MA-004-246-1

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université SAAD DAHLAB DE BLIDA
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de Master en Informatique

Option : GENIE DES SYSTEMES INFORMATIQUE

THEME

Contrôle de congestion dans les réseaux de capteurs sans fils en utilisant la colonie de fourmis: Application au protocole AntNet

Présenté par :

DERDER Zakaria

MESRATI Mehdi

Encadreur: M. HAMANI Nacer

Promotrice: Mme. REZOUG Nachida

Examinateurs: Mme. Bentaouni
M. Kamèche

Président de Jury: Mme. Mhoud Aouidate.

Promotion : 2014/2015

Remerciements

Tout d'abord, nous remercions le bon dieu de nous avoir donné l'occasion ainsi que la volonté et le courage pour accomplir le présent travail.

Nous tenons à remercier notre encadrant monsieur HAMANI Nacer pour nous avoir proposé ce projet.

Nous tenons à exprimer toute notre profonde reconnaissance à notre promotrice: Mme REZOUG. Merci d'avoir suivi notre travail avec assiduité et de nous avoir toujours épaulés dans les moments difficiles, notamment avec votre optimisme. Merci également pour votre disponibilité, votre sympathie, vos conseils, votre soutien, enthousiasme et vos compléments pédagogiques qui nous ont permis de mener ce travail à son terme.

Nous présentons également notre plus sincère gratitude aux membres du jury qui ont bien voulu examiner et évaluer notre travail. Merci pour l'honneur que vous nous faites en participant à la soutenance.

Un grand Merci à tous les enseignants de l'université SAAD DAHLAB DE BLIDA qui nous ont formés durant nos cinq années d'étude à l'USDB.

Une grande pensée à nos familles (nos parents, nos frères et sœurs), sans lesquelles nous n'arriverons pas là et pour lesquelles nous représentons un exemple de réussite. Ce travail leur est dédié.

Nous n'oublierons jamais les bons moments passés à l'université. Tous les collègues et amis qui nous ont soutenus pendant ces années d'études. Merci à (dans un ordre aléatoire) Akli pour ses encouragements continus, Ali, Salah, Berkan, Lyes, Houssem, Karim, Nesro, Amina, Sabrina, Chahinez, Amine, Zakaria, Djalil, Bessma, Sofian, Djamel, Sidahmed... Merci à tous ceux que nous avons eu le plaisir de les rencontrer durant cette année qui nous ont aidé, soutenu, accueilli, la liste serait trop longue pour figurer ici.

Enfin, il est sûr que nous oublions certaines personnes... qu'ils nous en excusent. Cinq années nous ont permis de rencontrer beaucoup de personnes, qui ont toutes eu un rôle dans nos vies et par conséquent dans la construction de ce travail. Ils se reconnaîtront.

Résumé

Un réseau de capteurs sans fil (RCSF) est un ensemble de capteurs spatialement distribués dans des environnements parfois hostiles afin de surveiller et de contrôler certains phénomènes physiques. La communication dans ce type de réseau nécessite la mise en place de protocoles de routage qui doivent prendre en considération les caractéristiques des réseaux de capteurs sans fils telles que leur topologie dynamique ainsi que leurs limitations en ressources.

Parmi les protocoles de routage proposés dans ce domaine : le protocole AntNet, qui vise à trouver une route optimale entre un capteur source et sa station de base, en s'appuyant sur l'optimisation par colonie de fourmis. Néanmoins, le protocole AntNet reste vulnérable aux situations de congestions qui sont causées par l'importance du trafic circulant dans le réseau.

L'objectif de ce travail est de proposer une amélioration au protocole AntNet afin de pallier au problème de congestion, à travers l'utilisation de mécanismes permettant la diversification des routes et l'équilibrage des charges du réseau.

La simulation et l'évaluation des performances de l'amélioration proposée par rapport au protocole classique sont réalisées en utilisant le simulateur NS2.

Mots clés : Réseaux de capteurs sans fils, AntNet, optimisation par colonie de fourmis, congestion.

Abstract

A wireless sensor network (WSN) is a set of spatially distributed sensors in often hostile environnements, in order to monitor and control certain physical phenomena. Communication in such networks requires the implementation of routing protocols that must take into account the characteristics of WSNs such as their dynamic topology and resource limitaions.

Among the routing protocols proposed in this area, AntNet, which aims to find an optimal path between the source node and its base station or sink, based on the ant colony optimization. Nevertheless, AntNet protocol remains vulnerable to congestion situations that are caused by the volume of traffic flowing through the network.

The objective of this work is to propose an improvement to AntNet Protocol, in order to overcome the congestion problem through the implementation of mechanisms for the diversification of routes and load balancing.

Simulations and performance evaluation of the proposed improvement over the conventional protocol are made using the NS2 simulator.

Keywords : Wireless sensor networks, AntNet, ant colony optimization, congestion.

ملخص

شبكة الاستشعار اللاسلكية (WSN) هي مجموعة من أجهزة الاستشعار موزعة في بيئات معادية أحيانا لرصد و مراقبة بعض الظواهر الفيزيائية. التواصل في مثل هذه الشبكات يتطلب تنفيذ بروتوكولات التوجيه التي يجب أن تأخذ بعين الاعتبار خصائصWSNs مثل الطوبولجيا الديناميكية و محدودية الموارد.

من بين بروتوكولات التوجيه المقترحة في هذا المجال: بروتوكول AntNet، والذي يهدف لإيجاد الطريق الأمثل بين محطة المصدر وقاعدة الاستشعار، بناء على التحسين عن طريق مستعمرات النمل ACO . ومع ذلك، لا يزال البروتوكول AntNet عرضة لحالات الازدحام التي تنتج عن حجم تدفق حركة المرور عبر الشبكة.

الهدف من هذا العمل هو اقتراح حل لتحسين البروتوكول AntNet ، للتغلب على مشكلة الازدحام من خلال استخدام آليات لتنويع طرق وموازنة الأحمال في الشبكة.

يتم تنفيذ عمليات المحاكاة و تقييم النموذج الذي نقترحه في برنامج المحاكاة NS2.

كلمات مفتاحية: شبكة الاستشعار اللاسلكية، AntNet، التحسين عن طريق مستعمرات النمل ، الازدحام.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

ACK : Acknowledgment.

ACO : Ant Colony Optimization.

ACQUIRE : Active Query Forwarding in Sensor Networks.

APTEEN : Adaptive Periodic TEEN.

B-Ant : Backward Ant.

CADR : Constrained Anisotropic Diffusion Routing.

CBQ : Class-Based Queueing.

CBR : Constant Bit Rate.

CODA : Congestion Detection and Avoidance.

CSMA : Carrier Sense Multiple Access.

CSMA/CA : Carrier Sense Multiple Access with Collision Avoidance.

CSMA/CD : Carrier Sense Multiple Access/Collision Detection.

DD : Directed Diffusion.

DRR : Deficit Round Robin.

DVMRP : Distance Vector Multicast Routing Protocol.

EEABR : Energy Efficient Ant-Based Routing Algorithm.

E and D ants : Energy delay Ant based.

ESRT : Event-To-Sink Reliable Transport.

F-Ant : Forward Ant.

FIFO : First In Finrst Out.

FTP : File Transfer Protocol.

HTTP : Hypertext Transfer Protocol.

HR : High Reliability.

IP : Internet Protocol.

LEACH : Low-Energy Adaptive Clustering Hierarchy.

LR : Low Reliability.

MADFT : Minimum Ant-Based Data Fusion Tree.

MSS : Maximum Segment Size.

NC : No Congestion.

NACK : Negative-ACKnowledgment.

NS : Network Simulator.

NAM : Network AniMator.

OSI : Open Systems Interconnection.

OTcl : Object oriented extension of Tcl.

OOR : Optimal Operating Region.

PEGASIS : Power-Efficient Gathering in Sensor Information Systems.

PDU : Protocol Data Unit.

QOS : Quality of Service.

RCSF : Réseau de Capteurs Sans Fil.

RR : Route Reflector.

RTT : Temps Abller-Retour.

RED : Random Early Detection.

RTP : Real-Time Transport Protocol.

SPIN : Sensor Protocols for Information Via Negotiation.

STCP : Sensor Transmission Control Protocol.

SFQ : Short Fairness Queueing.

TEEN : Threshold Sensitive Energy Efficient Sensor Network Protocol.

TSP : Travelling Salesman Problem.

TCP : Transmission Control Protocol.

Tcl : Tool Command Language.

UDP : User Datagram Protocol.

WSN : Wireless Sensor Network.

Introduction générale

Aujourd'hui, la miniaturisation croissante des équipements électroniques ainsi que les progrès réalisés dans les technologies de communications sans fil ont permis l'apparition de nœuds capteurs. Ces petites entités autonomes sont capables de mesurer des conditions ambiantes (température, pression, luminosité, etc.) et de les transmettre à des équipements informatiques afin de les traiter.

Le déploiement aléatoire de plusieurs de ces capteurs en vue de collecter et de transmettre des informations environnementales vers une ou plusieurs stations de base appelées nœuds puits, forme un réseau de capteurs sans fil.

La réduction de la taille et du coup des nœuds capteurs ainsi que l'évolution des supports de communications sans fil ont permis d'élargir le champ d'application des réseaux de capteurs sans fil. En effet, les applications militaires de tracking, le monitoring d'habitat, ainsi que l'agriculture de précision ne sont que quelques exemples d'une panoplie vaste et variée d'applications possibles du suivi continu offert par les réseaux de capteurs sans fil.

Si leurs perspectives d'utilisation sont claires et attrayantes, les problématiques qu'engendrent ces réseaux ne sont pas moins nombreuses : puissance de calcul limitée, faible mémoire, énergie limitée, etc. Parmi les problèmes cruciaux, il y a celui du routage, qui consiste à trouver un chemin entre un nœud émetteur d'une information à la station de collecte.

Plusieurs techniques ont été proposées afin de garantir une communication fiable entre les différents nœuds du réseau de capteurs sans fil, tout en prenant en compte les contraintes d'énergie afin de prolonger la durée de vie du réseau. Nous citons parmi les techniques, l'algorithme AntNet.

Proposé par Caro et Dorigo en 1997, AntNet est un protocole de routage basé sur

l'optimisation par colonies de fourmis. Ce dernier présente l'inconvénient d'utilisation d'une seule route entre un nœud capteur et sa station de base, ce qui pourrait épuiser l'énergie des capteurs composant cette route.

Ainsi, lorsqu'un nœud capteur veut transmettre des informations à la station de base, une route unique est déterminée. Si dans le cadre d'applications nécessitant un flux de données important, des capteurs communiquent simultanément avec la station de base, l'unicité de la route peut provoquer des situations de congestions qui peuvent se traduire par l'augmentation des délais d'acheminements des messages, voir même leurs pertes, amenant à une dégradation de la qualité de service du réseau. Pour faire face à ce problème, une amélioration au protocole AntNet doit être envisagée afin de trouver des routes diversifiées vers la station de base et d'équilibrer la charge de la totalité du réseau.

Notre travail vise à apporter des améliorations au protocole AntNet afin de faire évoluer ses performances en situation de congestion, à travers l'utilisation des approches de diversification des routes et de répartition de charges.

Pour cela, ce document est organisé en cinq chapitres suivant le plan méthodologique suivant :

- Dans le premier chapitre, nous présentons une description générale des réseaux de capteurs sans fils notamment leur architecture, le routage dans ce type de réseau ainsi que leurs contraintes.
- Le deuxième chapitre comporte une étude bibliographique sur la problématique de congestion dans les réseaux de capteurs sans fil et les différentes techniques proposées dans la littérature pour y remédier.
- Le troisième chapitre présente une description de la méta-heuristique d'optimisation par colonie de fourmis et des différents algorithmes basés sur celle-ci, notamment l'algorithme AntNet.
- Le quatrième chapitre constitue le cœur de notre travail. Nous y présentons le protocole AntNet ainsi que les solutions proposées pour remédier à la problématique

dans ce protocole.

- Enfin, dans le cinquième et dernier chapitre nous exposons une étude comparative du protocole de base par rapport à celui amélioré, et ce à travers une multitude de tests et simulations dont les résultats sont accompagnés d'interprétations.

Chapitre I

Généralités sur les réseaux de capteurs sans fil

1. Introduction

Les progrès récents réalisés dans les domaines de la micro-électronique et les technologies sans fil ont mené à la production de composants de tailles réduites dotés d'interfaces radio appelés capteurs sans fil. Ces derniers sont capables de communiquer entre eux et sont déployés en nombre dans des environnement parfois hostiles, afin d'observer et de transmettre les données captées à une station de base. Ce déploiement aboutit à un réseau de capteurs sans fil (RCSF).

Ces réseaux constitués donc de plusieurs nœuds sont caractérisés par leur facilité de déploiement ainsi que leur auto-organisation, ce qui explique leur utilisation dans une vaste gamme d'applications : militaires, environnementales, industrielles, etc.

Ce chapitre est une introduction aux réseaux de capteurs sans fil. Il présente des généralités sur ce type de réseau notamment, son architecture, les facteurs principaux qui influent sur sa conception et le routage dans ces réseaux, avant de conclure par les différentes applications potentielles de ce domaine.

2. Les réseaux de capteurs sans fil

2.1 Définition d'un réseau de capteurs sans fil

Un réseau de capteurs sans fil (RCSF) ou *Wireless Sensor Network* (WSN) en anglais, est un réseau composé d'un ensemble de nœuds capteurs capables de recevoir et de transmettre des données environnementales d'une manière autonome. Ils sont dispersés plus ou moins aléatoirement dans une zone géographique appelée zone de couverture ou de captage.

Les données captées sont acheminées à travers un routage multi-saut, vers un nœud collecteur appelé puits (*sink*) ou station de base. Celui-ci les transmet par la suite à un ordinateur central ou au centre de traitement afin de les analyser et prendre

d'éventuelles décisions [1].

2.2 Architecture d'un réseau de capteurs sans fil

2.2.1 Composition

Un RCSF est composé de deux types de nœuds : les capteurs et les puits. Les capteurs sont chargés de relever et d'acheminer les informations captées vers la station de base (puits). Cette dernière quand à elle, récupère les informations remontées par les différents capteurs et les transmet au centre de traitement. Les capteurs disposés de manière aléatoire forment ce que nous appelons la zone de couverture. (Figure I.1).

2.2.2 Collecte des informations

Il y a deux méthodes pour collecter les informations d'un réseau de capteurs sans fil : à la demande ou bien suite à un évènement, comme le montre les figures I.1 et I.2 [2].

- **A la demande :** lorsque l'on souhaite avoir l'état de la zone de couverture à un moment T, le puits émet des diffusions vers toute la zone pour que les capteurs remontent leur dernier relevé vers le puits. Les informations sont alors acheminées par le biais d'une communication multi-sauts.

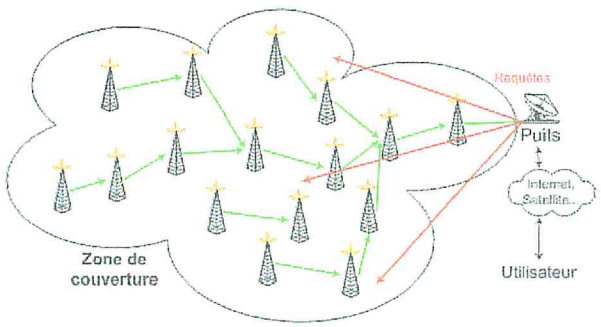

FIGURE I.1 – Collecte des informations à la demande [2].

- **Suite à un événement :** un événement se produit en un point de la zone de couverture (changement brusque de température, mouvement, etc.), les capteurs

situés à proximité remontent alors les informations relevées et les acheminent jusqu'au puits.

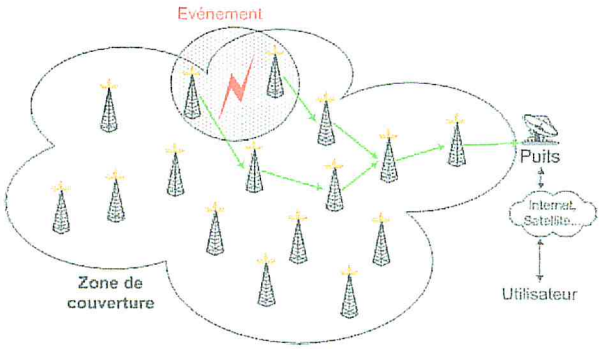

FIGURE I.2 – Collecte des informations suite à un événement[2].

3. Facteurs et contraintes de conception d'un RCSF

La conception et la mise en œuvre des RCSFs sont influencées par plusieurs contraintes qui peuvent être conceptuelles ou matérielles. Ces contraintes sont présentées dans ce qui suit :

3.1 Contraintes conceptuelles

Les principaux contraintes conceptuelles influant sur la conception et la mise en place d'un RCSF sont les suivantes :

- **Tolérance aux pannes** : dans les RCSF, un ou plusieurs capteurs peuvent ne pas fonctionner correctement. En effet, les capteurs sont des entités sensibles aux altérations d'états pouvant être provoqués par des phénomènes climatiques (humidité, chaleur, électromagnétisme), ou du fait d'une batterie faible. Dans ce cas de figure, le réseau doit être capable de détecter ce type d'erreur et d'y remédier, en cherchant par exemple à modifier ses tables de routage pour trouver un autre chemin permettant de transmettre l'information et de maintenir le réseau tou-

jours opérationnel. De la même manière, les capteurs doivent pouvoir détecter des capteurs défaillants qui envoient des informations erronées du fait de leur état [3].

– **Passage à l'échelle** : une des caractéristiques des RCSF est qu'ils peuvent contenir des centaines voir des milliers de nœuds capteurs. Suivant l'application, ce nombre peut encore augmenter jusqu'à des millions de capteurs. Les nouveaux schémas doivent pouvoir garantir un bon fonctionnement avec ce nombre élevé de capteurs. Ils doivent aussi exploiter la nature fortement dense des réseaux de capteurs [1].

- **Le coût de production** : pour que le RCSF justifie son importance par rapport aux réseaux traditionnels, il faut que le coût de production d'un seul capteur soit le minimal possible. Actuellement un capteur ne coûte pas plus de 1$ comparé à un nœud Bluetooth qui coûte environ 10$ [4].

- **Consommation énergétique :** les capteurs peuvent être équipés seulement d'une source d'énergie limitée. Dans certains scénarios d'application, il est impossible de les réapprovisionner en énergie. La durée de vie d'un capteur dépend donc fortement de la durée de vie de sa batterie et la gestion de la conservation d'énergie devient d'une haute importance. Cette énergie est consommée par les différentes unités des capteurs afin de réaliser les tâches de captage, traitement de données et communication. Cette dernière est l'opération qui consomme le plus d'énergie [1].

- **Agrégation des données :** les nœuds capteurs peuvent générer une redondance de données où des paquets similaires provenant de multiples nœuds sont générés. Ceci peut engendrer la réception d'informations redondantes au niveau de la station de base. Grâce à l'agrégation des données, des économies en énergie peuvent être obtenues puisque cette technique permet de réduire la quantité d'informations redondantes transmises par les capteurs. [5].

3.2 Contraintes matérielles

les contraintes matérielles liées à la conception et la mise en place des RCSF sont citées dans ce qui suit :

- **La dimension :** la taille réduite des capteurs peut présenter de nombreux avantages dont un déploiement flexible et simple du réseau. Cependant, la puissance des batteries utilisées pour alimenter les nœuds capteurs est limitée par la petite taille de ces derniers [1].

- **La puissance de calcul :** malgré les progrès récents dans la fabrication de capteurs de plus en plus puissants, les capteurs actuels souffrent d'un manque de puissance de calcul. Les processeurs des réseaux de capteurs sont différents de ceux d'une machine classique car ils utilisent souvent des micro contrôleurs à faibles fréquences [3].

4. Le routage dans les RCSF

Le routage est une opération importante dans les RCSF qui consiste à acheminer les données des nœuds capteurs à la station de base en minimisant la consommation d'énergie et maximisant la durée de vie du réseau. Pour cela, Les capteurs utilisent plusieurs modèles de communication détaillées ci-dessous.

4.1 Les types de communications dans les RCSF

Toute communication dans un RCSF peut être divisée en trois types [6] :

- **Un à un :** une source envoie à une destination, ce type de communication est utilisé pour transmettre des données d'un nœud à son voisin.

- **Un à plusieurs :** Aussi appelé en amont ou *upstream*, la direction du trafic dans

ce modèle est d'une source vers plusieurs destinataires. Ce modèle est utilisé par la station de base lorsqu'elle demande les données des nœuds par des requêtes ou pendant la phase de configuration du réseau.

- **Plusieurs à un :** de plusieurs source à un seul destinataire, ce modèle de communication s'illustre généralement lorsque les nœuds capteurs envoient leurs données à la station de base. Il est aussi appelé modèle en aval ou *downstream*.

4.2 Classification des protocoles de routage pour les RCSF

Les protocoles de routage pour les RCSF ont été largement étudiés récemment et différentes études on été publiées. Les protocoles proposés dans la littérature présentent des points communs et peuvent donc être classifiés suivant un certain nombre de critères. La figure ci-dessous résume une classification qui se base sur deux critères : la topologie (structure) du réseau et le mode d'établissement des chemins.

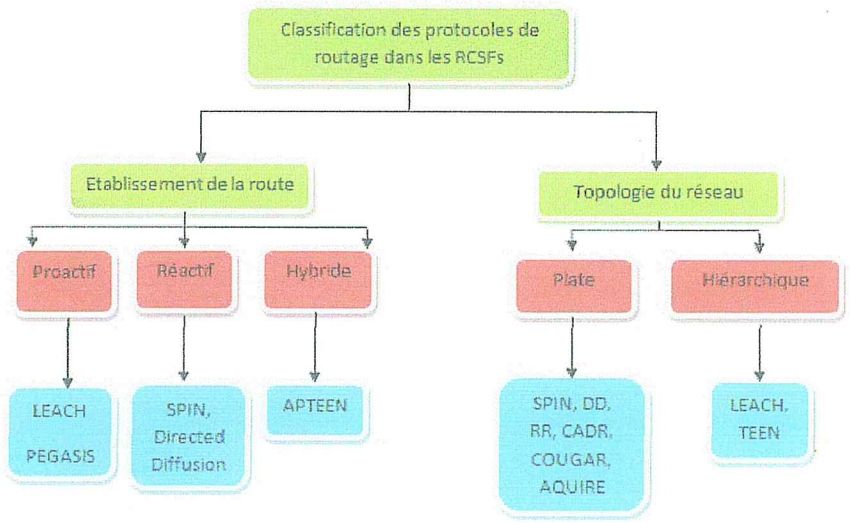

FIGURE I.3 – Classification des protocoles de routage pour les RCSF [7].

4.2.1 Selon la topologie du réseau

La topologie détermine l'organisation des capteurs dans le réseau. Globalement, il existe deux topologies dans les RCSFs : la topologie plate et la topologie hiérarchique

[7] :

- **Topologie plate :** dans une topologie plate, tous les nœuds capteurs possèdent le même rôle et collaborent entre eux pour accomplir la tâche de routage. Les réseaux plats sont caractérisés par : la simplicité des protocoles de routage, un coût de maintien réduit, une grande tolérance aux pannes ainsi qu'une habilité à construire de nouveaux chemins suite aux changements de topologie. Cependant, Les nœuds proches du puits participent plus que les autres aux tâches de routage. De plus, ces réseaux présentent une faible scalabilité dû au fonctionnement identique des nœuds et d'une manière distribuée nécessitant ainsi un grand nombre de messages de contrôle.

- **Topologie hiérarchique :** afin d'augmenter la scalabilité des réseaux, les topologies hiérarchiques on été introduites. Contrairement aux topologies plates où tous les nœuds capteurs possédaient le même rôle, les nœuds dans cette topologie, ont des différents rôles. En effet, certains nœuds sont sélectionnés pour exécuter des fonctions particulières. Une des méthodes les plus utilisées dans cette topologie est le clustering qui consiste en un partitionnement du réseau en groupes appelés clusters, chacun constitué d'un chef (clusterhead). L'inconvénient majeur de cette topologie est la surcharge des clusterheads qui induit un déséquilibre de la consommation d'énergie dans le réseau.

4.2.2 Selon le mode d'établissement des chemins

Suivant la manière de création et de maintien des routes, trois catégories de protocoles de routage peuvent être distinguées : les protocoles proactifs, les protocoles réactifs et les protocoles hybrides [8].

- **Les protocoles proactifs :** dans ce type de protocole, chaque nœud maintient une table de routage vers toute destination atteignable en associant à chaque destination un voisin direct par lequel les paquets vont être relayés. Ces tables sont maintenues même quand les routes ne sont pas utilisées. Ce type de protocole

permet la disponibilité immédiate des routes à chaque transmission. Les protocoles proactifs sont adaptés aux applications qui nécessitent un prélèvement périodique des données. Par conséquent, les capteurs peuvent se mettre en veille pendant les périodes d'inactivité, et n'enclencher leur dispositif de capture qu'à des instants particuliers. Nous pouvons citer en exemple dans cette catégorie, les protocoles LEACH et PEGASIS.

- **Les protocoles réactifs :** dits aussi les protocoles de routage à la demande, ils créent et maintiennent les routes selon les besoins. Lorsqu'un nœud à besoin d'une route, une procédure de découverte globale est lancée. La route trouvée est maintenue par une procédure de maintenance de routes jusqu'à ce que la destination soit inaccessible à partir du nœud source ou que le nœud source n'aura plus besoin de cette route. Les protocoles SPIN et DD (Directed Diffusion) sont des exemples de cette catégorie.

- **Les protocoles hybrides :** ces protocoles combinent les deux idées des protocoles proactifs et réactifs. Ils utilisent un protocole proactif pour apprendre le proche voisinage (par exemple le voisinage à deux ou à trois sauts). Ainsi, ils disposent de routes immédiatement dans le voisinage. Au-delà de la zone du voisinage, le protocole hybride fait appel à un protocole réactif pour chercher des routes. Le protocole APTEEN représente un des exemples de protocoles hybrides.

5. Domaines d'applications des RCSF

Les caractéristiques avantageuses des capteurs sans fils telles que leurs tailles de plus en plus réduite, la disponibilité de plusieurs types capteurs(optiques, thermiques ...etc) ainsi que leur coût de plus en plus faible leurs permettent d'envahir une multitude de domaines d'applications. Nous citons, entre autres, les domaines : militaire, médical, environnemental, industriel,...etc. Des exemples d'applications potentielles dans ces différents domaines sont exposés ci-dessous.

Chapitre II

La congestion dans les réseaux de capteurs sans fil

1. Introduction

Comme précisé dans le premier chapitre, la communication en amont dans les RCSF, c'est à dire celle qui s'effectue en direction du nœud sink à partir des nœuds sources d'informations, est de type *many-to-one* alors que la communication en aval est de type *one-to-many* (du sink vers les nœuds sources).

La nature convergente du trafic en amont peut engendrer parfois un flux de données important au niveau des nœuds avoisinants le sink que dans le reste des nœuds du réseau, ce qui en résulte un trafic intense dans cette région. Par conséquent, les nœuds proches du sink devront effectuer plus de tâches que les autres. Étant limités en ressources, ils peuvent se trouver face à un état où ils ne peuvent plus effectuer toutes ces tâches. Le terme employé pour désigner cette situation est la congestion [1].

Dans les RCSF, il existe plusieurs sources de congestion telles que le débordement des mémoires tampon des nœuds capteurs, les transmissions concurrentes ou encore la collision des paquets [12]. La congestion peut ainsi causer le retard de l'information, la perte de paquets et le gaspillage de la bande passante. De là, il est facile de constater que la congestion dégrade les performances du réseau en l'empêchant de garantir certaines exigences de qualité de service (Qaulity of Service) comme le temps réel, le routage de bout en bout ou la maximisation de la durée de vie du réseau. Ce problème motive le besoin de mettre en place des mécanismes de contrôle de congestion afin d'améliorer les performances et de prolonger la durée de vie du réseau [13].

L'objectif dans ce chapitre est de faire une étude bibliographique sur les protocoles ou techniques de contrôle de congestion proposées dans la littérature, et de comprendre leurs comportements afin de mieux assimiler le problème de congestion.

2. Définition et typologies de congestion

La congestion est généralement une situation où le trafic présent sur le réseau dépasse la bande passante disponible ou la capacité de traitement ou de stockage des éléments du réseau. Pour cela, deux types de congestions pourraient se produire dans un RCSF : la congestion au niveau du nœud (node-level congestion) et la congestion au niveau du lien (link-level congestion) [14].

La congestion au niveau du nœud (figure II.1) est causée par le débordement de la mémoire tampon du nœud et peut entraîner la perte et le retard des paquets. La perte des paquets à son tour, peut conduire à la retransmission et consomme donc de l'énergie supplémentaire. Dans chaque nœud capteur, le débordement du tampon pourrait se produire quand le taux d'arrivée des paquets excède leurs taux de traitement.

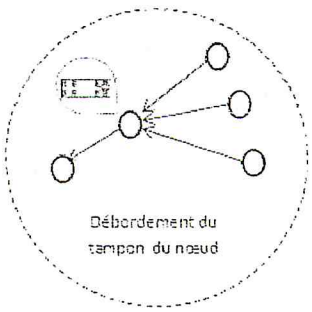

FIGURE II.1 – Congestion au niveau noeud [13].

Dans les RCSF, le canal de communication sans fil est partagé par plusieurs nœuds en utilisant des protocoles de contrôle d'accès au média de la famille CSMA (Carrier Sense Multiple Access). Les collisions pourraient donc se produire lorsque plusieurs nœuds capteurs actifs tentent de transmettre en même temps sur le canal.

Dans ce cas, on parle de congestion au niveau du lien (figure II.2). Ce type de congestion augmente le temps de traitement de paquets et diminue à la fois l'utilisation des liaisons et le débit global, en plus de gaspiller l'énergie des nœuds capteurs.

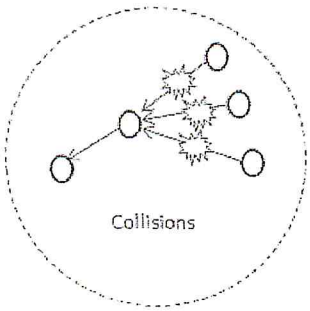

FIGURE II.2 – Congestion au niveau lien [13]

Les deux types de congestions, au niveau du nœud et au niveau du lien, ont un impact direct sur la consommation énergétique et la qualité de service.

3. Contrôle de congestion dans les RCSF

La congestion se produit lorsque le taux d'arrivée des paquets dépasse le taux de leur traitement. Ceci est plus susceptible de se produire au niveau des nœuds capteurs à proximité du puits, en raison de la nature convergente du trafic en amont.

La congestion se pose également sur la liaison sans fil en raison du bruit, des interférences, ou encore en raison d'erreurs de synchronisation de bits [15]. Le contrôle de congestion peut ainsi être classé comme suit :

3.1 Détection de la congestion

Les protocoles de détection de congestion utilisent un mécanisme pour savoir si une congestion s'est produite et à quel endroit. Une combinaison de paramètres sont utilisés par les différents protocoles afin détecter la congestion :

- **Occupation de la mémoire tampon (Buffer Occupancy) :** l'occupation de la mémoire tampon (buffer) se réfère aux emplacements utilisés de celle-ci sur la mémoire totale disponible. Un nœud capteur est constitué de tampons de taille limitée dans les RCSF en raison de la limitation de la mémoire. Lorsque la charge

du réseau augmente, les paquets sont perdus en raison du trafic excessif entrant. Dans les RCSF, le niveau de contention est contrôlé par la taille de la mémoire tampon du réseau, car si le nombre de nœuds sources est augmenté, le niveau de contention l'est également [16].

- **Taux de paquets (Packet Rate) :** le taux de paquets se réfère au nombre de paquets envoyés ou reçu dans un intervalle de temps spécifique. Si le taux de réception des paquets est supérieur à leur taux de transfert, le dépassement de tampon (buffer overflow) pourrait se produire car, comme souligné précédemment, les nœuds capteurs du RCSF disposent d'une mémoire limitée [17].

- **Packet Service Time/Packet Inter-Arrival Time :** le temps nécessaire pour traiter un paquet au niveau du nœud est connu comme *Packet Service Time*. Il fait référence en fait, à l'intervalle de temps entre l'arrivée du paquet au niveau du nœud et la transmission réussie du dernier bit de ce même paquet. *Packet Inter-Arrival Time* est l'intervalle de temps entre l'arrivée séquentielle de deux paquets d'un nœud source [18].

- **Node Delay :** correspond au délai prévu pour chaque paquet à chaque nœud. Ceci révèle l'état du trafic aux environs du nœud capteur. Les paquets sont retardés si la congestion se produit [19].

- **L'état du canal :** l'état du canal fournit l'information sur le degré d'occupation de celui-ci, ainsi que les interférences des environs, ce qui aide à détecter si le canal est prêt à transmettre et recevoir des données sans que cela résulte en une situation de congestion [19].

3.2 Notification de congestion

Après la détection de la congestion, l'information de notification de congestion doit être transmise à partir des nœuds congestionnés à leurs nœuds voisins, aux nœuds

sources ou aux nœuds destination. L'information de congestion peut être envoyée sous forme de bit de notification de congestion (*Congestion Notification Bit*) dans l'entête du paquet, ou en format complet qui comprend le degrés de congestion (*congestion degree*) ou le débit de données autorisé (allowable data rate). L'information de notification de congestion peut être catégorisée comme suit [17] :

- **Notification implicite de congestion :** l'information de congestion, dans ce type de notification, est incluse dans des paquets normaux de données. En recevant ou écoutant de tels paquets, les nœuds capteurs peuvent accéder à cette information.

- **Notification explicite de congestion :** ici, les informations de congestion sont envoyées dans un message de contrôle explicite afin de notifier les nœuds concernés.

3.3 Évitement de la congestion

Une manière directe d'éviter la congestion est d'arrêter tout simplement l'envoi de paquets dans le réseau, ou d'envoyer à un taux inférieur. Il exige également que les nœuds capteurs limitent leur flux à leurs voisins et les aident à faire face à la congestion. Les techniques utilisées afin d'éviter la congestion sont de deux types différents :

- **Ajustement du taux de transfert :** la décision d'ajuster le taux de transfert peut être faite par les nœuds congestionnés eux-mêmes, un nœud extérieur de la zone de congestion (puits), ou par une politique prédéterminée. Les mécanismes de réglage du taux de transfert sont classés comme étant soit centralisés ou distribués. Dans l'ajustement centralisé, les décisions de contrôle sont effectuées au nœud puits seulement. Dans l'ajustement distribué, les décisions de contrôle sont effectuées à chaque saut du réseau [20].

- **Redirection du trafic :** La tâche de redirection du trafic consiste à transmettre le trafic excédentaire à un réseau à grande vitesse secondaire, si disponible, qui a une longue portée de communication et qui est capable de transmettre le trafic

vers la station de base rapidement [17].

4. Efficacité énergétique dans les RCSF

Dans les RCSF, les protocoles de la couche transport doivent éviter au maximum les pertes de paquets car celles-ci se traduisent par le gaspillage d'énergie.

L'attribut clé des RCSF est défini dans la littérature par l'efficacité énergétique (Energy Efficiency). Un nœud capteur est constitué d'un ou plusieurs capteurs intégrés, de processeurs embarqués dotés d'une capacité limitée, et d'une communication radio à courte portée. Ces nœuds capteurs sont alimentés à l'aide de batteries et sont donc limités en énergie. Par conséquent, l'énergie consommée durant leur fonctionnement influence directement sur la durée de vie globale du réseau.

En cas d'un déploiement à large envergure de RCSF, il est souvent difficile ou très couteux de changer de batterie après le déploiement. La perte de paquets peut être dû à une erreur de bit (bit error) et/ou à la congestion. En cas de congestion, une quantité significative de paquets perdus a lieu en raison d'un grand manque d'espace au niveau de la mémoire tampon pour le très grand nombre de paquets. Ceci résulte en une retransmission de paquets provoquant une importante perte d'énergie ainsi qu'un retard des paquets [17].

5. Quelques protocoles de contrôle de congestion dans les RCSF

Il existe une multitude de protocoles de contrôle de congestion, certains utilisent un contrôle bout-en-bout de la congestion tandis que d'autres utilisent un contrôle saut-par-saut. Certains protocoles assurent la fiabilité de l'événement alors que d'autres offrent une fiabilité de paquets. Certains d'entre eux sont décrits dans ce qui suit.

5.1 Congestion Detection and Aviodance (CODA)

Proposée par [21], CODA est une technique de contrôle très efficace et économique en énergie conçue pour résoudre la congestion en amont. Elle comprend trois mécanismes :

1. **Détection de la congestion**

   La détection précise et efficace de la congestion joue un rôle important dans le contrôle de congestion dans les réseaux sans fil. CODA utilise une combinaison de conditions de chargement antérieur et présent du canal, ainsi que l'occupation actuelle du tampon, afin d'inférer une détection précise de la congestion à chaque récepteur, et à faible coût. Les RCSF doivent connaître l'état du canal, depuis que le support de transmission est partagé et pourrait être congestionné de trafic entre d'autres dispositifs du réseau. Cependant, l'écoute continue du canal pour mesurer la charge locale entraîne des coûts énergétiques significatifs. Par conséquent, CODA utilise un plan d'échantillonnage qui active la surveillance du canal au moment approprié pour minimiser les coûts tout en formant une estimation précise. Une fois que la congestion est détectée, les nœuds signalent leurs voisins en amont par l'intermédiaire d'un mécanisme de contre-pression (Backpressure mechanism).

2. **Open-loop, hop-by-hop backpressure**

   Dans CODA, un nœud diffuse des messages de notification de congestion explicites tant qu'il détecte l'état de congestion. Appelés aussi *Backpressure signals*, ces messages sont propagés en amont vers la source. Les nœuds les recevant peuvent répondre en supprimant des paquets ou en réduisant leur taux d'émission selon la politique de congestion locale. Lorsqu'un nœud intermédiaire (en direction de la source) reçoit le message , il décide soit de le faire propager encore plus en direction de la source ou pas, en fonction de ses propres conditions locales.

3. **Closed-loop multi-source regulation**

Ce mécanisme est capable de maintenir le contrôle de congestion sur plusieurs sources à partir d'un seul puits, dans le cas d'une congestion persistante. En effet, lorsque le taux de transmission d'une source est inférieur au débit théorique maximum, la source se régule par elle-même. Toutefois, lorsque ce débit est dépassé, la source devient plus susceptible de contribuer à la congestion et informe le puits, à travers l'ajout d'un bit dans les paquets qu'elle le lui transmet. Le puits répond par l'envoi de messages d'ACKs (*Acknowledgment*) à la source jusqu'à ce qu'il détecte la congestion. Quand celle-ci est détectée par le puits, il arrête l'envoi d'ACKs, en informant implicitement la source de diminuer son taux de transmission jusqu'à ce que la congestion disparaisse.

5.2 Event-to-Sink Reliable Transport (ESRT)

ESRT est une solution développée afin d'assurer la détection fiable d'événements dans les RCSF, tout en minimisant la dépense énergétique. Il comprend un composant de contrôle de congestion qui a pour objectif d'assurer à la fois la fiabilité et la conservation de l'énergie. Essentiellement, les algorithmes de ESRT fonctionnent au niveau du puits, avec des fonctionnalités minimales requises au niveau des nœuds capteurs [22].

L'idée clé dans ESRT est que le puits ordonne les nœuds sources d'ajuster leur fréquence d'activité selon la fiabilité mesurée au niveau du puits et de l'état de congestion dans le réseau. ESRT piste deux paramètres : l'indicateur d'intégrité, calculé par le puits, et l'état actuel de congestion [23].

Pour informer le puits de l'état actuel de congestion, chaque nœud capteur contrôle la taille de sa file d'attente. S'il constate que le prochain paquet risque de causer un débordement de sa file, un bit de notification de congestion sera ajoutée dans l'entête du paquet [23].

En se basant sur ces paramètres, l'algorithme ESRT établit un diagramme de transition à cinq états [1] :

- **No Congestion, Low Reliability (NC, LR) :** le réseau n'est pas congestionné, mais la fiabilité observée est inférieure à la fiabilité souhaitée. Dans ce cas, les sources doivent augmenter leurs taux d'activité pour augmenter la fiabilité.

- **No Congestion, High Reliability (NC, HR) :** le réseau n'est pas congestionné, mais la fiabilité observée est supérieure à la fiabilité souhaitée. Ainsi, le puits ordonne aux nœuds sources de réduire leurs taux d'activité prudemment, pour maintenir la fiabilité exigée.

- **Congestion, High Reliability (C,HR) :** le réseau est congestionné et la fiabilité est supérieure à celle souhaitée. Dans ce cas, les nœuds doivent réduire leurs taux jusqu'à ce que la congestion soit résolue ou la fiabilité descende en dessous du niveau souhaité.

- **Congestion, Low Reliability (C, LR) :** c'est le pire des états possibles, dans lequel ESRT réduit exponentiellement la fréquence d'activité pour alléger la congestion et améliorer potentiellement la fiabilité.

- **Optimal Operating Region (OOR) :** c'est la région d'exploitation optimale où le taux d'activité est juste suffisant pour atteindre la fiabilité souhaitée. Le but de ESRT est de maintenir toujours l'état du réseau dans OOR.

5.3 Sensor Transmission Control Protocol (STCP)

STCP [24] est un protocole de transport en amont, flexible, et qui fournit à la fois une fiabilité bout-en-bout ainsi qu'un mécanisme de contrôle de congestion.

En termes de fiabilité, STCP permet à l'application de spécifier le nombre de paquets, dans une taille de fenêtre fixée, devant être acheminés de façon fiable, et si les

données doivent être transférées en continu à un taux spécifique.

La station de base réceptrice utilise un mécanisme de *timeout* pour déterminer une éventuelle perte de paquets et ainsi les récupérer en envoyant un message NACK (*Negative-Acknowledgment*) au nœud capteur.

Pour les paquets évènementiels imprévisibles, le nœud source utilise un message d'accusé de réception ou ACK pour s'assurer que la station de base à reçu avec succès les paquets.

Chaque paquet réside au niveau du cache des nœuds sources à moins qu'un accusé de réception n'est reçu de la station de base. Dans le cas d'un mécanisme de contrôle de congestion, STCP utilise une simple méthode bout-en-bout et boucle fermée (*closed loop*), basée sur la surveillance de l'occupation de la file d'attente des paquets.

STCP suppose que tous les nœuds capteurs dans les RCSF ont une synchronisation d'horloge stricte avec la station de base, ce qui pourrait être une cause de problèmes de performances. En effet, les nœuds en attente d'une réponse ACK de la station de base entraîne une longue période de latence dans le cas des RCSF à très grande échelle.

6. Comparaison entre les protocoles de contrôle de congestion

En se basant sur les différents critères de contrôle de congestion cités précédemment, nous pouvons ainsi dresser un tableau comparatif entre les trois protocoles détaillés ci-dessus :

Protocole	Détection de congestion	Notification de congestion	Évitement de la congestion	Direction	End-to-end / Hop-by-hop	Conservation d'énergie
CODA	Occupation du tampon et conditions du canal	Explicite	Suppression de paquets	En amont	les deux	Oui
ESRT	Occupation du tampon	Implicite	Ajustement du taux de transfert	En amont	End to end	Oui
STCP	Occupation de la file d'attente des paquets	Implicite	Ajustement du taux de transfert et Redirection du trafic	En amont	End-to-end	Oui

TABLE II.1: Comparaison de caractéristiques techniques des protocoles CODA, ESRT et STCP

A travers ce tableau, il est remarquable qu'il existe des points communs entre ces trois protocoles. En effet, ce sont des protocoles en amont, détectent la congestion selon l'occupation de la mémoire tampon des nœuds capteurs et permettent une conservation d'énergie. Cependant, pour le reste des critères, une différence, notamment au niveau de l'évitement de congestion, du type de notification de congestion et de la manière dont la congestion est contrôlée est remarquée.

1. Introduction

Le routage des données de capteurs a été un des plus grands défis des chercheurs dans les RCSF. La plupart des recherches actuelles sur le routage sont axées sur des protocoles conscients de l'énergie afin de maximiser la durée de vie du réseau, qu'ils soient évolutifs pour un grand nombre de nœuds et tolérants aux différentes pannes [25]. De là, sont nés des protocoles basés sur l'optimisation par colonie de fourmis (*Ant Colony Optimization*) en s'inspirant de l'intelligence collective des fourmis.

Les fourmis sont capables de résoudre collectivement des problèmes complexes, en particulier celui qui vise à trouver le plus court chemin entre deux points dans un environnement difficile. Pour cela, elles communiquent entre elles localement et indirectement, à travers une substance chimique, appelée phéromone. Celle-ci est déposée par les fourmis au cours de leur progression entre leur nid et la source de nourriture.

Les algorithmes de routage basés sur les colonies de fourmis sont les protocoles les plus prometteurs concernant leurs critères qui sont très appropriés pour les RCSF [26]. En effet, les fourmis dans la nature sont de petites insectes avec des capacités et intelligence limitées, essayant de trouver et d'optimiser le chemin entre la source de nourriture et le nid, ce qui est comparable aux RCSF généralement composés de minuscules capteurs contraints par la limitation de mémoire et de batterie, déployés dans un environnement imprévisible et souvent hostile. Cette méta-heuristique a permis de résoudre différents problèmes d'optimisation combinatoire, notamment celui du voyageur de commerce.

Dans ce chapitre, nous allons aborder plus en détails sur l'optimisation par colonie de fourmis, son principe, ainsi que les algorithmes basés sur cette méta-heuristique.

2. L'optimisation par colonie de fourmis

2.1 Historique

La méta-heuristique d'optimisation par colonie de fourmis a été inspirée par les travaux de Deneubourg et ses collègues sur le comportement collectif des fourmis, en 1989. Ce concept est relativement récent, puisque il a été initialement introduit en 1991 par Colorni, Dorigo et Maniezzo, qui ont proposé un premier algorithme basé sur l'optimisation par colonie de fourmis, appelé AntSystem qui visait à résoudre le problème du voyageur de commerce.

Depuis, de nombreuses variantes du principe de base on vu le jour et elle s'est vite répandue, faisant ainsi l'objet de plusieurs améliorations dès 1995. Cependant, elle a été appliquée à d'autres problèmes d'optimisation combinatoire dès 1994.

2.2 Principe de l'ACO

Comme souligné précédemment, la méta-heuristique d'optimisation par colonie de fourmis a été conçue afin de résoudre des problèmes d'optimisation combinatoire. La source d'inspiration d'ACO a été principalement l'intelligence collective des fourmis réelles pour trouver le plus court chemin entre leur nid et la source de nourriture, en utilisant les phéromones. Elles utilisent ainsi l'environnement comme support de communication en s'échangeant localement et indirectement des informations sur l'état de leur travail à travers le dépôt de phéromones. Ce système porte le nom de stigmergie [27].

Par analogie à l'exemple biologique, ACO est basé sur la communication indirecte dans une colonie de simple agents, appelés fourmis (artificielles), à travers les pistes de phéromones (artificielles). Celles-ci servent dans ACO d'information numérique distribuée, que les fourmis utilisent pour construire de manière probabiliste des solutions au problème traité.

Un modèle expliquant le comportement collaboratif des fourmis réelles est le suivant (Figure III.1) :

- Une première fourmi parcourt au hasard l'environnement jusqu'à ce qu'elle trouve une source de nourriture (F), puis elle retourne au nid (N) en laissant sur son chemin une piste de phéromones.
- D'autres fourmis suivent l'un des chemins au hasard, en laissant aussi des traces de phéromone. Depuis que les fourmis sur le plus court chemin laissent des pistes de phéromones plus rapidement, il sera donc renforcé avec plus phéromone, le rendant plus attrayant pour les prochaines fourmis.
- Les fourmis deviennent de plus en plus susceptibles de suivre le chemin le plus court, car il est constamment renforcé par une plus grande quantité de phéromones, alors que les pistes de phéromones des chemins plus longs s'évaporent progressivement au fur et à mesure que le chemin n'est plus emprunté.

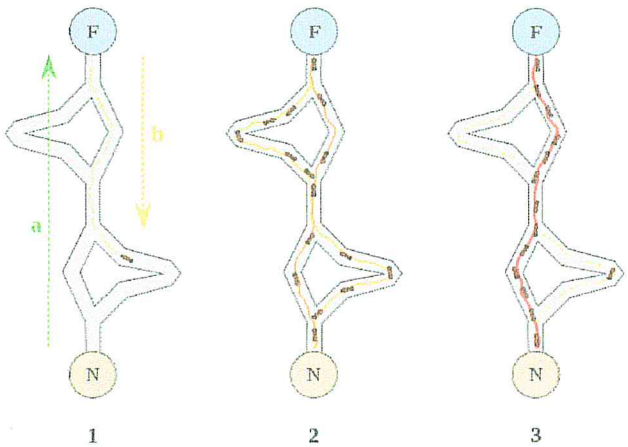

FIGURE III.1 – Choix du plus court chemin par une colonie de fourmis.

2.3 Expériences

2.3.1 Pont binaire de Deneubourg

L'expérience du point binaire de Deneubourg [28] 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. Après avoir laissé les fourmis évoluer sur le pont, Un graphe en fonction

du temps du nombre de fourmis empruntant chaque branche est obtenu. Le résultat de l'expérience est exposé à la figure III.2.

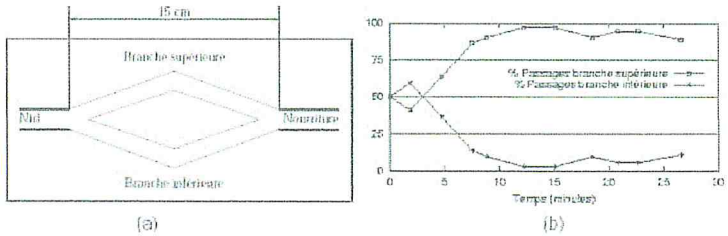

FIGURE III.2 – Expérience du pont binaire de Deneubourg [29].

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.

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 progressant et 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, et qui sera donc de plus en plus renforcé.

On en déduit que plus les fourmis suivent un chemin, plus celui-ci devient intéressant à suivre. Ainsi la probabilité avec laquelle une fourmi choisit un chemin, augmente avec le nombre de fourmis qui l'ont emprunté précédemment.

2.3.2 Expérience du double pont binaire

L'expérience du double pont binaire [30] à été réalisée pour voir 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 comme le montre deux illustrations de la figure III.3.

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 (figure III.3 (b)), 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.

FIGURE III.3 – Expérience du double pont binaire.

C'est ainsi que dans cette expérience, la constataion est 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 la notion de distance est associée à ce mécanisme.

Toutefois, si on suppose qu'au départ le chemin le plus court été bloqué par un obstacle et qu'il n y avait que le chemin long, les fourmis vont 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 nécessite 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.

2.3.3 Effet de la coupure d'une piste de phéromone

Cette fois, les fourmis sont en train de suivre une piste de phéromones, comme présenté à la figure III.4 partie (a). À un moment donné, un obstacle est utilisé pour barrer la route des fourmis. Les fourmis qui arrivent à côté de l'obstacle doivent choisir soit d'aller à gauche ou à droite (b).

Puisque 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.

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 et celle-ci sera donc rapidement abandonnée. les fourmis passeront toutes très rapidement par la piste la plus courte.

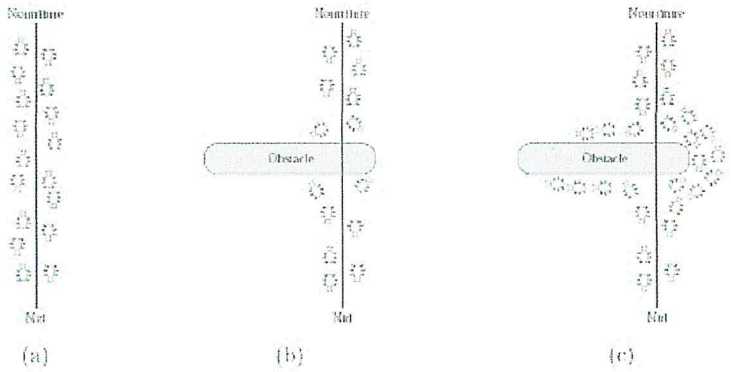

FIGURE III.4 – Effet de la coupure d'une piste de phéromone [29].

2.4 Similarités et différences avec les fourmis réelles

Les fourmis virtuelles ne permettent pas seulement de modéliser le comportement des fourmis réelles, elles peuvent être aussi dotées de capacités que les fourmis réelles

ne possèdent pas, ce qui les rend plus efficaces que ces dernières. Leurs similarités et différences sont détaillées ci-dessous [31] :

2.4.1 Points communs

- **Colonie coopérante d'individus :** à l'image des fourmis réelles, une colonie virtuelle est un ensemble d'entités non-synchronisées, qui collaborent entre eux 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 :** la communication dans une colonie de fourmis joue un rôle important dans leur comportement. Ainsi, ces entités communiquent entre eux indirectement à travers des pistes de phéromone. Cette forme de communication a pour rôle principal 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 court chemin :** parmi les similarités entre les fourmis réelles et virtuelles, il y a la recherche du plus court chemin entre un point de départ (le nid) à des sites de destination (la nourriture).

- **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.

2.4.2 Différences

Les fourmis virtuelles possèdent certaines caractéristiques que les fourmis réelles ne possèdent pas. Celles-ci sont détaillées comme suit :

- **Mémoire (état interne) de la fourmi :** les fourmis réelles ont une mémoire très limitée. Tandis que les fourmis virtuelles mémorisent en plus de l'historique de leurs actions, des données supplémentaires sur leurs performances.

- **Nature des phéromones déposées :** les fourmis réelles déposent une information physique sur les pistes 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 décrementation de la valeur des variables d'états, à chaque itération.

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

- **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.

3. Application de l'optimisation par colonie de fourmis au routage dans les RCSF

3.1 Première implémentation : AntSystem

Le problème du voyageur de commerce (Travelling Salesman Problem, TSP) est un problème largement étudié dans la littérature et pendant une longue période, a attiré un bon nombre d'efforts de recherche. Le TSP joue également un rôle important dans la recherche concernant l'ACO. Ainsi, le premier algorithme ACO, appelé Ant System [28], aussi bien que plusieurs algorithmes ACO qui ont suivis, a d'abord été testé sur le TSP.

Intuitivement, le TSP est le problème d'un vendeur qui, à partir de sa ville natale, veut trouver le plus court chemin qui l'emmène à travers un ensemble de villes de clients, et revenir à sa ville de départ, en visitant chaque ville exactement une fois. Plus formellement, le TSP peut être représenté par un graphe complet pondéré $G = (N,A)$, avec N étant l'ensemble de nœuds représentants les villes, et A étant l'ensemble d'arcs. Une valeur d_{ij} est attribuée à chaque arc (i,j)∈A qui représente la distance entre les villes i et j, avec i,j∈N.

Dans AS, m fourmis (artificielles) construisent en même temps un tour du TSP. Initialement, les fourmis sont mises sur des villes choisies au hasard. A chaque étape de la construction, une fourmi k applique une règle de choix d'action probabiliste, appelée règle aléatoire de transition proportionnelle (random propotional rule), pour décider de la prochaine ville à visiter. En particulier, la probabilité avec laquelle une fourmi k, à une ville i, choisit d'aller à la ville j est :

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}, \quad \text{if } j \in \mathcal{N}_i^k,$$

où $\eta_{ij} = \frac{1}{d_{ij}}$, appelée visibilité, et qui représente l'inverse de la distance entre les villes. Cette information est disponible à priori et est utilisée pour diriger le choix des

fourmis vers des villes proches, et éviter ainsi des villes lointaines. α et β sont deux paramètres contrôlant l'importance relative de l'intensité de la piste de phéromone τ_{ij} et de la visibilité η_{ij}. N_i^k représente les villes voisines non encore visitées par la fourmi k, se trouvant dans une ville i (la probabilité de choisir une ville en dehors de N_i^k est nulle)

Par cette règle probabiliste, la probabilité de choisir un arc particulier (i,j) augmente avec la valeur associée à la piste de phéromone τ_{ij} et à la visibilité η_{ij}. Ainsi, le rôle de α et β est le suivant : si $\alpha = 0$, seule la visibilité de la ville est prise en compte et donc les villes les plus proches sont susceptibles d'être sélectionnées. Tandis qu'avec $\beta = 0$, seules les pistes de phéromone sont prises en compte. 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.

Chaque fourmi k maintient une mémoire M^k contenant les villes déjà visitées, dans l'ordre de visite. Cette mémoire est utilisée pour déterminer le voisinage N_i^k susceptible d'être visité, dans la règle de construction donnée précédemment. En outre, la mémoire M^k permet à une fourmi k à la fois de calculer la longueur de la tournée T^k qu'elle a généré et de retracer le chemin afin de déposer de la phéromone.

Après que toutes les fourmis aient construit leurs tournées, les traces de phéromone sont mises à jour. Cela se fait d'abord par l'abaissement de la valeur de la phéromone sur tous les arcs par un facteur constant, puis en ajoutant la phéromone sur les arcs par lesquels les fourmis sont passées durant leurs tournées. L'évaporation de la phéromone est mise en œuvre par :

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \quad \forall (i,j) \in L.$$

où $0 < \rho \leq 1$ est le taux d'évaporation de phéromone. Le paramètre ρ est utilisé pour éviter l'accumulation illimitée des traces de phéromone et permet à l'algorithme d'oublier de mauvaises décisions prises précédemment. En fait, si un arc n'est pas choisi par les fourmis, sa valeur associée de phéromone diminue de façon exponentielle dans le nombre d'itérations. Après évaporation, toutes les fourmis déposent de la

phéromone sur les arcs franchis dans leurs tournées :

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \quad \forall (i,j) \in L,$$

où $\Delta\tau_{ij}^{k}$ est la quantité de phéromone déposée par une fourmi k sur l'arc qu'elle a visité. Elle est définie comme suit :

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/C^{k}, & \text{si l'arc (i,j) appartient à } T^{k} \\ 0, & \text{si non} \end{cases}$$

où C^{k}, la longueur du tour T^{k} construit par la k-ème fourmi, est calculé par la somme des arcs appartenant à T^{k}.

3.2 Principe des algorithmes de routage basés sur l'ACO

Tout protocole de routage basé sur l'optimisation par colonie de fourmis utilise deux types de fourmis [6] : les fourmis forward (F-Ant), qui vont suivre les branches du réseau, et les fourmis backward (B-Ant) qui ont pour rôle de mettre à jour les tables de routage et les quantités de phéromones. Le premier type de fourmi a la possibilité de garder dans sa mémoire l'état des nœuds qu'elle traverse, le temps de parcours et le chemin suivi ; comme elle transporte les données de la source à la destination. Elle est également capable d'engendrer une fourmis *B-Ant* à la fin de son parcours avant de mourir. la fourmi *B-Ant* a un chemin prédéterminé puisqu'elle parcourt uniquement les nœuds dans le sens inverse de la *F-Ant* l'ayant crée. Ceci sera abordé avec plus de détails dans les sections à venir.

Dans les réseaux de capteurs les fourmis sont lancées à partir du nœuds source vers la destination en parcourant un ensemble de nœuds voisins et en déposant une quantité de phéromone sur chaque arc visité. le choix du nœud suivant est fait selon une règle de décision probabiliste.

3.3 Exemples de protocoles de routage utilisant l'ACO

Il existe plusieurs protocoles de routage pour les RCSF qui utilisent l'ACO parmi lesquels nous citons :

3.3.1 Le protocole EEABR (Energy Efficient Ant-Based Routing Algorithm)

Proposé dans[32], EEABR est un protocole de communication pour les réseaux de capteurs sans fil basé sur l'ACO et qui vise à maximiser la durée de vie du réseau. Il a été prouvé que les opérations de transmission consomment beaucoup plus d'énergie que le traitement des données. Donc, il est préférable d'effectuer le plus possible de traitement sur les nœuds capteurs que de transmettre des données brutes à travers les fourmis à la station de base pour les traiter.

Dans le cas où le nombre de nœuds dans le réseau atteint 1000 unités, la mémoire de la fourmi doit être assez grande à un point où on ne peut plus transmettre sur le réseau. C'est la raison pour laquelle que dans cette approche la mémoire de la fourmi a été réduite à deux enregistrements qui concernent les deux derniers nœuds visités et sur chaque nœud, une structure de données stocke les informations qu'apporte la fourmi.

La table de routage quand à elle, stocke le nœud précédent, le nœud suivant, l'identification de la fourmi et une valeur de temporisation. Quand une fourmi forward est reçue, le nœud cherche l'identification de cette fourmi dans sa table de routage. S'il ne la trouve pas, il stocke les informations nécessaires et transmet la fourmi au nœud suivant. Si l'identification de la fourmi est trouvée, la fourmi est éliminée pour éviter le problème de boucle.

Lorsqu'un nœud reçoit une fourmi backward, il cherche dans sa table de routage le nœud suivant, où la fourmi doit être envoyée.

3.3.2 MADFT (Minimum Ant-Based Data Fusion Tree)

Proposé dans [33], MADFT est un protocole de routage qui vise à trouver des chemins optimaux entre les capteurs et la station de base en attribuant une fourmi à chaque nœud source. Si le nœud source fait partie d'un chemin qui a été découvert précédemment, la fourmi arrête la recherche, car l'itinéraire optimal est déjà trouvé. Sinon, elle essaie de trouver le plus court chemin pour atteindre la station de base ou le point d'agrégation le plus proche (un point d'agrégation est le nœud où plusieurs fourmis se rencontrent).

3.3.3 E and D ants (Energy delay Ant based)

[34] ont proposé ce protocole qui vise à trouver des chemins courts, minimiser la consommation d'énergie et les délais de transmission afin de fournir un service de livraison de données en temps réel, Pour cela, ils construisent des routes de la source à la destination à l'aides des fourmis qui explorent le réseau et choisissent les nœuds qui offrent un meilleur compromis entre le niveau d'énergie résiduelle et le temps de latence.

3.3.4 Le routage adaptatif et le routage adaptatif amélioré (AR et IAR)

Les algorithmes AR et IAR proposés dans [35] sont conçus pour optimiser plusieurs critères dans les RCSF tels que l'énergie et le temps de latence. Pour cela, avec le protocole AR, la fourmi forward choisit sa prochaine destination comme étant le nœud le plus proche d'elle. Mais, ce protocole n'a pas abouti à une solution optimale. Des améliorations ont été apportées au protocole AR où un coefficient représentant la distance entre le prochain relai de la fourmi forward et la destination a été ajouté ce qui a donné naissance au protocole IAR.

4. Le protocole AntNet

Basé sur l'optimisation par colonie de fourmis, AntNet [36] est un algorithme innovant pour le routage des paquets dans les RCSF, initialement proposé par M. Dorigo et G. Di Caro, en 1997. Dans AntNet, un groupe d'agents mobiles (ou fourmis artificielles) construisent des chemins entre deux nœuds, explorent le réseau en même temps et s'échangent les informations obtenus afin de mettre à jour les tables de routage [37].

4.1 Les types de fourmis d'AntNet

AntNet est idéalement décrit en termes de deux ensembles de fourmis artificielles introduits précédemment, les *forward ants*(F-ant) et *backword ants*(B-ant). Les fourmis, dans chaque ensemble, possèdent la même structure, mais elles sont différemment situées dans l'environnement ; autrement dit, elles peuvent sentir différentes entrées et produire différentes sorties indépendantes. Les fourmis communiquent d'une manière indirecte, selon le paradigme de stigmergie, à travers l'information qu'elles lisent et écrivent simultanément sur les nœuds du réseau qu'elles visitent.

Les fourmis F-ant construisent leurs chemins à partir du nœud source vers le nœud de destination. En faisant cela, elles recueillent des informations explicites sur le temps du parcours et gardent en mémoire le chemin suivi. Elle recueillent également des informations implicites sur l'état de charge du réseau. Le comportement de ce type fourmis est dicté par une loi probabiliste et sont également capables d'engendrer des fourmis B-ant à la fin de leurs parcours avant de mourir.

Les fourmis B-ant quant à elles, ont un chemin pré-déterminé puisqu'elles parcourent uniquement les nœuds dans le sens inverse des F-ant l'ayant créées, 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 retour, de s'adapter aux fluctuations.

4.2 L'algorithme AntNet

En Supposant un réseau de données comportant N nœuds, où s représente un nœud source générant un agent (fourmi) vers une destination d. Dans ce qui suit, une notation est attribuée à chaque élément du réseau ce qui nous permettra de décrire aisément les différentes étapes de l'algorithme AntNet par la suite [37] :

- F-ant, notée $Fs \rightarrow d$ qui va aller d'un nœud source s à un nœud destination d.
- B-ant, notée $Bd \rightarrow s$ générée par une $Fs \rightarrow d$, qui retourne au nœud source s en suivant le même chemin empreinté par la f-ant qui l'a crée, et ceux dans le but d'utiliser les informations récoltées par la f-ant afin de mettre à jour les tables de routage des nœuds visités.

Chaque fourmi transporte une pile ou *stack* en anglais, $Ss \rightarrow d(k)$ de données où l'index k se réfère au k-ème nœud visité dans une tournée, où $Ss \rightarrow d(0) = s$ et $Ss \rightarrow d(m) = d$, m étant le nombre de sauts réalisés par la $Fs \rightarrow d$ avant d'arriver au nœud d.

Soit k un nœud du réseau ; sa table de routage va contenir N entrées, une pour chaque destination possible.

Soit j une entrée de la table de routage d'un nœud k, et soit N_k l'ensemble des voisins du nœud k.

Soit P_{ij} la probabilité avec laquelle une fourmi ou un paquet de données dans k, aille à un nœud i, $i \in N_k$, j étant la destination $(j \neq k)$. Ainsi, Pour chacune des N entrées de la table du noeud k, ce sera n_k valeurs de P_{ji} telles que :

$\sum P_{ij} = 1$; j=1,...,N.

De manière informelle, l'algorithme AntNet peut être décrit comme suit :

$$P_{fd} \leftarrow P_{fd} + r(1 - P_{fd})$$

où r est un facteur de renforcement indiquant la bonté du chemin emprunté.

– Le reste des nœuds voisins j ($j \neq f$) verront leurs probabilités P_{nd} diminuées à travers l'expression :

$$P_{nd} \leftarrow P_{nd} - rP_{nd} \; n \in N_k, \; n \neq f.$$

5. Conclusion

Nous avons introduit dans ce chapitre un axe important de notre projet de recherche, et qui est l'optimisation par colonie de fourmis.

Après une première implémentation réussie à travers *AntSystem* qui a permis de résoudre le problème du voyageur de commerce, cette méta heuristique s'inspirant du comportement collectif d'une colonie de fourmis s'est vite popularisée pour donner naissance à une multitude de protocoles de routage tout en s'appuyant son idée de base.

Parmi ces protocoles, nous citons le protocole AntNet qui vise à construire un chemin optimal entre un nœud source et sa destination à travers un groupes d'agents ou fourmis mobiles. Ce protocole fera l'objet d'études et d'améliorations dans les chapitres à venir.

Chapitre IV

Conception de l'amélioration du protocole AntNet

1. Introduction

Comme évoqué précédemment, une des contraintes majeures des RCSF est l'énergie. En effet, les capteurs sont munis d'une source d'énergie limitée et sont généralement déployés dans des zones où il est quasiment impossible de les réapprovisionner en énergie. La durée de vie du RSCF est donc étroitement liée à cette contrainte.

Le phénomène de congestion représente un des problèmes les plus consommateur d'énergie, car ce dernier est lié au nombre de paquets envoyés. Plus le nombre de paquets grand, plus on a une grande surcharge sur le réseau provoquant une dégradation du niveau d'énergie des capteurs minimisant la durée du vie du réseau.

Dans le protocole AntNet, la congestion se produit lorsque le flux de données devient important sur la route sélectionnée entre le nœud source émetteur et la station de base (sink).

Nous présentons dans ce chapitre, après avoir précisé les obstacles rencontrés dans ce projet, une première partie dans laquelle l'architecture du protocole AntNet ainsi que son routage sont discutés, pour ensuite présenter dans la deuxième partie de ce chapitre, les approches de solution proposées permettant d'améliorer le protocole AntNet en situation de congestion.

2. Obstacles rencontrés

Durant ces trois derniers mois de travail, nous avons rencontrés une multitude de problèmes où il était difficile d'en trouver une solution. Dans ce qui suit est détaillé l'ensemble des obstacles auxquels nous étions obligés de faire face :

- Le protocole sur lequel nous travaillons, en l'occurrence AntNet, est un protocole destiné aux réseaux filaires, chose à laquelle nous nous sommes pas rendu compte arrivés au chapitre de conception de son amélioration. Ceci nous a poussé

à arrêter notre travail en cours, et de se focaliser, durant plusieurs semaines, sur comment pourrait-t-on l'adapter aux réseaux sans fil.

- Après avoir fait des recherches approfondies sur ce problème, nous avons trouvé qu'il existait un protocole basé sur les colonies de fourmis similaire au nôtre, et qu'il était adapté au sans fil. Il s'agit du protocole AntHocNet. Mais nous nous sommes vite rendus compte après s'être documentés, que c'était un protocole hybride *multi-path* utilisant plusieurs chemins pour atteindre la station de base, mais aussi qu'il gérait le problème de congestion en diversifiant les routes.
- L'orientation de notre projet a été pour le domaine des RCSF, mais vu les difficultés et pour ne pas bloquer notre travail, nous nous sommes limités à le tester dans un environnement filaire. Cette solution pourrait être adaptable aux réseaux sans fil.

3. L'architecture du modèle AntNet

Le protocole AntNet dispose d'une architecture en couches composée de trois couches principales [38] (Figure IV.1) :

- **La couche physique :** composée de capteurs servant au captage, traitement de données et à la communication. Elle inclut la couche physique et liaison de données du modèle OSI.

- **La couche auto-organisation :** dans les réseaux de capteurs, tous les nœuds participent au relai des messages. A chaque capteur de la couche physique est associé des agents implémentant le modèle AntNet, et qui utilisent les fourmis (F-ant et B-ant) pour trouver la route optimale vers la station de base.

- **La couche message :** consiste en colonie de fourmis en interaction avec les agents afin de transporter la donnée vers sa destination. En plus de cela, elle servent aussi à la mise à jour des tables de phéromones qui sont locales à chaque nœud et qui portent des informations relatives au voisinage de ce dernier.

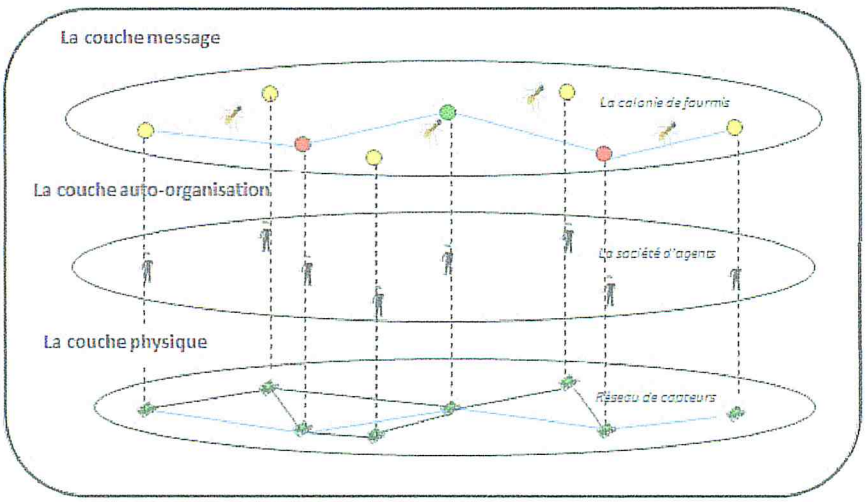

FIGURE IV.1 – La structure en couches du modèle AntNet.

3.1 Les interactions entre les agents et les fourmis

Les agents qui constituent la couche auto-organisation peuvent percevoir et exploiter les données que les fourmis (couche message) transportent, comme ils peuvent modifier les paramètres décisionnels de ces dernières. Les différentes interactions agents-fourmis sont présentées dans la figure suivante :

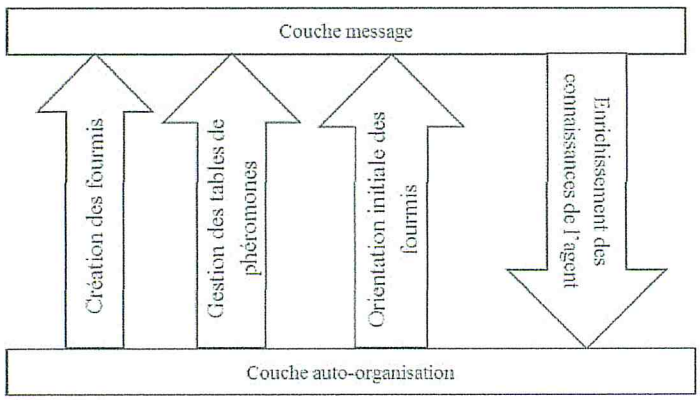

FIGURE IV.2 – Interactions entre les couches auto-ogannisation et message

- **Création des fourmis :** les agents de la couche auto-organisation génèrent les fourmis pour transporter leurs données à la station de base. Les PDU des deux couches agents et fourmis sont encapsulées comme suit :

  Ant-PDU :

@agentdest	Type	données

Agent-PDU :

@agentsrc	@agentdest	type	données

Concernant la couche fourmis :

- *@agentdest :* adresse de l'agent destinataire de la fourmi.
- *type :* identifie le type de la fourmi (forward, backward).
- *données :* consiste en une liste composée de nœuds déjà visités et la donnée, pour la fourmi forward. La route parcouru contenant les nœuds dont la quantité de phéromone doit être mise à jour, pour la fourmi backward.

Au niveau de la couche agent :

- *@agentsrc :* adresse de l'agent émetteur.
- *@agentdest :* adresse de l'agent destinataire de la fourmi.
- *type :* identifie le type de message (forward, backward, etc.).

- **Orientation initiale des fourmis :** Les agents créent des fourmis et donnent une orientation à leurs déploiements. Cette orientation est exprimée par initialisation des pistes de phéromone pour guider les premiers mouvements des fourmis. La phase d'initialisation de phéromones est lancée par les fourmis *F-ant*, ainsi ces dernières visitent tous les nœuds du réseau et déposent une quantité de cette matière proportionnelle au nombre de sauts nécessaires pour atteindre la station de base. Une fois arrivées à la station de base, celles-ci génèrent un autre type de fourmis *B-ant* qui prend le chemin inverse des *F-ant* pour atteindre le nœud source. A noter que les tables de routage des nœuds sont mises à jour par les *b-ant* en utilisant les informations recueillies par les *F-ant.* La *B-ant* augmente les probabilités associées au chemin empreinté à l'aller et diminue celles des autres chemins non utilisés. La *B-ant* s'arrête quand elle arrive au nœud source s.

- **La gestion des tables de phéromones :** La communication entre les fourmis se

fait à l'aide des tables de phéromones stockées au niveau de chaque capteur. Ces tables permettent de garder la trace des fourmis déjà passées par le nœud. La fourmi backward est chargée de faire des mises à jour au niveau des ces tables en augmentant les quantités de phéromones sur les liens parcourus par la fourmi.

- **Enrichissement des connaissances des agents :** Quand une fourmi arrive sur un nœud, elle apporte des informations sur son voisinage telles que la mobilité des nœuds voisins (leurs positions) et le nombre de sauts déjà effectués ce qui permet à l'agent de vérifier et de mettre à jour ses connaissances.

4. Le routage dans AntNet

Comme vu précédemment, le protocole AntNet crée une route unique entre le nœud source et la station de collecte. Lorsqu'un nœud veut envoyer un message, une fourmi *Forward* F-ant qui va chercher une route vers la destination est crée. Dans son processus de recherche de route, le paquet va être et à tout moment besoin de connaître son prochain saut parmi les nœuds voisins. le choix du prochain saut est assuré par la règle suivante :

$$p'_{nd} = \frac{p_{nd} + \alpha l_n}{1 + \alpha(|N_k| - 1)}$$

Où P_{nd} est la probabilité de choisir le nœud n comme prochain saut, d étant la destination. N_k représente l'ensemble des voisins du nœud actuel k et $|N_k|$ le cardinal de cet ensemble, alors que la correction heuristique *(heuristic correction)* l_n est une valeur normalisée [0,1] tel que 1-l_n est proportionnel à la longueur q_n de la file du lien connectant le nœud k avec son voisin n :

$$l_n = 1 - \frac{q_n}{\sum_{n'=1}^{|N_k|} q_{n'}}$$

La valeur de α dans cette règle pèse l'importance de la correction heuristique en respectant les valeurs de probabilités dans la table de routage.

Quand la fourmi *F-ant* arrive à sa destination, une fourmi *Backward B-ant* est crée au niveau de ce nœud. Le message de données est alors acheminé via la route indiquée par cette fourmi.

Les fourmis peuvent modifier les paramètre locaux des nœuds en modifiant leurs tables de phéromones (taux d'évaporation et d'occupation de phéromones) afin d'aider les prochaines fourmis et donner aux nœuds une vue plus claire sur leur environnement.

La classe *antnet* regroupe l'ensemble des tâches effectuées par les agents d'AntNet, elles se résument dans les fonctions décrites dans le tableau suivant :

routage de paquets dans AntNet. Les fourmis dotent les agents avec plus d'informations sur leurs voisinages, alors que les agents peuvent influencer sur les paramètres décisionnels des fourmis. L'envoi d'un message dans le protocole AntNet est décrit dans le diagramme présenté dans la figure suivante :

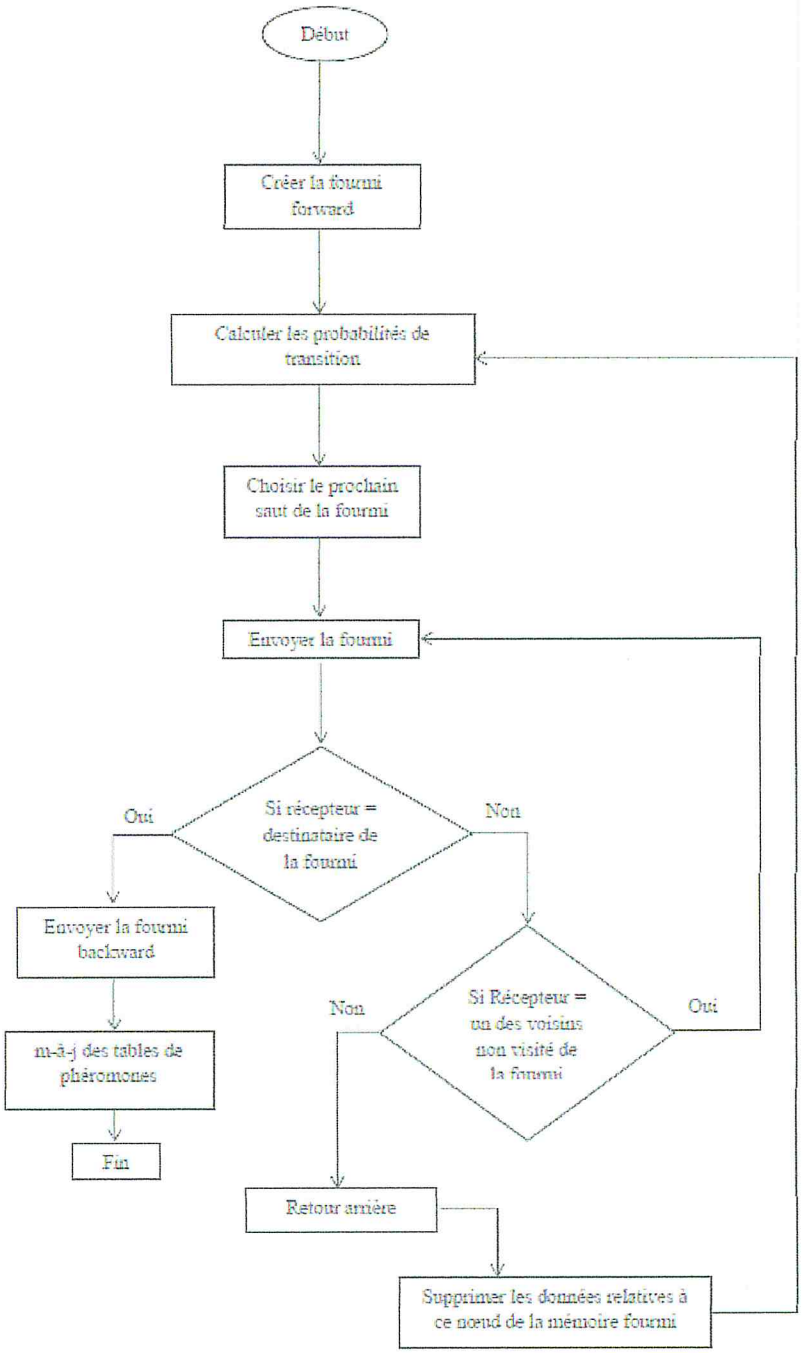

FIGURE IV.3 – Envoi d'un message via AntNet.

5. Limitations du protocole AntNet

Bien que le protocole AntNet semble idéal pour l'acheminement des données entre le capteur source et la station de base, il présente néanmoins des limitations, plus précisément les problèmes de stagnation auxquels il ne peut éviter.

La stagnation se produit quand un réseau atteint son état d'équilibre. Ceci est une propriété indésirable de l'algorithme de routage où les fourmis choisissent de manière récursive le même chemin pour arriver à leur destination. Par conséquent, la probabilité d'emprunter ce chemin sera élevée conduisant ainsi de plus en plus de paquets à être envoyés sur ce chemin, et réduisant bien évidemment les probabilités des autres chemins.

L'optimisation du routage peut donc se bloquer dans un optimum local et ne pas être en mesure de découvrir de nouveau chemins qui pourraient devenir optimaux, en cas de changements dans le trafic ou la topologie (défaillances d'un lien/nœud, suppression/ajout de nouveaux nœuds, etc.).

La stagnation devient un problème critique lorsque la taille du réseau augmente. En effet, cela va engendrer plus de paquets transitant dans le réseau et donc plus de trafic, l'utilisation d'un seul chemin pour acheminer un nombre important de paquets peut engendrer une situation de congestion se traduisant par la perte de paquets, l'augmentation du délai d'acheminement de messages, mais surtout le temps de traitement au niveau des nœuds intermédiaires, amenant à un épuisement des ressources énergétiques de ces capteurs.

Nous introduisons dans ce qui suit, notre modèle de solution à la congestion. Nous avons vu au deuxième chapitre qu'il existait deux types de congestion : la congestion au niveau nœud et la congestion au niveau du lien. Nous nous concentrons dans ce qui suit sur la congestion au niveau du lien.

La solution proposée, dans une situation pareille est de diminuer le taux de transfert des nœuds émetteurs à un taux acceptable en prenant compte de la bande passante du lien du nœud commun, équilibrant les charges du réseau et acheminant ainsi l'ensemble des données à la station de base, sans qu'il y est perte de paquets.

6.2 Cas 2 : Changement de route

Dans ce mode, après détection de congestion au niveau d'un lien, le trafic est redirigé à un autre chemin vers la destination. Le choix de ce chemin alternatif est fait en analysant l'ensemble des entrées correspondantes à la destination présentes dans la table de routage du nœud sur lequel le lien est congestionné. Parmi ces entrées, le chemin disposant de la plus haute valeur de probabilités (phéromones) sera sélectionné pour acheminer la donnée (Fig. IV.5).

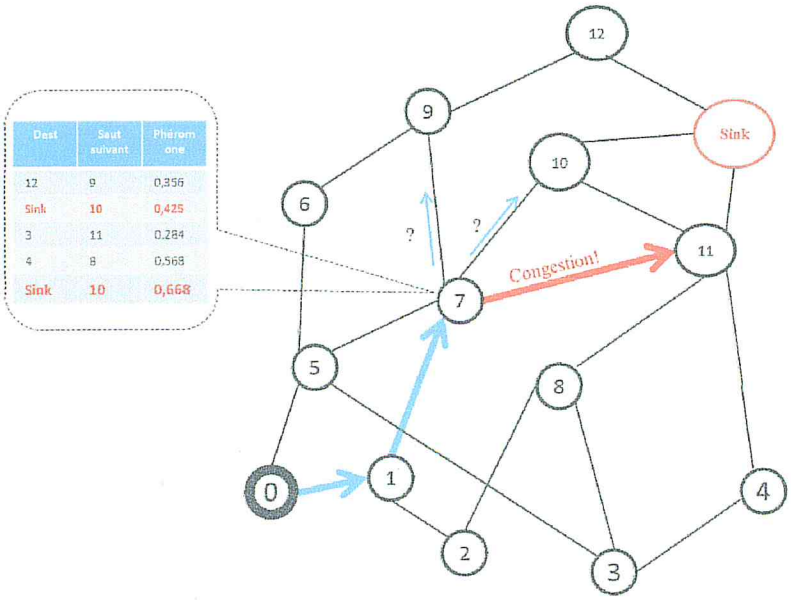

FIGURE IV.5 – Schéma illustratif du deuxième cas.

Comme le montre ce schéma, arrivée à un lien congestionné, la table de routage du nœud concerné (nœuds 7) est analysé pour chercher un chemin alternatif à celui-ci. On voit bien de ce schéma que la route *7-10-Sink* est empruntée car sa probabilité est élevée par rapport à l'autre chemin possible (*7-9-12-Sink*. Celui-ci est donc le nouveau

chemin optimal.

6.3 Cas 3 : Utilisation de la fenêtre de congestion du protocole TCP

Les protocoles TCP et UDP de la couche transport sont chargés d'acheminer les données entre source et destination. Alors que l'un (UDP) est un protocole sans connexion, c-à-d aucune session n'est établie entre les deux communicants au préalable, rendant l'acheminement de données non fiable, l'autre en contre partie, permet une transmission fiable de données à travers l'établissement de session entre l'émetteur et le récepteur (*Three-way Handshake*), mais aussi à travers l'utilisation de ce qu'on appelle accusés de réception ou ACK ; message d'acquittement envoyé par le destinataire à la source pour indiquer que la donnée à bien été reçue.

Contrairement au protocole UDP, qui lui ne dispose d'aucun mécanisme de contrôle de flux, le protocole TCP dispose d'un mécanisme pour le contrôle du flux permettant de réguler la quantité de données transmises sur le réseau. Le mécanisme en question est appelé fenêtre de congestion ou *congestion window* en anglais (*cwnd*).

6.3.1 Principe

La fenêtre de congestion représente un critère, parmi d'autres, determinant la performance du protocole TCP. La taille de la fenêtre représente la quantité de données, en termes de paquets, qu'une source peut transmettre avant qu'un accusé de réception ne doit être reçu.

Comme aucune connaissance du réseau n'est connue à priori, l'idée est donc de commencer d'une valeur très basse mais d'augmenter rapidement pour que l'efficacité reste bonne si le réseau est performant. La taille de la fenêtre est augmentée (donc le débit) jusque l'on arrive à une perte qui signifie la congestion. L'émetteur à ce moment là, va diminuer la fenêtre pour réduire son débit d'émission.

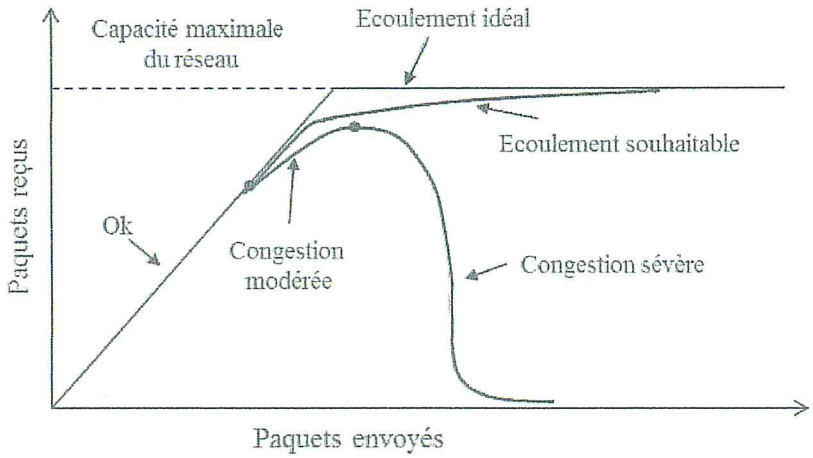

FIGURE IV.7 – Performance en congestion.

7. Conclusion

Visant à optimiser les performances du protocole AntNet en minimisant le taux de perte de paquets provoqué par les situations de congestion, nous avons proposé dans ce chapitre des approches basées sur le contrôle de congestion, afin de détecter et de gérer cette dernière.

Dans ces approches, diverses solutions sont proposées. Nous citons entre autres, l'équilibrage des charges qui est fait à travers la régulation du taux de transfert de la source émettrice dans le but de palier au problème de congestion.

Dans le chapitre suivant, nous allons tester les performances du protocole AntNet ainsi que son amélioration en parallèle, en utilisant le simulateur NS2.

Chapitre V

Simulations et résultats

1. Introduction

Afin d'évaluer les performances du protocole AntNet, nous avons fait recours à la simulation qui consiste à reproduire le comportement des entités du monde réel dans le monde virtuel. Pour cela, nous avons utilisé le simulateur NS qui simule les réseaux de capteurs et qui nous a permit d'implémenter et de tester notre protocole.

La simulation dans les réseaux de capteurs permet d'évaluer et de tester les protocoles dans ce domaine. Elle offre un gain considérable en temps et en argent, tout en permettant la variation des paramètres afin d'avoir une bonne visibilité des résultats.

Dans ce chapitre, nous débutons par la présentation de l'environnement de simulation. Ensuite, nous abordons les étapes d'implémentation du protocole AntNet et nous clôturons le chapitre par des tests et interprétations des résultats obtenus.

2. Motivations

Aujourd'hui, la simulation de l'Internet est devenu assez difficile pour plusieurs raisons, elles peuvent se résumer par la grande hétérogénéité et l'évolution rapide des technologies d'Internet. L'hétérogénéité commence avec les liens, comprend les protocoles et finit avec les trafics d'applications diverses et variées.

L'Internet évolue rapidement en terme de taille mais également en terme d'architecture et de topologie. C'est devant ce constat que des chercheurs ont essayé d'élever l'état de l'art de la simulation de l'Internet. L'étude des comportements à des échelles différentes d'un réseau n'est pas obtenue par la simulation parallèle (qui peut être utile pour accélérer la simulation) mais par l'utilisation de techniques d'abstraction appliquées à différents éléments de la simulation. VINT est un projet en cours qui développe le simulateur NS [40].

Le simulateur NS actuel est particulièrement bien adapté aux réseaux à commuta-

tion de paquets et à la réalisation de simulations de petite taille. Il contient les fonctionnalités nécessaires à l'étude des algorithmes de routage unipoint ou multipoint, des protocoles de transport, de session, de réservation, des services intégrés, des protocoles d'application comme HTTP. De plus le simulateur possède déjà une palette de systèmes de transmission (couche 1 de l'architecture TCP/IP), d'ordonnanceurs et de politiques de gestion de files d'attente pour effectuer des études de contrôle de congestion.

La liste des principaux composants actuellement disponible dans NS par catégorie est donnée dans le tableau suivant :

Application	**Web, ftp, telnet, générateur de trafic (CBR, ...)**
Transport	TCP, UDP, RTP, SRM
Routage	Statique, dynamique (vecteur distance) et routage multipoint (DVMRP, PIM)
Gestion de file d'attente	RED, DropTail, Token bucket
Discipline de service	CBQ, SFQ, DRR, Fair queueing
Système de transmission	CSMA/CD, CSMA/CA, lien point à point

TABLE V.1: La liste des principaux composants actuellement disponibles dans NS [40]

Prises ensembles, ces capacités ouvrent le champ à l'étude de nouveaux mécanismes au niveau des différentes couches de l'architecture réseau. NS est devenu l'outil de référence pour les chercheurs du domaine. Ils peuvent ainsi partager leurs efforts et échanger leurs résultats de simulations. Cette façon de faire se concrétise aujourd'hui par l'envoi dans certaines listes de diffusion électronique de scripts de simulations NS pour illustrer les points de vue.

3. Environnement de simulation

L'outil NS-2 fournit un ensemble d'objets TCL spécialement adaptés à la simulation de réseaux. Avec les objets proposés par ce moteur de simulation, on peut

représenter des réseaux avec liens filaires ou sans fils, des machines et routeurs (Node), des flux TCP et UDP (par exemple pour simuler un flux CBR), et sélectionner les politiques et règles régissant les files d'attentes mises en œuvre dans chacun des nœuds. Il est devenu aujourd'hui un standard de référence en ce domaine. Son utilisation est gratuite et il est exécutable tant sous Unix que sous Windows.

NS-2 ne permet pas de visualiser le résultat des expérimentations. Il permet uniquement de stocker une trace de la simulation, de sorte qu'elle puisse être exploitée par un autre logiciel, comme NAM [41].

3.1 Utilisation de l'interpréteur

Dans ce qui suit nous allons présenter les bases du langage Tcl, les principes de l'OTcl et les explications sur le mécanisme qui permet à un programme C d'utiliser un interpréteur Tcl.

3.1.1 Le langage de script TCL

Tcl est un langage de commande comme le shell UNIX mais qui sert à contrôler les applications. Son nom signifie Tool Command Language.

Le langage TCL est un langage de script puissant qui permet d'utiliser éventuellement une approche de programmation orientée objet. Il est facilement extensible par un certain nombre de modules.
l'interpréteur Tcl se présente sous la forme d'une bibliothèque de procédures C qui peut être facilement incorporée dans une application. Cette application peut alors utiliser les fonctions standards du langage Tcl mais également ajouter des commandes à l'interpréteur.

Dans notre cas, il est indispensable d'utiliser le langage TCL pour pouvoir travailler avec les objets fournis par NS-2.

3.1.2 OTcl

OTcl est une extension orientée objet de Tcl. Les commandes Tcl sont appelées pour un objet. En OTcl, les classes sont également des objets avec des possibilités d'héritage.

3.1.3 Liens C++ et Tcl

Construire une application avec un interpréteur Tcl revient à inclure une bibliothèque Tcl qui définit les commandes de bases de Tcl dans l'application. L'interpréteur effectue l'analyse syntaxique et appelle la fonction C correspondant à la commande Tcl. Ajouter une commande Tcl consiste à établir un lien entre un mot et une fonction C. Le mot sera le nom de la commande Tcl. La fonction C est définie dans le code source de l'application. Au démarrage, l'application procède dans son *main()* aux initialisations nécessaires et passe la main à l'interpréteur. L'application passe en mode interactif : à chaque commande tapée par l'utilisateur, la fonction C correspondante est appelée afin de réaliser la commande demandée.

3.2 L'outil de visualisation NAM

NAM (*Network AniMator*) est un outil de visualisation qui présente deux intérêts principaux : représenter la topologie d'un réseau décrit avec NS-2, et afficher temporellement les résultats d'une trace d'exécution NS-2. Par exemple, il est capable de représenter des paquets TCP ou UDP, la rupture d'un lien entre nœuds, ou encore de représenter les paquets rejetés d'une file d'attente pleine.

Ce logiciel est souvent appelé directement depuis les scripts TCL pour NS-2, de sorte à visualiser directement le résultat de la simulation (Fig. V.1).

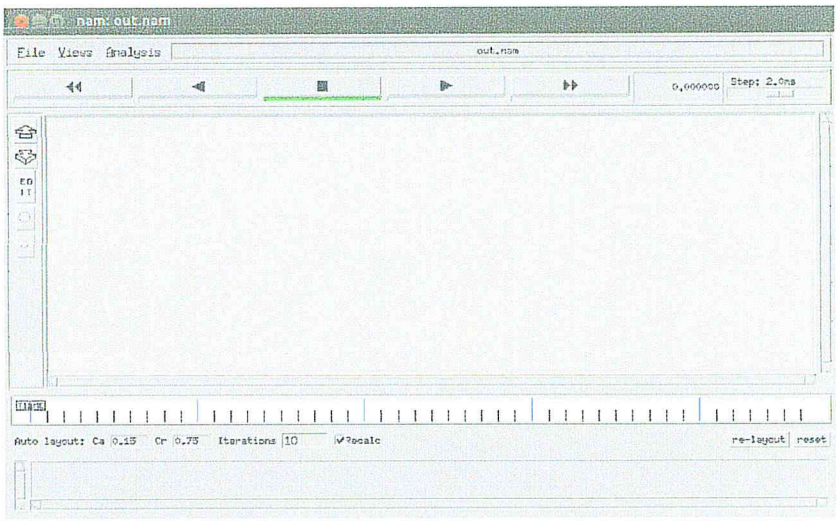

FIGURE V.1 – Inerface NAM.

4. Simulations et interprétations des résultats

Dans le cadre de notre projet de fin d'études, nous avons réalisé une étude détaillée du protocole AntNet afin d'éviter le problème de congestion dans les réseaux de capteurs. La problématique considérée en premier lieu concerne la gestion des files d'attentes, dans le but de modéliser les flux de données dans un réseau.

En effet, un réseau se compose d'un ensemble de files d'attentes intervenant à chaque étape du traitement : commutation, routage, réception de paquets ce sont ces dernières qui permettent de gérer les congestions sur un lien en limitant la quantité de paquets qui peuvent être mis en attente pendant que la liaison est occupée.

Cette partie comprendra en premier lieu, une étude de différentes types de file d'attente en situation de congestion, pour ensuite entamer les différentes simulations de nos améliorations apportées au protocole AntNet. Le tout accompagné d'analyses et interprétations des résultats.

4.1 Simulation de la congestion

Suite à notre familiarisation avec le simulateur NS-2, nous avons développé des simulations dont les détails opératoires et les résultats obtenus sont exposés ci-joint.

Dans le but de simuler une congestion dans un réseau, nous avons réalisé un montage correspondant au diagramme suivant :

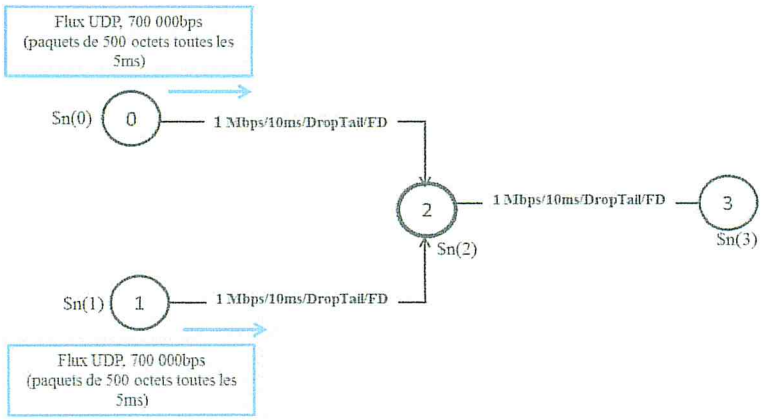

FIGURE V.2 – Schéma de congestion

Soit deux nœuds qui émettent chacun un CBR proche de 0.7 Mbps vers un nœud qui les écoutent, en passant par un nœud intermédiaire qui les relie. Ceci, conduit à la congestion qui aura lieu entre le nœud intermédiaire et celui de destination, problème causé par la limite du lien qui correspond à un seuil équivalant à 1 Mbps. sachant que la somme des CBR émis à la destination vaut 1,4 Mbps, valeur plus élevée.

4.2 Étude des différentes types de files d'attente

Dans cette partie, nous avons choisi d'étudier trois types de files d'attente les plus utilisées, en l'occurrence DropTail(FIFO), SFQ (Short Fairness Queueing) et RED (Random Early Detection). Cette étude va nous servir dans la suite de notre travail dans la mesure où elle nous permettra de déterminer la file d'attente la plus performante en termes de taux de pertes de paquets, en situation de congestion. Les différents calculs relatifs au nombre de paquets perdus et pourcentages sont fait en étudiant le fichier

trace résultant de chacune des simulations.

Afin d'obtenir des résultats concluants, nous avons opté pour une simple topologie composée de quatre nœuds, utilisant les mêmes paramètres de simulation pour les trois type de files donnés comme suit :

Paramètre	**Valeur**
Temps de début	0.5 s
Temps de fin	4.5 s
Taille de paquet	500 octets
Intervalle de génération de paquets	5 ms
Nombre total de paquets	1600

TABLE V.2: Paramètres de simulation.

4.2.1 DropTail

DropTail est un algorithme de file d'attente FIFO (First In, First Out – premier arrivé, premier servi). En s'appuyant sur le schéma de congestion vu précédemment, l'ensemble des files d'attente des liens est donc mis à DropTail. A l'exécution de la simulation, nous obtenons le résultat illustré dans la figure suivante :

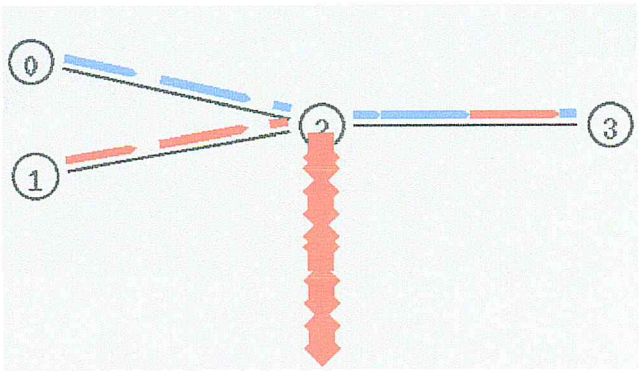

FIGURE V.3 – Résultat visuel de la simulation avec DropTail.

Interprétations :

Comme nous pouvons le constater et après un certain temps, le lien reliant les nœuds 2 et 3 devient saturé car la quantité de paquets à transmettre est trop importante par rapport à la taille du lien, et que la file d'attente du lien est surchargée. Comme la politique utilisée est de type FIFO, ce sont les premiers paquets reçus qui sont transmis.

Nous remarquons également, à partir des paramètres de simulation et en analysant le fichier trace qu'un paquet sur deux transmis du nœud 2 vers le nœud 3 provient du flux UDP du nœud 0 alors que l'autre moitié provient du flux UDP du nœud 1. Dès que la file d'attente est pleine, 553 paquets seront perdus sur un total de 1600 (34,5%) sont rejetés. Étant donné l'ordre d'arrivée des paquets, ce sont toujours les paquets provenant du nœud 1 qui sont rejetés, à hauteur de 69,1%.

Avec la discipline FIFO, tous les paquets provenant du nœud 0 sont transmis au nœud 3, et environ un quart des paquets du nœud 1 sont transmis au nœud 3, tandis qu'environ trois quarts des paquets émis depuis le nœud 1 à destination du nœud 3 sont perdus.

4.2.2 SFQ

Contrairement à DropTail, qui est un algorithme FIFO, SFQ (Stochastic Faireness Queueing) quand à lui, est supposé être un algorithme de répartition équitable. Afin d'étudier les performances de SFQ, nous avons donc remplacé la discipline de file d'attente des trois liens par une file SFQ. A l'exécution de la simulation, nous obtenons le résultat illustré dans la figure suivante :

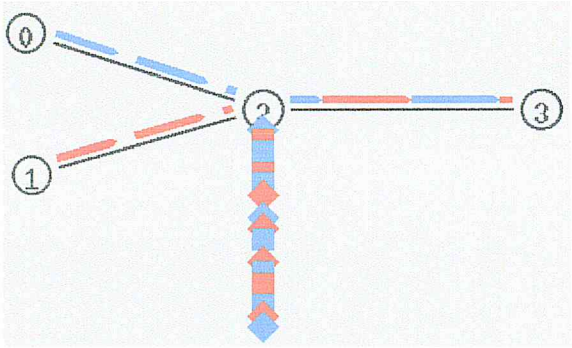

FIGURE V.4 – Résultat visuel de la simulation avec SFQ.

Interprétations :

Avec la discipline SFQ, nous avons remarqué que la moitié des paquets rejetés proviennent du flux UDP du nœud 0, alors que l'autre moitié provient du flux UDP du nœud 1. En analysant le fichier trace résultant qu'avec cette discipline, le nombre de paquets perdus est un peu plus grand par rapport à DropTail ; 582 paquets ont été rejetés, ce qui représente environ 36,75% du nombre total de paquets circulant dans le réseau.

Cette discipline présente donc l'avantage d'être en effet équitable : les deux flux ont la même priorité et sont acheminés dans les mêmes conditions, ce qui permet d'avoir une transmission harmonieuse des données sur le réseau. La discipline de file d'attente RED est étudiée dans la section qui suit.

4.2.3 RED

Nous avons testé le troisième type de file d'attente, qui est l'algorithme RED (Random Early Detection). Le déroulement de la simulation a donné le résultat visuel suivant :

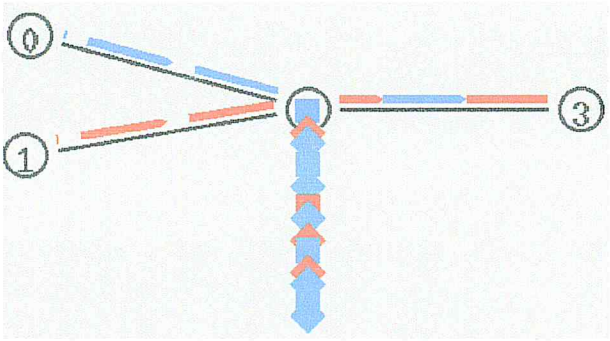

FIGURE V.5 – Résultat visuel de la simulation avec RED.

Interprétations :

Avec l'algorithme de file d'attente RED, on constate qu'en situation de congestion les paquets perdus proviennent des deux flux UDP des nœuds n0 et n1 sauf que cette perte n'est pas totalement équitable. En effet, 577 (36% du total) paquets sont perdus parmi lesquels 46,27% proviennent du flux émis par le nœud n0, tandis que 53,73% des paquets perdus sont émis par le nœud n1.

4.3 Conclusion de l'étude

D'après les résultats présentés dans cette partie, nous pouvons dire que l'ensemble des trois types de file d'attente ne permettent toujours que de détecter la congestion au niveau du nœud n2. Par conséquent, l'inconvénient est qu'une partie de la bande passante est consommée sur les liens n0-n2 et n1-n2 est inutilement perdue car approximativement un tiers des paquets sont jetés au niveau du nœud n2. Ces solutions ne permettent donc pas d'optimiser la bande passante globale du réseau.

Toutefois, d'un point de vue taux de perte de paquets, nous pouvons dire que DropTail représente la meilleure solution parmi les trois, puisqu'elle possède le taux de perte le moins élevé avec 34,5% des paquets. A noter que DropTail prend en compte l'ordre d'arrivée des paquets en situation de congestion, en rejetant toujours les paquets émis par le nœud depuis lequel les premiers paquets sont arrivés.

Par conséquent, la file d'attente DropTail sera utilisée dans les simulations à venir du protocole AntNet et de son amélioration.

4.4 Analyse de la bande passante en cas de congestion

La bande passante représente un critère important d'évaluation de tout protocole de routage. il permet de déterminer la quantité de données qui peuvent être acheminées en une unité de temps. C'est pourquoi, nous avons voulu voir l'importance de ce critère en situation de congestion. Nous avons donc mis en place une topologie de 10 nœuds dans laquelle trois nœuds (nœuds 1, 6 et 8) génèrent simultanément de 2 à 9.5 secondes de simulation, un trafic CBR vers une unique station de base (nœud 4). Cette dernière enregistre les nombre d'octets reçus pour ensuite calculer la bande passante en Mbits par seconde.

Afin de créer une situation de congestion, nous avons guider le trafic venant de deux des trois nœuds (nœuds 6 et 8), de manière à ce qu'ils passent par le même chemin, à un moment de la simulation. Comme ça le trafic combiné dépasse la bande passante du lien à travers lequel ils doivent passer, créant ainsi une congestion au niveau du lien (Figure V.6). Au niveau du nœud *sink* (station de base) est mesuré le nombre d'octets reçus. Les paramètres des trois sources de trafic est la suivante :

- **Trafic 1 :** du nœud 1, taille de paquets est à 100 octets, intervalle de génération des paquets est de 5 ms (200 paquets/s).
- **Trafic 2 :** provenant du nœud 6, taille de paquets est à 300 octets, l'intervalle est de 5 ms également.
- **Trafic 3 :** généré du nœud 8, la taille d'un paquet est mise à 400 octets et l'intervalle de génération est le même (5 ms).
- La bande passante du lien entre les nœuds 5 et 4 par lequel les trafics provenant des nœuds 6 et 8 doivent passer est fixée à 0.7 Mbits/s.

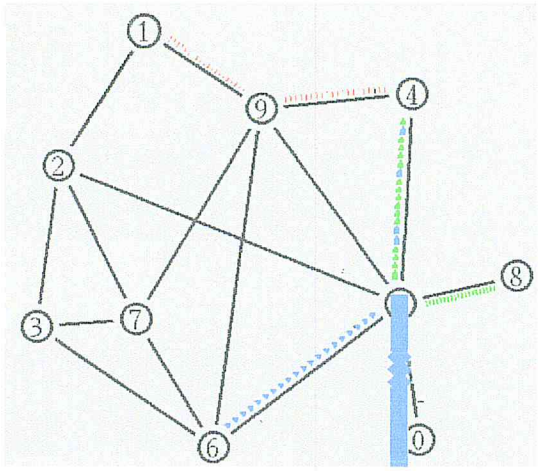

FIGURE V.6 – Congestion au niveau du lien entre 5 et 4.

Les résultats de l'évolution de la bande passante en fonction du temps(en secondes) est montré à travers le graphe comparatif ci-dessous où chaque correspond à la couleur du trafic sur la figure V.6 :

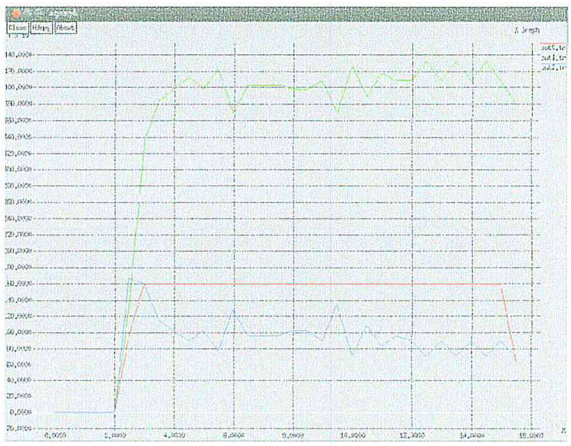

FIGURE V.7 – Graphe comparatif du nombre d'octets reçus au sink.

Interprétations

A partir des résultats obtenus, nous remarquons que le trafic généré au niveau du nœud 1 atteint rapidement le seuil de son débit (0,16 Mbits/s), avant d y rester durant le reste de la simulation. Par conséquent, le maximum du débit a été utilisé et tous les

paquets on été reçus au niveau du nœud sink.

Concernant les deux autres courbes, nous pouvons clairement constater une perte de paquets, car à aucun moment de la simulations ils n'atteignent leurs seuils : 0,48 Mbits/s pour le trafic provenant du nœud 6 (bleu) et 0,64 Mbits/s pour celui qui provient du nœud 8. Donc, vu la capacité du lien 5-4, une partie du débit des deux sources est inutilement perdue, ce qui explique qu'au niveau du sink, l'ensemble des paquets générés n'arrivent pas tous à destination.

4.5 Améliorations de AntNet

Nous présentons dans cette partie, les différentes solutions proposées dans le chapitre précédant pour apporter des améliorations au protocole AntNet, visant à gérer la congestion en minimisant en maximum le nombre de paquets perdus. A noter que AntNet ne permet pas le contrôle de congestion. En situation de congestion, celui-ci perd les paquets de manière continue, ce qui pénalise le processus de routage.

Parmi les approches proposées, nous avons implémenté deux solutions : la fenêtre de congestion du protocole TCP et le changement de route à détection de congestion.

4.5.1 La fenêtre de congestion de TCP

La fenêtre de congestion est un mécanisme important du protocole TCP car, comme expliqué dans le chapitre précédant, il permet de faire un contrôle de congestion au niveau du nœud source. Nous avons donc testé ce mécanisme sur le scénario de simulation suivant :

- Nous considérons un réseau de 10 capteurs (Fig. V.6) dispersés sur une surface donnée où un des nœuds envoi un flux CBR à la station de base.

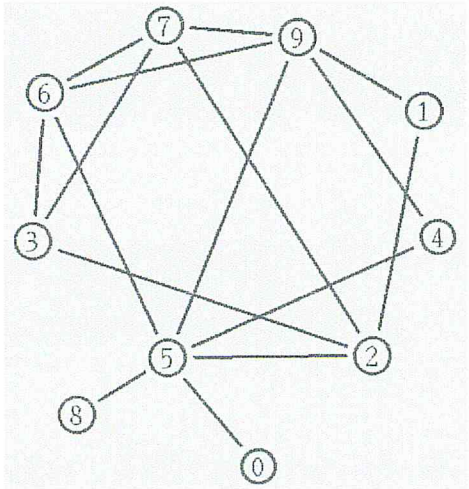

FIGURE V.8 – Scénario avec une topologie de 10 noeuds.

Les paramètres utilisés pour cette simulation sont les suivants :

Nombre de nœuds	10
Début du trafic	2.0 s
Durée de la simulation	60 s
Intervalle de génération de fourmis	50 ms
Taille de paquet	500 octets
Trafic	TCP + CBR
initialistion de cwnd	1

TABLE V.3: Paramètres de simulation.

Le module *Xgraph* de NS2 est utilisé pour tirer les résultats de ces simulations illustrés dans la figure suivante :

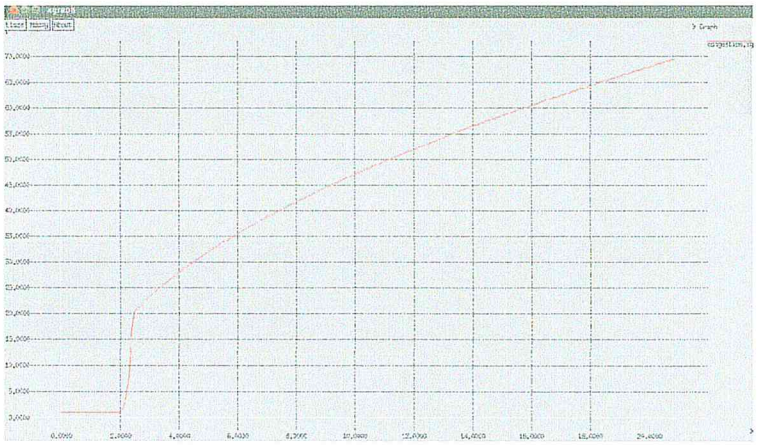

FIGURE V.9 – Résultat de la simulation pour le scénario.

Interprétations :

A travers l'analyse du graphe de la fenêtre de congestion résultant, nous pouvons constater que la fenêtre passe par deux états : l'état dit du démarrage lent (*slow start)*, et l'état évitement de congestion (*congestion avoidance*). En effet, nous nous apercevons qu'avant démarrage du trafic, la taille de la fenêtre de congestion est initialisée à 1 MSS (Maximum Segment Size), soit 1 paquet. Lorsque le trafic commence a être généré, ce qui correspond à la 2ème seconde, l'émetteur entre en slow start et, à mesure de réception d'ACKs, sa fenêtre de congestion est incrémentée de 1 pour chaque ACK reçu, ce qui revient à dire que sa taille est doublée à chaque transmission acquittée et le débit d'émission de l'émetteur est donc augmenté.

Ce processus se poursuit jusqu'à arrivée au seuil *twnd* à environ 2.5 secondes. Au delà de ce seuil, la fenêtre entre en état d'évitement de congestion dans lequel elle est augmentée de manière linéaire tant qu'aucune perte (congestion) n'est détectée, et ce jusqu'à arrivée à un débit maximal sans qu'il y est congestion.

De cette manière, les capacités du réseau sont exploitées au maximum, tout en

contrôlant le débit d'émission des émetteurs pour ne pas tomber sur des situations de congestion causant éventuellement des dégradations significatives des performances du réseau.

4.6 La perte de paquets dans le réseau

Pour voir la différence entre AntNet et l'amélioration apportée, le taux de perte de paquets représente un critère d'évaluation de performance important, car il résulte des situations de congestions auxquelles le réseau fait face. Nous avons donc effectué deux simulations avec deux scénarios différents, faisant une étude comparative entre AntNet et l'amélioration apportée, du nombre de paquets perdus en congestion. les paramètres de simulation sont les suivants :

Intervalle de génération de fourmis	5 ms
facteur de renforcement	0.05
Durée de simulation	10 s
Taille d'un paquet	500 octets
Débit d'envoi	0,8 Mbps

TABLE V.4: Paramètres de la simulation et du protocole AntNet.

4.6.1 Simulation 1 : Perte de paquets entre plusieurs topologie

Dans le tableau suivant est décrit le nombre de nœuds contenus dans chaque topologie :

Scénario 1	10 capteurs
Scénario 2	20 capteurs
Scénario 3	30 capteurs
Scénario 4	40 capteurs
Scénario 5	50 capteurs
Scénario 6	60 capteurs

TABLE V.5: Scénarios de simulation.

Après exécution, les résultats obtenus sont illustrés dans la figure suivante :

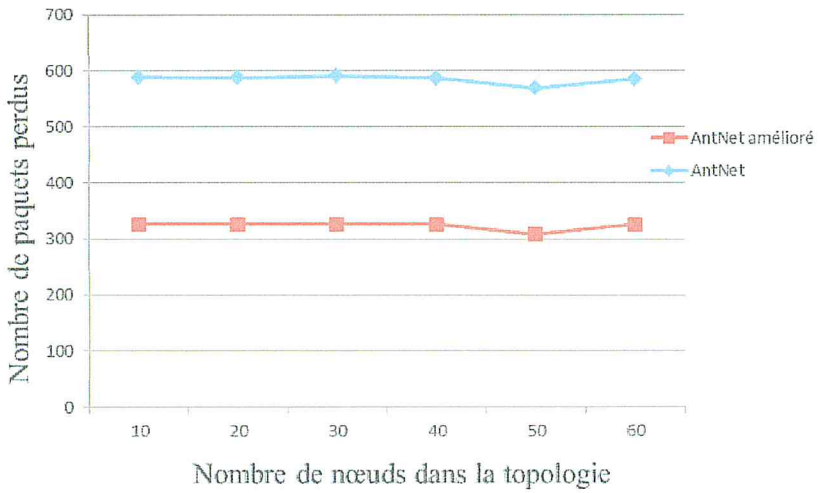

FIGURE V.10 – Comparaison entre AntNet et l'amélioration en termes pertes de paquets.

Interprétations :

Nous remarquons d'après l'analyse de la figure ci-dessus, que les performances du protocole AntNet amélioré sont nettement supérieures par rapport au protocole classique, en cas de situation de congestion. En effet, en regardant le nombre de paquets perdus dans chacun des protocoles, nous pouvons dire que l'amélioration proposée permet de minimiser le taux de perte de paquets, en situation de congestion à presque la moitié.

4.6.2 Simulation 2 : Perte de paquets par tranche de temps

Concernant la deuxième expérience et sur la base de la topologie de 50 nœuds, nous avons étudié et comparé la perte de paquets entre AntNet et celui amélioré.

Nous avons consigné respectivement au script de simulation comme paramètres le nœud 1 comme nœud source et le nœud 20 comme étant la station de base. La topologie résultante est la suivante :

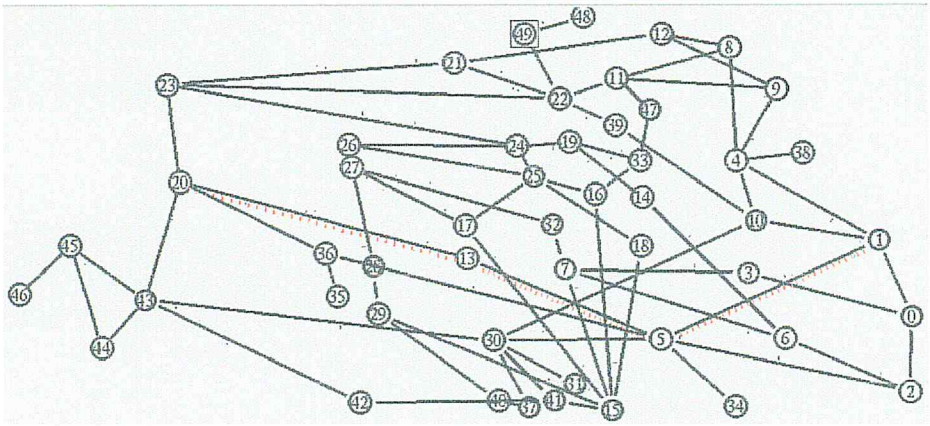

FIGURE V.11 – Scénario avec une topologie de 50 noeuds.

En analysant le fichier trace (Figure V.10), nous remarquons le chemin optimal parcouru par les paquets :

```
+ 2.005 1 5 cbr 1000 ------- 2 1.1 20.1 2 1703
- 2.005 1 5 cbr 1000 ------- 2 1.1 20.1 2 1703
+ 2.005 1 5 cbr 500 ------- 2 1.1 20.1 3 1704

+ 2.155667 5 13 cbr 1000 ------- 2 1.1 20.1 0 1700
- 2.155667 5 13 cbr 1000 ------- 2 1.1 20.1 0 1700

+ 2.318667 13 20 cbr 1000 ------- 2 1.1 20.1 0 1700
- 2.318667 13 20 cbr 1000 ------- 2 1.1 20.1 0 1700
```

FIGURE V.12 – Chemin optimal choisit pour acheminer les paquets

Le graphe obtenu ci-dessous représente le nombre de paquets perdus par les deux protocoles en dix secondes de simulation :

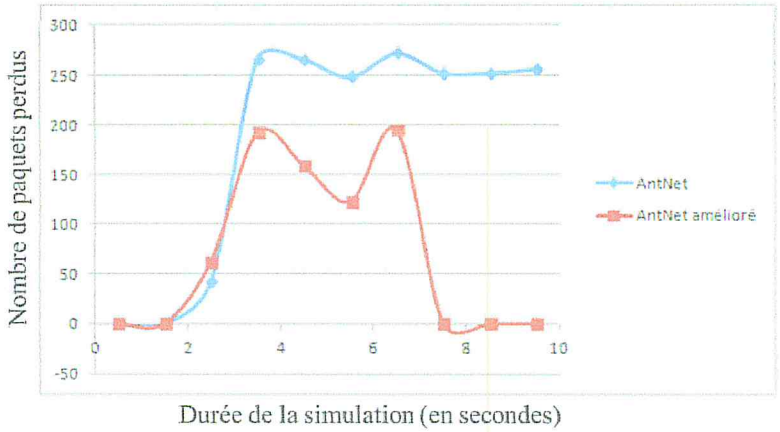

FIGURE V.13 – Comparaison entre AntNet et son amélioration

A titre comparatif, du nombre de paquets perdus lors de la simulation, nous notons que la perte s'amorce à partir d'une durée de simulation proche de 2.5s, logique car c'est un peu avant ce temps que le trafic commence à être généré. Le nombre de paquets perdus augmente ensuite jusqu'à atteindre un seuil d'environ 270 pour AntNet classique et 190 pour son amélioration. En ce qui concerne le protocole classique, le nombre de ses paquets perdus garde une valeur quasiment constante, après avoir atteint son seuil. Alors que pour son amélioration, à environ 3.7s nous remarquons une baisse du nombre de paquets perdus. Cela s'explique par la détection par le déclenchement de la solution en redirigeant le trafic vers un autre chemin menant à la destination. A environ 5.8s, nous constatons une remontée du nombre de paquets perdus qui ne peut s'expliquer que par une apparition d'une nouvelle situation de congestion au niveau du chemin alternatif choisit. La solution est appliquée une deuxième fois, c'est pourquoi nous voyons que le nombre de perdus perdus redescend rapidement : il n y a plus de situation de congestion et les paquets sont acheminés à destination.

En conclusion, on voit bien que l'AntNet amélioré conduit à moins de perte en ce qui concerne le nombre de paquets.

5. Conclusion

Disposant d'un seul chemin reliant la source et la station de base, les performances du protocole AntNet en cas de trafic excessif se dégradent de manière remarquable amenant à une congestion qui résulte par l'augmentation du délai d'acheminement mais surtout la perte de paquets, comme vu à travers les différentes simulations menées tout au long de ce chapitre.

L'amélioration du protocole AntNet que nous avons proposé se base principalement sur la présence de la congestion au niveau lien, et vise à optimiser les performances globale du réseau en offrant des alternatives aux routes créant une congestion tout en minimisant le taux de paquets perdus.

Conclusion générale

Les réseaux de capteurs représentent aujourd'hui, un axe de recherche très convoité et jouent un rôle essentiel dans le processus de collecte des informations. leur faible coût, facilité d'installation et auto organisation leurs permettent d'être appliqués dans plusieurs domaines différents. Néanmoins, ils restent encore vulnérables à différents problèmes qui dégradent leurs performances. Parmi les problèmes auxquels on peut faire face dans un réseau de capteurs est la congestion. Celle-ci, souvent causée par l'accumulation du trafic, amène à un gaspillage des ressources dont la bande passante et l'énergie. Il est donc primordial de mettre en place des mécanismes et protocoles pour le contrôle de congestion dans le but d'assurer le bon fonctionnement des réseaux de capteurs.

Nous étions intéressés tout au long de notre projet de fin d'étude à l'amélioration du protocole de routage AntNet et l'intégration de mécanismes de contrôle de congestion à son niveau. Pour cela, nous avons tout d'abord fait une étude bibliographique afin de comprendre et d'assimiler le domaine de recherche, mais aussi de bien cerner la problématique. Nous avons donc commencé par étudier de manière générale les réseaux de capteurs, ainsi que la fonction de routage dans ce type de réseau. Nous avons cité quelques uns des protocoles de routage proposés dans la littérature, en portant une attention particulière sur le protocole AntNet.

Après avoir fait l'étude bibliographique, nous avons abordé dans la conception de l'amélioration du protocole AntNet, et ce en prenant en compte l'état du trafic du réseau, minimisant ainsi le nombre de pertes en terme de paquets, en cas de congestion à travers plusieurs approches d'améliorations parmi lesquelles, nous pouvons citer celle qui vise à adopter un chemin alternatif vers la station de base à détection de congestion sur le chemin optimal, afin d'acheminer les données.

Les tests effectués sur le protocole AntNet et son amélioration sur des topologies filaires, ont montré à travers l'analyse comparative des résultats, que l'amélioration ap-

portée permet de minimiser le taux de perte de paquets quasiment de moitié, par rapport au protocole de base. Mais, nous avons remarqué que notre solution se base d'abord sur la détection de congestion, pour ensuite appliquer la solution. Elle minimise grandement le taux de perte mais permet quand même toujours la perte de paquets. Ce qui présente parfois un inconvénients en cas de réseaux de capteurs ou les pertes ne sont pas tolérées. Ce travail nous a quand même permis de nous initier à la recherche, en espérant avoir apporté une contribution aussi petite qu'elle soit, à ce domaine qui est en pleine évolution.

Comme perspectives, nous allons continuer à améliorer ce travail, tout d'abord par l'adaptation du protocole AntNet aux réseaux sans fils, car ces derniers sont la tendance actuelle et présente plusieurs avantages par rapport au réseaux filaires, notamment en termes d'installation et de maintenance, pour ensuite le comparer à d'autres protocoles. Tout cela est envisagé dans nos futurs travaux de recherche, qu'ils soient individuels, professionnels ou académiques.

Bibliographie

[1] Y. Younes. Minimisation d'énergie dans un réseau de capteurs. Master's thesis, Université Mouloud Mammeri, 2012.

[2] A. Bunel. Les réseaux de capteurs sans fil, `http://www-igm.univ-mlv.fr/~dr/XPOSE2006/Bunel/Presentation.html`, 2007.

[3] N. Boulahia and P. Owezarski. *3rd conference on Security in Network Architectures and Information Systems*, page 169. 2008.

[4] S. Akhenak and Reguia Zemouri. Une architecture prédictible distribuée pour la gestion de l'énergie dans un résaeu de capteurs sans fil. Master's thesis, Ecole Nationale Superieure d'informatique, 2013.

[5] M. Younis and K. Akkaya. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, Vol. 3(n. 3) :pp. 325–349, 2005.

[6] K. Khorchi and H.A. Tenboukti. Mécanisme de gestion de la confiance pour les réseaux de capteurs sans fil : application au modèle antmwac. Master's thesis, Ecole Nationale Supérieure d'Informatique, 2012.

[7] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks : A survey. *IEEE Communications*, Vol. 11(no.6) :pp. 6–28, 2004.

[8] R. Jurdak. *Wireless Ad Hoc and Sensor Networks :A Cross-Layer Design perspective*, pages 47–54. Springer Science and Business Media, 2007.

[9] S. Sentilles. Architecture logicielle pour capteurs sans-fil en réseau. Technical report, Université de Pau et des Pays de l'Adour, 2006.

[10] A. Milenković, C. Otto, and E. Jovanov. Wireless sensor networks for personal health monitoring : Issues and an implementation. *Elsevier*, 2006.

[11] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore. Environmental wireless sensor networks. *Proceedings Of The IEEE*, Vol. 98(no. 11) :pp. 1903–1917, 2010.

[12] R. Chakravarthi and C. Gomathy. Hop-by-hop rate control technique for congestion due to concurrent transmission in wireless sensor network. *World of Computer Science and Information Technology Journal*, Vol. 1(No. 8) :pp. 351–356, 2011.

[13] R.T. Malar. Congestion control in wireless sensor networks based multi-path routing in priority rate adjustment technique. *International Journal of Advanced Engineering and Applications*, pages 28–33, 2010.

[14] V. Vojayraja and R. Hemamalini. Congestion in wireless sensor networks and various techniques for mitigating congestion - a review. In *IEEE International Conference on Computational Intelligence and Computing Research*, 2010.

[15] C. Wang, K. Sohraby, B. Li, M. Daneshmand, and Y. Hu. A survey of transport protocols for wireless sensor networks. *IEEE Network*, Vol. 20(No. 3) :pp. 33–40, 2006.

[16] M.C. Vuran, V.C. Gungor, and O.B. Akan. On the interdependence of congestion and contention inwireless sensor networks. *Proceedings of SenMetrics*, pages 136–147, 2005.

[17] B. Sharma and T.C. Aseri. A comparative analysis of reliable and congestion-aware transport layer protocols for wireless sensor networks. *International Scholarly Research Network*, 2012.

[18] C. Gomathy, R. Chakravarthi, S.K. Sebastian, K. Pushparaj, and V.B. Mon. A survey on congestion control in wireless sensor networks. *International Journal of Computer Science and Communication*, Vol. 1(No. 1) :pp.161–164, 2010.

[19] A.J.D. Rathnayaka and V.M. Potdar. Wireless sensor network transport protocol : A critical review. *Journal of Network and Computer Applications*, Vol. 36(No. 1) :pp.134–146, 2013.

[31] M. Dorigo, G. D. Caro, and M. Gambaradella. Ant algorithms for discrete optimization. *MIT Press*, 2004.

[32] T. Camilo, C. Carreto, J. Silva, and F. Boavida. *Ant Colony Optimization and Swarm Intelligence*, pages 49–59. 2006.

[33] L. Juan, S. Chen, and Z. Chao. Ant system based anycast routing in WSN. pages 2420–2423. International Conference on Wireless Communications, Networking and Mobile Computing, 2007.

[34] Y. Chen, Y. Wen, and M. Pan. Adaptive ant-based routing in wireless sensor networks using energy delay metrics. *Journal of Zhejiang University Science A*, Vol. 9(n. 4) :pp. 531–538, 2008.

[35] A. Rahman, R. Ghasem, and M. B. Srivastava. Instrumentation and measurement technology conference proceedings. In *IMTC IEEE*, 2007.

[36] G. Di Caro and M. Dorigo. Antnet : A mobile agents approach to adaptive routing. Technical report, Université Libre de Bruxelles, 1997.

[37] B. Baran and R. Sosa. Antnet routing algorithm for data networks based on mobile agents. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, (n. 12) :pp. 75–84, 2001.

[38] N. Hamani, J. P. Jamont, M. Ocello, and M. Koudil. An optimized self organized approach to manage communication in wireless instrumentation systems. *IEEE Symposium*, pages 1–8, 2011.

[39] A. Chellama. Gestion de la mobilité au niveau de la couche transport. Master's thesis, Université Kasdi Merbah, 2009.

[40] V. Paxson and S. Floyd. Why we don't know how to simulate the internet. In *Proceedings of the Winter Simulation Conference*, 1997.

[41] J.K. Ousterhout. *Tcl and the Tk Toolkit.* READING : Addison-Wesley Publishing Company, Inc., 1994.

[20] A. Sridharan and B. Krishnamachari. Explicit and precise rate control for wireless sensor networks. *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 29–42, 2009.

[21] Y.C. Wan, S.B. Eisenman, and A.T. Campbell. CODA : Congestion detection and avoidance. *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003.

[22] Y. Sankarasubaranamiam, O.B. Akran, and I.F Akyildiz. ESRT : Event-to-sink reliable transport in wireless sensor networks. *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing*, pages 177–188, 2003.

[23] S. Misra, I. Woungang, and S.C. Misra, editors. *Guide to Wireless Sensor Networks*. Springer, 2009.

[24] Y. G. Iyer, S. Gandham, and S. Venkatesan. STCP : a generic transport layer protocol for wireless sensor networks. *Proceedings of the 14th International Conference on Computer Communications and Networks*, pages 449–454, 2005.

[25] K. Akkaya and M. Younis. Energy and qos aware routing in wireless sensor networks. *Springer Cluster Computing Journal*, Vol. 8 :pp. 179–188, 2005.

[26] B. Kadri, M. Feham, and A. Mhammed. Efficient and secured ant routing algorithm for wireless sensor networks. *International Journal of Network Security*, Vol. 16 :pp. 149–156, 2012.

[27] S.S. Dhillon and P. Van Mieghem. Performance analysis of the antnet algorithm. *Computer Networks*, 51 :pp. 2104–2125, 2005.

[28] J. Dréo and A. Pétrowski. *Métaheuristiques pour l'optimisation difficile*. Eyerolles, 2003.

[29] B. Sahraoui. Etude d'un protocole de routage basé sur les colonies de fourmis dans les réseaux de capteurs sans fil. Master's thesis, Université Abou Bakr Belkaid– Tlemcen, 2013.

[30] M. Dorigo, M. Birratari, and T. Stiitzle. *Ant Colony Optimization*. Tsinghua University Press, Beijing, 2007.