Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

**République Algérienne Démocratique et Populaire**
**Université Abou Bakr Belkaid– Tlemcen**
**Faculté des Sciences**
**Département d'Informatique**

**Mémoire de fin d'études**

**Pour l'obtention du diplôme de Master en Informatique**

***Option:*** Modèle Intelligent et Décision *(M.I.D)*

*Thème*

# Etude d'un protocole de routage basé sur les colonies de Fourmis dans les réseaux de capteurs sans fil

**Réalisé par :**

- ***Mr SAHRAOUI belkheyr***

*Présenté le 2013 devant le jury composé de MM.*

- *M^r^ Benamar Abdelkrim* *(Président)*
- *M^me^ LABRAOUI Nabila* *(Encadreur)*
- *Mr Lehsaini Mohamed* *(Examinateur)*
- *Mr Benmammar Badr* *(Examinateur)*

Année universitaire : 2012-2013

# *Remerciements*

*Grâce à Dieu vers lequel vont toutes les louanges, ce travail s'est accompli.*

*Je tiens à exprimer mes vifs remerciements envers toutes les personnes qui ont contribué au bon déroulement de ce travail.*

*En particulier, j'exprime ma gratitude à mon encadreur Mme Labraoui Nabila, Chercheur au laboratoire STIC et maître de Conférences à l'université Abou Bekr Belkaid de Tlemcen.*

*Je n'oublie pas mes parents pour leur contribution, leur soutien et leur patience, mes proches et amis qui m' ont soutenu et encouragé.*

*Je remercie aussi vivement tous les responsables de l'université de Tlemcen ainsi que toutes les personnes qui m'ont aidé de près ou de loin de nos études.*

# *Dédicace*

*Je remercie Dieu de m'avoir donné le courage pour accomplir ce modeste travail que je dédie :A mes très chers parents qui sont la bougie qui illumine ma vie*

*A mon frère Mohamed*

*A mes sœurs Fatma, Assia, Amal*

*A mes très chers amis Amine et Mokhtar et Boumedienne*

*A toute ma grande famille, à tous mes amis et à tous mes enseignants*

*Et à tous ceux que j'aime et qui m'aiment*

*SAHRAOUI belkheyr*

# SOMMAIRE

## Liste des figures :



## Liste des tableaux :

## *Introduction générale*

Les réseaux de capteurs sans fil (RCSF)- Wireless Sensor Networks (WSN) - sont considérés comme un type spécial de réseaux ad hoc. Les nœuds de ce type de réseau consistent en un grand nombre de micro-capteurs capables de récolter et de transmettre des données environnementales d'une manière autonome. La position de ces nœuds n'est pas obligatoirement prédéterminée. Ils sont dispersés aléatoirement à travers une zone géographique, appelée champ de captage, qui définit le terrain d'intérêt pour le phénomène capté. Les données captées sont acheminées à un nœud considéré comme un "point de collecte", appelé nœud puits (ou sink). Ce dernier peut être connecté à l'utilisateur du réseau via Internet ou un satellite. Ainsi, l'usager peut adresser des requêtes aux autres nœuds du réseau, précisant le type de données requises et récolter les données environnementales captées par le biais du nœud puits.

Si leurs perspectives d'utilisation sont claires et attrayantes, les problématiques qu'engendrent ces réseaux n'en sont pas moins nombreuses. A priori, ils ne dépendent d'aucune infrastructure et les capteurs n'ont aucune information relative au réseau auquel ils appartiennent. De plus, étant construits de façon ad hoc, ces réseaux doivent être auto-organisant. Parmi les problèmes cruciaux, nous citons celui du routage, qui consiste à acheminer un message d'un capteur vers un autre. Souvent, les réseaux contiennent une station de base chargée de collecter l'ensemble des informations perçues par les capteurs. Il s'agit alors de transmettre ces informations point à point vers cette station de base.

Bien d'autres problèmes tels que l'adressage ou la diffusion sont liés à ce type de réseau. Chaque méthode proposée doit assurer l'auto-stabilisation du réseau en garantissant la convergence vers une solution stable. D'autre part, l'énergie des capteurs étant limitée, cette contrainte doit être prise en compte afin d'allonger la durée de vie du réseau.

Ce mémoire se focalise sur la problématique du routage statique dans les réseaux de capteurs sans fil. Nous nous sommes également intéressés de près à l'algorithme de routage basé sur les colonies de fourmis nommé ACAWSN [SJ12] afin de

l'implémenter et d'étudier ses performances selon les métriques de la longueur de chemin à trouver et l'overhead.

La suite de ce document est constituée de 4 chapitres :

**Le chapitre 1** : nous présentons une description générale des réseaux de capteurs sans fil ainsi que leurs caractéristiques, contraintes et spécificités.

**Le chapitre 2** : est consacré à la problématique du routage et présente différent types des protocoles de routage.

**Le chapitre 3** : présente une description des algorithmes basés sur les colonies de fourmis et leur relation avec l'informatique.

**Le chapitre 4** : constitue le cœur de notre travail, dans ce chapitre nous présentons le System TinyOS (système embarque spécifique pour les capteurs sans fil :TinyOS 2.x) et le simulateur TOSSIM avec une brève description de ces composants, ses principales caractéristiques et fonctionnalités. Par la suite nous donnons les résultats sous forme de graphes de plusieurs simulations, effectuées pour obtenir des mesures pour voir la performance de protocole de routage ACAWSN [SJ12] qui est basé sur l'algorithme de colonie fourmis.

Ce travail est terminé par une conclusion générale et une bibliographie.

# *Chapitre I :*

# *Généralités sur les Réseaux de capteurs sans fil (WSN: Wireless Sensor Networks)*

# 1. Introduction

De nombreux systèmes nécessitent de prendre en compte l'environnement pour mesurer les phénomènes physiques afin de prendre les décisions nécessaires. Les progrès de ces dernières années en microélectronique et micromécanique ont permis de concevoir des capteurs de plus en plus petits, de plus en plus performants, autonomes et dont les capacités énergétiques ont évolué avec le temps. D'autre part, les techniques de réseaux mobiles permettent d'affranchir des fils et donc de déployer facilement des réseaux de capteurs dans les endroits même difficiles à y accéder.

De plus, dans la vie courante, l'utilisation des capteurs sans fil est en demande croissante pour la supervision et la sécurité. Les industries proposent alors des capteurs sans fil qui peuvent renseigner l'utilisateur sur plusieurs données. Ces capteurs peuvent être reliés formant ainsi un réseau sans fil se basant sur des protocoles pour se communiquer et proposer des programmes et des réseaux embarqués [KP09].

# 2. Histoire des réseaux de capteurs

Dans les années 1990, dans le monde de la recherche, est apparue une idée qui paraissait plutôt un rêve pour cette époque : imaginer un système nerveux central pour la Terre, capable de surveiller en temps réel les événements, ayant comme principaux bénéfices de pouvoir empêcher les accidents et d'économiser l'énergie. (*Cette poussière intelligente a mis longtemps à apparaître*) dit le professeur Pister, de l'Université de Californie à Berkeley. (*J'ai inventé l'expression il y a 14 ans. La poussière vraiment futée a mis le temps, mais elle est finalement arrivée*).

Aujourd'hui les réseaux de capteurs sont devenus des systèmes pouvant atteindre un très grand nombre de nœuds, avec une zone de couverture déterminée et déployés d'une manière plus ou moins dense dans un environnement hétérogène dont on mesure ainsi son état global. Les derniers progrès en terme de miniaturisation, ainsi que le remplacement du câblage classique par des technologies de communication radio, ont généré de nouvelles catégories d'applications qui visent de nombreux domaines : l'aéronautique, l'automobile, le médical, l'environnement, etc. De plus, les progrès des communications sans fil permettent aujourd'hui de répondre des exigences peu envisageables auparavant.

# 3. Présentation des réseaux de capteurs

## 3.1. Définition d'un capteur

Un capteur est un dispositif équipé de fonctionnalités de sensation avancées. Il mesure ou détecte un événement réel, comme le mouvement, la chaleur ou la lumière et convertit la valeur mesurée dans une représentation analogique ou numérique. Il prélève des informations et élabore à partir d'une grandeur physique (information d'entrée), une autre grandeur physique de nature électrique.

Un capteur est composé de 4 unités (voir la Figure 1.1) :

- ***l'unité d'acquisition*** : composée d'un capteur qui obtient des mesures sur les paramètres environnementaux et d'un convertisseur Analogique/Numérique qui convertit l'information relevée et la transmet à l'unité de traitement.
- ***l'unité de traitement*** : composée d'un processeur et d'une mémoire intégrant un système d'exploitation spécifique (TinyOS , par exemple). Cette unité possède deux interfaces, une interface pour l'unité d'acquisition et une interface pour l'unité de communication. Elle acquiert les informations en provenance de l'unité d'acquisition et les envoie à l'unité de communication. Cette unité est chargée aussi d'exécuter les protocoles de communications qui permettent de faire collaborer le capteur avec d'autres capteurs. Elle peut aussi analyser les données captées.
- ***l'unité de communication*** : unité responsable de toutes les émissions et réceptions de données via un support de communication radio. Elle peut être de type optique (les capteurs Smart Dust), ou de type radiofréquence (MICA2, par exemple).
- ***la batterie*** : un capteur est muni d'une batterie pour alimenter tous ses composants. Cependant, à cause de sa taille réduite, la batterie dont il dispose est limitée et généralement irremplaçable. Pour cela, l'énergie est la ressource la plus précieuse puisqu'elle influe directement sur la durée de vie des capteurs.

Il existe des capteurs qui sont dotés d'autres composants additionnels comme le système de positionnement GPS (Global Positioning System) et un mobilisateur lui permettant le déplacement.

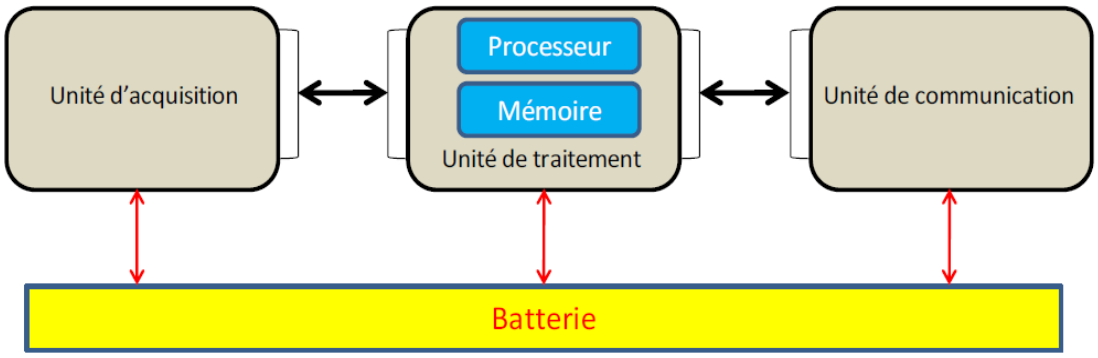



**Figure 1. 1 : Architecture physique d'un capteur**

## 3.2. Définition d'un RCSF ou WSN (Wireless Sensor Network)

Un réseau de capteurs sans fil est un type spécial de réseaux ad hoc avec un grand nombre de nœuds qui sont des micro-capteurs capables de recevoir et de transmettre des données environnementales d'une manière autonome sans intervention humaine [NB04]. La position de ces nœuds n'est pas obligatoirement prédéterminée, ils peuvent être aléatoirement dispersés dans une zone géographique appelée « champ de captage » correspondant au terrain d'intérêt pour le phénomène capté (par exemple : lâchée de capteurs sur un volcan pour étudier les phénomènes vulcanologiques et leurs évolutions).

Le réseau possède en général un nœud particulier, la base (ou sink), connectée avec les autres nœuds par un réseau filaire est reliée à une alimentation électrique.

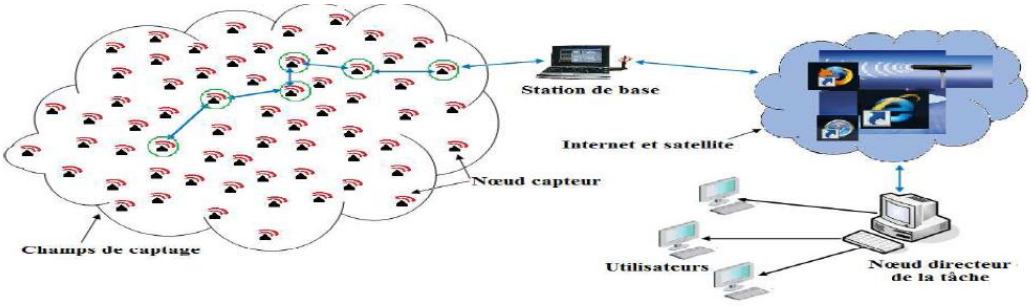



**Figure 1. 2 : Architecture de communication d'un RCSF.**

## 4. Architecture de communication dans les réseaux de capteurs

Le modèle de communication comprend cinq couches qui ont les mêmes fonctions que celles du modèle OSI ainsi que trois couches pour la gestion d'énergie, la gestion de la mobilité et la gestion des tâches.

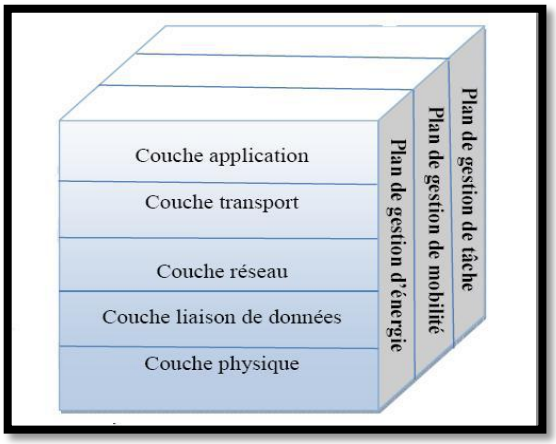



**Figure 1. 3: Modèle en couches du réseau de capteurs sans fil**

**Rôles des couches :**

- ✓ **Couche physique :** Matériels pour envoyer et recevoir les données.
- ✓ **Couche liaison de données :** Gestion des liaisons entre les nœuds et les stations de base, contrôle d'erreurs.
- ✓ **Couche réseau :** Routage et transmission des données.
- ✓ **Couche transport :** Transport des données, contrôle de flux.
- ✓ **Couche application :** Interface pour les applications au haut niveau.
- ✓ **Plan de gestion d'énergie :** Contrôle l'utilisation d'énergie.
- ✓ **Plan de gestion de mobilité :** Gestion des mouvements des nœuds.
- ✓ **Plan de gestion de tâche :** Balance les tâches entre les nœuds afin d'économiser de l'énergie.

## 5. Types de réseaux de capteurs sans fil

### 5.1. Réseaux de poursuite

Ces réseaux sont généralement développés par l'armée, ils peuvent servir à surveiller toutes les activités d'une zone stratégique ou d'accès difficile, ainsi on pourra détecter des agents chimiques, biologiques ou des radiations avant des troupes. On peut aussi penser à des capteurs embarqués sur les soldats pour faciliter leur guidage et le contrôle de leur position depuis la base.

### 5.2. Réseaux de collection des données d'environnement

Les nœuds de ce type de réseau peuvent avoir plusieurs fonctionnalités et différents types de capteurs. Ce type de réseau nécessite généralement un flux de données faible, une durée de vie importante ; il sert à la collecte périodique des données environnementales puis leur transmission vers la station de base.

### 5.3. Réseaux de surveillance et sécurité

La différence entre ce réseau et le réseau de collection d'environnement est que les nœuds ne transmettent pas l'ensemble des données collectées mais seulement les rapports concernant une violation de la sécurité. Ce sont en général des nœuds fixes qui contrôlent d'une façon continue la détection d'une anomalie dans le fonctionnement d'un système. Ainsi les altérations dans la structure d'un bâtiment, suite à un séisme, pourraient être détectées par des capteurs intégrés dans les murs ou dans le béton, sans alimentation électrique ou autres connexions filaires.

## 6. Fonctionnement d'un réseau de capteurs

Les données captées par les nœuds sont acheminées grâce à un routage multi-saut à un nœud considéré comme un "point de collecte" appelé nœud-puits (ou sink). Ce dernier peut être connecté à l'utilisateur du réseau via Internet, un satellite ou un autre système. L'usager peut adresser des requêtes aux autres nœuds du réseau, précisant le type de données requises pour récolter les données environnementales captées par le biais du nœud-puits.

## 7. Topologies des réseaux de capteurs

Un réseau de capteurs sans fil est composé d'un ensemble de nœuds capteurs et des Gateway qui s'occupent de collecter les données des capteurs et de les transmettre à l'utilisateur via l'internet ou le satellite, il existe plusieurs topologies pour les réseaux de capteurs.

### 7.2. Topologie en étoile

La topologie en étoile est un système uni-saut. Tous les nœuds envoient et reçoivent seulement des données avec la station de base. Cette topologie est simple et

elle demande une faible consommation d'énergie, mais la station de base est vulnérable et la distance entre les nœuds et la station est limitée.

**Avantage** : simplicité et faible consommation d'énergie des nœuds, moindre latence de communication entre les nœuds et la station de base.

**Inconvénient** : la station de base est vulnérable, car tout le réseau est géré par un seul nœud.

## 7.3. Topologie en toile (Mesh Network)

La topologie en toile est un système multi-saut. La communication entre les nœuds et la station de base est possible. Chaque nœud a plusieurs chemins pour envoyer les données. Cette topologie a plus de possibilités de passer à l'échelle du réseau, avec redondance et tolérance aux fautes, mais elle demande une consommation d'énergie plus importante.

**Avantage** : Possibilité de passer à l'échelle du réseau, avec redondance et tolérance aux fautes.

**Inconvénient** : Une consommation d'énergie plus importante est induite par la communication multi-sauts. Une latence est créée par le passage des messages des nœuds par plusieurs autres avant d'arriver à la station de base.

## 7.4. Topologie hybride

La topologie hybride est un mélange des deux topologies ci-dessus. Les stations de base forment une topologie en toile et les nœuds autour d'elles sont en topologie étoile. Elle assure la minimisation d'énergie dans les réseaux de capteurs.

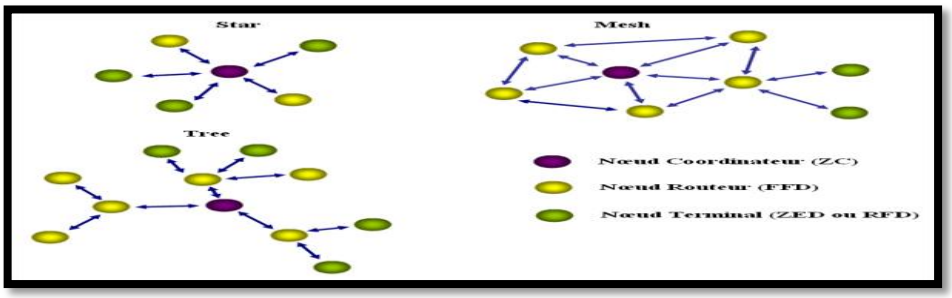



**Figure 1. 4 : Topologies des réseaux de capteurs**

## 8. Les plates-formes existantes [LN12].

Comme un certain nombre de technologies connues à ce jour, les nœuds de capteurs sans fil doivent être nés d'un projet militaire, ce qui entrave la mise en place d'un chrono graphie précise de leur développement. Cependant, le titre du premier prototype de nœuds de capteurs sans fil identifiable dans la bibliographie correspond sans aucun doute au module LWIM (Low-power Wireless Integrated Microsensors) développé dans le milieu des années 90 par l'Agence pour les Projets de Recherche Avancée de Défense (DARPA) des Etats-Unis et l'UCLA. Il s'agissait d'un géophone équipé d'un capteur de transmission radio fréquences et d'un contrôleur PIC. Depuis un peu plus de 10 ans, la technologie des capteurs sans fil a beaucoup évolué. Les modules deviennent de plus en plus petits et les durées de vie prévues augmentent. Aujourd'hui, le marché de noeuds a été ouvert à l'industrie. Le fournisseur le plus connu est *Crossbow Inc*., avec son offre de capteurs Mica2 et MicaZ (voir Figure 1.5).

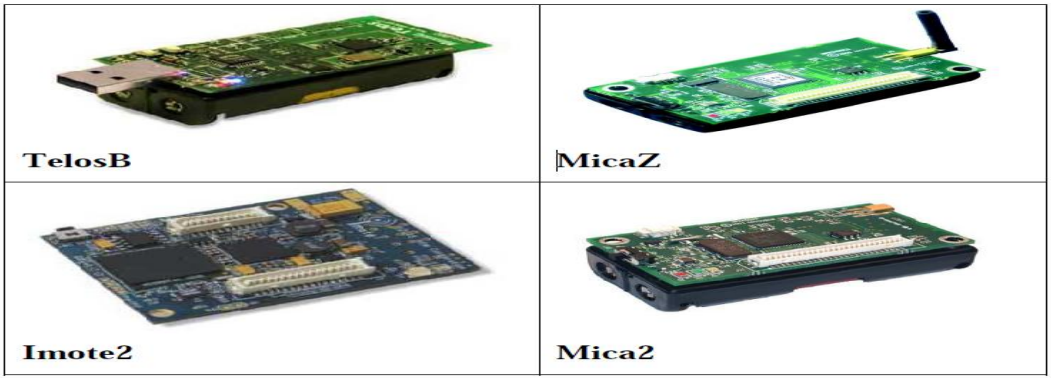



**Figure 1. 5: Quelques modèles de capteurs sans fil.**

## 9. Domaines d'application 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. Ils s'insèrent notamment dans d'autres systèmes tels que le contrôle et l'automatisation des chaînes de montage.

Ils permettent de collecter et de traiter des informations complexes provenant de l'environnement (météorologie, étude des courants, de l'acidification des océans, de la dispersion de polluants, etc).

Certains prospectivistes pensent que les réseaux de capteurs pourraient révolutionner la manière même de comprendre et de construire les systèmes physiques complexes, notamment dans les domaines militaire, environnemental, domestique, sanitaire et de la sécurité, etc.

- **Applications militaires :** On peut penser à un réseau de capteurs déployé sur un endroit stratégique ou d'accès difficile, afin de surveiller toutes les activités des forces ennemies, ou d'analyser le terrain avant d'y envoyer des troupes (détection d'agents chimiques, biologiques ou de radiations).
- **Applications domestiques :** En plaçant, sur le plafond ou dans le mur, des capteurs, on peut économiser l'énergie en gérant l'éclairage ou le chauffage en fonction de la localisation des personnes.
- **Applications environnementales :** Les réseaux de capteurs sont beaucoup appliqués dans ce domaine pour détecter des incendies, surveiller des catastrophes naturelles, détecter des pollutions et suivre des écosystèmes.
- **Applications agricoles :** Dans les champs agricoles, les capteurs peuvent être semés avec les graines. Ainsi, les zones sèches seront facilement identifiées et l'irrigation sera donc plus efficace.
- **Applications médicales :** Les réseaux de capteurs ont aussi des développements dans le domaine de diagnostic médical. Par exemple, des micro-caméras sont capables, sans avoir recours à la chirurgie, de transmettre des images de l'intérieur d'un corps humain avec une autonomie de 24 heures.
- **Applications transportés :** Il est possible d'intégrer des nœuds capteurs au processus de stockage et de livraison. Le réseau ainsi formé, pourra être utilisé pour connaître la position, l'état et la direction d'un paquet ou d'une cargaison.


**Figure 1. 6 : Applications des réseaux de capteurs**

# 10. Caractéristiques et Limites des réseaux de capteurs

## 10.1. Caractéristiques

**Absence d'infrastructure** : Les réseaux ad hoc, en général, se distinguent des autres réseaux mobiles par la propriété d'absence d'infrastructure préexistante et de tout genre d'administration centralisée. Les hôtes mobiles sont responsables d'établir et de maintenir la connectivité du réseau d'une manière continue.

**Taille** : Un grand nombre de nœuds dispersés aléatoirement (des réseaux de 10000 nœuds peuvent être envisagés)

**Contrainte d'énergie** : Dans plusieurs applications, les nœuds de capteurs sont placés dans des surfaces distantes, le service du nœud peut ne pas être possible, dans ce cas la durée de vie du nœud peut être déterminée par la durée de vie de la batterie, ce qui exige la minimisation des dépenses d'énergies.

**Topologie dynamique** : Les capteurs peuvent être attachés à des objets mobiles qui se déplacent d'une façon libre et arbitraire rendant ainsi, la topologie du réseau fréquemment changeante.

**Auto organisation du réseau** : Ceci peut être nécessaire dans plusieurs cas. Par exemple, un réseau comportant un grand nombre de nœuds placés dans des endroits hostiles où la configuration manuelle n'est pas faisable, doit être capable de s'auto-organiser. Un autre cas est celui où un nœud est inséré ou retiré (à cause d'un manque

d'énergie ou de destruction physique), ainsi le réseau doit être capable de se reconfigurer pour continuer sa fonction.

**Sécurité physique limitée** : Les RCSF mobiles sont plus touchés par les paramètres de sécurité que les réseaux filaires classiques. Cela se justifie par les contraintes et limitations physiques qui font que le contrôle des données transférées doit être minimisé.

## 10.2. Limites

- Les ressources de calcul et de mémoire des nœuds sont relativement faibles, par exemple les nœuds de capteur de type "mote" sont composés d'un microcontrôleur 8-bits 4MHz, 40 KOctets de mémoire et une radio avec un débit d'environ 10 kbps.
- Non seulement les capacités des nœuds sont faibles, mais en plus ils opèrent sur des piles et par conséquent ont une durée de vie limitée.
- L'énergie limitée des capteurs est probablement la caractéristique la plus pénalisante, le plus grand des défis dans le domaine des réseaux de capteurs reste de concevoir des protocoles, entre autre de sécurité, qui minimisent l'énergie afin de maximiser la durée de vie du réseau. En d'autres mots, l'énergie est sans aucun doute la ressource qui convient pour gérer avec la plus grande attention.

## 11. Conclusion

Dans ce chapitre, nous avons présenté les réseaux de capteurs, en parlant sur l'architecture, composants, fonctionnement, topologies utilisés, applications, ainsi que les caractéristiques et limites. Ainsi nous avons montré l'importance des réseaux de capteurs sans fil, qui sont en plein développement et deviennent de plus en plus répandus.

Actuellement, ils constituent un thème de recherche très dynamique, tiré vers le haut, par leurs utilisations dans divers domaines. En effet, leurs applications sont de plus en plus nombreuses et diversifiées. Cependant la réalisation de ces applications pose de grands défis auxquels il faut répondre ; le routage de ces réseaux est l'un des défis les plus importants à considérer.

Dans le chapitre suivant, nous introduirons en détail le routage dans les réseaux du capteur sans fil.

# *Chapitre II :*

# *Le routage dans les réseaux des capteurs sans fil*

# 1. Introduction

L'objectif principal d'un protocole de routage pour un réseau de capteurs sans fil est l'établissement correct et efficace d'itinéraires entre une paire de nœuds afin que des messages puissent être acheminés. Le protocole de routage permet aux nœuds de se connecter directement les uns aux autres pour relayer les messages par des sauts multiples et de transmettre les données vers un point de collecte.

Nous présenterons dans ce chapitre un état de l'art des principaux protocoles de routage dans les réseaux RCSF car la présentation de ces protocoles nous permettra de mieux analyser le fonctionnement.

# 2. Contraintes de routage dans les réseaux de capteurs sans fil

Le routage dans les réseaux de capteurs diffère de celui des réseaux Ad Hoc dans les points suivants :

- il n'est pas possible d'établir un système d'adressage global pour le grand nombre de nœuds.
- les applications des réseaux de capteurs exigent l'écoulement de données mesurées depuis des sources multiples vers la destination finale « *Sink* ».
- les différents capteurs peuvent générer produire les mêmes données à proximité d'un phénomène (problème de la redondance des données).
- les nœuds capteurs exigent ainsi une gestion soigneuse des ressources.

En raison de ces différences, de nouveaux protocoles de routage ont été proposés dans les réseaux de capteurs.

# 3. Classification des protocoles de routage dans les RCSF

Les protocoles de routage dans les réseaux peuvent être classés selon deux concepts :

- la structure de réseau,
- le type de protocole.

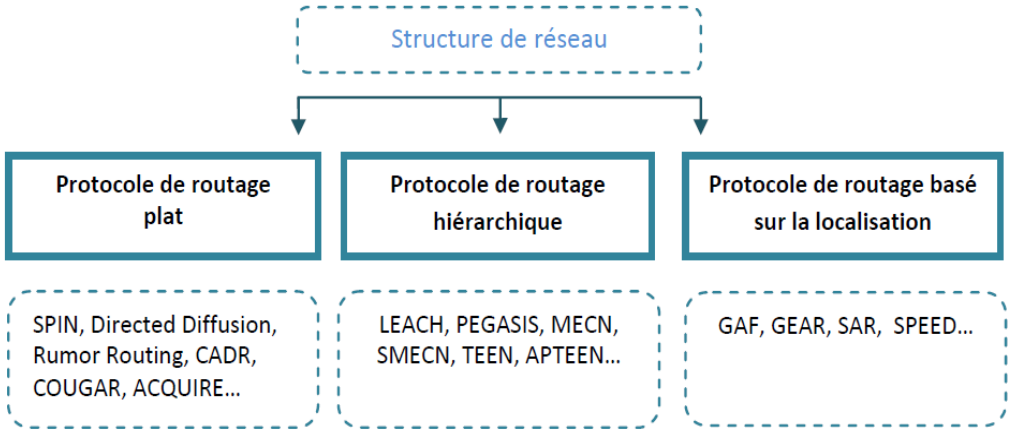



**Figure 2. 1 : Protocoles de routage pour les RCSF selon la structure du réseau.**

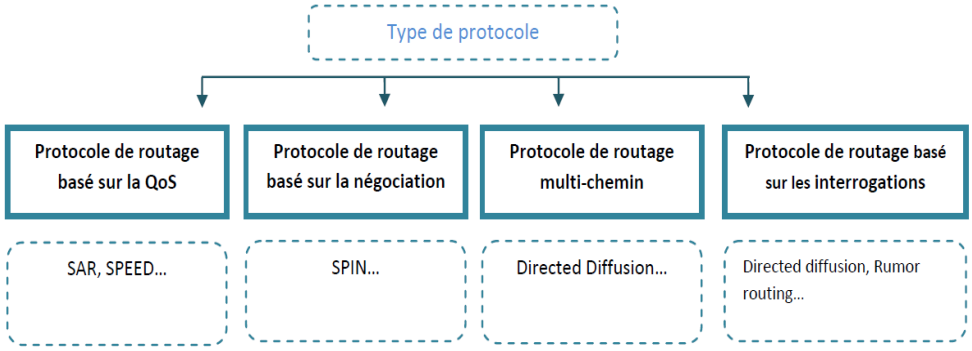



**Figure 2. 2 : Protocoles de routage pour les RCSF selon le type de protocole.**

## 3.1 Protocole de routage multi-chemin

Il se base sur l'adoption de plus qu'un chemin menant vers la destination, et ce, pour avoir des chemins de secours si jamais le chemin principal serait rompu.

## 3.2 Protocole de routage basé sur la négociation des données

En détectant le même phénomène, les nœuds capteurs inondent le réseau par les mêmes paquets de données. Ce problème de redondance peut être résolu en employant des protocoles de routage basés sur la négociation. En effet, avant de transmettre, les nœuds capteurs négocient entre eux leurs données en échangeant des paquets de signalisation spéciales, appelés *META-DATA.* Ces paquets permettent de vérifier si les nœuds voisins disposent des mêmes données à transmettre [DE07]. Cette procédure garantit que seules les informations utiles seront transmises et élimine la redondance des données.

### 3.3 Protocole de routage basé sur les interrogations

La collecte des informations sur l'état de l'environnement est initiée par des interrogations envoyées par le noeud « *Sink* ».

### 3.4 Protocole de routage basé sur la QoS

Ce type de protocoles tend à satisfaire certaines métriques, pendant la transmission des données vers la destination finale. Parmi ces métriques, nous citons : le délai de bout en bout, la gigue, PDR (*Paquet Delivery Ratio*), énergie consommée.

### 3.5 Les protocoles de routage plat (flat based-routing)

Ces protocoles considèrent que tous les nœuds sont identiques, c'est à dire ont les mêmes fonctions à exécuter sauf le nœud de contrôle (*sink* ) qui est chargé de collecter toutes les informations issues des différents nœuds capteurs pour les transmettre vers l'utilisateur final. La décision d'un nœud de router des paquets vers un autre dépendra de sa position et pourra être remise en cause au cours du temps.

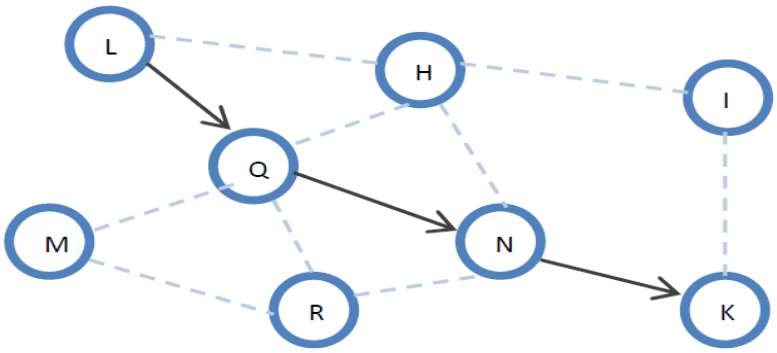



**Figure 2. 3 : Routage plat.**

### 3.6 Les protocoles de routage hiérarchique

Ces protocoles fonctionnent en confiant des rôles différents aux nœuds du réseau. Certains nœuds sont sélectionnés pour exécuter des fonctions particulières. Un nœud peut être, par exemple, une passerelle pour un ensemble de nœuds. Dans ce cas, le routage devient plus simple, puisqu'il s'agit de passer par les passerelles pour atteindre le nœud destination qui lui est directement attaché.

Un exemple est donné par la figure 2.4 : Pour que les paquets générés par le nœud F atteignent le nœud L, ils doivent passer par les passerelles P, S et R.

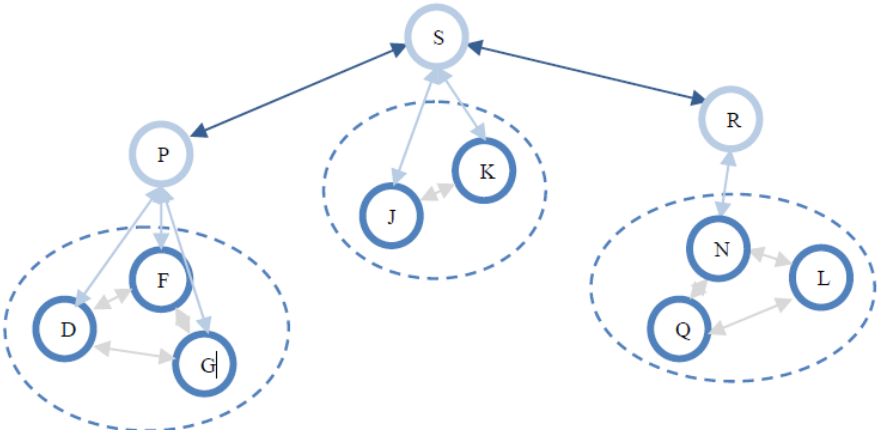



**Figure 2. 4 : Routage hiérarchique.**

Le principe des protocoles de routage hiérarchique est basé essentiellement sur les nœuds passerelles. En fait, les nœuds ordinaires savent que si le destinataire n'est pas dans leur voisinage direct, il suffit d'envoyer la requête à la passerelle qui la prendra en charge. À son tour, elle transmettra cette requête vers le nœud ciblé. Ce type de routage présent de nombreux avantages pour les réseaux dont leurs nœuds sont sédentaires et disposent de suffisamment d'énergie [DE07].

## 3.7 Les protocoles de routage avec localisation géographique

Un routage est dit géographique lorsque les décisions de routage sont basées sur la position des nœuds [EE04]. Les prés-requis pour effectuer un routage géographique dans un réseau ad hoc sont :

- Tous les nœuds possèdent un moyen de localisation, soit un système natif comme le GPS (*Global Position System*), soit un système logiciel comme un protocole de localisation.
- Un nœud source connaît toujours la position du nœud destinataire. Pour ce faire, soit tous les nœuds connaissent les positions initiales de tous les nœuds, soit un service de localisation doit être utilisé.

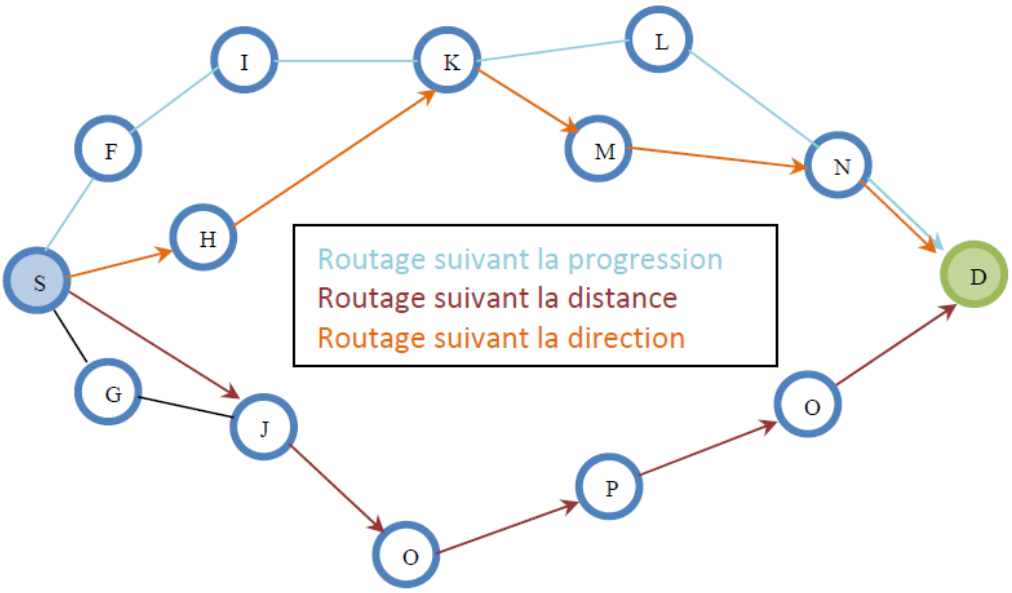



**Figure 2. 5 : Routage basé sur la localisation.**

# 4. Les protocoles de routage proposé pour les RCSF

Nous citons dans cette section quelques protocoles de routage proposé pour les réseaux de capteurs sans fil.

## 4.1. Protocoles de routage hiérarchiques

### 4.1.1. LEACH

L'idée est de former des clusters de nœuds capteurs en se basant sur la force du signal reçue et d'employer le *cluster-head* local comme routeur du *sink*. Tout traitement de données est local au cluster. Le rôle de *cluster-head* est échangé aléatoirement entre les nœuds afin d'équilibrer les charges[DE07].

### 4.1.2. PEGASIS (Power Efficient GAthering in Sensor Information Systems)

PEGASIS est une amélioration du protocole LEACH. Au lieu de former plusieurs clusters, PEGASIS forme des chaînes de nœuds de sorte que chaque nœud transmet et reçoit du nœud voisin appartenant à la chaine. Un seul nœud est choisi, parmi cette chaîne, pour transmettre au *sink*. Ce nœud est nommé ( *leader node* ). Les données recueillies se déplacent d'un nœud à un autre, et seront agrégées puis envoyées au *sink* par le noeud *leader*. Dans le PEGASIS hiérarchique les nœuds construisent une chaîne qui forme un arbre hiérarchique. Chaque nœud leader, choisi dans un niveau

particulier, transmet des données aux nœuds du niveau supérieur de la hiérarchie jusqu'a atteindre la station de base *Sink*[DE07].

#### 4.1.3. TEEN et APTEEN(Threshold sensitive Energy Efficient sensor Network protocol)

TEEN est un protocole hiérarchique conçu pour être sensible aux changements improviste des attributs détectés tels que la température. L'architecture du réseau est basée sur un groupement hiérarchique où les nœuds les plus proches forment des clusters. Après la construction des clusters, le *clusterhead* diffuse deux seuils aux nœuds. Qui sont la valeur minimale d'un attribut pour pouvoir être transmis et le degré minimale du changement de cet attribut. Le TEEN adaptatif (APTEEN) est une extension du TEEN basée sur la capture périodique des données et la réaction aux événements temps-réel. Quand la station de base forme les clusters, les cluster-heads diffusent les attributs, les seuils et le plan de transmission à tous les nœuds et effectuent également l'agrégation des données afin d'économiser l'énergie [DE07].

#### 4.1.4. SAR (Sequential Assignment Routing)

SAR est une approche multi-chemins qui s'efforce à réaliser l'efficacité énergétique et la tolérance aux fautes. SAR crée des arbres en prenant en compte les métriques QoS, la ressource énergétique sur chaque chemin et le niveau de priorité de chaque paquet. En utilisant ces arbres, des routes multiples du *sink* aux capteurs sont formés. Une ou plusieurs routes peuvent, alors, être emprunter.

### 4.2. Protocoles de routage non hiérarchiques :

#### 4.2.1. AODV (Ad-hoc On Demand Distance Vector)

C'est un protocole à vecteur de distance, comme DSDV, mais il est réactif plutôt que proactif comme DSDV. En effet, AODV ne demande une route que lorsqu' il en a besoin. AODV utilise les numéros de séquence d'une façon similaire à DSDV pour éviter les boucles de routage et pour indiquer la « nouveauté » des routes. Une entrée de la table de routage contient essentiellement l'adresse de la destination, l'adresse du nœud suivant, la distance en nombre de sauts (i.e. le nombre des noeuds nécessaires pour atteindre la destination), le numéro de séquence destination, le temps d'expiration de chaque entrée dans la table **[CS09]**.

#### 4.2.2. SPIN ( Sensor Protocols for Information via Negotiation)

L'idée derrière le SPIN est d'échanger des informations sur les données à envoyer en utilisant des paquets de signalisations spéciales nommées *meta-DATA*. Ceci permet d'éviter le problème des données redondantes. Chaque nœud, s'intéressant à la donnée référencée par ce paquet *meta-DATA*, peut les récupérer en envoyant un paquet de requête [DE07].

#### 4.2.3. DSDV (Destination Sequenced Distance Vector)

DSDV est un protocole proactif de routage à vecteur de distance. Chaque nœud du réseau maintient une table de routage contenant le saut suivant et le nombre de sauts pour toutes les destinations possibles. Des diffusions de mises à jour périodiques tendent à maintenir la table de routage complètement actualisée à tout moment [KB09].

#### 4.2.4. GSR (Global State Routing)

Le protocole GSR est un protocole similaire au protocole DSDV décrit précédemment. Ce protocole utilise les idées du routage basé sur l'état des liens (Link State, LS), et les améliore en évitant le mécanisme inefficace d'inondation des messages de routage. GSR utilise une vue globale de la topologie du réseau, comme c'est le cas dans les protocoles basés sur LS. Le protocole utilise aussi une méthode, appelée la méthode de dissémination, utilisée dans le DBF (Distributed Bellman-Ford) [KB09].

#### 4.2.5. DSR (Dynamic Source Routing)

DSR est un protocole de routage réactif qui utilise le routage de source afin d'envoyer des paquets de données. Dans ce type de routage, les entêtes des paquets de données portent la séquence des nœuds à travers lesquels le paquet doit passer. Ceci signifie que les nœuds intermédiaires ont juste besoin de garder des traces de leurs voisins intermédiaires afin de transférer les paquets de données. Le nœud source a besoin de savoir l'ordre complet des nœuds jusqu'à la destination **[CS09]**.

#### 4.2.6. OLSR (Optimized Link State Routing)

Comme son nom l'indique, OLSR est un protocole proactif à état des liens optimisé ; il permet d'obtenir aussi des routes de plus court chemin. Alors que dans un protocole à état des liens, chaque nœud déclare ses liens directs avec ses voisins à tout le réseau, dans le cas d'OLSR, les nœuds ne déclarent qu'une sous-partie de leur

voisinage grâce à la technique des relais multipoints (MultPoint Relaying, MPR) **[CS09].**

#### 4.2.7.GPSR (Greedy Perimeter Stateless Routing)

La topologie a un caractère relativement provisoire dû à la mobilité des noeuds dans les réseaux Ad-hoc et de capteurs mobiles. Pour cette raison, les protocoles de routage les plus étudiés pour ce type de réseaux sont les protocoles de routage géographique car ils permettent d'éviter la surcharge d'informations échangées entre les noeuds qui cherchent à obtenir la topologie du réseau ou à construire les tables de routage.

Ce protocole de routage géographique se base sur le fait que tous les nœuds connaissent leur position, par exemple, grâce à un équipement GPS (Global Positioning System) ou encore par un système de positionnement distribué [KB09].

## 5. Conclusion

La technologie des réseaux de capteurs reste très prometteuse, et leur défis majeur est de trouver des protocoles de routage qui permettent, à la fois, de :

- consommer le moins d'énergie possible,
- assurer la connectivité du réseau et la couverture du champ surveillé,
- assurer une livraison fiable et rapide,
- tolérer aux pannes,
- s'adapter aux changements de topologie …

Nous avons présenté dans ce chapitre les différents protocoles de routage déployés dans les réseaux de capteurs. Dans la suite, nous nous intéressons plus aux protocoles de routage qui sont basés sur les algorithmes de colonie de fourmis.

# *Chapitre III:*

# *L'Algorithme de colonies de fourmis*

# 1. Introduction

Les fourmis sont capables de résoudre collectivement des problèmes complexes, comme trouver le plus court chemin entre deux points dans un environnement accidenté. Pour cela, elles communiquent entre elles de façon locale et indirecte, grâce à une hormone volatile, appelée *phéromone* : au cours de leur progression, les fourmis déposent une trace de phéromone ; elles choisissent ensuite leur chemin de façon aléatoire, selon une probabilité dépendant de la quantité de phéromone précédemment déposée.

Ce mécanisme, qui permet aux fourmis de résoudre collectivement des problèmes complexes, est à l'origine des algorithmes à base de fourmis artificielles. Ces algorithmes ont été initialement proposés dans [DOR 92], comme une approche multi-agents pour résoudre des problèmes d'optimisation combinatoire.

L'idée est de représenter le problème à résoudre sous la forme de la recherche d'un meilleur chemin dans un graphe, puis d'utiliser des fourmis artificielles pour rechercher de bons chemins dans ce graphe.

Le comportement des fourmis artificielles est inspiré des fourmis réelles : elles déposent des traces de phéromone sur les composants du graphe et elles choisissent leurs chemins relativement aux traces de phéromone précédemment déposées ; ces traces sont évaporées au cours du temps.

Intuitivement, cette communication indirecte fournit une information sur la qualité des chemins empruntés afin d'attirer les fourmis, dans les itérations futures, vers les zones correspondantes de l'espace de recherche.

Ces caractéristiques du comportement des fourmis artificielles définissent le méta heuristique d'optimisation par une colonie de fourmis (Ant Colony Optimization (ACO) metaheuristic). Ce méta heuristique a permis de résoudre différents problèmes d'optimisation combinatoire, comme le problème du voyageur.

Nous allons tout d'abord exposer les différences et les points communs entre les fourmis virtuelles et les fourmis réelles, puis expliquer comment les fourmis virtuelles peuvent être exploitées pour résoudre un problème d'optimisation combinatoire.

# 2. Similarités et différences avec les fourmis réelles

Les fourmis virtuelles ont une double nature. D'une part, elles modélisent les comportements abstraits de fourmis réelles, et d'autre part, elles peuvent être enrichies par des capacités que ne possèdent pas les fourmis réelles, afin de les rendre plus efficaces que ces dernières.

Nous allons maintenant synthétiser ces ressemblances et différences.

## 2.1 Points communs

- **Colonie d'individus coopérants :** Comme pour les fourmis réelles, une colonie virtuelle est un ensemble d'entités non-synchronisés, qui se rassemblent ensemble pour trouver une "bonne" solution au problème considéré. Chaque groupe d'individus doit pouvoir trouver une solution même si elle est mauvaise.
- **Pistes de phéromones :** Ces entités communiquent par le mécanisme des pistes de phéromone. Cette forme de communication joue un grand rôle dans le comportement des fourmis : son rôle principal est de changer la manière dont l'environnement est perçu par les fourmis, en fonction de l'historique laissé par ces phéromones.
- **Évaporation des phéromones :** La méta-heuristique ACO comprend aussi la possibilité d'évaporation des phéromones. Ce mécanisme permet d'oublier lentement ce qui s'est passé avant. C'est ainsi qu'elle peut diriger sa recherche vers de nouvelles directions, sans être trop contrainte par ses anciennes décisions.
- **Recherche du plus petit chemin :** Les fourmis réelles et virtuelles partagent un but commun c'est la recherche du plus court chemin reliant un point de départ (le nid) à des sites de destination (la nourriture).
- **Déplacement locaux :** Les vraies fourmis ne sautent pas des cases, tout comme les fourmis virtuelles. Elles se contentent de se déplacer entre sites adjacents du terrain.
- **Choix aléatoire lors des transitions :** Lorsqu'elles sont sur un site, les fourmis réelles et virtuelles doivent décider sur quel site adjacent se déplacer. Cette prise de décision se fait au hasard et dépend de l'information locale déposée sur le site courant. Elle doit tenir compte des pistes de phéromones, mais aussi du contexte de départ.

### 2.2 Différences

Les fourmis virtuelles possèdent certaines caractéristiques que ne possèdent pas les fourmis réelles :

**1) Elles vivent dans un monde non-continu :** Leurs déplacements consistent en des transitions d'état.

**2) Mémoire (état interne) de la fourmi :** Les fourmis réelles ont une mémoire très limitée. Tandis que nos fourmis virtuelles mémorisent l'historique de leurs actions. Elles peuvent aussi retenir des données supplémentaires sur leurs performances.

**3) Nature des phéromones déposées :** Les fourmis réelles déposent une information physique sur la piste qu'elles parcourent, là où les fourmis virtuelles modifient des informations dans les variables d'états associées au problème. Ainsi, l'évaporation des phéromones est une simple décrémentation de la valeur des variables d'états à chaque itération.

**4) Qualité de la solution :** Les fourmis virtuelles déposent une quantité de phéromone proportionnelle à la qualité de la solution qu'elles ont découverte.

**5) Retard dans le dépôt de phéromone :** Les fourmis virtuelles peuvent mettre à jour les pistes des phéromones de façon non immédiate : souvent elles attendent d'avoir terminé la construction de leur solution. Ce choix dépend du problème considéré bien évidemment. Capacités supplémentaires. Les fourmis virtuelles peuvent être pourvues de capacités artificielles afin d'améliorer les performances du système. Ces possibilités sont liées au problème et peuvent être :

- l'anticipation : la fourmi étudie les états suivants pour faire son choix et non seulement l'état local.
- le retour en arrière : une fourmi peut revenir à un état déjà parcouru car la décision qu'elle avait prise à cet état a été mauvaise.

### 2.3 Expériences

#### 2.3.1 Pont binaire de Deneubourg [JD03]

L'expérience montre un nid d'une colonie de fourmis, qui est séparé d'une source de nourriture par un pont à deux voies de même longueur. On laisse évoluer les fourmis sur

le pont, on trace ainsi en fonction du temps, le graphe du nombre de fourmis empruntant chaque branche. Le résultat de l'expérience est exposé à la figure 3.1.
L'illustration (a) représente la configuration physique de l'expérience. Le graphique (b) indique l'évolution de ce système en fonction du temps : on constate que les fourmis ont tendance à emprunter le même chemin (par exemple celui du haut) après une dizaine de minutes.

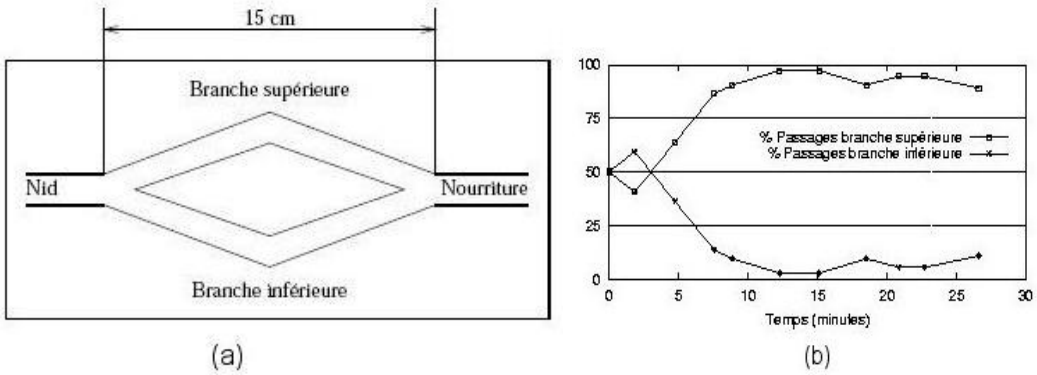



**Figure 3. 1 : Pont binaire de Deneubourg.**

Au départ, il n'y a pas de phéromone sur le pont. Donc, chaque branche peut être choisie par une fourmi avec la même probabilité. Néanmoins, dans notre exemple, après une certaine période, des variations aléatoires ont fait qu'un peu plus de fourmis ont choisi le chemin du haut plutôt que celui du bas.

Puisque les fourmis déposent des phéromones en avançant et puisqu'elles sont plus nombreuses en haut qu'en bas, le chemin du haut comportera plus de phéromones. Cette quantité supérieure de phéromone incite plus de fourmis à choisir la branche du haut, donc la quantité de phéromone déposée augmentera encore plus.

On en déduit que plus les fourmis ne suivent un chemin, plus ce chemin devient intéressant à suivre. Ainsi la probabilité avec laquelle une fourmi choisit un chemin, augmente avec le nombre de fourmis qui ont pris ce chemin précédemment.

### 2.3.2 Expérience du double pont binaire [DM07]

On peut se demander à présent quel serait l'effet de l'augmentation de la longueur d'une des deux branches du pont. L'effet produit sera que la branche la plus courte sera sélectionnée.

En effet, les premières fourmis qui reviennent au nid avec de la nourriture sont celles qui ont emprunté le chemin le plus court dans les deux sens. Ce chemin, marqué

deux fois par les phéromones, attire plus les autres fourmis que le long chemin, qui lui est marqué une seule fois dans le sens de l'aller. Cet effet se renforce au fur du temps, jusqu'à ce que toutes les fourmis choisissent le chemin le plus court.

C'est ainsi que dans cette expérience, on voit que les variations aléatoires sont réduites, puisque les deux chemins n'ont plus la même longueur. Contrairement à la première expérience, le comportement des fourmis qui consistait à suivre les pistes de phéromones n'est plus le seul mécanisme présent : maintenant on associe ce mécanisme à une notion de distance

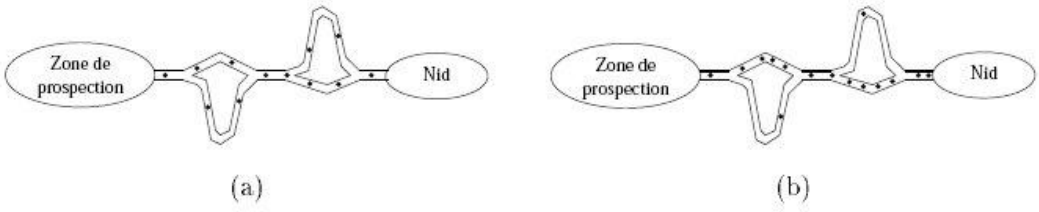



**Figure 3. 2 : Expérience du double pont binaire.**

Toutefois, quand le chemin le plus court n'est ouvert qu'après le chemin plus long (par exemple, quand le chemin était au départ bloqué par un obstacle), les fourmis continuent de parcourir le chemin le plus long car c'est le seul à être recouvert de phéromone. C'est ainsi que, l'évaporation des phéromones joue un rôle capital : les pistes de phéromones sont moins présentes sur les chemins les plus longs, car le réapprovisionnement en phéromone demande plus de temps que sur les chemins plus courts. Donc il suffit alors qu'une poignée de fourmis choisissent le chemin auparavant bloqué pour favoriser celui-ci. Ainsi, même quand des voies plus courtes ont été découvertes après, les fourmis finissent par les emprunter.

On peut généraliser cela à plus de deux chemins possibles : dans la figure 3.2 (a), on a utilisé un double pont avec quatre chemins possibles de différentes longueurs. On voit dans le dessin (b) que la plupart des fourmis finissent par choisir le chemin le plus court. Les expériences montrent que quand environ cent fourmis ont déjà emprunté le pont, plus de 90 pourcents d'entre elles sélectionnent le chemin le plus court : les fourmis convergent donc assez rapidement.

### 2.3.3 Effet de la coupure d'une piste de phéromone [Dréo 04]

Cette fois, les fourmis sont en train de suivre une piste de phéromones, comme présenté à la figure 3.3 (a). À un moment donné, on a un obstacle qui barre la route des

fourmis. Les fourmis qui arrivent à côté de l'obstacle doivent choisir soit d'aller à gauche soit d'aller à droite (b).

Puisqu'aucune phéromone n'est déposée le long de l'obstacle, il y a autant de fourmis qui partent à gauche qu'à droite. Néanmoins, puisque le chemin de droite est plus court que celui de gauche, les fourmis qui l'empruntent, vont retrouver plus vite la piste de phéromone de départ. Pour chaque fourmi allant du nid à la nourriture, on associe également une fourmi qui va de la nourriture au nid (en fait elles ont été séparées par l'apparition brutale de l'obstacle). Les phéromones de ces fourmis vont se superposer à droite. Donc quand elles vont rejoindre le chemin initial, le chemin de droite sera deux fois plus imprégnée de phéromone que la piste de gauche, où les deux fourmis n'ont pas encore pu rejoindre la piste initiale (ce chemin étant plus long).

Les fourmis qui arrivent à l'obstacle à partir de ce moment, préféreront suivre la piste de droite. Le nombre de fourmis qui passent par la droite va augmenter, ce qui augmentera encore la concentration de phéromones. De plus, l'évaporation des phéromones sera plus forte sur la piste de gauche du fait que sa longueur est supérieure. La piste de gauche sera dont rapidement abandonnée, parce qu'elle en est beaucoup moins imprégnée : les fourmis passeront toutes très rapidement par la piste la plus courte.

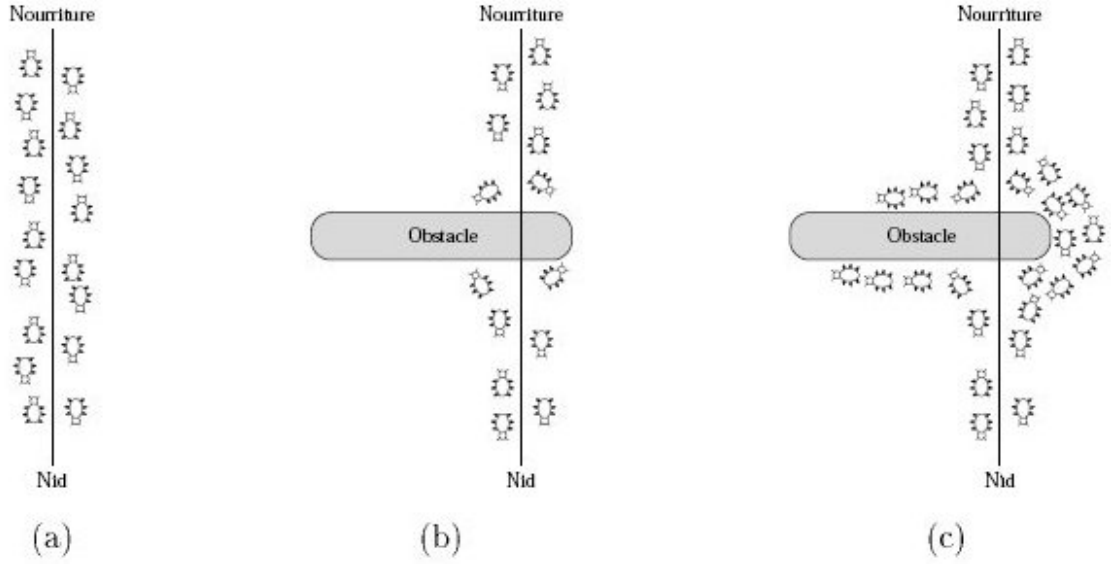



**Figure 3. 3 : Effet de la coupure d'une piste de phéromone.**

## 3. L'algorithme de colonies des fourmis

Le problème du voyageur de commerce (Travelling Salesman Problem, **TSP**) a fait l'objet de la première implémentation d'un algorithme de colonies de fourmis : Ant System (AS) [JD03].

Le problème du voyageur de commerce consiste à trouver le trajet le plus court (désigne par «tournée» ou plus loin par «tour») reliant n villes données, chaque ville ne devant être visitée qu'une seule fois. Le problème est plus généralement défini comme un graphe complètement connecte$(N, A)$, ou les villes sont les nœuds $N$ et les trajets entre ces villes, les arêtes $A$.

Dans l'algorithme AS, à chaque itération$r(1..t_{max})$, chaque fourmi $K(K = 1, \dots, m)$ parcourt le graphe et construit un trajet complet de $n = |N|$ étapes (on note$|N|$le cardinal de l'ensemble N). Pour chaque fourmi, le trajet entre une ville i et une ville j dépend de :

La liste des villes déjà visitées, qui définit les mouvements possibles à chaque pas, quand la fourmi k est sur la ville i :$J_i^k$ ;

L'inverse de la distance entre les villes : $n_{ij} = \frac{1}{d_{ij}}$appelée visibilité. Cette information «statique» est utilisée pour diriger le choix des fourmis vers des villes proches, et éviter les villes trop lointaines ;

La quantité de phéromone déposée sur l'arête reliant les deux villes, appelée l'intensité de la piste. Ce paramètre définit l'attractivité d'une partie du trajet global et change à chaque passage d'une fourmi. C'est, en quelque sorte, une mémoire globale du système, qui évolue par apprentissage.

$$P_{ij}^K(t) = \frac{(t_{ij}\,(t)^{\propto}\;(n_{ij})^{\beta}}{\sum_i^K (t_{ij}(t)^{\propto}(n_{ij})^{\beta}} \dots\dots\dots si\ j\ \in J_i^K\ (1)$$

La règle de déplacement (appelée «règle aléatoire de transition proportionnelle» par les auteurs) est la suivante :

Ou $\propto$ et $\beta$ sont deux paramètres contrôlant l'importance relative de l'intensité de la piste, $t_{ij}(t)$ et de la visibilité $n_{ij}$ .

Avec $\propto = 0$, seule la visibilité de la ville est prise en compte; la ville la plus proche est donc choisie à chaque pas. Au contraire, avec $\beta = 0$ , seules les pistes de phéromone jouent. Pour éviter une sélection trop rapide d'un trajet, un compromis entre ces deux paramètres, jouant sur les comportements de diversification et d'intensification est nécessaire.

$$\Delta t_{ij}^{K}(t) = \frac{Q}{L^{k}(t)} \dots\dots si(i,j) \in T^{k}(t) \quad (2)$$

Ou

$$\Delta t_{ij}^{K}(t) = 0 \dots\dots\dots\dots si\ (i,j) \notin T^{k}(t) \quad (3)$$

Après un tour complet, chaque fourmi laisse une certaine quantité de phéromone $\Delta t_{ij}^{K}(t)$ sur l'ensemble de son parcours, quantité qui dépend de la qualité de la solution trouvée :

Où

$T^{k}(t)$: est le trajet effectué par la fourmi $K$ à l'itération $t$

$L^{k}(t)$: la longueur de la tournée et $Q$ un paramètre fixé.

L'algorithme ne serait pas complet sans le processus d'évaporation des pistes de phéromone. En effet, pour éviter d'être piégé dans des solutions sous optimales, il est nécessaire de permettre au système «d'oublier» les mauvaises solutions. On contrebalance donc l'additivité des pistes par une décroissance constante des valeurs des arêtes à chaque itération. La règle de mise à jour des pistes est donc :

$$\tau_{ij}(t+1) = (1-p)\,\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (4)$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \quad (5)$$

Ou et *m* est le nombre de fourmis. La quantité initiale de phéromone sur les arêtes est une distribution uniforme d'une petite quantité $\tau_0 = 0$.

La figure suivant présente un exemple simplifié de problème du voyageur de commerce.

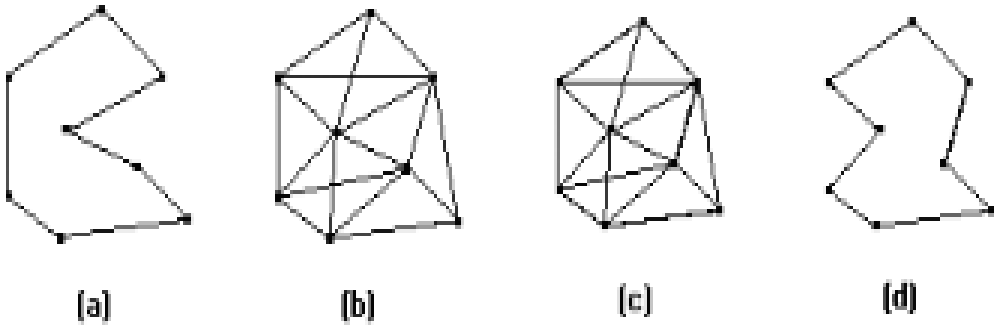



**Figure 3. 4 : Le problème du voyageur du commerce, les points représente les villes et l'épaisseur des arêtes la quantité de phéromone déposée. (a) exemple de trajet construit par une fourmi. (b) au début du calcul, tous les chemins sont explorés. (c) le chemin le plus court est plus renforcé que les autres. (d) l'évaporation permet d'éliminer les moins bonne solutions.**

## 4. Différent type des algorithmes de colonie de fourmi

Parmi les algorithmes de colonie de fourmi on a :

### 4.1. Ant System & elitisme [JD03]

Une première variante du « Système de Fourmis » a été proposée : elle est caractérisée par l'introduction de fourmis « élitistes ». Dans cette version, la meilleure fourmi (celle qui a effectué le trajet le plus court) déposé une quantité de phéromone plus grande, dans l'optique d'accroître la probabilité des autres fourmis d'explorer la solution la plus prometteuse.

### 4.2. Ant-Q [JD03]

Dans cette variante d'AS « Ant System », la règle de mise à jour locale est inspirée du « Q learning ». Cependant, aucune amélioration par rapport a l'algorithme AS n'a pu être démontrée. Cet algorithme n'est d'ailleurs, de l'aveu même des auteurs, qu'un pré version du « Ant Colony System ».

### 4.3. Ant Colony System

L'algorithme « Ant Colony System » (ACS) [JD03] a été introduit pour améliorer les performances du premier algorithme sur des problèmes de grandes tailles. ACS est fondé sur des modifications de AS .

### 4.4. ACS & 3-opt

Cette variante est une hybridation entre ACS et une recherche locale de type 3-opt.

Ici, la recherche locale est lancée pour améliorer les solutions trouvées par les fourmis et donc les ramener à l'optimum local le plus proche.

## 5. Relation avec l'informatique

En observant une colonie de fourmis à la recherche de nourriture dans les environs du nid, on s'aperçoit qu'elle résoudre des problèmes tels que celui de la recherche du plus court chemin. Les fourmis résolvent des problèmes complexes par des mécanismes assez simples à modéliser. Il est ainsi assez simple de simuler leur comportement par des algorithmes.

### 5.1. Optimisation des tables de routage

La flexibilité indéniable proposée par l'optimisation par colonie de fourmis a permis aux chercheurs d'adapter ces algorithmes a des données dynamiques. A l'heure actuelle, où les réseaux de communication et le nombre d'utilisateurs augmentent exponentiellement, il est nécessaire de gérer le réseau en temps réel pour éviter les encombrements ou les saturations des lignes. Le réseau est en grande partie lié à la notion de routage. En effet un réseau informatique peut être vu comme un graphe dont les arêtes sont les différentes lignes de communication (qui elles mêmes peuvent être vu comme des sous réseaux plus petits) et les sommets représentent les routeurs. Lorsqu'un routeur reçoit un paquet de la machine A qui doit arriver à la machine B, celui-ci a pour rôle d'envoyer le paquet vers le prochain nœud de son parcours. Les routeurs étant souvent interconnectés plusieurs routes sont possibles pour un même paquet, il doit donc essayer de choisir un trajet minimisant le délai d'acheminement en fonction de l'encombrement du réseau. On utilise principalement deux grandeurs pour exprimer la qualité d'un réseau :

- la bande passante (bit/s) qui représente le taux de transfert (soit la place sur une ligne).
- le délai moyen (s) qui est le temps d'acheminement moyen d'un paquet.

Un routage de bonne qualité permet d'augmenter la bande passante quand le réseau est chargé et de baisser le délai de transfert d'un paquet quand la charge est faible. La charge du réseau variant continuellement et de façon brutale, les algorithmes d'optimisation par colonie de fourmis semblent parfaitement indiqués pour les gestions des tables de routage. Nous détaillerons dans ce que suite le fonctionnement de l'algorithme antNet.

## 5.2. Principe de l'algorithme antNet

Les développeurs d'antNet on dû faire quelques adaptations aux algorithmes méta-heuristique déjà existants, que se soit la modélisation des fourmis ou le fonctionnement de l'algorithme.

### 5.2.1. Les fourmis du net

Les fourmis utilisées pour la gestion des tables de routage sont de deux types différents. Ainsi, il existe une fourmi "test" qui va suivre les branches du réseau comme si elle était un paquet et une fourmi "compte-rendu" qui a pour rôle de mettre à jour les tables (fourmis respectivement nommées F-ant et B-ant).

La première fourmi est un peu plus intelligente qu'un insecte lambda car elle a la possibilité de mémoriser l'état du réseau des noeuds qu'elle traverse. En pratique, elle garde en mémoire le temps de parcours et le chemin suivi. Le comportement de cette fourmi est dicté par une loi probabiliste et elle est également capable d'engendrer une fourmi B-ant à la fin de son parcours avant de mourir.

Les fourmis B-ant quant à elles ont un chemin pré-determiné puisqu'elles parcourent uniquement les noeuds dans le sens inverse de la F-ant l'ayant créée, mettant à jour les tables de routage de ceux-ci. En effet, grâce aux informations de la F-ant, la B-ant est au courant de l'encombrement du réseau permettant ainsi au routeur de s'adapter aux fluctuations.

### 5.2.2. Mécanisme de antNet

L'algorithme d'antNet est très proche de ceux utilisé pour le TSP, les routeurs pouvant être associés aux villes, ceci dit il y a quelques différences notables :

- il n'y a aucune contrainte sur les villes à visiter, c'est-à-dire qu'une fourmi n'est pas obligé de visiter tous les nœuds.
- il y a une piste de phéromone différente pour chaque nœud de destination.

- la dimension temporelle du problème rend plus délicate son évaluation.

***Algorithme :***

1. Chaque noeud lance une fourmi F-ant avec une destination aléatoire [JD03].

2. Chaque fourmi est routée grâce à la fonction stochastique f prenant en paramètre $\boldsymbol{T_{ijd}(t)}$ le taux de phéromones sur la piste et $\boldsymbol{\mu_{ij}(t)}$ l'information spécifique au problème (ici $\boldsymbol{\mu_{ij}}$ est proportionnel à la liste d'attente entre les nœuds **i** et **j**.

$\boldsymbol{P^{k}_{ijd}(t) = f(T_{ijd}(t), \mu_{ij}(t))}$ **,k :**représente fourmi **,t** :itération **.**

3. Chaque fourmi mémorise son parcours et le délai entre chaque nœud. Elle incrémente également le taux de phéromone des pistes traversées.

4. une fois arrivé à destination la F-ant génère une B-ant, celle-ci hérite des informations de la première et emprunte le même chemin en sens inverse.

5. les F-ant sont effacées... On met à jour les $\boldsymbol{T_{ijd}(t)}$ pour simuler l'évaporation.

6. la B-ant met à jour les tables de routage.

Bien entendu pour être efficace ce processus doit être lancé périodiquement de façon à pouvoir suivre les fluctuations temporelles de l'encombrement du réseau.

## 6. Conclusion

Dans ce chapitre, nous avons parlé de l'algorithme de colonies fourmis pour, ce qui nous permet de passer au dernier chapitre qui est le cœur de notre PFE, l'étude et l'implémentation de l'algorithme de colonie de Fourmi **ACAWSN**[SJ12].

# *Chapitre IV:*

# *Etude et Implémentation*

# 1. Introduction

Avant sa mise en place, le déploiement d'un réseau de capteurs nécessite une phase de simulation, afin de s'assurer du bon fonctionnement de tous les dispositifs. En effet, pour de grands réseaux le nombre de capteurs peut atteindre plusieurs milliers et donc un coût financier relativement important. Il faut donc réduire au maximum les erreurs de conception possibles en procédant à une phase de validation.

Parmi les domaines d'utilisation des systèmes embarqués on retrouve les réseaux de capteurs sans fil, ces derniers disposent des caractéristiques, comme la densité importante des nœuds, leurs autonomies énergétiques limitées et la topologie qu'ils forment, exigent des protocoles de routage spécifiques, différents de ceux déployés dans les réseaux usuels. De ce fait, le développement de nouveaux protocoles de routage s'avère indispensable. Ces protocoles doivent tenir compte de l'aspect fonctionnel de ces réseaux tout en optimisant les calculs nécessaires pour choisir la route la plus optimale.

Dans ce chapitre, nous allons en premier lieu, présenter la plate-forme logicielle que nous avons utilisé pour l'implémentation et la simulation, ensuite, nous allons présenter les contextes de simulation et les résultats pour le protocole de routage étudié avec discussion de résultat.

# 2. Environnement de travail

Dans cette section, nous présentons les outils utilisés pour la mise en œuvre du protocole « Ant Colony Algorithm in Wireless Sensor Networks » : ACAWSN [SJ12]. Nous commençons tout d'abord par TinyOS, le système d'exploitation conçu pour les dispositifs à ressources limitées en particulier les RCSF. Nous décrivons ensuite le langage de programmation NesC avec lequel nous avons implémenté le code du protocole. Nous terminons cette section par la présentation d'un simulateur des RCSF TOSSIM.

## 2.1. TinyOS

TinyOS est un système d'exploitation open-source conçu pour des réseaux de capteurs sans-fil. Il respecte une architecture basée sur une association de composants, réduisant la taille du code nécessaire à sa mise en place afin de respecter les contraintes de mémoires qu'observent les réseaux de capteurs. Pour autant, la bibliothèque de

composants de TinyOS est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données. En s'appuyant sur un fonctionnement événementiel, TinyOS propose à l'utilisateur une gestion très précise de la consommation du capteur et permet de mieux s'adapter à la nature aléatoire de la communication.

Le fonctionnement d'un système basé sur TinyOS s'appuie sur la gestion des évènements se produisant. Ainsi, l'activation de taches, leur interruption ou encore la mise en veille du capteur s'effectue à l'apparition d'évènements, ceux-ci ayant la plus forte priorité. Ce fonctionnement évènementiel s'oppose au fonctionnement dit temporel où les actions du système sont gérées par une horloge donnée.

### 2.2. NesC

NesC est un langage de programmation orienté composants syntaxiquement proche du langage C. Il est conçu pour la réalisation des systèmes embarqués distribués, en particulier, les RCSF.

Il existe trois types de fichiers sources des applications NesC: les fichiers interfaces et les fichiers configurations et modules qui constituent les composants.

-Une configuration définit les composants et/ou les interfaces utilisés par l'application déployée sur le capteur. Elle définit aussi la description des liaisons entre eux.

- Un module constitue la brique élémentaire du code et implémente une ou plusieurs interfaces.

- Une interface définit d'une manière abstraite les interactions entre deux composants.

Elle définit un fichier décrivant les commandes et les évènements proposés par le composant qui les implémente. Une commande doit être implémentée par le fournisseur de l'interface et un évènement doit être implémenté par l'utilisateur de l'interface.

On distingue les modules et les configurations dans le but de permettre aux concepteurs d'un système de construire des applications rapidement et efficacement. Par exemple, un concepteur peut fournir uniquement une configuration qui relie un ensemble de modules qu'il ne développe pas lui même. De plus, un autre développeur peut fournir une librairie de modules qui peuvent être utilisés dans la construction d'autres applications .

### 2.3. Tossim [GF08]

Avant sa mise en place, le déploiement d'un RCSF nécessite une phase de simulation afin de s'assurer du bon fonctionnement de tous les protocoles de communication qu'il utilise. En effet, pour de grands réseaux, le nombre de capteurs peut atteindre plusieurs milliers et entraine donc un coût financier relativement important. Ainsi, il faut réduire au maximum les erreurs de la conception. Malgré cela, il reste des facteurs réels qui ne peuvent être pris en compte par la simulation, tels que les contraintes physiques (perturbations électromagnétiques, inondations, etc.) ou les aléas (détériorations dues à un animal, etc.). Pour arriver à simuler le comportement des capteurs au sein d'un RCSF, un outil très puissant a été développé et proposé pour TinyOS sous le nom de TOSSIM. Le principal but de TOSSIM est de créer une simulation très proche de ce qui se passe dans les RCSF dans le monde réel. Une économie d'effort et une préservation du matériel sont possibles grâce à cet outil**.**
Pour une compréhension moins complexe de l'activité du réseau, TOSSIM utilise avec le langage python.

#### 2.3.1. Langage python

Python est un langage de programmation objet, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl et Tcl.

Le langage Python vous permet d'interagir avec le simulateur Tossim en cours dynamique, comme un puissant débogueur.

## 3. Description du protocole étudié :ACAWSN

Le protocole de routage ACAWSN proposé par Jun et al. [SJ12] s'appuie sur les informations de localisation de chaque nœud du capteur. Le Protocole se compose de deux phases que nous allons décrire.

**Phase 1**: Le protocole suppose que chaque nœud capteur connaît son emplacement et l'emplacement de la destination à priori. Ceci peut être réalisé à l'aide de GPS technologie par exemple ou par configuration manuelle. Une fois que les nœuds de capteurs sont initialement déployés, chaque nœud capteur localement diffuse un message de *Hello* à ses voisins pour former la table de voisinage. Ce message Hello contient les coordonnées géographiques ainsi que son identificateur (ID). Cette table de voisinage est supposée être identique à la table de routage. Une fois qu'un nœud capteur reçoit le message *Hello* des voisins, il enregistre l'ID du nœud expéditeur, son emplacement et calcule la distance entre lui et l'expéditeur. Après un temps prédéfini, selon la densité du réseau de capteur, le protocole suppose que tous les nœuds de capteurs ont identifié leurs voisins. Après cette étape, le protocole déclenche la phase 2.

**Phase 2** : nous supposons que le réseau de capteurs a un nœud de destination statique et $N$ nombre de nœuds de capteurs. Parmi ces capteurs, il existe des capteurs qui génère un nombre définit $n$ de fourmis dite *ant-forward* successivement pour trouver le plus court chemin entre lui-même et le sink. Avant de quitter le nœud courant, la *ant-forward* calcule la probabilité pour choisir le nœud suivant parmi les voisins selon la formule 6.

$$P_{ij}^{K}(t) = \frac{(t_{ij}\,(t)^{\propto}\ (n_{ij})^{\beta}}{\sum_{i}^{K}(t_{ij}(t)^{\propto}(n_{ij})^{\beta}} \dots\dots\dots si\ j\ \in J_{i}^{K} \qquad (6)$$

Où $t_{ij}$ (t) représente la quantité de phéromone déposé, et $n_{ij}$ représente la visibilité.

La fourmi passe ainsi de nœud en nœud jusqu'à arrivée au sink. Chaque fourmi comporte aussi certains paramètres globaux dans son en-tête de paquet tels que la liste des nœuds visités, correspondant à des valeurs de probabilité, la distance entre chaque lien. Le même processus continue jusqu'à ce que la destination est atteinte. Après avoir atteint la destination (le sink), la *ant-forward* calcule la longeur du chemin traversé entre la source et le sink et crée une fourmi dite *ant-backward*, s'autodétruit et remet le paramètres globaux à la *ant-backward*. La *ant-backward* suit le même chemin suivi par la fourmi qui la générée et dépose la phéromone sur chaque lien selon la formule 7,8 et 9. Pour le chemin plus long qui trouver par la fourmi :

$$\Delta\tau_{ij,k} = \frac{1}{L_k} \qquad (7)$$

Où $L_k$ représente le chemin le plus long entre la source et la destination.

Pour un chemin court trouvé par la fourmi :

$$\Delta\tau_{ij,k} = \frac{1}{L_{min}} \quad (8)$$

Pour les chemins n'est pas trouve par la fourmi :

$$\Delta\tau_{ij,k} = \frac{1}{1.5*L_k} \quad (9)$$

La table de routage de chaque nœud visitée se trouve alors modifiée, de telle sorte à ce que la probabilité des nœuds visités sera augmentée à l'encontre des nœuds non visités. Elle passe ainsi de nœud en nœud jusqu'à arrivé au nœud source (le nœud qui a généré la *ant-forward*). Une fois arrivée, la *ant-backward* s'auto-détruit.

Le même processus d'exploration du plus court chemin se poursuit avec les *n-1* foumis. Le chemin le plus court entre chaque nœud source et le sink est calculé une fois la nième *ant-bacward* a atteint la source.

Une étape d'évaporation de la phéromone est déclenchée après un certain délai de déroulement de l'algorithme selon la formule 10.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\, \tau_{ij,k}\,(t) \qquad (10)$$

Le chemin non visité aura ainsi une valeur de phéromone réduite.

Les détails du protocole sont illustrés dans l'algorithme 4.1.

Algorithme 4.1 Algorithme de colonies de fourmis pour RCSF:' ACAWSN [SJ12] '

```
N :nombre des nœuds
F :nombre des fourmis
V(x) :ensembles des voisins
Périodiquement
Pour chaque nœud  j=1 ... N
Envoyer un message 'hello' à les voisin
Fin pour
Pour  k = 1....F
Les nœuds Nk créer un agent  fourmi
Pour chaque nœud non visité dans N
La fourmi choisit le nœud suivant appartenant à v(x) selon la formule 6 jusqu'à arrivée au
sink
La fourmi bascule vers la phase retour suivant le chemin inverse de sa phase allé
Dépose un phéromone  sur les arcs selon la formule 7, 8 et 9
Fin Pour
Evaporer les pistes selon la formule 10
```

Fin Pour

# 4. Implémentation et déroulement

Le but principal de notre projet est d'implémenter un protocole de routage basé sur l'algorithme ACAWSN [SJ12].

## 4.1. Les fichiers de l'application

Notre application nommée ACAWSN est formée des composants suivants :

- un module, appelé « AcaC.nc ».
- une configuration, appelée « AppAcaC.nc ».
- le fichier d'entête, appelé « antMsg.h ».
- le fichier de configuration «simuletopologie.py »

Le fichier de configuration portant le nom de l'application elle-même avec une extension particulière (dans notre cas **AcaM.nc)** permet au compilateur NesC de générer un fichier exécutable. **ant.nc** est utilisé pour connecter le module aux autres composants que l'application ACAWSN.

## 4.2. Implémentation du protocole ACAWSN

Dans cette section, nous décrivons les structures de données ainsi que les principales commandes et événements nécessaires pour l'implémentation du protocole ACAWSN.

### 4.2.1. Structures de données

Le paquet dans TinyOS est envoyé dans une structure appelée Message _t, qui est contenue dans un champ « int8_t data[TOSH_DATA_LENGTH] ». Les structures de données du paquet diffèrent selon le rang du nœud (helloMsg, antMsg et antElit).

```
typedef struct antBack
{uint16_t idparent; //l'identificateur du capteur qui a généré la fourmi
Uint16_t linode[num_nod]; //la liste des nœuds visités
float quantiphiro; //la quantité du phéromone déposé
}antElit;
```

```
typedef struct helloMsg
{uint16_t ID;        //l'identificateur de capteur
uint8_t xnode;           //les cordonnées
uint8_t ynode;
}helloMsg;
```

```
Typedef struct antFor
{uint16_t idant    //l'identificateur de capteur qui A généré la ant
uint16_t linode[num_node]; //la liste des nœuds visités
uint16_t  count; //le temporisateur
uint16_t  longueur
}antMsg;
```

# 5. Métrique et simulation

## 5.1. Simulation et évaluation

Le but général dans cette section est d'implémenté et d'analyser ACAWSN **[SJ12].** Un algorithme de routage sur les réseaux de capteurs sans fil en particulier: dans chaque algorithme de routage, le but initial est de trouver un chemin entre un nœud source et un nœud destination. Plus ce chemin est court plus le protocole de routage est performant.

Dans notre étude l'utilisation des méta-heuristiques telles que les colonies de fourmis préconise la recherche d'un chemin optimal. Dans ce contexte, l'implémentation du protocole ACAWSN vise à évaluer sa performance en termes d'optimalité du chemin.

Pour évaluer cet algorithme une métrique d'évaluation a été retenue à savoir la longueur de chemin de routage à trouver. Cette métrique est évaluée en variant le nombre de fourmi dans le réseau.

## 5.2. Modèle du réseau

Dans notre étude, nous considérons un réseau de capteurs à architecture plate constituée de *N* nœuds capteurs statiques et une station de base (SB) statique. Chaque nœud capteur a un identificateur $Id_i$ unique dans le réseau, où $1 \leq i \leq N$. Lorsqu'un nœud capteur envoie un message 'Hello', ce message peut être écouté et reçu Simultanément par tous les autres capteurs voisins. Nous supposons que tous les capteurs sont physiquement identiques (par exemple des Mica2), alors que la station de base est supposés être robustes et dotée de ressources inépuisables. Nous supposons également, qu'il existe un canal de communication fiable que les nœuds capteurs peuvent utiliser pour la communication, Avant de lancer les simulations, nous devons ajuster certains paramètres qui constitueront le contexte de notre simulation. Ces paramètres sont présentés par le tableau suivant :

| Paramètre | valeur |
|---|---|
| **Nombre des nœuds** | 20 |
| $\boldsymbol{\alpha}$ **et** $\boldsymbol{\beta}$ | 1.5 |
| **Phéromone initiale** | 0.5 |
| $\boldsymbol{\rho}$ | 0.5 |

**Tableau 4. 1 : paramètres de simulation.**

## 5.3. Métrique d'évaluation

Dans notre simulation, nous nous intéresserons essentiellement à la métrique « longueur de chemin » à trouver par l'algorithme puisqu'elle constitue le paramètre le plus *critique* dans la détermination de l'importance d'un réseau de capteur. Nous considérons également la métrique de l'overhead, i.e le nombre d'octets générés par le protocole.

La longueur du chemin représente la longueur de chemin à trouver de l'algorithme lors du calcul des distances entre les nœuds visités par la fourmi jusqu'à la station de base. Nous supposons que nous connaissons également les vraies positions des nœuds. La longueur est calculée comme suit:

$$L_{SD} = \sum_{k=s}^{D} D_{ij,k} \quad (11)$$

Où $D_{i,j}$ représente la distance entre deux nœuds qui est calculé selon la formule (12).

$$D_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (12)$$

$x_i$ et $y_i$ représentent les cordonner pour un nœud.

## 5.4. Résultat de simulation

### 5.4.1. Evaluation de la longueur de chemin

Cette évaluation montre l'efficacité de la phase de la recherche de chemin de routage à trouver entre les sources et la station de base pour chaque fourmi. Le but est de connaitre le nombre de fourmi nécessaire dans le réseau pour atteindre un chemin optimal.

L'unité de la longueur du chemin est exprimé en mètre. La portée de communication de chaque nœud est laissée par défaut, à savoir 30 mètre maximum. Le tableau 4.2 illustre les résultats de notre simulation en variant le nombre de fourmis générées ainsi que le nombre d'itération de l'algorithme.

| **Nombre des itérations** | **Nombre des fourmis dans le réseau** | | | |
|---|---|---|---|---|
| | 2 | 3 | 4 | 5 |
| **2** | 151m | 129 m | 112 m | 87 m |
| **6** | 115 m | 110 m | 96 m | 56 m |

**Tableau 4.2 : la longueur du chemin de routage trouvé par les fourmis.**

Le graphe correspondant aux résultats illustrés dans le tableau 4.2 est le suivant :

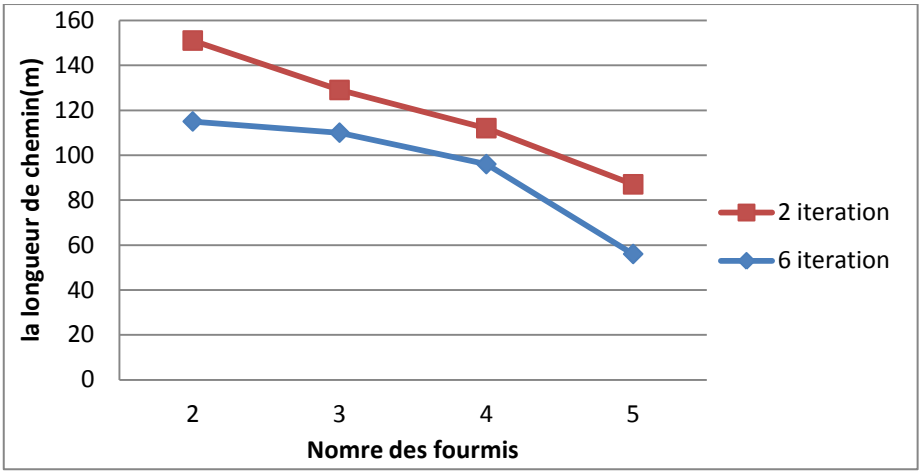



**Figure 4. 1 : la longueur de chemin trouvé par l'algorithme.**

La Figure 4.1, illustre la longueur du chemin de routage trouvé par l'algorithme pour un réseau constitué de 20 nœuds, en variant le nombre de fourmis entre 2, 3,4 et 5. Le graphe montre que plus le nombre de fourmi est grand plus le chemin est optimal. Egalement en augmentant le nombre d'itération de l'algorithme, le chemin sera encore meilleur. Ce qui nous laisse dire que plus le nombre d'itération n'est grand, plus les fourmis vont encore améliorer leur exploration et trouver encore un meilleur chemin. Cependant vu les contraintes énergétiques des réseaux de capteurs, un compromis doit être fait pour faire une balance entre performance du réseau et efficacité énergétique.

### 5.4.2. L'overhead généré

L'overhead constitue un paramètre très important pour évaluer la performance d'un protocole. Pour le calcul de l'overhead, nous considérons la topologie du réseau présenté dans figure 4.2. *S* représente le nœud source qui génère les fourmis et le nœuds *D* représente la destination.

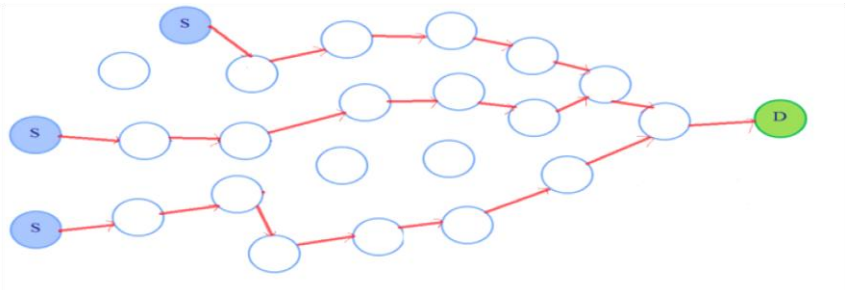



**Figure 4. 2 : la topologie du réseau.**

**Résultat pour une seule itération :** nous avons exécuté les scripts de simulation pour les différentes tailles du réseau, et nous avons aussi varié le nombre des fourmis. Les résultats dans le tableau 4.3 ci-dessous :

| Le nombre des nœuds | Le nombre des fourmis (Ant) | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **5** |
| 5 | 104 | 156 | 234 | 390 |
| 10 | 130 | 198 | 307 | 442 |
| 15 | 265 | 456 | 687 | 986 |
| 20 | 390 | 658 | 956 | 1520 |

**Tableau 4.3 : l'overhead généré (en octets pour 1 iteration ).**

**Le graphe :**

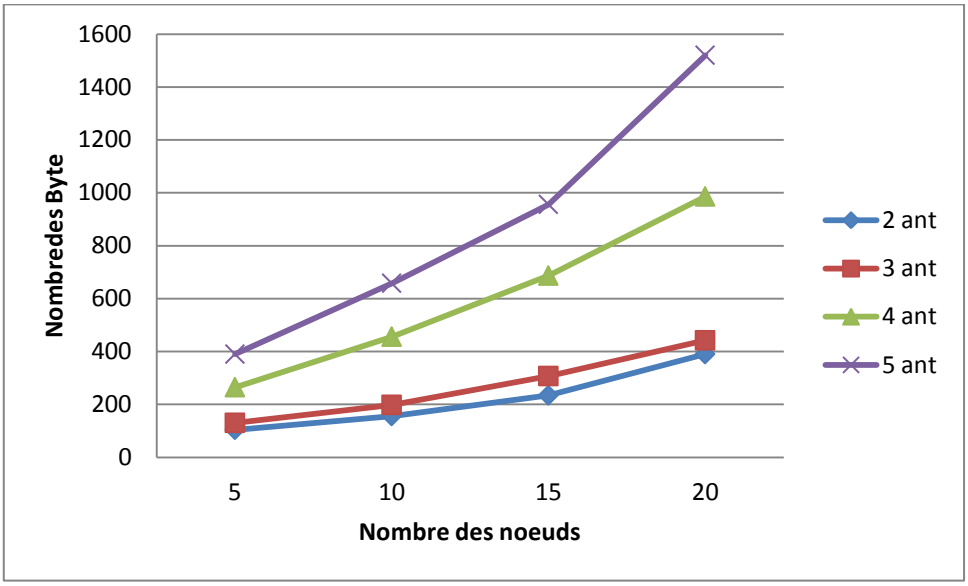



**Figure 4. 3 : l'overhead en fonction du nombre des nœuds et le nombre des fourmis (1 iteration ).**

**Observation et Discussion :** En observant les courbes de simulation dans la Figure 4.3, on remarque que l'overhead augmente avec l'augmentation de nombres de nœuds dans le réseau. En effet, l'augmentation du nombre de nœuds a pour effet d'augmenter le nombre de trafic. Mais on remarque aussi que l'overhead augmente avec l'augmentation de nombre des fourmis beaucoup plus élevé. Cette augmentation

remarquable est expliquée par le fait que le nombre des fourmis augmentent le nombre des paquets envoyés

Tout comme la simulation précédente, nous avons effectué une simulation sur plusieurs itérations.

**Résultat pour plusieurs itérations :** Les résultats de simulation dans le tableau 4.4 ci-dessous :

| Le nombre des nœuds | Le nombre des fourmis (Ant) | | | |
|---|---|---|---|---|
| | **2** | **3** | **4** | **5** |
| 5 | 208 | 312 | 468 | 780 |
| 10 | 260 | 396 | 1228 | 1768 |
| 15 | 530 | 912 | 2748 | 3944 |
| 20 | 780 | 1316 | 3824 | 6080 |

**Tableau 4.4 : l'overhead généré (en octets pour 3 iteration ). .**

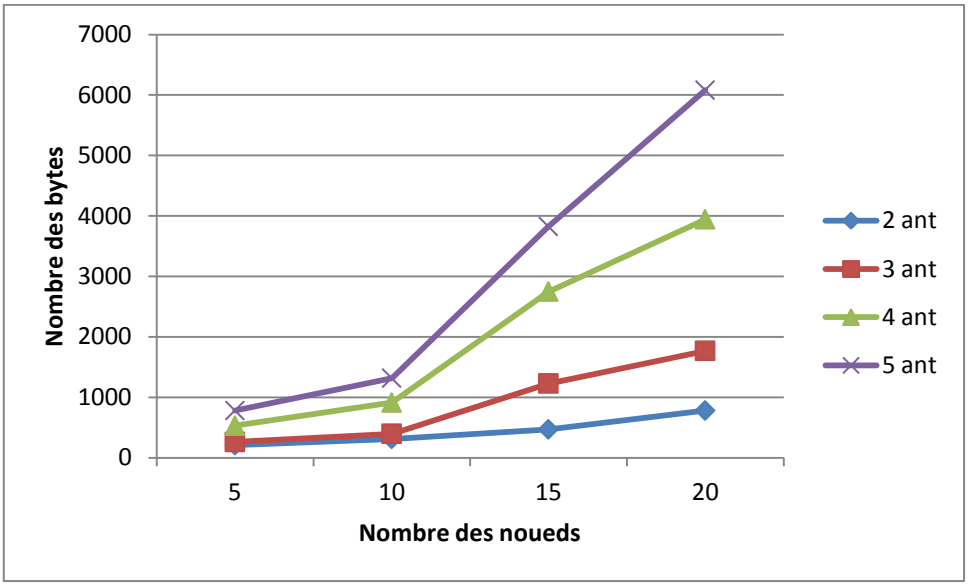



**Figure 4. 4 :l'overhead en fonction du nombre des nœuds et le nombre des fourmis (3 iteration ).**

**Observation et Discussion** : On observant le graphe de la figure 4.4, on remarque que l'overhead augmente en augmentant le nombre des nœuds et les fourmis ceci est normal puisque pour une seule itération on a trouvé un overhead très important par contre ici on à plusieurs itérations, ceci est expliqué par le nombre de messages qui se multiplie dans le protocole ACAWSN en augmentant le nombre des itérations. Où on a

plusieurs messages qui vont vers le sink et le nœud voisin ce qui introduit une charge de communication très élevée.

Néanmoins, l'overhead généré par les fourmis et le nombre d'itération doit être prix en compte, afin que le déroulement de l'algorithme ne consomme pas plus d'énergie qu'un simple flooding (broadcaste), sans ça l'utilisation des méta-heuristiques n'aura aucun intérêt.

Pour conclure, nous pouvons affirmer selon nos résultats de simulation de l'algorithme ACAWSN, que ce protocole est adapté uniquement pour un réseau à taille réduite et ne peut en aucun cas être utilisé à grande échelle à cause de l'overhead généré

## 6. Conclusion

Dans ce chapitre, nous avons présenté l'environnement de simulation avec lequel nous avons travaillé, à savoir le simulateur TOSSIM. Nous avons également définit les paramètres de simulation ainsi que les deus métriques d'évaluation prise en compte dans notre étude.

Les simulations réalisées au moyen du simulateur TOSSIM ont mené à étudier des différentes caractéristiques de l'algorithme de routage des réseaux de capteurs sans fil. Toute fois la simulation n'est pas une solution clé-en-main, puisque dans la réalité des cas non prévisibles peuvent se présenter pendant le testbed.

# Conclusion générale

Quel que soit le domaine d'étude considéré, jamais une avancée technologique ou scientifique ne s'est faite sans difficulté. Les réseaux de capteurs ne dérogent pas à cette règle et un certain nombre de problématiques doit être résolu avant de voir apparaître un monde « tous capteurs ».

Les réseaux de capteurs sans fil sont en plein développement, et deviennent de plus en plus répandus. Actuellement, ils constituent un thème de recherche très dynamique, tiré vers le haut, par leurs utilisations dans divers domaines. En effet, leurs applications sont de plus en plus nombreuses et diversifiées. Une problématique majeure dans les réseaux de capteurs, est le routage. Alors plusieurs recherches ont été faites pour la conception de l'algorithme qui tient compte de ce problème.

Contrairement au routage traditionnel, nous nous sommes intéressés un type de routage bio-inspiré, i.e inspiré de la biologie et plus particulièrement aux colonies de fourmis. Les colonies de fourmis sont des algorithmes inspirés du comportement des fourmis et qui constituent une famille des métas heuristiques d'optimisation. Une méta heuristique est un algorithme d'optimisation visant à résoudre des problèmes d'optimisation difficile pour lesquels on ne connaît pas de méthode classique plus efficace.

En effet, c'est dans le cadre de ce thème que s'oriente l'objectif de notre projet de fin d'études. Au cours de ce projet «étude un protocole de routage dans les réseaux de capteurs basé sur les algorithmes de colonies des fourmis » il nous a été demandé d'implémenter un algorithme de routage pour les réseaux de capteurs sans fil. nous avons présenté d'abord un état de l'art sur les réseaux de capteurs et leurs domaines d'applications et les différents protocoles de routage dans les RCSFs , Nous avons aussi présenté l'historique et algorithme de colonie de fourmi, Ensuite, nous avons implémenté et étudier l'algorithme ACAWSN amélioré[SJ12].

Deux métriques ont été prises en compte à savoir sont la longueur de chemin plus court à trouve par protocole ACAWSN et l'overhead.

Toutefois, tout au long de notre projet, nous avons constaté que l'implémentation et la simulation d'un réseau de capteurs sans-fil pose de grands défis auxquels il faut répondre. Parmi ces défis, la métrise du système d'exploitation TinyOs et du langage NesC à cause de la documentation très succincte, et du simulateur Tossim.

D'un point de vue personnel ce projet m'a permis de découvrir la programmation sur des systèmes embarqués, qui est beaucoup plus limitée en termes de ressources matérielles et de calculs. Les réseaux de capteurs sont le monde de demain, avec un très large éventail d'utilisation tel que la domotique, pour commander l'allumage de chauffage en fonction de la température, Il a été pour moi un grand plaisir de s'intégrer à la recherche dans ce domaine.

Comme perspective, nous comptons améliorer l'algorithme de routage ACAWSN qui est proposé par Sun Jun et al [SJ12] en ajoutant d'autres paramètres telle que la consommation d'énergie dans les nœuds puisqu'elle constitue le paramètre le plus critique dans la détermination de la durée de vie d'un réseau de capteur, ainsi que l'utilisation de la puissance de signal RSSI pour mesurer la distance entre les voisin. Il serait aussi plus intéressant de faire un test réel (expérimentation réelle) afin de faire une validation du protocole proposé.

# *Bibliographie*

[MD07] M. Dorigo, M. Birattari, and T. Stiitzle, «Ant Colony Optimization», Tsinghua University Press, Beijing, 2007.

[NJ] N. Jamal, A. Al-Karaki, E. Kamal, «Routing Techniques in Wireless Sensor Networks: A Survey», Dept. of Electrical and Computer Engineering Iowa State University.

[ME04] M. Erwan ERMEL, « Localisation et Routage géographique dans les réseaux sans fil hétérogènes », Université Pierre et Marie CURIE, 21 Juin 2004.

[SC09]S.Clément , « Quelques contributions dans les réseaux de capteurs sans fil: Localisation et Routage », thèse pour obtenir un diplôme de DOCTORAT, l'Université d'Avignon et des Pays de Vaucluse, 2009.

[VJ06] V.Julien , « Élaboration de couches MAC et réseau pour un réseau de capteurs », projet d'ingénieur I.I.E, 29 juin 2006.

[GF08] Grégory Faye, « rapport de stage ESIES mise en œuvre d'un réseau de capteurs sans fil et développement d'applications », 2008.

[LC10] L.Cobo,al,« Ant-based routing for wireless multimedia sensor networks using multiple qos metrics»,2010.

[WW09] W.Wang,al ,«Ant colony based routing algorithm for multi-sink networks», 2009.

[RG08] R.Ghasem Aghaei,al ,« Ant colony-dased many-to-one sensory data routing in wireless sensor networks», 2008.

[JY11] J.Yan,al,«Ant colony optimization for wireless sensor networks routing»,Proceedings of the 2011 International Conference on Machine Learning and Cybernetics, Guilin, 10-13 July, 2011,School of Computer Science & Technology, Soochow University, Soochow 215006, China,2011.

[SJ12] S.Jun,al, «Improved method of ant colonies to search independent data transmission routes in WSN», 20-21 September 2012.

[AC06] A.Costanzo ,al, «Optimisation par colonies de fourmis», 19 mai 2006.

[LN12] N.Labraoui , «La sécurité dans les réseaux sans fil ad hoc», mémoire de doctorat, Spécialité : Informatique, a l'université de Tlemcen faculté des sciences, 2012.

[KB09] K.Beydoun . « Conception d'un protocole de routage hiérarchique pour les réseaux de capteurs »Thèse de doctorat, Spécialité :Informatique, l'u.f.r des sciences et techniques de l'université de Franche-Comté,2009.

[JD03] J. Dréo, al, «Méta heuristiques pour l'optimisation difficile» livre Éditions Eyrolles ,2003.

[LY07] L.Yuhua , ol, « A routing strategy based on ant algorithm for WSN»,Department of Computer Science, Central China Normal University Wuhan 430079, Research Center of the Digital Space Technology, Central China Normal University Wuhan 430079, China2007.

[GW10] G.Wenjing,al, «A comprehensive routing protocol in wireless sensor network based on ant colony algorithm»,2010.

[JD00] J. Dréo ,« Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical».

[ME10] M. El mehdi Diouri, «Réseaux de capteurs sans fil: routage et sécurité», mémoire d'ingéniora en Informatique de l'INSA de Lyon, 2009-2010.

[ED07] E.DHIB « Routage avec QoS temps réel dans les réseaux de Capteurs »Ingénieur en Télécommunications option : Ingénierie des réseaux, école supérieure de communication de Tunis,2006/2007 .

# *Annexe 1 : Abréviations*

***DSDV*** **D**estination **S**equenced **D**istance **V**ector

***SPIN*** **S**ensor **P**rotocols for **I**nformation via **N**egotiation

***SAR*** **S**equential **A**ssignment **R**outing

***PEGASIS*** **P**ower Efficient **G**athering in **S**ensor **I**nformation **S**ystems

***AODV*** **A**d hoc **O**n **D**emand **V**ector Routing

***BSD*** **B**erkeley **S**oftware **D**istribution licence

***CH*** **C**luster **H**ead

***ID*** **ID**entifiant

***LED*** **L**ight **E**mitting **D**iode complement

***NesC*** **N**etwork **E**mbedded **S**ystem **C**

***OLSR*** **O**ptimized **L**ink **S**tate **R**outing Protocol

***OS*** **O**perating **S**ystem

***RAM*** **R**andom **A**ccess **M**emory

***RCSF*** **R**éseaux de **C**apteurs **S**ans **F**il

***SE*** **S**ystème d'**E**xploitations

***TinyOS*** **Tiny** micro threading **O**perating **S**ystem

***Tossim*** **T**inyos **o**perating **s**ystem **sim**ulator

***WSN*** **W**ireless **S**ensor **N**etwork

**ACA** **A**nt **C**olonny **A**lgorithme

**TSP** **T**ravelling **S**alesman **P**roblem

**RSSI** **R**eceived **S**ignal **S**trength **I**ndication

# *Annexe 1 : Mots clés en NesC*

| | |
|---|---|
| interface | A collection of event and command definitions |
| module | A basic component implemented in nesC |
| configuration | A component made from wiring other components |

| | |
|---|---|
| implementation | Contains code & variables for module or configuration |
| components | List of components wired into a configuration |

| | |
|---|---|
| provides | Defines interfaces provided by a component |
| uses | Defines interfaces used by a module or configuration |
| as | Alias an interface to another name |

| | |
|---|---|
| command | Direct function call exposed by an interface |
| event | Callback message exposed by an interface |

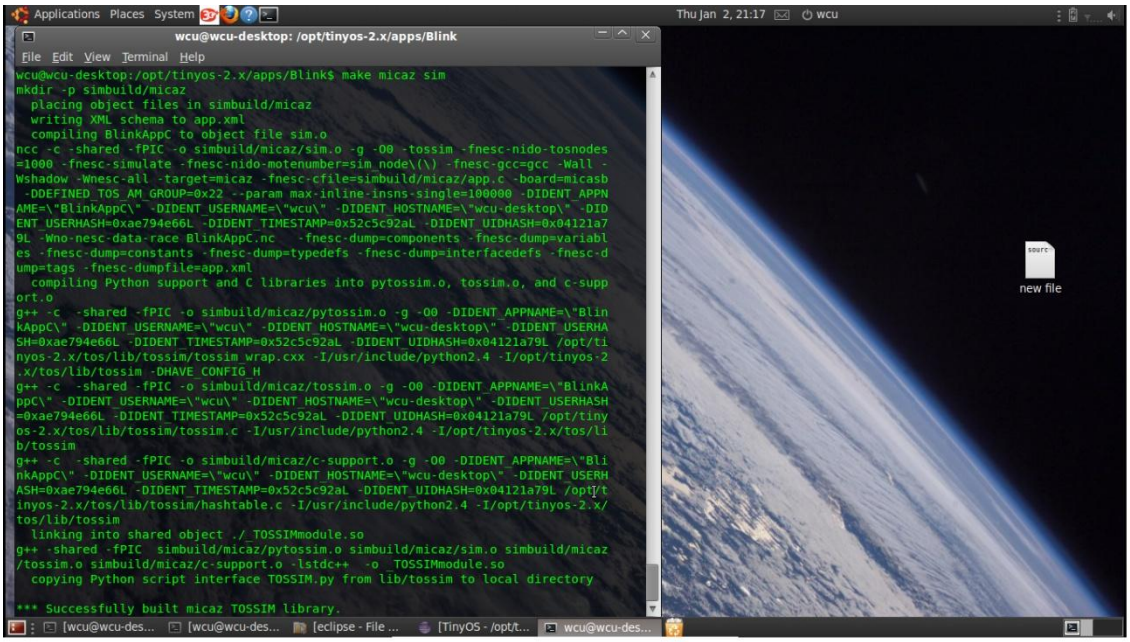
| | |
|---|---|
| call | Execute a command |
| signal | Execute an event |
| post | Put a task on the execution queue |
| task | A function to be executed in the background |

| | |
|---|---|
| includes | Include a header file |

| | |
|---|---|
| async | For commands, events executed asynchronously |
| atomic | Guarantees atomic execution of a statement |
| norace | Eliminates warnings of race conditions nesC detected |

# *Annexe 2 : Environnement d'exécution du simulateur*

Ubuntu-TOS est un système d'exploitation GNU Linux pour PC/MAC. Il est base sur la célèbre distribution Ubuntu, elle même basée sur Debian. L'intérêt de ce système est qu'il possède tous les outils nécessaires au développement d'applications basées sur TinyOS. En effet, une installation de la plate-forme de développement de TinyOS est longue et fastidieuse. Ubuntu-TOS permet donc de gagner du temps et de démarrer ce projet sur des bases saines. De plus, il est déjà entièrement paramètre pour dialoguer avec les capteurs et des alias sont déjà en place pour basculer facilement entre les différentes versions de TinyOS. Enfin, petit bonus, comme il est totalement dédie a TinyOS et au langage NesC.




C'est donc sur ce système que nous avons développe nos projets. Etant simple à installer. Cela m'a donc permis de gagner du temps, les créneaux horaires mis en place pour le projet de fin d'étude n'étant pas toujours judicieusement positionnent.

## ***Résumé***

Les réseaux de capteurs sont des réseaux formés d'un grand nombre de nœuds capteurs qui collaborent entre eux pour fournir un service bien déterminé. Cependant l'impossibilité d'une intervention humaine, a poussé les utilisateurs à s'intéresser à ces réseaux pour la surveillance et la sécurité de l'environnement ainsi la collection des données environnementales. Dans ce mémoire nous présentons une étude et une évaluation d'un protocole de routage dans les réseaux de capteurs qui est basé sur le principe des algorithmes des colonies des fourmis. En effet, chaque nœud d'un réseau a pour mission d'aider les autres dans la recherche de routes et de faire suivre les messages de ses voisins. En égard aux ressources matérielles limitées, le routage est délicat du fait qu'il faut trouver des solutions optimales pouvant allier performance et efficacité.


**Les mots clé :** réseaux capteur sans fil, algorithme colonie de la fourmi, routage.



## ***Abstract***

Sensor networks consist of a large number of sensors that collaborate to provide a service that have specified properties. But the impossibility of human intervention, prompted users to be interested in these networks for surveillance and safety of environment and collection of data too. In this memory we present a study and an evaluation of a routing protocol in the wireless sensor networks based on the principle of the ant colonies algorithm. Indeed, each node of a network is to help others in searching for paths and forward the messages of its neighbors. Given the limited hardware resources, routing is delicate because of the need to find optimal solutions that combine performance and efficiency.


**Keywords:** Wireless Sensor Network, Ant Colony Algorithm,router.



## ***ملخص***

شبكات الاستشعار والشبكات تحتوي على عدد كبير من أجهزة الاستشعار التي تتعاون على توفير خاصية الخدمة المحددة. لكن استحالة التدخل البشري، تدفع المستخدمين للاهتمام بهذه الشبكات لمراقبة وسلامة البيئة وجمع البيانات أيضا. في هذه المذكرة نقدم دراسة بروتوكول التوجيه استناداً إلى مبدأ الخوارزمية مستعمرات النمل في شبكات استشعار لاسلكية. و في الواقع، هو كل عقده شبكة مساعدة الآخرين في البحث عن الطرق.



**كلمات مفتاحية :** شبكات الاستشعار اللاسلكية ، مستعمرات النمل.