REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : rtic 13 /M2/2018

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Réseaux et technologies de l'information et de communication**

Modélisation, vérification et analyse des performances du protocole « Directed Diffusion » (Les réseaux de capteurs sans fils)

Par :

HAFEDH SAOUDI MABROUK

Soutenu le 25/06/2018, devant le jury composé de :

Bouchana Belkacem	M.C.A	Président
Djaber Khaled	M.A.A	Rapporteur
Ayad Soheyb	M.C.B	Examinateur

Remerciements

Avant tout, notre sincère louange à ALLAH le tout puissant qui nous a donné la fois, la volonté, la sante et la patience afin d'accomplir ce mémoire.

Nous remerciements les plus distingués à Mr. djaber khaled qui nous a honorés par son encadrement, par sa présence toujours avec nous, pour sa direction, son orientation, sa modestie, ses conseils et toutes ces remarques constructives pour le bon déroulement de notre projet.

Nous remercions beaucoup l'ensemble des enseignements du département d'informatique pour la formation qu'ils nous ont assurée tout le long de notre cursus universitaire, Enfin, nous adressons aussi un grand merci à tous les personnels du département d'informatique.

Résumé

Les réseaux de capteurs sans fils (RCSF) sont utilisés de plus en plus dans notre vie quotidienne ainsi que dans les domaines médicaux, militaires, agricoles et technologiques. Pour faire fonctionner un RCSF, on doit définir des protocoles de communication. Ces protocoles doivent garantir un bon fonctionnement de tels réseaux, mais ce n'est pas toujours le cas. On doit donc être sûr que l'information circule exactement comme prévu dans le réseau. Pour garantir vraiment un bon fonctionnement de ces réseaux, on doit faire subir ces protocoles à des tests.

Le meilleur moyen pour tester ces protocoles est l'utilisation des méthodes formelles. Nous avons spécifié formellement le protocole de routage «Directed Diffusion» en utilisant l'outil UPPAAL qui adopte le modèle des automates temporisés comme un modèle formel de spécification. Nous avons aussi passé ce modèle par le Model Checker de l'outil UPPAAL pour vérifier certaines propriétés exprimées dans la logique temporelle.

Afin de valider notre travail, nous avons analysé les performances de ce protocole par UPPAAL puis nous avons les comparés avec ceux du simulateur NS2 qui est l'un des simulateurs les plus utilisés dans le domaine des réseaux de communications.

Mots clés : Réseaux de capteurs sans fils, méthodes formelles, automates temporisés, Model Checking (Checker), UPPAAL, NS-2, analyse de performances, protocole de routage, Directed Diffusion.

Table des matières

Chapitre 1 : Les réseaux de capteur sans fils

Chapitre2 :Les Protocoles de routage dans un RCSF

Chapitre 3 : Modélisation du protocole de routage Directed Diffusion dans les outils «NS2 et UPPAAL»

Chapitre 4 : vérification et analyse de performance du protocole de routage «Directed Diffusion»

Liste des tables

Liste des Figures

Introduction générale

Les réseaux de capteurs sans fil sont considérés comme un type spécial de réseaux ad hoc, composés d'un grand nombre de capteurs matériellement petits, et placés généralement près des objets auxquels ils s'intéressent dans les environnements où ils sont déployés. Ces capteurs sont capables de récolter, traiter et acheminer les données environnementales de la région surveillée d'une manière autonome, vers des stations de collecte appelées nœuds puits ou stations de base.

Les réseaux de capteurs sans fils sont utilisés dans tous les domaines technologiques, médicaux et militaires. L'utilisation intense de ce type de réseaux nous a poussé à établir des protocoles de communication avec lesquels on garantit la sécurité, la fiabilité et la rapidité de la communication au sein d'un réseau de capteurs sans fils. Plusieurs protocoles de communication ont été proposés pour les réseaux de capteurs sans fils. Les applications dans lesquels ces capteurs sont impliquées peuvent être des applications de genre critique, où la fiabilité est une exigence. Le fonctionnement du système et des protocoles de communication dans ce système doivent être assurés. Pour assurer le bon fonctionnement de ce type d'applications critiques, on fait généralement appel à la vérification formelle. Les méthodes de spécification formelle peuvent être utilisées pour spécifier un protocole de communication dans un RCSF en tenant compte d'un ensemble de contraintes, de spécifier les propriétés exigées par l'utilisateur et par la suite, de vérifier que la spécification du protocole à développer satisfait les exigences exprimées par l'utilisateur.

L'objectif de notre travail est de modéliser, vérifier et analyser les performances du protocole de communication basé données « Directed Diffusion » en utilisant le simulateur de réseaux NS-2 d'une part et de l'autre part l'outil de vérification formelle UPPAAL. Ce protocole est basé sur l'échange de messages via le réseau sans fils. Cet échange passe par 03 phases qui sont : l'envoie d'intérêt, capture d'évènements et le renforcement positif d'un chemin optimal.

Une comparaison entre les résultats obtenus sera faite pour valider notre modélisation sous l'outil UPPAAL. Une discussion des résultats sera indispensable pour nous permettre de tirer des conclusions sur notre travail.

Pour atteindre notre objectif, nous avons organisé notre mémoire comme suit :

1er chapitre : il est consacré à la définition des réseaux de capteurs sans fils, leurs domaines d'utilisation, et les protocoles de communication proposés.

2ème chapitre : il est consacré aux protocoles de routage et la description détaillée du protocole « Directed Diffusion ».

3ème chapitre : représente notre contribution, il est consacré à la modélisation du protocole en utilisant les outils NS2 et UPPAAL.

4ème chapitre : représente une comparaison des résultats obtenus, après la vérification de quelques propriétés de ce protocole et l'analyse de ses performances.

Chapitre 1:

Les réseaux de capteur sans fils

1.1. Introduction:

La technologie sans fil devient de plus en plus très importante dans le domaine de la télécommunication par rapport aux réseaux filaires. Cette technologie permet aux utilisateurs d'accéder à l'information d'une manière souple et facilite la manipulation des informations en utilisant des unités de calcul mobiles tel que les PCs portables, les PDA et les capteurs. Ces derniers, représentent le futur des réseaux de communication. Les réseaux de capteurs sont le sujet de plusieurs projets de recherche afin de maîtriser cette technologie et les protocoles de communication nécessaires pour un bon fonctionnement au sein de ces réseaux. Dans ce chapitre, nous allons présenter les réseaux de capteurs sans fil et leurs domaines d'utilisation, les défis de leurs conceptions, la topologie et la pile protocolaire d'un réseau de capteur sans fils.

1.2. 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 communiquer 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 à coût raisonnable des micro-capteurs de quelques millimètres cubes de volume, susceptibles de fonctionner en réseaux [34]. Un capteur est composé de quatre unités de base (voir figure 1.1):

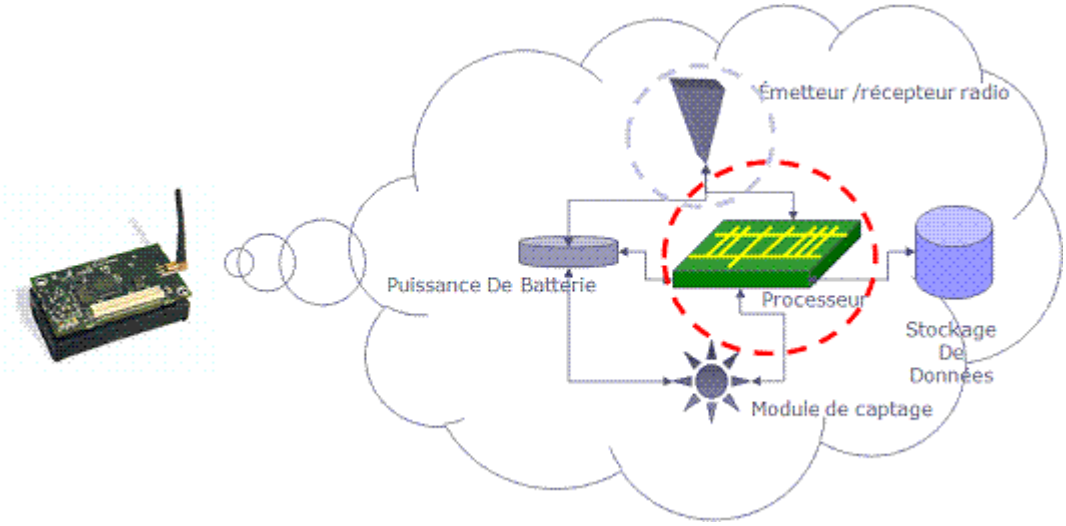

Figure 1.1 Les composants d'un nœud capteur [1]

- **L'unité d'acquisition:** elle est généralement composée de deux sous-unités qui sont les capteurs et les convertisseurs analogique-numérique ADCs (AnalogDigital 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.

- **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 stocker les données collectées.
- **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.
- **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).

- La figure 1.2 représente l'architecture d'un capteur :

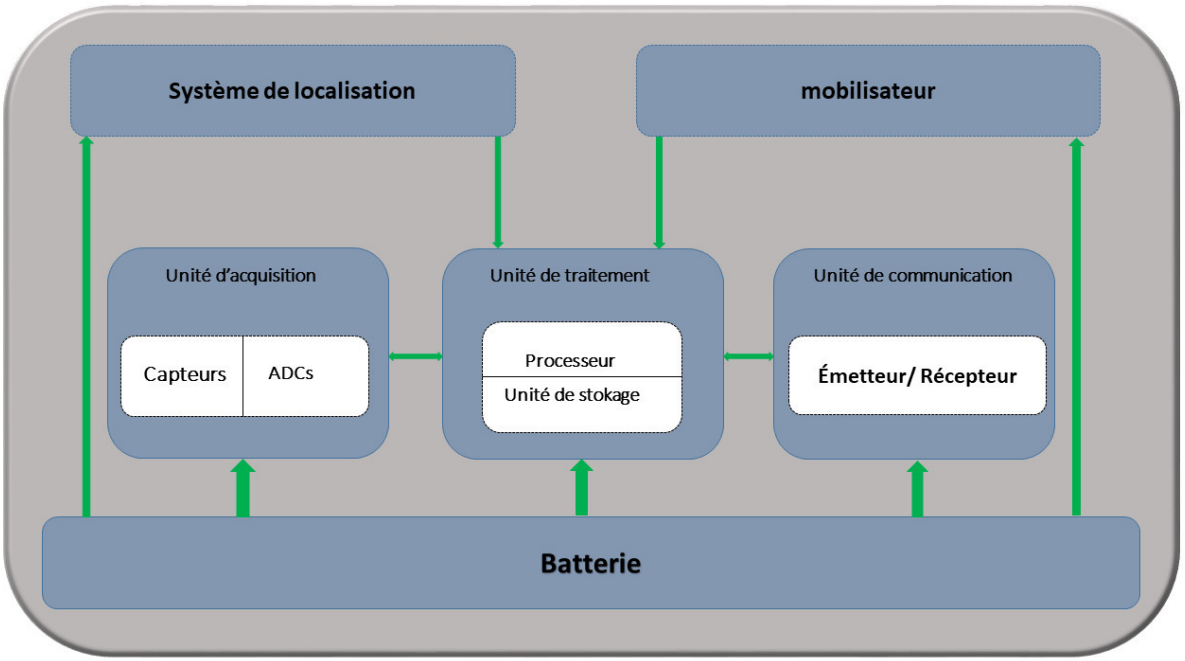

Figure 1.2 Architecture d'un nœud capteur

1.2.1. 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, ...) et l'évolution des supports de communication sans fil ont élargi le champ d'application des réseaux de capteurs. Les RCSF peuvent être utilisés dans plusieurs applications [34,35, 28, 46, 40]. Parmi elles, nous citons :

- **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 [23].

- **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 à recourir à de coûteux dispositifs de surveillance vidéo.

- **Gestion de stock :** on pourrait imaginer devoir 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.

- **Contrôle de la pollution :** des capteurs 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 [55].

- **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 éviterait aux éleveurs d'avoir recours à des chiens de berger.

- **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 [47].

- **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 réguler 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.

1.2.2. 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, 11]. La figure 2 montre un exemple d'un réseau de capteurs.

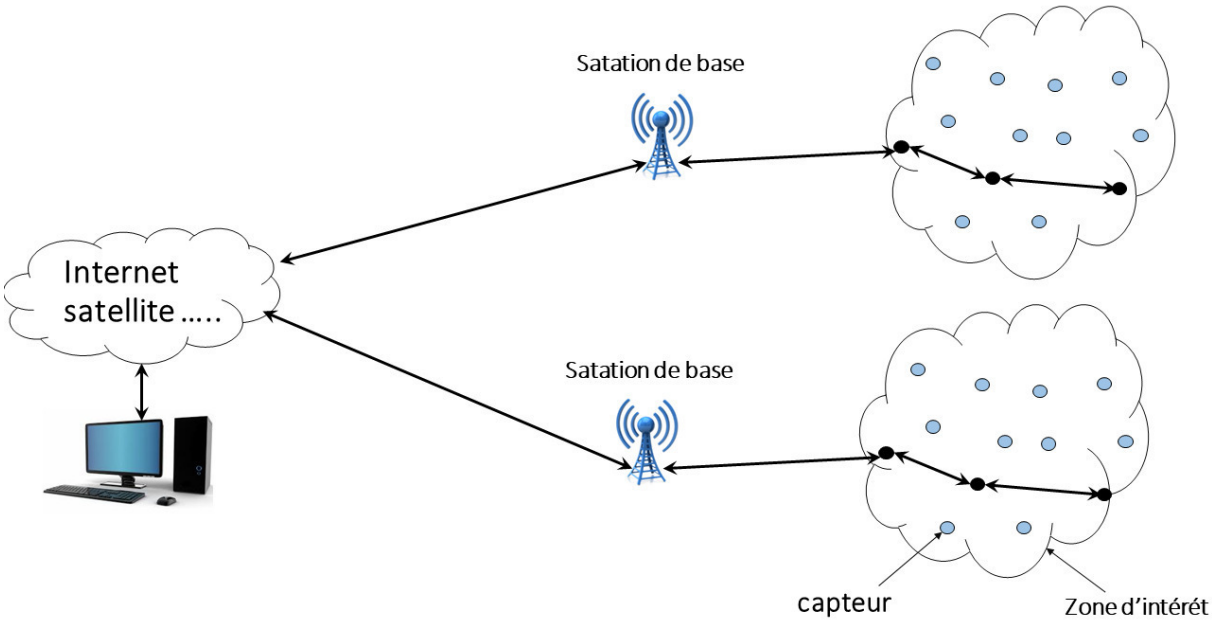

Figure 1.3 Exemple de réseaux de capteurs

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 , nommé aussi "Sink ou root" transmet ensuite ces données par Internet ou par satellite à l'ordinateur central «Gestionnaire de tâches» pour analyser ces données et prendre des décisions.

1.2.3. Contraintes de conception des RCSF

Les principaux facteurs et contraintes influençant l'architecture des réseaux de capteurs peuvent être résumés comme suit [14,21,13,16,15,19]:

- **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.

- **Le facteur d'échelle (Scalability**) : le nombre de nœuds de 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.

- **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.

- **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 [25],... Par conséquent, ils doivent pouvoir fonctionner sans surveillance dans des régions géographiques éloignées.

- **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.

- **Les contraintes matérielles :** la principale contrainte matérielle est la taille du 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.

- **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 [48] et les communications radio Zig Bee .

- **La consommation d'énergie :** un capteur, de par sa taille, est limité en énergie (<1.2V). 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 re-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 réduire cette consommation.

1.3. La topologie d'un réseau de capteur sans fil

Les caractéristiques de déploiement aléatoire, fonctionnement autonome, et fréquence élevé de pannes rendent la maintenance de la topologie d'un réseau de capteurs une tâche complexe. En effet plusieurs centaines de capteurs sont déployés avec une densité pouvant être supérieur à 20 nœuds par m3, ceci exige une bonne gestion de la maintenance de la topologie du réseau déployé. [24] Nous examinons, dans ce qui suit, les différents problèmes liés aux topologies des réseaux de capteurs et leurs changements.

1.3.1. Phase de pré-déploiement et de déploiement

Les nœuds capteurs peuvent être éparpillés sur le champ de captage en masse ou placés d'une manière individuelle et ceci par le biais de plusieurs moyens tels que:

- les jeter d'un avion
- utiliser une artillerie, roquette ou missile.
- les placer nœud par nœud d'une façon manuelle ou en utilisant des robots.

Le nombre important de nœud utilisés dans un réseau de capteurs empêche leur déploiement suivant un plan soigneusement établi, cependant un schéma général pour le déploiement initial doit être conçu pour permettre :

- de réduire le coût d'installation
- augmenter la flexibilité d'arrangement des nœuds
- faciliter l'auto-organisation des nœuds et leur tolérance aux pannes

1.3.2. Phase de post-déploiement

Après la phase de déploiement, la topologie du réseau peut subir des changements dus aux :

- changement de position des nœuds
- accessibilité à cause du brouillage ou des obstacles en mouvements
- épuisement d'énergie
- mal fonctionnement des nœuds ou des besoins pour leur application.

En effet, Bien que les nœuds d'un réseau de capteurs puissent être déployés d'une manière statique, la panne matérielle constitue un évènement très commun à cause de l'épuisement d'énergie ou la destruction. Il est possible également d'avoir un réseau de capteur avec des nœuds mobiles qui ont une mobilité très élevée. Par conséquent, la topologie du réseau de capteur est exposée fréquemment aux changements après la phase de déploiement.

1.3.3. Phase de redéploiement des nouveaux nœuds

Des nœuds capteurs additionnels peuvent être installés pour remplacer ceux qui sont en panne ou bien pour répondre aux besoins des tâches assignées au réseau. Cette addition entraîne la réorganisation du réseau et le changement de sa topologie.

Une bonne gestion du réseau, faisant face au facteur de changement fréquent de la topologie d'un réseau ad hoc caractérisé par une contrainte exigeante de consommation d'énergie doit passer obligatoirement par la conception des protocoles de routages spéciaux, cette problématique sera détaillée dans la section 5.

1.4. La pile protocolaire des RCSF

Il est noté qu'aucune pile protocolaire destinée aux RCSF n'a été standardisée. Cependant, la majorité des articles scientifiques, qui traitent la thématique des RCSF, se basent sur la pile protocolaire qui a été proposée dans [39]:

-Une couche physique.
-Une couche liaison de données.
-Une couche réseaux.
-Une couche transport.
-Une couche application.
-Un plan de gestion d'énergie.
-Un plan de gestion de mobilité.
-Un plan de gestion tâches.

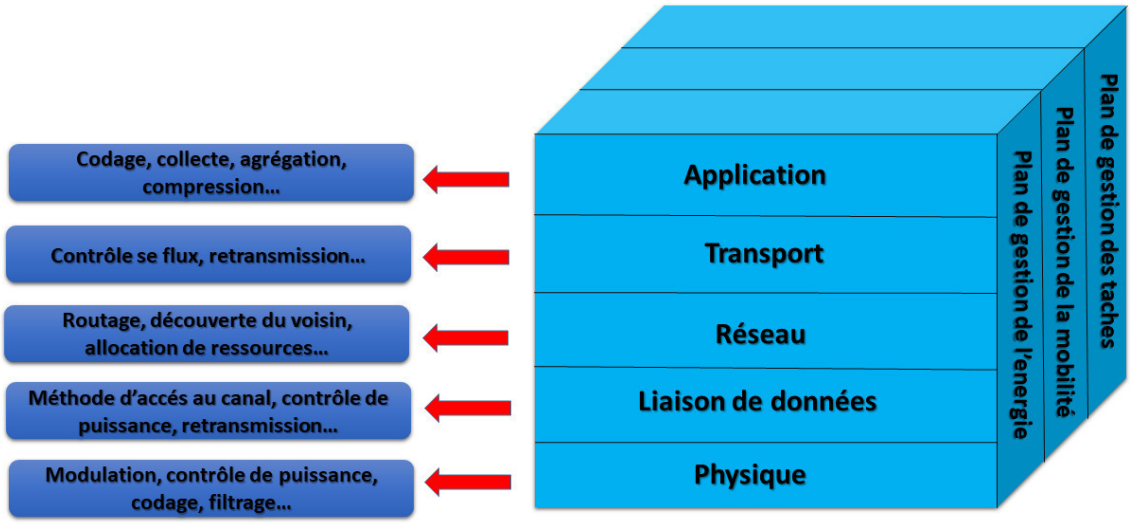

Figure 1.4 Pile protocolaire dans les réseaux de capteurs [39]

Nous nous intéressons aux trois premières couches:

1.4.1 La couche physique

Elle est responsable de la sélection de fréquence, la génération de la fréquence porteuse, la détection du signal, la modulation/démodulation et le cryptage/décryptage des informations. La consommation d'énergie au niveau de la couche physique peut être affectée par l'environnement de l'application, le choix du type de la modulation ou la bande de fréquence utilisée. Il est avantageux en manière d'économie d'énergie que le concepteur de la couche physique choisisse une transmission à multi-sauts plutôt qu'une transmission directe qui nécessite une puissance de transmission très élevée [39].

1.4.2. La couche liaison de données

Elle est responsable de la détection des trames de données, le contrôle d'accès au support (MAC) et le contrôle d'erreurs. Elle maintient aussi la fiabilité des connections point à point ou multipoints dans les RCSF [39].
La couche liaison de donnée contient deux sous-couches qui sont :

-**La sous-couche MAC :** Dans un RCSF, la couche MAC doit accomplir deux principales tâches qui sont celles de :

- établir des liaisons de communication entre les nœuds capteurs pour effectuer le transfert des données et permettre au réseau la capacité de s'auto-organiser.
- décider du moment et de la manière dont les nœuds capteurs peuvent accéder au canal avec un minimum de perte d'énergie [39].

-**la sous-couche de contrôle d'erreurs:** La technique de contrôle d'erreurs la plus utilisée dans les RCSF est le <Forward error correction> (FEC) ; Cette technique comporte de simples mécanismes de codage et de décodage (code de contrôle d'erreurs simples) [39].

1.4.3. La couche réseaux [29]

La couche réseau gère les échanges (et éventuellement les connexions) au travers du RCSF. Elle gère entre autre l'adressage et l'acheminement des données. Les applications des RCSF requièrent le plus souvent des protocoles de routage à multi sauts entre le nœud émetteur, le ou les nœuds relais et le nœud <Sink>. Les protocoles de routage traditionnels des réseaux ad hoc ne peuvent pas être utilisés dans les RCSF puisqu'ils ne satisfont pas les critères de conservation d'énergie et de scalabilité [29].

Les métriques considérées par les chercheurs pour déterminer la route la plus optimisée dans les réseaux RCSF sont :

-L'énergie nécessaire pour transmettre le paquet d'une manière fiable.
-L'énergie disponible dans chaque nœud capteur.

Les algorithmes de routage peuvent alors sélectionner les routes entre le nœud émetteur et le nœud <Sink> en se basant soit sur le maximum d'énergie disponible au niveau des nœuds intermédiaires, soit sur la route qui consomme le moins d'énergie pour transmettre d'un nœud vers un autre. Le type d'adressage le plus utilisé dans les RCSF est l'adressage géographique, c'est-à-dire que chaque nœud capteur est identifié dans le réseau par sa localisation [29].

1.5. Conclusion:

Dans ce chapitre, nous avons présenté les principaux éléments qui définissent un réseau de CSF. Ces éléments sont principalement : les composants d'un capteur, les applications d'un RCSF, les contraintes de conception et la topologie d'un RCSF.

Cette topologie engendre généralement un choix précis du protocole de routage, ainsi que nous allons présenter par conséquent les principaux protocoles de routage pour un RCSF, sous une classification selon le point d'intérêt.

Nous allons présenter dans le chapitre suivant le principe général d'un protocole de communication au sein d'un RCSF.

Chapitre 02 :

Les protocoles de routage dans un RCSF

2.1. Introduction :

Après avoir défini un réseau de capteurs, on abordera dans ce chapitre la communication entre les nœuds de ce réseau, en effet il existe plusieurs protocoles de routages avec chacun sa spécificité.

Le protocole de routage nommé « Directed Diffusion » est un protocole dont le principe est de trouver le meilleur chemin pour diriger les données d'un nœud source vers le nœud « Sink » (ou station de base). C'est ce que nous verrons par la suite dans un exemple concret.

2.2 Les principaux protocoles de routage dans les RCSF

La figure suivante résume les principaux protocoles de routage [12]:

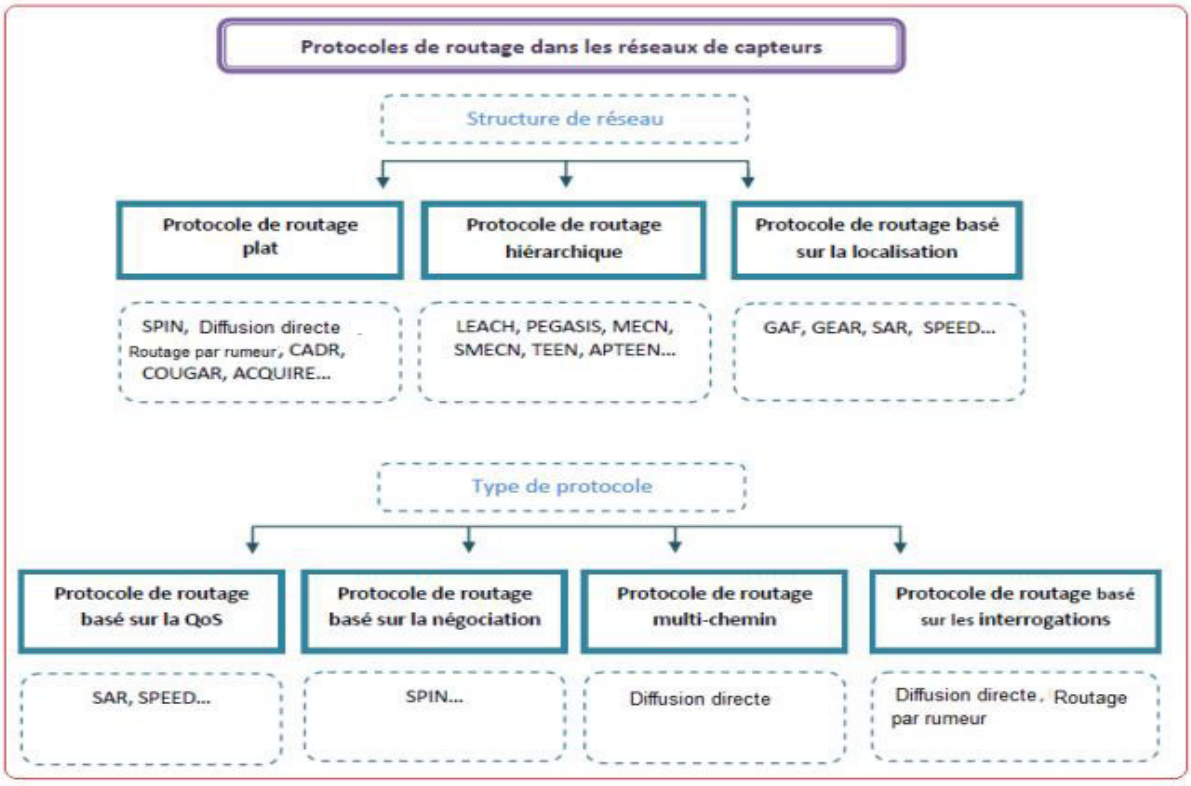

Figure 2.1 Les principaux protocoles de routages dans les RCSF

2.2.1. Les protocoles de routage basés sur la localisation

Les protocoles de routage basés sur la localisation [18] 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.

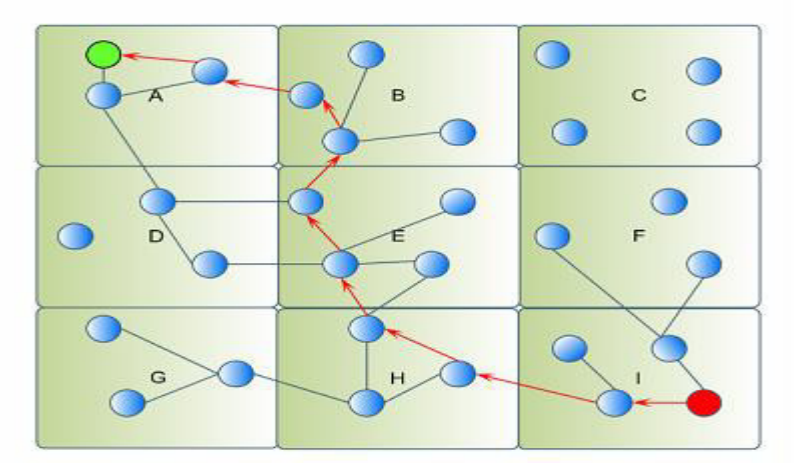

Figure 2.2 Topologie Basée Localisation [5]

2.2.1.1. Le protocole de routage « MECN »

Minimum Energie Communication Network (MECN) [52] est un protocole de routage qui cherche à établir et à entretenir une énergie minimale pour les réseaux sans fil en utilisant des GPS de faible puissance. MECN utilise une station de base comme destination de l'information, ce qui est toujours le cas pour les réseaux de capteurs. MECN identifie une région de relais pour chaque nœud. La région de relais se compose de nœuds dans une zone périphérique où la transmission à travers ces nœuds est plus économe en énergie que la transmission directe. L'idée principale de MECN est de trouver un sous-réseau qui a moins de nœuds et qui nécessite moins d'énergie pour la transmission entre deux nœuds quelconques. Cela est effectué en utilisant une recherche localisée pour chaque nœud en prenant en considération sa région de relais [18].

2.2.1.2. Le protocole de routage « GAF »

GAF (Geographic Adaptive Fidelity) [44] est un protocole de routage basé sur la localisation des nœuds. Il est conçu principalement pour les réseaux mobiles ad hoc, mais peut être applicable aux réseaux de capteurs. La localisation des nœuds dans GAF pourrait être fournie à l'aide d'un GPS ou d'autres techniques de localisation [18, 42, 37]. Il consiste à former des grilles virtuelles de la zone concernée en partitionnant cette zone où les nœuds sont déployés en de petites zones telles que,

pour deux grilles adjacentes Gx et Gy, tous les nœuds de Gx peuvent communiquer avec tous les nœuds Gy. Ainsi, ce système de partitionnement GAF assure la fidélité du routage car il existe au moins un chemin entre un nœud et la station de base.

GAF peut augmenter considérablement la durée de vie du réseau. En effet, un seul nœud dans chaque grille reste à l'état actif en faisant passer les autres nœuds de la grille à l'état de sommeil pour une certaine période de temps tout en assurant la fidélité du routage. Cependant, dans certains environnements où les nœuds sont fortement mobiles, la fidélité du routage pourrait être réduite si un nœud actif quitte la grille. Ainsi, le nombre de données perdues sera important.

2.2.1.3. Le protocole de routage «GEAR »

Le protocole de routage GEAR (Geographic and Energy Aware Routing) [43, 53, 9] a été suggéré par Y. Yu et D. Estrin. Il consiste à utiliser l'information géographique lors de la diffusion des requêtes aux régions cibles car les requêtes contiennent souvent des données géographiques. L'idée est de restreindre le nombre de données dans la diffusion dirigée en prenant en considération uniquement une certaine région, plutôt que d'envoyer les données à l'ensemble du réseau.

Avec le protocole GEAR, chaque nœud maintient le coût pour atteindre la destination en passant par ses voisins. Ce coût est divisé en deux parties : un coût estimé et un coût d'apprentissage. Le coût estimé est une combinaison de l'énergie résiduelle et de la distance jusqu'à destination. Le coût d'apprentissage est un raffinement du coût estimé qu'un nœud dépense pour le routage autour des trous dans le réseau. Un trou se forme quand un nœud n'a pas de voisin proche par lequel il peut atteindre la région cible. S'il n'y a pas de trous, le coût estimé est égal au coût d'apprentissage. Le coût d'apprentissage se propage d'un saut à chaque fois qu'un paquet atteint la destination [18, 43].

2.2.2. Les Protocoles hiérarchiques

L'objectif principal du routage hiérarchique [18, 20] est de maintenir efficacement la consommation d'énergie de nœuds de capteurs en les impliquant dans la communication multi-hop 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 2.3). LEACH (Low Energy Adaptive Clustering Hierarchical) [12] 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 aient été développés de manière indépendante.

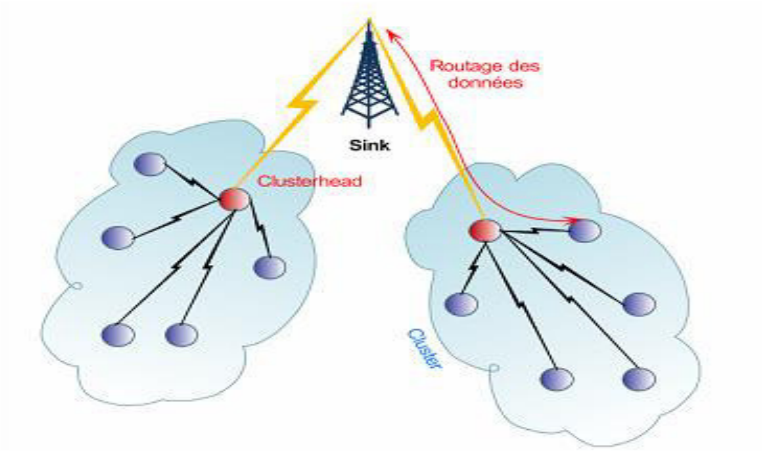

Figure 2.3 Topologie hiérarchique (LEACH) [5]

2.2.2.1. Le protocole de routage «LEACH »

LEACH est l'un des algorithmes de routage hiérarchique le plus populaire pour les réseaux de capteurs [12, 18, 45]. L'idée est de former des clusters de nœuds de capteurs basés sur les zones où il y a un fort signal reçu, puis utiliser des clusters-heads locaux comme passerelle pour atteindre la destination. Cela permet d'économiser de l'énergie car les transmissions ne sont effectuées que par les cluster-head plutôt que par tous les nœuds de capteurs.

2.2.2.2. Les protocoles de routage «PEGASIS & Hierarchical-PEGASIS»

Power-Efficient GAthering in Sensor Information Systems (PEGASIS) [18] est une version améliorée du protocole LEACH. PEGASIS forme des chaînes plutôt que des clusters de nœuds de capteurs afin que chaque nœud transmette et reçoive uniquement des données d'un voisin. Un seul nœud est sélectionné à partir de cette chaîne pour transmettre à la station de base. L'idée de PEGASIS est qu'il utilise tous les nœuds pour transmettre ou recevoir des données avec ses plus proches voisins. Il déplace les données reçues de nœud à nœud, puis les données seront agrégées jusqu'à ce qu'elles atteignent tous la station de base. Donc, chaque nœud du réseau est tour à tour un chef de file de la chaîne, ainsi que responsable pour transmettre l'ensemble des données recueillies et fusionnées par la chaîne de nœuds au niveau de la station de base [31].

2.2.2.3. Les protocoles de routage «TEEN et APTEEN»

Les protocoles Threshold sensitive Energy Efficient sensor Network protocol (TEEN) [41] et Adaptive Threshold sensitive Energy Efficient sensor Network protocol (APTEEN) [22] conviennent pour les applications critiques. Dans les deux protocoles, le facteur clé est la valeur de l'attribut mesuré. La caractéristique supplémentaire d'APTEEN est la capacité de changer la périodicité et les paramètres de TEEN en fonction des besoins des utilisateurs et des applications.

TEEN est conçu pour être sensible à des changements soudains des attributs tels que la température. La réactivité est importante pour les applications critiques dont le réseau fonctionne dans un mode réactif. L'architecture du réseau de capteurs est basée sur un groupement hiérarchique où les nœuds forment des clusters et ce processus va se répéter jusqu'à ce que la station de base soit atteinte [18] .

APTEEN est une extension de TEEN qui fait à la fois la collection des captures périodique de données et qui réagit aux événements critiques. Quand la station de base forme des clusters, les clusters head diffusent les attributs, les valeurs des seuils, ainsi que le calendrier de transmission à tous les nœuds. Le cluster-head effectue également l'agrégation de données afin d'économiser l'énergie.

2.2.3. Les protocoles de routage plate '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 de capteurs font qu'il est difficile de sélectionner un ensemble spécifique de nœuds de capteurs à interroger. Par conséquent, les données sont généralement transmises de chaque nœud de capteurs dans la région de déploiement avec une redondance importante. Cette réflexion a conduit au routage data-centric [38] 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[18]. Comme le montre l'exemple d'une approche data-centric dans la figure 5, les données provenant des deux sources sont agrégées au nœud B.Ensuite, la donnée combinée (1+2) est envoyée de B vers la destination.

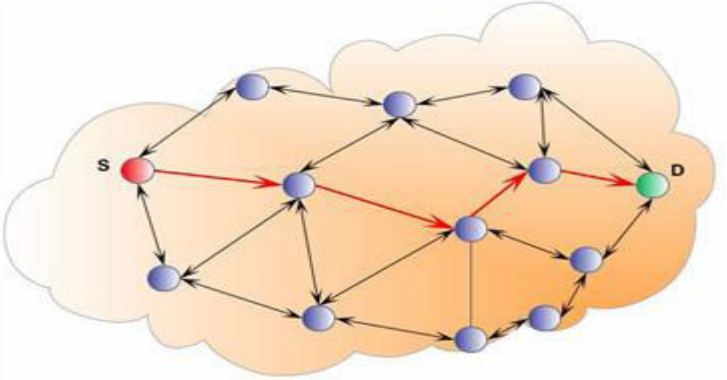

Figure 2.4 Topologie plate (Flat) [5]

2.2.3.1. Le protocole de routage « SPIN »

Un réseau de capteurs sans fil se compose de plusieurs capteurs qui sont déployés dans différentes régions. L'accès aux données d'un événement particulier pour une région ou une zone spécifique peut faire une grande différence. Des mesures peuvent être prises pour l'obtention de données à partir d'un endroit ciblé. Les données sont recueillies auprès de tous les capteurs et transmises de manière redondante sur le réseau, ce qui entraîne une utilisation inefficace de l'énergie. Afin de résoudre ces types de problèmes, les systèmes de routage data-centric ont évolué : la destination envoie des requêtes pour les capteurs du réseau dans un endroit choisi. Les attributs sont utilisés pour demander des données provenant des capteurs. SPIN (Sensor Protocols for Information via Negotiation) [54] est le premier protocole data-centric qui a été conçu pour les réseaux de capteurs sans fil. Il présente de nombreuses similitudes par rapport à la diffusion dirigée. Il est efficace dans la réduction des données redondantes et économise l'énergie [18].

La diffusion est le processus de collecte des observations de l'ensemble des capteurs individuel qui sont déployées dans le réseau et où tous les capteurs sont traités comme des nœuds destination [36]. Les tâches confiées à ces capteurs sont de recueillir le point de vue complet de l'environnement sous la forme de données et de renforcer une structure de réseau avec une tolérance aux fautes. La consommation d'énergie durant les calculs et les communications doit être contrôlée afin de prolonger la durée de vie des capteurs au sein du réseau.

2.2.3.2. Le protocole de routage par rumeur

Le routage par rumeur [51, 36] est principalement destiné pour des applications où le routage géographique n'est pas faisable. En général, la méthode (diffusion dirigée) utilise l'inondation pour envoyer la requête à l'ensemble du réseau où il n'y a pas de critère géographique pour diffuser les tâches. Toutefois, dans certains cas, peu de données sont demandées par les nœuds, donc l'utilisation d'inondation est inutile.

L'idée clé de cette méthode est de trouver les routes pour les requêtes vers les nœuds qui ont observé un événement particulier, plutôt que d'inonder tout le réseau pour récupérer des informations sur les événements survenus. Afin de diffuser un événement sur le réseau, l'algorithme de routage par rumeur emploie des paquets appelés agents. Quand un nœud détecte un événement, il ajoute cet événement à sa table locale, appelée table d'événements et génère un agent. Cet agent parcourt le réseau afin de propager des informations sur des événements locaux pour les nœuds distants. Quand un nœud génère une requête pour un événement, les nœuds qui connaissent le chemin, répondent à la requête en inspectant leur table événement. Par conséquent, il n'est pas nécessaire d'inonder tout le réseau, ce qui réduit le coût de communication. D'autre part, ce routage n'utilise qu'un seul chemin entre la source et la destination au lieu de la diffusion dirigée où les données peuvent être acheminées par des routes multiples.

Les résultats de simulation ont montré que le routage par rumeur peut réaliser des économies d'énergie signifiantes par rapport à la méthode d'inondation et peut également préserver la vie du nœud. Toutefois, le routage par rumeur fonctionne bien uniquement lorsque le nombre d'événements est faible. Pour un grand nombre d'événements, le coût du maintien des agents et des tables d'événements de chaque nœud devient impossible.

La table représente un comparatif de quelques protocoles de routage dans les RCSFs. [7]

	Classification	Mobilité	Basé sur la négociation	Agrégation	Mise à l'échelle	Multi chemin	L'utilisation d'énergie
LEACH	Hiérarchique	Station de base fixe	Non	Oui	Bonne	Non	Max
PEGASIS	Hiérarchique	Station de base fixe	Non	Non	Bonne	Non	Max
Hierarchical-PEGASIS	Hiérarchique	Station de base fixe	Non	Non	Bonne	Non	Max
TEEN and APTEEN	Hiérarchique	Station de base fixe	Non	Oui	Bonne	Non	Max
MECN	Hiérarchique	Non	Non	Non	Faible	Non	Min
GAF	Géographique	Limitée	Non	Non	Bonne	Non	Limitée
GEAR	Géographique	Limitée	Non	Non	Limitée	Non	Limitée
SPIN	Plat	Possible	Oui	Oui	Limitée	Oui	Limitée
La diffusion dirigée	Plat	Possible	Oui	Oui	Limitée	Oui	Limitée
routage par rumeur	Plat	Très limitée	Non	Oui	Bonne	Non	Min
GBR	Plat	Limitée	Non	Oui	Limitée	Non	Min

Table 2.1. Classification et comparaison des protocoles de routages dans RCSF

2.2.3.3. La diffusion dirigée

La diffusion dirigée [49, 50] est un protocole important dans le routage data-centric des réseaux de capteurs. L'idée vise à diffuser des données aux nœuds en utilisant un schéma de nommage pour les données. La raison principale derrière l'utilisation d'un tel système est de se débarrasser des opérations inutiles de routage de couche réseau afin d'économiser l'énergie.

La diffusion dirigée suggère l'utilisation de paires attribut-valeur pour les données et les requêtes des capteurs. Afin de créer une requête, un nœud est

défini à l'aide d'une liste de paires attribut-valeur comme le nom des objets, l'intervalle, la durée, la zone géographique, etc. Un paquet est diffusé par ce nœud vers la destination à travers ses voisins. Chaque nœud qui reçoit les paquets peut les stocker pour une utilisation ultérieure. Les paquets stockés sont ensuite utilisés pour comparer les données reçues. La requête contient aussi plusieurs champs de gradient. Un gradient est un lien réponse avec un voisin dont le paquet a été reçu et qui est caractérisé par le débit, la durée et la date d'expiration de données. Ainsi, en utilisant les intérêts et les gradients, les routes sont établies entre la destination et les sources. Plusieurs routes peuvent être établies de telle sorte que l'une d'elle est choisie par renforcement. La destination renvoie le message d'intérêt initial à travers la route choisie. Un intervalle plus petit renforce donc le nœud source sur ce chemin pour envoyer des données plus fréquemment [18].

2.3. Fonctionnement du protocole « DIRECTED DIFFUSION» [10]

Directed Diffusion (DD) fut l'un des premiers protocoles centrés-données. Il a été proposé par C.Intanagonwiwat, R.Govindan et D.Estrin [36] et il est devenu l'un des protocoles les plus répandus dans les réseaux de capteurs. Sa création représente une importante avancée dans le domaine du routage. Il a été une base pour la conception de plusieurs protocoles de routage pour les réseaux de capteurs. [10]

Il n'y a pas meilleur manière de comprendre le fonctionnement de la communication entre les nœuds capteurs que par un exemple, ce dernier définit les différentes étapes d'établissement de chemin de routage de ce protocole.

> *** Scénario** : Dans ce qui suit, nous prenons comme exemple, un réseau de capteurs connu pour détecter des cibles dans une région donnée. L'utilisateur de ce réseau le charge d'effectuer la tâche suivante : "Fournir, durant les T prochaines secondes et toutes les 10 ms, la position de n'importe quel véhicule roulant se trouvant dans la sous-région R du champ de captage".

Nous détaillerons, ci-après, les étapes du fonctionnement du protocole Directed Diffusion pour la dissémination des données.

2.3.1. Dissémination des intérêts et établissement des gradients

Le collecteur (nœud Sink) commence par envoyer périodiquement un message d'intérêt.

Cette tâche est définie par un ensemble de paires "attributs-valeurs". Une description simplifiée de l'exemple précédent pourrait être de la forme suivante :

Attribut d'intérêt	Description
Type= véhicule roulant	Le type de cible à détecter
Intervalle= 10ms	Envoyer des événements toutes les 10 ms
Durée= 10mn	Durant les prochaines 10 minutes
Rect= [0, 100, 200, 400]	Par les nœuds de cette région

Table 2.2 Les différents attributs d'un intérêt

Le message ainsi décrit permet de spécifier les données par lesquelles l'utilisateur est intéressé. Pour cette raison, il est appelé "intérêt". Il peut contenir différents attributs. L'exemple précédent spécifie : le type de cible détectée, le débit (l'intervalle) avec lequel les réponses doivent être envoyées, la durée de validité de l'intérêt, et la sous-région de captage concernée par l'intérêt.

Au départ, l'intérêt est diffusé par le nœud collecteur à tous ses voisins. Le débit demandé initialement par l'intérêt envoyé est plus faible que celui demandé par l'utilisateur (exemple : intervalle = 1s, au lieu de : intervalle = 10ms).

Chaque nœud du réseau maintient à son niveau un cache qui a pour but de garder trace des intérêts reçus. Chaque entrée du cache correspond à un intérêt distinct et contient plusieurs champs de gradients qui identifient les voisins depuis lesquels l'intérêt a été reçu. Un gradient spécifie une valeur et une direction dans laquelle les données répondant à l'intérêt seront envoyées. Le choix de la sémantique de la valeur associée au gradient dépend de l'application du réseau de capteurs. Dans l'exemple cité précédemment, cette valeur représente le débit avec lequel les données seront envoyées. Dans d'autres applications, cette valeur peut, par exemple, être une probabilité p avec laquelle une donnée est envoyée vers le voisin.

Chaque gradient possède une certaine durée de vie qui représente la durée de vie de l'intérêt associé. Quand un gradient expire, il est retiré de l'entrée d'intérêt. Quand tous les gradients d'une entrée d'intérêt expirent, l'entrée elle-même est supprimée du cache.

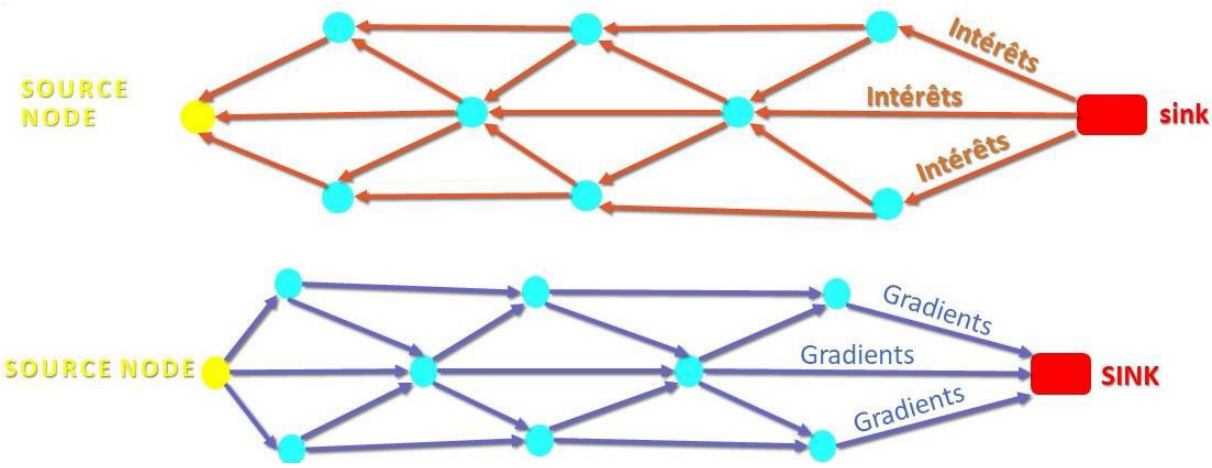

Figure 2.5 Propagation des intérêts et établissement des gradients

Quand un nœud reçoit un intérêt, il vérifie son cache d'intérêts. Si aucune entrée correspondante à l'intérêt reçu n'existe (il est différent des intérêts du cache), le nœud crée une nouvelle entrée dans son cache, dans laquelle il va définir un gradient vers le voisin à partir duquel l'intérêt a été reçu. S'il existe une entrée correspondante, mais aucun gradient pour l'émetteur de l'intérêt, le nœud ajoute un gradient vers le voisin émetteur. Finalement, s'il existe déjà une entrée et un gradient pour l'émetteur, le nœud les met simplement à jour.

Par la suite, chaque nœud ayant reçu un intérêt diffuse ce dernier à tous ses voisins. Bien que l'intérêt soit initialement généré par le nœud collecteur, chaque nœud émetteur semble être l'origine de cet intérêt auprès de ses voisins récepteurs.

De cette manière, l'intérêt est diffusé dans tout le réseau par la technique d'inondation. Cette technique peut causer une forte dépense en énergie. Cependant, l'inondation de tout le réseau est inévitable en l'absence d'informations sur les nœuds capteurs susceptibles de satisfaire l'intérêt.

Ainsi, un intérêt est disséminé à partir du nœud collecteur sur tout le réseau. Cette dissémination installe des gradients dans chaque nœud du réseau pour orienter les données de réponses à l'intérêt inondé, vers le nœud collecteur , telle que montre la (figure 2.6)

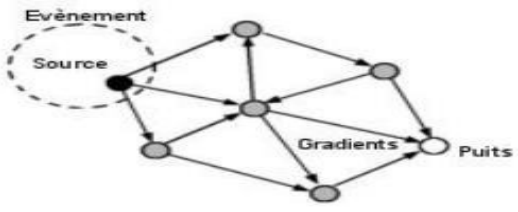

Figure 2.6 Etablissement des gradients [10]

Notons que le collecteur rediffuse périodiquement chaque intérêt. Ceci est nécessaire pour rétablir les gradients en cas de changements topologiques dans le réseau (défaillances de liens, mobilité, pannes de nœuds, etc.). Le taux de rafraîchissement d'intérêt est un paramètre de conception du protocole soumis à un compromis entre la charge du réseau et la fiabilité de transmission.

2.3.2. Propagation des données

Quand un nœud capte un évènement dans son champ de déploiement, il consulte son cache d'intérêts pour rechercher un enregistrement correspondant aux données captées. Si l'intérêt est trouvé, le nœud (qui devient désormais source de données pour cet intérêt) commence à envoyer l'évènement capté à tous ses voisins (constitués par les gradients établis dans la phase d'inondation d'intérêt) avec un faible débit appelé débit exploratoire. Suivant l'intérêt donné dans l'exemple adopté, l'évènement suivant forme une réponse pour le nœud collecteur (table 2.3) :

Attribut d'un événement	description
Type= véhicule roulant	Type du véhicule détecté
Instance= camion	Instance de ce type
Position= [125, 220]	Position de nœud
Intensité=0.6	Amplitude de signal capté
Confiance= 0.85	Degrés de confiance de la correspondance
Date= 01 :20 :40	Heure de génération de la donnée

Table 2.3 Les enregistrements correspondant aux données captées

Au départ, les données sont considérées comme exploratoires, et sont envoyées à tous les voisins, pour lesquels un gradient a été établi, avec un faible débit (dans l'exemple adopté, le débit est égal à 1 donnée par seconde). Ces données exploratoires voyagent vers le nœud collecteur dans le sens inverse de celui de la propagation des intérêts. Elles sont prévues pour la construction et la réparation des chemins qui seront adoptés par la suite.

Un nœud intermédiaire qui vient de recevoir un message de donnée exploratoire d'un nœud voisin, cherche dans son cache d'intérêts une entrée correspondante à l'évènement capté. Si aucune entrée n'est trouvée, le message de données est écarté. Sinon, le nœud vérifie alors son cache de données (associé à l'entrée de l'intérêt). Ce dernier garde trace des données récemment envoyées et permet, entre autres, l'empêchement des boucles. Si le message reçu correspond à une entrée du cache de données, il est abandonné car il a déjà été envoyé. Autrement,

le message est inséré dans le cache puis diffusé aux voisins pour lesquels un gradient a été établi.

Ainsi, chaque nœud intermédiaire exécute le même mécanisme en diffusant les données exploratoires vers tous ses voisins afin d'atteindre le nœud collecteur. Cette étape est caractérisée par la propagation des données exploratoires (appelée aussi phase d'exploration).

2.3.3. Renforcement Positif

La phase de propagation des données exploratoires a pour but d'explorer les chemins existants entre la source et le collecteur. Le renforcement positif sert à choisir une route (selon un certain critère) parmi celles découvertes, afin d'obtenir les données à haut débit appelées données renforcées. Pour cela, quand les données exploratoires atteignent le nœud collecteur, ce dernier choisit l'un de ses voisins appropriés et lui envoie un message de renforcement positif. Dans l'exemple adopté, un message de renforcement positif est identique à l'intérêt initial, mais le débit des données demandées est plus élevé (la valeur de l'attribut intervalle est plus petite. exemple : intervalle = 10ms). Un nœud intermédiaire qui reçoit ce message de renforcement, renforce son gradient vers l'émetteur. Dans l'exemple, un gradient est considéré comme étant renforcé si sa valeur (c'est-à-dire le débit de données demandé) est supérieure à 1 donnée / sec. Par la suite, le nœud doit, à son tour, envoyer le message de renforcement à l'un de ses voisins (en suivant le même critère de sélection). De cette manière, l'un des chemins explorés est récursivement renforcé (figure 2.7).

Pour le choix du chemin à renforcer, le nœud collecteur, et par la suite les nœuds intermédiaires, appliquent localement la règle de renforcement positif. Grâce au cache de données, un nœud choisit le premier voisin à partir duquel il a reçu une donnée exploratoire correspondant à l'intérêt. Par conséquent, un chemin ayant la plus faible latence est établi entre la source et le collecteur. D'autres règles peuvent être appliquées. Par exemple, un nœud peut choisir le voisin à partir duquel il a reçu le plus grand nombre de messages de données. Ainsi, le chemin le plus fiable est élu.

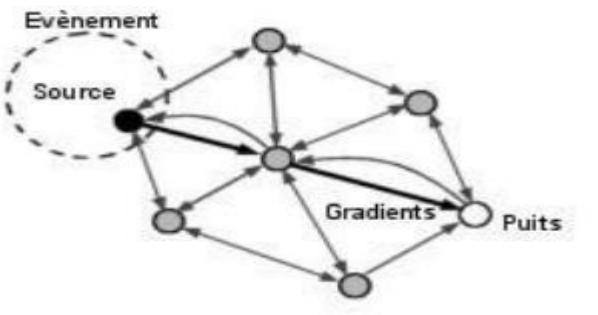

Figure 2.7 Renforcement d'un chemin [10]

Une fois qu'un chemin reliant le nœud source au nœud collecteur soit renforcé, les données générées par la source sont envoyées avec un débit plus élevé à travers ce chemin. Un nœud qui vient de recevoir une donnée renforcée enverra celle-ci uniquement aux voisins pour lesquelles il possède un gradient renforcé.

Cependant, même en présence d'un chemin renforcé, le nœud source rediffuse périodiquement des données exploratoires dans le but de renforcer ultérieurement un meilleur chemin vers le nœud collecteur. Dans notre exemple, les données exploratoires sont envoyées chaque seconde.

2.4. Conclusion

Nous avons présenté les principaux protocoles de routage pour les RCSF, ainsi qu'une classification de ces derniers. De mémé, nous avons détaillé le principe de fonctionnement du protocole Directed Diffusion (DD) qu'est l'un des protocoles les plus utilisés dans les RCSF, et qui était une source d'inspiration pour plusieurs d'autres protocoles.

Nous avons expliqué ses 03 principales phases, à savoir :

1. Envoi des intérêts.
2. Envoi des évènements.
3. Renforcement positif.

L'objectif du prochain chapitre est la modélisation du protocole de routage Directed Diffusion (DD) à l'aide des deux outils (UPPAAL et NS-2), cette modélisation sera suivie par l'analyse de quelques performances et la vérifier de quelques propriétés.

Chapitre 3 :

Modélisation du protocole de routage Diffusion Directe dans les outils « NS-2 et UPPAAL »

3.1 Introduction

Le but de notre travail est de modéliser, vérifier et analyse des performances du protocole de routage ‘‘Directed Diffision’’ dans un RCSF. Notre travail est donc basé sur trois grandes étapes qui sont :

1. La modélisation : En utilisant les automates temporisés, afin de capter l’aspect temporel du protocole ‘‘DD’’.
2. La vérification : En utilisant la technique ‘‘Model Checking’’ en utilisant le model checker de l’outil UPPAAL.
3. Analyse des performances du protocole DD avec l'outils NS-2.

Dans ce chapitre nous allons présenter la technique, le modèle et les outils nécessaires pour atteindre l’objectif fixé.

3.2 L’outil NS-2[56]:

NS-2 est un simulateur de réseaux connu, Il est développe en langage C++ en faisant intervenir des scripts de commande de simulation Tcl/Tk (scripts d’interprétations, évitant la manipulation du C++ aux utilisateurs). Le code source du simulateur est en C++, mais les paramètres de simulations sont définis dans des scripts Tcl. Une interface graphique de visualisation, pouvant présenter les résultats, est fournie avec la version 2.35 : Network Animato (Nam). Les résultats sont fournis de façon brute dans des fichiers textes qui peuvent être configures en détail dans les fichiers Tcl. Le traitement des fichiers de résultats se fait à l’aide de scripts externes, tels que des scripts "awk" (langage de traitement de lignes). Les résultats peuvent alors être exportés dans tout type de logiciels.

Figure 3.1: fonctionnement d'outil NS2**.**

3.2.1. Justification du choix de NS2:

NS est un outil logiciel de simulation de réseaux informatiques, il est parmi les simulateurs les plus utilisés dans les laboratoires de recherche, afin de simuler et étudier les performances des protocoles réseau. Il offre une plateforme de développement de nouveaux protocoles et permet de les tester.

Le simulateur NS actuel est particulièrement bien adapté aux réseaux à commutation de paquets et à la réalisation de simulations de grande taille (le test du passage à l'échelle). Il contient les fonctionnalités nécessaires à l'étude des algorithmes de routage unicast ou multicast, des protocoles de transport, de session, de réservation, des services intégrés, des protocoles d'application comme FTP. A titre d'exemple la liste des principaux composants actuellement disponibles dans NS par catégorie est :

- Application : Web, ftp, telnet, générateur de trafic (CBR...).
- Transport : TCP, UDP, RTP, SRM.
- Routage unicast : Statique, dynamique (vecteur distance).
- Routage multicast : DVMRP, PIM.
- Gestion de file d'attente : RED, DropTail, Token bucket.

3.2.2. L'outil de visualisation NAM:

NS-2 ne permet pas de visualiser le résultat des expérimentations. Il permet uniquement de stocker une trace de la simulation, de sorte qu'elle puisse être exploitée par un autre logiciel, comme NAM.

NAM est un outil de visualisation qui présente deux intérêts principaux : représenter la topologie d'un réseau décrit avec NS-2, et afficher temporellement les résultats d'une trace d'exécution NS-2. 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. Ce logiciel est souvent appelé directement depuis les scripts TCL pour NS-2, pour visualiser directement le résultat de la simulation.

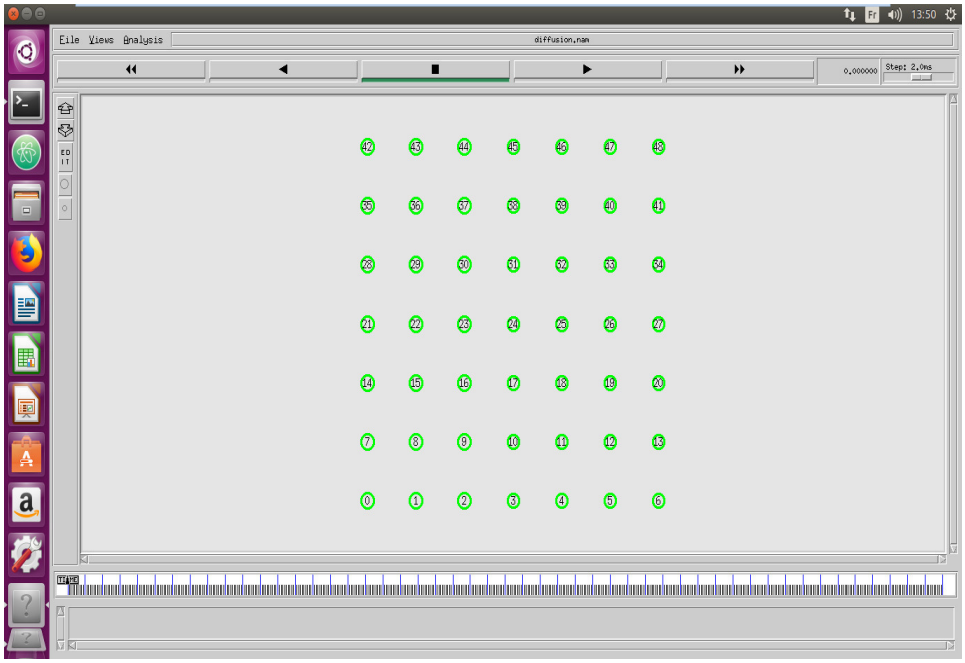

Figure 3.2: interface graphique NS2.

3.2.3. Caractéristiques de NS2:

• NS2 prend en charge la modélisation de la distribution du trafic.

• Il fournit une simulation d'une variété de protocoles HTTP, TCP, UDP,

SRM, RTP et certains algorithmes de routage.

• Il couvre presque toutes les variantes de TCP, plusieurs formes de multidiffusion, réseaux câblés, plusieurs protocoles de routage ad-hoc et modèles de propagation (mais pas les téléphones cellulaires), diffusion de données, satellite et autres.

• NS2 utilise différents générateurs de scénarios composés d'un générateur de topologie, d'un générateur d'agent et d'un générateur de routage.

• Le générateur de topologie peut générer une topologie à l'aide de n'importe quel générateur de graphe standard (GT-ITM, niveaux, etc.). Actuellement, il prend uniquement en charge le générateur GT-ITM et convertit le graphique de topologie au format NS.

• Le générateur d'agent peut être utilisé pour définir des agents de protocole de transport tels que TCP ou SRM.

• Le générateur de routage définit les protocoles de routage à utiliser dans la simulation. Les options incluent le routage unicast, le routage multicast et une option pour activer / désactiver l'adresse de développement.

• Dans NS2, le planificateur d'événements conserve une trace de l'heure de la simulation et libère tous les événements de la file d'attente des événements en appelant les composants réseau appropriés.

• Il fournit NAM et X-Graph pour animer et tracer le résultat de la simulation, respectivement.

3.2.4. Avantages:

1 - NS2 a un grand nombre de modèles disponibles, des modèles de mobilité réalistes, des scripts puissants et flexibles, une configuration de simulation, une large communauté d'utilisateurs et un développement continu.

2 - NS2 fournit un modèle de circulation et de mouvement facile en incluant un modèle énergétique efficace.

3 - Il fournit un ensemble de modèles de mobilité randomisés [29] et il existe plusieurs projets pour apporter des modèles de mobilité avancés aux simulateurs.

4 - Des scénarios complexes peuvent être facilement testés.

5 - Populaire pour sa modularité.

3.2.5. Limites:

1 - NS2 doit être recompilé à chaque fois s'il y a un changement dans le code utilisateur.

2 - Le système réel est trop complexe pour modéliser une infrastructure complexe.

3- Le mélange de compilation et d'interprétation a rendu difficile l'analyse et la compréhension du code.

4 - Y compris beaucoup de nœuds peut ralentir le processus de Simulation.

5 - l'utilisateur doit maitriser C++ et les scripts tcl/tk pour le moindre changement.

3.2.6. Installations et configurations:

Nous allons voir comment installer NS2 2.35 dans le système d'exploitation Linux Ubuntu 16.04 LTS .

Ce n'est pas une bonne idée d'utiliser des commandes directes dans le terminal comme:

sudo apt-get installer ns2

sudo apt-get install nam

Cela entraîne "Défaut de segmentation et core sous-évalué". Il suffit donc de suivre les instructions étape par étape ci-dessous pour une installation réussie.

ÉTAPE 1:

Installez toutes les dépendances nécessaires en utilisant les commandes ci-dessous l'une après l'autre.

sudo apt-get installer tcl8.5-dev tk8.5-dev
sudo apt-get install build-essentiel autoconf automake
sudo apt-get installer perl xgraph libxt-dev libx11-dev libxmu-dev

ÉTAPE 2:

Téléchargez le package NS2 à partir de:
"https://sourceforge.net/projects/nsnam/postdownload?source=dlp"

Copiez le fichier téléchargé dans votre dossier **/ Home** dans Ubuntu.

Faites un clic droit sur le fichier et sélectionnez l'option "Extraire ici". (Vous pouvez également le faire en utilisant la ligne de commande).

ÉTAPE 3:

Maintenant, allez dans le sous-dossier **ns-allinone-2.35 / ns-2.35 / linkstate.**

double-cliquez sur le fichier **"ls.h"** pour l'ouvrir.

aller à la ligne numéro 137 et changer la ligne ci-dessous

de

void eraseAll () {efface (baseMap :: begin (), baseMap :: end ()); }

à

void eraseAll () {this-> effacer (baseMap :: begin (), baseMap :: end ()); }

ÉTAPE 4:

Ouvrez le terminal en appuyant sur la combinaison de touches **"ALT + CNTL + T".** Et passer au dossier ns-allinne-2.35 de la maison à travers le terminal:

ubuntu @ ubuntu: ~ $ cd ns-allinone-2.35 /

ubuntu @ ubuntu: ~ / ns-allinone-2.35 $

Maintenant, tapez ./install sur le terminal:

ubuntu @ ubuntu: ~ / ns-allinone-2.35 $ ./install

appuyez sur Entrée et attendez un certain temps jusqu'à ce qu'il affiche des informations sur le chemin. C'est fait maintenant et vous êtes installé NS2.

ÉTAPE 5:

Il est maintenant temps de définir les informations de chemin. Dans le terminal, utilisez sudo gedit .bashrc et appuyez sur Entrée. Il demandera un mot de passe pour entrer (Ce n'est pas visible).

ubuntu @ ubuntu: ~ $ sudo gedit .bashrc

[sudo] mot de passe pour ubuntu:

Allez à la dernière ligne du fichier nouvellement ouvert (bashrc), copiez et collez ces 3 lignes. Assurez-vous que vous avez changé Ubuntu avec votre nom d'utilisateur sur Ubuntu.

PATH = $ PATH: /home/Ubuntu/ns-allinone-2.35/bin: /home/ Ubuntu /ns-allinone-2.35/tcl8.5.10/ Ubuntu: /home/ Ubuntu /ns-allinone-2.35/tk8.5.10/ Unix

LD_LIBRARY_PATH = $ LD_LIBRARY_PATH: /home/ Ubuntu /ns-allinone-2.35/otcl-1.14: /home/ Ubuntu /ns-allinone-2.35/lib

TCL_LIBRARY = $ TCL_LIBRARY: /home/ Ubuntu /ns-allinone-2.35/tcl8.5.10/library

Enregistrez le document et fermez-le. Rechargez le .bashrc en utilisant la commande suivante:

source ~ / .bashrc

ÉTAPE 6:

C'est fait! ouvrez le terminal et tapez **"ns"** appuyez sur Entrée. Vous obtiendrez un signe **%**, cela indique l'installation réussie.

3.3. Modélisation du protocole « DIRECTED DIFFUSION » dans NS2:

Nous allons démontrer commet modéliser le protocole DIRECTED DIFFUSION, qui est prédéfini dans NS2. Le protocole DD est défini par le biais de " initial code" écrit avec le langage C++. On peut le trouver dans le dossier "ns-allinone-2.35" sur le nom "DIRECTED_DIFFUSION".

```
set opt(chan)     Channel/WirelessChannel                    LL set mindelay_    50us
set opt(prop)     Propagation/TwoRayGround                   LL set delay_     30us
set opt(netif)      Phy/WirelessPhy                          LL set bandwidth_   0 ;# not used
set opt(mac)      Mac/802_11
set opt(ifq)      Queue/DropTail/PriQueue                    Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1
set opt(ll)       LL
set opt(ant)          Antenna/OmniAntenna                    # unity gain, omni-directional antennas
set opt(filters)      GradientFilter  ;# old name for twophasepull filter  # set up the antennas to be centered in the node and 1.5 meters above it
set opt(x)        560       ;# X dimension of the topography   Antenna/OmniAntenna set X_ 0
set opt(y)        560       ;# Y dimension of the topography   Antenna/OmniAntenna set Y_ 0
set opt(cp)       "./network"   ;# need to write new generate function  Antenna/OmniAntenna set Z_ 1.5
set opt(ifqlen)     50        ;# max packet in ifq             Antenna/OmniAntenna set Gt_ 1.0
set opt(nn)       49        ;# number of nodes                 Antenna/OmniAntenna set Gr_ 1.0
set opt(stop)     50.0      ;# simulation time
set opt(tr)       "output.tr"   ;# trace file                  # Initialize the SharedMedia interface with parameters to make
set opt(nam)            "diffusion.nam" ;# nam file            # it work like the 914MHz Lucent WaveLAN DSSS radio interface
set opt(energymodel)    EnergyModel     ;#set opt(energymodel)  Phy/WirelessPhy set CPThresh_ 10.0
set opt(initialenergy)  1000.0          ;# Initial energy in Joules  Phy/WirelessPhy set CSThresh_ 1.559e-11
set opt(txpower)    0.660                                      Phy/WirelessPhy set RXThresh_ 3.652e-10
set opt(rxpower)    0.350                                      Phy/WirelessPhy set Rb_ 2*1e6
set opt(idlepower)    0.035                                    Phy/WirelessPhy set Pt_ 0.2818
set opt(adhocRouting)   Directed_Diffusion                     Phy/WirelessPhy set freq_ 914e+6
set opt(prestop)    50.0                                       Phy/WirelessPhy set L_ 1.0
```

Figure 3.3 Déclaration global du système dans NS2

Pour définir le sink:

set snk_(0) [new Application/DiffApp/PingReceiver/TPP]

$ns_ attach-diffapp $node_(24) $snk_(0)

$ns_ at 1.1 "$snk_(0) subscribe"

Définir la position initiale des nœuds dans Nam :

for {set i 0} {$i < $opt(nn)} {incr i} {

    $ns_ initial_node_pos $node_($i) 20 }

- 20 définit la taille des nœuds dans Nam, doit l'ajuster en fonction de notre scénario.

- La fonction doit être appelée après la définition du modèle de mobilité.

Génération de topologie de réseau:

```
for {set i 0} {$i < $opt(nn) } {incr i} {
        set node_($i) [$ns_ node $i]
        $node_($i) random-motion 0
        $node_($i) set X_ [expr ($i%7)*80+40]
        $node_($i) set Y_ [expr ($i/7)*80+40]
        $node_($i) set Z_ 0;
        $god_ new_node $node_($i)
}
```

3.4. L'outil UPPAAL

UPPAAL est un outil de modélisation et de vérification de systèmes temps-réel qui peuvent être représentés sous forme de processus non-déterministes et interactifs ayant une structure de contrôle finie et des horloges avec des valeurs réelles. Un modèle d'un système dans UPPAAL se compose d'un réseau de processus décrits par des automates temporisés étendus communiquant par des canaux ou des structures de données partagées. La conception de l'outil UPPAAL est basée sur l'architecture Client/serveur, comme le montre la Figure 3.4.

Le serveur est représenté par la machine UPPAAL développée en C++ et le client est représenté par l'interface GUI développée en JAVA TM. La communication entre la machine UPPAAL et son interface graphique est établie à travers des protocoles internes.

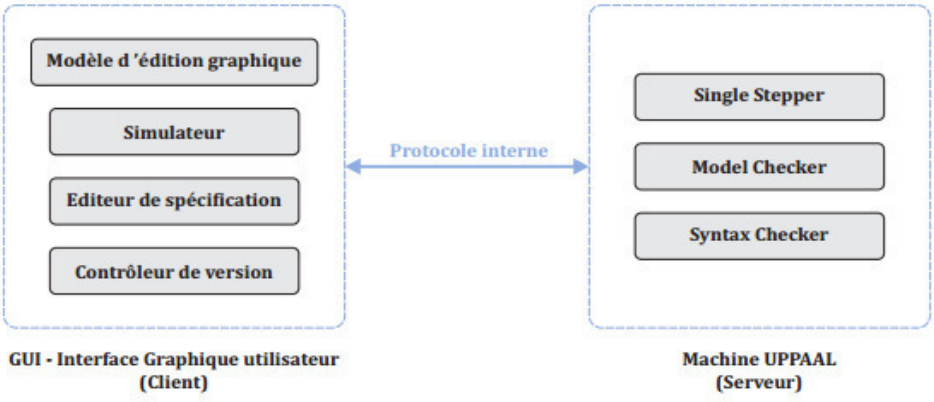

Figure 3.4 Architecture d'UPPAAL

3.4.1. Les systèmes de transitions [6]

Les systèmes de transitions (automates et réseaux de Pétri) sont les principaux modèles utilisés pour spécifier le comportement de systèmes réactifs et/ou temps réel. Un système de transitions est un automate. Le terme système de transitions est surtout utilisé par ceux qui s'intéressent à la modélisation de systèmes. Le terme automate est plus général.

Les automates constituent des techniques de base que l'informaticien se doit de connaître. Ils sont utilisés essentiellement dans la théorie des langages et dans la modélisation et vérification de systèmes communicants (qui englobent les systèmes temps réel). La notion d'automates est utilisée dans différentes méthodes de conception (UML, SDL et autres), avec des notations graphiques souvent différentes d'une méthode à l'autre.

Les automates offrent des mécanismes qui sont à la fois formels et simples à comprendre et à utiliser (surtout lorsqu'ils sont utilisés sous forme graphique). Ils servent aussi à vérifier formellement (notamment par les techniques du *model Checking*) des propriétés sur les systèmes modélisés.

Un automate temporisé est un automate doté d'une horloge (ou de plusieurs horloges) qui compte(nt) le temps qui s'écoule. Les horloges peuvent être remises à zéro indépendamment les unes des autres. Le tir des transitions tient compte des valeurs des horloges.

3.4.2. Justification du choix d'UPPAAL

Nous allons travailler par l'outil UPPAAL, vu que ce dernier utilise les automates temporisés comme un modèle pour modéliser les systèmes temps réel. Ce choix donc est motivé par la simplicité des AT (des machines états-transitions) et leurs extensions temporelles qui facilite la modélisation du passage du temps ou les autres contraintes temporelles (qui caractérisent les systèmes temps réel).

UPPAAL est doté d'un simulateur qui nous permet de capter le comportement de notre système, ainsi que les valeurs qui le caractérisent.

UPPAAL, utilise pour la vérification un model checker, ce qui facilite la vérification de notre système, le vérifier d'une manière formelle, automatique et exhaustive.

3.4.3. Le Model-Checker UPPAAL

UPPAAL est un outil intégré pour l'environnement de modélisation, la simulation et la vérification des systèmes temps réel, développé conjointement par la recherche fondamentale en informatique à l'université d'Aalborg au Danemark et au Département des technologies de l'information à l'université d'Uppsala en Suède.

Il est approprié pour les systèmes qui peuvent être modélisés comme un ensemble de processus son déterministe fini, avec la structure de contrôle et valeur réelle d'horloges, de communiquer à travers des canaux ou des variables partagées.

Domaines d'application typiques comprennent les contrôleurs temps réel et des protocoles de communication en particulier, le moment où ces aspects sont essentiels. [8]

3.4.4. Description

Uppaal se compose [8] de trois parties principales:

- ***Un langage de description*** : est un langage de commande surveillé non déterministe avec des types de donnée (par exemple tableaux, délimitée entiers, etc..) il sert de modélisation ou de conception de langage pour d'écrire le comportement du système Comme des réseaux d'automates étendus avec horloges et des données variables.
- ***Le simulateur*** : est un outil de validation qui permet l'examen de possibles exécutions dynamiques d'un système au début de la conception (ou de modélisation). Il offre ainsi un moyen peu couteux de détection des défauts avant vérification par le vérificateur de modèle exhaustif qui couvre le comportement dynamique du système.
- ***Le modèle vérificateur*** : peut vérifier l'accessibilité et l'invariabilité par l'exploration des propriétés de l'état espace d'un système, c'est-à-dire l'accessibilité de l'analyse en termes symboliques représentés par des contraintes.

Un autre élément important pour l'efficacité est l'application d'une technique qui réduit symboliquement la vérification des problèmes à celle de la manipulation efficace et la résolution de contrainte.

Pour faciliter la modélisation et le débogage, le vérificateur du modèle Uppaal génère automatiquement une trace de diagnostic qui explique pourquoi la propriété est (ou n'est pas) satisfaite par un modèle du système.

3.4.5. Caractéristiques de UPPAAL

l'éditeur graphique permet la description graphique des système. Une simulation graphique qui fournit des graphiques de visualisation et d'enregistrement de la dynamique des comportements possible du système. Il est également utilisé pour visualiser les traces générées par le model de vérification. [30]

3.4.5.1. Syntaxe [3]

Un modèle UPPAAL est basé sur les automates temporisés. La description d'un modèle se compose de trois parties : i) les déclarations globales et locales, ii) les modèles d'automates, et iii) la définition du système. Ce volet présente la syntaxe d'UPPAAL.

- **Déclarations** : Les déclarations peuvent être locales ou globales. UPPAAL permet de déclarer des horloges (type clock), des entiers (type int), des booléens (type bool), des canaux de synchronisation (type chan), des tableaux, des structures de données (type struct), et de définir des types de données (typedef).

- **Locations** : Un automate temporisé est considéré comme un graphe à états. Les locations représentent les états dans ce graphe. Elles se communiquent via les transitions. Un identifiant optionnel peut être associé à chaque location afin de l'identifier. Elles peuvent être étiquetées par des invariants. Un invariant peut être une expression composée d'un ensemble de contraintes sur des horloges, une différence entre des horloges ou des expressions booléennes ne faisant pas intervenir des horloges. UPPAAL distingue trois types de locations :
    - **Location initiale** : Chaque automate doit avoir une location initiale représentée pas un double cercle.
    - **Location urgente** : La location urgente ne permet pas de faire passer le temps quand le processus s'y trouve. Sémantiquement, une location urgente est équivalente à une location dont l'invariant est x <= 0 ou x est une horloge remise à zéro sur toutes les transitions entrantes de la location en question.
    - **Location engagée** : La location engagée ne permet pas l'écoulement du temps quand un processus s'y trouve. Elles sont utiles pour assurer l'atomicité de la synchronisation entre trois processus où plus. Elles sont plus prioritaires que les locations urgentes.

- **Transitions** : Les locations sont connectées entre elles par les transitions (edges). Ces dernières sont étiquetées par des sélections, des gardes, des synchronisations et des mises à jour des variables et des fonctions.

  -Sélection : Cette option permet d'associer, d'une façon indéterminée, à une variable donnée une valeur dans un intervalle bien déterminé. Les gardes, la synchronisation et les mises à jour peuvent être spécifiées en fonction de la valeur sélectionnée.

  -Garde : Une transition est franchie si et seulement si la garde spécifiée est satisfaite. Les gardes expriment des contraintes d'horloges ou de données nécessaires pour franchir une transition.

  -Synchronisation : Les différents automates dans un modèle UPPAAL peuvent se synchroniser via des canaux de synchronisation permettant l'envoi et la réception d'un message. Deux transitions dans deux processus différents peuvent se synchroniser si et seulement si les gardes des deux transitions sont satisfaites et elles admettent deux étiquettes de synchronisation a1 ! (envoi) et a2 ? (réception) où a1 et a2 représentent le même canal de synchronisation. Quand deux processus se synchronisent, les deux transitions sont franchies au même temps. Les mises à jour de la transition portant l'étiquette a1! se font avant celles de la transition portant l'étiquette a1 ?. Il existe deux types de canaux de synchronisation :

  i)Les canaux de synchronisation binaires qui permettent de synchroniser deux processus via un seul canal de synchronisation a (au moyen de deux actions a! et a?) et

  ii) Les canaux de synchronisation de diffusion (broadcast) qui permettent de synchroniser un processus avec plusieurs autres processus. Une transition avec une étiquette de synchronisation a!, avec ‘‘a’’ est un canal broadcast, peut synchroniser avec chaque transition ayant une étiquette de synchronisation a ? et dont les gardes sont satisfaites.

  -Mise à jour : L'exécution de la mise à jour sur une transition permet de changer l'état du système. Une mise à jour est une liste d'expressions séparées par des virgules et exécutées dans un ordre séquentiel. L'opérateur d'assignement est ‘‘ = ’’.

3.5. Modélisation du protocole « DIRECTED DIFFUSION » dans Uppaal:

Notre système est composé de deux types de composant, les capteurs et une station de base (Sink). Ci-dessous, nous présentons une description du rôle et du fonctionnement de chacun de ces composants.

Les figures ci-dessous présentent la configuration globale du système sous l'UPPAAL. Tout d'abord nous déclarons les constantes, les canaux de communication, et les variables globales. Ensuite chaque composant sera spécifié à part.

```
//Déclaration global
clock sys;                                  // l'horloge
const int N = 50;                           // nombre de capteurs
//Les tables et les matrices……
typedef int[1,N] id_t;                      // identificateur d'un capteur
typedef int interet_sink_t [id_t];          // la table des capteurs collectent avec le sink
typedef int interet_t[id_t][id_t];          // matrice des intérêts du capteurs
typedef int even_t[id_t][id_t];             // matrice des évènements du capteurs

typedef int even_renf_t[id_t][id_t];        // matrice des évènements renforces
int  tab_renf[id_t];                        // la table de renforcement
// Les types des canaux……..
chan  RP[N], is_panne, even_renf_sink, even_renf[N+1] ;
broadcast chan initia, event, interet, fin_event;
//Les procédures ……..
Void Mat_even(interet_t Mat)                //créer la matrice d'évènements
Void TAB_renf(id_t renf)                    //créer la table de renforcement positif
Void reset_renf()                           //initialiser la table de renforcement
Void Mat_evn_renf()                         //créer la matrice d'évènements renforces
Void Mat_even_renf_panne(id_t m)            //modifier la matrice d'évènements renforces
Void cal_nbr_capt_on (id_t capt)            //calcule le nombre des capteurs connectés
void panne_capt(id_t panne                  //modifier la matrice des intérêts après la panne
```

Figure 3.5 Déclaration global du système dans UPPAAL

Ci-dessous, nous présentons les différentes spécifications des composants du système. Pour chaque composant, il y'a une template (automate temporisé étendu) ainsi qu'une déclaration locale associée à cette template.

- **Station de base (sink)**

C'est le centre vers lequel les données collectées par les capteurs sont envoyées. Au début elle doit initialiser les capteurs dans le réseau par un canal de diffusion (Broadcast Channel) **initia !**. Nous utilisons d'autres canaux de communication et de variables comme suit :

interet! : Envoie un intérêt à capter par le Sink vers les capteurs.

is_panne? : Modélise la panne d'un capteur.

event? : Recevoir un évènement de l'un des capteurs.

fin_event! : Pour faire fin de l'envoie d'un évènement une fois atteint le sink.

RP[e]! : Envoie le renforcement positif vers le capteur qui a envoyé l'évènement.

Even_renf_sink ?: Recevoir un évènement renforcer de l'un des capteurs.

Cond_env_inter : Condition de début d'envoie des intérêts par le Sink.

condition_env_renf : Condition de début d'envoie de renforcement positif.

cal : Calcule l'énergie.

h : Condition de la fin de réception d'un évènement par le Sink.

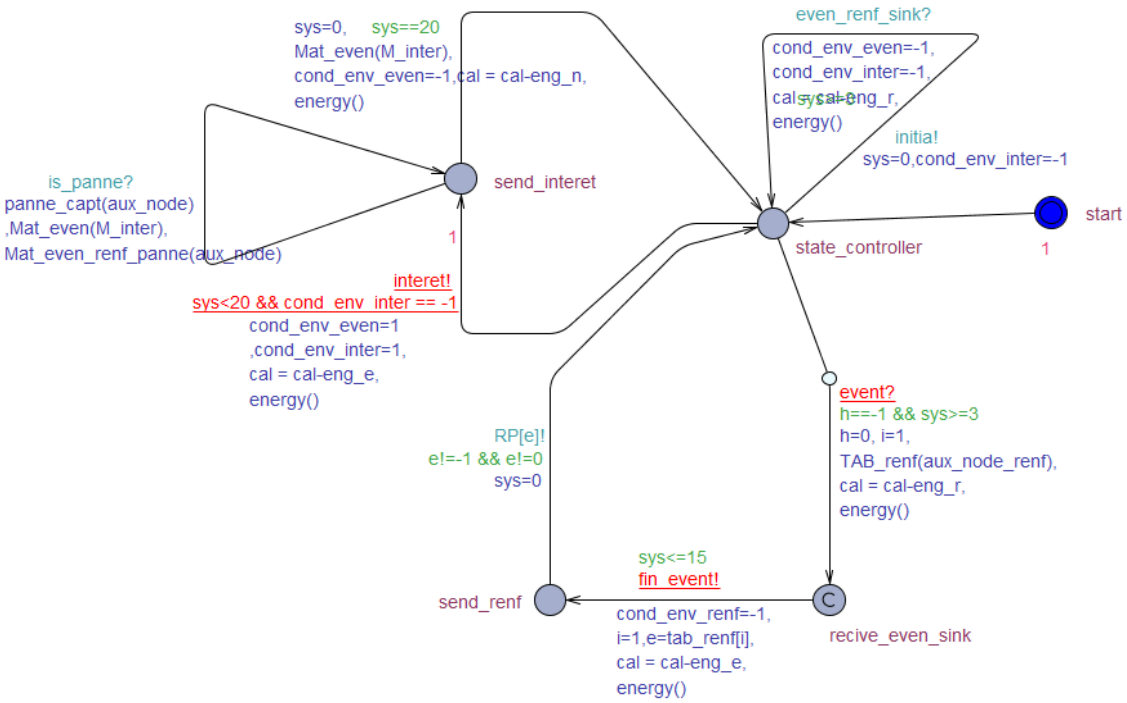

Figure 3.6 Automate temporisé qui modélise de Sink

- **Les capteurs**

Ils représentent un groupe de dispositifs connectés entre eux pour envoyer des données vers le Sink. Ils sont initialisés par le canal de diffusion **initia ?**. Nous utilisons aussi les canaux de communication et les variables suivantes :

can_hear_sink[id]: elle sert à identifier le capteur lors d'une envoie ou d'une réception.

condition_env_even : Condition de début d'envoi d'un évènement.

aux_node=id: L'affectation de l'identificateur de capteur comme un indice lors de l'envoie des intérêts

aux_node_even=id: L'affectation de l'identificateur de capteur comme un indice lors de l'envoie des évènements.

aux_node_renf=id: L'affectation de l'identificateur de capteur comme un indice lors de l'envoie d'un renforcement.

can_hear_sink[id] : Pour le passage d'un capteur vers le Sink où inversement.

can_hear[id][aux_node]: pour envoyer des intérêts vers les capteurs connectés au capteur actuel.

can_hear_even[id][aux_node_even]: pour envoyer des évènements vers les capteurs connectés au capteur actuel.

can_hear_even_renf[id][aux_node_renf]: pour envoyer des évènements renforcer vers les capteurs connectés au capteur actuel.

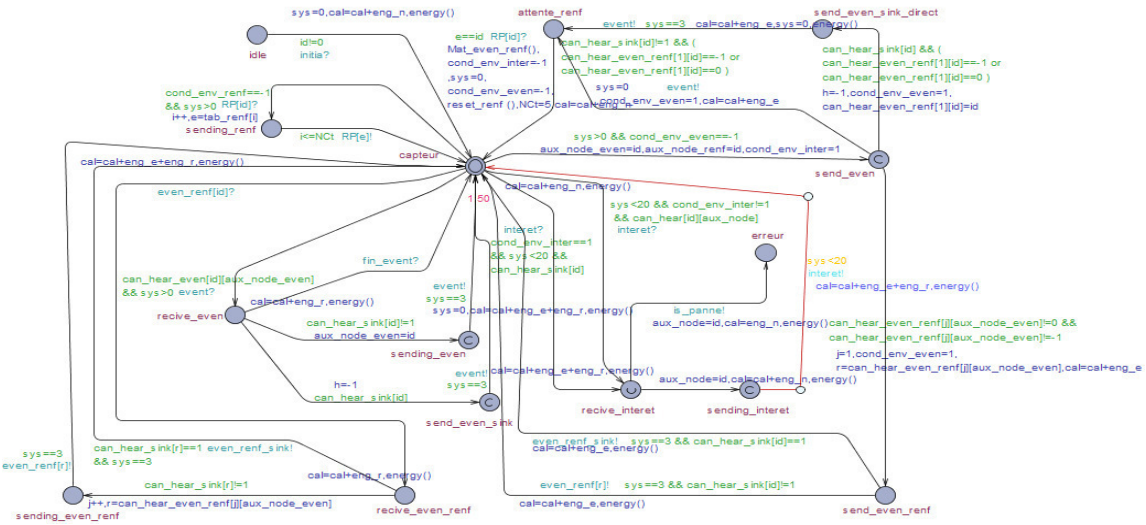

Figure 3.7 automate temporisé qui modélise un capteur

3.6. Conclusion:

Ce chapitre a été consacré pour présenter la modélisation de protocole dans les deux outils « Ns2 et Uppaal ». Nous avons réussi à modéliser la station de base (Sink) ainsi que les capteurs. Cette modélisation prend en charge les fonctionnalités de base de chacun des capteurs. Ces fonctionnalités se résument dans l'envoie et la réception des intérêts, des évènements et des renforcements positifs. Cette modélisation nous a aidé à mieux comprendre le fonctionnement d'un RCSF en utilisant le protocole de routage « Directed diffusion ».

Chapitre 4:

Vérification et analyse de performance du protocole de routage « Directed Diffusion »

4.1. Introduction :

Le but de ce chapitre est de valider notre modélisation fait par les outils UPPAAL et NS2.cette validation sera faite après une comparaison entre les résultats obtenus par les deux outils. Cette comparaison est le fruit de l'extraction des valeurs obtenus en utilisant des métriques spécifiques pour ces réseaux. Nous terminons ce chapitre par la discussion des résultats obtenus.

4.2. Les synchronisations et la vérification du protocole :

4.2.1. Les synchronisations:

- **L'initialisation**

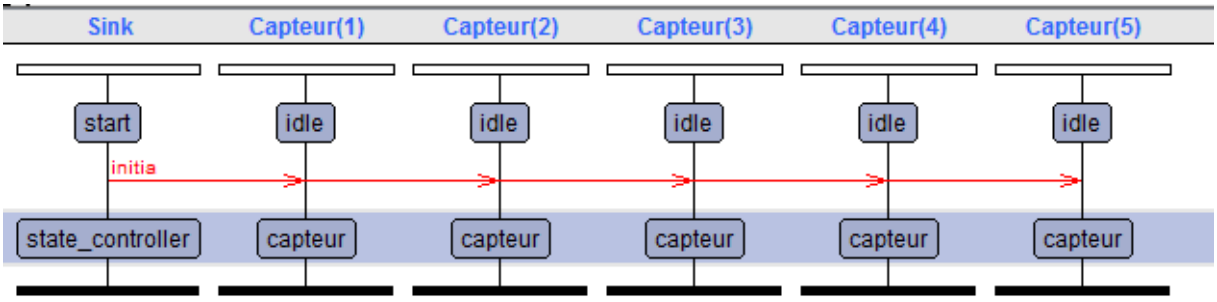

Figure 4.1 L'initialisation des capteurs

- **L'envoie des intérêts**

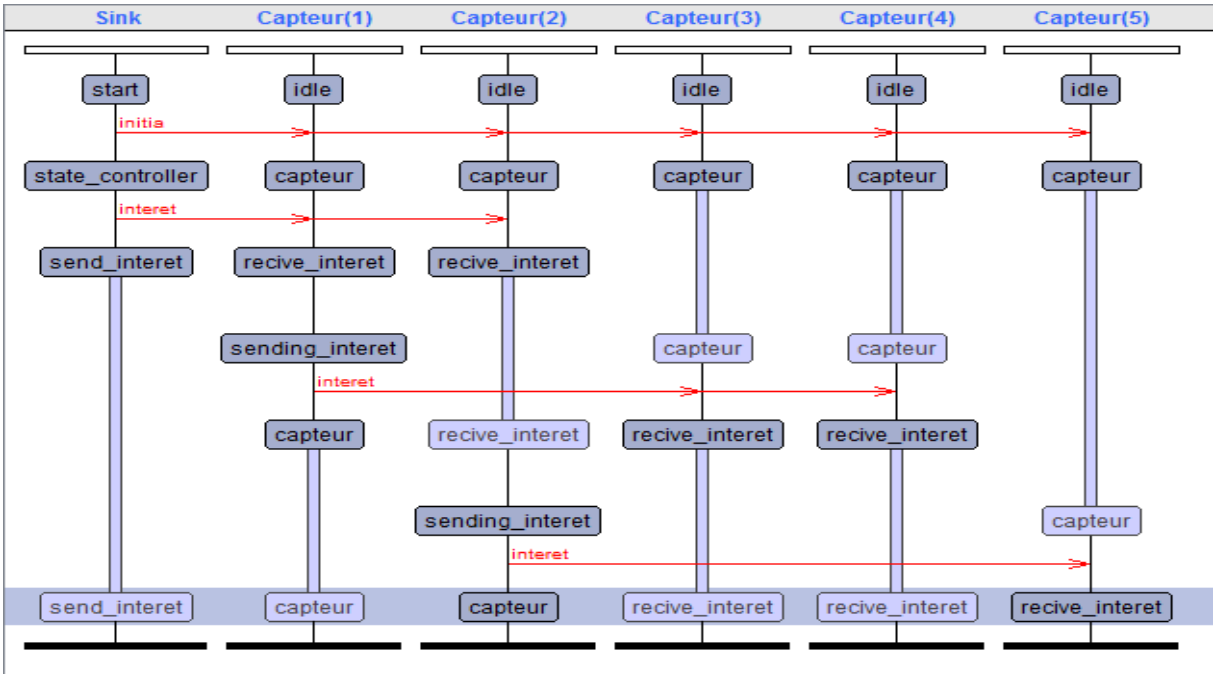

Figure 4.2 L'envoie des intérêts

- **Cas d'une panne**

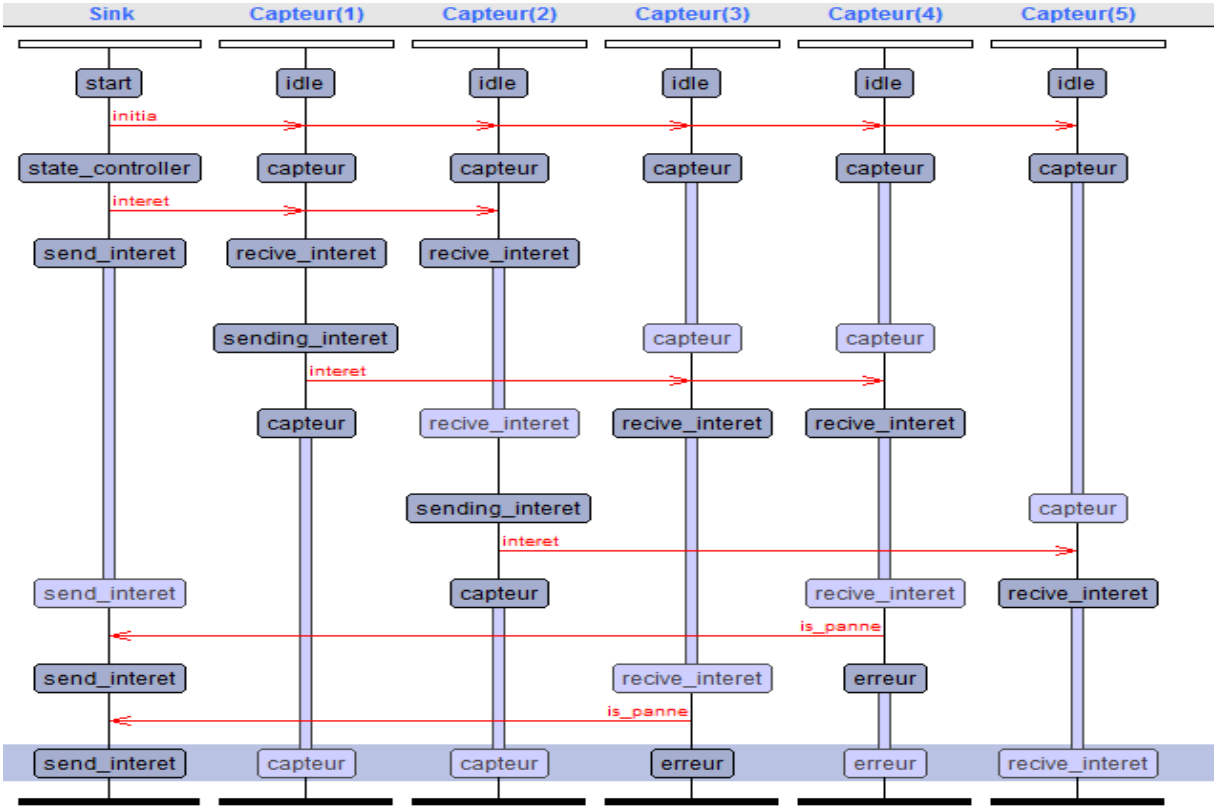

Figure 4.3 L'envoie de panne

- **L'envoie des évènements**

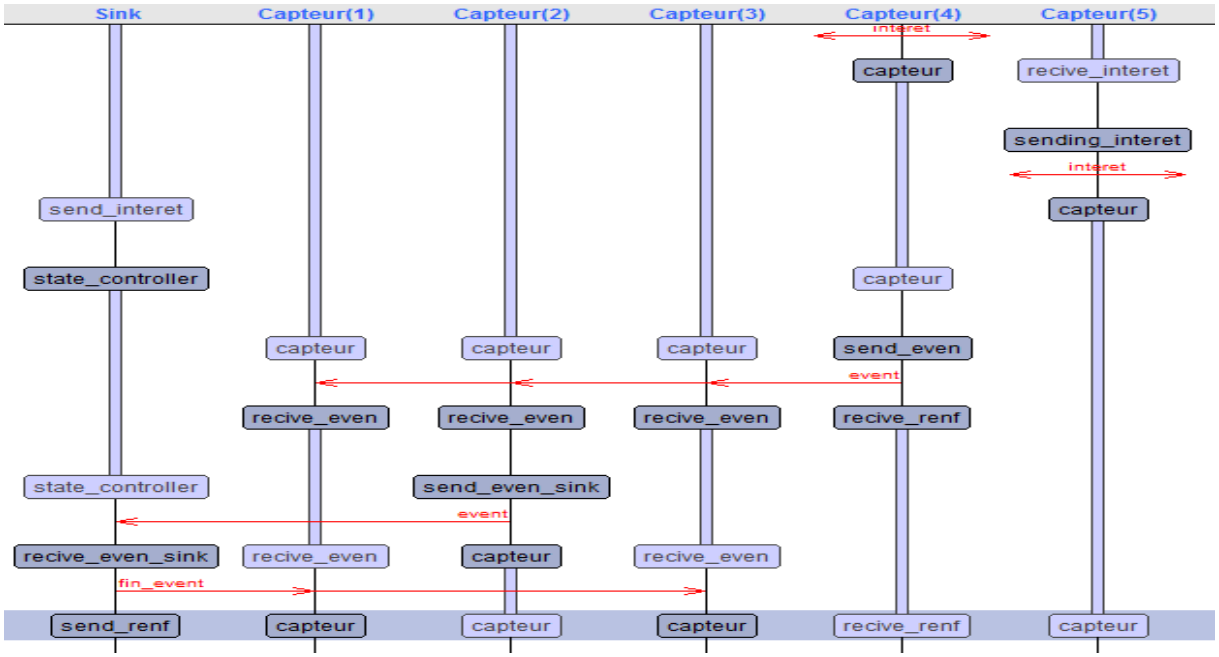

Figure 4.4 L'envoie des évènements d'un capteur vers le Sink

- **L'envoie de renforcements**

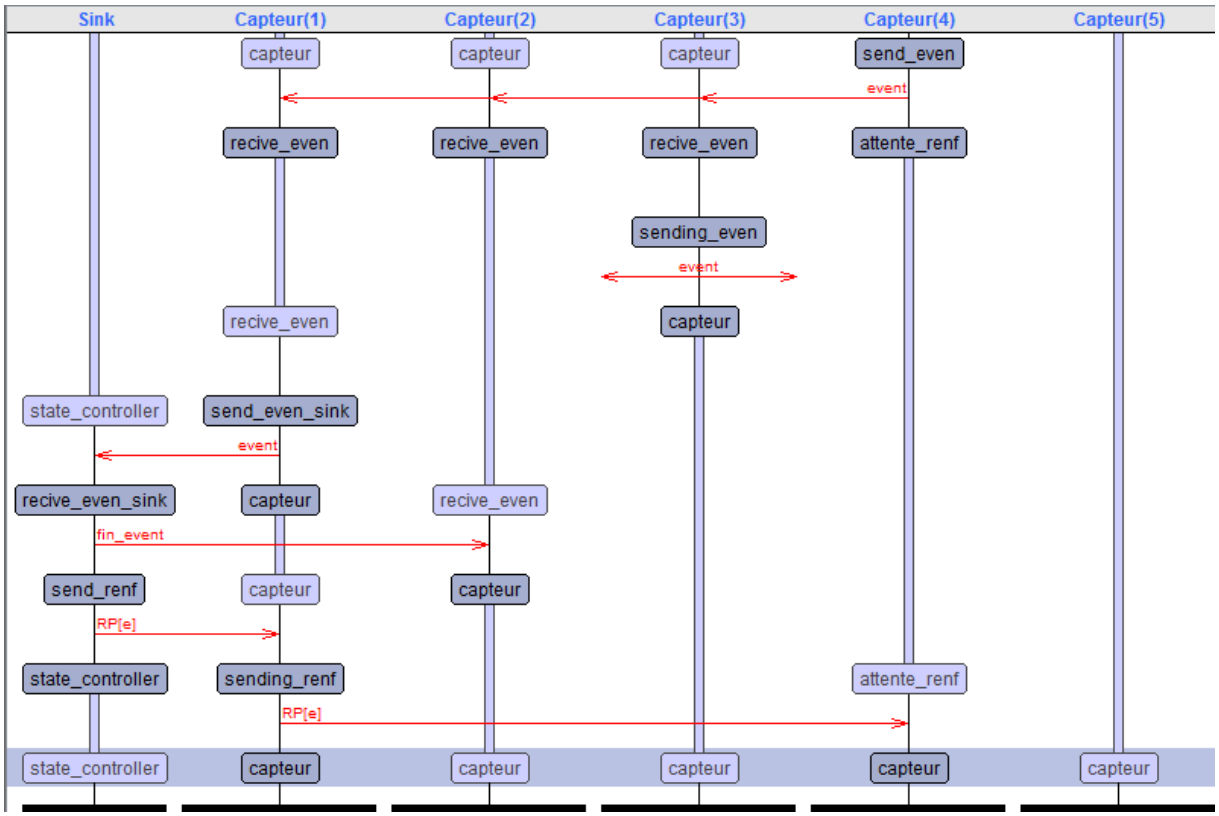

Figure 4.5 L'envoie de renforcement par le Sink vers le capteur qui envoie l'évènement

- **L'envoie des évènements par le chemin de renforcement**

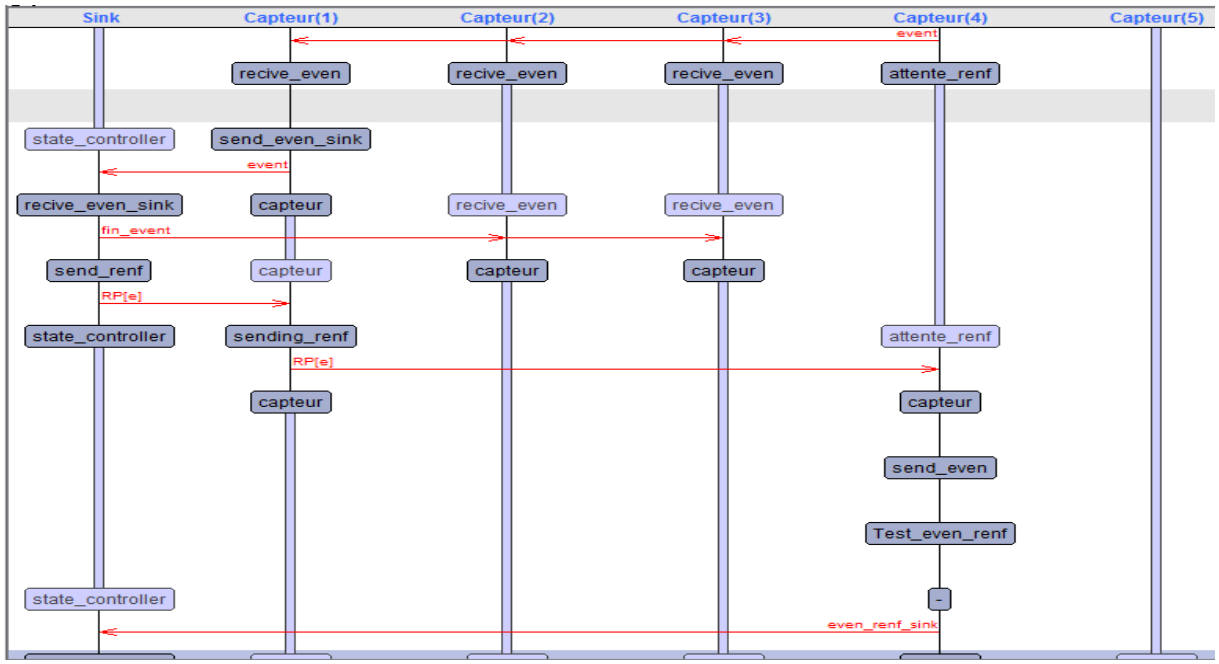

Figure 4.6 L'envoie d'un évènement par le chemin de renforcement vers le Sink

4.2.2. La Vérification:

L'objectif principal de l'usage de l'UPPAAL c'est de vérifier certaines propriétés dans le système spécifié. Nous avons établi les propriétés suivantes qu'on va essayer de les vérifier dans cette section.

- **L'initialisation**

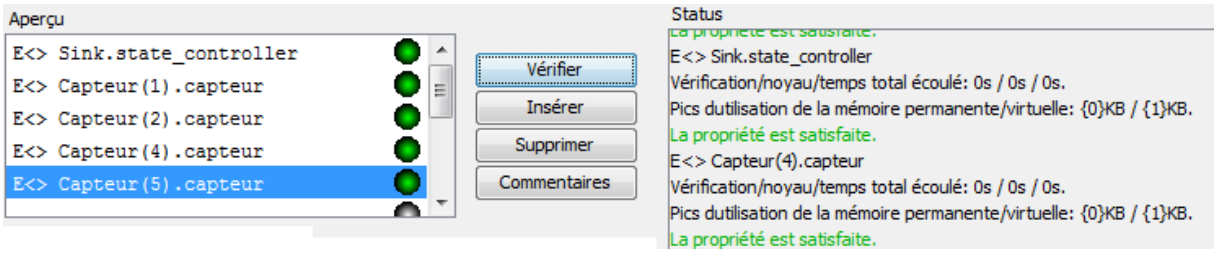

Figure 4.7 L'initialisation des capteurs

- **L'envoie des intérêts**

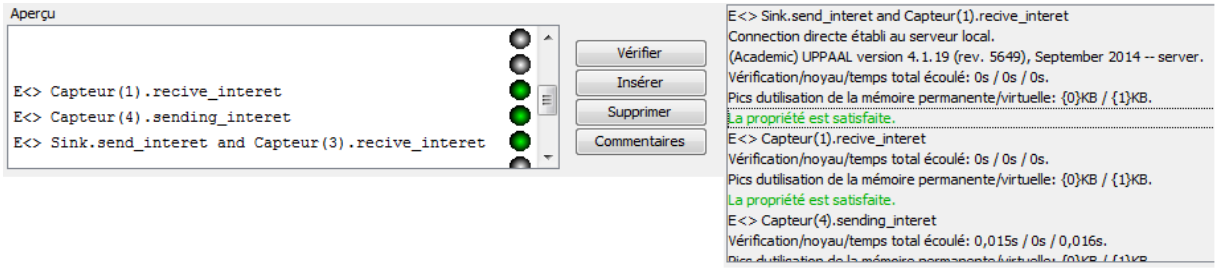

Figure 4.8 L'envoie des intérêts

- **L'envoie des évènements par le chemin de renforcement**

Figure 4.9 L'envoie d'un évènement par le chemin de renforcement vers le Sink

Absence de l'inter-blocage dans le système

- **A [] not deadlock** : cette propriété spécifie que tous les chemins d'exécution ne contiennent pas de blocage.
- **E<> not deadlock** : cette propriété spécifie qu'il y'a au moins un chemin sans blocage.

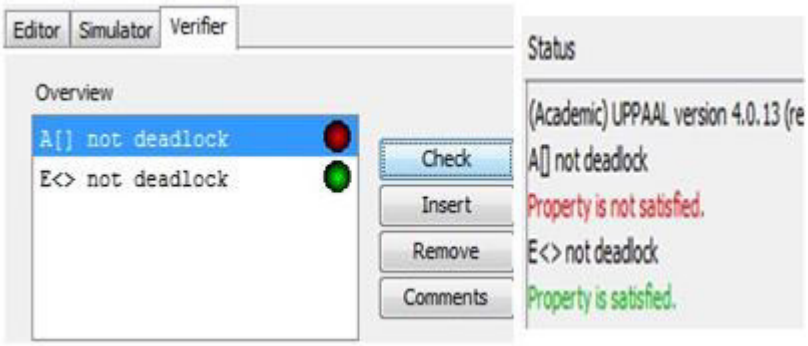

Figure 4.10 vérification d'inter-blocage du système

La propriété: **A [] not deadlock** n'est pas vérifiée. La Figure 4.11 montre un scénario de blocage qui représente un contre-exemple, où le système se bloque.

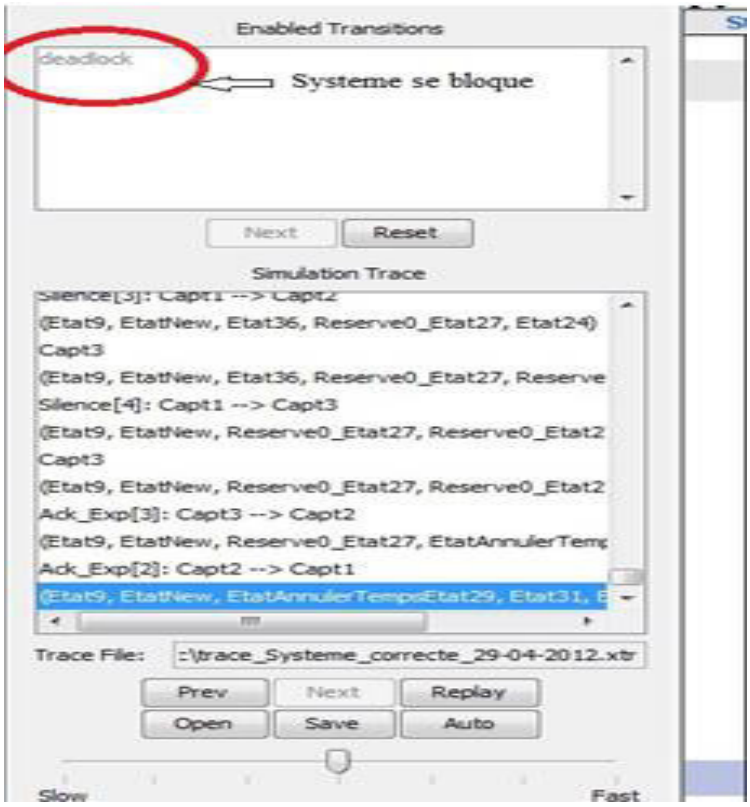

Figure 4.11 Scénario d'un blocage

4.3. L'analyse des performances:

4.3.1. Les métriques de performances [56] :

La simulation donne un ensemble varié d'informations concernant les déplacements effectués (direction, vitesse, etc.) ainsi que tous les échanges de données entre les nœuds du réseau, ces échanges sont les identifiants des nœuds et des destinations, le temps de génération des données, les types de paquets échangés, etc. Tout au long de la simulation, toutes ces informations sont placées dans un fichier appelé fichier de trace en permettant de mesurer l'efficacité des transmissions selon un ensemble de métriques, utilisées pour l'évaluation de certains paramètres.

4.3.2 Les métriques choisies [56]:

- **Le délai moyen:**

Est le temps écoulé entre le moment de la génération d'un paquet puis son envoi de la couche de transport du nœud source et le moment de réception de ce paquet par la couche de transport du nœud destinataire. Cette durée inclut le temps de traitement de l'acheminement de l'acheminement, le temps de routage dans les files d'attente du routeur et le délai de propagation. Le délai moyen de bout en bout est mesuré indépendamment pour tous les paquets reçus pendant la simulation, puis nous calculons la moyenne, de sorte qu'elle est calculée directement en divisant le temps total de livraison par rapport au nombre total de paquets. Cette métrique permet de mesurer l'effet de la mobilité sur l'efficacité de la transmission en termes de temps de réponse et en termes de choix de chemins optimaux.

$$Délai = \frac{le\ temps\ \ total\ de\ livraison\ des\ paquets\ (sec)}{le\ nombre\ des\ paquets\ de\ données\ reçus\ (paquet)}$$

- **L'énergie dissipée moyenne:**

Est le comte de l'énergie absente perdant de déplacement des paquets aux Source jusqu'à le Sink par l'intermédiaire des paques et l'énergie consommé par chaque nœud

$$\textbf{L'énergie} = \frac{\textbf{(L'énergie total - l'énergie dissipée)}}{\textbf{nombre des paques recevez}}$$

4.4. Le Scénario de la simulation :

Dans cette simulation, nous allons utiliser ns-2 pour étudier le délai moyen et l'énergie moyenne dissipée en utilisant le routage de diffusion dirigée dans le réseau IEEE 802.11. Le but principal de cette expérience est d'évaluer le retard et la consommation d'énergie du protocole de routage.

- Nous allons construit un scénario de simulation avec (n = 49) nœuds statiques avec émetteur variable et un seul récepteur.

- Nous allons choisi le protocole de routage "diffusion dirigée" et nous allons défini délai à 30 µs.

-La dimension du terrain est de 560 × 560 et les nœuds capteurs: nœud 1, nœud 2, nœud 3, . . ., nœud 8, nœud 9, . . ., le nœud 49 sont situés à (X = 40, Y = 40), (X = 120, Y = 40), (X = 200, Y = 40), . . ., (X = 40, Y = 120), (X = 120, Y = 120), . . ., (X = 520, Y = 520), respectivement.

- Nous allons Configurer le nœud 25 comme Sink et nous allons choisi au hasard les sources des nœuds restants. Variez votre nombre de sources comme telles 1, 5, 10, 15 et 20 sources. Le temps de démarrage pour les sources est de 0,4 seconde et pour le Sink de 1,1 seconde et le temps d'arrêt pour les sources / collecteurs est de 50 secondes.

- Nous allons réglé le modèle d'énergie radioélectrique de manière à ce que l'énergie initiale soit de 1 000 J, la puissance dissipée au repos de 35 mW, la puissance de réception est de 350 mW et la puissance d'émission est de 660 mW.

- Et Pour chaque nombre aléatoire de sources {1, 5, 10, 15, 20}, on va exécuter la simulation avec les paramètres comme ci-dessus pendant 30 fois avec une valeur de départ variable.

- Et à la fin de chaque simulation Nous allons Calculer:

a) le délai moyen.

b) l'énergie dissipée moyenne.

a) Tracer le délai moyen en fonction du nombre de sources:

20 sources	15 sources	10 sources	5 sources	1 source	temps
13.1623	11.109	13.71	11.9862	7.00389	1.666667
17.417	16.7039	13.6355	11.9723	8.30001	3.333333
13.1612	16.0714	22.1097	18.2845	6.96622	5
15.6641	11.4368	17.3615	19.8043	17.5521	6.666667
12.8507	14.473	13.9572	11.9381	6.89927	8.333333
17.8618	14.8854	13.8317	15.4344	6.93181	10
17.39	15.2951	18.6076	11.9846	13.4581	11.66667
13.0037	14.3662	17.5477	11.8873	15.91	13.33333
17.4061	11.0582	13.8887	11.8337	11.3267	15
13.5351	11.6346	13.6881	17.3742	15.1319	16.66667
12.7543	13.7587	18.6281	14.9475	12.3628	18.33333
14.5436	15.8186	13.8469	12.0017	17.537	20
12.7257	15.3914	13.959	11.8294	16.3653	21.66667
21.0838	14.3005	17.6944	20.593	25.5086	23.33333
12.9755	14.1033	18.1314	12.0164	8.56392	25
12.9444	15.3661	13.9249	19.8068	6.92806	26.66667
15.9496	15.886	13.9747	14.6326	10.4691	28.33333
14.5695	10.312	13.7943	15.9136	6.94837	30
17.7968	11.4618	13.8004	12.0611	11.1512	31.66667
12.6922	14.3795	21.7752	15.3594	6.83128	33.33333
15.5579	12.7392	13.7429	19.7212	9.60357	35
13.027	16.529	13.8176	11.8563	13.564	36.66667
13.2188	10.974	13.9079	11.9125	25.5535	38.33333
12.5077	14.8961	17.9007	16.8905	14.7354	40
15.3957	17.8677	13.8679	17.2558	8.56392	41.66667
13.9498	16.4338	13.9575	11.9702	11.0496	43.33333
16.6285	14.1825	22.6789	16.3928	18.4426	45
13.9303	16.7663	14.0492	23.3384	18.232	46.66667
14.6221	14.3368	26.7541	11.995	13.7732	48.33333
13.1328	15.8318	13.7634	12.004	12.9584	50

Table 4.1. résultat de calcule de délai moyen

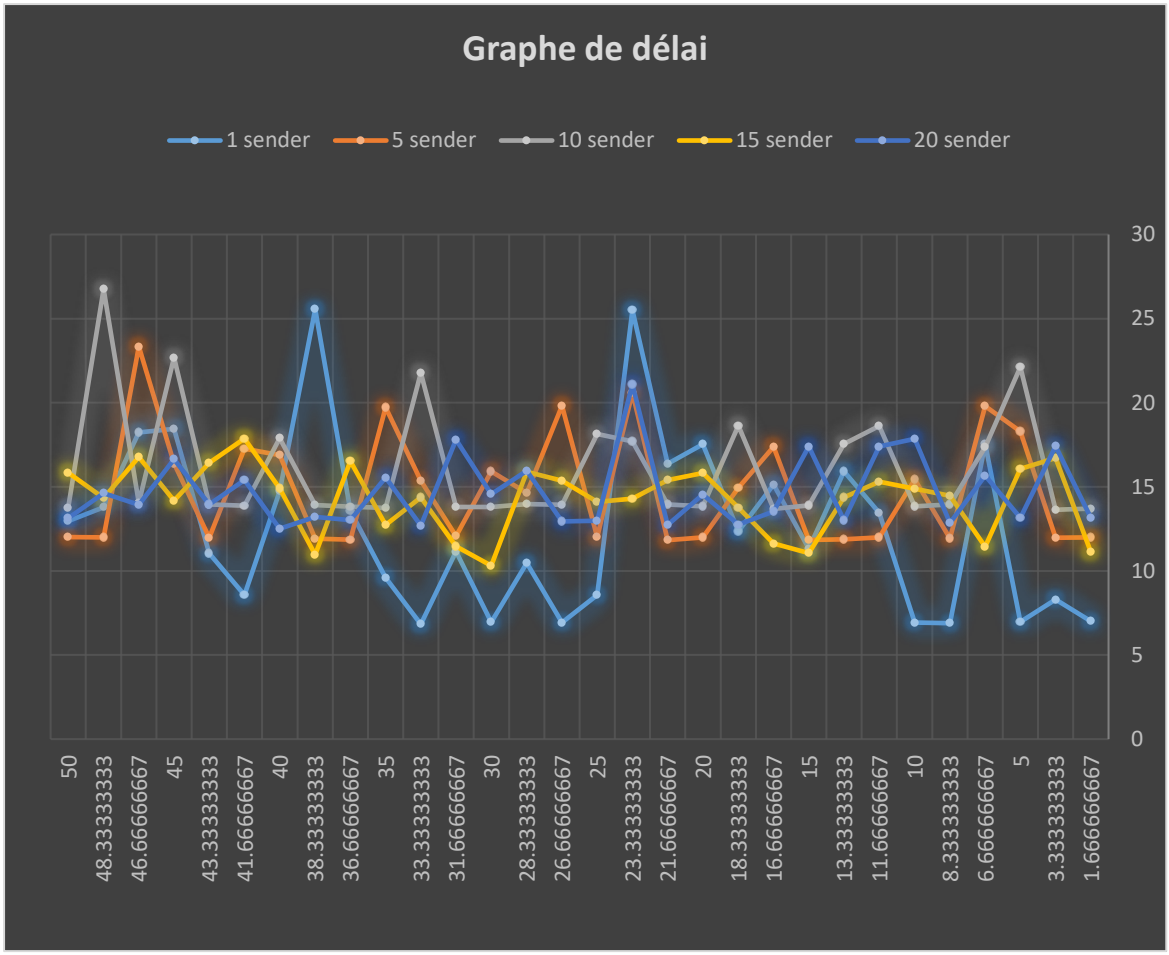

Figure 4.12 graphe de délai moyen

-La figure 4.12 représente les valeurs que nous avons eu après la simulation paramétrée, ou chaque trace représente un cas dépendant du nombre de sources impliqués.

-On observe que les graphes s'interfèrent beaucoup entre eux, et ne possèdent pas de méthode modéré ce qui nous laissent ressortirent que la valeur du délai moyen n'est pas liée aux nombre des sources de transmission. Mais plutôt du nombre de sauts nécessaires pour acheminer un paquet donné d'une source jusqu'au Sink.

b) Tracer l'énergie dissipée moyenne en fonction du nombre de sources :

20 sources	15 sources	10 sources	5 sources	1 source	Temps
0.410164	0.514753	0.634969	1.026655	1.113484	1.666667
0.345482	0.529527	0.599357	0.656958	0.865884	3.333333
0.367794	0.411621	0.503381	0.691996	1.119327	5
0.447207	0.508135	0.457724	0.545481	0.717127	6.666667
0.40832	0.533516	0.57588	0.676303	1.117472	8.333333
0.371309	0.505833	0.488907	0.705128	1.140655	10
0.363902	0.642395	0.492118	0.697194	0.894155	11.66667
0.511442	0.394787	0.501974	0.793473	0.730368	13.33333
0.425873	0.535954	0.545497	0.724822	0.891836	15
0.362535	0.486466	0.555521	0.650123	0.616035	16.66667
0.416061	0.490092	0.658229	0.740384	0.735124	18.33333
0.451443	0.425973	0.554247	0.682219	0.930996	20
0.48065	0.496211	0.548713	0.673976	0.883379	21.66667
0.360195	0.403986	0.518964	0.59451	0.649204	23.33333
0.411999	0.513413	0.835956	0.66565	0.866647	25
0.437202	0.467803	0.514106	0.498736	1.115342	26.66667
0.387923	0.450313	0.570616	0.86963	0.877497	28.33333
0.44007	0.596246	0.490295	0.598584	1.120444	30
0.427778	0.523589	0.605423	0.740213	0.878314	31.66667
0.487117	0.465229	0.558246	0.585671	1.171359	33.33333
0.427545	0.465597	0.51491	0.539145	0.87888	35
0.372494	0.436611	0.793254	0.703319	0.874084	36.66667
0.439288	0.538721	0.532662	0.862193	0.645988	38.33333
0.464311	0.467517	0.612245	0.59217	0.629264	40
0.353163	0.421429	0.517494	0.634667	0.866647	41.66667
0.416533	0.480409	0.635428	0.729711	0.897256	43.33333
0.403504	0.461899	0.462025	0.635592	0.765023	45
0.417994	0.500211	0.695075	0.525742	0.558169	46.66667
0.37513	0.482526	0.397963	0.656176	0.71957	48.33333
0.395755	0.413506	0.661873	0.754214	0.733794	50

Table 4.2. résultat de calcule d'énergie dissipée moyenne.

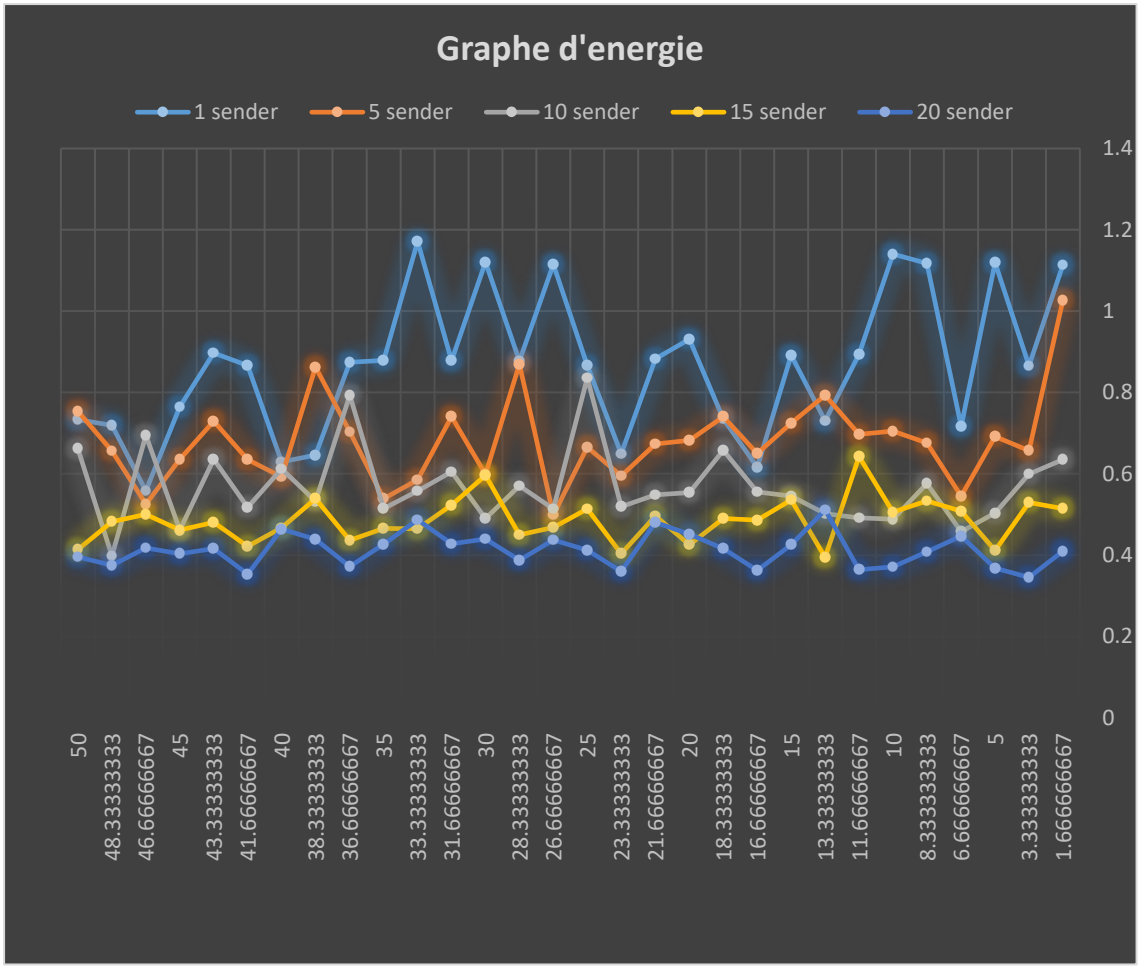

Figure 4.13 graphe d'énergie dissipée moyenne

-Nous observons que les graphes de la méthode moderée ont généralement un ordre comme suit: **(1 source > 5 sources > 10 sources > 15 sources > 20 sources).** Cela montre une relation entre le nombre des sources et de la quantité d'énergie consommée . Il est claire que plus le nombre des sources augmente ,la consommation d'énergie augmente .

4.5. Comparaison entre les résultats obtenus :

Dans ce qui suit, nous allons faire une comparaison entre les résultat obtenus en utilisant le simulateur NS-2 d'une part et l'outil UPPAAL d'une autre part, afin de valider notre modélisation faite sur ce dernier.
Les courbes suivants représentent des comparaisons de différents cas de consommation d'énergie dans les deux outils (NS-2 et UPPAAL).

1) Cas de 10 sources:

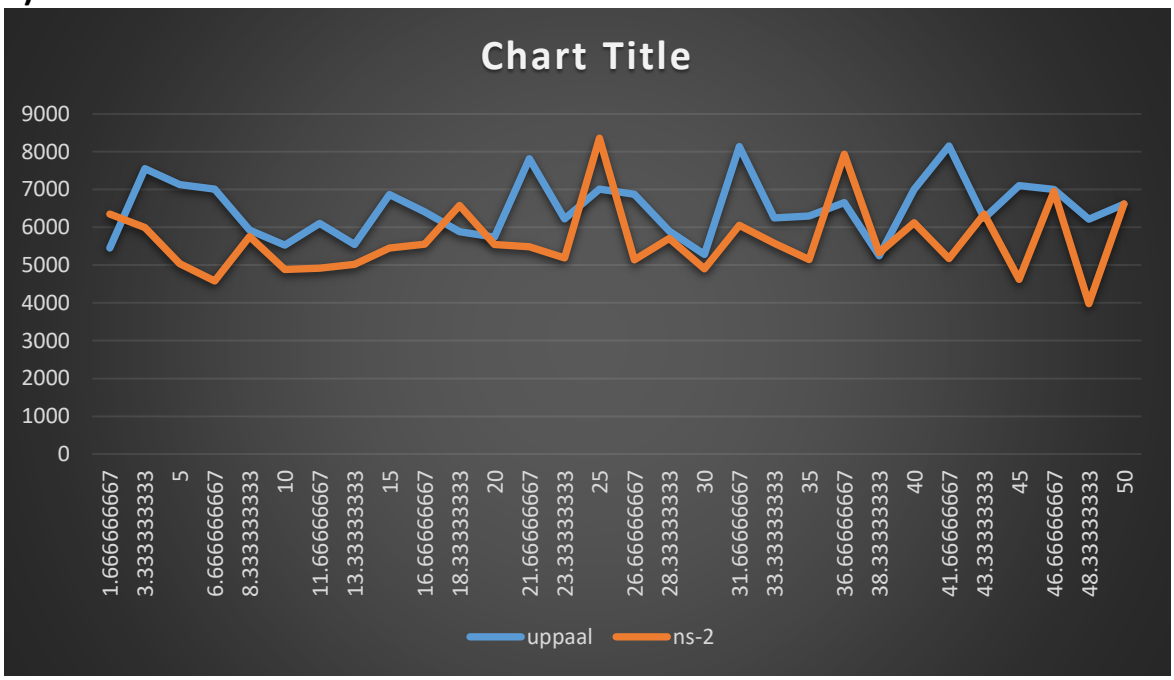

Figure 4.14 comparaison entre les résultats : cas de 10 sources

2) Cas de 20 sources:

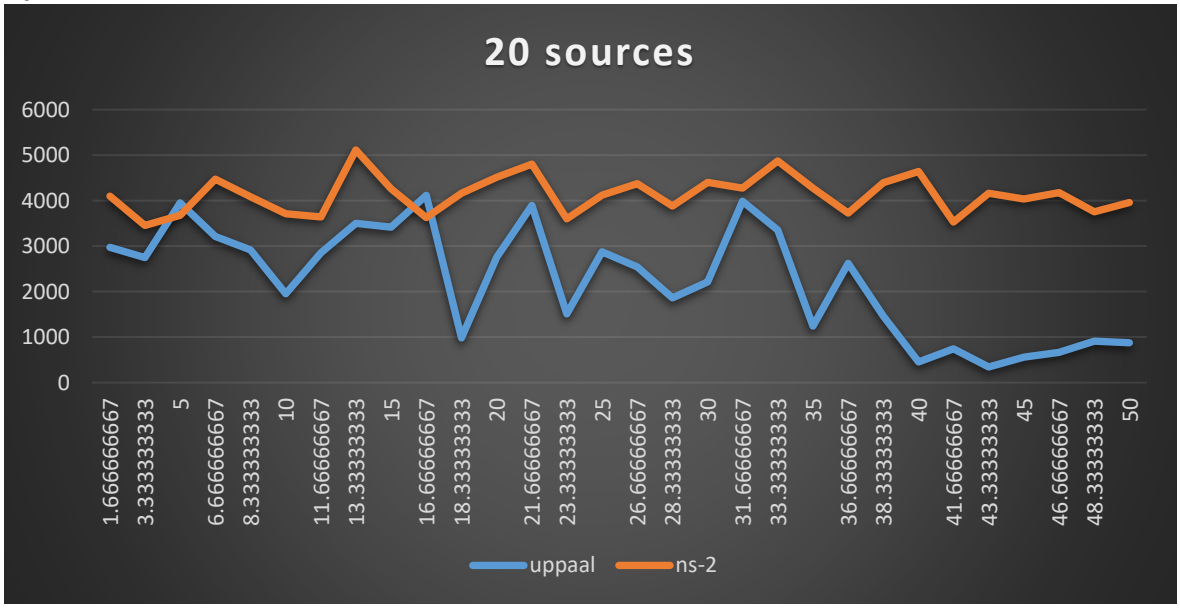

Figure 4.15 graphe comparable entre résultat du cas de 20 source

4.5.1. Discussion des résultats:

En observant la figure 4.14, nous constatons que les courbes sont dans la même plage des valeurs. Ce qui signifie une similitude entre la simulation faite sur NS-2, et celle faite sur Uppaal, ce qui valide notre modélisation (dans le cas de 10 sources). Cependant, nous pouvons remarquer un décalage entre les deux traces dans la figure 4.15. Ce décalage est dû au fait que les valeurs obtenues par NS-2 sont légèrement supérieures à celle obtenues par UPPAAL. Cela est expliqué par l'existence d'autres messages envoyés dans le réseau qui ne sont pas modélisés par UPPAAL. Ces messages sont liés au comportement des nœuds et pas aux règles du protocole. Parmi ces messages, nous pouvons citer ceux qui sont liés au "mode veille". Dans un RCSF, les nœuds inactifs pendant un certain temps passent en mode veille, ce qui engendre un changement dans la table de routage, vu que ces nœuds vont être considérés comme "en panne", ce qui implique plus de sauts qu'en habituel. Ce phénomène se manifeste le plus si le nombre de nœuds augmente.

4.6. Conclusion :

Ce chapitre a été consacré pour présenter notre travail. Nous avons vérifié quelques propriétés de ce protocole en utilisant le Model Checker de l'outil UPPAAL. La technique de Model Checking nous a permis d'explorer toutes les possibilités dans lesquelles notre RCSF peut l'être. Et aussi nous avons fait une simulation du mode de fonctionnement du protocole avec l'outil NS2. Nous avons le poussé ou maximum de ces limites de fonction du côté consommation d'énergie et aussi du côté de délais de transmission et ce qui nous a permet de tracer des courbes et de déduire les capacités de ce protocole ainsi que les facteurs qui affectent à ces capacités. Nous avons comparé les résultats pour valider notre modélisation et nous avons les discuté pour tirer des conclusions sur notre travail et comment l'améliorer.

Conclusion et perspectives

Ce travail rentre dans le cadre de la spécification et la vérification formelle. Ces deux dernières sont très importantes pour garantir la fiabilité de plusieurs types de systèmes. Ces systèmes sont souvent des systèmes critiques ou des systèmes où leur dysfonctionnement engendre des catastrophes. Pour cette raison, nous avons fait appel à ces méthodes formelles pour modéliser et vérifier et analyser les performances d'un protocole de communication très utilisé dans les RCSFs et qui représente la base de plusieurs autres protocoles de routage. Ce protocole est utilisé dans plusieurs domaines comme la surveillance territoriale, la protection des espèces rares et les systèmes d'alertes d'incendie dans les grandes forêts. La spécificité de ce protocole, nous a poussé à l'utiliser des méthodes formelles pour garantir son bon fonctionnement.

Ce travail nous a aidés à :

- Comprendre le fonctionnement général des RCSF et les défis de conception qui leurs sont attachés.
- Avoir une idée très claire sur les protocoles de routage au sein d'un RCSF, notamment le protocole : « Directed Diffusion ».
- Acquérir des connaissances plus au moins avancées sur la modélisation par les outils UPPAAL et NS2.
- Mieux comprendre le fonctionnement du protocole « Directed Diffusion » à travers des simulations.
- Vérifier des propriétés attendues de ce protocole.
- analyser des performances de ce protocole.

Nous allons essayer d'améliorer ce travail de la manière suivante :

- Prendre en charge plusieurs intérêts à la fois.
- Prendre en charge le changement de la topologie pour des cas spécifiques (des capteurs qui se déplacent).
- Utiliser un modèle statistique sur UPPAAL-SMC pour mieux analyser les performances de ce protocole.
- Optimiser le modèle si possible.

Références

[1] HADJADJI Samia, ‘‘Système de supervision des réseaux de capteurs sans fils’’. Université Taheri Mohamed Béchar, 2015.

[2] BENDERRADJI Safieddine, ‘‘Transmission en multi-canal dans les réseaux de capteurs sans fils’’. Université Mohamed Khider, 2015.

[3] Majda Moussa, ‘‘Vérification et configuration automatiques de pare-feux par model checking et synthèse de contrôleur’’, Université de Montrèal, 2014.

[4] ABI-AYAD Salima, ‘‘Evaluation du Protocole Directed Diffusion dans un réseau de capteurs sans fil’’. Université Abou Bakr Belkaid– Tlemcen, 2014.

[5] Djarallah Abdallah, “Spécification et vérification de protocoles dans les réseaux de capteurs sans fils’’. Université Mohamed Khider, 2012.

[6] Z. Mammeri, ‘‘Systèmes de transitions Automates à états finis’’, cours (M2P GLRE Génie Logiciel, logiciels Répartis et Embarqués), Cour master1 .2010/2011.

[7] YOUSEF Yaser, ‘‘Routage pour la Gestion de l’Energie dans les Réseaux de Capteurs Sans Fil’’. Université de Haute Alsace ,2010.

[8] Mr Kaid Omar Ilyes, ‘‘Modélisation d’un contrôleur de température d'un réacteur d'avion par Uppaal’’, Mémoire fin d'étude licence en informatique de l’université d’oran 2008-2009.

[9] K. Zeng, K. Ren, W. Lou and P. J. Moran, “Energy aware efficient geographic routing in lossy wireless sensor networks with environmental energy supply”. Kluwer Academic Publishers Hingham, MA, USA, 2009.

[10] F.Z. BENHAMIDA, Tolérance Aux Pannes Dans Les Réseaux De Capteurs Sans

Fil , thèse de magister, 2009.

[11] C. B. Abbas, R. González, N. Cardenas and L. J. G. Villalba, “A proposal of a wireless sensor network routing protocol”. Springer Science and Business Media.Telecommunication Systems. March 2008, pp. 61–68.

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

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

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

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

[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] M. Beigl, C. Decker, A. Krohn, T. Riedel, and T. Zimmer, “Low Cost Sensor Networks at Scale”. 7th International Conference on Ubiquitous Computing Demonstration Proceedings(ubicomp’05), 2005.

[18] 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.

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

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

[22] N. V. Subramanian, “Survey on Energy-Aware Routing and Routing Protocols for Sensor Networks”. Technical Report, Computer Science, University of North Carolina, Charlotte. 2004.

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

[24] Lyes KHELLADI et Nadjib BADACHE, ‘‘Les réseaux de capteurs : état de l’art’’. Université Houari Boumadien - Alger, 2004.

[25] 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.

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

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

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

[29] C.CHONG, S. KUMAR, "Sensor Networks Evolution, Opportunities, and Challenges" in IEEE Procedings, 2003, volume 91, numéro 8, pages 1247-1256.

[30] Faïez CHARFI, ‘‘Une approche d’interfaçage de CoD à UPPAAL pour la spécification et la vérification des systèmes temps réel’’, Mémoire de D.E.A en informatique d’UNIVERSITE DE TUNIS EL MANAR, Septembre 2003.

[31] S. Lindsey and C. Raghavendra, “PEGASIS: Power-Efficient Gathering in Sensor Information Systems”. Proceedings of the IEEE Aerospace Conference, vol. 3, Big Sky, MT, USA, March 2002, pp. 1125-1130.

[32] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and J. Anderson3, “Wireless sensor networks for habitat monitoring”. Atlanta, Georgia,USA, 2002.

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

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

[35] 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.

[36] B. Krishnamachari, D. Estrin, and S. Wicker, “Modeling Data Centric Routing in Wireless Sensor Networks”. In the Proceedings of IEEE INFOCOM, New York, NY, June 2002.

[37] R. Shah and J. Rabaey, “Energy Aware Routing for Low Energy Ad Hoc Sensor Networks”. In the Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Orlando, FL, March 2002.

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

[39] I. AKYILDIZ, W. SU, Y. SANKARASUBRAMANIAM, E. I CAYIRCI, "A Survey on Sensor Networks" in IEEE Communications Magazine, Aout 2002, numéro 8, pages 102-114.

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

[41] A. Manjeshwar and D. P. Agrawal, “TEEN : A Protocol for Enhanced Efficiency in Wireless Sensor Networks”. 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, San Francisco, CA, April 2001.

[42] L. Doherty, K. S. J. Pister, and L. El Ghaoui, “Convex position estimation in wireless sensor networks”. In Proceedings of the IEEE INFOCOM, vol.3, Alaska, 2001, pp.1655-1663.

[43] Y. Yu, D. Estrin, and R. Govindan, “Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks,” UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023, May 2001.

[44] Y. Xu, J. Heidemann, and D. Estrin, “Geography-informed Energy Conservation for Ad hoc Routing”. In Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM’01), Rome,Italy, July 2001, pp.70-84.

[45] 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.

[46] 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.

[47] 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.

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

[49] C. Intanagonwiwat, R. Govindan and D. Estrin, “Directed diffusion: A scalable and robust communication paradigm for sensor networks”. In the Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00), Boston, MA, August 2000.

[50] D. Estrin, R. Govindan, J. Heidemann, and Satish Kumar, “Next century challenges: Scalable Coordination in Sensor Networks”. In the Proceedings of the 5th annual ACM/IEEE international conference on Mobile Computing and Networking (MobiCom’99), Seattle, WA, August 1999.

[51] E. M. Royer and C. K. Toh, “A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks”. IEEE Personal Communications, Vol. 6, No. 2, April 1999, pp. 46-55.

[52] V. Rodoplu and T. H. Ming, “Minimum energy mobile wireless networks”. IEEE Journal of Selected Areas in Communications, Vol. 17, No. 8, 1999, pp. 1333-1344.

[53] S. Singh and C. Raghavendra, “PAMAS: Power aware multi-access protocol with signaling for ad hoc networks”. ACM Computer Communication Review, July 1998, pp. 5-26.

[54] H. Qi, P. T. Kuruganti and Y. Xu, “The Development of Localized Algorithms in Wireless Sensor Networks”. Published on 202 SENSORSISSN, 22 July, pp. 1424- 8220.

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

[56] chaita chouayb , ‘‘ Évaluation et performance de protocole de routage dans les réseau vanet’’. Université Mohamed Khider, 2017.