REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE: Mathématiques et Informatique

DEPARTEMENT: Informatique

N° :……..

DOMAINE : Mathématiques et Informatique

FILIERE : Informatique

OPTION : Réseaux et Technologie de l'Information et de la Communication

Mémoire présenté pour l'obtention

Du diplôme de Master Académique

Par: HADJ HAFSI Mariem

Intitulé

Etude du mécanisme de la collecte de données dans les réseaux de capteurs sans fil avec puits mobiles

Soutenu devant le jury composé de :

……..……	Université de M'sila	Président
M. KHETTAF Abdelouahab	Université de M'sila	Rapporteur
……..……	Université de M'sila	Examinateur

Année universitaire : 2017 /2018

Dédicaces

A mes parents

Je vous dis que ce que je suis aujourd'hui, c'est grâce à votre amour, à votre patience et vos innombrables sacrifices. Que ce modeste travail, soit pour vous une petite compensation et reconnaissance vis à ce que vous avez fait d'incroyable pour moi.

Que Dieu, le tout puissant, vous préserve et vous procure santé et longue vie afin que je puisse à mon tour de vous combler.

A mes très chers frères

Aucune dédicace ne serait exprimée assez profondément ce que je ressens vis à vous, je vous dirais tout simplement un grand merci, je vous aime.

A mes très chères amies

En témoignage de l'amitié sincère qui nous a liés et des bons moments passés ensemble. Je vous dédie ce travail en vous souhaitant un avenir radieux et plein de bonnes promesses.

Hadj Hafsi Mariem

Remerciements

Tout d'abord, je remercie Allah le tout puissant, à la sagesse et au savoir infinis, « ""Gloire à toi ! Nous n'avons de savoir que ce que Tu nous as appris. Certes c'est Toi l'Omniscient, le sage, le tout miséricordieux le très miséricordieux ""» (Sourate al-Baqarah, verset 32).

Je tien à remercier mon encadreur Mr A. khettaf pour le grand honneur qu'il m'a fait en me proposant le sujet de ce mémoire de fin d'étude. J'ai eu l'honneur et le privilège de travailler sous son assistance et de profiter de ses qualités humaines, professionnelles et de sa grande expérience, il m'a guidé tout au long de ce travail. L'élaboration avec amabilité et dynamisme le caractérisant. Que ce modeste travail puisse satisfaire mes examinateurs, pour qu'ils en témoignent ma gratitude et reconnaissance pour l'aide et les conseils qu'il m'a prodigué, ainsi que pour le savoir qu'il m'a inculqué.

Je remercie tous mes enseignants de l'université de M'sila.

Mes remerciements vont également aux membres de jury d'avoir accepté de juger mon travail.

Je remercie vivement toute ma famille, en particulier mes parents, pour m'avoir toujours soutenu au cours de mes études. Qu'ils trouvent ici le fruit de leur patience et du soutien permanent qu'ils m'ont prodigué pour affronter tous les moments difficiles.

Je tien également à remercier mes collègues pour leur

TABLE DES MATIERES

CHAPITRE 02: ALGORITHMES DE LA COLLECTE DE DONNEES DANS RCSF AVEC PUITS MOBILES

CHAPITRE 03: IMPLEMENTATION ET SIMULATION

LISTE DES FIGURES

CHAPITRE 01: LES RESEAUX DE CAPTEURS SANS FIL (RCSF)

CHAPITRE 02: ALGORITHMES DE LA COLLECTE DE DONNEES DANS RCSF AVEC PUITS MOBILES

CHAPITRE 03 : IMPLEMENTATION ET SIMULATION

LISTE DES TABLEAUX

CHAPITRE 03 : IMPLEMENTATION ET SIMULATION

INTRODUCTION GENERAL

Depuis quelques décennies, le besoin d'observer et de contrôler les phénomènes physiques tels que la température, la pression ou encore la luminosité est essentiel pour de nombreuses applications industrielles et scientifiques. Par exemple, dans le domaine de l'écologie, la surveillance de polluants comme l'Ozone, le NO2 ou encore le CO2, pourrait considérablement augmenter la qualité de vie dans les villes [1, 2, 3, 4].

Les progrès technologiques dans les domaines de la microélectronique, des communications sans fil, couplé aux efforts de miniaturisation et de réduction des coûts de production des composants électroniques, ont permis le développement de nouvelles générations de petits appareils électroniques, autonomes, équipés de capteurs et capables de détecter, de calculer, de stocker et de communiquer entre eux sans fil. Ces petits dispositifs sont appelés des nœuds capteurs ou « motes ». Ensemble, ils forment un réseau appelé réseau de capteurs sans fil (RCSF) qui est capable de superviser une région ou un phénomène dans une zone d'intérêt, de fournir des informations utiles par la combinaison des mesures prises par les différents capteurs [5].

Les capteurs sont, généralement, dispersés aléatoirement dans une zone géographique, appelée champ de captage. Les données détectées sont acheminées grâce à un routage à un nœud considéré comme un point de collecte, appelé station de base ou puits. Ce dernier peut être connecté à l'utilisateur du réseau via Internet ou un satellite. Ainsi, l'utilisateur peut adresser des requêtes aux autres nœuds du réseau, en précisant le type de données requises et récolter les données captées par le biais de la station de base.

Le paradigme de communication dans ce type de réseau est du type N-vers-1, où les capteurs collectent des données de l'environnement qui les entoure (i.e., température, pression, ...), et les disséminent vers un point central, appelé ´également un puits. Ce puits a pour principal objectif la collecte des différentes données générées par les capteurs.

Plusieurs ´études supposent que ce puits est statique ou immobile. Cependant, cette hypothèse n'est pas toujours réalisable notamment pour des raisons de sécurité ou de déploiement. De plus, la mobilité de puits permet d'augmenter la durée de vie du réseau en ´évitant de surcharger les nœuds présents dans le voisinage de puits.

Nous envisageons, à travers ce travail, de s'initier au domaine des réseaux de capteurs sans fil avec puits mobiles. Principalement, nous allons faire une étude sur le problème de

collection de données dans RCSF. Le but de cette étude est d'implémenter et d'évaluer une méthode de collection des données détectée par les nœuds capteurs dans le réseau.

A cet effet, nous avons organisé notre travail comme suit :

- Le chapitre 1 sera consacré à la description de l'architecture d'un capteur et la présentation du principe de fonctionnement des RCSF, ainsi que des ces caractéristiques et ces domaines d'application.

- Chapitre 2 traitera la problématique de la collection des données dans les réseaux de capteurs. Nous allons présenter les différents algorithmes existants pour collecter les informations d'environnement en utilisant un puits mobile qui se déplace sur certain nœuds PR (point de rendez-vous), et en fin nous allons faire une étude comparative relative à ces algorithmes de collection des données en basant sur certain métriques.

- Chapitre 3 exposera notre travail, qui sera décrit en trois parties :

  - Présentation de l'environnement d'implémentation Network Simulator (NS2), et leurs objets et composants.

  - Conception et implémentation de notre travail en utilisant le simulateur NS2.

  - Nous allons présenter les résultats de simulation, avec l'interprétation et évaluation des performances de notre travail.

Nous terminons notre étude par une conclusion générale y compris les perspectives de recherche de ce travail.

CHAPITRE 01

LES RESEAUX DE CAPTEURS SANS FIL (RCSF)

1. Introduction

Les réseaux de capteurs sans fil (RCSF) représentent un domaine de forte expansion, surtout dans les domaines de l'instrumentation, des économies d'énergie et de la gestion de l'environnement. D'ailleurs, en 2003 le magazine américain Technology Review a considéré que le réseau de capteurs sans fil est l'une des dix nouvelles technologies qui bouleverseront le monde et notre manière de vivre et de travailler.

Leur importance provient aussi de leurs nombreux domaines d'application tels que la détection et la surveillance des désastres, le contrôle de l'environnement et la cartographie de la biodiversité, le bâtiment intelligent, l'agriculture, la surveillance et la maintenance préventive des machines, la médecine et la santé, la logistique et les transports intelligents, etc.

Dans ce chapitre, nous décrivons les réseaux de capteurs, leurs caractéristiques ainsi que leurs domaines d'applications.

2. Qu'est-ce qu'un capteur sans fil

Un capteur sans fil est un petit dispositif électronique capable de mesurer une valeur physique environnementale (température, lumière, pression, etc.) et de la transmettre à un centre de contrôle via une station de base. Les progrès conjoints de la microélectronique, des technologies de transmission sans fil et des applications logicielles ont permis de produire des micro-capteurs de quelques millimètres cubes de volume, susceptibles de fonctionner en réseaux [1]. Un capteur est composé de quatre unités de base (voir figure 1.1):

2.1. L'unité d'acquisition

Elle est généralement composée de deux sous-unités qui sont les capteurs et les convertisseurs analogique-numérique ADC (Analog Digital Converter). Les capteurs obtiennent des mesures sur les paramètres environnementaux et les transforment en signaux analogiques. Les ADCs convertissent ces signaux analogiques en signaux numériques.

2.2. L'unité de traitement

Elle est composée de deux interfaces qui sont une interface avec l'unité d'acquisition et une autre avec le module de transmission. Elle contrôle les procédures permettant au nœud de collaborer avec les autres nœuds pour réaliser les tâches d'acquisition et de stocker les données collectées.

2.3. Un module de communication (Transceiver)

Il est composé d'un émetteur/récepteur permettant la communication entre les différents nœuds du réseau via un support de communication radio.

2.4. Batterie

Elle alimente les unités que nous avons citées et elle n'est généralement ni rechargeable ni remplaçable. La capacité d'énergie limitée au niveau des capteurs représente la contrainte principale lors de conception de protocoles pour les réseaux de capteurs.

Il existe des capteurs qui sont dotés d'autres composants additionnels tels que les systèmes de localisation GPS (Global Position System).

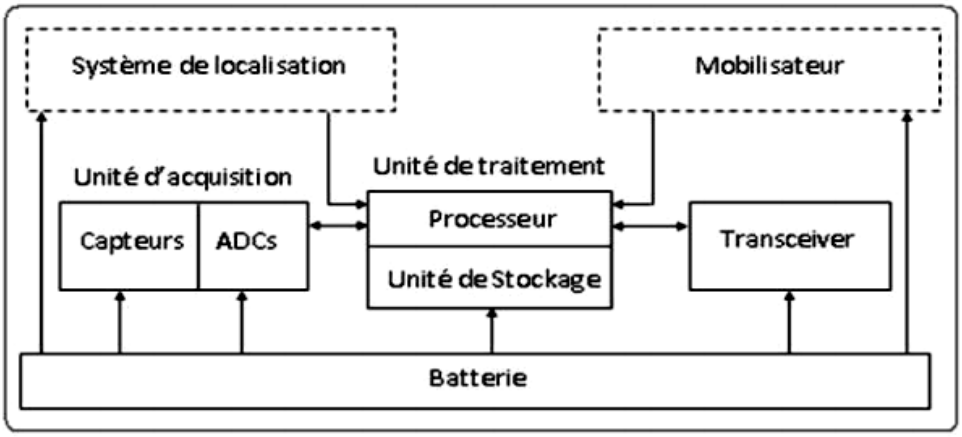


Figure 1. 1 Les composants d'un nœud capteur

3. Quelques applications des réseaux de capteurs

La diminution de taille et de coût des micro-capteurs, l'élargissement de la gamme des types de capteurs disponibles (thermique, optique, vibrations, ...) ansi l'évolution des supports de communication sans fil ont élargi le champ d'application des réseaux de capteurs. Les RCSF peuvent être appliqués en plusieurs domaines [2, 3, 4, 5]. Parmi elles, nous citons :

3.1. Découverte de catastrophes naturelles

On peut créer un réseau autonome en dispersant les nœuds dans la nature. Des capteurs peuvent ainsi signaler des événements tels que les feux de forêts, les tempêtes ou les inondations. Ceci permet une intervention beaucoup plus rapide et efficace des secours [6].

3.2. Détection d'intrusions

En plaçant à différents points stratégiques des capteurs, on peut ainsi prévenir des cambriolages ou des passages de gibier sur une voie de chemin de fer (par exemple) sans avoir recourir à des dispositifs coûteux de surveillance vidéo.

3.3. Gestion de stock

On pourrait imaginer de stocker des denrées nécessitant un certain taux d'humidité et une certaine température. Dans ces applications, le réseau doit pouvoir collecter ces différentes informations et alerter en temps réel si les seuils critiques sont dépassés.

3.4. Contrôle de la pollution

Des capteurs placés au-dessus d'un emplacement industriel offrent la possibilité de détecter et de contrôler des fuites de gaz ou de produits chimiques. Ces applications permettent de donner l'alerte en un temps record et de pouvoir suivre l'évolution de la catastrophe [7].

3.5. Agriculture

Des nœuds peuvent être incorporés dans la terre et on peut interroger le réseau sur l'état du champ et déterminer par exemple les secteurs les plus secs afin de les arroser en priorité. On peut aussi imaginer équiper des troupeaux de bétail de capteurs pour connaître en tout temps, leur position ce qui permet aux éleveurs d'éviter d'avoir recours à des chiens de berger.

3.6. Surveillance médicale

En implantant sous la peau de mini capteurs vidéo, on peut recevoir des images d'une partie du corps en temps réel sans aucune chirurgie. On peut ainsi surveiller la progression d'une maladie ou la reconstruction d'un muscle [8].

3.7. Surveillance de barrages

On peut inclure sur les parois des barrages des capteurs qui permettent de calculer en temps réel la pression exercée. Il est donc possible de régler le niveau d'eau si les limites sont atteintes. On peut aussi imaginer inclure des capteurs entre les sacs de sables formant une digue de fortune. La détection rapide d'infiltration d'eau peut servir à renforcer le barrage en conséquence. Cette technique peut aussi être utilisée pour d'autres constructions tels que ponts, voies de chemins de fer, routes de montagnes, bâtiments et autres ouvrages d'art.

4. Architecture d'un RCSF

Tous les capteurs respectent globalement la même architecture basée sur un noyau central autour duquel s'articulent les différentes interfaces d'entrée-sortie, de communication et d'alimentation [9, 10]. La figure 1.2 montre un exemple d'un réseau de capteurs.

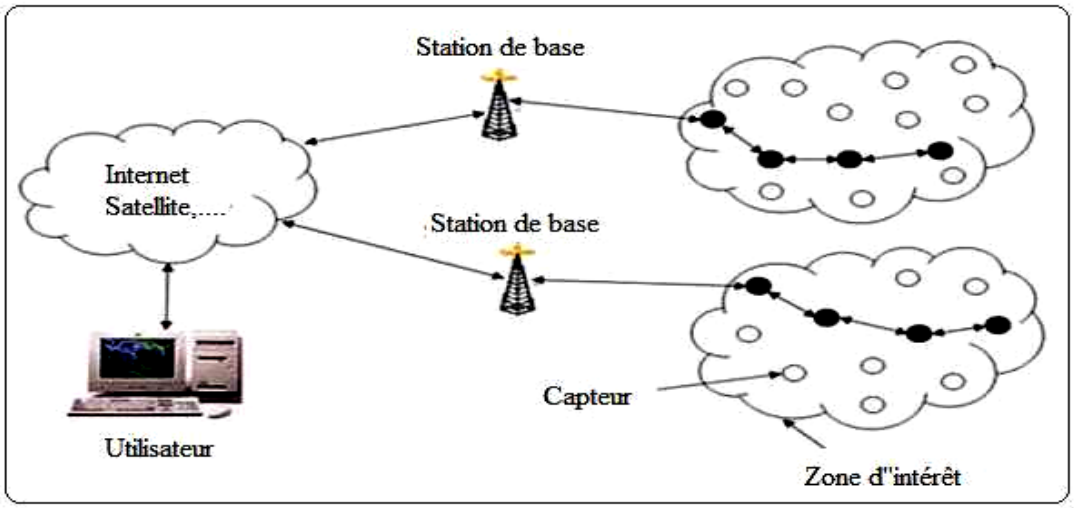


Figure 1. 2 Exemple de réseaux de capteurs

Un RCSF est composé d'un ensemble de nœuds capteurs qui sont organisés en champs «Sensor Fields». Chacun de ces nœuds a la capacité de collecter des données et de les transférer au nœud passerelle par l'intermédiaire d'une architecture multi-sauts. Le nœud passerelle transmet ensuite ces données par Internet ou par satellite à l'ordinateur central «Gestionnaire de tâches» pour analyser ces données et prendre des décisions.

5. Contraintes de conception des RCSF

Les principaux facteurs et contraintes influençant l'architecture des réseaux de capteurs peuvent être résumés comme suit [1]:

5.1. La tolérance aux fautes

La tolérance aux fautes est la capacité de maintenir les fonctionnalités du réseau en présence de fautes. La fiabilité des réseaux de capteurs sans fil est affectée par des défauts qui se produisent à cause de diverses raisons telles que le mauvais fonctionnement du matériel ou à cause d'un manque d'énergie. Ces problèmes n'affectent pas le reste du réseau.

5.2. Le facteur d'échelle (Scalability) :

Le nombre de nœuds capteurs augmente sur un réseau sans fil et ce nombre peut atteindre le million. Un nombre aussi important de nœuds engendre beaucoup de transmissions entre les nœuds et peut imposer des difficultés pour le transfert de données.

5.3. Les coûts de production

Souvent les réseaux de capteurs sont composés d'un très grand nombre de nœuds. Le prix d'un nœud est critique afin de pouvoir concurrencer un réseau de surveillance traditionnel.

5.4. L'environnement

Les capteurs sont souvent déployés en masse dans des endroits tels que des champs de bataille, à l'intérieur de grandes machines, au fond d'un océan, dans des champs biologiquement ou chimiquement souillés [14],... Par conséquent, ils doivent pouvoir fonctionner sans surveillance dans des régions géographiques éloignées.

5.5. La topologie de réseau

Le déploiement d'un grand nombre de nœuds nécessite une maintenance de la topologie. Cette maintenance consiste en trois phases : déploiement, post-déploiement (les capteurs peuvent bouger, ne plus fonctionner,...) et redéploiement de nœuds additionnels.

5.6. Les contraintes matérielles

La principale contrainte matérielle est la taille de capteur. Les autres contraintes sont la consommation d'énergie qui doit être moindre pour que le réseau survive le plus longtemps possible, qu'il s'adapte aux différents environnements (fortes chaleurs, eau,..), qu'il soit autonome et très résistant vu qu'il est souvent déployé dans des environnements hostiles.

5.7. Les médias de transmission

Dans un réseau de capteurs, les nœuds sont reliés par une architecture sans fil. Pour permettre des opérations sur ces réseaux dans le monde entier, le média de transmission doit être standardisé. On utilise le plus souvent l'infrarouge, le Bluetooth [17] et les communications radio Zig Bee [18].

5.8. La consommation d'énergie

Un capteur, à part sa taille, est limité en énergie (< 1.2 V). Dans la plupart des cas le remplacement de la batterie est impossible. Ce qui veut dire que la durée de vie d'un capteur dépend grandement de la durée de vie de la batterie. Dans un réseau de capteurs (multi-sauts) chaque nœuds collecte des données et envoie/transmet des valeurs. Le dysfonctionnement de quelques nœuds nécessite un changement de la topologie du réseau et un ré-routage des paquets. Toutes ces opérations sont gourmandes en énergie, c'est pour cette raison que les recherches actuelles se concentrent principalement sur les moyens de réduction de cette consommation [20].

6. Consommation d'énergie dans les RCSFs

La première étape dans la conception de système énergétique de capteurs consiste à analyser les caractéristiques de consommation d'énergie d'un nœud capteur sans fil. Cette analyse systématique de l'énergie d'un nœud capteur est extrêmement importante pour identifier les problèmes dans le système énergétique afin de permettre une optimisation efficace. L'énergie consommée par un capteur est principalement due aux opérations suivantes : la détection, le traitement et la communication [29].

6.1. Energie de capture

Les sources de consommation d'énergie des nœuds par rapport aux opérations de détection ou de capture sont : l'échantillonnage, la conversion analogique-numérique, le traitement de signal et l'activation de la sonde de capture [24].

6.2. Energie de traitement

L'énergie de traitement est composée de deux sortes d'énergie: l'énergie de commutation et l'énergie de fuite. L'énergie de commutation est déterminée par la tension d'alimentation et la capacité totale commutée au niveau logiciel (en exécutant un logiciel). Par contre, l'énergie de fuite correspond à l'énergie consommée lorsque l'unité de calcul n'effectue aucun traitement. En général, l'énergie de traitement est faible par rapport à celle nécessaire pour la communication.

6.3. Energie de communication

L'énergie de communication se décline en trois parties: l'énergie de réception, l'énergie de l'émission et l'énergie en état de veille. Cette énergie est déterminée par la quantité des données à communiquer et la distance de transmission, ainsi que par les propriétés physiques du module radio. L'émission d'un signal est caractérisée par sa puissance, quand la puissance d'émission est élevée le signal aura une grande portée et l'énergie consommée sera plus élevée. Notons que l'énergie de communication représente la portion la plus grande de l'énergie consommée par un nœud capteur.

7. Techniques de minimisation de la consommation d'énergie

Dans les réseaux ad hoc, la consommation de l'énergie a été considérée comme un facteur déterminant mais pas primordial car les ressources d'énergie peuvent être remplacées par l'utilisateur.

Ces réseaux se focalisent plus sur la qualité de service QoS (Quality of Service) que sur la consommation de l'énergie. Par contre, dans les réseaux de capteurs, la consommation d'énergie est très importante puisque généralement les capteurs sont déployés dans des zones inaccessibles. Ainsi, il est difficile voire impossible de remplacer les batteries après leur épuisement.

De ce fait, la consommation d'énergie au niveau des capteurs a une grande influence sur la durée de vie du réseau. Après la description des principales causes de consommation d'énergie dans les RCSF, nous présentons dans ce qui suit les différentes techniques utilisées pour minimiser cette consommation. Ces techniques sont appliquées soit au niveau de la couche liaison soit au niveau de la couche réseau. Le schéma suivant donne un aperçu global de ces mécanismes :

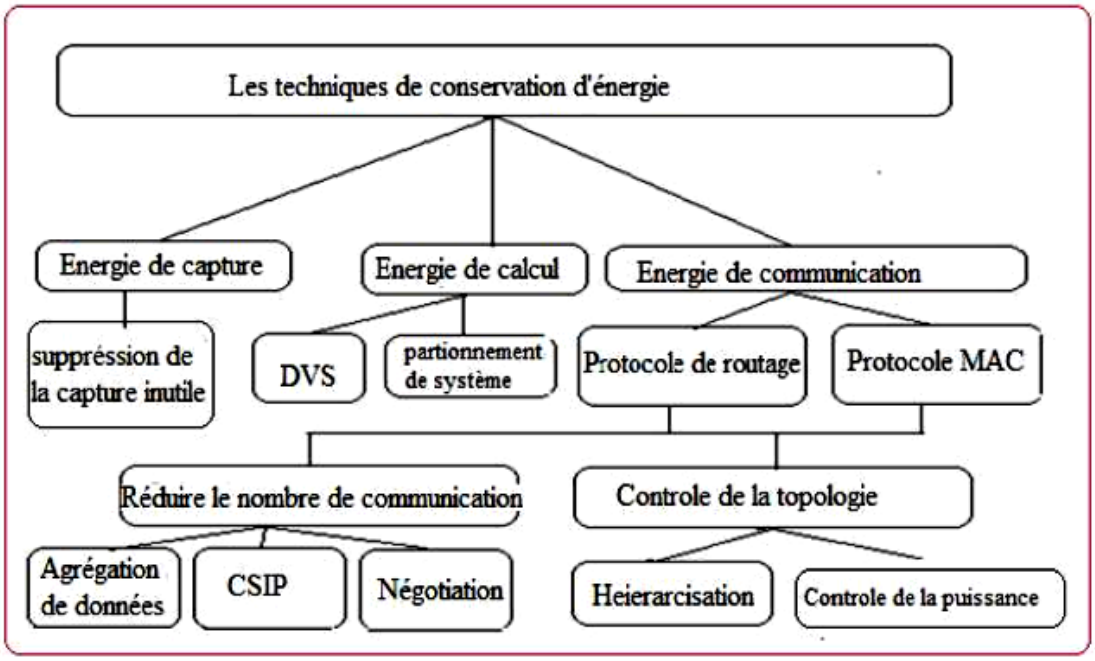


Figure 1. 3 Les techniques de conservation d'énergie

L'énergie du capteur peut être économisée soit au niveau de la capture, au niveau de traitement ou au niveau de la communication.

A. La seule solution apportée pour la minimisation de la consommation d'énergie au niveau de la capture consiste à réduire les fréquences et les durées de captures.

B. L'énergie de calcul peut être optimisée en utilisant deux techniques :

- L'approche DVS (Dynamique Voltage Scaling) [25] qui consiste à ajuster de manière adaptative la tension d'alimentation et la fréquence du microprocesseur pour économiser la puissance de calcul sans dégradation des performances.
- L'approche de partitionnement de système qui consiste à transférer un calcul prohibitif en temps de calcul vers une station de base qui n'a pas de contraintes énergétiques et qui possède une grande capacité de calcul [26].

C. La minimisation de la consommation d'énergie pendant la communication est étroitement liée aux protocoles développés pour la couche réseau et la sous-couche MAC. Ces protocoles se basent sur plusieurs techniques : l'agrégation de données, la négociation et la technique CSIP (Collaborative Signal and Information Processing). Cette dernière technique est une discipline qui combine plusieurs domaines [27] : la communication et le calcul à basse puissance, le traitement de signal, les algorithmes distribués, la tolérance aux fautes, les systèmes adaptatifs et la théorie de fusion des capteurs et des décisions.

Ces techniques ont le but de réduire le nombre d'émission/ réception des messages.

8. Les critères de performance des protocoles de routage en RCSF

La performance des réseaux de capteurs sans fil est fondée sur les facteurs suivants:

8.1. Evolutivité

L'évolutivité est un facteur important dans les réseaux de capteurs sans fil. Une zone de réseau n'est pas toujours statique, elle change selon les besoins des utilisateurs. Tous les nœuds dans le domaine du réseau doivent être évolutifs ou ils doivent être en mesure de s'adapter aux changements dans la structure de réseau en fonction de l'utilisateur.

8.2. L'énergie

Chaque nœud utilise peu d'énergie pour des activités telles que la détection, le traitement, le stockage et la transmission. Un nœud dans le réseau doit savoir combien d'énergie sera utilisée pour effectuer une nouvelle tâche à laquelle il est soumis. L'énergie consommée peut varier selon le type de fonctionnalité ou l'activité qu'il est chargé à accomplir.

8.3. Le temps de traitement

Il se réfère au temps pris par le nœud dans le réseau pour assurer l'ensemble d'opérations commençant par la détection, le traitement des données ou le stockage de données, la transmission ou la réception sur le réseau.

8.4. Le schéma de transmission

La transmission de données par les nœuds capteurs vers la destination ou la station de base se fait par un schéma de routage à un seul saut ou à multi sauts.

8.5. La capacité du réseau

Tous les nœuds du réseau de capteurs utilisent certaines ressources du réseau qui les aident à accomplir certaines activités comme la détection ou la transformation.

8.6. Synchronisation

Dans les communications radio entre les nœuds capteurs d'un RCSF, les capteurs écoutent en permanence les transmissions et consomment de l'énergie s'ils ne sont pas synchronisés les uns les autres. Pour cela, un nœud doit avoir la même notion de temps pour se mettre en veille et se réveiller que ses voisins.

8.7. Contrôle de paquets

Un paquet envoyé avant une transmission entre deux nœuds est appelé le paquet de contrôle. Le paquet de contrôle contient le nombre de bits de données envoyés, l'adresse de nœud de destination et certaines informations qui contribuent à éviter les collisions pendant la transmission.

9. Les principaux protocoles de routage dans les RCSF

9.1. Les Protocoles hiérarchiques

L'objectif principal du routage hiérarchique [24, 23] est de maintenir efficacement la consommation d'énergie de nœuds capteurs en les impliquant dans la communication multi-sauts au sein d'un cluster et en effectuant l'agrégation et la fusion des données afin de diminuer le nombre de messages transmis à la destination.

La formation de clusters est généralement fondée sur la réserve d'énergie des capteurs et sur les capteurs qui sont à proximité de cluster-Head (voir figure 1.4). LEACH (Low Energy Adaptive Clustering Hierarchical) [21] est l'une de premières approches de routage pour les réseaux de capteurs. L'idée proposée par LEACH a été une inspiration pour de nombreux protocoles de routage hiérarchique, bien que certains protocoles a été développés de manière indépendante.

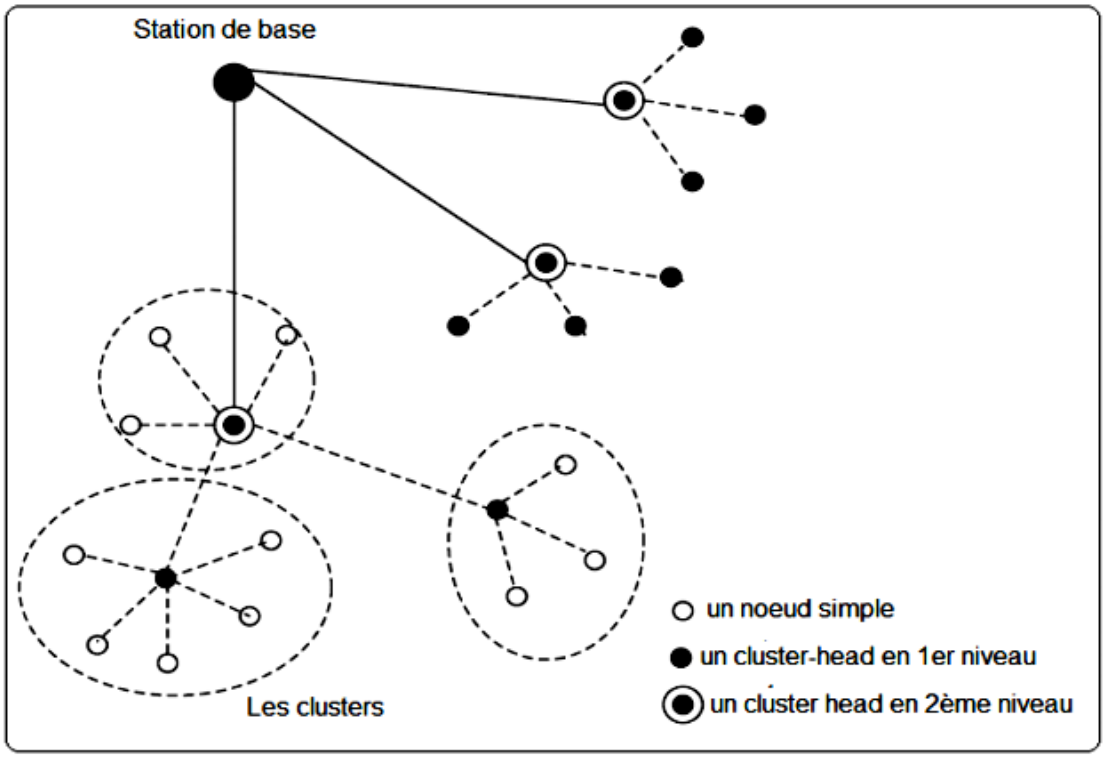


Figure 1. 4 Topologie hiérarchique

9.2. Les protocoles de routage basés sur la localisation

Les protocoles de routage basés sur la localisation [22] utilisent les informations d'emplacement pour guider la découverte de routage et la transmission des données. Ils permettent la transmission directionnelle de l'information en évitant l'inondation d'information dans l'ensemble du réseau. Par conséquent, le coût de contrôle de l'algorithme est réduit et le routage est optimisé. De plus, avec la topologie réseau basée sur des informations de localisation de nœuds, la gestion du réseau devient simple. L'inconvénient de ces protocoles de routage est que chaque nœud doit connaître les emplacements des autres nœuds.

9.3. Les protocoles de routage 'data-centric'

Dans de nombreuses applications de réseaux de capteurs, vu le nombre élevé de nœuds déployés, il n'est pas possible d'attribuer des identificateurs globaux à chaque nœud. Cette absence d'identification globale avec le déploiement aléatoire de nœuds capteurs font qu'il est difficile de sélectionner un ensemble spécifique de nœuds capteurs à interroger. Par conséquent, les données sont généralement transmises de chaque nœud capteurs dans la région de déploiement avec une redondance importante. Cette réflexion a conduit au routage data-centric [28] qui est différent du traditionnel routage où les routes sont créées entre les nœuds adressables gérée dans la couche réseau. Le destinataire envoie des requêtes à certaines régions et attend à recevoir des données provenant des capteurs situés dans les régions sélectionnées.

Comme les données sont demandées à travers des requêtes, le nommage est nécessaire pour préciser les propriétés des données [22].

10. Conclusion

Dans la première partie de ce chapitre, nous avons défini un réseau de capteurs sans fil comme nous l'avons considéré comme un type particulier de réseau ad hoc. Puis, nous avons décrit brièvement un réseau de capteur, ses domaines d'applications, son architecture, ses principales contraintes de conception et la consommation d'énergie.

Dans la deuxième partie de ce chapitre, nous avons mis l'accent sur les principaux protocoles de routage dans les réseaux de capteurs. Nous avons résumé les protocoles de routage dans les réseaux de capteurs. Enfin, nous avons classé les approches en trois catégories principales : les protocoles hiérarchiques, les protocoles basés sur la localisation et les protocoles data-centric.

Dans le prochain chapitre, nous présentons certains algorithmes de La collecte de données dans RCSF Avec puits mobiles.

CHAPITRE : 02

ALGORITHMES DE LA COLLECTE DE DONNEES DANS RCSF AVEC PUITS MOBILES

1. Introduction

Les réseaux de capteurs sans fil (RCSF) sont composés d'un grand nombre de nœuds capteurs déployés dans un terrain. Ils ont des applications variées de grande utilisation, dont certaines comprennent des surveillances de l'environnement agriculture, domotique, transport intelligent, militaire et santé.

Chaque nœud capteur ayant une capacité de collecter et de traiter les données, et de transmettre toutes les données détectées à un ou plusieurs puits nœuds via leur émetteur-récepteur sans fil d'une manière multi-sauts. En plus, il est équipé d'une batterie, ce qui peut être difficile ou impossible à remplacer, tenu en compte le nombre des nœuds capteurs et environnement déployé. Ces contraintes ont conduit à une intensification des efforts de recherche pour concevoir des protocoles économes en énergie.

Dans les communications multi saut, les nœuds qui sont proches au puits, prendre la responsabilité de transmettre les données des nœuds les plus éloignés vers le puits. Ainsi, plus le capteur est proche au puits, plus sa batterie s'épuise vite, alors que ceux les plus loin peut maintenir plus de 90% de leur énergie initiale Cela conduit à l'épuisement non uniforme de l'énergie, qui entraîne la partition du réseau en raison de la formation de trous d'énergie En conséquence, si un puits déconnecte les autres nœuds, ce qui affecte le RCSF.

Ces puits mobiles surveillent et collectent les données détectées directement à partir des nœuds et aident automatiquement ces nœuds pour sauver à économiser l'énergie qui autrement serait consommée par des communications multi-saut, le chemin de transmission de puits mobile dépend du temps réel de donnés détecté par les nœuds sources, les données environnementales doivent être collectées par un puits mobile rapidement.

De plus un puits mobile peut changer sa position après un certain temps pour sélectionner et collecter autres données détectées dans tout le réseau, pour faire cette opération le puits mobile nécessite de sélectionner des nœuds pour collecter des données, par conséquent, il aidera à équilibrer la consommation d'énergie.

Dans ce chapitre, nous discutons comment cette opération fonctionne, en présentant trois algorithmes de collection de données dans RCSF avec puits mobiles qui sont

- Conception de rendez-vous avec un chemin variable de station de base. RD-VT (Rendez-vous Design with a Variable BS Track)
- Planification du Rendez-vous basé sur l'utilitaire gourmand, RP-UG (rendez-vous planning utility-based greedy).

- Planification du Rendez-vous au poids, WPR (weighted rendez-vous planning)

Apres nous allons faire une étude comparative et évaluation de la performance sur ces trois Algorithmes.

2. Les algorithmes de la collecte de données dans RCSF avec puits mobiles

2.1. Conception de rendez-vous avec un chemin variable de station de base

Dans cette section, nous étudions le problème de conception de rendez-vous lorsque la station de base peut se déplacer librement dans le déploiement du réseau le long d'une route minimum [30]. Notre objectif est de trouver une tour plus long que L et l'ensemble des chemins qui est liés à tous les sources, de telle sorte que la longueur euclidienne totale des chemins est minimisée, le problème est formé comme suit :

Étant donné un ensemble des sources S situées sur un plan 2D, [31] trouvez un tour U pas plus de L et un ensemble d'arborescence géométriques $\{T_i (V_i, E_i)\}$ qui sont liées sur U et $S \subseteq \cup_i V_i$; tel que P_i P (u, v) $\in E_i$ | uv |, est minimisé, où (u, v) est un arc de l'arbre T_i, et | uv | est sa longueur euclidienne [32].

- Un exemple de la solution est illustré sur la figure 2.1 (b), Les nœuds sources et PR sont désignés par des cercles blancs et noirs, respectivement. Le puits mobile visite les trois nœuds PR (point de rendez-vous) :PR1, PR2 (qui sont aussi des nœuds sources) et RP3. Deux arborescences sont liés dans PR1 et PR3 et se connectent toutes les sources.

L'objectif est de minimiser la longueur totale des arêtes des deux arbres. On peut montrer que ce problème est traveling salesman problem (TSP) [33]. Plus précisément, un cas particulier de la version de décision du problème est de demander s'il existe un ensemble de PR tels que le la consommation d'énergie du réseau est nulle. Afin de mettre la consommation d'énergie du réseau zéro, toutes les sources doivent être PR. En d'autres termes, la station de base doit visiter tous les PR lors d'une tournée pas plus que L. Ceci est exactement la version de décision de la Problème GTSP dans lequel un vendeur doit visiter un ensemble de sites sur une tournée en ne dépassant pas une limite donnée.

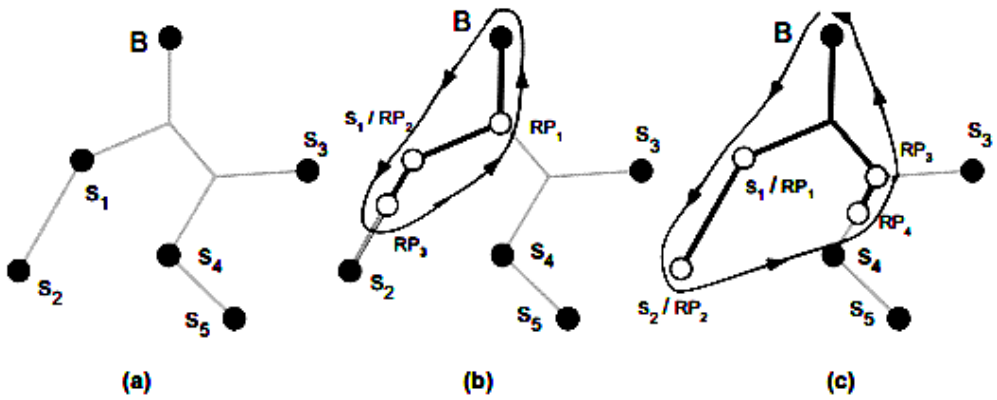


Figure 2. 1 Un exemple de l'exécution de l'algorithme RD-VT (Rendez-vous Design with a Variable BS Track).

2.2. Planification du Rendez-vous basé sur l'utilitaire gourmand

Nous passons maintenant notre attention à la planification de rendez-vous lorsque le mouvement des éléments mobile ME (mobile éléments) n'est pas obligatoirement l'arborescence de routage. Dans cette section nous présentons un algorithme glouton appelé RP-UG (rendez-vous planning utility-based greedy).

RP-UG fonctionne en itérations [34]. [35] Dans chaque itération, le tour de ME actuel est élargi en incluant un nouveau PR avec la plus grande utilité jusqu'à ce que la durée maximale de la visite soit atteinte. L'utilité d'un PR est définie comme le rapport de l'énergie du réseau économisé en l'incluant sur le tour de ME, cette action augmente la durée de la tour. L'observation importante est que l'utilité d'un PR varie avec la longueur de tour de ME, qui est illustré dans la Figure 2.2.

Supposons que chaque arc ayant une longueur d'une unité. Si le tour de ME est longue pour couvrir B avec un autre nœud, soit n_2, n_3 ou n_4, n_2 devrait être choisi car il économise la distance (2 unités pour chaque source) ces données se déplacent le long de l'arborescence. Cependant, si le tour de ME peut couvrir B, n_3 et n_4, l'utilité de n_2 devient être nulle car toutes les données sont ramassé par le ME à n_3 ou n_4 avant d'atteindre n_2. La structure itérative permet à RP-UG de mettre à jour dynamiquement utilité des nœuds lorsque le tour ME est développé et donc donne le meilleur choix des nœuds PR.

RP-UG ajoute ensuite des nœuds virtuels à l'arborescence de telle sorte que chaque arc plus long que la constante L_0 est divisé en plusieurs segments de ligne de longueur L_0. L_0 est un paramètre défini par l'application selon le compromis souhaitable entre la solution, la

qualité et la complexité de calcul. Un L_0 plus petit fournit RP-UG plus de choix de PR candidats tout en résultant en plus itérations.

RP-UG utilise la procédure TSP(I) pour calculer le minimum durée d'un tour qui visite tous les points de l'ensemble **I**. Le TSP(I) peut être implémenté par le problème du voyageur de commerce géométrique [36].

Au début d'une itération, RP-UG trouve tous les candidats PR (étape 2). Un nœud est un candidat PR s'il peut être visité ensemble avec tous les PR existants par un tour ne dépassant pas L. L'utilité du candidat x est définie comme :

$$u(x) = s_i \in S \sum dT(s_i, Q) - s_i \in S \sum dT(s_i, Q \cup \{x\}) / TSP(Q \cup \{x\}) - TSP(Q) \quad (1)$$

Où **dT** ($\mathbf{s_i}$, **I**) est la distance minimale le long de l'arborescence **T** de $\mathbf{s_i}$ à tout nœud dans **I**. u(x) est égal au rapport de la réduction de la distance totale que les morceaux de données voyagent le long de l'arborescence. Initialement la tournée augmentée. RP-UG ajoute ensuite le candidat PR avec la plus grande utilité à la liste PR (étape 4). Ensuite, tous les PR's dont les utilitaires devenir zéro sont supprimés de la liste PR (étape 5), qui est nécessaire car l'ajout d'un nouveau PR peut invalider certains ceux existants comme discuté plus tôt. Enfin, si tous les nœuds sources sont inclus dans la liste PR, RP-UG se termine parce que le ME peut visiter chacun d'eux dans les délais et l'énergie du réseau total est zéro. Sinon, une nouvelle itération est lancée pour trouver plus de PR. La complexité de RP-UG est O(|V| 2 · C(TSP)) où C(TSP) est la complexité de la procédure TSP. Il existe un nombre des algorithmes TSP avec différents compromis entre la complexité et la qualité de la solution de l'algorithme de [37] a un rapport d'approximation de (1 + 1 / c) et une complexité de O (|V| (log|V|) O(c)) pour tout fixe c> 1. Nous notons que RP-UG est seulement exécuté par la BS ou ME, qui avoir plus de puissance de calcul que les nœuds de réseau.

- Nous discutons maintenant un exemple d'exécution de RP-UG. [38] Figure **2.2** montre les listes PR et les tours ME à la fin de trois itérations. n_2 est inclus dans le tour à la première itération.

Dans l'itération 2, n_3 a la plus grande utilité parmi tous les nœuds. Bien que n_4 et n_3 provoquent la même augmentation de la durée de la tour, n_4 a une utilité plus petite où il enregistre seulement 1 puits de distance sur l'arc (n_4, n_2) qui est traversée par les données de s_3 tandis que n_3 enregistre 2 sur le bord (n_3, n_2) (1 pour le données de chacun de s_1 et s_2). n_4 est ajouté sur le tour en itération 3. n_2 est retiré de la visite (à l'étape 5 de RP-UG) parce que n_3 et n_4 sur le tour rend son utilité à zéro. En conséquence, la durée de la visite est raccourcie et, par conséquent, plus de PR peuvent être inclus dans les itérations suivantes.

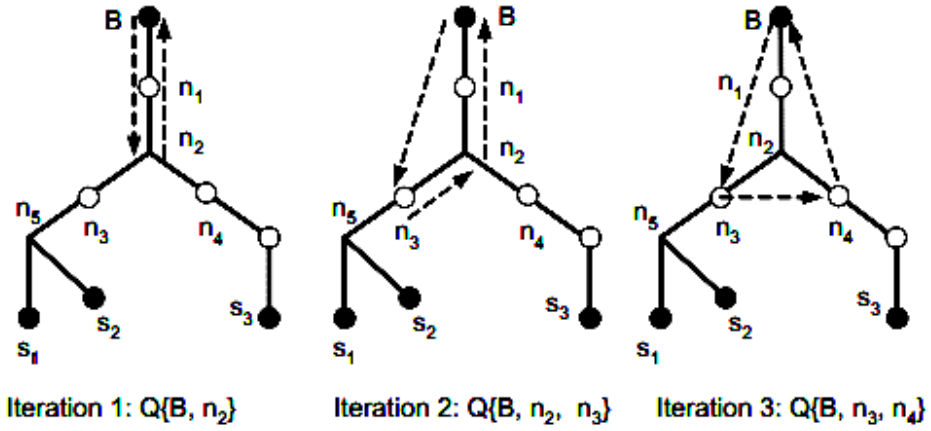


Figure 2. 2 un exemple d'exécution de RP-UG les listes PR et les tours ME à la fin de trois itérations.

2.3. Planification du rendez-vous au poids

WPR (weighted rendez-vous planning) désigne préférentiellement les nœuds de capteurs avec le plus haut poids (weight) comme un PR [39].

Notre algorithme consiste de 3 étapes essentielles pour déterminer les nœuds PR, les étapes sont :

- Répartition de réseau.
- Sélectionner les Points de départ pour les puits mobile.
- Déterminer les nœuds PR.

Nous commencent par la première étape dans notre algorithme WPR :

❖ Répartition de réseau.

Effectuez la première partition. Ceci est similaire à la méthode dans [40] pour diviser le réseau. Cependant, nous divisons le réseau en **c** parties, nous sélectionnons un nœud dans chaque partie comme point central initial, il devait être le plus proche du centre du cluster, tous les nœuds calculent la distance entre eux et les points de centre des clusters :

C = {c1, c2, c3, c4,..., cn}, la distance entre un nœud et les points du cluster sont Dc= {Dc_1, Dc_2, Dc_3, ..., Dc_n}, Nous trions les valeurs de DJ dans un ordre décroissant, le nœud qui a la plus petite distance entre eux et un cluster particulier ce que signifient que le nœud appartient à ce cluster.

❖ Sélectionner les Points de départ pour les puits mobile :

La deuxième étape consiste à sélectionner une tête de cluster (Cluster Head) pour chaque cluster. Pour faire ça, il est important d'obtenir la position avec la distance minimal de tous les nœuds aux points de centre de tous les clusters :

$Sk=\sum_{i=1}^{nr}(nr-uk)^2$ (2)

cette expression exprimée la distance entre les nœuds $N=\{n_1,n_2,n_3,\ldots n_r\}$ et les points de centre de cluster $K=\{u_1,u_2,u_3,\ldots,u_k\}$, et de façon similaire nous trions les valeurs de *SK* dans un ordre décroissant pour connaître les nœuds qui sont les têtes de cluster.

❖ Déterminer les nœuds PR

Pour déterminer les nœuds PR, nous devons calculer le poids de nœud par la multiplication de nombre de paquets qu'il avance par le nombre de sauts entre le nœud i et le nœud le plus proche PR dans un cluster. [41] Ainsi, le poids de nœud capteur i est calculé comme :

Wi = NFD (i) * H (i, M). (3)

En Basant sur (3), les nœuds capteurs qui sont à un saut d'un PR et ayant seulement un paquet de données automatiquement obtient le poids minimum. [41] Par conséquent, les nœuds qui sont plus éloignés de la sélection PRs ou ayant plus d'un paquet à transmettre au plus proche PR ont la plus haute priorité d'être recruté comme PR.

De (2) et (3), la consommation d'énergie est relativement dépend au nombre de sauts entre les nœuds sources, la destination, et le nombre de paquets de données à transférer. [42] Par conséquent, la visite du nœud le plus élevé (en poids) peut réduire le nombre de transmissions multi-saut et minimise ainsi la consommation d'énergie. En outre, comme les zones denses créent des points de congestion dus au plus grand nombre de nœuds, la probabilité d'apparaitre des trous d'énergie sont plus susceptibles dans le réseau [43]. Par conséquent, un puits mobile qui visite préférentiellement ces zones empêchera la formation de trous d'énergie dans le réseau [44].

Cet algorithme montre comment WPR fonctionne. [45] Il prend comme une entrée G (V, E), et la sortie est un ensemble de PR. WPR ajoute la station de base comme le premier PR.

Ensuite, il ajoute le nœud le plus élevé en termes de poids. Après cela, les appels WPR TSP (·) pour calculer le coût de la visite. Si la longueur de tour est inférieure à la longueur requise L-max, le nœud sélectionné est retiré de la liste de PR.

Après le nœud est ajouté en tant que PR, WPR supprime les PRs. de la tour qui ne reçoivent plus de paquets de données à partir des nœuds. C'est parce que l'ajout de capteur à la tour peut réduire le nombre de paquets de données dirigés vers ces PRs. Par conséquent, [46] cette étape offre WPR plus d'opportunités d'ajouter d'autres nœuds dans la tour [47].

La longueur maximale de la tour est de L-max = 90 m [48] [49]. WPR commence à partir du nœud récepteur et l'ajoute à la tour, c'est-à-dire, M = [puits]. Ensuite, un SPT enraciné sur le nœud récepteur est construit [voir Figure 2.3 (a)]. [50] Lors de la première itération, WPR ajoute le nœud 10 à tour parce qu'il a le poids le plus élevé, ce qui donne M = [puits, 10].

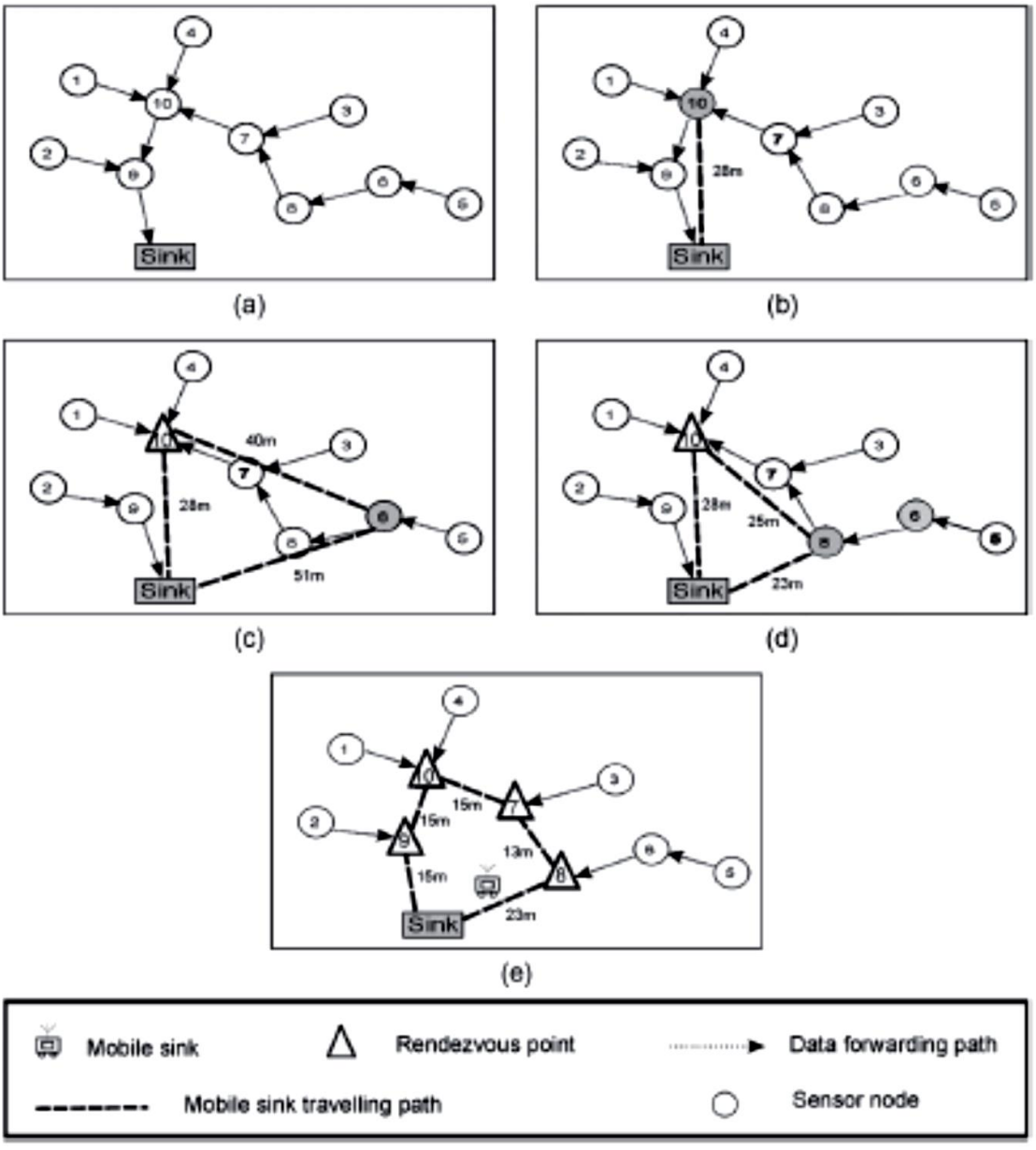


Figure 2. 3 Exemple de WPR fonctionnant dans un WSN avec dix nœuds.

Comme le montre la figure 2.3 (b), la longueur de la tournée de M est inférieure à la longueur de la visite requise (56 <90), ce qui signifie que le nœud 10 reste dans la dernière tournée. [51] Dans la deuxième itération, WPR recalcule le poids des nœuds capteurs car le nœud 10 maintenant est partie de la tournée. Dans cette itération, WPR sélectionne le nœud 6 comme prochain PR, qui a le poids le plus élevé. Comme le montre la figure 2.3 (c), La durée de la visite de M = [puits, 10, 6] est plus grande que celle requise longueur (119> 90). Par conséquent, WPR supprime le nœud 6 de la tour M = [puits, 10].

Dans la troisième itération, [53] le poids des nœuds de capteurs ne changera pas parce que le nœud 6 n'est pas sélectionné comme PR mais reste marqué et ne sera pas sélectionné. WPR sélectionne le nœud 8 car il a le poids le plus élevé et est pas marqué [voir Fig. 2.3 (d)]. La fonction TSP renvoie 76 m pour M = [puits, 10, 8], qui est inférieur à 90 m. Par conséquent, le nœud 8 est ajouté à la visite. Le processus continue, jusqu'à l'aboutissement d'une tournée finale de M = [puits,8, 7, 10, 9] avec une longueur de la tour de 81 m, qui est inférieure à la longueur de la tournée requise [voir Fig. 2.3 (e)].[54] Comme le montre la figure 2.2, le tour final calculé par WPR est toujours inclut les nœuds capteurs qui ont plus de paquets de données à transmettre que d'autres nœuds comme PR. [55][56] Ceci assure une consommation d'énergie uniforme et atténue le problème de trou d'énergie. C'est la clé avantage de WPR sur CB (cluster-based), RD-VT et RP-UG.

3. Etude comparative et évaluation de la performance des Algorithme de collection des données

Selon le travail effectué dans [57] qui évaluent les performances de chaque algorithme de collection des données par rapport des contraintes. Le RCSF a beaucoup de contraints comme la durée de vie de réseau, la consommation de l'énergie, Les figures suivantes illustrent la performance par rapport au contraints précédents :

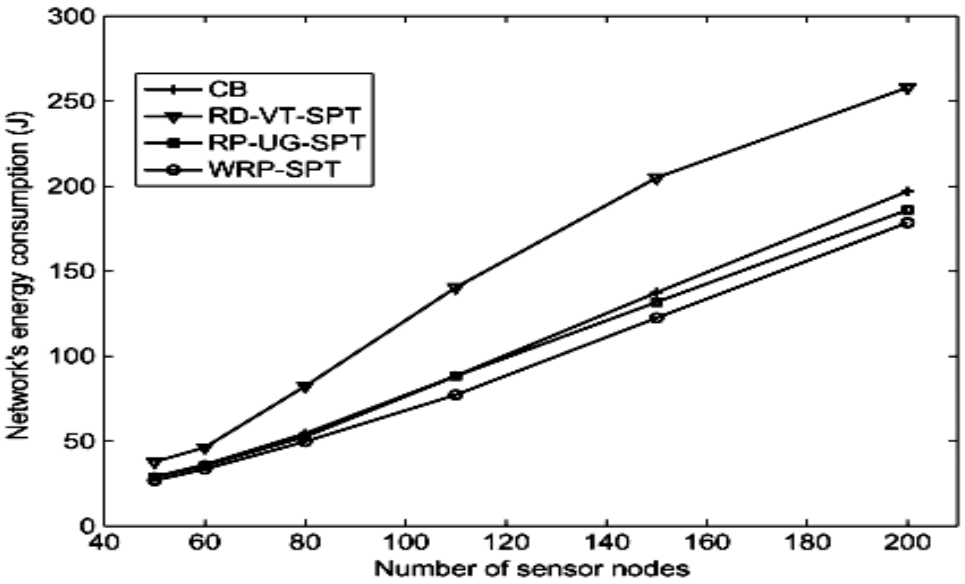


Figure 2. 4 Consommation d'énergie réseau pour WPR, CB, RD-VT et RP-UG.

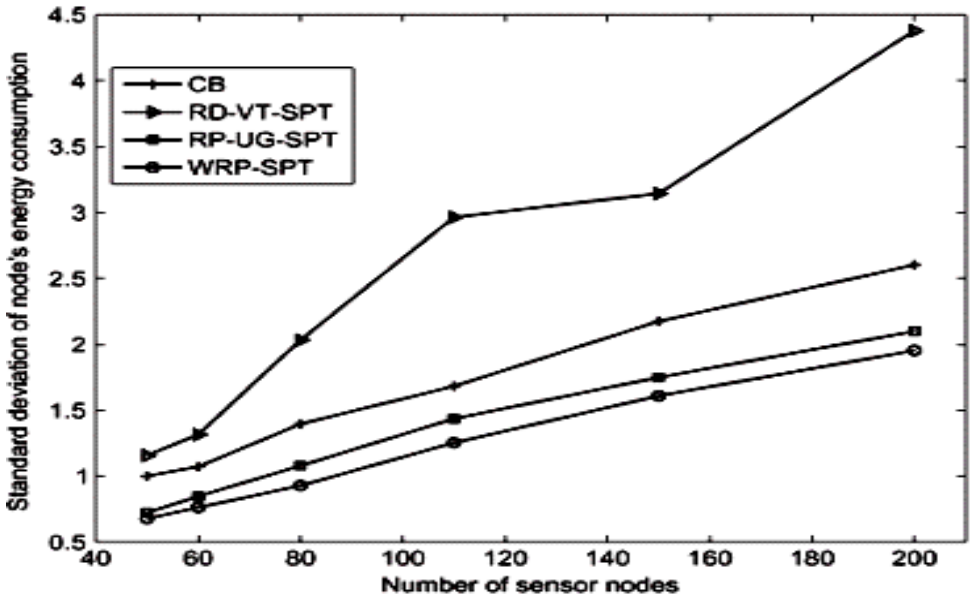


Figure 2. 5 Déviation standard de la consommation d'énergie des nœuds capteurs pour les modèles WPR, CB, RD-VT et RP-UG.

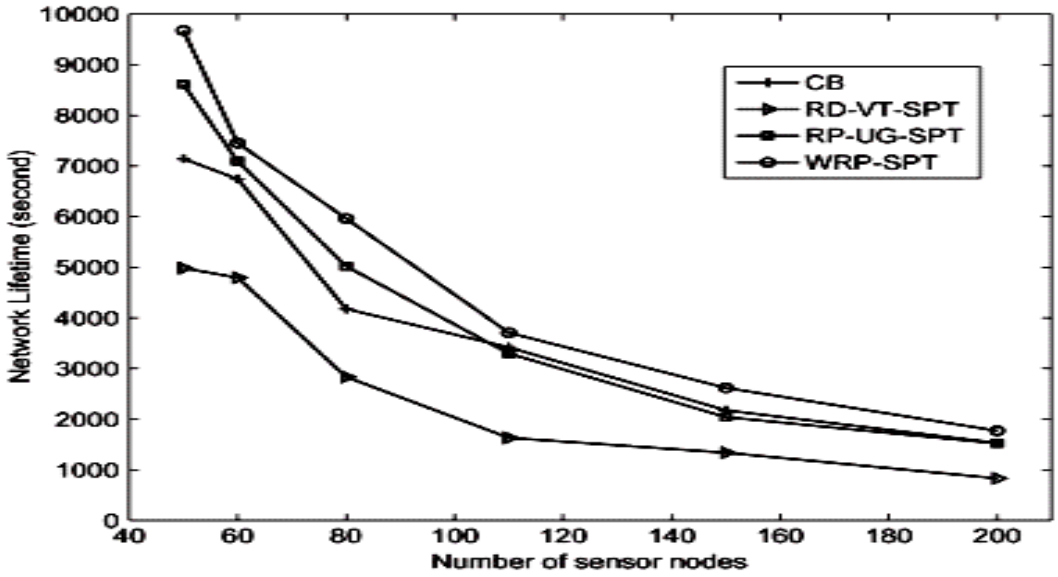


Figure 2. 6 Durée de vie du réseau pour WPR, CB, RD-VT et RP-UG.

Dans les figures précédentes, on remarque qu'il y a des algorithmes de collection des données sont plus fiables que les autres par exemple en terme de la consommation d'énergie et la durée de vie de réseau WPR est le meilleur algorithme par rapport aux autres algorithmes motionnés.

4. Conclusion

Dans ce chapitre, nous avons présenté trois algorithmes (RD-VT), (RP-UG) et WPR en détaille pour contrôler le mouvement d'un puits mobile dans un RCSF et faire la collection des données entre les nœuds capteurs. Après, nous avons fait une étude comparative selon le travail effectuée dans [57], [58] et nous conclurons que WPR est le meilleur algorithme en terme de la durée de vie de réseau et la consommation d'énergie. Dans le prochain chapitre, nous allons implémenter l'algorithme WPR

CHAPITRE : 03

IMPLEMENTATION ET SIMULATION

1. Introduction

Les réseaux de capteurs sont des réseaux composés de nœuds communicants à fortes contraintes (calcul, ´énergie, stockage, etc.). Chaque capteur génère des données (température locale, taux d'humidité, etc.) qu'il doit ensuite transmettre à une ou plusieurs stations de collecte (aussi appelées puits) par communications multi-sauts (i.e. chaque capteur génère et transmet ses données ainsi que celles d'autres capteurs), comme nous avons discuté dans le chapitre précédent, les puits mobiles responsables de parcourir le réseau et de collecter les données détectée par des nœuds spécial qui appel PR (Point de Rendez-vous).

Ce chapitre traite trois parties essentielles, premièrement, nous allons présentons les outils de simulation NS2, la deuxième partie décrit la phase d'implémentation de notre algorithme WPR (weighted rendez-vous planning), et l'implémentation de notre proposition pour améliorer les performances, nous allons utiliser la version 2.34 de NS2 installée sous Ubuntu (voir Annexes A).

Nous allons présenter une série de simulations réalisées à l'aide du simulateur NS-2 dans la troisième partie, et nous allons interpréter les résultats obtenus par NS2 pour évaluer les performances de nos travaux.

2. Implémentation

Avec l'évolution des réseaux sans fil et l'élaboration de plusieurs normes pour ces réseaux, et le besoin des simulations dans le contexte de l'évaluation des performances, de nombreux simulateurs des réseaux ont été développés. Les simulateurs les plus connus sont : NS-2 (Network Simulator 2), OPNET et GloMoSim/Qualnet, Tossim etc. [59]

Dans cette partie, nous allons présenter l'environnement d'implémentation (NS2), et leurs objets et composants, comme nous allons expliquer comment utiliser ce simulateur pour implémenter un nouveau protocole qui permet de collecter des données détectées de façon efficace dans un réseau de capteurs sans fil avec puits mobiles.

2.1. NS2 network simulator 2

Dans le cadre de notre étude qui vise à collecter les données dans un réseau de capteurs sans fil avec puits mobiles, nous avons choisi le simulateur NS-2 (open source) développé à Lawrence Berkeley National Laboratory (LBNL). C'est le simulateur de réseaux le plus utilisé par la communauté des chercheurs dans les RCSFs, il est utiles pour permettre aux

chercheurs de tester de nouveaux protocoles réseaux ou des modification sur les protocoles existants dans un environnement contrôlé et reproductible. NS-2 offre plusieurs avantages qui sont :

- NS2 est open source.
- Il est extensible, donc n'importe qui peut ajouter son propre protocole, ou faire une modification pour tester son algorithme.
- NS2 est orienté objet basé sur C++.

À la simulation, NS utilise OTcl (orienté objet Tool Command Langage) qui est un langage pour interpréter des scripts de simulation de l'utilisateur. Le langage OTcl est en fait une extension orientée objet du langage Tcl. Dans ce chapitre l'utilisation du langage Tcl sera expliquée en détail.

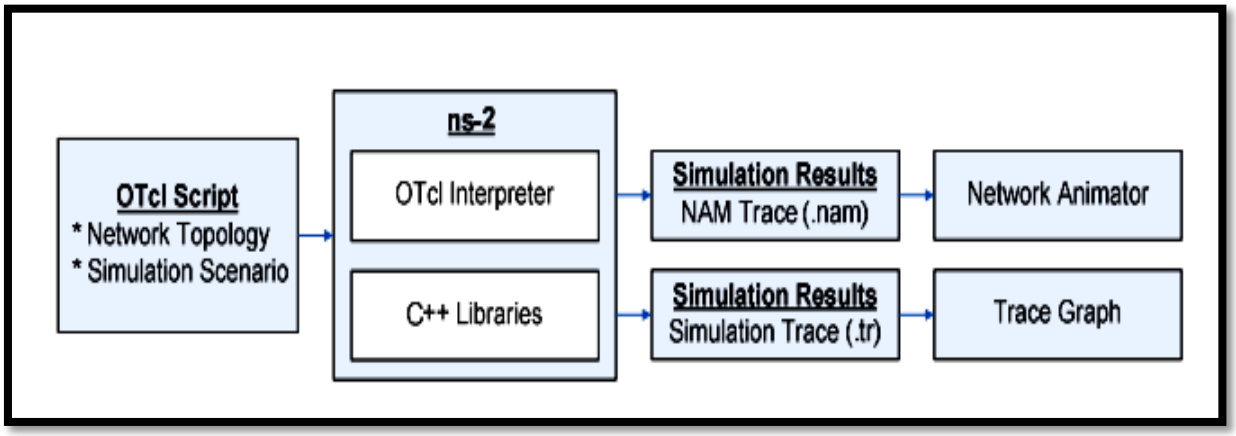


Figure 3. 1 schéma de NS2

Le résultat d'une simulation est un fichier trace de l'extension **«.tr »** contenant tous les événements de la simulation. Un traitement ultérieur de ce fichier permet d'en soustraire l'information souhaitée. Par ailleurs, le simulateur permet la création d'un fichier d'animation (dont l'extension .nam), permettant de visualiser la simulation sur une interface graphique NAM (Network Animator). La figure 3.1 représente de façon simplifiée le point de vue d'un utilisateur de NS.

2.1.1. Les composants du réseau

Les composants du réseau sont : nœud, lien, la file d'attente, etc. Certains de ces derniers sont des composants simples, ils sont créés à partir de classes C++, les autres sont des éléments composés. Ces composants sont

- **Node** : un nœud peut être une machine hôte, un switch, un routeur, une passerelle, etc.
- **Agent** : La classe agent fournit des méthodes utiles au développement, C'est la classe de base pour définir des nouveaux protocoles dans NS-2.
- **Application** : Classe mère de toutes les applications (Ftp, Telnet, Web).
- **Queue** : la classe mère de tous les buffers (DropTail, RED)
- **LinkDelay** : cette classe simule le délai de propagation et le temps de transmission sur les liens du réseau.
- **Packet** : la classe de tous les paquets circulant sur le réseau.
- **TimerHundler** : la classe mère de tous les timers (temporisateurs) utilisés par les protocoles du réseau [60].

2.1.2. Visualisation des résultats

➢ **Fichier trace :**

On peut demander à NS de récolter un certain nombre de données statistiques sur le déroulement de la simulation et de les sauvegarder dans un fichier "trace". La commande qui permet l'obtention d'un tel fichier (dans cet exemple nommé out.all) est la suivante :

```
set trace [open out.all]
$ns trace-all
$trace proc finish {} {
global trace Close $trace}
```

Ce fichier comportera tous les événements survenus lors de la simulation : l'arrivée d'un paquet à un nœud, départ d'un paquet d'un nœud, perte d'un paquet ou encore réception d'un paquet par un agent. Chaque paquet peut donc être suivi tout au long de son parcours. En traitant ce fichier de très grande taille on pourra en extraire l'information souhaitée : calcul du débit aux nœuds, évaluation des pertes, ….etc.

➢ **Interface graphique Nam (Network Animator) :**

Comme expliqué précédemment NAM (Network Animator) est un outil d'animation graphique basé sur Tcl/TK. Il permet de visualiser le déroulement d'une simulation en affichant la topologie du réseau et le déplacement des paquets. Son interface utilisateur est semblable à celle d'un lecteur CD (play, fast forward, pause, etc.) et permet notamment le contrôle de la vitesse du déroulement de la simulation.

2.2. Les étapes d'implémentation

- Installation de NS2.34 sous Ubuntu **10.10** (voir annexe A).
- Intégration des packages de **LOCP** dans NS2 (voir annexe B).
- Les fichiers principaux de protocole de collection des données **LOCP.**
- Implémentation de l'algorithme WPR, et l'extraction des nœuds PR.
- Collection des données par le puits mobile.
- Proposition de l'utilisation de plus un puits mobile.

2.3. Implémentation de nouveau protocole de collection des données dans RCSF

L'objectif de notre étude est la collecte de données détectées sur tous le réseau avec des puits mobiles. Nous avons donc utilisé les composants mentionnés précédemment pour construire un nouveau protocole de collection des données qui s'appelle **LOCP**, ce dernier utilise les informations (échange des paquets) fournies par la couche réseau.

2.3.1. Les fichiers principaux de notre implémentation

Notre protocole **LOCP** est implémenté sur 3 fichiers principaux :

1. Le fichier **Locp_packet.h**: ici, ils sont déclarés les types de paquets qu'utilise **LOCP** dans les échanges entre les nœuds du réseau.
2. Le fichier **locp.h** : c'est le fichier principal (header) dans lequel ils sont définis tous les Timers (temporisateurs) & les méthodes nécessaires, et l'agent qui exécute les fonctionnalités du protocole.
3. Le fichier **locp.cc** : dans lequel tous les Timers, les Tclhooks (liens entre les composants c++ et ceux d'OTcl) sont implémentés, ainsi que l'agent et ces fonctionnalités.

Nous allons commencer par le fichier Locp_packet :

- **Locp_packet.h**:

  Ce fichier contient les types de paquets pour notre protocole **LOCP** (Le format des paquets **hdr_locp_beacon)**, c'est un paquet qui est diffusé par les nœuds entre eux pour construire une vision sur ses voisins (voir figure 3.2) :

```
struct hdr_locp_beacon {
        u_int8_t        pkt_type;    // type of packet : Beacon or Error
        u_int32_t       beacon_id;   // unique identifier for the beacon
        nsaddr_t        beacon_src;  // source address of beacon, this is sink address
        double          beacon_received_power;//get reveived power from mac layer
        double          X_pos;//x coordinate of an ancre
        double          Y_pos;//y coordinate of an ancre
        double          timestamp;   // emission time of beacon message
```

Figure 3. 2 Format de paquet hdr_locp_beacon.

- **Le fichier locp.h**

Il contient des définitions & signatures des méthodes utilisées dans notre protocole voir figure 3.3 qui affichent une partie de ce fichier :

```
class WFRP : public Agent {
 public:
        WFRP(nsaddr_t id);
        void            recv(Packet *p, Handler *);
        int             command(int, const char *const *);
        // Agent Attributes
        nsaddr_t        index;      // node address (identifier)
        // Node Location
double          posx;       // position x;
double          posy;       // position y;
void            send_beacon(double x,double y); // Send Packets
// Remaining Energy
double          iEnergy;
int Pos;
double distance;
void Triangulation_model(Packet *p);
void Trilatelation_model(Packet *p);
void InverseMatrice(double M[1][1]);
double distanceCalculate(double x1, double y1, double x2, double y2);
                int Ancr_;
     wfrpBeaconTimer            bcnTimer;
     wfrpEnregisterTimer        EnrgTimer;

                double Pt;
                double L;
                double Gt;
                double Gr;
                double lambda;
double TS;

        int Nbr_Ancr;
        int Mthd;
```

Figure 3. 3 Fichier locp.h

- **Le fichier (classe) locp.cc**

Il représente le fichier le plus important dans notre implémentation, où on a deux méthodes principales qui sont : méthode **send_becon** et méthode **recv**.

2.3.2. Les méthodes de notre implémentation

- **méthode send_becon** :

Cette méthode responsable de diffuser des paquets contenant des informations comme la position de la source, entre les nœuds pour détecter leurs voisins, **hdr_locp_beacon** est le type de paquet dans notre protocole (voir figure 3.4).

```
void
LOCP::send_beacon(double x,double y) {
        Packet *p = Packet::alloc();
        struct hdr_cmn *ch = HDR_CMN(p);
        struct hdr_ip *ih = HDR_IP(p);
        struct hdr_locp_beacon *bcn = HDR_LOCP_BEACON(p);
        // Write IP Header
        ih->saddr() = index;
        ih->daddr() = IP_BROADCAST;
        ih->sport() = RT_PORT;
        ih->dport() = RT_PORT;
        // Write Beacon Header
        bcn->pkt_type = WFRP_BEACON;

        bcn->beacon_id = seqno;
        bcn->beacon_src = index;


        bcn->X_pos = x;
        bcn->Y_pos = y;

        bcn->timestamp = CURRENT_TIME;
        double ran=Random::uniform();

printf("S (%.6f): send beacon by %d \n", CURRENT_TIME, index);

        Scheduler::instance().schedule(target_, p, CURRENT_TIME+ran);
}
```

Figure 3. 4 Méthode send_becon

Lors de la réception d'un paquet au niveau de la couche liaison et après vérification qu'il s'agit d'un paquet **hdr_locp_beacon**, il y a invocation de la méthode du protocole permettant le traitement du paquet au niveau de la couche réseau : il s'agit de la méthode **recv()** de la classe **locp.cc** qui se charge de vérifier la validité du paquet. Elle détermine si ce paquet est de type **hdr_locp_beacon** ou non, en enregistrant des informations concernant la distance (voir figure 3.5).

```
void
LOCP::recv(Packet *p, Handler*) {
        struct hdr_cmn *ch = HDR_CMN(p);
        struct hdr_ip *ih = HDR_IP(p);
        if(ch->ptype() != PT_LOCP)  return;
        if(ch->ptype() == PT_LOCP) {
                struct hdr_locp_beacon *bcn = HDR_LOCP_BEACON(p);
                recp++;
                double Pr=bcn->beacon_received_power;
                double Dist=DistanceX(Pr,Pt,Gt,Gr,L,lambda);
                it = Voisins.begin();
                it = Voisins.insert ( it , bcn->index );
                itV = DistanceV.begin();
                itV = DistanceV.insert ( itV , DistanceX(Pr,Pt,Gt,Gr,L,lambda) );
                f1 = fopen("Res_Test.txtX", "a+");

                fprintf(f1,"%d %d %f \n",index,bcn->beacon_src,Dist);
        }
}
```

Figure 3. 5 Méthode recv().

2.3.3. Implémentation de l'algorithme WPR

Dans cette patrie nous expliquons comment LOCP sélectionne les nœuds PR, par conséquent notre protocole basée sur les Timers, où nous avons implémenté cinq Timers qui exécutent séquentiellement, ces Timers sont :

- **locpBeaconTimer** : Timer responsable de diffusion de paquet **hdr_locp_beacon** entre les nœuds, la figure suivante montre l'implémentation de cette Timer sous NS2 :

```
void
locpBeaconTimer::handle(Event*) {
        if(agent->Ancr_){
                agent->send_beacon(agent->iNode->X(),agent->iNode->Y());
        }
        Scheduler::instance().schedule(this, &intr,SEND_BECON_MESSAGE);
}
```

Figure 3. 6 locpBeaconTimer.

L'exécution de ce Timer invoque automatiquement la méthode recv() expliquée précédemment, les données reçues par le nœud sont sauvegardées dans un fichier qui appel **Res_Test.txtX.**

- **locpEnregisterTimer**: Timer responsable d'enregistrer les informations (nœud position, nombre des voisins, numéro de clusterHead) (Figure 3.7**)** de chaque nœud dans un fichier qui appel **Res_Test.txt**.

```
void locpEnregisterTimer::handle(Event*) {
if(CURRENT_TIME==3){
double X;
X=agent->Ver_Cluster(agent->iNode->X(),agent->iNode->Y());
FILE* f= fopen("Res_Test.txt", "a+");
fprintf(f,"%d %d %f %f %d %f \n",agent->index,agent->recp,
agent->iNode->X(),agent->iNode->Y(),agent->CLUSTER_NODE,X);
fclose (f);
}
Scheduler::instance().schedule(this, &intr,1);}
```

Figure 3. 7 Timer locpEnregisterTimer.

Comme il est illustré dans la figure **3.7,** le Timer **locpEnregisterTimer** utilise la méthode **Ver_Cluster()**, cette fonction détermine le cluster de chaque nœud, et la tête de cluster (Cluster Head) de chaque cluster, stocke les résultats dans un fichier appelée **Res_Test.txt.**

- **wfrpBeaconTimer** : qui fait la lecture des informations du fichier **Res_Test.txt** et calcule le nombre des sauts entre nœud particulier et le cluster Head (**Hi**) premièrement, et calcule le nombre des paquets transmis de nœud particulier vers le cluster Head (**NFD(i)**) deuxièmement et enfin enregistre des informations dans un fichier qui appel NFD.txt (Figure 3.8).

```
void
wfrpBeaconTimer::handle(Event*) {
if(CURRENT_TIME==4 ){
agent->ReadFile();
agent->Ver_PR();
std::vector<int>::iterator it;
FILE* f12 = fopen("NFD.txt", "a+");

if (it != agent->C1.end()) {
agent->Path= agent->dijkstra(agent->graph,agent->index,agent->C1[0],agent->Path);
fprintf(f12,"%d ",agent->Path.size()-1); for(int i=0;i<agent->Path.size();i++)
fprintf(f12,"%d ",agent->Path[i]);fprintf(f12,"\n");fclose (f12);}

if (it != agent->C2.end()) {
agent->Path= agent->dijkstra(agent->graph,agent->index,agent->C2[0],agent->Path);
fprintf(f12,"%d ",agent->Path.size()-1); for(int i=0;i<agent->Path.size();i++)
fprintf(f12,"%d ",agent->Path[i]); fprintf(f12,"\n");fclose (f12);}

if (it != agent->C3.end()){
agent->Path= agent->dijkstra(agent->graph,agent->index,agent->C3[0],agent->Path);
fprintf(f12,"%d ",agent->Path.size()-1);for(int i=0;i<agent->Path.size();i++)
fprintf(f12,"%d ",agent->Path[i]); fprintf(f12,"\n"); fclose (f12);}

if (it != agent->C4.end()){
agent->Path= agent->dijkstra(agent->graph,agent->index,agent->C4[0],agent->Path);
fprintf(f12,"%d ",agent->Path.size()-1);for(int i=0;i<agent->Path.size();i++)
fprintf(f12,"%d ",agent->Path[i]); fprintf(f12,"\n");fclose (f12);}
}
}
Scheduler::instance().schedule(this, &intr,1);}
```

Figure 3. 8 Timer wfrpBeaconTimer.

- **WieghtCalculated** : ce Timer fait la lecture du fichier **NFD.txt**, à travers ce fichier, il calcule le poids de chaque nœud (**Wi** = Weight) pour extraire les nœuds PR, et sauvegarde les résultats obtenus dans un fichier **Res_Timer_Send_Msg.txt** (Figure 3.9).

```
void CalCule_Wieght::handle(Event*) {
if(CURRENT_TIME==5){
std::vector<int> res;
if(agent->index==0){//
int Forw_Tree=0;
for(int i=1;i<V;i++) {
std::pair<int, int> p;
p=agent->ReadFile_Wt(i);//*Timer lire les info dans le fichier
//printf("pere de noeud %d est %d \n",agent->index,agent->Forward_True);
int Hi = p.first;//apprendre Hi nomber des sauts entre noued source et sa cluster head
int Ci = p.second;// apprendre Ci (Forwarding_Tree) nombre des paquet transmit par le noeud source
int Wi=Hi*Ci;//calculer Wi le poids de chaque neoud (Wieght)
res.push_back(Wi);
}
agent->travllingSalesmanProblem(agent->index);
//cete methode pour trouver le plus court tour pour un puits mobile
}
Scheduler::instance().schedule(this, &intr,1);}
```

Figure 3. 9 Timer WieghtCalculated.

- **Send_Msg** : pour lancer la procédure de collection de données, ce Timer invoque une série de transmissions entres les nœuds et les nœuds PR, chaque nœud envoie ses paquets à son propre nœud PR, la figure suivante montre le code de ce Timer :

```
void Send_Msg::handle(Event*) {
        if(CURRENT_TIME==6){
              if(!agent->PR_ or agent->index!=0 or agent->index!=49) //noeud 0 : Station de base
                                                                     //noeud 49 : Puits mobile
                    agent->send_beacon_msg(agent->Forward_True,0,agent->Path_to_PR,agent->Father_Pr);
              }
        Scheduler::instance().schedule(this, &intr,1);
}
```

Figure 3. 10 Timer Send_Msg.

2.3.4. Collection des données par le puits mobile

L'idée principale dans le cadre de notre stratégie est de déplacer le puits mobile pour collecter les données rassemblées dans les nœuds PR, après la détermination des nœuds PR par les étapes précédentes, le puits mobile circule sur les nœuds PR sélectionnée pour collecter les données, nous sommes capables de mettre en œuvre cette opération, grâce à simulateur ns2, précisément langage TCL, qui nous permet de se déplacer un puits.

En utilisant **setdest**, (set destination), on peut crée le cheminement des mouvements de puits mobiles, la figure suivante illustre l'utilisation de cette méthode pour déplacer les puits.

```
$ns at  7.0 " $node_(1) setdest 0.0  30.0  30.0"
```

Figure 3. 11 Script TCL de mobilité.

$ns at [temps_de_mouvement] [nœud_mouvementée][posX] [posY] [vitesse].

Selon la figure ci-dessus, le nœud(1) au moment 7.0s commence à se déplacer vers la destination (0,30) à une vitesse de 30 m/s. Cette ligne de commande peut être utilisée pour modifier la direction et la vitesse de déplacement de puits mobile. Dans chaque mouvement, le nœud(1) arrive aux nœuds PR pour la collection des données.

2.4. Proposition de l'utilisation de plus un puits mobile

Dans le réseau de capteurs, la collection de données est un problème réel, en particulier lorsque la file d'attente des capteurs n'a pas de la capacité de stocker des données pendant longtemps, et si un nœud n'envoie pas ses données détectées, et continue à recevoir des informations détectées de l'environnement, évidemment, les dernières données seront abandonnées.

Dans cette section, nous discutons la possibilité d'utiliser plus d'un puits mobile, théoriquement, l'utilisation d'un seul puits mobile cause l'augmentation de la durée de collection des données et de la consommation de l'énergie, pour gagner le temps et l'énergie, l'utilisation de plus d'un puits mobile minimise le temps de collection des données.

Nous allons proposer une solution de minimiser le temps de collection des données, deux puits mobiles circulant sur le réseau, la figure suivante présente un exemple de notre travail.

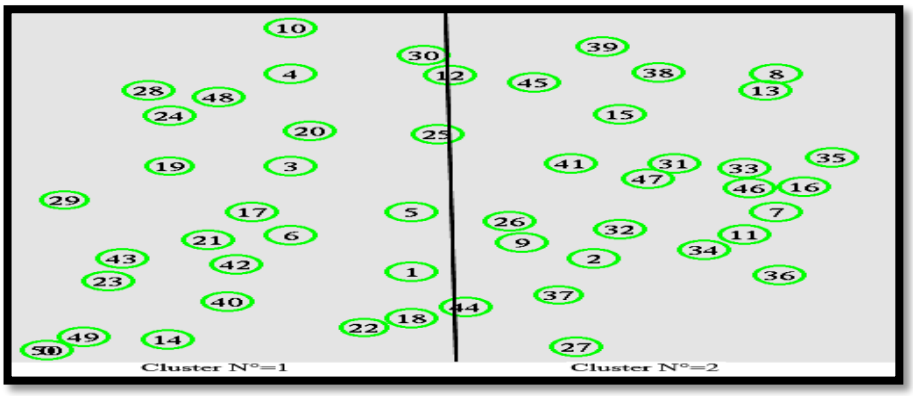


Figure 3. 12 Topologie de scénario de simulation.

L'idée est de diviser le réseau en deux clusters et d'utiliser deux puits mobiles, afin de parcourir les nœuds PR et pour garantir la collection des données rapidement et de garantir que le puits mobile peut circuler dans le réseau plusieurs fois.

3. Simulation et interprétation des résultats

3.1. Simulation

3.1.1. Intérêt et nécessité de la simulation

L'évaluation de performances de protocole de collection de données peut se faire en utilisant NS2, le recours à la simulation présente de nombreux avantages. En effet, elle est plus rapide, moins coûteuse en ressources, répétable et permettant d'isoler des paramètres qui peuvent parfois affecter les performances. De plus, il existe des scénarios très difficiles à étudier dans la réalité [60].

3.1.2. Hypothèses

Certaines hypothèses sont nécessaires pour pouvoir mettre en œuvre notre travail, en particulier :

- Chaque nœud dans la zone de captages d'un autre nœud est un voisin de ce dernier.
- La surface de notre simulation de 2D (deux dimensions X et Y) et la distance en mètres.
- Les liens de communication entre les nœuds sont des liens bidirectionnels.
- Les puits mobiles sont des nœuds qui ayant une trajectoire spécifique.
- Chaque nœud dans la simulation représente un nœud de capteurs sans fil.

3.1.3. Les Métriques de Performance

Les simulations sont faites par rapport à deux métriques, dans le but de tester notre protocole sur différents aspects. Les principales métriques d'évaluation des performances de notre protocole de LOCP sont :

- ✓ Consommation d'énergie.
- ✓ Temps de collection des données.

3.1.4. Modèle de simulation NS2

Tout d'abord il faut commencer par initialiser les valeurs des paramètres à utiliser pendant les simulations. Ces paramètres sont offerts par le simulateur NS-2 et sont paramétrables via les scripts Tcl.

Le simulateur NS-2 implémente les différentes couches nécessaires (application, transport, routage, MAC et physique) pour la simulation d'un réseau ad hoc. Le tableau 3.1 résume les différents paramètres utilisés dans les simulations.

Paramètres	**Valeur**
Protocole de collection	WPRP
Modèle de propagation	Two-Ray-Ground
Couche MAC	Mac/802_11
Portée de transmission	13 mètres
Temps de simulation	30 seconds
Surface de simulation	70*70 m²

Tableau 3. 1 les paramètres de simulation de LOCP.

3.2. Discussion et interprétation des résultats

Dans cette partie, nous présentons les résultats de simulation et nous discuterons ensuite les résultats obtenus.

3.2.1. Résultats de protocole de collection des données avec puits mobiles

Nous commençons par les résultats des Timers, et comme nous avons expliqué précédemment, l'exécution de chaque Timer produit un fichier qui contient des résultats concernant notre travail.

- **locpBeaconTimer** : comme nous l'avons déjà dit, ce Timer provoque la diffusion des paquets de type **hdr_locp_beacon** entre les nœuds**,** la figure suivante montre quelque résultats du fichier **Res_Test.txtX** :

noeud-id	voisins-id	distance
21	40	11.139417
43	40	10.555873
22	40	10.328069
23	40	8.854842
14	40	7.877069
42	40	6.735374
46	34	11.478518
7	34	8.329610
2	34	7.590540
36	34	6.871775
32	34	6.794835
11	34	3.849406
33	41	11.810687
26	41	11.166330
25	41	10.420166
15	41	9.388793
31	41	7.074359

Figure 3. 13 Contenu de fichier Res_Test.txtX.

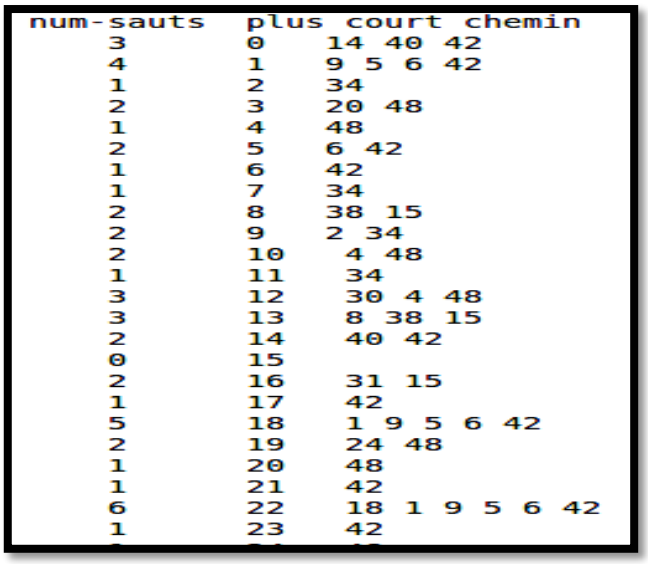
Chaque nœud enregistre ces informations comme identificateur de nœud, nombre des voisins et la distance entre eux.

- **locpEnregisterTimer** : la figure suivante représente une partie des résultats de cette Timer de fichier **Res_Test.txt**, la 1^{ére} colonne représente l'identificateur de nœud, la 2ème représente le nombre des voisins, la 3ème et 4ème présentent la position 2D, la 5ème colonne représente le cluster de ce nœud, et la dernière colonne représente la distance entre le nœud et son propre cluster.

noeud	numvoisin	pos-X	pos-Y	Numcluster	distace-noeud-cluster
0	5	35.000000	0.000000	1	24.748737
1	9	30.000000	17.000000	1	12.509996
2	6	45.000000	20.000000	2	7.905694
3	5	20.000000	40.000000	3	12.747549
4	7	20.000000	60.000000	3	7.905694
5	6	30.000000	30.000000	1	17.677670
6	6	20.000000	25.000000	1	7.905694
7	10	60.000000	30.00000	0	2 14.577380
8	2	60.000000	60.000000	4	10.606602
9	7	39.068194	23.365805	2	14.656775
10	2	20.000000	70.00000	0	3 17.677670
11	9	57.388529	25.08917	3	2 9.027362
12	5	33.179170	59.76763	7	3 17.281635
13	3	59.049339	56.34326	0	4 7.593714
14	3	9.951248	2.316531	1	16.956456
15	6	47.156346	51.18364	0	4 5.503403
16	7	62.301931	35.48219	7	4 19.638826
17	7	16.898279	30.00000	0	1 12.514474
18	5	30.000000	7.000000	1	16.324828
19	5	10.000000	40.00000	0	3 14.577380
20	6	21.600246	47.58202	6	3 6.403006

Figure 3. 14 Contenu de fichier Res_Test.txt.

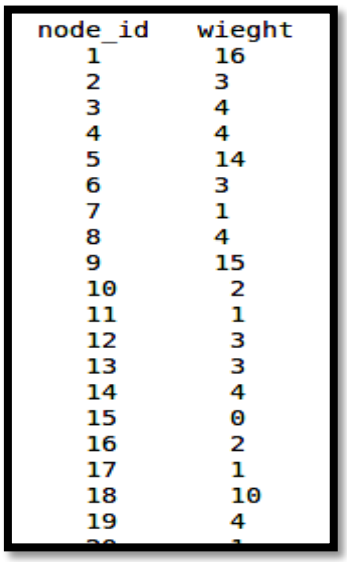
- **wfrpBeaconTimer** : la figure suivante représente les résultats obtenus dans le fichier **NFD.txt**, où la 1ère colonne représente le nombre de sauts entre un nœud particulier et son propre PR.



num-sauts	plus court chemin
3	0 14 40 42
4	1 9 5 6 42
1	2 34
2	3 20 48
1	4 48
2	5 6 42
1	6 42
1	7 34
2	8 38 15
2	9 2 34
2	10 4 48
1	11 34
3	12 30 4 48
3	13 8 38 15
2	14 40 42
0	15
2	16 31 15
1	17 42
5	18 1 9 5 6 42
2	19 24 48
1	20 48
1	21 42
6	22 18 1 9 5 6 42
1	23 42

Figure 3. 15 Contenu de fichier NFD.txt.

- **WieghtCalculated** : la figure suivante représente les résultats du fichier **Res_Timer_Send_Msg.txt,** la 1ère colonne représente l'identificateur de nœud et la 2ème colonne représente le poids de chaque nœud.



node_id	wieght
1	16
2	3
3	4
4	4
5	14
6	3
7	1
8	4
9	15
10	2
11	1
12	3
13	3
14	4
15	0
16	2
17	1
18	10
19	4

Figure 3. 16 Les poids des nœuds.

Apres l'appel de méthode TSP qui retourne le plus courte chemin (route) de puits mobile entre les nœuds PR et la station de base, la figure suivante illustre le résultat de cet algorithme dans le terminal de NS2.

```
sayad@ubuntu:~/apps/ns-allinone-2.34/ns-2.34$ ns soutnance.tcl
num_nodes is set 50
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 57.8
SORTING LISTS ...DONE!

nodes PR are :
node 42 :
node 34 :
node 48 :
node 15 :
node 14 :
node 1 :
node 5 :
node 19 :
node 8 :
```

Figure 3. 17 Les nœuds PR.

Comme il est indiqué dans la figure 3.17, les nœuds PR sélectionnées dans notre scénario de simulation sont les nœuds : **14, 1, 42, 5, 19, 48, 8, 15 et 34** respectivement.

- **Send_Msg** : pour lancer la procédure de collection de données, ce Timer invoque une série des transmissions entres les nœuds et les nœuds PR, chaque nœud envoie ses paquets à son propre nœud (**Figure 3.18**):

```
sayad@ubuntu:~/apps/ns-allinone-2.34/ns-2.34$ ns soutnance.tcl
num_nodes is set 50
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 57.8
SORTING LISTS ...DONE!
S (6.000000): send beacon by  0 to its Forwarding Tree Node 14 and  Father_PR is 14
S (6.000000): send beacon by  1 to its Forwarding Tree Node 9 and  Father_PR is 1
S (6.000000): send beacon by  2 to its Forwarding Tree Node 34 and  Father_PR is 34
S (6.000000): send beacon by  3 to its Forwarding Tree Node 19 and  Father_PR is 19
S (6.000000): send beacon by  4 to its Forwarding Tree Node 48 and  Father_PR is 48
S (6.000000): send beacon by  5 to its Forwarding Tree Node 49 and  Father_PR is 5
S (6.000000): send beacon by  6 to its Forwarding Tree Node 5 and  Father_PR is 5
S (6.000000): send beacon by  7 to its Forwarding Tree Node 34 and  Father_PR is 34
S (6.000000): send beacon by  8 to its Forwarding Tree Node 13 and  Father_PR is 8
S (6.000000): send beacon by  9 to its Forwarding Tree Node 1 and  Father_PR is 1
```

Figure 3. 18 Propagation des paquets des données.

- **Exploiter les résultats du fichier trace**

NS-2, permet d'afficher temporellement les résultats d'une trace d'exécution NS-2 dans un fichier d'extension .tr. Par exemple, il est capable de représenter des

paquets TCP ou UDP, la rupture d'un lien entre nœuds, ou encore de représenter les paquets rejetés d'une file d'attente pleine. Cet outils est souvent appelé directement depuis les scripts TCL pour NS-2, pour visualiser directement le résultat de la simulation, la figure suivante présente le fichier **Trace_file.tr**

```
s 6.103528178 _42_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.995671 ei 0.000 es 0.000 et 0.001 er 0.003] ------- [42:255 -1:255 10 0]
r 6.104932198 _21_ RTR  --- 0 LOCP 100 [0 ffffffff 2a 800] [energy 99.995172 ei 0.000 es 0.000 et 0.001 er 0.004] ------- [42:255 -1:255 10 0]
r 6.104932204 _6_ RTR  --- 0 LOCP 100 [0 ffffffff 2a 800] [energy 99.995671 ei 0.000 es 0.000 et 0.001 er 0.003] ------- [42:255 -1:255 10 0]
r 6.104932206 _40_ RTR  --- 0 LOCP 100 [0 ffffffff 2a 800] [energy 99.995333 ei 0.000 es 0.000 et 0.001 er 0.004] ------- [42:255 -1:255 10 0]
r 6.104932210 _43_ RTR  --- 0 LOCP 100 [0 ffffffff 2a 800] [energy 99.995172 ei 0.000 es 0.000 et 0.001 er 0.004] ------- [42:255 -1:255 10 0]
r 6.104932216 _23_ RTR  --- 0 LOCP 100 [0 ffffffff 2a 800] [energy 99.995671 ei 0.000 es 0.000 et 0.001 er 0.003] ------- [42:255 -1:255 10 0]
r 6.104932216 _17_ RTR  --- 0 LOCP 100 [0 ffffffff 2a 800] [energy 99.995172 ei 0.000 es 0.000 et 0.001 er 0.004] ------- [42:255 -1:255 10 0]
r 6.104932227 _1_ RTR  --- 0 LOCP 100 [0 ffffffff 2a 800] [energy 99.994173 ei 0.000 es 0.000 et 0.001 er 0.005] ------- [42:255 -1:255 10 0]
s 6.141572998 _22_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.997169 ei 0.000 es 0.000 et 0.001 er 0.002] ------- [22:255 -1:255 10 0]
```

Figure 3. 19 Fichier de traceTrace_file.tr.

S [temps_de_emmision] [nœud_source][protocole] [energie_initial] [Energie_actuel].

Le caractère S signifie Send (émission), et le caractère R signifie Receive (Réception), la première ligne indique que le nœud numéro 42 a envoyé un paquet au moment 6.1835 avec l'énergie initial 100 & l'énergie actuelle 99.995671, la deuxième ligne indique que le nœud numéro 21 a reçu le paquet au moment 6.1849.

- **Les résultats de collection des données avec le puits mobiles**

Afin de collecter les données, et comme nous l'avons montré ci-dessus, **setdest** donne l'avantage de mobilité de n'importe quel nœud, dans notre scénario de simulation, nous choisissons le nœud N°=49 comme le puits mobile responsable de collecter les données dans les nœuds PRs. mentionnées précédemment, le script TCL suivant affiche le déplacement du nœud N°49 dans le réseau.

```
$ns at 9.0 "$mnode_(49) setdest 9.0  2.0 90.0"
##14
$ns at 10.0 "$mnode_(49) setdest 30.0  17.0 90.0"
##42
$ns at 11.0 "$mnode_(49) setdest 15.0  18.0 90.0"
##5
$ns at 12.0 "$mnode_(49) setdest 30.0  30.0 90.0"
##19
$ns at 13.0 "$mnode_(49) setdest 10.0  40.0 90.0"
##48

$ns at 14.0 "$mnode_(49) setdest 14.0  54.0 90.0"
##8
$ns at 16.0 "$mnode_(49) setdest 60.0  60.0 90.0"
##15
$ns at 18.0 "$mnode_(49) setdest 47.0  51.0 90.0"
 ##34
$ns at 20.0 "$mnode_(49) setdest 54.0  21.0 90.0"
##0
$ns at 22.0 "$mnode_(49) setdest 0.1  0.1   90.0"
```

Figure 3. 20 Script TCL de mobilité de puits mobile.

Après l'exécution de cette script dans le terminal de NS2 par la commande **[nsnom_fichier.tcl]**, le simulateur permet la création d'un fichier d'animation (extension .nam), permettant de visualiser la simulation sur l'interface graphique (Figure 3.21) NAM (Network Animator).

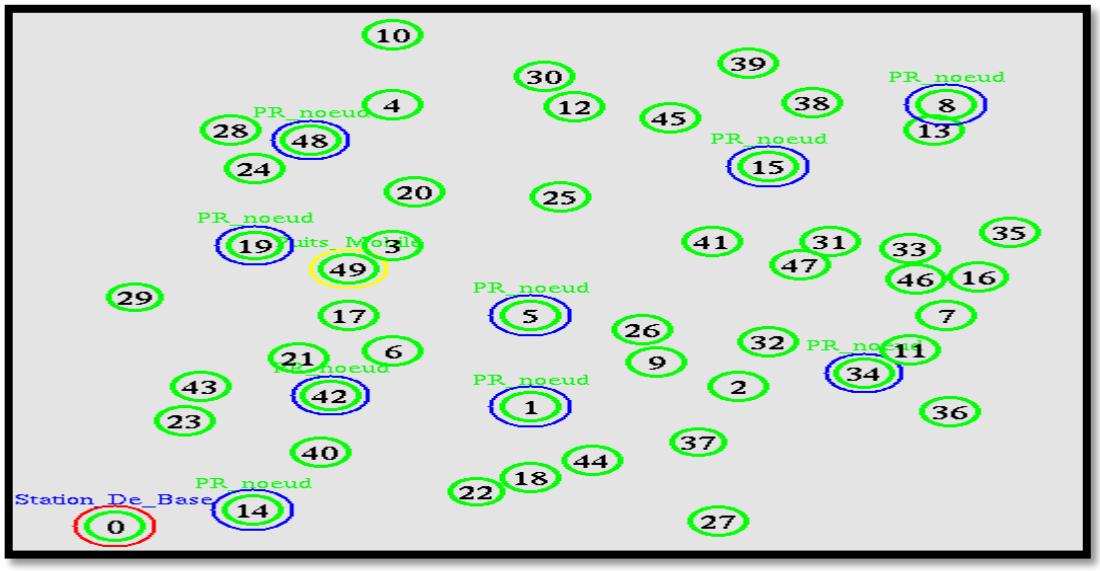


Figure 3. 21 Sélection de l'cheminement de puits mobile.

Le nœud en cercle rouge représente la station de base, qui traite tous les informations du réseau, les nœuds en cercle bleu représentent les nœuds PR, et le nœud en jaune représente le puits mobile qui circule sur les nœuds PR pour collecter les données

Pour mieux comprendre le mouvement de puits mobile et la collecte de données à partir des nœuds PR, la figure suivante exprime le fichier (Trace.tr) de trace généré par NS2.

```
M 9.00000 49 (3.00, 3.00, 0.00), (9.00, 2.00), 90.00
s 9.446244334 _14_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.993838 ei 0.000 es 0.000 et 0.002 er 0.004] ------- [14:255 49:255 10 0]
r 9.451635358 _49_ RTR  --- 0 LOCP 100 [13a 31 e 800] [energy 99.991600 ei 0.000 es 0.000 et 0.003 er 0.006] ------- [14:255 49:255 10 0]
M 10.00000 49 (9.00, 2.00, 0.00), (30.00, 17.00), 90.00
s 10.388544217 _1_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.987847 ei 0.000 es 0.000 et 0.002 er 0.010] ------- [1:255 49:255 10 0]
r 10.393255217 _49_ RTR  --- 0 LOCP 100 [13a 31 1 800] [energy 99.989362 ei 0.000 es 0.000 et 0.004 er 0.007] ------- [1:255 49:255 10 0]
M 11.00000 49 (30.00, 17.00, 0.00), (15.00, 18.00), 90.00
s 11.636935723 _42_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.989041 ei 0.000 es 0.000 et 0.002 er 0.009] ------- [42:255 49:255 10 0]
r 11.642586744 _49_ RTR  --- 0 LOCP 100 [13a 31 2a 800] [energy 99.987124 ei 0.000 es 0.000 et 0.005 er 0.008] ------- [42:255 49:255 10 0]
M 12.00000 49 (15.00, 18.00, 0.00), (30.00, 30.00), 90.00
M 13.00000 49 (30.00, 30.00, 0.00), (10.00, 40.00), 90.00
s 13.096671832 _5_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.989540 ei 0.000 es 0.000 et 0.002 er 0.009] ------- [5:255 49:255 10 0]
r 13.101903039 _49_ RTR  --- 0 LOCP 100 [13a 31 5 800] [energy 99.984886 ei 0.000 es 0.000 et 0.006 er 0.009] ------- [5:255 49:255 10 0]
M 14.00000 49 (10.00, 40.00, 0.00), (14.00, 54.00), 90.00
s 14.023076628 _19_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.992188 ei 0.000 es 0.000 et 0.002 er 0.006] ------- [19:255 49:255 10 0]
r 14.028967682 _49_ RTR  --- 0 LOCP 100 [13a 31 13 800] [energy 99.982648 ei 0.000 es 0.000 et 0.007 er 0.010] ------- [19:255 49:255 10 0]
s 15.088432567 _48_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.992688 ei 0.000 es 0.000 et 0.002 er 0.006] ------- [48:255 49:255 10 0]
r 15.094123589 _49_ RTR  --- 0 LOCP 100 [13a 31 30 800] [energy 99.980410 ei 0.000 es 0.000 et 0.008 er 0.011] ------- [48:255 49:255 10 0]
M 16.00000 49 (14.00, 54.00, 0.00), (60.00, 60.00), 90.00
s 16.238006078 _8_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.994837 ei 0.000 es 0.000 et 0.002 er 0.003] ------- [8:255 49:255 10 0]
M 18.00000 49 (60.00, 60.00, 0.00), (47.00, 51.00), 90.00
s 18.874564756 _15_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.989844 ei 0.000 es 0.000 et 0.002 er 0.008] ------- [15:255 49:255 10 0]
r 18.879455762 _49_ RTR  --- 0 LOCP 100 [13a 31 f 800] [energy 99.978173 ei 0.000 es 0.000 et 0.009 er 0.013] ------- [15:255 49:255 10 0]
M 20.00000 49 (47.00, 51.00, 0.00), (54.00, 21.00), 90.00
s 20.607495227 _34_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.989344 ei 0.000 es 0.000 et 0.002 er 0.009] ------- [34:255 49:255 10 0]
r 20.612886247 _49_ RTR  --- 0 LOCP 100 [13a 31 22 800] [energy 99.975935 ei 0.000 es 0.000 et 0.010 er 0.014] ------- [34:255 49:255 10 0]
M 22.00000 49 (54.00, 21.00, 0.00), (0.10, 0.10), 90.00
s 25.191670366 _49_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.975935 ei 0.000 es 0.000 et 0.010 er 0.014] ------- [49:255 0:255 10 0]
r 25.196601369 _0_ RTR  --- 0 LOCP 100 [13a 0 31 800] [energy 99.992295 ei 0.000 es 0.000 et 0.003 er 0.005] ------- [49:255 0:255 10 0]
D 30.000000000 _8_ IFQ  END 0 LOCP 100 [0 0 8 800] [energy 99.994382 ei 0.000 es 0.000 et 0.002 er 0.003] ------- [8:255 49:255 10 0]
```

Figure 3. 22 Fichier de trace Trace.tr.

Nous discutons les résultats obtenus par le fichier **Trace.tr**, d'après la figure ci-dessus les lignes qui commence par le caractère M signifie le déplacement du nœud N° 49 (puits mobile) sur le réseau pour collecter les données, et les lignes en jaunes signifie la réception des paquets collectés, dans les dernières lignes nous remarquons que le puits mobile 49 a retourné a la station de base et il a envoyé un paquets de collection des données, la station de base a reçu cette paquets de données, nous remarquons qu'après chaque arrivée de puits mobile au niveau des nœuds PR, ce dernier envoie les données collectées, et ensuite le puits mobile reçoit ces paquets, et ces résultats représentent ce que nous voulons de notre travail.

3.2.2. Proposition de l'utilisation de plus un puits mobile

Comme nous expliquons que la possibilité d'utiliser plus d'un puits mobile, améliorera la performance de notre protocole en termes d'énergie et le temps de la collection des données, en commençons par l'implémentation avec deux puits mobiles, dans notre scénario de simulation les nœuds proposée comme puits mobiles sont : nœud N° 49 et N°50 respectivement, la figure suivante représente une partie de script TCL qui indique la mobilité :

```
##14
$ns at 9.0 "$mnode_(49) setdest 9.0  2.0 90.0"
##1
$ns at 10.0 "$mnode_(49) setdest 30.0  17.0 90.0"
##42
$ns at 11.0 "$mnode_(49) setdest 15.51  18.70 90.0"
##19
$ns at 12.0 "$mnode_(49) setdest 10.0  40.0 90.0"
##48
$ns at 13.0 "$mnode_(49) setdest 14.0  54.0 90.0"
##0
$ns at 15.0 "$mnode_(49) setdest 0.1  0.1   90.0"


## 34
$ns at 9.0 "$mnode_(50) setdest 54.0  21.0 90.0"
##5
$ns at 10.0 "$mnode_(50) setdest 30.0  30.0 90.0"
##15
$ns at 11.0 "$mnode_(50) setdest 47.0  51.0 90.0"
##8
$ns at 13.0 "$mnode_(50) setdest 60.0  60.0 90.0"
##0
$ns at 15.0 "$mnode_(50) setdest 0.1  0.1   90.0"
```

Figure 3. 23 Script TCL de mobilité de deux puits mobiles.

L'exécution de cette script invoque la génération de fichier animation .nam qui permet de visualiser la simulation sur l'interface graphique NAM (Network Animator), la figure suivante montre l'exécution de cette commande dans le terminal de simulateur NS2 :

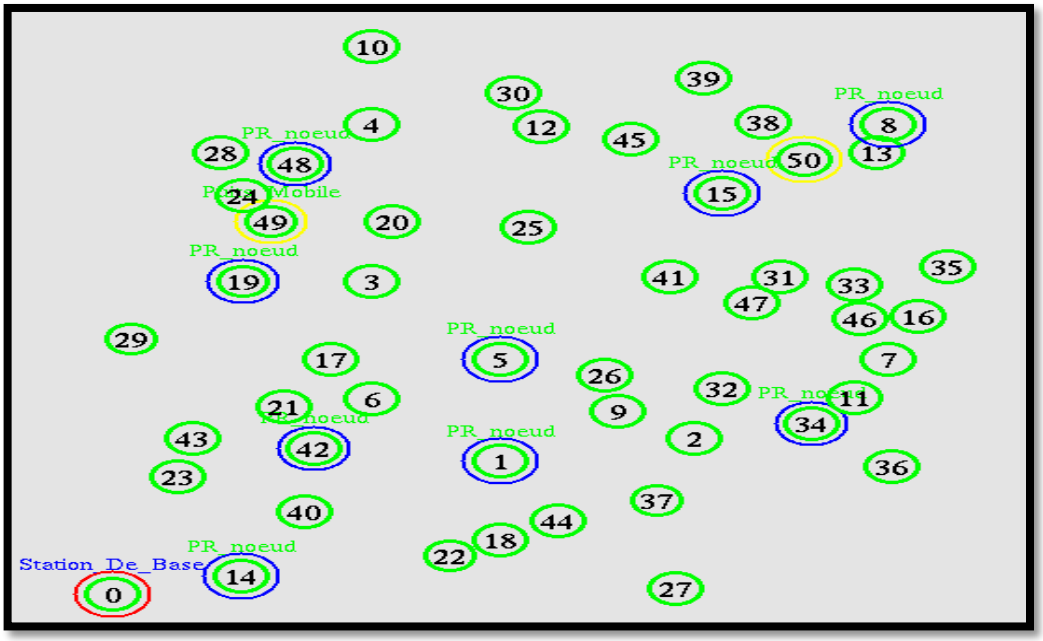


Figure 3. 24 Scenario de topologie de deux puits mobiles.

D'après la figure ci-dessus, nous remarquons dans un certain moment, les nœuds en jaune (puits mobiles) circulant sur les nœuds PR pour collecter les données, le puits mobile N° 50 déplace vers PR N°8, et le puits mobile N°49 déplace vers le PR N°48, pour suivre la traçabilité des paquets dans le réseau, NS2 génère un fichier de trace contenant les informations mentionnée précédemment, dans ce cas

de scénario, nous étudions le fichier **Trace_F.tr**, la figure suivante représente une partie de ce fichier :

```
M 10.00000 49 (9.00, 2.00, 0.00), (30.00, 17.00), 90.00
M 10.00000 50 (54.00, 21.00, 0.00), (30.00, 30.00), 90.00
s 10.003382199 _34_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.992340 ei 0.000 es 0.000 et 0.002 er 0.006] ------- [34:255 50:255 10 0]
r 10.009313219 _50_ RTR  --- 0 LOCP 100 [13a 32 22 800] [energy 99.996094 ei 0.000 es 0.000 et 0.003 er 0.001] ------- [34:255 50:255 10 0]
s 10.671001100 _1_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.989344 ei 0.000 es 0.000 et 0.002 er 0.009] ------- [1:255 49:255 10 0]
r 10.675972100 _49_ RTR  --- 0 LOCP 100 [13a 31 1 800] [energy 99.992857 ei 0.000 es 0.000 et 0.004 er 0.003] ------- [1:255 49:255 10 0]
s 10.810511062 _5_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.990539 ei 0.000 es 0.000 et 0.002 er 0.008] ------- [5:255 50:255 10 0]
r 10.815642062 _50_ RTR  --- 0 LOCP 100 [13a 32 5 800] [energy 99.992054 ei 0.000 es 0.000 et 0.004 er 0.004] ------- [5:255 50:255 10 0]
M 11.00000 49 (30.00, 17.00, 0.00), (15.51, 18.70), 90.00
M 11.00000 50 (30.00, 30.00, 0.00), (47.00, 51.00), 90.00
s 11.791431679 _42_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.989540 ei 0.000 es 0.000 et 0.002 er 0.009] ------- [42:255 49:255 10 0]
r 11.796882679 _49_ RTR  --- 0 LOCP 100 [13a 31 2a 800] [energy 99.988818 ei 0.000 es 0.000 et 0.005 er 0.006] ------- [42:255 49:255 10 0]
s 11.922116992 _15_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.992340 ei 0.000 es 0.000 et 0.002 er 0.006] ------- [15:255 50:255 10 0]
r 11.927047998 _50_ RTR  --- 0 LOCP 100 [13a 32 f 800] [energy 99.989817 ei 0.000 es 0.000 et 0.005 er 0.005] ------- [15:255 50:255 10 0]
M 12.00000 49 (15.51, 18.70, 0.00), (10.00, 40.00), 90.00
M 12.00000 50 (47.00, 51.00, 0.00), (60.00, 60.00), 90.00
s 12.694680702 _8_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.996334 ei 0.000 es 0.000 et 0.002 er 0.002] ------- [8:255 50:255 10 0]
r 12.699831702 _50_ RTR  --- 0 LOCP 100 [13a 32 8 800] [energy 99.987579 ei 0.000 es 0.000 et 0.006 er 0.006] ------- [8:255 50:255 10 0]
s 12.819717324 _19_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.993339 ei 0.000 es 0.000 et 0.002 er 0.005] ------- [19:255 49:255 10 0]
r 12.825168324 _49_ RTR  --- 0 LOCP 100 [13a 31 13 800] [energy 99.986580 ei 0.000 es 0.000 et 0.006 er 0.007] ------- [19:255 49:255 10 0]
M 13.00000 49 (10.00, 40.00, 0.00), (14.00, 54.00), 90.00
M 13.00000 50 (60.00, 60.00, 0.00), (1.10, 1.10), 90.00
s 13.446244334 _48_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.994337 ei 0.000 es 0.000 et 0.002 er 0.004] ------- [48:255 49:255 10 0]
r 13.451635356 _49_ RTR  --- 0 LOCP 100 [13a 31 30 800] [energy 99.984342 ei 0.000 es 0.000 et 0.007 er 0.009] ------- [48:255 49:255 10 0]
M 14.00000 49 (14.00, 54.00, 0.00), (1.10, 1.10), 90.00
s 14.388544217 _50_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.987579 ei 0.000 es 0.000 et 0.006 er 0.006] ------- [50:255 0:255 10 0]
r 14.812459267 _0_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.997668 ei 0.000 es 0.000 et 0.001 er 0.005] ------- [0:255 -1:255 10 0]
s 15.520389808 _49_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.984342 ei 0.000 es 0.000 et 0.007 er 0.009] ------- [49:255 0:255 10 0]
r 15.671830634 _0_ RTR  --- 0 LOCP 100 [0 0 0 0] [energy 99.997169 ei 0.000 es 0.000 et 0.001 er 0.005] ------- [0:255 -1:255 10 0]
D 30.000000000 _49_ IFQ  END 0 LOCP 100 [0 0 31 800] [energy 99.983888 ei 0.000 es 0.000 et 0.008 er 0.009] ------- [49:255 0:255 10 0]
D 30.000000000 _50_ IFQ  END 0 LOCP 100 [0 0 32 800] [energy 99.986853 ei 0.000 es 0.000 et 0.006 er 0.007] ------- [50:255 0:255 10 0]
```

Figure 3. 25 Fichier de trace Trace_F.tr.

Nous remarquons, dans la figure ci-dessous, les mouvements des deux puits mobiles, les lignes en jaune représentent le temps de transmission des paquets entre les deux puits mobiles et les nœuds PR, chaque puits mobile circule dans son propre cluster, et suit un chemin spécifique pour collecter les données, à la fin du fichier nous remarquons le retour des deux puits mobiles à la station de base pour passer les informations détectées par le réseau.

3.2.3. Comparaison entre les deux techniques de collection de données

Nous avons implémenté deux techniques de collection de données qui sont : Collection des données par un puits mobile, et par deux puits mobiles, les métriques de mesure de performance sont :

- ✓ La consommation d'énergie.
- ✓ Le temps de collection des données.

Les deux figures suivantes (Figure 3.26 et Figure 3.27) montrent les résultats obtenus concernant les métriques : temps de collection des données, et consommation d'énergie de puits mobile N° 49 :

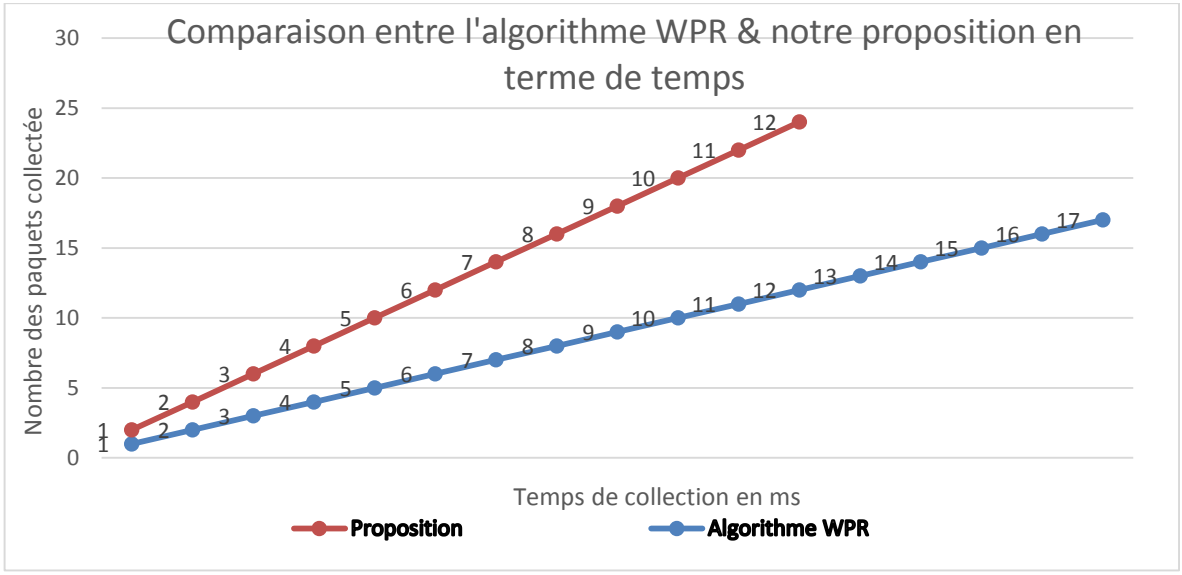


Figure 3. 26 Comparaison en termes de temps de collection des données.

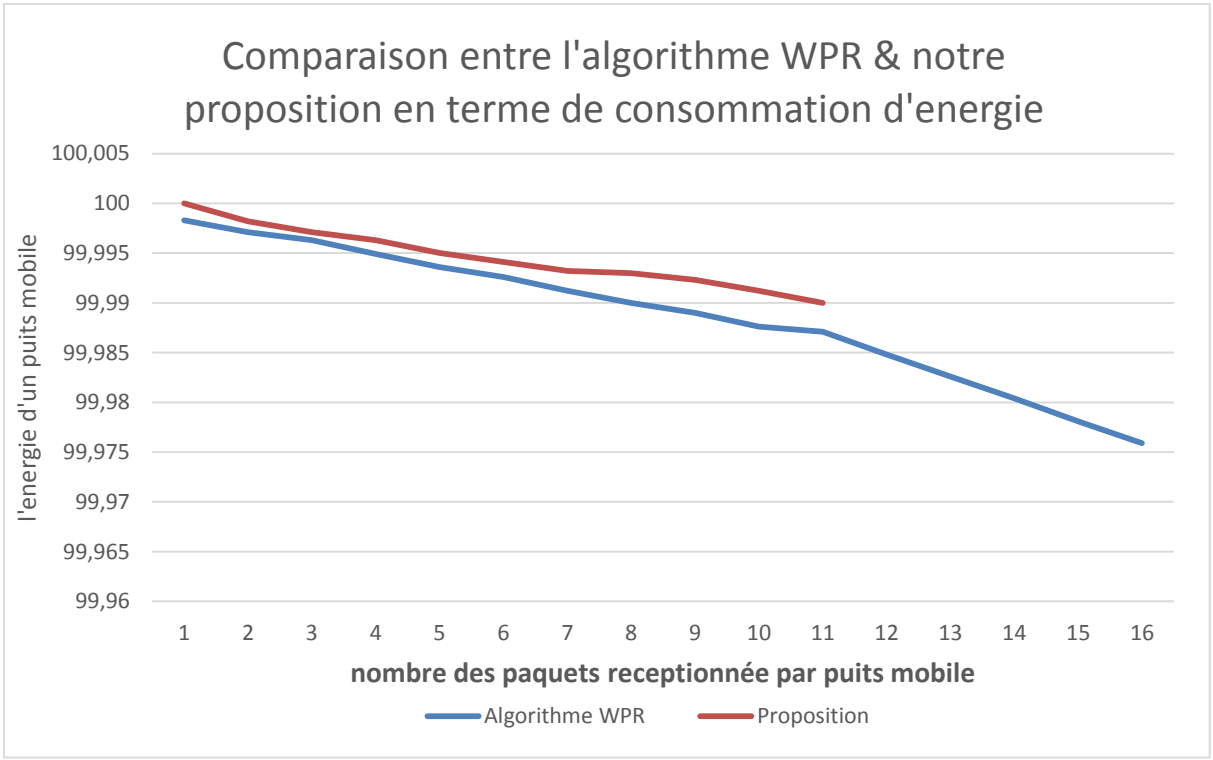


Figure 3. 27 Comparaison en termes de consommation d'énergie.

3.2.4. Discussion

Comme il est présenté dans les deux Figures 3.25 & Figure 3.26, il est évident que la performance de notre proposition est meilleure que l'algorithme WPR implémentée, que ce soit, le temps de collection des données ou la consommation d'énergie sont diminués. Nous concluons que l'utilisation de deux puits mobiles Influèrent la consommation d'énergie et le temps de collection des données.

4. Conclusion

Dans ce chapitre nous avons présenté l'environnement d'implémentation Network Simulator (NS2), et leurs objets & composants, et nous avons expliqué la conception de notre protocole de collection des données son implémentation sous NS2 en détaille, nous avons, ensuite, présenté notre proposition pour améliorer la performance de ce protocole, en utilisant deux puits mobiles. Nous avons fait une comparaison entre les deux implémentations (Algorithme WPR & notre proposition) en terme de consommation d'énergie et le temps de collection des données, nous avons exposé les résultats de notre travail et concluons que la proposition est plus performante que l'algorithme original.

.

CONCLUSION GENERALE

Les réseaux de capteurs sans fil (RCSF) ne cesse d'attirer l'attention de la communauté des chercheurs, et ce grâce à leur potentiel application dans un grand nombre de domaines, tels que : l'agriculture, l'industrie, le militarisme, l'habit intelligent, le transport etc.

Cependant, ils restent encore de nombreux problèmes à résoudre dans ce domaine afin de pouvoir les utiliser dans les conditions réelles. L'un des problèmes qu'on peut rencontrer dans ce genre de réseau est la collection des données dans un réseau da capteurs sans fil.

Dans ce mémoire, nous avons abordé en premier lieu, un état de l'art sur les réseaux de capteurs, et leurs caractéristiques, ainsi leurs domaines d'application.

Dans Le deuxième chapitre, nous avons présenté la notion de collection des données dans les RCSF, qu'elle base sur l'utilisation des puits mobiles qui se déplacent à travers le réseau, nous avons présenté 3 algorithmes de collection des données (RD-VT), (RP-UG) et WPR, respectivement.

Dans le troisième chapitre, nous avons implémenté un nouveau protocole de collection des donnée sous NS2 (network simulateur) après la description de ces composants, nous avons proposé une nouvelle approche basée sur l'utilisation de deux puits mobiles au lieu d'un puits, et ce pour accélérer le processus de collection des données. Nous avons discuté nos résultats d'implémentation, Les résultats obtenus par simulations ont montré que l'utilisation de deux puits mobiles soit plus performante que l'utilisation d'un seul puits mobile, en termes de consommation d'énergie et le temps de collection des données

Perspectives futures

Dans les travaux futures, nous envisageons de travailler sur les puits mobiles rechargeable qui passent l'énergie d'une manière sans fil aux nœuds PR (nœuds de point de rendez-vous), et en même temps, collectent les données au niveau de ces nœuds.

BIBLIOGRAPHIE

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci , “A Survey on Sensor Networks”. IEEE Communications Magazine, August 2002.

[2] B. Jung and G. Sukhatme, “Multi-target tracking using a mobile sensor network”. In Proceedings of the IEEE International Conference on Robotics and Automation, May 2002.

[3] C. Chong and S. Kumar, “Sensor networks: Evolution, opportunities, and challenges” Proceedings of IEEE, August 2003.

[4] E. M. Petriu, N. D. Georganas, D. C. Petriu, D. Makrakis, and V. Z. Groza, “Sensor-based information appliances”. IEEE Information and Measurement Magazine, December 2000.

[5] B. Sibbald, “Use computerized systems to cut adverse drug events”. Canadian Medical Association Journal, June 2001.

[6] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, “Impact of Radio Irregularity on Wireless Sensor Networks”. MobiSys 2004.

[7] ALERT Systems. http://www.alertsystems.org/.

[8] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, and T. Porcheron, “Monitoring behavior in home using a smart fall sensor”. IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology, October 2000.

[9] J. Lester Hill, “System Architecture for Wireless Sensor Networks”. University of California, Berkeley, 2003.

[10] V. Handziski, A. Kopke, H. Karl, and A. Wolisz, “A common wireless sensor network architecture”. TechnischeUniversitت Berlin, July 2003, pp.10-17.

[11] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, “Fault Tolerance in Wireless Ad hoc Sensor Networks”. Proceedings of IEEE Sensors 2002, June 2002.

[12] L. Paradis and Q. Han, “A Survey of Fault Management in Wireless Sensor Networks”. Plenum Press New York, NY, USA, 2007.

[13] F. Nekoogar, F. Dowla, and A. Spiridon, “Self organization of wireless sensor networks using ultra-wideband radios”. Atlanta, GA, United States, September 2004.

[14] C.Y. Chong and S.P. Kumar, “Sensor networks: Evolution, opportunities, and challenges”. Proceedings of the IEEE, vol. 91, n.8, 2003, pp. 1247-1256.

[15] T. Zhao, W. D. Cai, and Y. J. Li, “A Sensor Network Topology Inference Algorithm, omputational Intelligence and Security”. 2007 International Conference on. January 2008.

[16] V. Handziski, J. Polastrey, J. H. Hauer, C. Sharpy, A. Wolisz, and D. Cullery, “Flexible Hardware Abstraction for Wireless Sensor Networks”. In Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN '05),February 2005.

[17] Jaap C. Haartsen, “The Bluetooth radio system”. IEEE Personal Communications Magazine, February 2000, pp. 28-36.

[18] Practel, Inc. ZigBee, “Technology for Wireless Sensor Networks”. April 2006.

[19] I. Teixeira, J. F. de Rezende, A. de Castro, and A. C. P. Pedroza, “Wireless Sensor Network: Improving the Network Energy Consumption”. in XXI Symposium Brazilian Telecommunications, SBT’04, Belem, Brazil, September 2004.

[20] E. Souto, R. Gomes, D. Sadok and J. Kelner, “Sampling Energy Consumption in Wireless Sensor Networks”. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1 (SUTC'06), June 2006.

[21] M. Ali and S. K. Ravula, “Real-time support and energy efficiency in wireless sensor networks”. Technical report, IDE0805, January 2008.

[22] K. Akkaya, and M. Younis, “A Survey on Routing Protocols for Wireless Sensor Networks”. Journal of Ad Hoc Networks, Vol. 3, No. 3, May 2005, pp. 325-349.

[23] J.N. Al-Karaki and A.E. Kamal, “Routing techniques in wireless sensor networks: a survey. Wireless Communications”. IEEE, Decembre 2004.

[24] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless sensor networks ”. In the Proceeding of the Hawaii International Conference System Sciences, Hawaii, January 2000.

[25] S. Ziane and A. Mellouk, “A swarm intelligent scheme for routing in mobile ad networks”. Systems Communications, IEEE, Aug 2005.

[26] PaoloSanti, “Topology Control in Wireless Ad Hoc and Sensor Networks”. Hardcover, july 2005.

[27] S. Kumar, D. Shepherd, and F. Zhao, “Collaborative signal and information processing in micro-sensor networks”. IEEE Signal Processing Magazine, March 2002.

[28] B. Krishnamachari, D. Estrin and S. Wicker, “Modelling Data-Centric Routing in Wireless Sensor Networks”. USC Computer Engineering Technical Report CENG 02-14, 2002.

[29] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, “Energy-aware wireless microsensor networks”. IEEE Signal Processing Magazine, Vol. 19, No. 2, March 2002, pp.4050.

[30] D. J. Chang and E. K. Morlok. Vehicle speed profiles to minimize work and fuel consumption. Journal of Transportation Engineering, 131(3), 2005.

[31] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme. Robomote: enabling mobility in sensor networks. In IPSN, 2005.

[32] R. Pon, M. A. Batalin, J. Gordon, A. Kansal, D. Liu, M. Rahimi, L. Shirachi, Y. Yu, M. Hansen, W. J. Kaiser, M. Srivastava, G. Sukhatme, and D. Estrin. Networked infomechanical systems: a mobile embedded networked sensor platform. In IPSN, 2005.

[33] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava. Mobile element scheduling with dynamic deadlines. IEEE Transactions on Mobile Computing, 6(4), 2007.

[34] “Multi-sector crisis management consortium (mscmc),” http://www.mscmc.org.

[35] NingXu, SumitRangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin, “A wireless sensor network for structural monitoring,” in SenSys, 2004.

[36] E. Ekici; YaoyaoGu; D. Bozdag, “Mobility-based communication in wireless sensor networks,” IEEE Communications Magazine, vol. 44 , no. 7, 2006.

[37] Richard Pon, Maxim A. Batalin, Jason Gordon, AmanKansal, Duo Liu, Mohammad Rahimi, Lisa Shirachi, Yan Yu, Mark Hansen, William J. Kaiser, Mani Srivastava, Gaurav Sukhatme, and Deborah Estrin, “Networked infomechanical systems: a mobile embedded networked sensor platform,” in IPSN, 2005.

[38] Arun A. Somasundara, Aditya Ramamoorthy, and Mani B. Srivastava, “Mobile element scheduling with dynamic deadlines,” IEEE Transactions on Mobile Computing, vol. 6, no. 4, 2007.

[39] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: A survey,” Comput. Netw., vol. 38, no. 4, pp. 393–422 , Mar. 2002

[40] Wang, C.; Li, J.; Ye, F.; Yang, Y. A mobile data gathering framework for wireless rechargeable sensor networks with vehicle movement costs and capacity constraints. IEEE Trans. Comput. 2016, 65, 2411–2427. [CrossRef]

[41] I. Bekmezci and F. Alagz, “Energy efficient, delay sensitive, fault tolerant wireless sensor network for military monitoring,” Int. J. Distrib. Sens. Netw., vol. 5, no. 6, pp. 729–747, 2009.

[42] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, “Wireless sensor networks for habitat monitoring,” in Proc. 1st ACM Int. Workshop Wireless Sens. Netw. Appl., New York, NY, USA, Sep. 2002, pp. 88–97.

[43] J. Zhang, W. Li, Z. Yin, S. Liu, and X. Guo, “Forest fire detection system based on wireless sensor network,” in Proc. 4th IEEE Conf. Ind. Electron. Appl., Xi’an, China, May 2009, pp. 520–523.

[44] L. Ruiz-Garcia, L. Lunadei, P. Barreiro, and I. Robla, “A review of wireless sensor technologies and applications in agriculture and food industry: State of the art and current trends,” Sensors, vol. 9, no. 6, pp. 4728–4750, Jun. 2009.

[45] N. Wang, N. Zhang, and M. Wang, “Wireless sensors in agriculture and food industry–recent development and future perspective,” Comput. Electron. Agriculture, vol. 50, no. 1, pp. 1–14, Jan. 2006.

[46] A. Wheeler, “Commercial applications of wireless sensor networks using zigbee,” IEEE Commun. Mag., vol. 45, no. 4, pp. 70–77, Apr. 2007.

[47] W. Chen, L. Chen, Z. Chen, and S. Tu, “Wits: A wireless sensor network for intelligent transportation system,” in Proc. 1st Int. Multi-Symp. Comput. Comput. Sci., Hangzhou, China, Jun. 2006, vol. 2, pp. 635–641.

[48] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, “Cartel: A distributed mobile sensor computing system,” in Proc. 4th Int. Conf. Embedded Netw. Sens. Syst., New York, NY, USA, Oct. 2006, pp. 125–138.

[49] L. Yu, N. Wang, and X. Meng, “Real-time forest fire detection with wireless sensor networks,” in Proc. Int. Conf. Wireless Commun., Netw. Mobile Comput., Wuhan, China, Sep. 2005, vol. 2, pp. 1214–1217.

[50] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” IEEE Commun. Mag., vol. 40, no. 8, pp. 102–114, Aug. 2002.

[51] J. Lian, K. Naik, and G. B. Agnew, “Data capacity improvement of wireless sensor networks using non-uniform sensor distribution,” Int. J. Distrib. Sens. Netw. vol. 2, no. 2, pp. 121–145, Apr.–Jun. 2006.

[52] J. Li and P. Mohapatra, “An analytical model for the energy hole problem in many-to-one sensor networks,” in Proc. 62nd IEEE Veh. Technol. Conf., Dallas, TX, USA, Sep. 2005, vol. 4, pp. 2721–2725.

[53] J. Luo and J.-P. Hubaux, “Joint mobility and routing for lifetime elongation in wireless sensor networks,” in Proc. IEEE INFOCOM, Miami, FL, USA, Mar. 2005, vol. 3, pp. 1735–1746.

[54] R. C. Shah, S. Roy, S. Jain, and W. Brunette, “Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks,” Ad Hoc Netw., vol. 1, no. 2/3, pp. 215–233, Sep. 2003.

[55] X. Li, A. Nayak, and I. Stojmenovic, Sink Mobility in Wireless Sensor Networks. Hoboken, NJ, USA: Wiley, 2010, pp. 153–184.

[56] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas, “Efficient data propagation strategies in wireless sensor networks using a single mobile sink,” Comput. Commun., vol. 31, no. 5, pp. 896–914, Mar. 2008.

[57] X. Wu, G. Chen, and S. Das, “Avoiding energy holes in wireless sensor networks with nonuniform node distribution,” IEEE Trans. Parallel Distrib. Syst., vol. 19, no. 5, pp. 710–720, May 2008.

[58] W. Liang, J. Luo, and X. Xu, “Prolonging network lifetime via a controlled mobile sink in wireless sensor networks,” in Proc. IEEE Global Telecommun. Conf., Miami, FL, USA, Dec. 2010, pp. 1–6.

[59] Kamal BEYDOUN, “ Conception d’un protocole de routage hiérarchique pour les réseaux de capteur,” L’U.F.R Des Sciences et Techniques de L’Université de FRANCHE- COMTE, pp 31, 16 Décembre 2009.

[60] A. MAKHOUL, “Réseaux de capteurs : localisation, couverture, et fusion de données,” thèse de doctorat. LIFC, Université de Franche-Comté, Novembre 2008.

ANNEXES

Annexe A : Installation NS2 sous Ubuntu

- Installer les logiciels complémentaires avec la commande suivante :

sudoapt-getinstallbuild-essential autoconfautomakelibxt-dev libx11-dev libxmu-dev

- Installer le compilateur C++ avec la commande suivante:

  sudoapt-getinstall g++-4.3

- Copier le package d'installation« ns-allinone-2.34.tar .gz» dans votre dossier personnel (/home/kamal).
- Extraire le fichier dans le même répertoire personnel.
- Entrer dans le répertoire de ns-allinone-2.34/otcl-1.13 et modifier les fichiers suivants :
  - Le fichier Makefile.in : remplacer

    `CC = @CC@ Par CC = gcc-4.3`
  - Le fichier Configue.in : remplacer CC = @CC@ Par CC = gcc-4.3
- Lancer l'installation de NS2 :

  - cd ns-allinone2.34

  - ./install

  - ./validate
- Editer quelque chemine. Dans le fichier "~/.bashrc"

  - sudogedit .bashrc
- Ajouter les lignes suivantes au fichier : Mettre à la place de « kamal » votre répertoire personnel

```
#environment values for NS2/NAM
#LD_LIBRARY_PATH
OTCL_LIB=/home/kamal/ns-allinone-2.34/otcl-1.13
NS2_LIB=/home/kamal/ns-allinone-2. 34/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR
_LOCAL_LIB
# TCL_LIBRARY
TCL_LIB=/home/kamal/ns-allinone-2.34/tcl8.4.18/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB: $USR_LIB
# PATH
XGRAPH=/home/kamal/ns-allinone-2.34/bin:/home/kamal/ns-allinone-
2.34/tcl8.4.18/unix:/home/kamal/#ns-allinone-2.34/#tk8.4.18/unix
NS=/home/kamal/ns-allinone-2.34/ns-2.34/
NAM=/home/kamal/ns-allinone-2.34/nam-1.14/
```

- Sauvegarder.
- Redémarrer l'ordinateur.
- Taper dans un terminal « ns », il doit s'afficher le symbole « % », dans ce cas le NS2 a été bien installé.
- Installer **xgraph** avec la commande :
  sudoapt-getinstallxgraphygraph.

Annexe B : Intégration du protocole LOCP sous NS2

- L'intégration de protocole LOCP nécessite la modification des fichiers suivants :
  - **$NS_ROOT/Makefile**
  - **$NS_ROOT/queue/priqueue.cc**
  - **$NS_ROOT/common/packet.h**
  - **$NS_ROOT/trace/cmu-trace.h**
  - **$NS_ROOT/trace/cmu-trace.cc**
  - **$NS_ROOT/tcl/lib/ns-packet.tcl**
  - **$NS_ROOT/tcl/lib/ns-lib.tcl**
  - **$NS_ROOT/tcl/lib/ns-agent.tcl**
  - **$NS_ROOT/tcl/lib/ns-mobilenode.tcl**
- Nous commençons avec fichier Makefile `~/ns-allinone-2.34/ns-2.34/`**`Makefile`**, ajouter la line suivant à la ligne 269 :

  `mkdir LOCP.`
- ajouter les lignes suivantes dans le fichier ~/ns-allinone-2.34/ns-2.34/queue/priqueue.cc à la ligne line 93 :

```
// LOCP patch
case PT_LOCP:
```

- modifier le fichier ~/ns-allinone-2.34/ns-2.34/common/**packet.h** :

```
// LOCPpacket
staticconstpacket_t PT_LOCP = 62;

// insert new packet types here
staticpacket_t PT_NTYPE = 63;
```

- Puis ajouter dans la ligne 254 de ~/ns-allinone-2.34/ns-2.34/common/packet.h :

```
type == PT_AODV ||
type == PT_LOCP)
```

- et à la ligne 390 dans le même fichier :

```
// LOCP patch
name_[PT_LOCP] = "LOCP";
```

- Pour ajouter la fonction de trace nous avons ajoutées la ligne suivant dans ~/ns-allinone-2.34/ns-2.34/trace/**cmu-trace.h** à la ligne 163:

```
voidformat_LOCP(Packet *p, int offset);
```

- Dans le fichier ~/ns-allinone-2.34/ns-2.34/trace/**cmu-trace.cc** nous ajoutons les lignes suivantes à la ligne 1071 :

```
// LOCP patch
void
CMUTrace::format_LOCP(Packet *p, int offset)
{
   structhdr_LOCP *wh = HDR_LOCP(p);
   structhdr_LOCP_beacon *wb = HDR_LOCP_BEACON(p);
   structhdr_LOCP_error  *we = HDR_LOCP_ERROR(p);

   switch(wh->pkt_type) {
         caseLOCP_BEACON:

                if (pt_->tagged()) {
                      sprintf(pt_->buffer() + offset,
                                    "-LOCP:t %x -LOCP:h %d
-LOCP:b %d -LOCP:s %d "
                                    "-LOCP:px %d -LOCP:py
%d -LOCP:ts %f "
                                    "-LOCP:c BEACON ",
                                  wb->pkt_type,
                                  wb->beacon_hops,
                                  wb->beacon_id,
                                  wb->beacon_src,
                                  wb->beacon_posx,
                                  wb->beacon_posy,
                                  wb->timestamp);
                } else if (newtrace_) {

                      sprintf(pt_->buffer() + offset,
                                    "-P LOCP -Pt 0x%x -Ph
%d -Pb %d -Ps %d -Ppx %d -Ppy %d -Pts %f -Pc BEACON ",
                                  wb->pkt_type,
                                  wb->beacon_hops,
                                  wb->beacon_id,
                                  wb->beacon_src,
                                  wb->beacon_posx,
                                  wb->beacon_posy,
                                  wb->timestamp);

                } else {

                      sprintf(pt_->buffer() + offset,
                                    "[0x%x %d %d [%d %d]
[%d %f]] (BEACON)",
                                  wb->pkt_type,
                                  wb->beacon_hops,
                                  wb->beacon_id,
```

```
                                        wb->beacon_src,
                                        wb->beacon_posx,
                                        wb->beacon_posy,
                                        wb->timestamp);
                }
                break;

        caseLOCP_ERROR:
                // TODO: need to add code
                break;

        default:
#ifdef WIN32
                fprintf(stderr,
                                "CMUTrace::format_LOCP:
invalidLOCPpackettypen");
#else
                fprintf(stderr,
                                "%s: invalidLOCPpackettypen",
__FUNCTION__);
#endif
                abort();
   }
}
```

- Toujours la modification : ~/ns-allinone-2.34/ns-2.34/tcl/lib/**ns-packet.tcl** @ ligne 172 :

```
# LOCP patch
LOCP
```

- Modification de ~/ns-allinone-2.34/ns-2.34/tcl/lib/**ns-lib.tcl** @ ligne 633 :

```
LOCP {
set ragent [$self create-LOCP-agent $node]
}
```

- Le même fichier ajouté dans la ligne 860 :

```
Simulator instproccreate-LOCP-agent { node } {
#  CreateLOCProuting agent
set ragent [new Agent/LOCP [$nodenode-addr]]
$self at 0.0 "$ragent start"
$node set ragent_ $ragent
return $ragent
}
```

- Modifier ~/ns-allinone-2.34/ns-2.34/tcl/lib/**ns-agent.tcl** line 202 :

```
Agent/LOCPinstprocinitargs {
$self next $args
}

Agent/LOCP set sport_   0
Agent/LOCP set dport_   0
```

- Modifier : ~/ns-allinone-2.34/ns-2.34/tcl/lib/**ns-mobilenode.tcl** line 201 :

```
# Specialprocessing for LOCP
setLOCPonly [string first "LOCP" [$agent info class]]
if {$LOCPonly != -1 } {
$agent if-queue [$self set ifq_(0)]   ;# ifqbetween LL and
MAC
}
```

- Nous avons fini. arrivé à ~ / ns-allinone-2.34 / ns-2.34 / répertoire et faire :

```
MAKE CLEA N
MAKE
```

ملخص

شبكة الاستشعارات اللاسلكية هي عبارة عن مجموعة مستقلة من أجهزة الاستشعار مخصصة لالتقاط الظواهر الفيزيائية في منطقة دراسة ما. أثبتت العديد من الدراسات فوائد استخدام عقدة متنقلة لتقليل استهلاك طاقة و جمع البيانات في شبكة الاستشعارات اللاسلكية (RCSF)

في هذه الحالة، يجب جمع جميع البيانات المكتشفة في وقت معين. تم اقتراح نهج لمواجهة هذا التحدي حيث لا تزور فيه العقدة المتنقلة إلا نقاط اللقاء فقط. و تقوم العقد المستشعرة التي ليست نقطة لقاء بنقل بياناتها بقفزات متعددة إلى أقرب نقطة لقاء.

في عملناهذا، سنركز على انشاء بروتوكول يعتمد على خوارزميات قادرة على تحديد هذه النقاط من أجل جمع المعلومات عن طريق العقد المتنقلة. مع القيام بتحليل أداء هذا البروتوكول بالمحاكاة المنجزة بواسطة المحاكي NS2 .

الكلمات المفتاحية

شبكة الاستشعارات اللاسلكية، NS2، المحاكاة ، جمع المعلومات في شبكة الاستشعارات اللاسلكية، عقدة متنقلة، القاعدة المركزية، TSP ، نقطة لقاء RP ، WPR ، TCL، LOCP، RCSFs.

Abstract

A WSN is an autonomous set of sensor nodes dedicated to the capture of physical phenomena in an area of interest. Several studies have demonstrated the benefits of using a mobile sink to reduce node energy consumption and collect data in (WSN). In this case, all the detected data must be collected in a given time constraint. An approach proposed to meet this challenge in which a mobile sink will only visits meeting points (RP). The Sensor nodes that are not RPs transmit their multi-hop detected data to the nearest RP.

In this work, we will focus on the implementation of a collection data protocol based on algorithms that can choose those RP nodes, in order to collect the data by the mobiles sinks, and to make the analysis of the performance of this Protocol by the simulation established with the network simulator 2 (NS2).

Keywords:

WSN, NS2, simulation, data collection in WSN, mobile sink, base station, TSP, rendez-vous points RP, WPR,TCL, LOCP.

Résumé

Un RCSF est un ensemble autonome des nœuds capteurs dédiés à la capture des phénomènes physiques dans une zone d'intérêt. Plusieurs études ont démontré les avantages de l'utilisation d'un puits mobile pour réduire la consommation d'énergie des nœuds et pour collecter des données dans (RCSF). Dans ce cas, toutes les données détectées doivent être collectées à base d'une contrainte de temps donnée. Une approche proposée pour relever ce défi dans laquelle un puits mobile ne visite que des points de rendez-vous (PR). Les nœuds de capteurs qui ne sont pas des PR transmettent leurs données détectées par multi-sauts au plus proche PR.

Dans ce travail, nous allons se focaliser sur l'implémentation d'un protocole de collection des données basé sur des algorithmes capables de choisir ces nœuds PR pour l'objectif de collection des données avec puits mobiles, et de faire l'analyse des performances de ce protocole par la simulation établie avec le network simulateur 2 (NS2).

Mots-clés:

RCSF, NS2, simulation, collection des données dans RCSF, puits mobile, station de base, TSP, points de rendez-vous PR, WPR, TCL, LCOP.