calculer son emplacement. Alors que le chapitre cinq donne des résultats de simulation sur MATLAB pour différents algorithmes de localisation. Les algorithmes proposés sur une grande collaboration montrent une grande efficacité pour les autres algorithmes de localisation. Cela nous permet de dire que notre algorithme proposé devient numéro un sur les autres algorithmes, alors qu'il montre une haute performance sur de nombreux aspects sur le programme Contiki Cooja et sur des appareils du monde réel comme le programme MICAz et MATLAB.

Chapitre 1

La motivation pour travailler sur le sujet du réseau de capteurs sans fil

1.1 Infrastructure des WSNs

Un réseau de capteurs sans fil est une combinaison d'appareils électroniques qui communiquent entre eux sur un support sans fil. Ces appareils, comme les motes MICAz [12], sont capables d'effectuer différentes tâches, par exemple, la surveillance des phénomènes naturels, la localisation pour l'emplacement du capteur. Dans une application WSN, les capteurs produisent de l'information en détectant l'environnement, comme la température, l'humidité et les vibrations. Ensuite, ces données produites sont transmises à une station de base, soit directement, soit par le biais d'une approche à sauts multiples. La station de base agit comme une passerelle vers une infrastructure principale du WSN et vers d'autres réseaux tels que l'Internet, comme le montre la figure 1.1. L'infrastructure de base fournit des nœuds formant un réseau qui se compose généralement d'unités de bases de données pour stockage, d'une interface utilisateur pour gérer le réseau et des outils d'analyse et de visualisation des données.

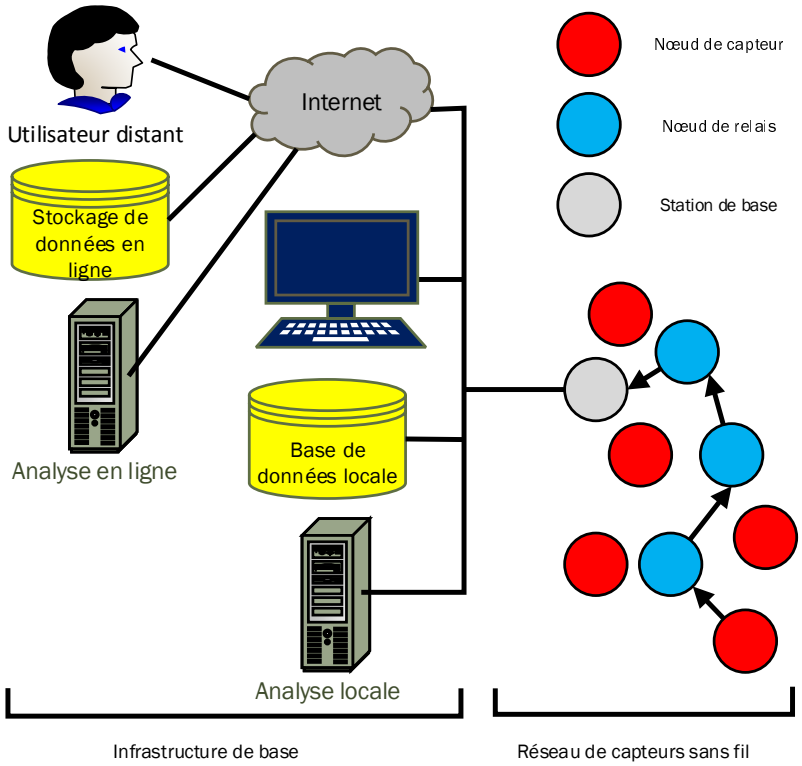

Figure 1.1 – Cadre général des WSNs.

1.2 Plateforme et nœuds des WSNs

Les applications WSN sont dérivées de plusieurs exigences [5]. Par exemple, une application de surveillance nécessite des mesures périodiques à faible consommation d'énergie, de ressources radio et d'autres ressources. D'autre part, une application de suivi d'objets nécessite la mesure en temps réel de capteurs de mouvement à haute énergie, ressources radio et mémoire. Par conséquent, les nœuds agissent dans le réseau sur la base des exigences pratiques de l'application WSN. Un WSN est constitué de quatre unités de traitement de base. Les composantes sont données à la figure 1.2 et sont décrites comme suit:

1. Une **unité de détection** qui surveille l'environnement et mesure les phénomènes physiques, par exemple, la température et les vibrations, etc.
2. Des **unités de calcul** qui se composeent de divers composants électroniques tels qu'un micro-contrôleur, un bloc de stockage de données, tel que la mémoire morte (ROM pour "Read Only Memory"), la mémoire vive (RAM pour "Random Access Memory"), ou une mémoire flash. De

plus, cette unité comprend également des ports d'entrée/sortie, un convertisseur numérique, des minuteurs et d'autres périphériques de support.

3. Une **unité radio** qui permet la transmission de données via des émetteurs-récepteurs sans fil et une(des) antenne(s).
4. Une **unité d'alimentation** qui fournit la tension d'alimentation du système à tous les autres composants électroniques du nœud. L'unité d'alimentation peut prendre plusieurs sources d'énergie telles que des batteries, ou des sources d'énergie câblées.

Un nœud du WSN exécute le logiciel qui l'implémente sur le réseau comme le montre la figure 1.1. Un nœud peut utiliser un système d'exploitation (OS pour "Operating System") pour contrôler les composants matériels au moyen d'un ensemble de routines de code appelé couche d'abstraction matérielle (HAL pour "Hardware Abstraction Layer") [26]. Le HAL améliore l'accès d'un développeur d'application au matériel sous-jacent via une interface de programmation. De plus, l'OS implémente le protocole réseau qui permet aux nœuds de communiquer entre eux. Les OSs des WSNs sont conçus pour s'adapter à une mémoire limitée de nœuds et gérer efficacement de ressources rares [21].

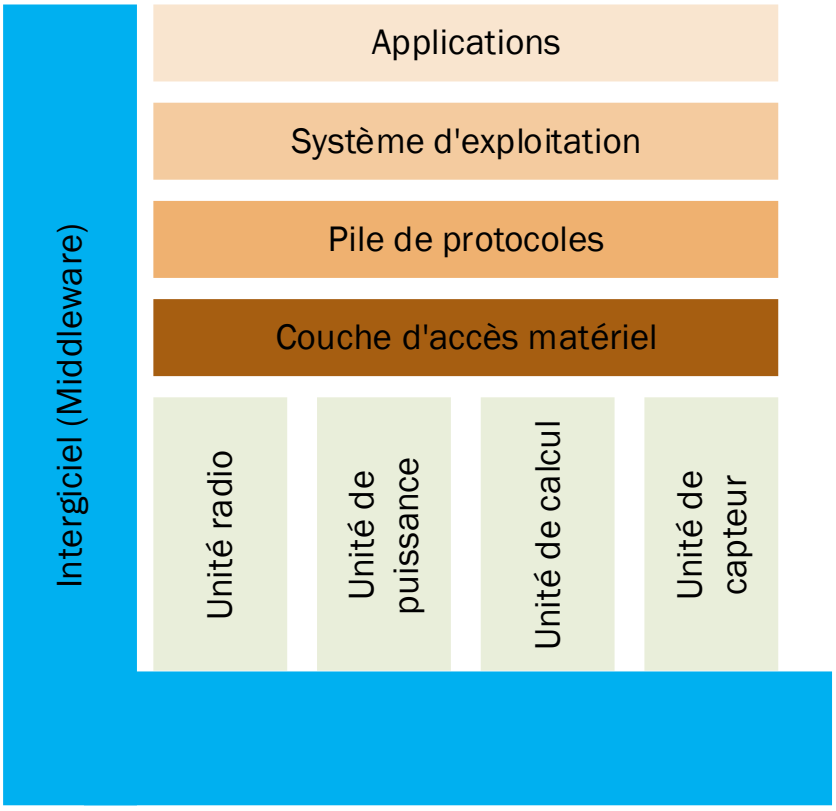

Figure 1.2 – Composantes d'un WSN.

1.3 Pile de protocoles et intergiciels

Les protocoles des réseaux WSNs sont présentés dans une architecture en couches appelée pile de protocoles [14], comme suit:

1. **Couche physique**: gère le matériel de communication radio. Par exemple, une séquence de bits qui est acheminée sur une liaison radio entre les nœuds. Les principales fonctions de l'environnement physique comprennent : réglage de la puissance d'émission, sélection des fréquences du canal, modulation et démodulation. De plus, la couche physique peut mesurer l'indicateur d'intensité du signal reçu (RSSI pour "Received Signal Strength Indicator"), effectuer un contrôle cyclique de redondance pour permettre de vérifier les erreurs de transmission, et aussi crypter et décrypter les messages pour la sécurité réseau.
2. **Couche MAC** (MAC pour "Medium Access Control"): la responsabilité de la couche MAC se concentre sur les fonctions d'accès au réseau, comme l'accès aux canaux radio, le contrôle de file d'attente des trames, l'assemblage des trames et le contrôle des erreurs de transmission. En outre, la couche MAC peut exécuter d'autres fonctions de gestion, telles que l'analyse du réseau, l'estimation du cycle de sommeil et des distances. Cependant, la couche MAC considère le lien entre les nœuds après découverte des voisins et échange ensuite des trames avec les nœuds voisins.
3. **Couche réseau**: la couche réseau est comme un guide pour un nœud pour décider vers quel nœud il doit envoyer le message. Ses principales fonctions incluent : la découverte d'itinéraires, la construction d'itinéraires, l'entretien des itinéraires, l'approche à sauts multiples, la sélection du saut suivant, le contrôle de la file d'attente des paquets et le transfert des données, parmi d'autres tâches. Les versions 4 et 6 du protocole Internet (IPv4 et IPv6 respectivement) sont un exemple de protocoles de couche réseau.
4. **Couche transport**: la couche transport améliore les fonctions qui permettent une livraison fiable de bout en bout des messages et le contrôle de la congestion dans les réseaux sans fil. Les protocoles de couche typiques comme le protocole TCP (pour "Transmission Control Protocol") et UDP (pour "User Datagram Protocol") sont trop complexes pour les WSNs et sont utilisés après modification. Par exemple, le système d'exploitation Contiki Cooja OS utilise un protocole appelé "simple udp" pour la transmission des messages afin de simplifier la version originale d'UDP.

5. **Couche application**: la couche application fournit les programmes d'application, qui contiennent généralement les algorithmes et les procédures pour effectuer des tâches spécifiques. Ces tâches sont prises en compte par les exigences de l'application et sont généralement l'interaction demande-réponse, la collecte de données, l'agrégation de données ou la fusion de données. En outre, ces programmes d'application ont besoin d'autorisations, par exemple à quel moment ou dans quelles conditions les données doivent être collectées auprès des capteurs, ou à quel moment les données doivent être transmises.
6. **Intergiciels** : les intergiciels (Middleware) forment l'ensemble de tous les services de la pile de protocoles, du système d'exploitation OS, HAL et permettent de les afficher à la couche application pour accéder directement aux ressources du noeud.

Chapitre 2

Algorithmes de localisation

2.1 Introduction à la localisation dans Les WSNs

Dans un WSN, les nœuds de capteurs sont déployés dans un environnement réel afin de collecter des données de l'environnement physique avoisinant. Une fois collectées, ces informations sont transmises des nœuds de capteurs à la station de base ou à une voie d'accès pour afficher ces informations. De plus, l'emplacement des nœuds est très important pour connaître leur position géographique, car toute information est inutile si les nœuds n'ont aucune idée de leur position géographique. Notez que si un nœud estime mal son emplacement, alors cette erreur d'estimation se propage globalement à travers le réseau et d'autres nœuds, entraînant ainsi la propagation d'informations erronées sur l'emplacement d'autres nœuds. Pour déterminer la position des nœuds, les capteurs s'appuient principalement sur la distance entre les nœuds conscients des coordonnées (position connue) et les nœuds non conscients des coordonnées (position inconnue). Comme solution typique, le GPS (pour "Global Positioning System") est le moyen le plus simple pour localiser les nœuds, ce qui devient très coûteux si un grand nombre de nœuds existent dans un réseau. Plusieurs chercheurs ont été attirés par ce sujet de la localisation à faible coût. Jusqu'à présent, de nombreux algorithmes ont été proposés dans la littérature [10] [9]. Ces algorithmes de localisation peuvent être classés en deux grandes catégories, comme le montre la figure 2.1.

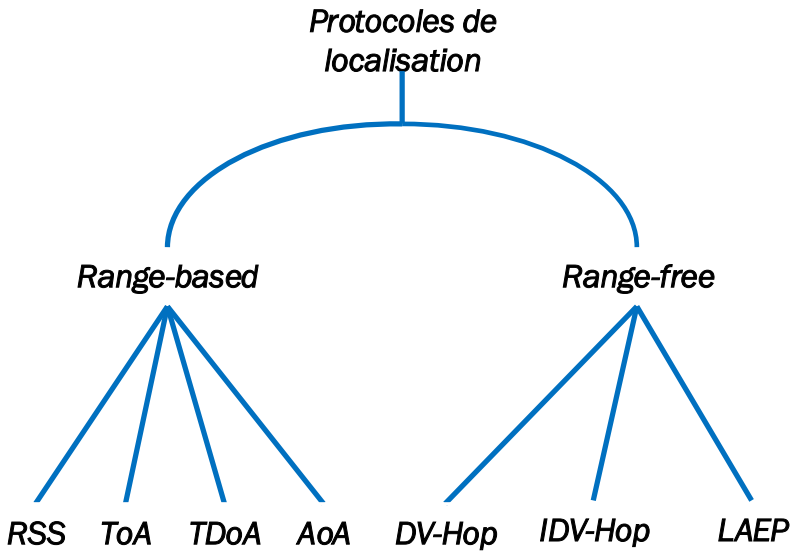

Figure 2.1 – Catégories de protocoles de localisation.

2.2 Concepts généraux sur le "range-based" et le "range-free"

Les algorithmes range-based [6] (ex: Angle d'arrivée (AoA), Force du signal reçu (RSS)) s'appuient sur le concept de distance physique pour estimer l'emplacement du nœud comme mesurer le temps ou la puissance entre les nœuds pour calculer l'emplacement du nœud . Contrairement aux algorithmes range-free [22] (ex: DV-Hop citep dvhop, LEAP citep LEAP) s'appuient sur des informations de connectivité entre les nœuds. Les informations de connectivité désignent les informations du réseau local telles que les coordonnées des ancres et le nombre de sauts (ex: nombre de sauts) pour chaque ancre. Des algorithmes range-based sont utilisés pour la trilatération afin de déterminer l'emplacement d'un point en utilisant la géométrie des triangles. Les autres algorithmes sont utilisés des angles et des distances pour trouver l'emplacement d'un point. Furthur, la trilatération utilise des distances entre au moins deux points de référence.

2.3 Algorithmes basés sur les distances "Range-based"

Les techniques de localisation "range-based" nécessitent l'existence d'ancres qui ont des informations de localisation précises et connues des nœuds réguliers. En utilisant plusieurs techniques de télémétrie, les nœuds réguliers restants du réseau estiment leur distance à trois ancres ou plus. Sur

la base de ces informations, l'emplacement d'un nœud est estimé. Par conséquent, le principe des techniques de télémétrie repose sur l'estimation de la distance physique entre les nœuds des capteurs. Cette distance est obtenue par la mesure d'une certaine caractéristique des signaux échangés entre les capteurs. Ces caractéristiques comprennent l'heure d'arrivée, la force du signal et l'angle d'arrivée qui peuvent être évalués pour aider à estimer la distance entre les nœuds.

2.3.1 Temps d'arrivée (TOA pour "Time of Arrival")

L'heure d'arrivée (TOA) [3] repose sur des mesures précises de la transmission et des signaux horaires reçus entre deux nœuds. Ces mesures servent à estimer la distance en fonction du temps de propagation et la vitesse du signal en fonction du type de mesure. Selon cette mesure, trois types d'acquisition TOA peuvent être effectués:

Unidirectionnel

L'approche unidirectionnelle mesure le temps de propagation, qui apparaît dans la différence entre le temps de transmission du signal et le temps d'arrivée du signal, comme le montre la figure 2.2. Dans cette méthode, le nœud récepteur calcule sa propre position. Cependant, il nécessite une synchronisation très précise des horloges de l'émetteur et du récepteur.

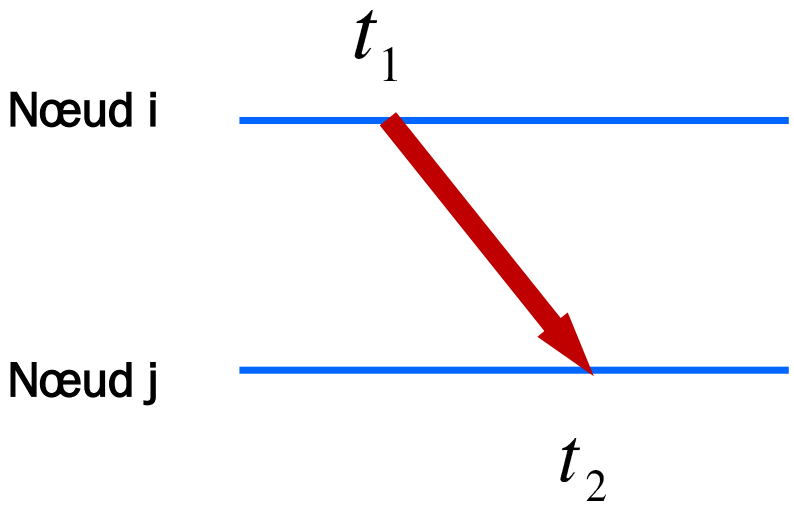

Figure 2.2 – TOA unidirectionnel.

Bidirectionnel

L'heure d'arrivée bidirectionnelle est préférée, où le temps aller-retour d'un signal est mesuré au niveau du dispositif émetteur. Comme le montre la figure 2.3, le nœud émetteur est alors capable de calculer l'emplacement du récepteur.

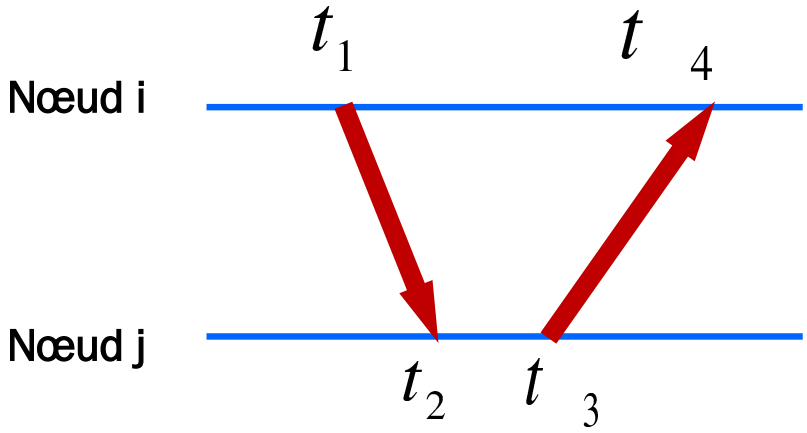

Figure 2.3 – TOA bidirectionnel.

2.3.2 Décalage du temps d'arrivée (TDOA pour "Time Difference of Arrival")

Le décalage de temps d'arrivée (TDOA) utilise deux signaux qui voyagent à des vitesses différentes, comme le montre la figure 2.4. Cette approche permet au récepteur de déterminer son emplacement et peut fournir des mesures très précises sans exiger que les horloges de l'émetteur et du récepteur soient synchronisées. De plus, l'approche TDOA nécessite des ressources matérielles supplémentaires.

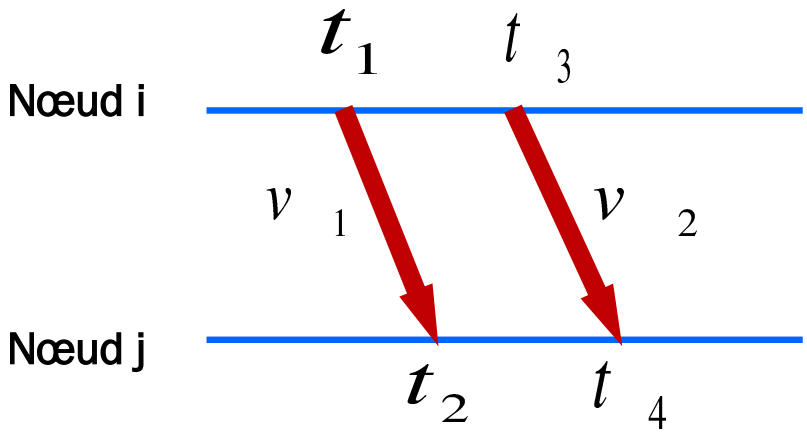

Figure 2.4 – TDOA.

2.3.3 Angle d'arrivée (AOA pour "Angle of Arrival")

Le concept d'angle d'arrivée sert à déterminer la direction de propagation du signal, généralement à l'aide d'un réseau d'antennes ou de microphones. L'angle d'arrivée est défini par l'angle entre la direction de propagation et une direction de référence connue comme une orientation. L'AOA fournit une localisation à précision élevée en fonction de la précision de la mesure. Cependant, le matériel de mesure d'angle d'arrivée peut augmenter considérablement la taille et le coût des réseaux d'antennes plus complexes.

2.3.4 Intensité du signal reçu (RSS pour "Received Signal Strength")

La force du signal reçu (RSS) [25] considère la force d'un signal reçu mesurée à l'antenne du récepteur. RSS détermine par la puissance de transmission, la distance entre l'émetteur et le récepteur, et l'environnement radio. La force du signal reçu du nœud i considéré comme émetteur au nœud j considéré comme récepteur au temps t est représentée comme suit:

$$P_R^{ij}(t) = P_T^i - 10\eta \log\left(d_{ij}\right) + X_{ij}(t), \tag{2.1}$$

où P_T^i est une constante due à la puissance d'émission et aux gains d'antenne des nœuds des capteurs, η est la constante d'atténuation, et $X_{ij}(t)$ est le facteur d'incertitude dû au multitrajet et à l'ombrage. Comme la puissance transmise par l'ancre, P_T^i, est habituellement fixe et connue par les récepteurs, la puissance du signal reçu peut être utilisée pour estimer la distance avec une erreur proportionnelle au facteur d'incertitude, $X_{ij}(t)$, et l'erreur de mesure RSSI.
La précision des techniques de télémétrie RSSI est limitée. Tout d'abord, les effets de l'ombrage et des trajets multiples tels que modélisés par le terme $X_{ij}(t)$ en (2.1) peuvent être graves et nécessiter de multiples mesures de distance. La difficulté d'estimer les paramètres du modèle de canal dans (2.1) est un autre défi majeur que posent les mesures de distance basées sur le RSSI. L'inconvénient du RSS apparaît en affectant le bruit de l'environnement, ce qui rend le RSS sans précision pour mesurer la distance entre l'émetteur et le récepteur.

2.4 Algorithmes à portée libre "Range-free"

2.4.1 Méthode de localisation DV-Hop

La méthode DV-Hop [15] est similaire aux schémas de routage vectoriel de distance alors qu'elle contient trois étapes. Tout d'abord, les ancres diffusent les messages de localisation dans le fichier réseau. Ces messages contiennent une identification (ID), un emplacement (X,Y) et un nombre de sauts. Un nœud qui reçoit un message stocke ce message dans sa mémoire, puis diffuse ce message après avoir incrémenté le nombre de sauts de 1 dans un réseau. Si le message nombre de sauts reçu est supérieur au message stocké, le message est annulé par le nœud du capteur, sinon si le message nombre de sauts reçu est inférieur au message stocké, le nœud met à jour sa mémoire par une nouvelle information puis diffuse ce message après incrémentation du nombre de sauts par 1. Cependant, si le nœud est conscient des informations du nœud d'ancrage, le nœud diffusera ces informations après avoir incrémenté le nombre de sauts par 1. Par conséquent, les messages de réception au nœud de la même ancre permettent d'établir le chemin le plus court. Lorsqu'un nœud reçoit au moins trois informations d'ancres, cette étape est terminée.
Au cours de ce mécanisme, l'ancre stocke le message de localisation des ancres, afin de calculer la longueur moyenne des sauts (AHL pour "Average Hop Length"). Par exemple, comme le montre la figure 2.5, nous avons trois ancres identifiées par les étiquette ($a1$, $a2$ and $a3$), les noeuds réguliers représentés sans étiquette, et la ligne d'intersection entre les cercles est considérée comme un nombres de sauts.

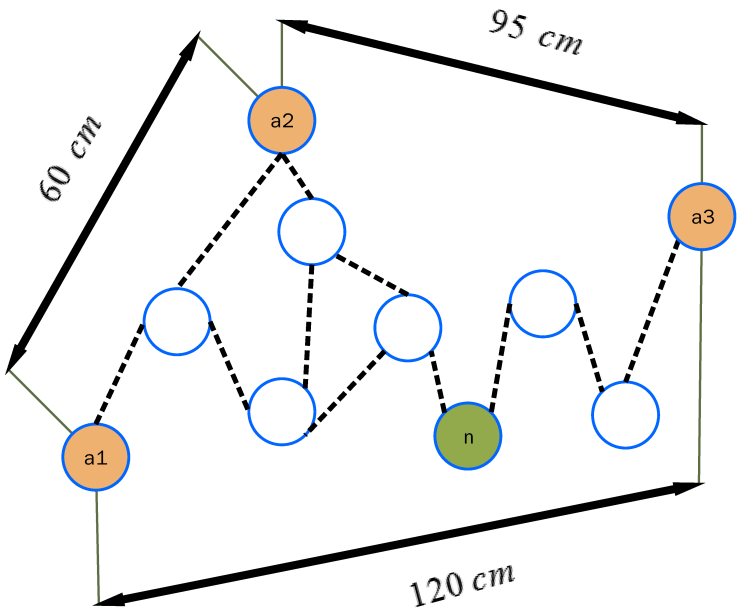

Figure 2.5 – L'algorithme DV-Hop: exemple.

Chaque nœud d'ancrage est capable de calculer son AHL en s'appuyant sur l'utilisation des distances entre lui et les autres nœuds d'ancrage dans le réseau et en évaluant le nombre de sauts que le message prend pour arriver des autres nœuds d'ancrage à lui-même. Les distances entre tous les nœuds d'ancrage peuvent être mesurées car les coordonnées de chaque nœud sont connues. Dans la figure 2.5, supposons que les distances entre les ancres deviennent de $a1$ à $a2$ soit 60cm, de $a1$ à $a3$ soit 120cm et de $a2$ à $a3$ 95cm. L'unité de distance peut être modifiable, mais nous avions l'habitude de prendre l'échelle de l'unité de distance en cm pour un exemple graphique.
Le nombre de sauts vient en ajoutant au message diffusé depuis le nœud d'ancrage à tous les autres nœuds du réseau un nombre variable. Le nombre de variables est dénombrable entre les nœuds, tandis que chaque nœud reçu est capable de gérer ce nombre de variables. Si le message reçu contient un nombre de variables supérieur au nombre de variables de nœud stocké dans la base de données de nœuds, le nœud rejette ce message si le message reçu contient un nombre de variables inférieur à son nombre de variables stockées, le nœud met à jour sa base de données en adoptant à la place le nombre de variables de réception de celui stocké, puis diffuse le message après avoir incrémenté le nouveau nombre de variables de mise à jour de un. Par exemple, le nombre de sauts

dans la figure 2.5 représente par la ligne pointillée entre les nœuds représentés par des cercles de styles différents. Les sauts sont déterminés graphiquement dans la figure 2.5 par le chemin le plus court du message d'ancrage entre les nœuds, par exemple le nombre de sauts entre les nœuds de $a1$ à $a2$ est de 2, de $a1$ à $a3$ est 7 et de $a2$ à $a3$ est 6.

Après chaque nœud d'ancrage diffuse son message dans le réseau. Le nœud d'ancrage est capable d'obtenir son AHl en faisant la somme des distances entre lui-même et les autres nœuds du réseau et en divisant cette somme par la somme du nombre mesuré de sauts entre lui-même et les autres nœuds du réseau. Par exemple, $AHL(a1)$ calcule en faisant la somme entre les distances de l'ancre $a1$ aux autres ancres existantes dans le réseau comme le montre la figure 2.5 les distances de $a1$ à $a2$ est de 60 cm et de $a1$ à $a3$ est de 120 cm, le résultat de la somme de ces deux distances est de 180 , tandis que ce résultat de la somme est divisé en calculant la somme du nombre de sauts entre l'ancre $a1$ et les autres, comme le montre la figure 2.5 de $a1$ à $a2$ est 2 et de $a1$ à $a3$ est 7, alors la somme des sauts devient 9, puis $AHL(a1)$ est égal à $\frac{120+60}{7+2} = \frac{180}{9} = 20$. Les autres nœuds d'ancrage AHL calculés comme:

$$\begin{aligned} AHL(a2) &= \frac{95+60}{6+2} = 19.37, \\ AHL(a3) &= \frac{120+95}{6+7} = 16.53. \end{aligned} \tag{2.2}$$

Après que tous les nœuds d'ancrage mesurent AHL, chaque nœud d'ancrage diffuse son AHL à travers le réseau. Le nœud régulier reçoit ces messages AHL de toutes les ancres et les stocke dans sa mémoire, puis en appelant le nombre de sauts stocké qui a été conservé en mémoire pour chaque ancre, le nœud régulier calcule les distances jusqu'à chaque ancre en multipliant le nombre de sauts minimum et longueur moyenne de saut AHL à chaque nœud d'ancrage du réseau. Comme le montre la figure 2.5, pour le nœud n, le nombre minimum attendu de sauts de n à $a1$ est de 4, de n à $a2$ est de 3 et de n à $a3$ est 3, alors la distance d'estimation entre le nœud régulier n et chaque nœud d'ancrage est:

$$\begin{aligned} n - a1 &= 4 * 20 = 80, \\ n - a2 &= 3 * 19.37 = 58.11, \\ n - a3 &= 3 * 16.53 = 49.61. \end{aligned} \tag{2.3}$$

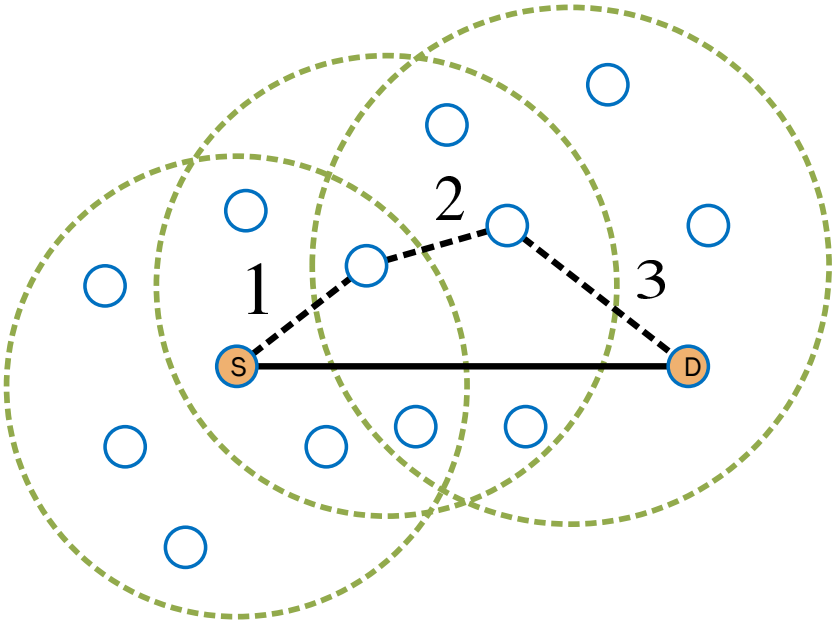

Figure 2.6 – Déploiement d'un WSN clairsemé (sparse).

Désormais, chaque nœud régulier est capable d'estimer sa position en utilisant la multilatération.

2.4.2 Méthode de localisation LEAP

LEAP [24] est une approche qui découle d'une analyse analytique spécifique. Dans un WSN à forte densité, un chemin plus court peut être couvert entre n'importe quelle paire de capteurs. La figure 2.6 illustre une telle situation. En ajoutant un saut à partir du nœud source, la distance cumulative augmente probablement d'une distance de transmission. Ainsi, si tous les nœuds ont la même plage de communication, la distance d'estimation entre deux paires (ex : du nœud S au nœud D) peut être calculée par le produit de la plage de communication et du nombre de sauts minimal entre eux. Ce produit mathématique est représenté comme suit:

$$d = h * r_0, \tag{2.4}$$

où h est le nombre de sauts entre S et D, et r_0 est la capacité de transmission qui représente la capacité du nœud à communiquer directement avec d'autres nœuds dans le cas où les nœuds sont couverts par la communication zone du nœud.

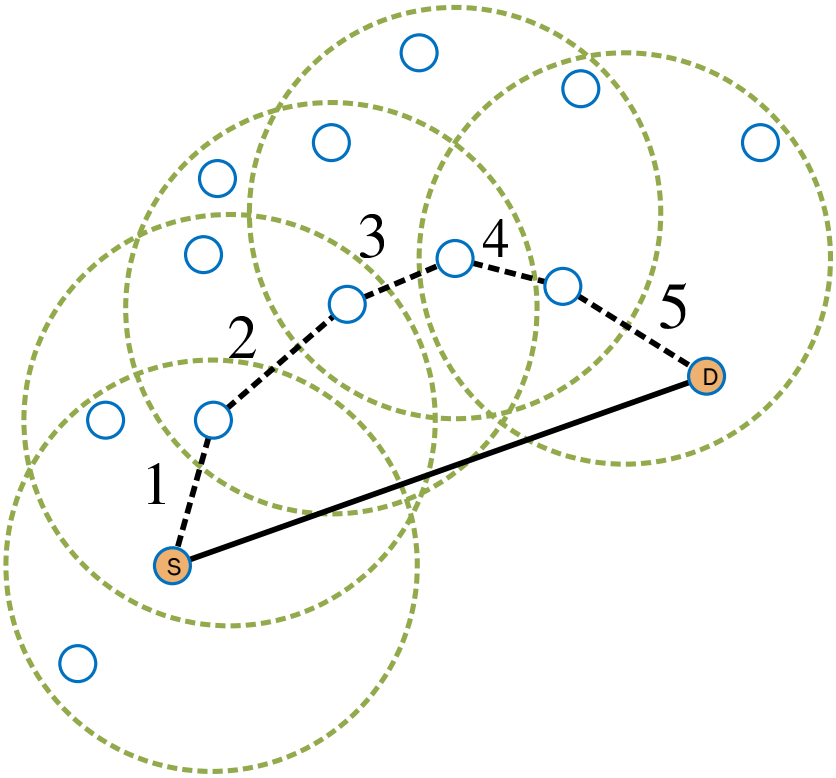

Figure 2.7 – Déploiement d'un WSN clairsemé (sparse)

Cependant, si nous prêtons attention au cas où les nœuds sont peu déployés dans le réseau, comme le montre la figure 2.7, cela nous donne une idée que la densité de nœuds dans un WSN sparse n'est pas suffisante pour construire un chemin à sauts multiples le plus court entre les capteurs. Dans ce cas, pour tout capteur intermédiaire le long du trajet, le message du capteur de transfert est extrêmement faible si le capteur est situé à la limite de sa portée de communication. Alors que la figure 2.6 repose sur un WSN densément distribué qui donne un nombre de sauts entre le nœud S et le nœud D égal à 3. En revanche, dans la figure 2.7, nous voyons un déploiment sparse du WSN, qui donne un nombre de sauts entre les nœuds S et D égal à 5. En effet, les nœuds situés à la limite du nœud émetteur ont une faible probabilité de recevoir le message. Par conséquent, l'estimation de la distance est loin d'être précise.

Afin d'optimiser ce problème, une analyse spécifique est nécessaire pour estimer la distance entre deux nœuds quelconques en utilisant la progression attendue du saut déterminée par l'équation suivante:

$$d = h * E(R), \tag{2.5}$$

où h est le nombre de sauts entre les nœuds, et $E(R)$ est la progression de sauts prévue entre les capteurs voisins dans un WSN.

Stratégie LEAP

Tableau 2.1 – Symboles

Symbole	Signification
Eo	les progrès prévu de sauts
Ai	ID de l'ancre Ai
$\sum_{Ec}^{i}$	Progression cumulative des sauts courant pour l'ancre Ai
$\sum_{Hc}^{i}$	Nombre cumulatif de sauts courant pour l'ancre Ai
$\sum_{Er}^{i}$	Progression cumulative des sauts reçu pour l'ancre Ai
$\sum_{Hr}^{i}$	Nombre cumulatif de sauts reçu pour l'ancre Ai

Description du processus LEAP : le processus LEAP [16] commence par la diffusion des messages de localisation à partir des ancres. Chaque message de localisation contient les informations info= $[Ai, Xi, Yi]$, $\sum_{Hc}^{i}$ et $\sum_{Ec}^{i}$, où (Xi, Yi) représente les coordonnées connues pour Ai, $\sum_{Hc}^{i}$ et $\sum_{Ec}^{i}$ sont le nombre cumulatif de sauts et la progression cumulative des sauts pour l'ancre Ai, respectivement. Initialement, les termes $\sum_{Hc}^{i}$ et $\sum_{Ec}^{i}$ ont une valeur vide (zéro) lorsque le processus démarre. Lorsque le nœud reçoit le message diffusé par l'ancre Ai pour la première fois, il met à jour sa mémoire en enregistrant le message de localisation reçu, puis modifie le message de localisation reçu et le diffuse dans le voisinage. Cette modification se produit dans les deux paramètres du message de localisation $\sum_{Hc}^{i}$ et $\sum_{Ec}^{i}$ tandis que le premier en incrémentant un à la somme $\sum_{Hr}^{i}$ et le second paramètre en ajoutant à $\sum_{Er}^{i}$ le terme Eo selon (2.5). En outre, le nœud régulier compte une ancre différente en incrémentant une variable compteur entière dans sa propre mémoire, le compteur est initialement remis à zéro. En raison de l'inondation du message de localisation dans le réseau, ce cas donne une probabilité élevée pour qu'un noeud reçoive un message de localisation qui contient le même ID d'ancre et qui vérifie déjà ses informations. S'il reçoit un nouvel ID d'ancre, le

nœud stocke ce message dans sa mémoire, incrémente le nombre de variables locales de un et fait la modification précédente puis le diffuse à ses voisins. Si le nœud reçoit un message de localisation concernant un nœud d'ancrage qui a déjà été enregistré dans la mémoire du nœud, le nœud vérifie sa mémoire pour voir si $\sum_{Ec}^{i} > \sum_{Er}^{i} + Eo$; le nœud met à jour sa mémoire par le message reçu et rediffuse ce message après avoir effectué la modification, sinon le message est ignoré par le nœud.

2.4.3 Algorithme proposé

Estimation de la distance

Pour que le i-ème nœud régulier (c'est-à-dire le $(N_a + i)$-ème nœud) puisse se localiser, il faut estimer les distances qui le séparent d'au moins 3 ancres. Jusqu'à présent, dans la plupart des algorithmes analytiques, le i-ème noeud régulier estime sa distance à la k-ème ancre $d_{k-(N_a+i)}$ en utilisant seulement l'information n_k comme suit:

$$\hat{d}_{k-(N_a+i)} = n_k \bar{h}_{\mathrm{s}}, \tag{2.6}$$

où $\bar{h}_{\mathrm{s}}$ est une taille de sauts moyenne prédéfinie.
Soit ϵ l'erreur d'estimation de la distance entre la taille moyenne $\bar{h}_{\mathrm{s}}$ et la taille réelle h_{s}. Nous avons:

$$\epsilon_{ik} = \bar{h}_{\mathrm{s}} - h_{\mathrm{s}}, \tag{2.7}$$

Malheureusement, en raison de cette méthode d'estimation de la distance, on s'attend à ce que des erreurs cumulatives se produisent lors de l'estimation de la distance entre chaque paire de nœuds d'ancrage réguliers, ce qui nuit à la précision de la localisation. Dans ce mémoire, nous proposons d'estimer directement (c.-à-d. sans recourir aux distances entre les nœuds intermédiaires consécutifs) la distance d_{k-i} entre la k-ième ancre et le i-ième nœud régulier. Cela devrait permettre d'éviter les erreurs d'estimation cumulées encourues par les estimations fondées sur l'EHP (pour "Expected Hop Progress") [24]. En fait, une telle erreur, qui a un effet négatif sur la précision de localisation, augmente avec n_k. Comme l'illustre la figure 2.8, désignons par X la ligne droite entre les deux nœuds ancre et régulier. Ensuite, nous pouvons le calculer comme suit :

$$X = z \times \cos(\theta), \tag{2.8}$$

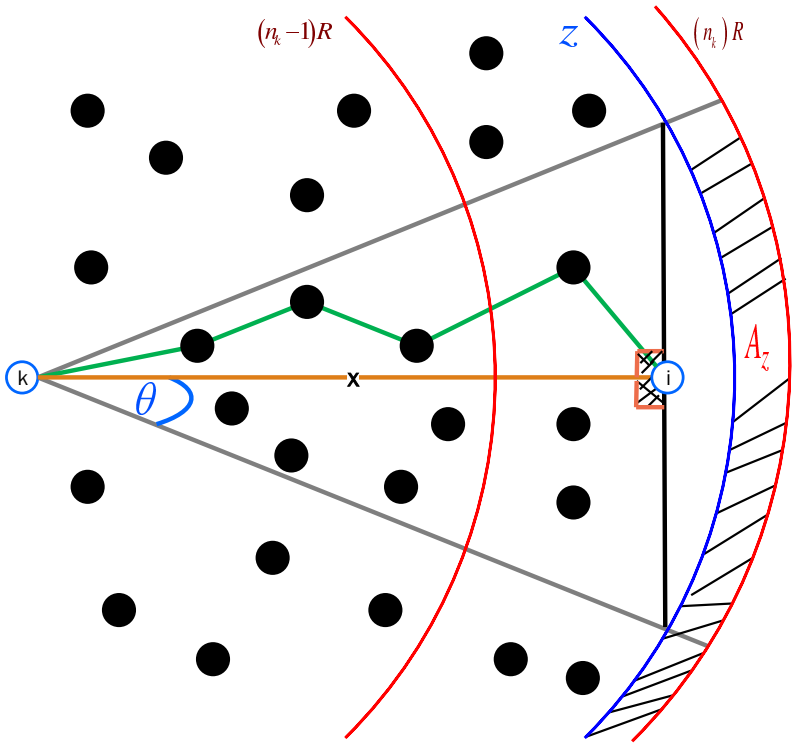

Figure 2.8 – Estimation de la distance.

où z est une variable aléatoire qui représente le rayon du secteur ayant le $\star$-ième noeud comme un centre et $\frac{\pi}{3}$ comme un angle. Alors il est facile de montrer que:

$$\begin{aligned} \mathrm{E}\{X|n_k\} &= \frac{\int_0^{\frac{\pi}{3}} \cos(\theta) \int_{(n_k-1)T_c}^{n_k T_c} z f_Z(z)\, \partial z \partial\theta}{\int_0^{\frac{\pi}{3}} \int_{(n_k-1)T_c}^{n_k T_c} f_Z(z)\, \partial z \partial\theta} \\ &= \frac{3\sqrt{3} \int_{(n_k-1)T_c}^{n_k T_c} z f_Z(z)\, \partial z \partial\theta}{2\pi \int_{(n_k-1)T_c}^{n_k T_c} f_Z(z)\, \partial z \partial\theta}, \end{aligned} \tag{2.9}$$

où $f_Z(z)$ est la fonction de densité de probabilité de la variable aléatoire Z.

Pour dériver $\mathrm{E}\{X|n_k\}$, on commence par dériver la fonction de distribution cumulative conditionnelle $F_{Z|n_k}(z|n_k)$ de Z par rapport à n_k. Comme on peut le voir à la figure 2.8, $Z \leq z$ est garantie seulement s'il n'y a pas de nœuds dans la zone A:

$$A_z = S(k, \theta, n_k T_c) - S(k, \theta, z), \tag{2.10}$$

$S(\star, \theta, x)$ est le secteur ayant le $\star$-ème noeud comme centre, $\frac{\pi}{3}$ comme angle, et x comme rayon. $F_{Z|n_k}(z|n_k)$ peut alors être définie comme suit:

$$F_Z(z)\,(z|n_k) = P\,(Z \leq z|n_k) = P\,(\mathrm{E})\,, \tag{2.11}$$

où $P(\mathrm{E})$ est la probabilité que l'événement $\mathrm{E} = \{\text{aucun noeud dans la zone } A_Z\}$ se produise.
Étant donné que les nœuds sont déployés uniformément dans la zone environnante (cf. figure 2.8), la probabilité d'avoir K noeuds en A_z suit une distribution binomiale $\mathrm{Bin}\,(N, p)$ où $p = \frac{A_z}{S}$ et S est la surface totale de la zone. Pour N relativement grand et p petit, on peut facilement montrer que $\mathrm{Bin}\,(N, p)$ peut être approchée avec précision par une distribution de Poisson $\mathrm{Pois}(\lambda A_z)$ où $\lambda = N/S$ est la densité moyenne des noeuds dans le réseau. Par conséquent, pour un grand nombre de nœuds N et un petit p, nous avons

$$F_{Z|n_k}\,(z|n_k) = e^{-\lambda A_z}. \tag{2.12}$$

La figure 2.8 permet de montrer facilement que:

$$A_z = \frac{\pi}{3}\left(n_k^2 T_c^2 - z^2\right), \tag{2.13}$$

où T_c est la plage de communication de chaque nœud tandis que le nœud peut communiquer directement avec tous les autres au cas où les nœuds seraient couverts par sa zone de communication, et par conséquent:

$$f_Z(z) = \frac{2\lambda\pi}{3} z e^{\frac{-\lambda\pi}{3}\left(n_k^2 T_C^2 - z^2\right)}. \tag{2.14}$$

Après injection de (2.14) dans (2.9), changement de variable et une intégration partielle, $\mathrm{E}\,\{X|n_k\}$ est donné par (2.15) dont le calcul repose uniquement sur des informations déjà disponibles localement au i-ème nœud régulier et n'entraîne pas de retards de traitement supplémentaires, de coûts ou de consommation électrique. Une telle caractéristique est en fait très appropriée et souhaitable pour les WSNs où l'énergie est une ressource rare et le coût du matériel doit être maintenu aussi bas que possible. Il découle de (2.15) que $\mathrm{E}\,\{X|n_k\}$ augmente avec R et il peut être facilement calculé à chaque nœud régulier étant donnée la connaissance a priori de la distribution des nœuds avant le déploiement WSN. Par conséquent, le i-ème nœud régulier est capable d'obtenir un estimation initiale de ses coordonnées (x_0, y_0) à l'aide de toute technique de multilatération.

$$\mathrm{E}\{X|n_k\} = \frac{3e^{-\frac{\lambda\pi}{3}n_k^2T_c^2}\left(2\sqrt{3}\left((n_k-1)e^{\frac{\lambda\pi}{3}(n_k-1)^2T_c^2} - n_k e^{\frac{\lambda\pi}{3}n_k^2T_c^2}\right)\sqrt{\lambda}T_c - 3erfi\left(\sqrt{\frac{\lambda\pi}{3}}(n_k-1)T_c\right) + 3erfi\left(\sqrt{\frac{\lambda\pi}{3}}n_kT_c\right)\right)}{4\pi\sqrt{\lambda}\left(e^{\frac{1}{3}(1-2n_k)\lambda\pi T_c^2}-1\right)} \tag{2.15}$$

où n_k est le numéro du nombre de sauts, T_c est la capacité de transmission, $lambda$ est la densité des nœuds, erf est une fonction d'erreur et une fonction entière utilisée dans l'analyse, la fonction erf est intervenue dans le fichier des probabilités et statistiques le $erfi$ considère la fonction inverse de la fonction erf qu'elle montre comme: $erfi = \frac{2}{\pi} \int_0^x e^{x^2} dt$.

Stratégie de l'algorithme proposé

Comme toute première étape d'un algorithme de localisation, les ancres diffusent un paquet qui se compose de deux parties : l'en-tête et la partie utile des données. La partie entête porte l'ID d'identification de l'ancre, tandis que la partie utile de données porte la position dé l'ancre (x_k, y_k) et un nombre de sauts n initialisé à 1. Le noeud qui reçoit ce message le stocke dans sa base de données et remplace sa valeur de comptage des sauts n_k par la valeur de comptage des sauts $n_k = n$, puis diffuse le paquet après avoir augmenté n de 1. L'inondation du même paquet entre les noeuds du réseau donne donc une probabilité élevée pour que les noeuds reçoivent plusieurs fois le même paquet. Ensuite, le nœud vérifie immédiatement les informations de sa base de données. Si le nombre de sauts reçus n est inférieur au nombre de sauts stockés n_k, le nœud met à jour le nombre de sauts stockés par le nombre de sauts reçu, puis réémet le paquet après avoir incrémenté le nombre de sauts par 1. Si le nombre de sauts n est supérieur au nombre de sauts stockés n_k le noeud rejette simplement le paquet reçu. Cependant, si les nœuds ignorent l'information de la k-ème ancre, ils ajoutent 1 à la valeur de compte de saut n puis diffusent le paquet reçu. Ce mécanisme a lieu en continu jusqu'à ce que tous les nœuds prennent connaissance de la position de toutes les ancres et trouvent en conséquence le chemin le plus court vers chaque k-ème ancre.

2.5 Technique de positionnement par trilatération

Le principe de la technique de trilatération [11] est basé sur les propriétés du triangle, afin d'estimer l'emplacement des nœuds réguliers. Cette méthode nécessite au moins trois ancres qui connaissent leur position. La technique de trilatération exploite l'information de ces trois ancres et utilise un centre de localisation vers le nœud régulier, comme le montre la figure suivante:

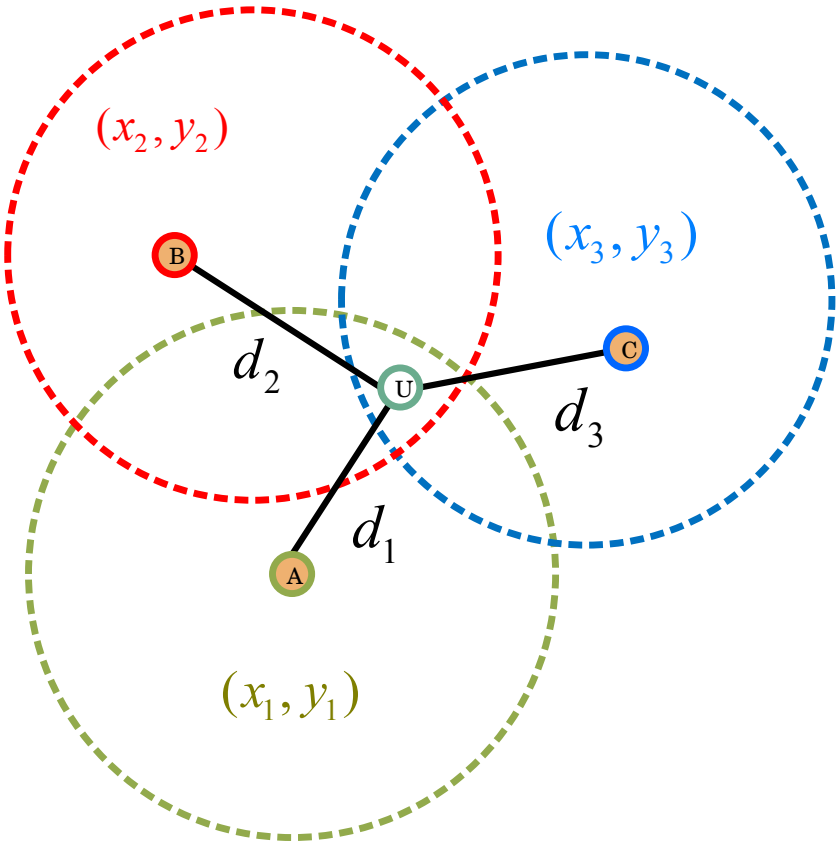

Figure 2.9 – Exemple de méthode de trilatération.

Le point intérieur estime son emplacement à l'aide des coordonnées obtenues des trois ancres et correspondant aux distances par rapport à chacune d'elles. En effet, l'intersection des trois cercles est considérée comme l'emplacement du point intérieur d'un triangle. Le cercle est déterminé par l'ancre en tant que centre et la distance entre le centre et le point intérieur est la portée de communication (ex : rayon du cercle). Comme le montre la figure 2.9 dans une telle situation, les nœuds d'ancrage sont les trois points qui ont une étiquette appelée A, B et C avec des coordonnées spécifiques, respectivement (x_1, y_1), (x_2, y_2) et (x_3, y_3) et le nœud régulier U a des coordonnées (X, Y). Comme le montre la figure 2.9 dans une telle situation, les nœuds d'ancrage sont les trois points qui ont une étiquette appelée A, B et C avec des coordonnées spécifiques, respectivement (x1, y1), (x2, y2) et (x3, y3) et le nœud régulier U a des coordonnées (X, Y). Ensuite, les distances entre le nœud

régulier et les nœuds d'ancrage peuvent être calculées comme:

$$\begin{aligned}(d_1)^2 &= (X - x_1)^2 + (Y - y_1)^2,\\ (d_2)^2 &= (X - x_2)^2 + (Y - y_2)^2,\\ (d_3)^2 &= (X - x_3)^2 + (Y - y_3)^2,\end{aligned} \tag{2.16}$$

Après linéarisation de ce système ci-dessus, on obtient :

$$AP = -\frac{1}{2}B, \tag{2.17}$$

où $P = [X, Y]$, A est une matrice de dimension $(N_a - 1) \times 2$, où N_a est le nombre d'ancres existant dans le réseau:

$$A = 2 * \begin{bmatrix}(x_1 - x_3) & (y_1 - y_3)\\ (x_2 - x_3) & (y_2 - y_3)\end{bmatrix}, \tag{2.18}$$

et B est un vecteur de dimension $(N_a - 1) \times 1$ comme suit:

$$B = \begin{bmatrix}d_1^2 - d_3^2 - x_1^2 + x_3^2 - y_1^2 + y_3^2\\ d_2^2 - d_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2\end{bmatrix}, \tag{2.19}$$

Puisque A est une matrice non inversible, $\hat{X}$ et $\hat{Y}$ peuvent être estimés avec le pseudo-inverse de A comme suit:

$$P \begin{bmatrix}X\\ Y\end{bmatrix} = (A^T A)^{-1} * (A^T B) \tag{2.20}$$

Par conséquent, le nœud régulier est capable d'obtenir une première estimation de ses coordonnées en se basant sur les définitions de A et B qui dépendent d'informations locales.

Chapitre 3

Contiki Cooja

3.1 Aperçu de Contiki

Contiki est un système d'exploitation open-source, intégré, modulaire, flexible et générique, basé sur un modèle d'exploitation hybride pour réseaux de capteurs. Ce système a été conçu en 2004 par un groupe de développeurs de l'industrie et par Adam Dunkels de l'Institut Suédois d'Informatique. Il fonctionne sur des microcontrôleurs de faible puissance et permet le développement d'applications qui permettent une utilisation efficace de l'équipement tout en fournissant une communication sans fil standardisée de faible puissance pour une gamme de plates-formes de microcontrôleurs telles que l'Atmel AVR et le TI MSP430, qui sont utilisées dans les familles Telos, Tmote et MICAz.
Cooja simule les applications Contiki. Le développement d'applications Contiki nécessite la connaissance de Contiki. Les principales caractéristiques de Contiki sont mentionnées ci-dessous:
. Système d'exploitation open source (OS) pour les nœuds WSN.
. Chargement dynamique des modules.
. Consomme de la mémoire et des ressources efficacement.
. Contient un protocole réseau standard comme TCP, UDP, IPv4, IPv6, RPL.
. Inclut un protocole MAC simple tel que ContikiMAC.
. Fournit une estimation de la puissance.
. Utilise les protothreads comme fil léger sans pile.
. Utilise un système de fichiers pour la lecture et l'écriture sur flash externe.
. Permet d'inclure une pile réseau pour un fonctionnement réseau simple.

. Similaire à la pile Rime.

. Peut être utilisé sur une large gamme de plates-formes telles que MICAz.

3.2 Conception et caractéristiques de Cooja

Cooja se compose du type de nœud, de la mémoire de nœud, de l'interface, du plugin et de l'observateur. La figure 3.1 montre la conception de Cooja.

Type de nœud: La simulation Cooja peut contenir plusieurs types de nœuds. Les nœuds qui exécutent le même programme d'application, le même système d'exploitation et des composants matériels similaires appartiennent au même type de nœud. Par exemple, un nœud MICAz exécutant "hello world" sur le système d'exploitation Contiki est un type de nœud unique. Les noeuds qui ont le même type de noeud compilent leur programme d'application en chargeant le programme d'application sur Contiki.

Mémoire de nœud: En partageant le même programme d'application, les nœuds obtiennent le même type de nœud, mais chaque nœud est attribué séparément et dans une mémoire différente. La mémoire de nœud est interchangeable pendant la simulation en fonction des variables du programme d'application; par exemple, la variable de comptage des sauts mise à jour après la réception d'une nouvelle variable de comptage des sauts qui est inférieure à la variable stockée.

Interface: Les interfaces interagissent avec le composant matériel et le système d'exploitation et sont utilisées pour les services du nœud. Les interfaces ont de nombreuses propriétés telles que la modification de la variable mémoire, les événements déclencheurs et l'appel des fonctions du système d'exploitation. Par exemple, pendant l'inondation du message d'ancre à travers le réseau, un événement est généré et ajouté à la file d'attente. Quand la dernière est en cours d'utilisation, une LED peut être allumée en interagissant avec l'interface LED.

Observateurs Cooja: Cooja a des observateurs pour écouter les interfaces. Bien que ces fonctions soient enregistrées avec les interfaces correspondantes, elles commencent à écouter les interfaces. Par exemple, le média radio est un observateur standard pour les émetteurs radio de tous les nœuds d'une simulation. Le média radio écoute les émetteurs radio, met les données à la disposition des autres émetteurs qui attendent ces données et collecte les données du réseau.

Plugins Cooja: Les plugins Cooja sont des interfaces utilisateur graphiques qui permettent à l'uti-

lisateur de contrôler une édition et de visualiser la simulation. Par exemple, un plugin de contrôle de simulation est utilisé pour démarrer, arrêter et mettre en pause. La fenêtre de simulation principale de Cooja contient un certain nombre de plugins Cooja, comme le montre la figure 3.2. De petites descriptions du plugin qui apparaît dans la fenêtre principale de Cooja sont données ci-dessous.

1- La fenêtre réseau affiche le nœud réseau simulé. Cette fenêtre peut être conforme aux propriétés du réseau, telles que la portée de communication de chaque nœud, le trafic radio, les adresses IP des nœuds, la position du nœud, les LEDs et le type du nœud. Le nœud peut également ajuster sa position et peut être zoomé vers l'avant ou vers l'arrière ou réinitialisé pour afficher le réseau.

2- Chaque nœud peut afficher ses sorties sur la fenêtre de sortie en utilisant les messages 'printf' du nœud.

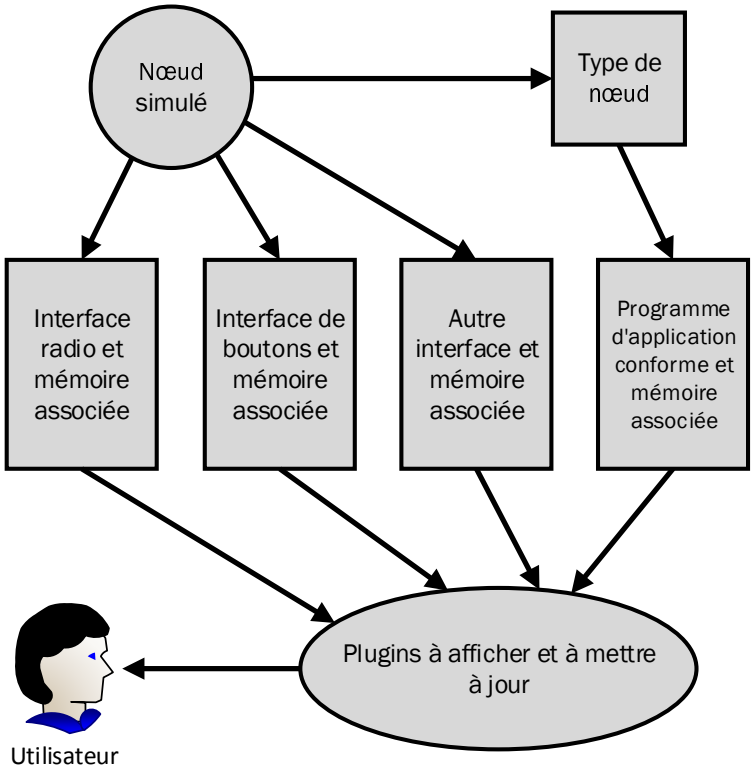

Figure 3.1 – Conception de Cooja.

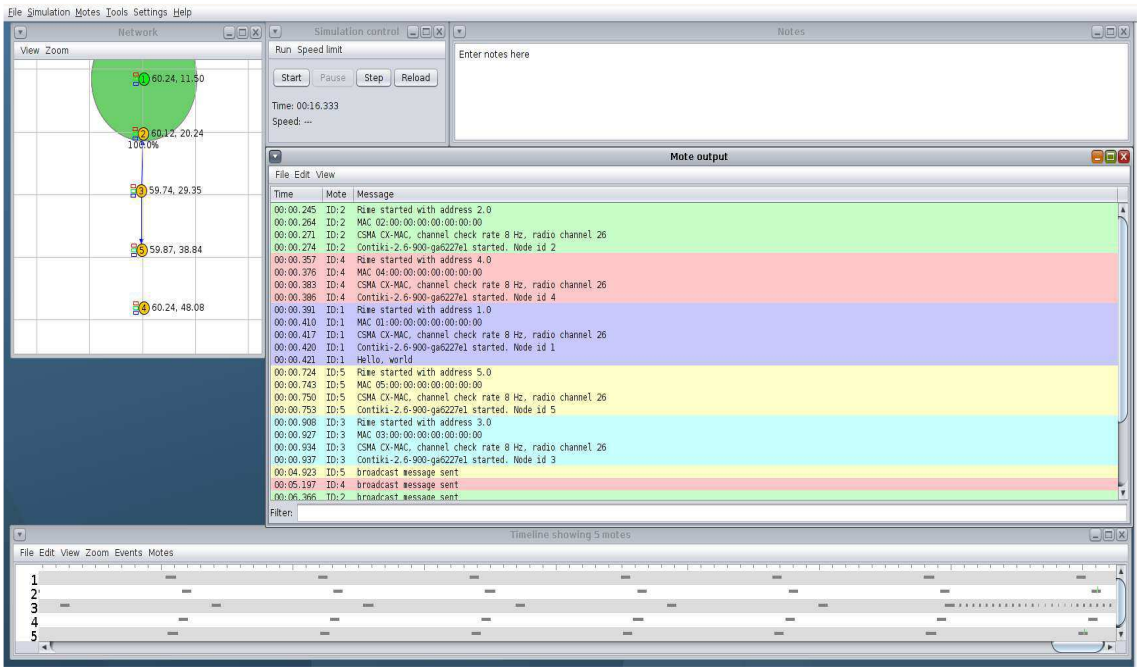

Figure 3.2 – Simulation Cooja.

3.3 Structure de Cooja

Les nœuds au niveau des instructions sont émulés matériellement puisque le fichier exécutable est contraint par une architecture spécifique. Lors de la simulation du code natif des nœuds, le type de nœud est considéré comme un lien entre le nœud et le système Contiki compilé. Le type de noeud charge la bibliothèque de compilation et toutes les interactions entre le simulateur et la bibliothèque. Le nœud capable de gérer des fonctions interactives avec le système Contiki de charge comme la copie et le remplacement du segment de mémoire, l'initialisation du système Contiki et enfin une fonction qui donne la permission à un système Contiki de gérer un événement.

Les nœuds du même type partagent la même mémoire de code de programme. En raison des différents types de nœuds, il existe différentes mémoires de données pour chaque nœud de capteur simulé. Cooja génère et stocke une copie de la mémoire de l'environnement C dans chaque noeud. Cooja exécute un code natif en effectuant des appels Java Native Interface (JNI) depuis un environnement Java vers un système Contiki compilé. Cependant, le simulateur Java a le plein contrôle d'une mémoire de nœuds simulés.

Lorsque le nœud de capteur est en cours d'exécution, sa mémoire de données est copiée dans l'environnement C et un appel JNI est effectué vers le noyau Contiki déclenché par événement. Le noyau

Contiki envoie un seul événement à partir de sa file à l'un des processus en cours d'exécution dans le système. Dans ce cas, la mémoire de données mise à jour est récupérée dans l'environnement Java. Pour chaque système Contiki chargé, Cooja doit trouver les adresses des fonctions et variables. Cela se produit en analysant le fichier de carte généré au moment du lien. Le fichier de carte se compose d'informations sur les adresses des symboles dans la bibliothèque. Cooja est capable de calculer les adresses de toutes les variables de la bibliothèque en comparant l'adresse mémoire absolue d'une variable au moment de l'exécution et son adresse relative classée dans le fichier de carte.
Le fichier de carte analysé permet au simulateur de récupérer et de modifier des variables pendant les simulations. C'est le principal moyen de communication entre une interface et le système Contiki. La figure 3.3 représente la structure Cooja.

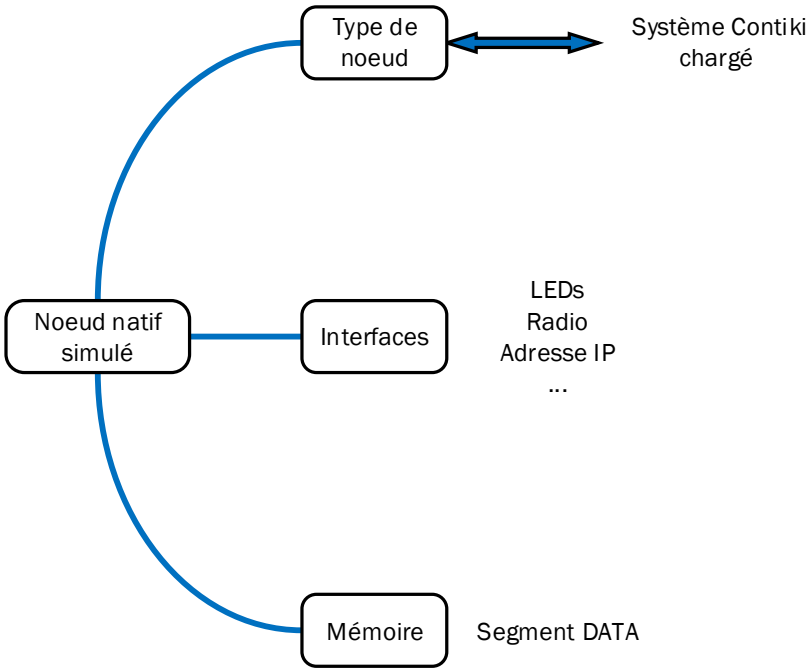

Figure 3.3 – Structure Cooja.

3.4 Application Contiki

Une application Contiki fait référence à l'exécution d'un programme sur un nœud matériel. Le programme est établi par le langage C écrit qui consiste en un "processus". Pour une application complète de Contiki, les étapes suivantes doivent être considérées.

1. **Déclaration du processus**: Chaque processus doit être déclaré dans les applications WSN en utilisant la macro PROCESSUS. Cela prend deux paramètres, le premier spécifie le nom du processus et le second donne le nom d'un processus au format sting. La macro PROCESSUS est notée comme:
   $PROCESS(hello\_world\_process, "Ahelloworldprocess");$
2. **Processus à impliquer**: Lors de la macro déclaration des processus, les processus peuvent être exécutés dans l'application Contiki. La macro fait référence aux processus en tant que paramètres. Par exemple:
   $AUTOSTART\_PROCESSES(\&process-one, \&process-two);$
3. **Définition du processus**: La macro définit le processus en prenant le nom d'un processus comme paramètre. Il contient également deux autres paramètres, *ev* et *data*. Le paramètre *ev* représente l'événement qui sera enregistré dans le processus tandis que *data* spécifie les données, ce qui donne avec l'événement enregistré:
   $PROCESS\_THREAD(hello\_world\_process, ev, data);$
4. **Limites du processus**: Un protothread est considéré comme une fonction C normale. La fonction commence et se termine entre des macros spécifiques (PROCESS_BEGIN() and PROCESS_END();), tandis que l'ensemble des fonctions de protothread entre ces deux macros peut être utilisé comme indiqué ci-dessous :
   $PROCESS\_BEGIN();$
   $printf("Hello, world!");$
   $PROCESS\_END();$
5. **Manipulation d'événement**: Les macros de manipulation d'événement sont capables de gérer l'événement d'une manière continue comme une boucle While. Ces macros interrompent le processus en cours et attendent que l'événement soit terminé. Une fois l'événement réalisé, la macro permet l'exécution du processus. Le code de gestion des événements est exécuté sous la forme:
   $while(1)\{$
   $PROCESS\_WAIT\_EVENT();$
   $if(ev == sensors\_event \&\& data == \&button\_sensor)\{$
   $leds\_toggle(LEDS\_ALL);$
   $\}$
   $\}$

Le code ci-dessus améliore l'attente du processus jusqu'à ce qu'on appuie sur le bouton pour faire basculer toutes les LEDs.

6. **Minuteurs**: Contiki utilise différents types de minuteurs pour le développement d'applications, comme le minuteur d'événements, qui génère et interrompt le processus jusqu'à ce que la minuterie expire. Un temps d'événement (temporisateur) est déclaré et initialisé comme suit:
   $static\, struct\, etimer\, et; /* timer\, declaration */$
   $etimer\_set(\&et, CLOCK\_SECOND * 6); /* timer\, initialized\, and\, last\, for\, 6\, seconds */$
   $PROCESS\_WAIT\_EVENT\_UNTIL(etimer\_expired(\&et));$

3.5 Online stimulation Contiki Cooja

Notre niveau de référence de performance est indiqué par comparaison sous le même ensemble de l'algorithme proposé par rapport aux meilleures solutions de localisation sans plage représentatives rapportées dans la littérature (par exemple, DV-Hop [15] et LAEP [24]). Alors que dans toutes les stimulations, les nœuds sont déployés uniformément dans une zone 2 -D, $S = 50^2 \times 50^2$ et tous les nœuds ont la même plage de communication $R = 10m$ sur le réseau (c.-à-d. homogène). En outre, la différence entre les motifs normaux et le motif de la station de base est l'ID. Nous supposons que la station de base est la raison pour laquelle ID est égal à un. Pour les motifs normaux, l'ID doit être supérieur à un.

La figure 3.4 montre les informations du réseau où les nœuds ont reçu un message d'un nœud spécifique et ils stockent un nombre de sauts dans la mémoire. Où les nœuds d'ancrage ont des formes de cercle vert et les nœuds réguliers ont des formes de cercle orange. La topologie du réseau comme le montre la figure 3.4 un nœud qui se trouve au milieu de la topologie est connecté au nœud précédent et au suivant. Le nœud 12 est connecté au nœud 8 (précédent) et est connecté au nœud 2 (suivant) et l'autre nœud est connecté avec la même stratégie avec le nœud précédent et le nœud suivant pour le milieu de la topologie et avec un nœud au bord de la topologie. La station de base est le nœud de la façon dont détient le $ID = 1$. Tous les autres nœuds peuvent envoyer leur résultat final à la station de base soit directement ou indirectement en utilisant le WSN à sauts

multiples pour les nœuds, car ils ne peuvent pas communiquer directement entre eux en raison de la limitation de la plage de communication.

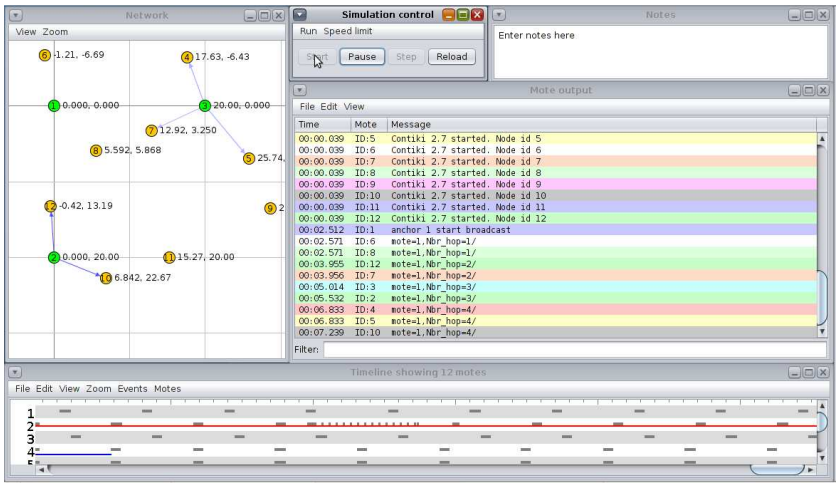

Figure 3.4 – Informations sur le réseau.

La figure 3.5 montre l'implémentation de l'un des algorithmes (sans plage) DV-Hop sur le programme Contiki Cooja. Comme indiqué dans la figure 3.5 après que les nœuds ont reçu les informations sur les trous des nœuds d'ancrage, les nœuds réguliers sont en mesure d'estimer leurs positions en utilisant l'algorithme de trilatération comme indiqué dans la figure 3.5 dans la sortie *Mote* (ex: $MatriceA$). Le résultat sur le nœud 8 comme indiqué dans la figure 3.5 *Section* Network les coordonnées réelles du nœud 8 sont (x= 55.01, y= 42.25) tandis que la sortie *Mote* montre les coordonnées d'estimation sont (x= 46.0209, y= 34,7093), cela signifie que l'erreur est significativement importante par rapport à la réalité en utilisant l'algorithme DV-Hop sur le programme Contiki Cooja.

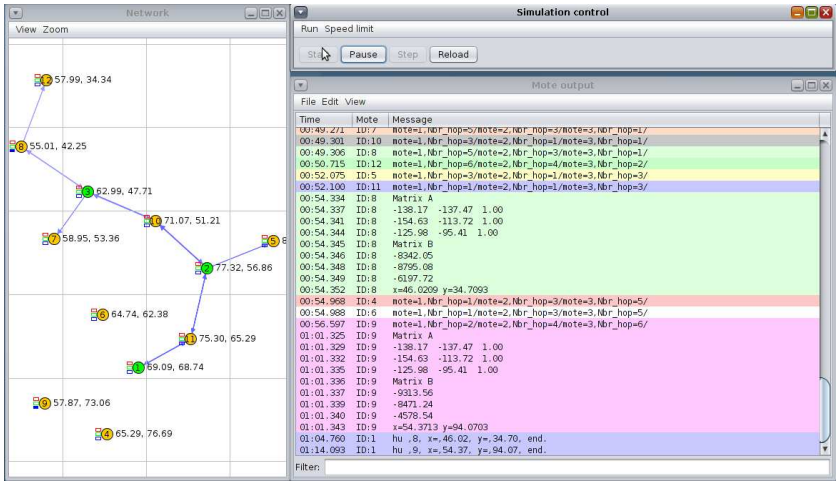

Figure 3.5 – Amélioration de l'algorithme de localisation DV-Hop.

La figure 3.6 les performances de l'algorithme proposé sur le programme Contiki Cooja. Comme le montre la figure 3.6 dans la section *Moteoutput*, les nœuds peuvent utiliser la trilatération comme le nœud 6 (ex: *MatrixA* et *MatrixB*) après avoir reçu toutes les informations sur les nœuds d'ancrage. Envoyez ensuite le résultat final à la station de base. la sortie *Mote* montre que le nœud 6 a un résultat d'estimation précis par (x= 3,7035, y= 13,1837) dans la sortie *Mote* alors que les coordonnées réelles sont (x= 2,613, y= 13,59) dans la section *Netwrok* . Par conséquent, l'algorithme proposé sur les tests sur le programme Contiki Cooja donne une grande précision.

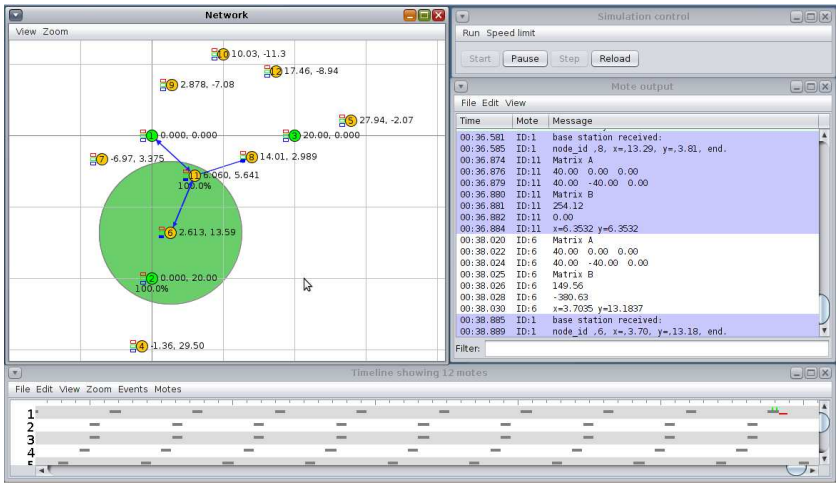

Figure 3.6 – Algorithme de localisation proposé.

Chapitre 4

Implémentation matérielle

4.1 Configurations matérielle et logicielle requises pour les applications WSN

4.1.1 MICAz Mote

MPR2400 MICAz mote est conçu pour les réseaux de capteurs hautement intégrés. MICAz offre un rapport de débit de données élevé de 250 kbps par l'utilisation d'une radio prête à fonctionner en zig-zag. La bande radio 2.4 GHz prête à fonctionner en zig-zag est un émetteur-récepteur RF conforme à la norme IEEE 802.15.4 et exécute le Moteworks à partir de sa mémoire flash interne [19] basée sur le microcontrôleur ATmega 128L à faible puissance. Le connecteur d'extension à 51 broches permet au MICAz mote de se connecter avec des périphériques extrêmes (ex : carte d'acquisition de données) et l'entrée/sortie analogique et numérique peut gérer l'instruction à 51 broches. La MICAz mote offre une sécurité matérielle (AES-128). De plus, MICAz mote peut être exploité pour faire fonctionner simultanément l'application capteur et la communication réseau.

4.1.2 MIB520CB Mote INTERFACE BOARD

La connexion entre le MIB520CB mote et le MICAz mote est le connecteur d'extension à 51 broches. La carte d'interface contrôle la communication et la programmation du MICAz mote

par identification USB. La connexion USB entre la carte d'interface et le PC offre pour la carte d'interface deux ports com, un port d'entrée pour la programmation en système et un port de sortie pour les données de communication via USB [4]. Notez que la carte d'interface MIB520CB mote doit installer FTDI FT2232C pour utiliser le port USB comme port COM virtuel.

4.1.3 Configuration logicielle

MoteWorks optimise les réseaux à faible charge de batterie et fournit une solution de bout en bout pour toutes les applications WSN. MoteWorks inclut un système d'exploitation open source qui prend plusieurs formes et doit être choisi en fonction des exigences de l'application. Dans certaines applications, il est très important que les nœuds fonctionnent sans surveillance pendant une très longue période de temps, alors que dans d'autres applications, il est très important que les nœuds soient capables de traiter une énorme quantité d'informations à court terme. Dans ce mémoire, nous avons donné la priorité au système d'exploitation Contiki au lieu de Tinyos, après avoir fait une comparaison [7] entre Tinyos et les systèmes d'exploitation Contiki. Cependant, Contiki est plus flexible lorsque le logiciel du nœud doit être mis à jour souvent pour une densité élevée de nœuds.

4.2 Configuration de l'application WSN pour l'algorithme de localisation

Cette section présente en détail comment configurer les équipements logiciels et matériels dans notre application WSN. Alors que n'importe quel algorithme de localisation peut être perceptible pour l'utilisateur en l'implémentant dans le monde réel.

4.2.1 Configuration matérielle

Certaines exigences devraient être prises en considération. La première est de connecter le MICAz mote au PC via USB pour compiler les informations du logiciel. La seconde, est l'établissement d'une station de base ou la passerelle qui va recevoir les informations de transmission des nœuds et afficher les données reçues sur le PC, comme le montre la figure 4.1:

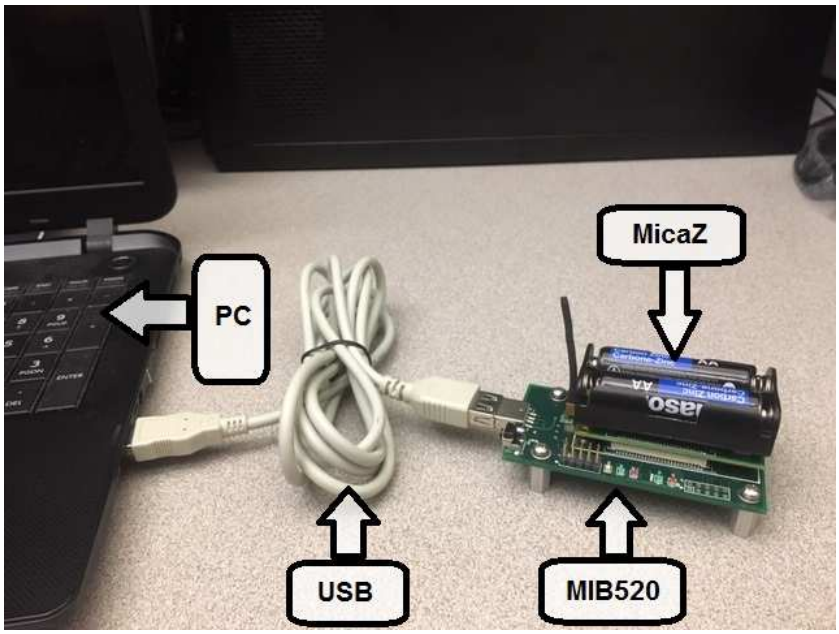

Figure 4.1 – Connexion entre MICAz et MIB520.

Notez que la station de base est construite sans batterie, grâce à l'offre de la carte d'interface qui rend le MICAz mote ON pendant la connexion à celle-ci.

4.2.2 Téléchargement du fichier exécutable dans la mote

Pour compiler le fichier exécutable dans le mote, l'équipement MIB520 doit être connecté au PC via un port USB et doit être alimenté. Dans un premier temps, il faut éteindre la mote et la connecter au MIB520. On doit s'assurez que la connexion est extrêmement bloquée. Toutes les étapes peuvent être consultées dans [8].

Aussi, un pré-compilateur approprié est nécessaire, comme Programmers Notepad 2. Programmers Notepad 2 est l'un des composants de MoteWorks. Après avoir réalisé toutes ces instructions, il construit le réseau avec des paramètres connus tels que le nombre d'ancres et de nœuds réguliers. Dans le simulateur Contiki Cooja, le fichier exécutable sera généré dans chaque nœud en choisissant l'ID d'identification et en transmettant TX. La première peut être ajustée en utilisant la bibliothèque C (node-id.h), qui donne l'ID au nœud grâce à une fonction flexible, c'est-à-dire node-id-burn(). La seconde peut être ajustée aussi en utilisant la bibliothèque C (dev/cc2420.h) Celle-ci dispose d'une fonction cc2420-set-txpower() qui modifie la puissance TX au niveau requis en dBm [23]. Ensuite, l'utilisateur transfère les fichiers exécutables au MoteWorks dans un réper-

toire d'application (ex : Blink.nc) attaché dans Programmers Notepad 2, à travers ce répertoire: $C : /Memsic/cygwin/opt/MoteWorks/apps/general\ /Blink/build/micaz$.
Notez que le fichier exécutable doit être renommé main.exe. Maintenant, en travaillant dans Notepad 2 on clique sur tools− > shell pour remplir le texte d'édition par une syntaxe générale pour l'installation qui est:
"make $platform$ re/install, n $programmer$, $port$"
où $platform$ désigne un type de processeur Mote, n est le nom de l'identification du nœud, $programmer$ est le nom de l'équipement mote qui est utilisé, $port$ est le numéro du PC hôte auquel le programmateur est connecté. Les étapes pour connaître l'attribution des numéros de port COM disponibles sont données dans [20]. Dans notre étude, les numéros de port COM attribués sont COM3 et COM4. La différence entre l'installation et la réinstallation est que le commentaire d'installation programme le périphérique Mote, tandis que le commentaire de réinstallation télécharge le programme pré-compilé dans le périphérique Mote. La forme finale de la syntaxe que nous avons utilisée pour télécharger le firmware dans le mote est:
"make $micaz$ reinstall,n mib520, com3".
Notre avons utilisée la couche Rime offerte par le système d'exploitation Contiki. Rime est définie par une pile de communication en couches légères [13] pour les réseaux de capteurs. Rime est un intermédiaire entre l'effort de la diffusion locale du voisin et l'effort d'inondation du réseau. Rime layer est conçue pour être beaucoup plus simple que les protocoles existants pour les abstractions de communication modulaire pour les réseaux de capteurs.
Notre formulaire de demande est en mode basse consommation d'énergie. En mode basse consommation d'énergie, la consommation d'énergie est réduite en raison de l'option "idle", que le processus prend au cas où il n'y a pas d'action à faire.
La différence entre les motes normales et la mote de la station de base est l'ID. Nous supposons que la station de base est la mote qui détient le numéro d'identité 1. Pour les motes normales, l'ID doit être supérieur à 1.

4.2.3 Configuration du logiciel

Pour afficher les résultats d'une estimation de réseau en PC, nous avons proposé dans notre étude une interface utilisateur GUI, qui a la capacité de recueillir les données des capteurs par l'intermédiaire de l'interface de communication standard USB. Les paramètres USB sont sélectionnés

par l'utilisateur afin de choisir un port COM spécifique qui convient au cas. Dans notre étude, le COM4 est le port choisi pour lire les données.

4.2.4 Mise en œuvre en conditions réelles

Des applications WSN ont été mises en œuvre à de nombreux endroits, avec des tâches différentes; par exemple, en Colombie, un WSN détecte et prévient les incendies de forêt [18] [1] et en Inde, un WSN est modifié pour être utile pour la gestion des catastrophes [17]. Dans les deux exemples précédents, il manque l'information sur la position des données de détection. De plus, dans notre laboratoire WirelessLab (Canada), nous avons modifié un WSN pour améliorer l'efficacité des algorithmes de localisation de la littérature [10] dans le monde réel. Cette section montre l'application de WSN dans le monde réel pour obtenir l'emplacement des MICAz motes qui sont des nœuds réguliers. De plus, nous mentionnons la méthode de débogage utilisée et quelques instructions qui devraient être vérifiées pour faire l'implémentation pour les applications WSN. Une application WSN est testée en utilisant les paramètres indiqués dans le tableau 4.1. Dans nos ex-

Tableau 4.1 – Les équipements d'implémentation matérielle

Nom	Valeur
Densité d'ancre	3
Densité de noeuds réguliers	25
Superficie	2 m * 2 m
Capacité de transmission	10 cm

périences d'implémentation matérielle, le nombre minimum d'ancres MICAz motes est 3 et les 25 MICAz régulières sont déployées au hasard sur un banc d'essai en grille, comme illustré à la figure 4.2. Le MICAz mote étiqueté avec des autocollants bleus identifie les ancres alors que ceux étiquetés avec des autocollants jaunes identifient les nœuds réguliers.

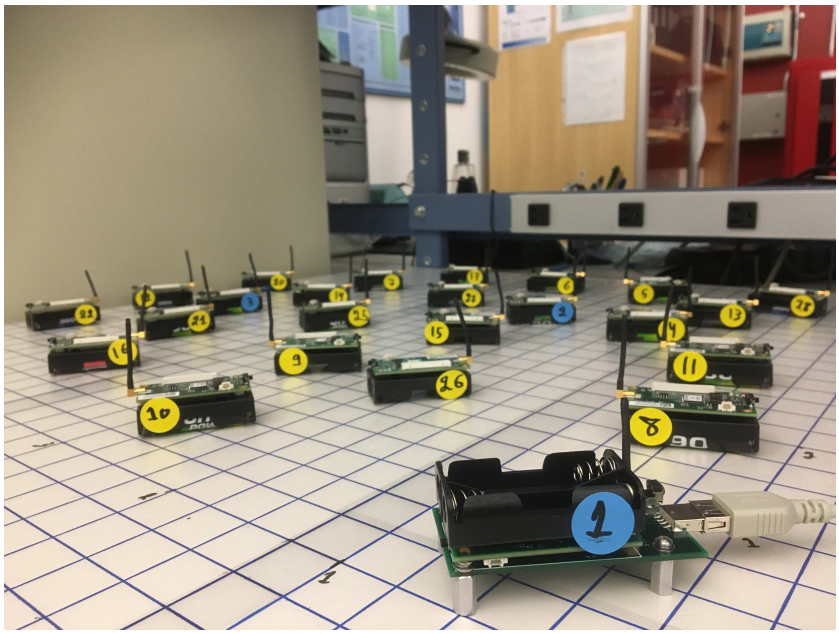

Figure 4.2 – Implémentation matérielle.

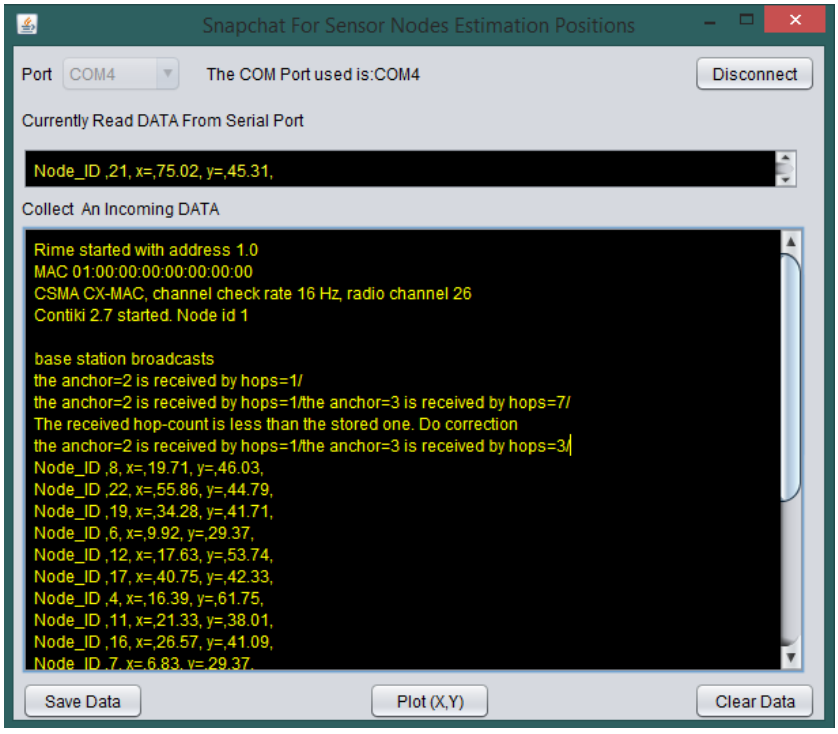

Figure 4.3 – Information sur le réseau.

Comme le montre la figure 4.3, l'interface Java GUI nous montre les détails qui se sont produits après l'implémentation de notre algorithme proposé. Comme donnant le $COM4$ utilisé pour lire les données de la station de base et les afficher sur l'application GUI. Les informations de la station

de base apparaissent sur le texte déposé de couleur jaune, la couche Rim est utilisée sur tous les nœuds. Nous considérons dans cet exercice que le nœud un est la station de base. Le nœud on joue le rôle de nœud d'ancrage et d'autre part, il joue le rôle de la station de base afin de recueillir les informations du réseau. Par exemple, comme le montre la figure 4.3 par le jaune écrit "the anchor 2 is received by hops=1/" ce qui signifie que l'ancre 2 est un espoir loin de l'ancre. Une fois que tous les nœuds d'ancrage du réseau ont diffusé leurs informations, les nœuds réguliers sont en mesure d'estimer leur emplacement et d'envoyer leur estimation à la station de base comme indiqué dans la figure 4.3 $Node_ID, 11, x =, 21.33, y = 38, 01,$.

4.2.5 Protocoles de communication

La communication entre les nœuds prend deux formes. La première forme est le protocole de communication de diffusion où tous les nœuds d'ancrage diffusent leurs informations par paquets sur le réseau et le même protocole de diffusion est appliqué dans tous les nœuds réguliers comme le montre la figure 4.4.

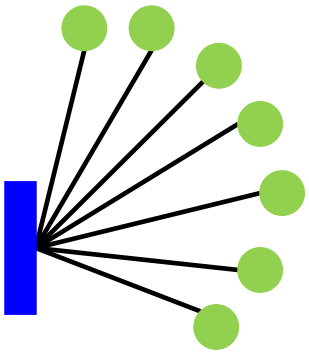

Figure 4.4 – Protocole de diffusion.

Après que les nœuds réguliers calculent leur position en utilisant la méthode de trilatération, tous les nœuds du réseau prennent la deuxième forme pour envoyer ces coordonnées d'estimation [27], à savoir le protocole de communication Multicast, comme le montre la figure 4.5.

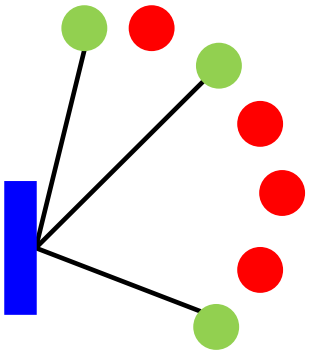

Figure 4.5 – Protocole de communication Multicast.

Le protocole de communication Multicast [2] est une communication de groupe et prend en compte autant de distributions que possible. Alors que la transmission du paquet doit être adressée à un nœud spécifique du réseau, par exemple dans notre algorithme proposé, la transmission du paquet est adressée au nœud qui a l'ID d'identification 1, alors que pendant la première diffusion de l'ID de nœud d'ancrage 1, tous les nœuds ont conservé cette identification (ex: ID = 1) dans leur mémoire. Cependant, l'un des avantages de l'utilisation du protocole de communication Multicast est la réduction de la consommation électrique.

4.2.6 Étapes de débogage

Pour éviter que l'application WSN ne soit hors service, les instructions suivantes doivent être vérifiées :

1. Utiliser les LEDs pour faire le débogage matériel, dans notre travail nous avons utilisé les LEDs comme indiqué dans le tableau:

Tableau 4.2 – Débogage matériel

LED Jaune	utilisé pour montrer la préparation des ancres pour diffuser leurs paquets
LED Verte	utilisé pour montrer si le paquet est reçu
LED Rouge	utilisé pour voir si les nœuds réguliers reçoivent l'information de toutes les ancres du réseau

2. La station de base doit être établie en installant une unité MICAz mote à l'exclusion de la source d'alimentation (batteries) sur la carte d'interface MIB520 mote. En outre, la station de base doit être connectée au PC via le port USB.
3. Pendant le processus de compilation du firmware dans le mote, on choisit un pré-compilateur qui répond aux exigences du système.
4. Pour obtenir une lecture précise du capteur, celui-ci peut être fixé au sol. En effet, la stratégie de notre algorithme est implémentée en 2 dimensions pour obtenir une localisation précise. Ainsi, si le capteur n'est pas complètement fixé, il captera une fausse lecture, ce qui causera une estimation de localisation éloignée par rapport à une estimation précise.
5. Les batteries doivent être alimentées pour transmettre avec succès les signaux dans le réseau. Sinon, le WSN n'a pas la capacité de transmettre un signal à d'autres voisins du WSN.

4.2.7 Résultats de tests sur la version matérielle

La métrique que nous adoptons est l'erreur quadratique moyenne normalisée (NRMSE pour "Normalized Root Mean Square Error") définie comme suit:

$$NRMSE = \frac{1}{N_u} \sum_{i=1}^{N_u} \frac{\sqrt{(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2}}{T_c}. \tag{4.1}$$

La figure 4.6 représente les résultats d'implémentation du monde réel pour les algorithmes de localisation. L'algorithme proposé améliore les performances par rapport aux autres algorithmes sur une implémentation matérielle, comme le montre la figure 4.6 en augmentant le nombre de nœuds d'ancrage, l'erreur diminue, ce qui est attendu car en augmentant les nœuds d'ancrage l'erreur doit être diminuée.

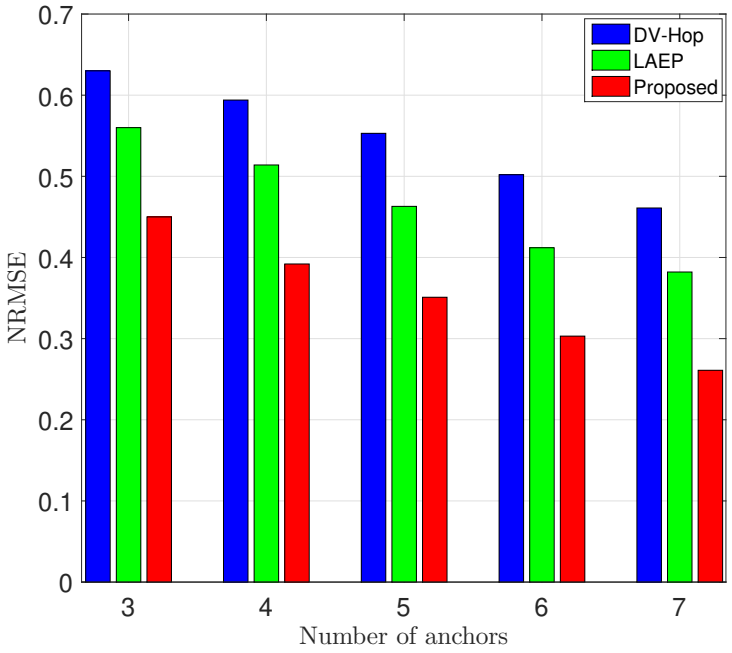

Figure 4.6 – NRMSE en fonction du nombre d'ancres.

Chapitre 5

Résultats de Simulations

5.1 Simulations

Nous évaluons la performance de l'algorithme de localisation proposé par rapport aux meilleures solutions de localisation "range-free" de portée actuellement disponibles dans la littérature. Nous le faisons d'abord par des simulations informatiques avec MATLAB dans la première sous-section suivante, puis par des expériences matérielles avec les motes MICAz et le logiciel Contiki dans une deuxième sous-section.

5.1.1 Résultats de simulation par ordinateur

Notre base de référence de performance est établie par comparaison dans les même conditions de l'algorithme proposé par rapport aux meilleures solutions de localisation "range-free" représentatives rapportées dans la littérature [10]. Dans toutes les simulations, les nœuds sont déployés uniformément dans une zone carrée bidimensionnelle $S = 50 \times 50\ m^2$. La portée de communication de chaque nœud est identique sur l'ensemble du réseau (c'est-à-dire homogène). La métrique que nous adoptons est l'erreur quadratique moyenne normalisée (NRMSE pour "Normalized Root Mean Square Error") définie comme suit:

$$NRMSE = \frac{1}{N_u} \sum_{i=1}^{N_u} \frac{\sqrt{(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2}}{T_c}. \tag{5.1}$$

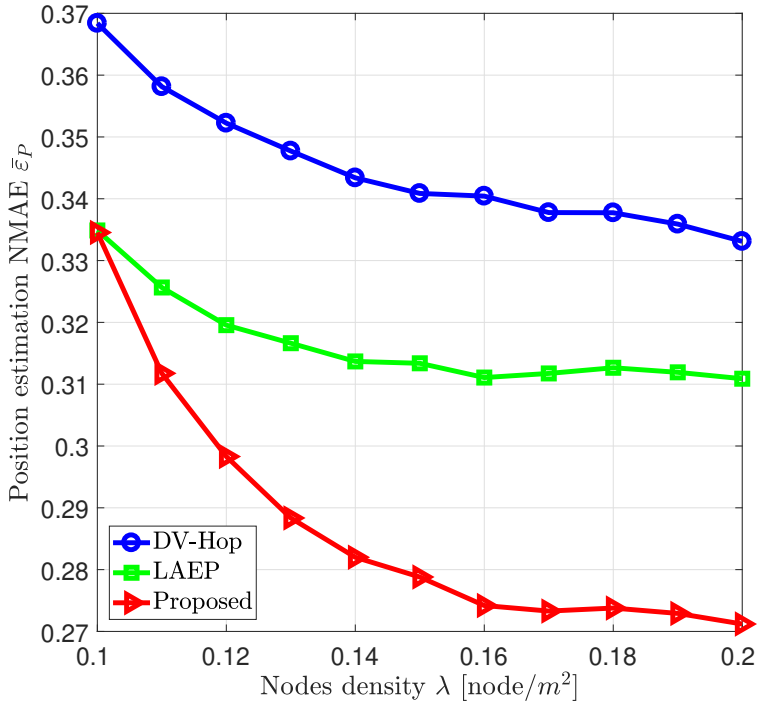

Figure 5.1 – NRMSE en fonction de la densité d'ancres.

La figure 5.1 montre le NRMSE obtenu par DV-Hop, LEAP et l'algorithme proposé pour différentes valeurs de densité de nœuds avec un pourcentage d'ancrage constant de 4% et une plage de communication définie sur 10 m^2. l'algorithme proposé dépasse LEAP et DV-Hop, en particulier à une densité plus élevée de nœuds d'ancrage.

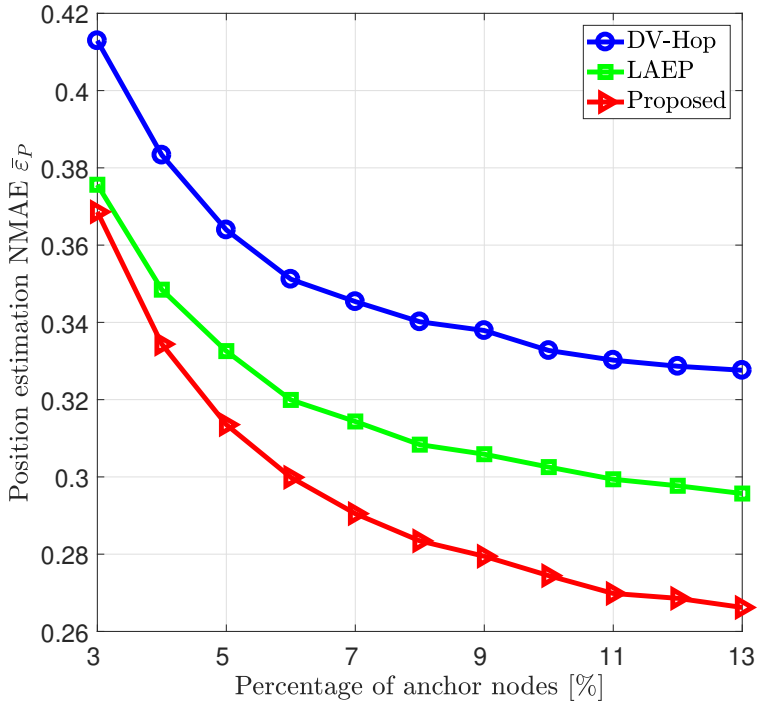

Figure 5.2 – NRMSE par rapport au pourcentage d'ancres.

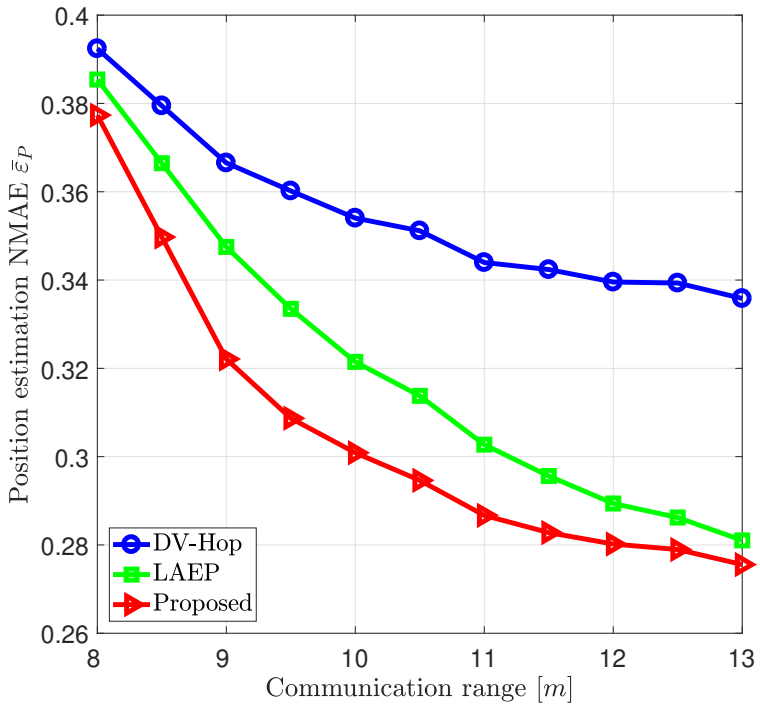

Figure 5.3 – NRMSE par rapport à la portée de communication R.

La figure 5.2 montre l'emplacement NRMSE pour différents pourcentages des nœuds d'ancrage à un nombre constant de 250 nœuds réguliers. Encore une fois, l'algorithme proposé montre une précision plus élevée que DV-Hop et LEAP, en particulier lorsque le pourcentage de nœuds d'ancrage augmente.

La figure 5.3 illustre la localisation NRMSE obtenue par DV-Hop, LEAP et l'algorithme proposé par rapport à la gamme de communication T_c à un nombre constant de 300 nœuds réguliers. l'algorithme proposé donne des résultats spectaculaires par rapport aux autres algorithmes DV-Hop et LEAP.

Conclusion

Cette thèse propose une nouvelle solution de localisation adaptée aux WSN à faible coût qui offre une très haute précision et qui qui permet au nœud régulier d'estimer directement sa distance à tous les nœuds d'ancrage du réseau en s'appuyant uniquement sur les informations disponibles localement. De plus, cette thèse vise à exploiter le système d'exploitation Contiki Cooja pour implémenter matérielle l'algorithme de localisation proposé ainsi que les autres algorithmes sur un noud de capteur (i.e., motes MICAZ) qui sera capable d'exécuter ces algorithmes en temps réel et d'envoyer leur position estimée à la station de base.

Le dernier chapitre introduit les réseaux de capteurs WSNs et présente les principales caractéristiques d'un nœud du WSN.

Chapitre 2 présente un aperçu de la littérature sur les techniques utilisées pour localiser les nœuds de capteurs. Les défis à surmontés pour garantir un algorithme de localisation robuste et précis dans des conditions réelles sont aussi détailles et discutés dans ce chapitre. De plus, un nouvel algorithme de localisation distribué a été dans ce chapitre. La distance entre le capteur régulier et les ancres est localement calcule en exploitent uniquement les informations disponibles au niveau de chaque capteur, réduisant ainsi l'overhead nécessaire à la localisation.

Le troisième chapitre présente le Conception, caractéristiques et la structure de Contiki Cooja. Une attention particulière est portée à la construction d'une application Contiki qui fait référence à l'exécution d'un programme sur un nœud matériel.

Étant donné que l'implémentation matérielle de l'algorithme proposé est délicate et difficile, Le quatrième chapitre décrit en détail les instructions de l'implémentation matérielle des algorithmes de localisation sur les nœuds de capteurs tels que la configuration matérielle, configuration logi-

cielle, téléchargement d'un fichier exécutable, le protocole de communication, la façon de déboguer efficacement notre implémentation sur la plate-forme matérielle, etc.

Chapitre 5 présente les résultats de simulation et de l'implémentation de notre algorithme de localisation proposé ainsi que les autres algorithmes de localisation les plus populaires dans la littérature: DV-hop et LAEP. Il a été démontré que notre algorithme surpasse constamment les meilleurs algorithmes de localisation représentatifs actuellement disponibles dans la littérature en ce qui concerne la précision.

Les nouveautés introduites dans cette thèse sont la précision de notre algorithme de localisation proposé et la façon dont on a utilisé le système d'exploitation Contiki Cooja pour implémenter et valider notre algorithme proposé dans le monde réel en utilisant les motes MICAZ. En plus, la création d'une nouvelle application avancée d'interface utilisateur graphique (GUI) qui permet à l'utilisateur de visualiser les informations reçues des nœuds.

Malgré l'importance des contributions présentées ci-dessus, les travaux réalisés dans cette thèse ont soulevé un certain nombre de problèmes qui mériteraient d'être traitées par la suite. Nous les présentons ici.

En raison de la diffusion simultanée du message d'initialisation de localisation par les ancres, la congestion dans le réseau représente le principal obstacle dans cette étude. Nous avons supposé par simplicité que la diffusion du message d'initialisation de localisation est périodique. Or cette proposition est loin d'être vraie. Par conséquent, d'autres travaux comprenant différentes solutions pour réduire ou éviter la congestion dans le réseau devront être ajoutés à notre étude.

Durant cette étude, nous avons pris en compte uniquement une configuration isotrope. En effet, cette supposition est loin d'être réaliste vu que des obstacles peuvent également existes dans les environnements pratiques. Il est très probable que le chemin le plus court entre une ancre et un capteur soit courbé ce qui entraîne une surestimation de la distance entre ces deux nœuds. Ceci rend évidemment la localisation moins précise. Donc une étude approfondie sur l'effet des obstacles sur la précision de notre algorithme devrait être étudié dans le futur.

Enfin, la performance des algorithmes de de localisation a été évalué dans un milieu statique, où tous les nœuds de capteurs sont supposées d'être fixes. Donc, une étude approfondie sur la modélisation de mobilité des nœuds dans le réseau devrait être abordée dans le futur.

Références

[1] A. Boukerche, H.A.B.F. Oliveira ENAL (2009). “DV-Loc : a scalable localization protocol using Voronoi diagrams for wireless sensor networks”. *IEEE Wireless. Commun. Mag., vol. 16, no. 2, pp. 50-55.*

[2] A. EL Assaf, S. Zaidi SA & Kandil N (2013). “Hop-Count Based Localization Algorithm for Wireless Sensor Networks ”. *IEEE Trans. parallel and distributed Sys., vol. 24, no. 11, pp. 1943-1951.*

[3] A. EL Assaf, S. Zaidi SA & Kandil N (2015). “Cost-Effective and Accurate Nodes Localization in Heterogeneous Wireless Sensor Networks”. *Proc. IEEE ICC'2015, London, United Kingdom.*

[4] Annunziato A (2018). IMWS-5G 2018 Index. *IEEE MTT-S International Microwave Workshop Series on 5G Hardware and System Technologies (IMWS-5G).*

[5] Chor S (2013). "Energy Neutral Routing for energy harvesting Wireless Sensor Networks". *2013 IEEE Wireless Communications and Networking Conference (WCNC).*

[6] D. Ma, MJ. Er BW (2010). “Analysis of Hop-Count-Based Source-to-Destination Distance Estimation in Wireless Sensor Networks With Applications in Localization ”. *IEEE Trans. Veh. Technol., vol. 59, no. 6, pp. 2998-3011.*

[7] Dunkels-B A & Voigt GT (2012). " Contiki - a lightweight and flexible operating system for tiny networked sensors". *Annual IEEE International Conference on Local.*

[8] Farooq MO (2015). "Contiki-Based IEEE 802.15.4 Channel Capacity Estimation and Suitability of Its CSMA-CA MAC Layer Protocol for Real-Time Multimedia Applications". *Mobile Information Systems.*

[9] H. Shen, Z. Ding SD & Zhao C (2014). “Multiple Source Localization in Wireless Sensor Networks Based on Time of Arrival Measurement”. *IEEE Trans. Signal Process., vol. 62, no. 8, pp. 1938-1949.*

[10] J. Rezazadeh, M. Moradi AI & Dutkiewicz E (2014a). “Superior Path Planning Mechanism for Mobile Beacon-Assisted Localization in Wireless Sensor Networks,”. *IEEE Sensors J., vol. 14, no. 9, pp. 3052-3064.*

[11] J. Rezazadeh, M. Moradi AI & Dutkiewicz E (2014b). “Superior Path Planning Mechanism for Mobile Beacon-Assisted Localization in Wireless Sensor Networks ”. *IEEE Sensors J., vol. 14, no. 9, pp. 3052-3064.*

[12] Jafarkhani J (2016). "Sensor Deployment in Heterogeneous Wireless Sensor Networks". *2016 IEEE Global Communications Conference (GLOBECOM).*

[13] Jiannong S & Chen C (2010). "Accurate and Energy-Efficient Range-Free Localization for Mobile Sensor Network". *IEEE Transactions on Mobile Computing.*

[14] Jolly Soparia NB (2014). " A Survey on Comparative Study of Wireless Sensor Network Topologies". *Internal Journal of Computer Applications, pp. 40-43.*

[15] Junguo Wenbin, Li-Dongxu Xiaoming SZ (2010). "Study on improved DV-Hop node localization algorithm in wireless sensor network". *IEEE Conference on Industrial Electronics and Applications.*

[16] Lakafosis V & Tentzeris M (2009). "From single-to multihop : The status of wireless localization ". *IEEE Microw. Mag., vol. 10 , no. 7, pp. 34-41.*

[17] M. Bahrepour, N. Meratnia MPZPH (2010). "Distributed Event Detection in Wireless Sensor Networks for Disaster Management". *International Conference on Intelligence Networking and Collaboration Systems.*

[18] Mohamed Hefeeda MB (2007). "Wireless Sensor Networks for Early Detection of Forest Fires". *2007 IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems.*

[19] Nelson Carlos, Gil Inigo AZ (2010). "Analysis of IEEE 802.15.4 Throughput in Beaconless Mode on micaZ under TinyOS 2". *2010 IEEE 72nd Vehicular Technology Conference - Fall.*

[20] Pham C (2014). "Communication performances of IEEE 802.15.4 wireless sensor motes for data-intensive applications: A comparison of WaspMote, Arduino MEGA, TelosB, MicaZ and iMote2 for image surveillance". *Journal of Network and Computer Applications.*

[21] Radu John SS (2008). On composability of localization protocols for wireless sensor networks. *IEEE Network.*

[22] S. Zaidi, A. EL Assaf SA & Kandil N (2015). "Range-Free Nodes Localization in Mobile Wireless Sensor Networks ". *Proc. IEEE ICUWB'2015 Workshop on V2X Communications: Safety, Automated Driving, and Other Applications, Montreal, QC, Canada.*

[23] Wen JCBGBZG (2016). A 4 dBm IP1dB 20.8 dBm UP3 wideband complementary SF feedback LNTA with derivative superposition method. *IEEE International Symposium on Circuits and Systems (ISCAS).*

[24] Y. Wang XW (2009). "Range-Free Localization Using Expected Hop Progress in Wireless Sensor Networks". *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 20, NO. 10.*

[25] Yue Wang, Feng Zheng MWWXTK (2013). "Reference Selection for Hybrid TOA/RSS Linear Least Squares Localization". *Conferences, 2013 IEEE.*

[26] Zhang Y & w. Guo X (2010). Design and Implementation of Wireless Sensor Network Node Based on .Net Micro Framework. *International Conference on Internet Technology and Applications, Wuhan, pp. 1-3, Available: doi: 10.1109/ITAPP.2010.556629.*

[27] Ziguo Z & He T (2011). "RSD : A Metric for Achieving Range-Free Localization beyond Connectivity". *IEEE Trans. parallel and distributed Sys., vol. 24, no. 11, pp. 1943-1951.*